Piracy prevention methods in software business

University of Oulu
Department of Information
Processing Science
Johanna Korhonen
Bachelor's Thesis
18.5.2015

Abstract

There are various forms of piracy in software business, and many prevention techniques have been developed against them. Forms of software piracy are, for example, cracks and serials, softlifting and hard disk loading, internet piracy and software counterfeiting, mischanneling, reverse engineering, and tampering. There are various prevention methods that target these types of piracy, although all of these methods have been broken. The piracy prevention measures can be divided into ethical, legal, and technical measures. Technical measures include measures like obfuscation and tamper-proofing, for example. However, relying on a single method does not provide complete protection from attacks against intellectual property, so companies wishing to protect their product should consider combining multiple methods of piracy prevention.

Keywords: software piracy prevention, anti-piracy, intellectual property, copy protection

Table of Contents

Abstract 2
Table of Contents 3
1. Introduction 4
2. Types of piracy 5
   2.1 Cracks and serials 5
   2.2 Softlifting and hard disk loading 6
   2.3 Internet piracy and software counterfeiting 6
   2.4 Mischanneling 6
   2.5 Reverse engineering 7
   2.6 Tampering 7
   2.7 Hardware techniques for breaking copy protection 7
   2.8 Summary of addressed piracy types 8
3. Prevention methods in general 9
   3.1 Ethical measures 9
   3.2 Legal measures 10
   3.3 Summary of ethical and legal measures 11
4. Technical measures 12
   4.1 Software-based measures 12
      4.1.1 Obfuscation 12
      4.1.2 Tamper-proofing 13
      4.1.3 Watermarking 14
   4.2 Hardware-based measures 15
      4.2.1 CD-ROM 15
      4.2.2 Dongle 15
      4.2.3 Expansion card 16
   4.3 Summary of technical measures 17
5. Conclusions 18
References 20

1. Introduction

The companies operating in the field of software business constantly face the problems caused by software piracy, one of which is the loss of revenue. In order to combat software piracy many companies apply different measures against unauthorized software distribution. There are many forms of piracy, and it is important to learn about them in order to combat them.

There has been quite a lot of research on the methods of piracy prevention, especially on the technical methods. However, studies suggest that technical prevention methods are not enough when combating piracy since all of the methods have been broken.

The methods against software piracy can be divided into two categories: preventive methods and deterrents. The deterrents are used to hinder users from pirating software by threatening with legal consequences while the preventive methods aim to make pirating so difficult that the perpetrator depletes their resources and is forced to withdraw. (Gopal, Ram D.Sanders, 1997) This thesis concentrates on the preventive methods, which can then be divided into three classes: ethical, legal, and technical measures. The aim of ethical measures is to make piracy seem morally unappealing in the eyes of consumers while the legal measures attempt to prevent piracy by influencing consumers so that they avoid piracy in fear of legal consequences. The point of the technical measures is either to increase the difficulty of duplicating software or to increase the chances for the perpetrator of getting caught. (Cronin, 2002) The focus of this thesis is on the technical viewpoint; however, the other two will also be addressed briefly.

The main question is what kind of methods software companies have used against piracy, and, as mentioned earlier, the focus is on the technical measures. The different preventive technical measures against piracy are defined and their features and applicability will be discussed.

However, despite the diversity of technical measures against illegal duplicating of software the protection methods fail in protecting software. The nature of these protection measures is static, and none of these methods provide adequate protection because they all have been breached. (Anckaert, De Sutter, & De Bosschere, 2004) In addition, the software protection measures may discourage consumers from buying the legitimate product, and thus a new approach to piracy is needed (Sudler, 2013).

In the following chapter we will address several forms of piracy briefly. Chapter 3 is dedicated for ethical and legal measures of piracy prevention, whereas chapter 4 focuses solely on the technical measures. Conclusions are presented in chapter 5.

This research was conducted as a literature review. The literature used in this thesis was accessed via academic databases and Oulu University library, and the academic databases used were Google Scholar, ACM Digital Library, ScienceDirect, IEEE Xplore, SpringerLink and Web of Science.

2. Types of piracy

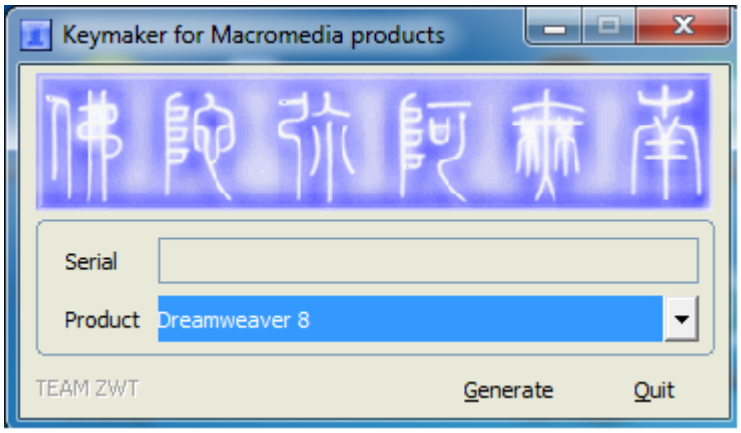
In order to develop new methods against software piracy it is important to consider the different types of piracy. Software piracy results in lost revenue worth billions of dollars annually (Fu, Richard, & Chen, 2006). The Software Producers Association estimated its losses to be 11.4 billion dollars in 1997. In addition to the lost revenue there are also other disadvantages caused by software piracy: for example, in order to compensate for their losses software companies may increase their prices. Revenue losses may lead to lost jobs, and there is also a possibility of losing both domestic and foreign investments. (Hamade, 2006)

Some of these forms of software piracy are cracks and serials, softlifting and hard disk loading, internet piracy and software counterfeiting, and mischanneling (Anckaert et al., 2004). Malicious reverse engineering and software tampering can also be regarded as attacks against intellectual property (Collberg & Thomborson, 2002). In addition, there are also hardware techniques which aim to undo the copy protection of the software (Shi, Lee, Lu, & Zhang, 2004). There are also other forms of attacks against intellectual property, but in this thesis we will focus on the aforementioned ones.

2.1 Cracks and serials

Cracks and serials are a very widespread form of piracy, and in this piracy type a legal evaluation version is obtained by a entering a license code or applying a patch which undoes the copy protection. This type of piracy is popular because the distribution of license codes or patches is much more effortless than that of complete software. (Anckaert et al., 2004)

After breaking the copy protection mechanisms the attacker often creates modified versions of the software, and in these cracked versions the copy protection is disabled. Cracked software is also often illegally distributed to the public. (Chang, H. & Atallah, M.J., 2002)



Picture 1. Screenshot of a key generator software developed by pirates. The software generates valid CD keys for circumventing software copy protections.

One of the main factors that contribute to the widespread nature of cracking lies in the simplicity of inspection and modification enabled by current software debugging and editing tools. With the help of hex editors and sophisticated debuggers crackers can target specific parts of the program and apply their modifications. However, tamper-proofing mechanisms attempt to counter this form of piracy. (Chang & Atallah, 2002)

2.2 Softlifting and hard disk loading

The act of obtaining copyrighted software legally and installing it on more computers than allowed is called softlifting while hard disk loading refers to the act in which software is illegally installed into hardware. Both these aforementioned forms include installing the software on more computers than the license allows. (Anckaert et al., 2004)

Softlifting can be called a software equivalent of shoplifting, and an example of this type of piracy would be bringing home a copy of software intended for work use (Koen & Im, 1997). Softlifting is usually called software piracy by individuals, and it is widespread consumer behavior (Simpson, Banerjee, & Simpson, 1994). An example of hard disk loading could be a case where an electronics company installs illegal copies onto the hard disks of computers in order to further their business (Hamade, 2006).

2.3 Internet piracy and software counterfeiting

Internet piracy is a form of piracy in which unauthorized copies of copyrighted software are distributed in an electronic form whereas software counterfeiting depicts a term in which copyrighted software is illegally duplicated and distributed while making the unauthorized copies appear genuine. (Anckaert et al., 2004)

Counterfeit replicas are often altered by, for example, inserting malicious code or removing anti-tamper protections. Software counterfeiting also often extends to labels, brochures, authenticity certificates or any other documents that prove the authenticity of the purchased product. (Goertzel, 2011)

Software counterfeiting raises many concerns. Firstly, suppliers do not have any obligation to provide support in problem situations for customers who, wittingly or unwittingly, purchased a counterfeited product. In addition, counterfeit products may also tarnish the reputation of a legitimate vendor, possibly leading to loss of brand integrity. (Goertzel, 2011)

2.4 Mischanneling

An example of mischanneling would be a situation where, for example, an academic license is used for commercial gain, so the software is utilized for something that it is clearly not intended. Currently there are no means of protection against this form of piracy, and this is something that software cannot be protected from. (Anckaert et al., 2004)

2.5 Reverse engineering

A malicious reverse engineer aims to understand the hidden methodology of the software without negotiating for a license (Collberg & Thomborson, 2002). Reverse engineering is often countered with obfuscation (Danger, Guilley, & Praden, 2014). This type of piracy can also serve multiple purposes, which range from assisting competitors to breaking software protection mechanisms (Fu et al., 2006).

There are a number of tools which can be utilized in reverse engineering attacks. Commonly used tools in reverse engineering are debuggers, disassemblers, and decompilers. Debuggers enable the dynamic analysis of the program during its execution whereas disassemblers generate assembler code from executables. The function of decompilers is to recreate high-level source code which then corresponds to the original code of the target software. There are also several specialized tools for reverse engineering, and these tools can be used for monitoring system calls executed by the program, for example. (Fu et al., 2006)

It is practically impossible to completely prevent reverse engineering on current platforms, and thus all the measures countering this form of piracy are for merely increasing the amount of work needed for a computer program to be successfully reverse engineered (Fu et al., 2006).

2.6 Tampering

Tampering is identified as another form of attack on intellectual property. Tampering is often combined with other forms of piracy, like reverse engineering, for example. However, tampering often occurs independently of other forms of attack, and an example of this would be an attacker manipulating the software for financial gain. (Collberg & Thomborson, 2002) Tampering can also be defined as unwanted software modification, which includes both automated changes made by viruses and manual code changes made by end users (Naumovich & Memon, 2003).

Tampering can be, for example, extracting or modifying information encrypted in the software. Accessing and tampering such information may lead to significant financial losses for the owner of the intellectual property. (Collberg & Thomborson, 2002)

2.7 Hardware techniques for breaking copy protection

There are three typical hardware-based techniques for breaking copy protection: software execution trace analysis, mod-chip spoofing devices, and machine emulator (Shi et al., 2004).

In trace analysis the attacker can collect information on target software via logging and analyzing software traces. A lot of programming primitives are very predictable even in their encrypted form; for example, conditional branches are easily identifiable and traceable even after encryption. In addition, technologically skilled users are often able

to build custom tracing devices from cheap and readily available components. (Shi et al., 2004)

Another form of hardware techniques for breaking copy protection are mod-chip spoofing devices. Mod-chips can be used for recording and replaying memory and bus transactions. In addition, they can be used for hijacking the signal of another device. (Shi et al., 2004)

The third form of hardware-based methods for undoing copy protection is using emulators. If the software can be executed on a machine emulator without authorization, the software copy protection can be deemed broken. It is very difficult to fight against this type of undoing copy protection without relying on encryption. However, it is possible to prevent the machine emulator from breaking protection on software rights by protecting software confidentiality. (Shi et al., 2004)

2.8 Summary of addressed piracy types

All in all, there are various forms of piracy to be countered. Software piracy causes all sorts of problems to the companies operating in the field of software business, the most notable of them being the loss of revenue which then in itself contributes to several other problems.

In this chapter we addressed quite a few software piracy types: cracks and serials, softlifting and hard disk loading, internet piracy and software counterfeiting, mischanneling, reverse engineering, and tampering. In addition, we also addressed the hardware-based attacks against copy protection. All of these piracy types are very different, and software companies have attempted to tackle these piracy types through various preventive methods, which will be discussed in the following chapters.

3. Prevention methods in general

The methods against piracy can be divided into three classes: ethical measures, legal measures, and technical measures. Each of these classes has their own means against piracy which will be briefly described in following subchapters. The focus is on the technical measures which will be addressed more thoroughly in the following chapter, and in this section we will address only the ethical and legal measures.

3.1 Ethical measures

The aim of the ethical measures is to make software piracy appear morally unappealing (Cronin, 2002). In this case deterrents like shame, formal sanctions or moral beliefs have been utilized in the fight against software piracy. Shame is defined as a feeling of guilt or embarrassment induced if others were to know of one's socially undesirable actions, while formal sanctions are explicit penalties one will receive in the case of misconduct. Moral beliefs appeal to the morality of one's actions: potential offenders may refrain from acts of piracy because they view the deed as morally wrong. However, only shame and moral beliefs seem to have a significant effect on deterring piracy. (Siponen, Vance, & Willison, 2012) Studies have demonstrated that ethics has an impact on individuals' tendency to engage in acts of software piracy and thus they should not be disregarded (Ram & Lawrence, 1997).

The era of internet has led to new forms of ethical breach in the field of ICT. Technologies like P2P enable illegal distribution of music, movies, and software alike. (Rekha & Pillai, 2014) Studies have shown that ethics play an important part in decreasing the intent to pirate while increasing the intent to purchase (Thongmak, 2015). However, relying on ethical awareness is not sufficient in itself for preventing unethical behavior (Rekha & Pillai, 2014).

Nevertheless, in order to combat software piracy having an understanding of its ethical implications is important. Rekha & Pillai (2014) note that in addition to understanding ethical implications of software piracy the users need to be ethically empowered so that they practice what they believe.

Sometimes ethical and technical measures can be combined. A fairly recent example of this is the case of Oxford Deluxe dictionary application. When the application detected that user had a pirated version on their mobile device, the application, which had asked for Twitter access upon installation, then proceeded to automatically post accusations shaming users of pirating software on their own Twitter accounts. The automatically posted tweets said, "How about we all stop using pirated iOS apps? I promise to stop. I really will." and used the hashtag #softwarepirateconfession. However, there was a backlash when the application auto-posted accusing tweets even on behalf of users who had purchased a legitimate copy of the application because the checking for legitimate users failed. (Ødegård, 2012)

Another very similar case of combining ethical and technical measures is the case of Tweetbot. Tweetbot also utilized Twitter in shaming users who had pirated the software, but instead of auto-posting tweets the application merely auto-filled the tweet composition box with a message accusing the users of pirating, giving the users the option to delete the message instead of posting it for all to see. Tapbot, the developer of Tweetbot, had a significant motivation behind including the deterrent: third party Twitter clients can only have a limited number of users, so users who had pirated the application both took away money from the developer and removed potential future customers by filling up slots. (Plafke, 2013)

3.2 Legal measures

The legal measures attempt to prevent piracy by creating fear of consequences of being caught in the act of piracy. Legal measures act as a deterrent by threatening possible offenders with legal sanctions. (Cronin, 2002) A lot of countries have attempted to alleviate the problem of software piracy via copyright, patent contract, and trade secret legislation extensions. Software has also been recognized as a type of literary and artistic work that is subject to intellectual property right protection. (Liu, Cheng, Tang, & Eryarsoy, 2011) Software piracy is most rampant in areas like Eastern Europe, Latin America, and Asia-Pacific region, and it is estimated that the percentage of in-use pirated copies exceed 90% in countries like Russia and China (Moores & Dhillon, 2000).

Legal deterrents are beyond the control of software publishers, and instead they rely on consumers' awareness of copyright laws and the enforcement of these laws by the government. (Liu et al., 2011) In addition, legislation may only guarantee protection in jurisdictions where laws regarding intellectual property rights are in effect. However, there are international treaties and conventions which compel member countries to enforce minimum standards of intellectual property protection. For example, World Intellectual Property Organization, which administers these treaties regarding intellectual property, has 180 member countries. (Hamade, 2006)

However, it is questionable whether the legal measures are effective in the fight against piracy. In fact, there is no evidence that increased awareness of piracy as a crime and the fear of legal sanctions have slowed down the spreading of piracy. (Bono, Rubin, Stubblefield, & Green, 2006)

In addition, software companies should never rely solely on legal measures. Security through legality is something that system designers should not depend on and instead they should focus on other preventive measures. (Bono et al., 2006)

3.3 Summary of ethical and legal measures

Ethical and legal measures are widely used in the battle against software piracy. Ethical measures attempt to make software piracy seem unethical and appeal to the sense of moral. Legal measures, however, attempt piracy prevention by the threat of legal sanctions in case of a software pirate being caught.

Ethical measures have been shown to have an impact on users' willingness to pirate software, and thus they should not be ignored. Legal measures on their behalf don't seem to have an impact on slowing down the spreading of software piracy. However, both of these measures can be combined with the technical measures, which will be addressed in the next chapter.

4. Technical measures

There are software-based and hardware-based technical measures against software piracy. In addition, these measures can be distinguished into token-based and token-less copy protection techniques: the difference between these categories is that token-based measures utilize either an intangible or physical token in order to protect the software from unauthorized duplicating. (Djekic & Loebbecke, 2005) Another way to distinguish these technical means is whether they attempt to prevent the duplication of software or increase the likelihood of getting caught (Cronin, 2002).

It is important to notice that these prevention methods can also be applied program components, such as class libraries, in addition to complete programs. (Naumovich & Memon, 2003)

4.1 Software-based measures

The copy protection techniques based on software add specific functions to the application software. Software-based measures include measures like software tokens, watermarking, obfuscation, and encryption. The latter three measures are embedded in the source code of the application and they do not interfere with users. (Djekic & Loebbecke, 2007) However, it is debatable whether watermarking, obfuscation and encryption count as preventive measures because they do not directly affect the ability to copy software (Djekic & Loebbecke, 2007). While the complete protection of software might be an unattainable goal, it is possible to achieve some degree of protection with methods like the aforementioned software watermarking, tamper-proofing and obfuscation (Collberg & Thomborson, 2002).

4.1.1 Obfuscation

Obfuscation is an anti-piracy method which attempts to transform the target program into a more complex version which is harder to reverse engineer (Collberg & Thomborson, 2002). Obfuscation aims to increase the difficulty of software duplication. There are various techniques for obfuscation, and programs known as obfuscators transform human-readable code into obfuscated code, thus increasing the difficulty of reverse engineering and providing security (Yuvaraj, Venkatraj, & Yogesh, 2011). Code obfuscations aim to obscure the purpose of the code without changing its operation, and code obfuscations can be applied to source code, byte code, or binary executable code (Goertzel, 2011). Obfuscation aims to disguise other software-based protection mechanisms inserted in the program (Naumovich & Memon, 2003).

Obfuscation can be divided into four different categories: lexical obfuscation, data obfuscation, control obfuscation and layout obfuscation. In lexical obfuscation program identifiers are renamed without giving away their meaning whereas in data obfuscation the data structures and their purposes are hidden. Control obfuscation aims to obscure the workflow while layout obfuscation obscures the logic inherent in breaking a program into procedures. (Naumovich & Memon, 2003; Xu & Xiang, 2012)

However, these types of obfuscation all have their issues. For example, lexical obfuscations alone are insufficient because a determined attacker can deduct the meaning of the program identifiers based on context whereas control obfuscation should always be resistant to automated analyses, which are present in many compilers. (Naumovich & Memon, 2003)

4.1.2 Tamper-proofing

Tamper-proofing causes the program to malfunction when it detects that the program has been modified. In order to prevent tampering attacks, tamper-proofing code should be added to the program. The purpose of tamper-proofing code is to detect whether the program has been altered in any way and cause the program to fail when tampering is obvious. (Collberg & Thomborson, 2002) Tamper-proofing can also either be built into the program or be provided externally. Tamper-proofing can be provided by operating systems, for instance. (Naumovich & Memon, 2003)

There are many methods for preventing software tampering. Some of the tamper-proofing mechanisms are checksums, guards, assertion checking, software aging, and cryptographic techniques. (Naumovich & Memon, 2003)

Checksums are regarded as quite a straightforward tamper-proofing mechanism. One method is to examine the program before running it and compare it with the original program using checksums. The problem with checksums, however, is that their nature may be difficult to conceal, and in case of detection the attacker can easily remove them. (Naumovich & Memon, 2003)

Guards are small program units which work together in a network to protect each other, and there are multiple guards in one network. The strength of guards lies in their numbers, because guards can be disabled only by removing all of them. In case of attack where the attacker manages to identify and remove a number of guards the remaining guards detect this form of tampering and prevent the program from functioning. (Naumovich & Memon, 2003)

The idea of assertion checking as a tamper-proofing mechanism is to check intermediate program results. An example of this would be checking the values of variables at specific points, and if the variable has been assigned a value other than what it is supposed to have, the program fails. This, however, is not a very reliable tamper-proofing method because unexpected values may be a result of a bug instead of tampering. In addition, a large number of assertion checks may slow down the program significantly, and this mechanism is difficult to automate. There is also a possibility that even after tampering the program may not produce invalid intermediate results. (Naumovich & Memon, 2003)

Another tamper-proofing mechanism is software aging, which relies on periodic software updates. Vendors can force the illegal resellers of their program to provide their own updates and thus increasing the likelihood of getting caught by offering frequent updates that lessen the utility of older versions of software. Software aging, however, is only useful on applications that rely on format-specific data, like Microsoft Word. (Naumovich & Memon, 2003)

Cryptographic techniques are another method of tamper-proofing, and in this method the software is protected from modifying by blocking the attacker from seeing the source code of the software. This option might seem appealing; however, this method has some technical obstacles. For example, distributing cryptographic keys might prove to be difficult, and the software must be decrypted before running, which may lead to performance issues. (Naumovich & Memon, 2003)

4.1.3 Watermarking

In watermarking a copyright notice is embedded in the software code, enabling the owners of the software assert their intellectual property rights (Collberg & Thomborson, 2002). The copyright notice can be extracted from the program to identify the owner of the copyright or an authentic user of the program (Ma et al., 2015).

Software watermarks can be divided into two categories: static and dynamic. Static watermarks are stored in the application executable while dynamic ones are constructed during runtime and stored in the dynamic state of the program (Collberg & Thomborson, 2002). Dynamic watermarking is considered a more reliable and secure solution because it retrieves the copyright notice by running along a specific path of the watermarked program while examining its behavior (Ma et al., 2015).

However, there are some limitations with existing techniques which affect the use of dynamic watermarking. For example, the current solutions introduce some specific data structures and instruction patterns which can be targeted by possible attackers, making the watermark vulnerable. Another factor to take into account in dynamic watermarking is that the code inserted by dynamic watermarking is often quite independent and separate from other parts of the program, which makes the use of watermark more obvious. Finally, because an external extractor is needed for retrieving the watermark from the program's execution recordings, the watermarked program might need to leave some information for the extractor for finding the hidden watermark. This is problematic because the information left by the program could potentially be exploited by attackers. (Ma et al., 2015)

There is also a technique related to watermarking called fingerprinting. In this case a watermark that varies from one copy of software to another is created. The main distinction between general watermarking and fingerprinting is that watermark can only identify the authors of the software while fingerprint can identify the system on which the program was run. (Naumovich & Memon, 2003)

4.2 Hardware-based measures

The measures based on hardware use specific hardware tokens which are required for successfully installing or running the software These tokens include techniques like serial numbers, CD-ROM, Challenge-Response, Dongle, and expansion card (Petar Djekic & Loebbecke, 2007). Hardware security tokens are typically portable and small in size while having little to no user interface (Lu & Ali, 2008). In order to enable partial or complete functionality the software must retrieve the information stored on the hardware component, thus validating the use of the program (Naumovich & Memon, 2003).

These tokens may be intangible or physical, and they protect the application software from illegal copying. The applications check for the presence of the token and often they will refuse to install or run if the token is not present. (P. Djekic & Loebbecke, 2005) However, in order to achieve overall security, the communication between the hardware security token and the host computer must be secured when connecting the token to the host (Lu & Ali, 2008).

4.2.1 CD-ROM

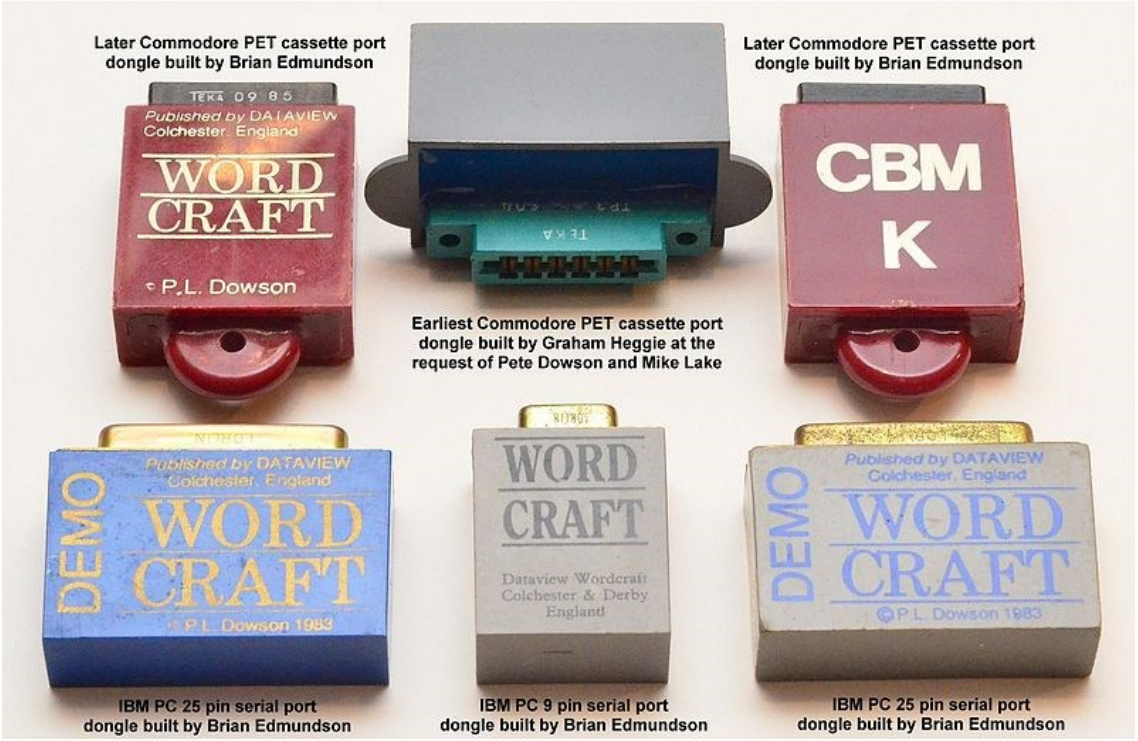
In the case of CD-ROM, one or more CDs act as a hardware token. To protect the hardware token from duplication it is physically tampered, like in the case of floppy disks. If the attackers wish to circumvent this form of copy protection, they need either to modify the protection software or duplicate the token using a CD burner. (Djekic & Loebbecke, 2007)

4.2.2 Dongle

Dongles are typically used for protecting high-end low-quantity software. They are shipped to customers alongside the software they aim to protect, and the software won't install or run unless it detects the particular dongle it was shipped with. (Piazzalunga, Salvaneschi, Balducci, Jacomuzzi, & Moroncelli, 2007) Dongles are also referred to as hardware keys, and this is a hardware-based measure which utilizes a small stick which can be plugged into an appropriate port, like a USB port, for example. If the user wants to run the software, the token needs to be connected to the computer. Breaking this type of protection requires modifying the source code in order to skip token checks. (Djekic & Loebbecke, 2007)

The dongle various logical resources including a unique serial number, a unique independent software vendor identification number, which is assigned to each dongle customer by the vendor; an access password for unlocking the dongle's functionality, a persistent memory in which the software can write, and a symmetric encryption engine and storage for symmetric keys. However, it is important to note that not all of these aforementioned logical resources are always present in a dongle. (Piazzalunga et al., 2007)

It is also important to note that typically the dongle in itself isn't the point of attack for hackers. Instead hackers may utilize weaknesses in the interaction between the software and the dongle API, which is often the weakest link in protection. (Piazzalunga et al., 2007)



Picture 2. Early protection dongles. (Trusley, 2011)

4.2.3 Expansion card

Expansion card refers to the type of protection where the hardware token is integrated into the computer as a PCI-card. This type of hardware token often also provides the core functionalities of the software. This type of hardware-based protection is the strongest because duplicating the token is very difficult and modifying the software is not enough for breaking this copy protection mechanism. (Djekic & Loebbecke, 2007)

There is also a card type known as smart cards, and they embody the user license. Smart cards are portable, secure, and tamper-resistant Smart cards have been quite unpopular with users; the primary objection has been that the copy protection mechanisms for different software packages have a tendency to interfere with each other and cause problems, even if the design of each individual product has been done well. There is also a problem called smart card juggling: the users may have to repeatedly insert different cards into the smart card reader in order to prove the presence of a hardware token, which may feel annoying to users. (Aura, T., & Gollmann, 1999; Lu & Ali, 2008)

Djekic & Loebbecke (2007) concluded in their research that these aforementioned types of hardware-based protection do not lead to less piracy. In addition, all hardware-based protection techniques tend to lack user-friendliness, which then makes it more difficult for users to use them for protecting the software with these mechanisms (Naumovich & Memon, 2003).

4.3 Summary of technical measures

All in all, there are many ways for categorizing technical piracy prevention methods. In this thesis we used the viewpoint of software-based measures versus hardware-based measures. Software-based measures include techniques like obfuscation, tamper-proofing, and watermarking, while the hardware-based measures include tokens like CD-ROM and Dongle.

Obfuscation attempts to make the software more difficult to reverse engineer, while tamper-proofing prevents software tampering. Watermarking attempts to prevent the unauthorized duplication of software.

The idea of hardware-based prevention mechanisms is that the software cannot be installed or run without the appropriate hardware being connected to the computer. CD-ROM, Dongle, and expansion cards are all examples of hardware-based measures. These measures, however, do not lead to less piracy and they tend lack user-friendliness.

As already stated before, none of the technical measures are alone sufficient at protecting software from piracy, and thus they should be combined with ethical and legal measures.

5. Conclusions

Piracy is by no means a new phenomenon. For example, music, motion pictures and software have experienced unauthorized copying before the industries even realized its impact. New technology can be regarded as a catalyst for rising piracy because now people can access high quality duplicates easier than ever. For example, the internet enables rapid, international sharing of digital content. (Sudler, 2013)

We have addressed multiple methods of software piracy prevention, which range from ethical measures to technical measures, with the focus being on the latter one. However, it is important to note that none of the addressed technical measures provide adequate copy protection since they all have been broken (Anckaert et al., 2004). The effectiveness of technical piracy prevention methods is limited because they are only effective until a hacker manages to break them successfully (Liu et al., 2011).

Since relying on a single technique does not provide sufficient copy protection, multiple technologies should be combined. In addition to the technological solutions software companies should consider designing new business models which should be utilized in unison with the technical measures. Software piracy should not be ignored because it has had a massive impact on traditional supply chains. (Sudler, 2013)

For example, Apple has demonstrated that the combination of new technology and new business models may actually help combat piracy by leveraging it. Fan base, product loyalty and sales may increase through the leveraging of piracy, and the combination of content availability and innovative business models gives software companies back their power over their supply chains.(Sudler, 2013)

Gabe Newell, the co-founder and managing director of video game development and online distribution company Valve Corporation, stated when asked about his views regarding digital rights management that almost always piracy is a service problem instead of a pricing problem. He stretches the importance of availability of product when combating piracy: "*For example, if a pirate offers a product anywhere in the world, 24 x 7, purchasable from the convenience of your personal computer, and the legal provider says the product is region-locked, will come to your country 3 months after the US release, and can only be purchased at a brick and mortar store, then the pirate's service is more valuable"*. Valve's solution has been to create greater service value than pirates, which has proven to be a successful approach for Valve, and Newell states that "piracy is basically a non-issue for our company". (Tufnell, 2011)

Indeed, convenience and availability when it comes to services seem to be major factors in combating piracy. For example, Netflix, provider of on-demand streaming media, acquires content rights based on piracy rates. They compete in high-piracy areas by providing popular content and setting prices according to local piracy rates. Reed Hastings, the CEO of Netflix, has stated that the introduction of Netflix in Canada halved the BitTorrent traffic in the country. (Price, 2015)

The music industry has probably been the industry that has been hit the heaviest by piracy. However, the introduction of Spotify, which focuses on streaming of music, has significantly cut piracy. Spotify has offers both free listening and subscription services for listening to music. Daniel Ek, the CEO of Spotify, has stated that Spotify actively fights against piracy and that they aim to recover money for artists and music business which have been deeply impacted by piracy. (Roddy, 2014)

All in all, software businesses should innovate new business models which provide goods and services conveniently to consumers in order to fight piracy. Technical piracy prevention methods are only effective until the first successful hacker, and ethical and legal measures seem to have only a minor effect in reducing piracy. In addition to creating new business models software companies should consider using combining multiple piracy prevention methods. The aforementioned examples of companies fighting piracy demonstrate that availability and convenience of services are key factors in the fight against piracy.

References

Anckaert, B., De Sutter, B., & De Bosschere, K. (2004). Software piracy prevention through diversity. In *Proceedings of the 4th ACM workshop on Digital rights management - DRM '04* (p. 63). New York, New York, USA: ACM Press. http://doi.org/10.1145/1029146.1029157

Aura, T., & Gollmann, D. (1999). Software license management with smart cards. (p. 9). Retrieved May 12, 2015, from https://www.usenix.org/legacy/events/smartcard99/full_papers/aura/aura.pdf

Bono, S., Rubin, A., Stubblefield, A., & Green, M. (2006). Security through legality. *Communications of the ACM*, *49*(6), 41. http://doi.org/10.1145/1132469.1132499

Chang, H., & Atallah, M. J. (2002). Protecting software code by guards. In *Security and privacy in digital rights management* (pp. 160-175). Springer Berlin Heidelberg.

Collberg, C. S., & Thomborson, C. (2002). Watermarking, tamper-proofing, and obfuscation - tools for software protection. *IEEE Transactions on Software Engineering*, *28*(8), 735–746. http://doi.org/10.1109/TSE.2002.1027797

Cronin, G. (2002). A Taxonomy of Methods for Software Piracy Prevention. Retrieved April 26, 2015, from http://www.veryquick.org/writing/piracytaxonomy.pdf

Danger, J.-L., Guilley, S., & Praden, F. (2014). Hardware-enforced Protection against Software Reverse-Engineering based on an Instruction Set Encoding. In *Proceedings of ACM SIGPLAN on Program Protection and Reverse Engineering Workshop 2014 - PPREW'14* (pp. 1–11). New York, New York, USA: ACM Press. http://doi.org/10.1145/2556464.2556469

Djekic, P., & Loebbecke, C. (2005). Software Piracy Prevention through Digital Rights Management Systems. *Seventh IEEE International Conference on E-Commerce Technology (CEC'05)*, 504–507. http://doi.org/10.1109/ICECT.2005.86

Djekic, P., & Loebbecke, C. (2007). Preventing application software piracy: An empirical investigation of technical copy protections. *The Journal of Strategic Information Systems*, *16*(2), 173–186. http://doi.org/10.1016/j.jsis.2007.05.005

Fu, B., Richard, G., & Chen, Y. (2006). Some new approaches for preventing software tampering. In *Proceedings of the 44th annual southeast regional conference on - ACM-SE 44* (p. 655). New York, New York, USA: ACM Press. http://doi.org/10.1145/1185448.1185592

Goertzel, K. M. (2011). Protecting Software Intellectual Property Against Counterfeiting and Piracy. Retrieved April 26, 2015, from http://static1.1.sqspcdn.com/static/f/702523/14121190/1315886341690/201109-Goertzel.pdf?token=YtVwWq9EtA559HHmosPjqePQ1bQ%3D

Gopal, R. D., & Sanders, G. L. (1997). Preventive and deterrent controls for software piracy. *Journal of Management Information Systems*, 29-47.

Hamade, S. N. (2006). The Legal and Political Aspects of Software Piracy in the Arab World. In *Third International Conference on Information Technology: New Generations (ITNG'06)* (pp. 137–142). IEEE. http://doi.org/10.1109/ITNG.2006.129

Koen, C. M., & Im, J. H. (1997). Software piracy and its legal implications. *Information & Management*, *31*(5), 265–272. http://doi.org/10.1016/S0378-7206(96)01090-7

Liu, Y., Cheng, H. K., Tang, Q. C., & Eryarsoy, E. (2011). Optimal software pricing in the presence of piracy and word-of-mouth effect. *Decision Support Systems*, *51*(1), 99–107. http://doi.org/10.1016/j.dss.2010.11.032

Lu, H. K., & Ali, A. (2008). Communication Security between a Computer and a Hardware Token. In *Third International Conference on Systems (icons 2008)* (pp. 220–225). IEEE. http://doi.org/10.1109/ICONS.2008.36

Ma, H., Lu, K., Ma, X., Zhang, H., Jia, C., & Gao, D. (2015). Software Watermarking using Return-Oriented Programming. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security - ASIA CCS '15* (pp. 369–380). New York, New York, USA: ACM Press. http://doi.org/10.1145/2714576.2714582

Moores, T., & Dhillon, G. (2000). Software piracy: a view from Hong Kong. *Communications of the ACM*, *43*(12), 88–93. http://doi.org/10.1145/355112.355129

Naumovich, G., & Memon, N. (2003). Cover feature - Preventing piracy, reverse engineering, and tampering. *Computer*, *36*(7), 64–71. http://doi.org/10.1109/MC.2003.1212692

Piazzalunga, U., Salvaneschi, P., Balducci, F., Jacomuzzi, P., & Moroncelli, C. (2007). Security Strength Measurement for Dongle-Protected Software. *IEEE Security & Privacy Magazine*, *5*(6), 32–40. http://doi.org/10.1109/MSP.2007.176

Plafke, J. (2013). If you pirate Tweetbot, it will make you tweet that you pirated it. Retrieved from http://www.geek.com/apps/if-you-pirate-tweetbot-it-will-make-you-tweet-that-you-pirated-it-1541405/

Price, R. (2015). Netflix has an ingenious, piracy-combating way to set its international pricing. Retrieved May 13, 2015, from http://uk.businessinsider.com/netflix-piracy-international-pricing-streaming-earnings-2015-4?r=US

Rekha, A. G., & Pillai, R. R. (2014). Piracy in the digital age: Is ethical awareness turning into action? In *2014 IEEE International Symposium on Ethics in Science, Technology and Engineering* (pp. 1–4). IEEE. http://doi.org/10.1109/ETHICS.2014.6893456

Roddy, M. (2014). Spotify says fights piracy, has paid $2 billion to artists, industry. Retrieved May 13, 2015, from http://www.reuters.com/article/2014/11/11/us-music-spotify-idUSKCN0IV1V420141111

Shi, W., Lee, H.-H. S., Lu, C., & Zhang, T. (2004). Attacks and risk analysis for hardware supported software copy protection systems. In *Proceedings of the 4th ACM workshop on Digital rights management - DRM '04* (p. 54). New York, New York, USA: ACM Press. http://doi.org/10.1145/1029146.1029156

Simpson, P. M., Banerjee, D., & Simpson, C. L. (1994). Softlifting: A model of motivating factors. *Journal of Business Ethics*, *13*(6), 431–438. http://doi.org/10.1007/BF00881451

Siponen, M., Vance, A., & Willison, R. (2012). New insights into the problem of software piracy: The effects of neutralization, shame, and moral beliefs. *Information & Management*, *49*(7-8), 334–341. http://doi.org/10.1016/j.im.2012.06.004

Sudler, H. (2013). Effectiveness of anti-piracy technology: Finding appropriate solutions for evolving online piracy. *Business Horizons*, *56*(2), 149–157. http://doi.org/10.1016/j.bushor.2012.11.001

Thongmak, M. (2015). Antecedents and consequences of the intention of young consumers to pirate or buy copyright products. In *2015 International Conference on Industrial Engineering and Operations Management (IEOM)* (pp. 1–9). IEEE. http://doi.org/10.1109/IEOM.2015.7093796

Tufnell, N. (2011). Interview: Gabe Newell. Retrieved May 12, 2015, from http://www.tcs.cam.ac.uk/interviews/0012301-interview-gabe-newell.html

Xu, G., & Xiang, G. (2012). A method of software watermarking. In *2012 International Conference on Systems and Informatics (ICSAI2012)* (pp. 1791–1795). IEEE. http://doi.org/10.1109/ICSAI.2012.6223392

Yuvaraj, N., Venkatraj, D., & Yogesh, P. (2011). Piracy curbing thumb drives. In *2011 International Conference on Research and Innovation in Information Systems* (pp. 1–3). IEEE. http://doi.org/10.1109/ICRIIS.2011.6125706

Ødegård, A. (2012). Enfour, Inc. screws up big time, makes dictionary app auto-post false accusations on users' Twitter accounts. Retrieved from http://www.pocketables.com/2012/11/enfour-inc-screws-up-big-time-makes-dictionary-app-auto-post-false-accusations-on-users-twitter-accounts.html