VERIFICATION AND APPLICATION OF CONCEPTUAL MODEL AND SECURITY REQUIREMENTS ON PRACTICAL DRM SYSTEMS IN E-LEARNING

Maria Nickolova, Eugene Nickolov

Abstract: The paper represents a verification of a previously developed conceptual model of security related processes in DRM implementation. The applicability of established security requirements in practice is checked as well by comparing these requirements to four real DRM implementations (Microsoft Media DRM, Apple's iTunes, SunnComm Technologies's MediaMax DRM and First4Internet's XCP DRM). The exploited weaknesses of these systems resulting from the violation of specific security requirements are explained and the possibilities to avoid the attacks by implementing the requirements in designing step are discussed.

Keywords: Security, DRM protection, e-learning.

ACM Classification Keywords: D.4.6 Security and Protection

Introduction

"E-learning" - instructional content or learning experiences through electronic technology - is expanding rapidly and is transforming how and where students learn. E-learners can be taught in very large numbers, but also in very small classes, or even as individuals, anytime and anywhere. As a result, e-learning becomes a highly cost-effective and adaptable medium for small education and training institutions and small businesses with limited resources for large overheads. E-learning offers potentially universal access to "the best" content, regardless of location, and it can transform education and training from a passive consumption experience to a more flexible and learner-cantered one. As a consequence of these changes, a new digital educational content market is born with a new commercial approach: from the distribution and sale of tangible goods to the distribution and licensing of intangible products. This new approach has a very strong impact also on the management of the rights linked to educational content (copyright and licensing, etc.).

There are many reasons for wanting to manage the rights associated with the e-learning content. Authors and artists wish to control what is done with their creations, scholars want to ensure that they receive proper attribution, commercial enterprises wish to support business models that involve licenses and fees, and e-learners want an environment free of juridical problems and unexpected costs. Although rights themselves are not technological in nature (they are defined by laws, beliefs and practices), technology can be used to transmit, verify, interpret and enforce rights as they apply to digital content and services – this is the so called digital rights management (DRM). DRM systems are designed to provide a solution for a security problem and that is why their own security is a very important factor in that process.

Although DRM systems use various security techniques (cryptography, watermarking, fingerprinting, etc.), to evaluate the own security of an entire DRM system we need an all-encompassing security model, because the security of the individual components of a system does not guarantee the security of the system as a whole. In [1] we described the security related motives, responsibilities and goals of all participators involved with a DRM system. Based on them, we built a generic conceptual model of security-related processes of DRM implementation used in e-learning and established the core security requirements for this system. The next logical step is to verify the applicability of these security requirements in practice and to analyze their fulfillment in some real DRM implementations.

Microsoft Windows Media DRM

Windows Media DRM (WMDRM) is designed by Microsoft to provide secure delivery of audio and/or video content for the Windows Media platform. The principal scheme of WMDRM (the role of the DRM system is to deliver protected content to the user which can only be accessed with a legitimate license) matches the generic model described in [1] and includes the following components:

- Windows Media Rights Manager SDK for packaging content and issuing licenses

- Windows Media Format SDK (WMF SDK) for building Windows applications which support DRM and the Windows Media format
- Windows Media DRM for Portable Devices (WMDRM-PD) for supporting offline playback on portable devices
- Windows Media DRM for Network Devices (WMDRM-ND) for streaming protected content to devices attached to a home network

Media Player stores the DRM key in separate DLL files in a Windows directory. The encryption is placed in the blackbox.dll data file. After personalizing the first license in the data file IndivBox.key (also a DLL), a specific version of the blackbox.dll for the individual PC is written. This special version also includes the hardware ID of the PC.

There have been two well-known successful attacks on Microsoft's DRM system [2]. Both have worked in the same way. Rather than break the encryption itself, which is practically incomputable, they hook or interfere with the "black box" component as it runs to dump out the content keys or the unencrypted content from the memory.

The first attack consisted of producing an unprotected copy of the protected WMA file. For the attack to work the user must already have acquired a license for the original WMA file and Windows Media Player must be able to play it. The cracking software plays the file which usually would lead to a wave output of the file's contents to the speaker system of the PC. However, the cracking software captures this sound output, converts it back into an unprotected WMA file and saves it on the disk. Using this mechanism, the loss of quality is minimal; the only loss of information during the process occurs during the audio encoding back to WMA which is not a lossless format. Apart from that, all information is preserved, including stereo information. The resulting WMA files are not protected in any way.

This sort of attack could have been prevented if the following security requirements, we set respectively for the content creator and content publisher were met by the Microsoft DRM implementation:

- (authentication) No component of the DRM scheme sends the protected content to another component, unless the receiving component is authenticated as an official component (i.e. created by the DRM developer) and allowed to receive the content from the sending component
- (secrecy) The content is only accessible by the end user specified in the license and all information and remains secret until it has been converted into analogue form.

Further, all attacks of this type target the same part of the DRM system each and every time. This means that the attack is possible on all installed systems, which violates the requirement about damages limitation we set for the content creator (Prevent the possibility for "break once, run always").

After the first successful attack Microsoft released a new version of Windows DRM that was not vulnerable. Of course content packaged under the new system was not accessible with the old system. But soon a tool called "FreeMe" was created that attacks content protected by the new DRM version which is able to determine the keys used to encrypt the content. This is a violation of the following secrecy requirement: No secret information necessary for the operation of the components or pertaining to the content (e.g. cryptographic keys, content) can be discerned from the content creator /licensed user pair, or from the communication channel between them.

The manner in which the key could be retrieved is the same for each installed system, which again violates the requirement about damages limitation.

In early 2005 Microsoft has released a patch for their DRM system which implements measures that support the requirement about damages limitation and uses another method to hide the used keys. Patched systems are no longer vulnerable to this particular attack.

But this newest DRM technology is used by hackers in another way: to install spyware, adware, diallers and computer viruses on PCs. In 2005 two Trojans, Trj/WmvDownloader.A and Trj/WmvDownloader.B arrived on a compromised computer as a license-protected multimedia files that are either Windows Media Audio (.wma) or Windows Media Video (.wmv) formats. Usually, when someone tries to start a license-protected file but doesn't own the necessary license, WMP searches for a license on the Internet. The two Trojans pretend to download the corresponding license, but instead they redirect to a Web site (serve.alcena.com or protectedmedia.com) that loads a large quantity of modem diallers and other viruses and installs a list of more than a dozen spyware and adware programs onto the user's PC.

Fortunately, Windows Media DRM is designed to be renewable, that is on the assumption that it will be cracked and must be constantly updated by Microsoft. The result is that while the DRM scheme has been cracked several times, it usually does not remain cracked for long.

Apple iTunes DRM

iTunes is sold in the Advanced Audio Coding format (AAC) [3]. Copy protection for the iTunes Music Store's media is provided by Apple Digital Rights Management system, also known as FairPlay. The manner in which iTunes operates also conforms to our model in [1]. Content is delivered to a license creator, who binds it to a license. The user can then acquire the bound content together with an appropriate license.

Fairplay limits the usage of the media to a number of computers and iPods. The default license allows protected files to be played on up to five different computers, and to burn them to CDs. Another limitation is that the only operating systems that are accessible to iTunes are Mac OS and Microsoft Windows, which leaves some widespread systems such as Linux uncovered. It is clear that the constraints of iTunes DRM protection violate the requirement for usability and universality:

- The DRM scheme must not be too restrictive for consumers who legitimately paid for content and want to share it on several devices
- The DRM must work in disconnected environments, multiplatform and multi business models

This lack of universality has logically led to the “hacking” of Apple's FairPlay system although the audio data inside the AAC file is protected by strong encryption.

The first attack was realized by the Norwegian Jon Lech Johansen. Because the iTunes DRM scheme is not compliant to the requirement of self-protection (Besides the basic requirements valid for secure software the DRM implementation must be reverse-engineering-proof and tamper-resistant), he reverse engineered successfully Apple FairPlay and released in November 2003 QTFairUse (written in C) - an open source program which dumps the raw output of a QuickTime AAC stream to a file before it is converted into analogue form. FairPlay works by first decoding the file using a user key from iPod or the Windows system and then creating new mp3 files with the metadata intact without the DRM code. Although the resulting raw AAC files were unplayable by many media players, their raw format required only some trivial additions to convert them to an MP4 files that could be played on any computer. This attack is very similar to the first attack upon Microsoft's Windows DRM system, and the remarks we made above may apply in this case as well (this attack exploits the lack of compliance to the same requirements for authentication and secrecy).

The second attack on iTunes shows also a close resemblance to the second attack on Windows DRM. The program iTunes uses HTTP XML messages to communicate with the iTunes Music Store. The keys used by iTunes (to protect the main key for the encrypted content) are stored encrypted with a system key. The communication is encrypted using an AES128 CBC algorithm. The public disclosure of a list of common XML commands for the iTunes Music Store on the Internet quickly resulted in an exploit, enabling an attacker to compromise the encryption of protected content. The encryption key was determined to be “8a 9d ad 39 9f b0 14 c1 31 be 61 18 20 d7 88 95”. It is known how to reconstruct this key for the Windows platform and for Apple's portable mp3 player. This means that it became possible to remove the encryption from the protected content for these platforms. The considerations about the second attack on MS DRM apply here as well (this attack exploits the lack of compliance to requirements about secrecy).

2005 saw another exploit working on Apple's DRM [4] that dealt with the communication protocol between Apple iTunes and Apple iTunes Music Store. The attacking program interfaced directly with the Apple iTunes Music Store online service, bypassing the security verification within the Apple iTunes program. When content was purchased, the attacking program was able to download an unprotected file from Apple’s servers because the addition of DRM was performed in the Apple iTunes program. This attack exploits the non-compliance to the requirements about independence and authentication:

- The implementation must not rely on a trusted software component or on the user’s computer/device to perform integrity checking, decrypt the content or enforce the usage rights associated with the e-learning content.

- No component of the DRM scheme sends the protected content to another component, unless the receiving component is authenticated as an official component (i.e. created by the DRM developer) and allowed to receive the content from the sending component.

Recently, a third attack on the system was crafted by Johanson [5]. Working with Travis Watkins and Cody Brocious, he went a step further and in March 2005, introduced PyMusique, an open source iTunes music store client that exploited another hole in Apple's DRM, file encoding. Music files transferred from the iTunes music store do not get encrypted by the server, but by the user's iTunes software. PyMusique mimics the iTunes application and connects to the iTunes music store server to purchase (even to register) using real iTunes user identification. As a result, during the purchase of a song, PyMusique does not encrypt the file. Another user advantage of PyMusique is that unlike iTunes, the user can re-download the song just in case he unintentionally deleted the original file. PyMusique also performs iTunes functionalities like browsing through and previewing available songs.

Because this attack consists of replacing a system component with an attacker-supplied component it is clear that the requirement about authentication is violated (no component should send the content to another component, unless the other component identifies itself as a legitimate component).

iTunes version 4.6 was especially vulnerable to PyMusique, but Apple quickly upgraded it to iTunes 4.7. As a counter-attack, Johanson and his partners released PyMusique version 0.4, the last version for Windows and PyMusique's descendent SharpMusique (a C# port of PyMusique) [5].

Another security problem in Apple's DRM scheme is that iTunes up to version 4.7 contains a vulnerability caused by a bug in the way the software handles "m3u" and "pls" playlist files. The problem specifically exists when parsing playlist files that contain long URL file entries. Malicious playlist files can come with either the .m3u or .pls extension. Though their formats are different, the vulnerability in each is the same.

For example a malicious .pls file containing a long URL looks like this:

File1=http://[A x 3245]1234

And a malicious .m3u file with a long URL looks like this:

http://[A x 3245]1234

In both cases '[A x 3245]' represents any string of 3,245 bytes in length. Opening either a malicious playlist file on the Microsoft Windows platform will cause iTunes to crash with an access violation when attempting to execute instruction 0x34333231 (the ASCII representation of 1234). An attacker can exploit this vulnerability to redirect the flow of control and eventually execute some arbitrary code. This attack is conditioned by the violation of the requirement about safety (The specific DRM solution must be carefully checked up from the end user's security point of view).

SunnComm Technologies's MediaMax DRM

The MediaMax technology is based on the active protection method relying on software on the end-user's computer that actively intervenes to block access to the e-learning content by programs other than the DRM vendor's own software. To install the DRM MediaMax uses Windows autorun, which (when enabled) automatically loads and runs software from the disk inserted into the computer's drive. Once the DRM software is installed, every time a new CD is inserted the software runs a recognition algorithm to determine whether the disk is associated with the DRM scheme. If it is, the active protection software won't allow access to the disk, except when it originates from the vendor's own content player application. The proprietary player application, which is shipped with the disk, gives the user limited access to the content.

A security problem arose when in March 2005 MediaMax version 5.0.21 was released. Its installer sets file permissions that allow any process, user, or network client, including low rights accounts, to read, modify, delete or replace the executable code with malicious one. Granting all users "Full Control" rights to executables that will be automatically run by high rights users creates simple but serious security vulnerability. After MediaMax software installation the "SunnComm Common" directory contains the executable MMX.EXE, which inherits the weak security protections. The next time a user plays a MediaMax-protected CD, the possible attack code will be executed with almost complete control over the system. This security hole results from the violation of the safety requirement (The specific DRM solution must be carefully checked up from the end user's security point of view).

Usually, this problem could be fixed by manually correcting the wrong permissions. However, aiming to enforce the DRM protection 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. This principle violates the priority requirement (In case of contradiction between the requirements for illegitimate access prevention and these for end user security, the last must prevail).

The worst is that a malicious attack is possible even if the end user has never consented to the installation of MediaMax, which could be triggered immediately whenever the user inserts a MediaMax CD. The attacker could place a hostile code in the DllMain procedure of a code file called MediaMax.dll, which MediaMax installs even before displaying the user license agreement. 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 DllMain procedure to execute, along with any attack code placed there. This security problem results from the violation of one of the principal security goals of the content creator (The site structure (or physical media) should not use any potentially harmful methods of content protection (active protection measures as ActiveX controls, backdoors capabilities, etc.) which could put at risk the security of end users' systems).

Fixing this problem permanently without losing the use of protected disks required installing a patch from SunnComm. Unfortunately, the patch released initially 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.

But the worst is that the MediaMax uninstaller which is supposed to recover the end user security protection introduces another security hole. It uses a proprietary ActiveX control, *AxWebRemove.ocx*, which 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 itself is not removed during the uninstall process, so its methods can be invoked later by any web page without further browser security warnings. An attacker could create a web page that invokes the installation method and provides a *validate burl* pointing to a page under the attacker's control and the possibility to execute an arbitrary attacker code on the user's machine.

First4Internet's XCP DRM

This DRM technology uses an active protection method similar to the one used by MediaMax and causes similar security problems due to the violation of the same security requirements. The active protection software is installed together with a second program - a 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, which begin with that prefix, become invisible to normal programs and administration tools. The rootkit is a kernel-level driver named *sysaries* that is set to automatically load early in the boot process. When the rootkit starts, it hooks several Windows system API functions: NtCreateFile, NtEnumerateKey, NtOpenKey, NtQueryDirectoryFile NtQuerySystemInformation by modifying the system service dispatch table (the kernel's *KeServiceDescriptorTable* structure) and changes them to point to functions within the rootkit, so that calls are handled by the rootkit rather than the original kernel function. The rootkit calls the real kernel function with the same parameters, filters and results before returning them to the application.

But besides the security hole it opens, the rootkit has at least two harmful side effects. First, its design, based on well-known malware methods, tends to trigger security alarms. Second, it facilitates another privilege escalation attack which allows an unprivileged application to crash the system. If a malware 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. Users experiencing system instability due to these rootkit bugs would have great difficulty to diagnose the problem, since they likely would be unaware of the rootkit's presence.

Lessons Learned

As we have seen above, although the main goal of the DRM technology is to provide security, its practical implementations often lead to serious security problems due partly to the fact that DRM technologies are based

on secrecy and work with lack of transparency. Besides by improving strategy and technology, DRM security could be increased by decreasing attacker's motivation. Currently most DRM implementations prevent end users from playing their purchased music on any portable digital music players other than the manufacturer's ones. As a result, some end users with technical know-how are tempted to exploit any potential weakness and to break the DRM protection. Another problem is the interoperability between different DRM technologies. If any portable player could play a content encoded in any DRM format this would eliminate the need for paying customers to circumvent DRM technologies in order to play their music anyway. But the principal issue in DRM security is that although nowadays there are many practical DRM implementations there are no established security standards for protecting sensitive rights information, authenticating entities in transactions and providing data integrity. Maybe it will not be possible to create one standard, but several encompassing ones could be accomplished in the near future if the existing DRM vendors conduct a dialogue about the direction of DRM evolution and reach some agreement about the principal problems in today's DRM technologies: interoperability, use of DRM integrated hardware instead of active protection and smart design. Of course, considering security requirements during the process of establishing DRM standards is important and will ensure that end systems will be:

- specified with security in mind not as an afterthought
- trusted by the two parties – those that are interested in the protection of the content and those who use it
- relatively easy to implement
- efficient to operate and maintain
- easy to use from the end user's perspective throughout the lifetime of the protected content
- scalable
- interoperable
- adaptable to permit different security paradigms
- renewable from a security perspective

Conclusions

Like most computer systems and software, DRM technologies have their own inherent security requirements. Defining, understanding and applying these requirements are a fundamental part of establishing effective, interoperable DRM standards, technologies that implement these standards and systems that incorporate these technologies. In this paper we described and discussed the applicability of our security requirements by comparing them to four existing DRM systems. We explained the exploited weaknesses of Microsoft Media DRM, Apple's iTunes, SunnComm Technologies's MediaMax DRM and First4Internet's XCP DRM through the violation of specific security requirements and showed that they could have been identified in an early state and avoided. The generic nature of the requirements implies that they are applicable to most DRM technologies and therefore should be met by any matching systems. Further developments can lead to establishing a theoretical basis that simulates DRM functioning and evaluates how well it conforms to the principal security requirements.

References

[1] M. Nickolova, E. Nickolov, Conceptual Model and Security Requirements for DRM Techniques Used for E-Learning Objects Protection. International Journal «Information Theories and Applications», to be published

[2] T. Hauser and C. Wenz, DRM under attack: weaknesses in existing systems. Digital Rights Management: Technological, Economic, Legal and Political Aspects, volume 2770, pages 206 – 223, Springer-Verlag GmbH, November 2003.

[3] International Organization for Standardization (ISO). ISO/IEC TR 13818-5:1997/Amd 1:1999 - Advanced Audio Coding (AAC).

[4] http://www.computing.co.uk/computeractive/news/2012404/fairplay-crackers-core-itunes

[5] http://nanocrew.net/about/

Authors' Information

Maria Nickolova – National Laboratory of Computer Virology; BAS, Acad.G.Bonthev St., bl.8, Sofia 1113, Bulgaria; e-mail: maria@nlcv.bas.bg

Eugene Nickolov – National Laboratory of Computer Virology; BAS, Acad.G.Bonthev St., bl.8, Sofia 1113, Bulgaria; e-mail: eugene@nlcv.bas.bg