Lessons from the Sony CD DRM Episode

J. Alex Halderman and Edward W. Felten[*]
Center for Information Technology Policy
Department of Computer Science
Princeton University

Extended Version
February 14, 2006[†]

Abstract

In the fall of 2005, problems discovered in two Sony-BMG compact disc copy protection systems, XCP and MediaMax, triggered a public uproar that ultimately led to class-action litigation and the recall of millions of discs. We present an in-depth analysis of these technologies, including their design, implementation, and deployment. The systems are surprisingly complex and suffer from a diverse array of flaws that weaken their content protection and expose users to serious security and privacy risks. Their complexity, and their failure, makes them an interesting case study of digital rights management that carries valuable lessons for content companies, DRM vendors, policymakers, end users, and the security community.

1 Introduction

This paper is a case study of the design, implementation, and deployment of anti-copying technologies. We present a detailed technical analysis of the security and privacy implications of two systems, XCP and MediaMax, which were developed by separate companies (First4Internet and SunnComm, respectively) and shipped on millions of music compact discs by Sony-BMG, the world's second largest record company. We consider the design choices the companies faced, examine the choices they made, and weigh the consequences of those choices. The lessons that emerge are valuable not only for compact disc copy protection, but for copy protection systems in general.

Before describing the technology in detail, we will first recap the public events that brought the issue to prominence in the fall of 2005. This is necessarily a brief account that leaves out many details, some of which will appear later in the paper as we discuss each part of the technology. For a fuller account, see [10].

The security and privacy implications of Sony-BMG's CD digital rights management (DRM) technologies first reached the public eye on October 31, 2005, in a blog post by Mark Russinovich [29]. While testing a rootkit detector he had co-written, Russinovich was surprised to find an apparent rootkit (software designed to hide an intruder's presence [16]) on one of his systems. Investigating, he found that the rootkit was part of a CD DRM system called XCP that had been installed when he inserted a Sony-BMG music CD into his computer's CD drive.

[*]{jhalderm, felten}@cs.princeton.edu
[†]Revision 1, February 15, 2006

News of Russinovich's discovery circulated rapidly on the Internet, and further revelations soon followed, from us[1], from Russinovich, and from others. It was discovered that the XCP rootkit makes users' systems more vulnerable to attacks, that both CD DRM schemes install risky software components without obtaining informed consent from users, that both systems covertly transmit usage information back to the vendor or the music label, and that none of the protected discs include tools for uninstalling the software. (For these reasons, both XCP and MediaMax seem to meet the consensus definition of spyware.) These and other findings outraged many users.

As the story was picked up by the popular press and public pressure built, Sony-BMG agreed to recall XCP discs from stores and to issue uninstallers for both XCP and MediaMax, but we discovered that both uninstallers created serious security holes on users' systems. Class action lawsuits were filed soon after, and government investigations were launched, as Sony-BMG worked to repair relations with its customers.

While Sony-BMG and its DRM vendors were at the center of this incident, its implications go beyond Sony-BMG and beyond compact discs. Viewed in context, it is a case study in the deployment of DRM into a mature market for recorded media. Many of the lessons of CD DRM apply to other DRM markets as well.

Several themes emerge from this case study: similarities between DRM and malicious software such as spyware, the temptation of DRM vendors to adopt malware tactics, the tendency of DRM to erode privacy, the strategic use of access control to control markets, the failure of ad hoc designs, and the force of differing incentives in shaping behavior and causing conflict.

Outline The remainder of the paper is structured as follows. Section 2 discusses the business incentives of record labels and DRM vendors, which drive their technology decisions. Section 3 gives a high-level technical summary of the systems' design. Sections 4–9 each cover one aspect of the design in more detail, discussing the design choices made in XCP and MediaMax and considering alternative designs. We discuss weaknesses in the copy protection schemes themselves, as well as vulnerabilities they introduce in users' systems. We cover installation issues in Section 4, recognition of protected discs in Section 5, player software in Section 6, deactivation attacks in Section 7, uninstallation issues in Section 8, and compatibility and upgrading issues in Section 9. Section 10 concludes and draws lessons for other systems.

2 Goals and Incentives

The goals of a CD DRM system are purely economic: the system is designed to protect and enable the business models of the record label and the DRM vendor. Accordingly, any discussion of goals and incentives must begin and end by talking about business models. The record label and the DRM vendor are separate actors whose interests are not always aligned. We will see that incentive gaps between the label and the DRM vendor can be important in explaining the design and deployment of CD DRM systems.

2.1 Record Label Goals

We first examine the record label's goals. Though the label would like to keep the music from the CD from being made available on peer-to-peer (P2P) file sharing networks, this goal is not feasible [4]. If even one user can rip an unprotected copy of the music and put it on a P2P network, it will be available to the whole world. In practice, every commercially valuable song appears on P2P networks immediately upon release,

[1]As news of the rootkit spread, we added to the public discussion with a series of 27 blog posts analyzing XCP and MediaMax. This paper provides a more systematic analysis, along with much new information. Our original blog entries can be read at http://www.freedom-to-tinker.com/?cat=30&m=2005.

if not sooner. No CD DRM system can hope to stop this. Real systems do not appear designed to stop P2P sharing, but seem aimed at other goals.[2]

The record label's goal must therefore be to retard disc-to-disc copying and other local copying and use of the music. Stopping local copying might increase sales of the music—if Alice cannot copy a CD to give to Bob, Bob might buy the CD himself.

Control over local uses can translate into more revenue for the record label. For example, if the label can control Alice's ability to download music from a CD into her iPod, the label might be able to charge Alice an extra fee for iPod downloads. Charging for iPod downloads creates new revenue, but it also reduces the value to users of the original CD and therefore reduces revenue from CD sales. Whether the new revenue will outweigh the loss of CD revenue is a complex economic question whose answer depends on detailed assumptions about users' preferences; generally, increasing the label's control over uses of the music will tend to increase the label's profit.

Whether the label would find it more profitable to control a use, as opposed to granting it for free to CD purchasers, is a separate question from whether copyright law gives the label the right to file lawsuits relating to that use. Using DRM to enforce copyright law exactly as written is almost certainly not the record label's profit-maximizing strategy.

Besides controlling use of the music, CD DRM can make money for the record label because it puts software onto users' computers, and the label can monetize this installed platform. For example, each CD DRM album includes a special application for listening to the protected music. This application can show advertisements or create other promotional value for the label; or the platform can gather information about the user's activities, which can be exploited for some business purpose. If taken too far, these become spyware tactics; but they may be pursued more moderately, even over user objections, if the label believes the benefits outweigh the costs.

2.2 DRM Vendor Goals

The CD DRM vendor's primary goal is to create value for the record label in order to maximize the price the label will pay for the DRM technology. In this respect, the vendor's and label's incentives are aligned.

However, the vendor's incentives diverge from the label's in at least two ways. First, the vendor has a higher risk tolerance than the label, because the label is a large, established business with a valuable brand name, while the vendor (at least in the cases at issue here) is a start-up company with few assets and not much brand equity. Start-ups face many risks already and are therefore less averse to taking on one more risk. The record label, on the other hand, has much more capital and brand equity to lose if something goes horribly wrong. Accordingly, we can expect the vendor to be much more willing to accept security risks than the label.

The second incentive difference is that the vendor can monetize the installed platform in ways the record label cannot. For example, once the vendor's DRM software is installed on a user's system, the software can control use of other labels' CDs, so a larger installed base makes the vendor's technology more attractive to other labels. This extra incentive to build the installed base will make the vendor more aggressive about pushing the software onto users' computers than the label would be.

In short, incentive differences make the vendor more likely than the label to (a) cut corners and accept security and reliability risks, and (b) push DRM software onto more user's computers. If the label had perfect knowledge about the vendor's technology, this incentive gap would not be an issue—the label would simply insist that the vendor protect the label's interests. But if, as seems likely in practice, the label has imperfect knowledge of the technology, then the vendor will sometimes act against the label's interests.

[2]Music industry *rhetoric* about DRM often focuses on P2P, and some in the industry probably still think that DRM can stop P2P sharing. We believe that industry decision makers know otherwise. The design of the systems we studied supports this view.

2.3 DRM and Market Power

DRM affects more than just the relationships among the label, the vendor, and the user. It also impacts the label's and vendor's positions in their industries, in ways that will shape the companies' DRM strategies.

For example, DRM vendors are in a kind of standards war—a company that controls DRM standards has power to shape the online music business. DRM vendors fight this battle by spreading their platforms widely. Record labels want to play DRM vendors off against each other and prevent any one vendor from achieving dominance.

Major record companies such as Sony-BMG are parts of larger, diversified companies, and can be expected to help bolster the competitive position of their corporate siblings. For example, parts of Sony sell portable music players in competition with Apple, so Sony-BMG has an incentive to take steps to undermine Apple's market power.

Having examined the goals and motivations of the record labels and DRM vendors, we now turn to a description of the technologies they deployed.

3 CD DRM Systems

CD DRM systems must meet difficult requirements. Copy protected discs must be reasonably compliant with the CD Digital Audio standard so that they can play in ordinary CD players. They must be unreadable by almost all computer programs in order to prevent copying, yet the DRM vendor's own software must be able to read them in order to give the user controlled access to the music.

Most CD DRM systems use both passive and active anti-copying measures. Passive measures change the disc's contents in the hope of confusing most computer drives and software, without confusing most audio CD players. Active measures, in contrast, rely on software on the computer that actively intervenes to block access to the music by programs other than the DRM vendor's own software.

Active protection software must be installed on the computer somehow. XCP and MediaMax use Windows autorun, which (when enabled) automatically loads and runs software from a disc when the disc is inserted into the computer's drive. Autorun lets the DRM vendor's software run or install immediately.

Once the DRM software is installed, every time a new CD is inserted the software runs a recognition algorithm to determine whether the disc is associated with the DRM scheme. If the disc is associated, the active protection software will interfere with accesses to the disc, except those originating from the vendor's own music player application. This proprietary player application, which is shipped on the disc, gives the user controlled access to the music.

As we will discuss further, all parts of this design are subject to attack by a user who wants to copy the music illegally or who wants to make uses allowed by copyright law but blocked by the DRM. The user can defeat the passive protection, stop the DRM software from installing itself, trick the recognition algorithm, defeat the active protection software's blocking, capture the music from the DRM vendor's player, or uninstall the protection software.

The complexity of today's CD DRM software offers many avenues of attack. On the whole, today's systems are no more resistant to attack than were simpler early CD DRM systems. When there are fundamental limits to security, extra complexity does not mean extra security.

Discs Studied Sony deployed XCP on 52 titles (representing more than 4.7 million CDs) [1]. We examined three of them in detail: Acceptance, *Phantoms* (2005); Susie Suh, *Susie Suh* (2005); and Switchfoot, *Nothing is Sound* (2005). MediaMax was deployed on 37 Sony titles (over 20 million CDs) as well as dozens of titles from other labels [1]. We studied three albums that used MediaMax version 3—Velvet Revolver, *Contraband* (BMG, 2004); Dave Matthews Band, *Stand Up* (Sony, 2005); and Anthony Hamilton, *Comin'* *from Where I'm From* (Arista/Sony 2005)—and three albums that used MediaMax version 5—Peter Cetera, *You Just Gotta Love Christmas* (Viastar, 2004); Babyface, *Grown and Sexy* (Arista/Sony, 2005); and My Morning Jacket, *Z* (ATO/Sony, 2005). Unless otherwise noted, statements about MediaMax apply to both version 3 and version 5.

4 Installation

Active protection measures cannot begin to operate until the DRM software is installed on the user's system. In this section we consider attacks that either prevent installation of the DRM software, or capture music files from the disc in the interval after the disc has been inserted but before the DRM software is installed on the computer.

4.1 Autorun

Both XCP and MediaMax rely on the autorun feature of Windows. Whenever removable media, such as a floppy disc or CD, is inserted into a Windows PC (and autorun is enabled), Windows looks on the disc for a file called `autorun.inf`; if a file with that name is found, Windows executes commands found in it. Autorun is commonly used to pop up a splash screen or simple menu, for example, to offer to install software found on the disc. However, the autorun mechanism will run any program that the disc specifies.

Other popular operating systems, including MacOS X and Linux, do not have an autorun feature, so this mechanism does not work on these systems. XCP ships only Windows code and so has no effect on other operating systems. MediaMax ships with both Windows and MacOS code on the CD, but only the Windows code can autorun. The MacOS code relies on the user to double-click an installer on the CD, which few users will do. For this reason, we will not discuss the MacOS version of MediaMax further.

Current versions of Windows ship with autorun enabled by default, but the user can choose to disable it. Many security experts advise users to disable autorun to protect against disc-borne malware [6]. If autorun is disabled, the XCP or MediaMax active protection software will not load or run. Even if autorun is enabled, the user can block autorun for a particular disc by holding down the Shift key while inserting the disc [13]. This will prevent the active protection software from running.

Even without disabling autorun, a user can prevent the active protection software from loading by covering up the portion of the disc on which it is stored. Both XCP and MediaMax discs contain two sessions, with the first session containing the music files and the second session containing DRM content, including the active protection software and the autorun command file. The first session begins at the center of the disc and extends outward; the second session is near the outer edge of the disc. By covering the outer edge of the disc, the user can prevent the drive from reading the second session's files, effectively converting the disc back to an ordinary single-session audio CD. The edge of the disc can be covered with nontransparent material such as masking tape, or by writing over it with a felt-tip marker [27]. Exactly how much of the disc to cover can be determined by iteratively covering more and more until the disc's behavior changes, or by visually inspecting the disc to look for a difference in appearance of the disc's surface which is often visible at the boundary between the two sessions.

4.2 Temporary Protection

Even if the copy protection software is allowed to autorun, there is a period of time, between when a protected disc is inserted and when the active protection software is installed, when the music is vulnerable to copying. It would be possible to have the discs immediately and automatically install the active protection software, minimizing this window of vulnerability, but legal and ethical requirements should preclude this option. Installing software without first obtaining the user's consent appears to be illegal in the U.S. under the Computer Fraud and Abuse Act (CFAA) as well as various state anti-spyware laws [2, 3].

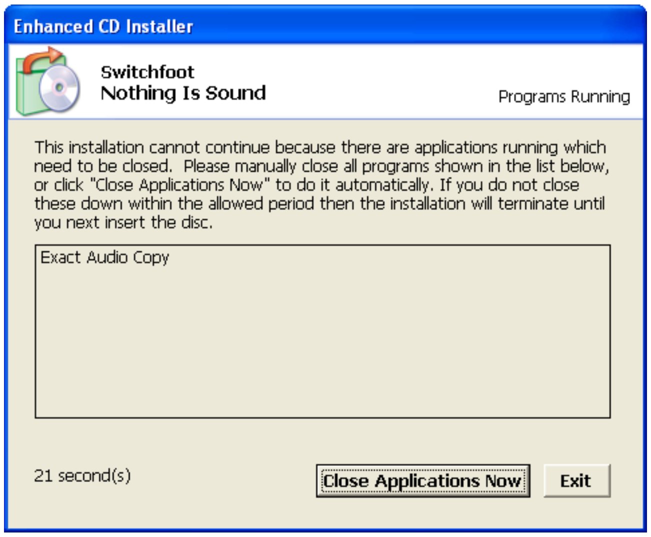
Software vendors conventionally obtain the user's consent to the installation of their software by displaying an End User License Agreement (EULA) and asking the user to accept it. Only after the user agrees to the EULA is the software installed. The EULA informs the user, in theory at least, of the general scope and purpose of the software being installed, and the user has the option to withhold consent by declining the EULA, in which case no software is installed. As we will see below, the DRM vendors do not always follow this procedure.

If the discs didn't use any other protection measures, the music would be vulnerable to copying while the installer waited for the user to accept or reject the EULA. Users could just ignore the installer's EULA window and switch tasks to a CD ripping or copying application. Both XCP and MediaMax employ temporary protection mechanisms to protect the music during this time.

4.2.1 XCP Temporary Protection

The first time an XCP-protected disc is inserted into a Windows machine, the Windows autorun feature launches the XCP installer, the file `go.exe` located in the `contents` folder on the CD. The installer displays a license agreement and prompts the user to accept or decline it. If the user accepts the agreement, the installer installs the XCP active protection software onto the machine; if the user declines, the installer exits after ejecting the CD, preventing other applications from ripping or copying it.

While the EULA is being displayed, the XCP installer continuously monitors the list of processes running on the system. It compares the image name of each process to a blacklist of nearly 200 ripping and copying applications hard coded into the `go.exe` program. If one or more blacklisted applications are running, the installer replaces the EULA display with a warning (shown at right) indicating that the applications need to be closed in order for the installation to continue. It also initiates a 30-second countdown timer; if any of the applications are still running when the countdown reaches zero, the installer ejects the CD and quits.[3]


Enhanced CD Installer

Switchfoot
Nothing Is Sound

Programs Running

This installation cannot continue because there are applications running which need to be closed. Please manually close all programs shown in the list below, or click "Close Applications Now" to do it automatically. If you do not close these down within the allowed period then the installation will terminate until you next insert the disc.

Exact Audio Copy

21 second(s)

This technique might prevent some unsophisticated users from copying the disc while the installer is running, but it can be bypassed with a number of widely known techniques. For instance, users might kill the installer process (using the Windows Task Manager) before it can eject the CD, or they might use a ripping or copying application that locks the CD tray, preventing the installer from ejecting the disc.

The greatest limitation of the XCP temporary protection system is the blacklist. Users might find ripping or copying applications that are not on the list, or they might use a blacklisted application but rename its executable file to prevent the installer from recognizing it. Since there is no mechanism for updating the blacklist on existing CDs, they will gradually become easier to rip and copy as new applications not on the

[3]Similar application blacklisting techniques have been used in other security contexts. The client software for World of Warcraft, a massively multiplayer online role playing game, checks running applications against a regularly updated blacklist of programs used to cheat in the game [15].

blacklist come into widespread use. Application developers may also adapt their software to the blacklisting technique by randomizing their process image names or taking other measures to avoid detection.[4]

4.2.2 MediaMax Temporary Protection

MediaMax employs a different—and highly controversial—temporary protection measure. It defends the music while the installer is running by installing, and at least temporarily activating, the active protection software *before* displaying the EULA. The software is installed without obtaining consent, and it remains installed (and in some cases, permanently active) even if the user explicitly denies consent by declining the license agreement.

MediaMax discs install the active protection driver by copying a file called `sbcphid.sys` to the Windows drivers directory, configuring it as a service in the registry, and launching it. Initially, the driver's startup type is set to "Manual," so it will not re-launch the next time the computer boots; however, it remains running until the computer is shut down, and it remains installed permanently [13]. Albums that use MediaMax version 5 additionally install components of the MediaMax player software before displaying a license agreement. These files are not removed if the EULA is declined.

Even more troublingly, under some common circumstances the MediaMax installer will permanently activate the active protection software (by setting its startup type to "Auto," which causes it to be launched every time the computer boots). This behavior is related to a mechanism in the installer apparently intended to upgrade the active protection software if an older version is already installed. Under the following scenarios, it is triggered even if the user previously declined the EULA:

- The user inserted a MediaMax version 3 album, then sometime later inserts an MediaMax version 5 album.
- The user inserted a version 5 album, then sometime later inserts a version 3 album.
- The user inserted an version 5 album, reboots, then sometime later inserts the same album or another version 5 album.

These steps do not have to take place in a single session. They can happen over a period of weeks or months, as users purchase new albums.

We can think of two possible explanations for this behavior. Perhaps the vendor, SunnComm, did not test these scenarios to determine what their software did, and so did not realize that they were activating the software without consent. Or perhaps they did know what would happen in these cases and deliberately chose these behaviors. Either possibility is troubling, indicating either a deficient design and testing procedure or a deliberate decision to install software after the user denied permission to do so.

Even if poor testing is the explanation for *activating* the software without consent, it is clear that SunnComm deliberately chose to *install* the MediaMax software on the user's system even if the user did not consent. These decisions are difficult to reconcile with the ethical and legal requirements on software companies. But they are easy to reconcile with the vendor's platform building strategy, which rewards the vendor for placing its software on as many computers as possible.

Even if no software is *installed* without consent, the temporary *activation* of DRM software, by both XCP and MediaMax, before the user consents to anything raises troubling ethical questions. It is hard to argue that the user has consented to loading running software merely by the act of inserting the disc. Most users do not expect the insertion of a music CD to load software, and although many (but not all) of the affected discs did contain a statement about protection software being on the discs, the statements generally

[4] An extreme extension of this would be to adopt rootkit-like techniques to conceal the copying application's presence, just as XCP hides its active protection software.

were confusingly worded, were written in tiny print, and did not say explicitly that software would install or run immediately upon insertion of the disc. Some in the record industry argue that the industry's desire to block potential infringement justifies the short-term execution of the temporary protection software on every user's computer. We think this issue deserves more ethical and legal debate.

4.3 Passive Protection

Another way to prevent copying before active protection software is installed is to use passive protection measures. Passive protection exploits subtle differences between the way computers read CDs and the way ordinary CD players do. By changing the layout of data on the CD, it is sometimes possible to confuse computers without affecting ordinary players. In practice, the distinction between computers and CD players is imprecise. Older generations of CD copy protection, which relied entirely on passive protection, proved easy to copy in some computers and impossible to play on some CD players [12]. Furthermore, computer hardware and software has tended to get better at reading the passive protected CDs over time as it became more robust to all manner of damaged or poorly formatted discs. For these reasons, more recent CD DRM schemes rely mainly on active protection.

XCP uses a mild variety of passive protection as an added layer of security against ripping and copying. This form of passive protection exploits a quirk in the way Windows handles multisession CDs. When CD burners came to market in the early 1990s, the multisession CD format was introduced to allow data to be appended to partially recorded discs. (This was especially desirable at a time when recordable CD media cost tens of dollars per disc.) Each time data is added to the disc, it is written as an independent series of tracks called a session. Multi-session compatible CD drives see all the sessions, but ordinary CD players, which generally do not support the multisession format, recognize only the first session.

Some commercial discs use a variant of the multisession format to combine CD audio and computer accessible files on a single CD. These discs adhere to the Blue Book or "stamped multisession" format. According to the Blue Book specification, stamped multisession discs must contain two sessions: a first session with 1–99 CD audio tracks, and a second session with one data track. The Windows CD audio driver contains special support for Blue Book discs. It presents the CD to playing and ripping applications as if it was a normal audio CD. Windows treats other multisession discs as data-only CDs.

XCP discs deviate from the Blue Book format by adding a second data track in the second session. This causes Windows to treat the disc as a regular multisession data CD, so the primary data track is mounted as a file system, but the audio tracks are invisible to player and ripper applications that use the Windows audio CD driver. This includes Windows Media Player, iTunes, and most other widely used ripper applications. We developed a procedure for creating discs with this flavor of passive protection using only standard CD burning hardware and software.

This variety of passive protection provides only limited resistance to ripping and copying. There are a number of well-known methods for defeating it:

- *Advanced ripping and copying applications* avoid the Windows CD audio driver altogether and issue commands directly to the drive. This allows programs such as Nero [24] and Exact Audio Copy [33] to recognize and read all the audio tracks.

- *Non-Windows platforms,* including MacOS and Linux, read multisession CDs more robustly and do not suffer from the limitation that causes ripping problems on Windows.

- The *felt-tip marker trick,* described above, can also defeat this kind of passive protection. When the second session is obscured by the marker, CD drives see only the first session and treat the disc as a regular audio CD, which can be ripped or copied with ease.

5 Disc Recognition

The active protection mechanisms employed by XCP and MediaMax regulate access to raw CD audio, blocking access to the audio tracks on albums protected with a particular scheme while allowing access to all other titles.

To accomplish this, the schemes install a background process that interposes itself between applications and the original CD driver. In MediaMax, this process is a kernel-mode driver called `sbcphid.sys`. XCP uses a pair of filter drivers called `crater.sys` and `cor.sys` that attach to the CD-ROM and IDE devices [29]. In both schemes, the active protection drivers examine each disc that is inserted into the computer to see whether access to it should be restricted. If the disc is recognized as copy protected, the drivers monitor for attempts to read the audio tracks, as would occur during a playback, rip, or disc copy operation, and corrupt the audio returned by the drive to degrade the listening experience. MediaMax introduces a large amount of random jitter, making the ripped audio sound like it has come from a badly scratched or damaged CD; XCP replaces the audio with random noise.

Each scheme's active protection software interferes with attempts to rip or copy any disc that is protected by the same scheme, not merely the disc from which the software was installed. This requires some mechanism for identifying discs that are to be protected. This section discusses the security requirements for such a recognition system, describes the design and limitations of the actual recognition mechanism employed by the MediaMax scheme, and presents an improved design that better satisfies the requirements.

5.1 Recognition Requirements

Any disc recognition system detects some distinctive feature on discs protected by a particular copy protection scheme. Ideally, such a feature would satisfy these requirements:

1. *Uniqueness.* The feature should identify protected discs without accidentally triggering the copy protection on unprotected titles.

2. *Detectability.* It should be possible for the active protection drivers running on client systems to reliably and quickly detect the feature in protected discs. In practice, this limits the amount of audio that can be read from the disc before deciding whether to apply protection.

3. *Indelibility.* The feature should be hard to remove without substantially degrading the quality of the audio; that is, it should be difficult to make a copy of the copy protected disc that does not itself trigger the protection.

4. *Unforgeability.* It should be difficult to apply the feature to an unprotected album without the cooperation of the protection vendor, even if the adversary has access to protected discs.

This last requirement stems from the DRM vendor's platform building strategy, which tries to put the DRM software on as many computers as possible and to have the software control access to all marked discs. If the vendor's identifying mark is forgeable, then a record label could mark its discs without the vendor's permission, thereby taking advantage of the vendor's platform without paying.[5]

There are advantages and disadvantages for an entity placing unauthorized marks. Copyright would prohibit rogue publishers from distributing an installer for the active protection software, though they might depend on the existing installed base if the software was included on many widely sold titles. They would

[5]Forging a mark is probably not copyright infringement. Unlike the musical work in which it is embedded, the mark itself is functional and contains little or no expression, and therefore seems unlikely to qualify for copyright protection. In principle, the mark recognition process could be covered by a patent, but we are unaware of any such patent relating to XCP or MediaMax. Even if the vendor does have a legal remedy, it seems worthwhile to design the mark to prevent forgery if the cost of doing so is low.

also be prevented from employing the components of the protection software that allow users to access restricted copies of the music; however, they could create their own software to provide this capability if they desired. On the other hand, free riding publishers would not be restricted to marking their disc for only one scheme. By identifying their discs as copy protected with multiple schemes (e.g., both XCP and MediaMax), they could invoke multiple layers of security and provide stronger protection than is available with any single technique, all without paying. Preventing free riding by publishers requires some kind of disc authentication mechanism to control access to installed active protection software—a meta-copy protection technique.

5.2 MediaMax Disc Recognition

To find out how well the disc recognition mechanisms employed by CD DRM systems meet the ideal requirements, we examined the recognition system built into MediaMax. This system drew our attention because MediaMax's creators have touted their advanced disc identification capabilities, including the ability to identify individual tracks within a compilation as protected [22]. (XCP appears to use a less sophisticated disc recognition system based on a marker stored in the data track of protected discs; we did not include it in this study.)

We determined how MediaMax identifies protected albums by tracing the commands sent to the CD drive with and without the active protection software running. These experiments took place on a Windows XP VMWare virtual machine running on top of a Fedora Linux host system, which we modified by patching the kernel IDE-SCSI driver to log all CD drive activity.

With this setup we observed that the MediaMax software executes a disc recognition procedure immediately upon the insertion of a CD. The MediaMax driver reads two sectors of audio at a specific offset from the beginning of audio tracks—approximately 365 and 366 frames in (a CD frame stores $\frac{1}{75}$ second of sound). On unprotected discs, the software scans through every track in this way, but on MediaMax-protected albums, it stops after the first three tracks, apparently having detected an identifying feature. The software decides whether or not to block read access to the audio solely on the basis of information in this region, so we inferred that the identifying mechanism takes the form of an inaudible watermark embedded in this part of the audio stream.[6]

Locating the watermark amid megabytes of audio might have been difficult, but we had the advantage of a virtual Rosetta Stone. The actual Rosetta Stone—a 1500 lb. granite slab, unearthed in Rosetta, Egypt, in 1799—is inscribed with the same text written in three languages: ancient hieroglyphics, demotic (simplified) hieroglyphics, and Greek. Comparing these inscriptions provided the key to deciphering Egyptian hieroglyphic texts. Our Rosetta Stone was a single album, Velvet Revolver's *Contraband*, released in three different versions: a U.S. release protected by MediaMax, a European release protected by a passive scheme developed by Macrovision, and a Japanese release with no copy protection. We decoded the MediaMax watermark by examining the differences between the audio on these three discs. Binary comparison revealed no differences between the releases from Europe and Japan; however, the MediaMax-protected U.S. release differed slightly from the other two in certain parts of the recording. By carefully analyzing these differences—and repeatedly attempting to create new watermarked discs using the MediaMax active protection software as an oracle—we were able to deduce the structure of the watermark.

The MediaMax watermark is embedded into the audio of each track in 30 *clusters* of modified audio samples. Each cluster is made up of 288 marked 16-bit audio samples followed by 104 unaltered samples. Three mark clusters exactly fit into one 2352-byte CD audio frame. The watermark is centered at approximately frame 365 of the track; though the detection routine in the software only reads two frames, the mark extends several frames to either side of the designated read target to allow for imprecise seeking in the audio

[6]By locating the watermark nearly five seconds after the start of the track rather than at the very beginning, MediaMax reduces the likelihood that it will occur in a very quiet passage (where it might be more audible) and makes cropping it out more destructive.

portion of the disc (a typical shortcoming of inexpensive CD drives). The MediaMax driver detects the watermark if at least one mark cluster is present in the region read by the detector.

A sequence of 288 bits that we call the *raw watermark* is embedded into the 288 marked audio samples of each mark cluster. A single bit of the raw watermark is embedded into an unmarked audio sample by setting one of the three least significant bits to the new bit value (as shown in bold below) and then setting the two other bits according to this table:[7]

Original bits	Marked bits					
	0__	_**0**_	__**0**	**1**__	_**1**_	__**1**
_____________111	**0**11	1**0**1	11**0**	**1**11	1**1**1	11**1**
_____________110	**0**11	1**0**1	11**0**	**1**10	1**1**0	11**1**
_____________101	**0**11	1**0**1	10**0**	**1**01	1**1**0	10**1**
_____________100	**0**11	1**0**0	10**0**	**1**00	1**1**0	10**1**
_____________011	**0**11	0**0**1	01**0**	**1**00	0**1**1	01**1**
_____________010	**0**10	0**0**1	01**0**	**1**00	0**1**0	01**1**
_____________001	**0**01	0**0**1	00**0**	**1**00	0**1**0	00**1**
_____________000	**0**00	0**0**0	00**0**	**1**00	0**1**0	00**1**

The position of the embedded bit in each sample follows a fixed sequence for every mark cluster. Each of the 288 bits is embedded in the first-, second-, or third-least-significant bit position of the sample according to this sequence:

```
2,3,1,1,2,2,3,3,2,3,3,3,1,3,2,3,2,1,3,2,2,3,2,2,2,1,3,3,2,1,2,3,3,1,2,2,3,1,2,3,3,1,1,2,2,1,1,3,
3,1,2,3,1,2,3,3,1,3,3,2,1,1,2,3,2,2,3,3,3,1,1,3,1,2,1,2,3,3,2,2,3,2,1,2,2,1,3,1,3,2,1,1,2,1,1,1,
2,3,2,1,1,2,3,2,1,3,2,2,2,3,1,2,1,3,3,3,3,1,1,1,2,1,1,2,2,2,2,3,1,2,3,2,1,3,1,2,2,3,1,1,3,1,1,1,
1,2,2,3,2,3,2,3,2,1,2,3,1,3,1,3,3,3,1,1,2,1,1,2,1,3,3,2,3,3,2,2,1,1,1,2,2,1,3,3,3,3,3,1,3,1,1,3,
2,2,3,1,2,1,2,3,3,2,1,1,3,2,1,1,2,2,1,3,3,2,2,3,1,3,2,2,2,3,1,1,1,1,3,2,1,3,1,1,2,2,3,2,3,1,1,2,
1,3,2,3,3,1,1,3,2,1,3,1,2,2,3,1,1,3,2,1,2,2,2,1,3,3,1,2,3,3,3,1,2,2,3,1,2,3,1,1,3,2,2,1,3,2,1,3
```

The active protection software reads the raw watermark by reading the first, second, or third bit from each sample according to the sequence above. It determines whether the resulting 288-bit sequence is a valid watermark by checking certain properties of the sequence (represented below). It requires 96 positions in the sequence to have a fixed value, either 0 or 1. Another 192 positions are divided into 32 groups of linked values (denoted a–z and α–ζ below). In each group, three positions share the same value and three share the complement value. This allows the scheme to encode a 32-bit value (value A), though in the discs we studied it appears to take a different random value in each mark cluster of each protected title. The final 32 bits of the raw watermark may have arbitrary values (denoted by _ below) and encode a second 32-bit value (value B). MediaMax version 5 uses this value to distinguish between original discs and backup copies burned through it proprietary player application.

$$
\begin{gathered}
0, a, b, c, d, e, 0, 0, f, 0, g, 0, h, 0, i, d, j, \bar{j}, k, 0, l, m, 0, n, o, p, \bar{e}, q, \bar{e}, r, 0, \bar{p}, s, d, \bar{m}, t, u, v, w, t, \bar{l}, a, x, c, u, 0, \bar{r}, l, \\
f, \bar{d}, v, 0, m, 0, \bar{q}, 0, y, c, z, 0, j, \bar{i}, \bar{g}, \alpha, \bar{s}, \bar{w}, \bar{h}, v, y, n, 0, 0, \bar{h}, \bar{j}, \bar{u}, a, \beta, 0, \bar{v}, g, j, 0, 0, \bar{\beta}, \bar{i}, e, \bar{z}, 0, r, \gamma, \bar{a}, \delta, \bar{d}, \bar{z}, 0, \bar{v}, \\
\epsilon, 0, x, s, \bar{g}, \bar{r}, 0, \bar{b}, o, b, r, 0, y, \bar{\beta}, \bar{m}, h, 0, \bar{a}, n, \bar{f}, \bar{t}, 0, \bar{o}, 0, \bar{\gamma}, \bar{\epsilon}, \bar{e}, 0, 0, \bar{k}, \bar{c}, \bar{x}, 0, \bar{f}, p, z, \bar{x}, i, 0, 0, \alpha, \bar{g}, 0, 1, w, \bar{t}, \bar{n}, \bar{w}, \\
i, 0, 0, \bar{j}, m, x, \beta, \bar{y}, \bar{p}, \bar{q}, 0, 0, 0, e, \bar{\beta}, 0, 0, 1, g, 0, p, l, 0, \bar{\alpha}, t, h, \bar{d}, \bar{\epsilon}, \bar{w}, \gamma, \bar{\delta}, 0, \bar{p}, q, \bar{f}, 0, 1, \zeta, 0, \bar{c}, \zeta, \bar{\alpha}, \bar{s}, \bar{b}, \bar{\gamma}, \beta, 0, o, \\
0, q, \bar{i}, 0, 0, \bar{\alpha}, s, \epsilon, \bar{\epsilon}, \bar{h}, 0, \bar{k}, \bar{n}, \bar{\zeta}, \alpha, \bar{s}, \bar{z}, \bar{n}, \bar{c}, \bar{o}, \bar{b}, 0, \bar{t}, 0, \bar{y}, \bar{v}, 0, \zeta, \bar{o}, 0, \bar{\zeta}, 0, u, \gamma, 0, \bar{y}, k, \bar{u}, z, \bar{\delta}, \bar{q}, k, \bar{r}, \bar{u}, \bar{\zeta}, \bar{\gamma}, \bar{l}, \bar{l}, \\
w, \bar{k}, \bar{a}, 0, \bar{\delta}, 0, \epsilon, \bar{m}, b, f, 0, 0, \bar{x}, \delta, \delta, 0, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_, \_
\end{gathered}
$$

[7] This design seems to be intended to lessen the audible distortion caused by setting one of the bits to the watermark value. The change in the other two bits reduces the magnitude of the difference from the original audio sample, but it also introduces a highly uneven distribution in the three least significant bits that makes the watermark easier to detect or remove.

5.3 Attacks on the MediaMax Watermark

The MediaMax watermark fails to satisfy the indelibility and unforgeability requirements of an ideal disc recognition system. Far from being indelible, the mark is surprisingly brittle. Most advanced designs for robust audio watermarks [8, 7] manipulate the audio in the frequency domain and attempt to resist removal attempts that use lossy compression, multiple conversions between digital and analog formats, and other common transformations. In contrast, the MediaMax watermark is applied in the time domain and is rendered undetectable by even minor changes to the file. An adversary without any knowledge of the watermark's design could remove it by converting the tracks to a lossy format like MP3 and then burning them back to a CD, which can be accomplished easily with standard consumer applications. This would result in some minor loss of fidelity, but a more sophisticated adversary could prevent the mark from being detected with almost no degradation by flipping the least significant bit of one carefully chosen sample from each of the 30 watermark clusters, thereby preventing the mark from exhibiting the pattern required by the detector.

The watermark also fails to satisfy the unforgeability requirement. The mark's only defense against forgery is its complicated, unpublished design, but as is often the case this security by obscurity has proved tedious rather than impossible to defeat. As it turns out, an adversary needs only limited knowledge of the watermark—its location within a protected track and its confinement to the three least significant bits of each sample—to forge it with minimal loss of fidelity. Such an attacker could transplant the three least significant bits of each sample within the watermarked region of a protected track to the corresponding sample from an unprotected one. Transplanting these bits would cause distortion more audible that that caused by embedding the watermark since the copied bits are likely to differ by a greater amount from the original sample values; however, the damage to the audio quality would be limited since the marked region is only 0.4 seconds in duration. A more sophisticated adversary could apply a watermark to an unprotected track by deducing the full details of the structure of the watermark, as we did; she could then embed the mark in an arbitrary audio file just as well a licensed disc producer.

As a proof-of-concept, we created a utility called `scmark` that can detect, embed, or remove the MediaMax watermark. The program is invoked on one or more WAVE audio files as follows:

```
usage: scmark --detect [-p <pos>] [-c <count>] <file.wav> [files]
              --embed [[a] <b>] [-p <pos>] [-c <count>] <file.wav> [files]
              --remove [-p <pos>] [-c <count>] <file.wav> [files]
```

When `scmark` is executed with only the `--detect` parameter on a track ripped from a MediaMax-protected album, the watermark is detected almost instantaneously. The `-p <pos>` parameter gives a hint as to the position of the watermark within the file; and `-c <count>` indicates how many mark clusters to read or write. For embedding, the `a` and `b` parameters specify the two 32-bit values A and B encoded in the watermark, as described in Section 5.2. The `--remove` switch searches for an existing watermark and removes it by overwriting it with a random raw watermark that lacks the properties required by the MediaMax detector.

5.4 Secure Disc Recognition

Having shown that the MediaMax watermark fails to provide either strong resistance to removal or strong resistance to forgery, we ask whether it is possible to securely accomplish either or both of these goals.

As far as indelibility is concerned, watermarking schemes have a poor history of resisting removal [8, 20, 26]. This is especially true against an adversary who has oracle access to the watermark detector, as was the case with a previous application of watermarks to audio copy protection, SDMI [8], and with CD DRM systems. Making marks that are both indelible and unforgeable is likely much more difficult. There seems to be tension between marks that are difficult to remove and ones that are hard to forge. Enforcing both requirements creates two ways to fool the detector—by rendering the mark invisible and by making it appear forged. If, as in CD DRM systems, either situation leads to the same result (no protection), the attacker's power is magnified.

In contrast, a mark strongly robust to forgery is simple to create based on digital signatures if we aren't concerned with its being easy to remove. A very simple scheme works as follows:

1. To sign an audio track, the licensed publisher reads a fixed portion L_1 of the audio data (say, the first ten seconds), then computes a cryptographic hash of L_1 and signs it using a public key signature algorithm to derive the signature $S_{L_1} := Sign_{K_S}(Hash(L_1))$. S_{L_1} is then stored at a second location in the track by setting the LSB of each sample in the region to the corresponding bit in the signature. A 320-bit DSA signature could be embedded in this way using approximately the same space as one mark cluster of the MediaMax watermark.

2. The publisher keeps the signing key K_S secret, and builds the corresponding verification key K_V into the client software. When presented with a CD, the software checks for a valid signature. First it reads the audio from the signed area of the track and hashes it, and it locates and extracts the signature stored in the LSBs in the second mark location. Next, it verifies the signature on the hash using K_V. If the signature is correct, the watermark is valid and genuine; otherwise, forgery or data corruption is indicated.

The scheme could be strengthened against natural errors by applying the mark to several regions of the disc, as is the MediaMax watermark.

Forging such a mark would require defeating the digital signature scheme or splicing both L_1 and S_{L_1} from a legitimately marked album. We set L_1 to be several seconds of audio to make such splicing less appealing.

Clearly this watermark is highly vulnerable to removal. If even a single bit of the hashed region is changed, the mark will not be recognized as valid. Yet the watermark MediaMax actually uses is also vulnerable to corruption by a single bit (in each mark cluster) while being far less resistant to forgery. Though robustness to removal could be improved by using error correction methods, we believe that robustness, while desirable in principle, is of limited value in real CD DRM applications, and should not be traded off against forgeability. Removal of the watermark is unlikely to be the weakest link protecting the audio, and while the gains from creating a more indelible watermark are slight, the loss to free riders from an easily forgeable mark is potentially much greater.

6 CD DRM Players

Increasingly, personal computers—and portable playback devices that attach to them—are users' primary means of organizing, transporting, and enjoying their music collections. Sony-BMG and its DRM vendors recognized this trend when they designed their copy protection technologies. Rather than inhibit all use with PCs, as some earlier anti-copying schemes did [12], XCP and MediaMax provide their own proprietary media players, shipped on each protected CD, that allow certain limited uses subject to restrictions imposed by DRM software.[8]

The XCP and MediaMax players launch automatically using autorun when a protected disc is inserted into a PC. Both players have similar feature sets. They provide a rudimentary playback interface, allowing

[8]The restrictions imposed by the DRM players only loosely track the contours of copyright law. Some uses that could be prohibited under copyright—such as burning three copies to give to friends—are allowed by the software, while some perfectly legal uses—like transferring the music to one's iPod—are prevented.

users to listen to protected albums, and they allow access to “bonus content,” such as album art, liner notes, song lyrics, and links to artist web sites. The players access music on the disc, despite the active protection, by using a special back door interface provided by the active protection software.

XCP and MediaMax version 5 both permit users to burn copies of the entire album a limited number of times (typically three). These copies are created using a proprietary burning application integrated into the player. The copies include the player applications and the same active (and passive, for XCP) protection as the original album, but they do not allow any subsequent generations of copying.

Another feature of the player applications allows users to rip the tracks from the CD to their hard disks, but only in DRM-protected audio formats. Both schemes support the Windows Media Audio format by using a Microsoft product, the Windows Media Data Session Toolkit [23], to deliver DRM licenses that are bound to the PC where the files were ripped. The licenses allow the music to be transferred to portable devices that support Windows Media DRM or burned onto CDs, but the Windows Media files will not be usable if they are copied to another PC. Because XCP and MediaMax create Windows Media files, they are vulnerable to any attack that can defeat Windows Media DRM. Often, DRM interoperation allows attacks on one system to defeat other systems as well, because the attacker can transfer protected content into the system of her choice in order to extract it.

The XCP and MediaMax version 5 players both exhibit similar spyware-like behavior: phoning home to the vendor or record label with information about users’ listening habits despite statements to the contrary from the vendors. Whenever a protected disc is inserted, the players contact web servers to retrieve images or banner ads to display. Part of the request is a code that identifies the album. XCP discs contact a Sony web site, `connected.sonymusic.com` [28]; MediaMax albums contact `license.sunncomm2.com`, a site operated by MediaMax’s creator, SunnComm. These connections allow the servers to log the user’s IP address, the date and time, and the identity of the album. This undisclosed data collection, in combination with other practices—installation without informed consent and the lack of an uninstaller—make XCP and MediaMax fit the consensus definition of spyware.

6.1 Attacks on Players

The XCP and MediaMax version 5 players were designed to enforce usage restrictions specified by content providers. In practice, they provide minimal security because there are many ways that users can bypass the limitations. Perhaps the most interesting class of attacks targets the limited number of burned copies permitted by the players. Both players are designed to enforce this limit without communicating with any networked server; thus, the player must keep track of how many allowed copies remain by storing state on the local machine.

It is well known that DRM systems like this are vulnerable to rollback attacks. A rollback attack backs up the state of the machine before performing the limited operation (in this case, burning the copy). When the operation is complete, the old system state is restored, and the DRM software is not able to determine that the operation has occurred. This kind of attack is easy to perform with virtual machine software like VMWare, which allows the entire state of the system to be saved or restored in a few clicks. XCP and MediaMax both fail under this attack, which allows unlimited copies to be burned with their players.

A refined variation of this attack targets only the specific pieces of state that the DRM system uses to remember the number of copies remaining. The XCP player uses a single file, `%windir%\system32\$sys$filesystem\$sys$parking`, to record how many copies remain for every XCP album that has been used on the system.[9] Rolling back this file after a disc copy operation would restore the original number of copies remaining.

[9]This file is hidden and protected by the XCP rootkit. Before the user can access the file, the rootkit must be disabled, as described in Section 7.2. We did not determine how the MediaMax player stores the number of copies remaining.

A more advanced attacker can go further and modify the `$sys$parking` file to set the counter to an arbitrary value. The file consists of a 16 byte header followed by a series of 177 byte structures. For each XCP disc used on the machine, the file contains a whole-disc structure and an individual structure for each track. Each disc structure stores the number of permitted copies remaining for the disc as a 32-bit integer beginning 100 bytes from the start of the structure.

The file is protected by primitive encryption. Each structure is XORed with a repeating 256-bit pad. The pad—a single pad is used for all structures—is randomly chosen when XCP is first installed and stored in the system registry in the key `HKLM\SOFTWARE\$sys$reference\ClassID`. Note that this key, which is hidden by the rootkit, is intentionally misnamed "ClassID" to confuse investigators. Instead of a ClassID, it contains the 32 bytes of pad data.

Hiding the pad actually doesn't increase the security of the design. An attacker who knows only the format of the `$sys$parking` file and the current number of copies remaining can change the counter to an arbitrary value without needing to know the pad. Say the counter indicates that there are x copies remaining and the attacker wants to set it to y copies remaining. Without decrypting the structure, she can XOR the padded bytes where the counter is stored with the value $x \oplus y$. If the original value was padded with p, the new value is $(x \oplus p) \oplus (x \oplus y) = (y \oplus p)$, y padded with p.

6.1.1 iPod Compatibility

Ironically, Sony itself furnishes directions for carrying out another kind of attack on the player DRM. Conspicuously absent from the XCP and MediaMax players is support for the Apple iPod—by far the most popular portable music player. A Sony FAQ blames Apple for this shortcoming and urges users to direct complaints to them: "Unfortunately, in order to directly and smoothly rip content into iTunes it [sic.] requires the assistance of Apple. To date, Apple has not been willing to cooperate with our protection vendors to make ripping to iTunes and to the iPod a simple experience." [32]. Strictly speaking, it is untrue that Sony requires Apple's cooperation to work with the iPod, as the iPod can import MP3s and other open formats. What Sony has difficulty doing is moving music to the iPod while keeping it wrapped in copy protection. This is because Apple has so far refused to support interoperation with its FairPlay DRM.

Yet so great is consumer demand for iPod compatibility that Sony gives out—to any customer who fills out a form on its web site [31]—instructions for working around its own copy protection and transforming the music into a DRM-free format that will work with the iPod. The procedure is simple but cumbersome: users are directed to use the player software to rip the songs into Windows Media DRM files; use Windows Media Player to burn the files to a blank CD, which will be free of copy protection; and then use iTunes to rip the songs once more and transfer them to the iPod.

6.2 XCP's Hidden iPod Support

A further irony came to light in the weeks following the public disclosure of the XCP rootkit when it was discovered that XCP itself infringes on the copyrights to several open source software projects. In one case, Sam Hocevar found conclusive evidence [14] that part of XCP's code was copied from a program called DRMS [18], which he co-authored with Jon Lech Johansen and released under the terms of the GNU General Public License (GPL). This was particularly curious, because the purpose of DRMS is to break Apple's FairPlay DRM. Its presence in XCP is interesting enough to warrant a digression from our discussion of player-related attacks.

We discovered that XCP utilizes the DRMS code not to remove Apple DRM but to add it, as part of a hidden XCP feature that provides iTunes and iPod compatibility. This functionality shipped on nearly every XCP CD, but it was never enabled or made visible in the XCP user interface. Despite being inactive, the code appears to be fully functional and was compatible with the current version of iTunes when the first

XCP CDs were released.[10] This strongly suggests that the infringing DRMS code was deliberately copied by XCP's creator, First4Internet, rather than accidentally included as part of a more general purpose media library used for other functions in the copy protection system.

This isn't the first time another vendor has tried to make its DRM compatible with FairPlay. FairPlay is the only DRM compatible with the iPod, and Apple has declined to license it to rival music distributors, effectively locking rivals out from the iPod platform (at least as long as the rivals insist on using DRM). In 2004, RealNetworks attempted to work around Apple and reverse engineered FairPlay so that Real Player could create FairPlay files for use with the iPod [9]. Apple responded by making vague legal threats and updating iTunes to break this compatibility. The evidence suggests that First4Internet wanted to create their own iPod compatibility system, but rather than take the time to reverse engineer FairPlay themselves, they copied critical pieces of code from DRMS in violation of the GPL license.

Understanding how XCP uses code from DRMS requires some background information about FairPlay. When a customer purchases a song from the iTunes Music Store, she receives a FairPlay encrypted audio file that can only be played with knowledge of a secret key assigned to her by Apple. iTunes retrieves this key from an Apple server and stores it on the hard drive in an encrypted key database (a file called `SC Info.sidb`). When the user plays the song again, or if she copies it to an iPod, iTunes reads her key from the database instead of reconnecting to the server [34].

FairPlay's security depends on the encrypted key database being difficult for anyone but Apple to decipher, so it is protected using a proprietary encryption method and a system-dependent secret key.[11] iTunes encrypts the key database using a two step process. First it pads the plaintext database by XORing it with the output of a proprietary pseudorandom number generator (PRNG) using a system-dependent seed; then it applies AES encryption in ECB mode with a system-dependent key. As a consequence of this design, the code for the PRNG is exactly the same whether the file is being encrypted or decrypted. To decrypt, iTunes applies AES decryption, then XORs the same PRNG output again. This explains why parts of the DRMS code—in particular, a function called `DoShuffle`, which computes the PRNG's output—are useful for encryption as well as their original purpose, decryption.

The proprietary PRNG must have been especially difficult to reverse engineer. Rather than expend this effort themselves, XCP's authors appear to have lifted the DoShuffle code verbatim from DRMS. XCP uses this code to manipulate the iTunes key database in the process of adding FairPlay protection. Starting with an unencrypted audio file, such as a track from a protected CD, XCP applies AAC compression in memory, then encrypts using the same algorithm as FairPlay. Instead of using an Apple-assigned user key, XCP creates a new random user key and, with the help of the DRMS code, adds it to the iTunes key database. This ensures that the song file can only be used on the computer where it was created.

The XCP FairPlay compatibility code is contained in a file named `ECDPlayerControl.ocx` that is installed the first time an XCP CD is played. The code can be tested by jumping to a function at debugger offset 0x10010380 (apparently the start of a thread for transferring music to iTunes). This function takes one parameter, a wide character string of the form `<MP3><"C:\test.mp3">`. This syntax causes the function to convert an MP3 file to a FairPlay-protected AAC file. Variations can be used to specify other audio sources: WAV files, raw audio files, standard unprotected audio CDs, and XCP copy-protected CDs. Before calling the function, it is necessary to initialize a Windows `CriticalSection` object and set the `ECX` register to the object's address minus 0x6C.

The entry function calls a subroutine (offset 0x10027D20) that converts an audio file into a FairPlay-protected AAC file. This call another subroutine (offset 0x1008A470) that reads the iTunes key database,

[10] XCP's FairPlay-compatibility code works with iTunes up to iTunes version 4.8. iTunes 4.9, released June 28, 2005, includes changes unrelated to FairPlay that cause the XCP code to fail. XCP CDs released after this date do not appear to contain an updated version of the code.

[11] As security experts predicted, this protection was quickly broken. Today DRMS is able to defeat FairPlay because Jon Lech Johansen reverse engineered the database decryption code in iTunes.

decrypts it, and, if necessary, adds the XCP user key to the database and re-saves it in encrypted form. The iTunes database encryption function (0x1008A0C0) and decryption function (0x1008A300) both make use of the `DoShuffle` routine (0x10089E00) taken from DRMS.

6.3 Attacks on the Backdoor

We note that the DRM vendor must include some kind of limited back door so that the vendor's own CD-reading software can reliably access the music on the disc. The active protection software will offer some kind of interface that can be called to get access to the raw data from the disc, or to deactivate (temporarily) the active protection. This back door interface must be protected so that an ordinary program cannot use the back door to access the music. There are three ways to protect the back door.

1. *Secret interface*. The back door interface can be kept secret so that ordinary programmers do not know how to call it. This method is disfavored because its security by obscurity method violates Kerckhoffs's Principle [19].

2. *Secret key passed as argument*. There can be a secret key that must be passed to the back door interface, and the active protection software can be programmed to ignore requests that do not contain the correct key. This is superior to the secret interface method because it relies on a randomly generated key rather than secrecy of algorithm information. But an adversary who can interpose himself into the back door interface can observe the key, and can act as a man in the middle to modify the calls being made to the active protection software.

3. *Cryptographic protocol*. The vendor's application software can use a cryptographic protocol to communicate with the active protection software. The sort of protocol used for secure remote procedure call on a network would be suitable, with the network messages replaced by calls across the back door interface, so that each back-and-forth pair of messages in the protocol was replaced by a single call and return. This approach makes interface snooping and interposition attacks useless (assuming the protocol is properly secure). However, it cannot stop an adversary from reverse engineering the vendor's application-level software to learn the protocol and extract any keys.

6.4 MediaMax Player Security Risks

Besides suffering from several kinds of attacks that expose the music content to copying, the MediaMax version 5 player makes the user's system more vulnerable to attack. When a MediaMax CD is inserted into a computer, Windows autorun launches an installer from the disc. Even before displaying a license agreement, MediaMax copies almost twelve megabytes of files and data related to the MediaMax player to the hard disk and stores them in a folder named `%programfiles%\Common Files\SunnComm Shared`. Jesse Burns and Alex Stamos of iSec Partners discovered that the MediaMax installer sets file permissions that allow any user to modify its code directory and the files and programs in it [5].

As Burns and Stamos realized, the lax permissions allow a non-privileged user to replace the executable code in the MediaMax player files with malicious code. The next time a user plays a MediaMax-protected CD, the attack code will be executed with that user's security privileges. The MediaMax player requires Power User or Administrator privileges to run, so it's likely that the attacker's code will run with almost complete control of the system.

Normally, this problem could be fixed by manually correcting the errant permissions. However, MediaMax aggressively updates the installed player code each time the software on a protected disc autoruns or is launched manually. As part of this update, the permissions on the installation directory are reset to the insecure state.

We discovered a variation of the attack suggested by Burns and Stamos that allows the attack code to be installed even if the user has never consented to the installation of MediaMax, and to be triggered immediately whenever the user inserts a MediaMax CD. In the original attack, the user needs to accept the MediaMax license agreement before attack code can be inserted or executed, because the code is placed in a file called `MMX.EXE` that is not copied to the system until after the agreement is accepted. In our attack, the attacker places hostile code in the `DllMain` procedure of a code file called `MediaMax.dll`, which MediaMax installs even before displaying the EULA. The next time a MediaMax CD is inserted, the installer autoruns and immediately attempts to check the version of the installed `MediaMax.dll` file. To do this, the installer calls the Windows `LoadLibrary` function on the DLL file, which causes the file's `DllMain` procedure to execute, together with any attack code inserted there.

This problem is exacerbated because parts of the MediaMax software are installed automatically and without consent. Users who declined the EULA would likely assume that MediaMax was not installed, and so most would be unaware that they were vulnerable. The same installer code performs the dangerous version check as soon as the CD is inserted. A CD that prompted the user to accept a license before installing code would give the user a chance to head off the attack.

Fixing this problem permanently without losing the use of protected discs requires installing a patch from MediaMax. Unfortunately, as we discovered, the initial patch released by Sony-BMG in response to the iSec report was capable of triggering precisely the kind of attack it was supposed to prevent. In the process of updating MediaMax, the patch checked the version of `MediaMax.dll` just like the MediaMax installer does. If this file was already modified by an attacker, the process of applying the security patch would execute the attack code. Prior versions of the MediaMax uninstaller had the same vulnerability, though both the uninstaller and the patch have since been replaced with versions that do not suffer from this problem.

7 Deactivation

Active protection methods install and run software components that interfere with accesses to a CD. Users can remove or deactivate the active protection software by using standard system administration tools that are designed to find, characterize, and control the programs installed on a machine. Deactivating the protection will enable arbitrary use or ripping of the music, and it is difficult to stop if the user has system administrator privileges. In this section, we discuss how active protection may be deactivated.

7.1 Deactivating MediaMax

The MediaMax active protection software is simple to deactivate since it is a single device driver with a consistent service name, `sbcphid`. The service can be manipulated using the Windows XP command line utility `sc`. To check the status of the service, a user can open a command prompt windows and issue the command `sc query sbcphid`; if the reported state is "RUNNING" then the MediaMax driver is active. It can be deactivated using the command `sc stop sbcphid`. To permanently remove it, a user can issue the command `sc delete sbcphid`, then delete `%windir%\system32\drivers\sbcphid.sys`, the driver's program file. Once the driver is deactivated, MediaMax-protected albums can be accessed as if they were unprotected.

7.2 Defenses Against Deactivation

To counter deactivation attempts, a vendor might try technical tricks to evade detection and frustrate removal of the active protection software. An example is the rootkit-like behavior of XCP, discovered by Mark Russinovich [29]. When XCP installs its active protection software, it also installs a second program—the rootkit—that conceals any file, process, or registry key whose name begins with the prefix `$sys$`. The result is that XCP's main installation directory, and most of its registry keys, files, and processes, become invisible to normal programs and administration tools.

The rootkit is a kernel-level driver named `$sys$aries` that is set to automatically load early in the boot process. When the rootkit starts, it hooks several Windows system calls by modifying the system service dispatch table (the kernel's `KeServiceDescriptorTable` structure) which is an array of pointers to the kernel functions that implement basic system calls. The rootkit changes five of these addresses to point to functions within the rootkit, so that calls to the patched system calls are handled by the rootkit rather than the original kernel function. The rootkit calls the real kernel function with the same parameters and filters the results before returning them to the application.

The system calls intercepted by the rootkit are:

1. `NtQueryDirectoryFile` – This function is used to list the contents of a directory; the rootkit version filters out directory entries that begin with `$sys$`, rendering such files and directories invisible to applications.

2. `NtCreateFile` – This call is used for creating and opening files. The rootkit version returns an invalid filename error when programs attempt to open existing files with names starting with `$sys$`, protecting XCP's files from reading or writing by other programs.

3. `NtQuerySystemInformation` – One use of this function is to obtain a list of running processes. The rootkit filters out any processes with names prefixed by `$sys$`, making them invisible to other applications.

4. `NtEnumerateKey` – This function returns a list of the subkeys of a registry key. The rootkit filters the results to remove subkeys with names starting with `$sys$`. Note that it does not conceal individual fields within the registry ("values" in Windows parlance) with names starting with `$sys$`.

5. `NtOpenKey` – This function opens a registry key for reading or modifying. The rootkit intercepts this function call but does not alter its behavior. Its authors may have intended to restrict access to hidden registry keys in the same way that the hooked `NtQueryDirectoryFile` call restricts access to hidden files, but they did not ship a working implementation of this behavior.

On intercepting a function call, the rootkit checks the name of the calling process. If the name of the calling process begins with `$sys$`, the rootkit returns the results of the real kernel function without alteration so that XCP's own processes have an accurate view of the system.

The XCP rootkit increases users' vulnerability to attack by allowing any software to hide—not just XCP. Malware authors can exploit the fact that any files, registry keys, or processes with names beginning in `$sys$` will be hidden, thereby saving the trouble of installing their own rootkits. Malware that lacks the privileges to install its own rootkit can still rely on XCP's rootkit.

Only kernel-level processes can patch the Windows system service dispatch table, and only privileged users—normally, members of the Administrators or Power Users groups—can install such processes. (XCP itself requires these privileges to install.) Malicious code running as an unprivileged user can't normally install a rootkit that intercepts system calls. But if the XCP rootkit is installed, it will hide all programs that adopt the `$sys$` prefix so that even privileged users will be unable to see them. This vulnerability has already been exploited by at least two Trojan horses seen in the wild [21, 17].

Another privilege escalation attack facilitated by the XCP rootkit allows an unprivileged application to crash the system. Russinovich demonstrated this problem using an automated testing program he created called NTCRASH2 [30]. This utility makes repeated system calls with randomly generated invalid parameters. The original Windows kernel functions handle invalid inputs correctly and the system remains stable, but with the XCP rootkit installed, certain invalid inputs result in a system crash.

We investigated the specific circumstances when these crashes occur. The rootkit's implementation of NtCreateFile can cause a crash if it is passed an invalid pointer as its ObjectAttributes argument, or if it is passed a valid ObjectAttributes structure that points to a ObjectName structure with an invalid Buffer pointer. We do not believe that an attacker could exploit these flaw to execute code; however, they do allow an unprivileged user to bring the system to a halt. As Russinovich and others have pointed out, these problem illustrates the security danger of installing software in secret. Users experiencing system instability due to these rootkit bugs would have great difficulty diagnosing the problem, since they likely would be unaware of the rootkit's presence.

7.3 Deactivating XCP

XCP's active protection is more complicated to deactivate than MediaMax's, because it comprises several processes that are more deeply entangled in the system configuration and are hidden by the XCP rootkit. Deactivation requires a three-step procedure, which we describe here in detail so that affected users can decontaminate their systems.

1. The first step is to remove the rootkit. From the command prompt, run `sc delete $sys$aries`. Delete the rootkit's program file `%windir%\system32\$sys$filesystem\aries.sys`, and reboot the system. Disabling the rootkit exposes the previously hidden files, registry entries, and processes.

2. Next, edit the system registry to remove references to XCP's filter drivers and CoDeviceInstallers. XCP uses the Windows filter driver facility to intercept commands to the CD drives and IDE bus. If these filter drivers are not removed, the CD and IDE device drivers will fail to initialize after the program files for the filter drivers are deleted. This can cause the CD drives to malfunction, or, worse, cause the system to fail to boot because the IDE device driver is disabled.

   First remove references to the `$sys$cor` filter driver, which intercepts commands sent to the IDE device. Use the Windows Registry Editor to search for occurences of `$sys$cor` in registry entries named `UpperFilters`. Edit each list of filters to remove the reference to `$sys$cor`. (You will need to temporarily change the security permissions on the enclosing registry key to grant yourself permission to edit the filters list.) References to this filter driver may occur in multiple registry keys; be sure to remove them all.

   Repeat this step to remove references to the `$sys$crater` filter driver, which intercepts commands sent to the CD drive. This filter driver appears in devices' `LowerFilters` lists. Be sure to remove all occurences.

   Search the registry once again for `$sys$caj.dll`. This file is configured as a CoDeviceInstaller for the CD-ROM and IDE devices. It installs the filter drivers when any new CD drive or IDE bus device is configured. Remove the lines from any list of CoDeviceInstallers in which they appear: `$sys$caj.dll,CoInstallCdrom`, `$sys$caj.dll,CoInstallPC`.

3. The next step is to delete the XCP services and remove the XCP program files. Open a command prompt and issue these commands:

   ```
   sc delete $sys$crater
   sc delete $sys$lim
   sc delete $sys$oct
   ```

```
sc delete cd_proxy
sc delete $sys$drmserver
sc delete $sys$cor
del %windir%\system32\$sys$filesystem\crater.sys
del %windir%\system32\$sys$filesystem\lim.sys
del %windir%\system32\$sys$filesystem\oct.sys
del %windir%\system32\drivers\$sys$cor.sys
del %windir%\system32\$sys$caj.dll
del %windir%\system32\$sys$upgtool.exe
```

Reboot and remove two remaining XCP program files:

```
del %windir%\CDProxyServ.exe
del %windir%\system32\$sys$filesystem\$sys$DRMServer.exe
```

Performing these steps will deactivate the XCP active protection, leaving only the passive protection on XCP CDs in force. The procedure could easily be automated to create a point-and-click removal tool.

7.4 Impact of Spyware Tactics

The use of rootkits and other spyware tactics harms users by undermining their ability to manage their computers. If users lose effective control over which programs run on their computers, they can no longer patch malfunctioning programs or remove unneeded programs. Managing a system securely is difficult enough without spyware tactics making it even harder.

Though it is no surprise that spyware tactics would be attractive to DRM designers, it is a bit surprising that mass-market DRM vendors chose to use those tactics despite their impact on users. If only one vendor had chosen to use such tactics, we could write it off as an aberration. But two vendors made that choice, which is probably not a coincidence. We suspect that the vendors let the lure of platform building override the risk to users.

7.5 Summary of Deactivation Attacks

Ultimately, there is little a CD DRM vendor can do to stop users from deactivating active protection software. Vendors' attempts to frustrate users' control of their machines are harmful and will trigger a strong backlash from users. In practice, vendors will probably have to provide some kind of uninstaller—users will insist on it, and some users will need it to deal with the bugs and incompatibilities that crop up inevitably in complex software. Once an uninstaller is released, users can use it to remove the DRM software. Determined users will be able to keep CD DRM software off their machines.

8 Uninstallation

The DRM vendors responded to user complaints about spyware-like behavior by offering uninstallers that would remove their software from users' systems. Uninstallers had been available before but had been very difficult to acquire. For example, to get the original XCP uninstaller, a user had to fill out an online form involving personal information, then wait a few days for a reply email, then fill out another online form and install some software, then wait a few days for yet another email, and finally click a URL in the last email. It is hard to explain this complexity of this procedure, except as a way to deter users from uninstalling XCP.

The uninstallers, when users did manage to get them, did not behave like ordinary software uninstallers. Normal uninstallers are programs that can be acquired and used by any user who has the software. The first XCP uninstaller was customized for each user so that it would only work for a limited time and only on the computer on which the user had filled out the second form. This meant, for example, that if a user uninstalled XCP but it was reinstalled later—say, if the user inserted an XCP CD—the user could not use the same uninstaller again but would have to go through the entire process again to request a new one.

Customizing the uninstaller is more difficult, compared to a traditional uninstaller, for both vendor and user, so it must benefit the vendor somehow. Probably the benefit is to the vendor's platform building strategy, which takes a step backward every time a user uninstalls the software. Customizing the uninstaller allows the vendor to control who receives the uninstaller and to change the terms under which it is delivered.

As user complaints mounted, Sony-BMG announced that unrestricted uninstallers for both XCP and MediaMax would be released from the vendors' web sites. Both vendors chose to make these uninstallers available as ActiveX controls. By an unfortunate coincidence, both uninstallers turned out to open the same serious vulnerability on any computer where they were used.

8.1 MediaMax Uninstaller Vulnerability

The MediaMax uninstaller used a proprietary ActiveX control, called `AxWebRemove.ocx`, created and signed by SunnComm. Users visiting the MediaMax uninstaller web page were prompted to install the control, then the web page uninstalled MediaMax by invoking one of the control's methods. This method, `Remove`, took two parameters: `key`, and `validate_url`. When `Remove` was called it issued an HTTP GET to `validate_url` to validate `key`. If `key` was valid, the server at `validate_url` would respond with the message `true,`<*uninstall_url*>, where *uninstall_url* was the URL of a DLL file containing code to uninstall MediaMax. The control would retrieve this DLL file from the Internet and save it to a temporary location, then call a function in the DLL named `ECF7` to perform the uninstallation. If the function returned success, the control would issue a second HTTP GET request to `validate_url` to report that the uninstall was successful and that the single-use key should be retired.

This design is vulnerable because the control accepts an arbitrary `validate_url` parameter and does not check that the DLL supplied by the server at that URL is authentic. The ActiveX control is not itself removed during the uninstallation process, so its methods can be invoked later by any web page without further browser security warnings. An attacker can create a web page that invokes the `Remove` method and provides a `validate_url` pointing to a page under the attacker's control. This page can accept whatever key is presented and return an `uninstall_url` pointing to a DLL created by the attacker. When the MediaMax control executes the uninstall function in this file, arbitrary attacker code will execute on the user's machine.

8.2 XCP Uninstaller Vulnerability

The XCP uninstaller contains the same design flaw and is only slightly more difficult to exploit. XCP's ActiveX-based uninstaller invokes a proprietary ActiveX control named `CodeSupport.ocx`. (Early versions of XCP's rootkit removal patch utilized the same control.) Usually this control is installed in the second step of the three-step XCP uninstall process. In this step, a pseudorandom code generated by the ActiveX control is sent to the XCP server. The same code is written to the system registry. Eventually the user receives an email with a link to another web page that uses the ActiveX control to remove XCP, but only after verifying that the correct code is in the registry on the local system. This check tethers the uninstaller to the machine from which the uninstallation request was made. Due to this design, the vulnerable control may be present on a user's system even if she never performed the step in the uninstallation process where XCP is removed.

Matti Nikki first noted that the XCP ActiveX control contains suspiciously-named methods, including `InstallUpdate(url)`, `Uninstall(url)`, and `RebootMachine()` [25]. He demonstrated that the control was still present after the XCP uninstallation was complete, and that its methods (including one that rebooted the computer) were scriptable from any web page without further browser security warnings.

We found that the `InstallUpdate` and `Uninstall` methods have an even more serious flaw. Each takes as an argument a URL pointing to a specially formatted archive that contains updater or uninstaller code and data files. When these methods are invoked, the archive is retrieved from the provided URL and stored in a temporary location. For the `InstallUpdate` method, the ActiveX control extracts from the archive a file named `InstallLite.dll` and calls a function in this DLL named `InstallXCP`.

Like the MediaMax ActiveX control, the XCP control does not validate the download URL or the downloaded archive. The only barrier to using the control to execute arbitrary code is the proprietary format of the archive file. We determined the format by disassembling the control. The archive file consists of several blocks of gzip-compressed data, each storing a separate file and preceded with a short header. At the end of the archive, a catalog structure lists metadata for each of the blocks, including a 32-bit CRC. The control verifies this CRC before executing code from the DLL.

With knowledge of this file format, we constructed an archive containing benign proof-of-concept exploit code. The most difficult detail was the CRC, which is computed with an apparently proprietary algorithm that proved tedious to reverse engineer. We saved the trouble by having the ActiveX control compute the CRC for us. The control checks the CRC by computing a CRC for the file data in the archive and verifying that it matches the CRC specified in the archive catalog. We inserted a break point where the comparison occurs and ran the control on an archive containing code we prepared. We then took the CRC computed by the control and placed it in the archive catalog. Thus modified, the archive passed the CRC check and the ActiveX control executed our code. (This illustrated why digital signatures, rather than CRCs, must be used to validate code from untrusted sources.)

This procedure would allow a malicious web site to execute arbitrary code on the user's machine. Like the MediaMax uninstaller flaw, it is especially dangerous because users who have completed the XCP uninstallation may not be aware that they are still vulnerable.

Obviously, these vulnerabilities could have been prevented by careful design and programming. But they were only possible at all because the vendors chose to deliver the uninstallers via this ActiveX method rather than using an ordinary download. We conjecture that the vendors made this choice because they wanted to retain the ability to rewrite, modify, or cancel the uninstaller later, in order to further their platform building strategy.

9 Compatibility and Software Updates

Compared to other media on which software is distributed, compact discs have a very long life. Many compact discs will still be inserted into computers and other players twenty years or more after they are first bought. If a particular version of DRM software is shipped on a new CD, that software version may well try to install and run decades after it was developed. The same is not true of most software, even when shipped on a CD-ROM. Very few if any of today's Windows XP CDs will be inserted into computers in 2026; but today's music CDs will be, so their DRM software must be designed carefully for future compatibility.

The software should be designed for *safety*, so as not to cause crashes or malfunction of other software, and may be designed for *efficacy*, to ensure that its anti-copying features remain effective.

9.1 Supporting Safety by Deactivating Old Software

Safety is easier to achieve, and probably more important. One approach is to design the DRM software to be inert and harmless on future systems. Both XCP and MediaMax do this by relying on Windows autorun, which is likely to be disabled in future versions of Windows for security reasons. If the upcoming Windows

Vista disables autorun by default, XCP and MediaMax will be inert on most Vista systems. Perhaps XCP and MediaMax used autorun for safety reasons; but more likely, this choice was expedient for other reasons.

Another safety technique is to build in a sunset date, after which the software will make itself inert. A sunset would improve safety but would have relatively little effect on record label revenue for most discs, as we expect nearly all revenue from the disc to have been extracted from the customer in the first three years after she buys the disc. If more copies of the disc were pressed, with updated DRM software, these could have a later sunset.

9.2 Updating the Software

When a new version of DRM software is released, it can be shipped on newly pressed CDs, but existing CDs cannot be modified retroactively. Updates for existing users can be delivered either by download or on new CDs. Downloads are faster but require an Internet connection; CD delivery is slower but can reach non-networked machines.

Users will generally cooperate with updates that help them, by improving safety, or otherwise making the software more useful. But updates to retain the efficacy of the software's usage controls will not be welcomed by users. Usage controls provide no value to individual user, but only reduce what the user can do with music from a disc. Because usage controls typically control some uses that are allowed under copyright law, even law-abiding users would prefer to avoid usage controls, and therefore would not welcome updates that prolonged their efficacy.

Users have many ways to stop updates from downloading or installing, for example by write-protecting the software's code so that it cannot be updated, or using a personal firewall to block network connections to the vendor's download servers. System security tools, which are designed generally to stop unwanted network connections, downloads, and code installation, can be set to treat CD DRM software as malware. If users want to block updates, makers of system security tools will have an incentive to provide tools capable of doing so.

A DRM vendor who wants to deliver unwanted updates has two options. First, the vendor can simply offer updates and hope some users will not bother to block them. For the vendor and record label, this is better than nothing. Alternatively, the vendor can try to force users to accept updates.

9.3 Forcing Updates

If a user has the ability to block DRM software updates, a vendor who wants an update must somehow convince the user that updating is in her best interest. One approach is to make a non-updated system painful to use.

Ruling out dangerous and legally risky tactics such as logic bombs that destroy the user's system or hold her (unrelated) data hostage, the vendor's strongest tactic for forcing updates is to make the DRM software block all access to protected CDs until the user accepts an update. The DRM software might check with a network server, which would produce periodically a digitally signed and dated certificate listing allowed versions of the DRM software. If the software on the user's system found that its version number was not on the list (or if it could not get a recent list), it would block all access to protected discs. The user would then have to update to a new version to get access to her protected CDs.

This approach would convince some users to update, and would thereby prolong the DRM's efficacy for those users. But it has several drawbacks. If the computer is not networked, the software will eventually lock down because it cannot get certificates. (If the software kept working in this case, users could avoid updates by preventing the DRM software from making network connections.) A bug in the software could cause an accidental but irreversible lockdown. The software could lock itself down if the vendor's Internet site is shut down, for example if the vendor goes bankrupt.

Strong-arm tactics can also be counterproductive, by giving the user further reason to defeat or remove the DRM software.[12] The software is more likely to remain on the user's system if it does not behave annoyingly. Trying to force updates can reduce the DRM system's efficacy if it convinces users to remove the DRM altogether.

Users will tend to be suspicious of software updates in any case. From the user's standpoint, every software update is a security risk, which could carry hostile or buggy code onto the user's system. Users will worry that the update adds a security hole or backdoor into the vendor's software, or that the encryption key that the vendor uses to sign updates has been compromised. Careful users try to minimize the amount of new software coming onto their systems, and so they will tend to resist software updates, especially mandatory ones.

Given the problems with forced updates, and the user backlash they likely would have triggered, we are not surprised that neither XCP nor MediaMax tried to force updates.

10 Conclusion

Our analysis of Sony-BMG's CD DRM carries wider lessons for content companies, DRM vendors, policymakers, end users, and the security community. We draw six main conclusions.

First, the design of DRM systems is driven strongly by the incentives of the content distributor and the DRM vendor, but these incentives are not always aligned. Where they differ, the DRM design will not necessarily serve the interests of copyright owners, not to mention artists.

Second, DRM, even if backed by a major content distributor, can expose users to significant security and privacy risks. Incentives for aggressive platform building drive vendors toward spyware tactics that exacerbate these risks.

Third, there can be an inverse relation between the efficacy of DRM and the user's ability to defend the computer from unrelated security and privacy risks. The user's best defense is rooted in understanding and controlling which software is installed on the computer, but many DRM systems rely on undermining the user's understanding and control.

Fourth, CD DRM systems are mostly ineffective at controlling uses of content. Major increases in complexity have not increased their effectiveness over that of early schemes, and may in fact have made things worse by creating more avenues for attack. We think it unlikely that future CD DRM systems will do better.

Fifth, the design of DRM systems is only weakly connected to the contours of copyright law. The systems make no pretense of enforcing copyright law as written, but instead seek to enforce rules dictated by the label's and vendor's business models. These rules, and the technologies that try to enforce them, implicate other public policy concerns, such as privacy and security.

Finally, the stakes are high. Bad DRM design choices can seriously harm users, create major liability for copyright owners and DRM vendors, and ultimately reduce artists' incentive to create.

Acknowledgments

We are grateful for the expert legal advice of Deirdre Mulligan and her colleagues at U.C. Berkeley: Aaron Perzanowski, Sara Adibisedeh, Azra Medjedovic, Brian W. Carver, Jack Lerner, and Joseph Lorenzo Hall. We are also grateful to Clayton Marsh at Princeton. Sadly, research of this type does seem to require support from a team of lawyers.

[12]Users could also mislead the DRM software about the date and time, but most users with the inclination to do that would probably just remove the DRM software altogether.

We thank the readers of Freedom to Tinker for their comments on partial drafts that we posted there; thanks especially to C. Scott Ananian, Randall Chertkow, Tim Howland, Edward Kuns, Jim Lyon, Tobias Robison, Adam Shostack, Ned Ulbricht, and several pseudonymous commenters. Shirley Gaw, Janek Klawe, and Harlan Yu provided helpful last-minute suggestions. Thanks to Claire Felten for help with copy-editing.

This material is based upon work supported under a National Science Foundation Graduate Research Fellowship. Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Class action complaint. In *Hull et al. v. Sony BMG et al.*, 2005. http://www.eff.org/IP/DRM/Sony-BMG/sony_complaint.pdf.

[2] Consolidated amended class action complaint. In *Michaelson et al. v. Sony BMG et al.*, 2005. http://sonysuit.com/classactions/michaelson/15.pdf.

[3] Original plantiff's petition. In *State of Texas v. Sony BMG Music Entertainment*, 2005. http://www.oag.state.tx.us/newspubs/releases/2005/112105sony_pop.pdf.

[4] Peter Biddle, Paul England, Marcus Peinado, and Bryan Willman. The Darknet and the future of content distribution. In *ACM Workshop on Digital Rights Management*, November 2002.

[5] Jesse Burns and Alex Stamos. Media Max access control vulnerability, November 2005. http://www.eff.org/IP/DRM/Sony-BMG/MediaMaxVulnerabilityReport.pdf.

[6] Computer Associates. Disabling autorun. http://www3.ca.com/securityadvisor/pest/collateral.aspx?cid=76351.

[7] Ingemar Cox, Joe Kilian, Tom Leighton, and Talal Shamoon. Secure spread spectrum watermarking for multimedia. *IEEE Transactions on Image Processing*, 6(12):1673–1687, 1997.

[8] Scott A. Craver, Min Wu, Bede Liu, Adam Stubblefield, Ben Swartzlander, Dan S. Wallach, Drew Dean, and Edward W. Felten. Reading between the lines: Lessons from the SDMI challenge. In *Proc. 10th USENIX Security Symposium*, August 2001.

[9] Steven Davis. New RealPlayer avoids Apple DRM license. *eWeek*, January 2004. http://www.eweek.com/article2/0,1895,1523392,00.asp.

[10] Edward W. Felten and J. Alex Halderman. Digital rights management, spyware, and security. *IEEE Security and Privacy*, January/February 2006.

[11] Allan Friedman, Roshan Baliga, Deb Dasgupta, and Anna Dreyer. Underlying motivations in the broadcast flag debate. In *Proc. Telecommunications Policy Research Conference*, September 2003.

[12] J. Alex Halderman. Evaluating new copy-prevention techniques for audio CDs. In *Proc. ACM Workshop on Digital Rights Management (DRM)*, Washington, D.C., November 2002.

[13] J. Alex Halderman. Analysis of the MediaMax CD3 copy-prevention system. Technical Report TR-679-03, Princeton University Computer Science Department, Princeton, New Jersey, 2003.

[14] Sam Hocevar. Suspicious activity? Indeed, November 2005. http://sam.zoy.org/blog/2005-11-21-suspicious-activity-indeed.

[15] Greg Hoglund. 4.5 million copies of EULA-compliant spyware, October 2005. http://www.rootkit.com/blog.php?newsid=358.

[16] Greg Hoglund and James Butler. *Rootkits: Subverting the Windows Kernel*. Addison-Wesley, 2005.

[17] Kazumasa Itabashi. Trojan.Welomoch technical description, December 2005. http://securityresponse.symantec.com/avcenter/venc/data/trojan.welomoch.html.

[18] Jon Lech Johansen and Sam Hocevar. DRMS source code. http://trac.videolan.org/vlc/file/trunk/modules/demux/mp4/drms.c.

[19] Auguste Kerckhoffs. La cryptographie militaire. *J. des Sciences Militaires*, 9:161–191, 1883.

[20] Darko Kirovski and Fabien A.P. Petitcolas. Replacement attack on arbitrary watermarking systems. In *Proc. ACM Workshop on Digital Rights Management*, 2002.

[21] Yana Liu. Backdoor.Ryknos.B technical description, November 2005. http://securityresponse.symantec.com/avcenter/venc/data/backdoor.ryknos.b.html.

[22] MediaMax Technology Corp. Annual report (S.E.C. Form 10-KSB/A), September 2005.

[23] Microsoft Corporation. Windows Media data session toolkit. http://download.microsoft/com/download/a/1/a/a1a66a2c-f5f1-450a-979b-ddf790756f1d/Data_Session_Datasheet.pdf.

[24] Nero AG. Nero Burning ROM. http://ww2.nero.com/enu/Products.html.

[25] Matti Nikki. Muzzy's research about Sony's XCP DRM system, December 2005. http://hack.fi/~muzzy/sony-drm/.

[26] Fabien A.P. Petitcolas, Ross J. Anderson, and Markus G. Kuhn. Attacks on copyright marking systems. In *Information Hiding*, pages 218–238, 1998.

[27] K. Reichert and G. Troitsch. Kopierschutz mit filzstift knacken. *Chip.de*, May 2002.

[28] Mark Russinovich. More on Sony: Dangerous decloaking patch, EULAs and phoning home, November 2005. http://www.sysinternals.com/blog/2005/11/more-on-sony-dangerous-decloaking.htm.

[29] Mark Russinovich. Sony, rootkits and digital rights management gone too far, October 2005. http://www.sysinternals.com/blog/2005/10/sony-rootkits-and-digital-rights.html.

[30] Mark Russinovich. Sony's rootkit: First 4 Internet responds, November 2005. http://www.sysinternals.com/blog/2005/11/sonys-rootkit-first-4-internet.html.

[31] Sony-BMG Music Entertainment. Portable device: iPod information. http://cp.sonybmg.com/xcp/english/form10.html.

[32] Sony-BMG Music Entertainment. XCP frequently asked questions. http://cp.sonybmg.com/xcp/english/faq.html.

[33] André Wiethoff. Exact Audio Copy. http://www.exactaudiocopy.de/.

[34] Wikipedia. Fairplay. http://en.wikipedia.org/wiki/FairPlay.