Possibilities of Using Watermarks to Protect Software Code

Vadym Poddubnyi[1], Roman Gvozdev[2], Oleksandr Sievierinov[3], Oleksandr Fediushyn[4]

[1,2,3,4] *Kharkiv National University of Radio Electronics, 14 Nauky Ave, Kharkiv, 61166, Ukraine.*

Abstract
This paper considers methods for software code protection from modifying and illegal distribution. Including methods based on digital watermarks, and zero digital signs. One of the promising methods of program code protection is the KeySplitWatermark method. The paper considers it and the possibility of modernization.

Keywords
Watermarks, software, zero watermarks, KeySplitWatermark.

1. Introduction

The problem of software protection from attackers appeared with the advent of the first commercial program. Despite the modernization of software development, delivery, and integrity facilities, the annual cost of distributing unlicensed software is approximately $46.3 billion. Although in recent years the percentage of unlicensed software in the world has decreased from 39% to 37%, the problem of protecting software code and programs in general will remain relevant. This problem is especially important for the post-Soviet space, so in Ukraine the percentage of unlicensed software is 82%, in Russia 62% and in Belarus 82%, which is similar to the indicators of developing countries in Africa (Nigeria 80%, Kenya 74%, Zambia 80%) [1].

It should be noted that not only unlicensed distribution can cause damage, attackers can embed malicious elements in the program, use separate modules of the program, etc. (Figure 1).

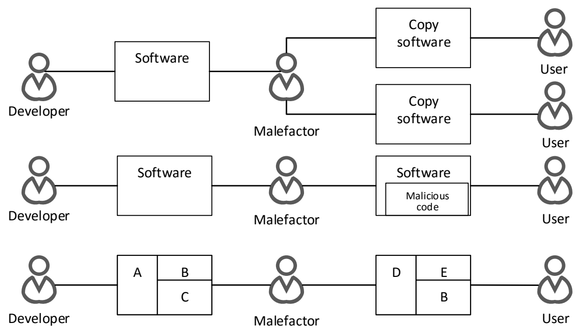

Figure 1. Possible software attacks

2. Methods of program code protection

To reduce the loss from unlicensed distribution and embedding malicious elements in the program code, software developers are forced to use a variety of protections.

Some of the most common methods of software removal are:

1. Adding program code to prevent intrusions;
2. Obfuscation of the program code;
3. Digital watermarks [2].

Obfuscation - is the process of code reorganization, primarily aimed at complicating

III International Scientific And Practical Conference "Information Security And Information Technologies", September 13–19, 2021, Odesa, Ukraine
EMAIL: vadym.poddubnyi@nure.ua (A. 1); roman.hvozdov@nure.ua (A. 2); oleksandr.sievierinov@nure.ua (A. 3), oleksandr.fediushyn@nure.ua (A. 4)
ORCID: 0000-0002-4380-491X (A.1); 0000-0002-5408-943X (A.2); 0000-0002-6327-6405 (A.3); 0000-0002-3600-405X (A.4)

the disassembly of software code by an attacker. It involves modifying a program, or adding code to a program to increase its complexity.

The main methods of obfuscation:

- Formatting transformations that change only the appearance of the program. This group includes conversions that delete comments, indents in program text, or rename IDs.
- Transform data structures that change the data structures that the program works with. This group includes, for example, transformations that change the hierarchy of class inheritance in a program, or transformations that combine scalar variables of the same type into an array.
- Convert a program's control flow to change the structure of its control flow graph, such as sweeping loops, selecting code snippets into procedures, and more.
- Preventive transformations that target certain decompilation methods or use bugs in certain decompilation tools.

The downside of obusfuscation is the complexity of the development process and modernization of software, and the software after obusfuscation may be more complex and slower [3].

To ensure the integrity of the software, developers add to the programs special modules that to check software integrity. Such code blocks check the hash values of the program and its components, encrypt and decrypt the program code, or monitor the status of the program (respond to incorrect data or commands, etc.).

To protect the program from hacking, you need to make sure that it "works as intended" even if attacker tries to interrupt, control or change the execution of the program code.

It should be noted that this is different from obfuscation, where the goal is to make it more difficult for an attacker to understand and read the program.

The disadvantages of this method are the increase in the number of resources for the operation of the program, as it requires additional resources of the protection module. Such modules may also conflict with other software. Also, such modules can interfere with the operation of parts of the program or other programs.

In practice, the line between protection against unauthorized access and obfuscation is blurred: a program that is more difficult to understand because it has been confusing will also be more difficult to modify and attack.

Digital watermarks are special secret messages that are embedded in the program code or program data, they serve to confirm the authorship and preserve the integrity of the data.

Since its inception, digital watermarks have been commonly used for multimedia data embedded in various signal characteristics (frequency, brightness, color, etc.). However, over time, digital watermarks began to be used to protect software.

3. Watermark type

According to the methods of embedding in the program code, digital watermarks are divided into static and dynamic. Static watermarks are embedded in program code or data as opposed to dynamic ones, which store the watermark during program execution. [4]

According to their characteristics, digital watermarks are divided into:

- Fragile. Digital watermarks that are impossible to detect, with the slightest modification. Used to control integrity;
- Semi-fragile. Digital watermarks that can withstand some changes in the carrier digital watermark. Is used to detect an attack;
- Reliable. Watermarks are resistant to all types of attacks. Used for authentication and authentication.

There are various types of embedding digital watermark in the program, the most common of which are:

1. Replacement of the code;
2. Replacement of code logic;
3. QP algorithm;
4. QPS algorithm;
5. Digital watermark on the basis of graphs.

The downside of digital signs is that the digital watermark increases the size of the program. Static watermarks cannot fully protect data and require additional protection methods [4].

Watermarks and protection against unauthorized access are also related. In fact, if perfect protection against unauthorized access were available, it would be easy to add watermarks, watermarks should be combined with any trivial algorithm to protect against unauthorized access, and an attacker would not be able to find or destroy the tag. Precisely because there is no perfect protection against unauthorized access, you need to worry about masking watermarks.

It is assumed that an attacker who can find a watermark will also be able to change the program to destroy the sign [5]. A graphical representation of the digital watermark is shown in Figure 2.

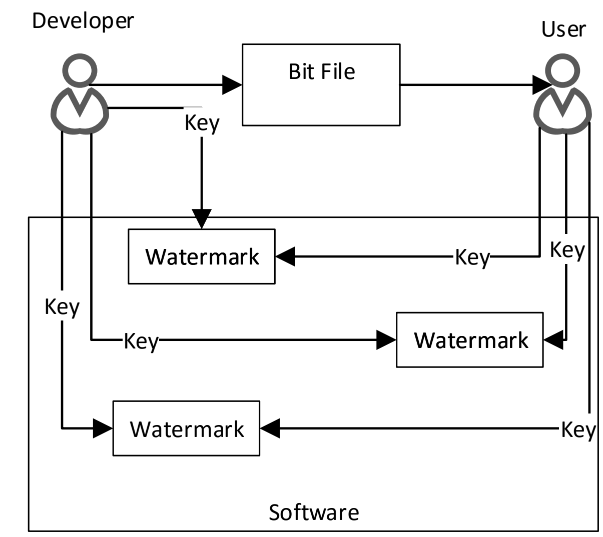

Figure 2. Graphic representation of a digital watermark

4. Zero digital watermark

One of the methods of solving the problems of digital watermarks is "zero watermarks".

A traditional digital watermark hides information about the owner or creator of an object or objects group of objects somewhere inside that object. This hidden information can later be used for many purposes: maintaining integrity, detecting intentional or accidental interference, protecting data copyright, etc.

Zero watermarks, unlike "normal" digital watermarks, are not embedded in program code. Program, data, or code structure is used to generate a null character.

Also, one of the advantages of zero digital characters is that they are resistant to compression of the embedded object.

Graphical representation of the zero digital sign is shown in Fig. 3.

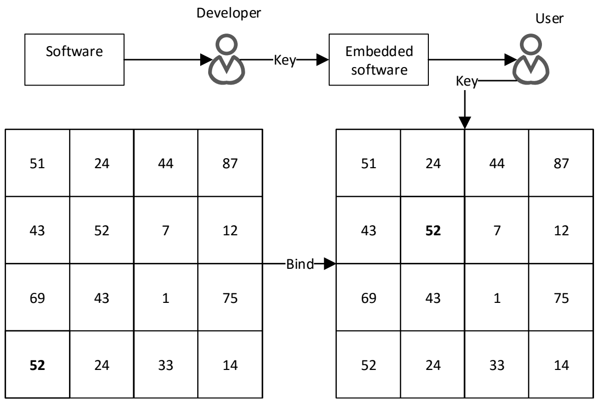

Figure 3. Graphical representation of the zero digital watermark

Zero digital watermarks are widely used in medicine [6] [7] to protect patient data, but zero digital signs can also be used to protect software.

One example of zero digital watermark algorithms for program code protection is the algorithm considered by KeySplitWatermark [8]. There are also algorithms for fragile digital watermarks to protect the database from modifications [9].

These algorithms use statistical data and asymmetric encryption using a certification authority to generate digital watermarks. The characteristics of this type of digital watermarks indicate the prospects for their use to protect software code from unauthorized changes or from unlicensed distribution.

5. KeySplitWatermark Algorithm

KeySplitWatermark algorithm is presented by a group of developers from different universities around the world such as China, Pakistan, India and others. KeySplitWatermark is a new approach based on a blind zero watermark to protect software source code from cyberattacks.

KeySplitWatermark first analyzes the program code to determine the keywords, and then divides the code into sections based on the selected keyword. The algorithm generates a unique key using keywords and the program code itself. If you have any copyright concerns in the future, you can use this key to verify ownership. The implementation algorithm does not make any changes to the program code to create watermarks, and the extraction algorithms do not require the use of watermarks as input, which makes it blind (zero digital sign).

The watermark algorithm consists of two components; embedding and removing watermarks. Watermark embedding is performed by the original owner of the software, and removal is later performed by a trusted third party.

In this algorithm, the program code is first pre-processed to identify the ten most common characters and the five most common keywords. It is then divided into sections based on the user-selected keyword KeySplitWatermark, in which the implementation algorithm accepts the following input:

- Source code: The source code of the software to which the watermarks should be applied.
- Cipher: a numeric value that will be used in the key generation process.
- Watermark: ASCII character group.

The implementation algorithm generates the owner key as the output. This key is written to the certificate authority and then used to remove the watermark (if necessary). The extraction algorithm accepts the following input data:

- Attacked code file: A program code file that has been modified or used illegally as a copyright infringement.
- Owner key: It is obtained from the certification authority to identify the original owner

The certificate authority is a requirement of this algorithm that registers content to the copyright owner. When an attack is suspected, this trusted third party removes the watermarks and provides the original code of the recovery software if a counterfeit is detected. The fake code is replaced by the original code, which makes the actions of the attacker invalid.

The graphical representation of the algorithm is shown in Figure 2.

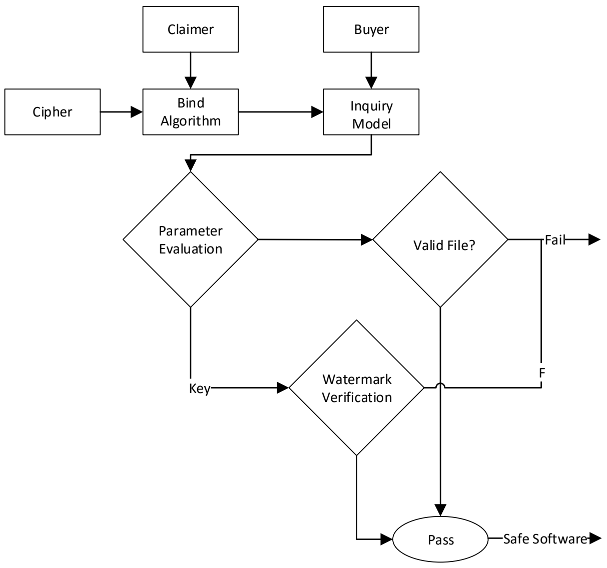

Figure 4. Graphical representation of the KeySplitWatermark algorithm

It is impossible to destroy a watermark without a significant change in the code, and if any changes occur in the code, the source code is restored. The results of research conducted by the authors prove that KeySplitWatermark is reliable, secure and efficient with minimal computational requirements.

The results of research conducted by the authors prove that KeySplitWatermark is reliable, secure and efficient with minimal computational requirements (Table 1)[8].

To evaluate the reliability of KeySplitWatermark, developers of the algorithm used ASProtect, Upx and Aspack to attack the program with watermarks and check the correctness of the removed watermark. The results of the experiment are shown in Table 2.

The watermark can be properly removed after encryption, shelling, and watermark compression attacks. The initial semantics of the program are preserved, although various attacks are carried out.

The algorithm is promising, has potential and requires detailed analysis and study [8]. Since the algorithm is new, the following vectors of research and modernization are offered as improvements:

1. Use Unicode instead of ASCII to generate keywords;
2. Parse program code with keyword pairs to increase the number of code split combinations;
3. National algorithms for certificate authority.

Switching to Unicode is suggested to potentially increase the languages to use and increase the length of the keywords generated.

The use of keyword pairs should expand the variability of the choice and potentially increase the stability of the algorithm. It is also proposed to increase the number of keywords for the same purpose.

The use of national algorithms (such as DSTU 7624 [10], DSTU 4145[11], DSTU 7564[12]) can improve the stability of the algorithm.

A promising task is to create a certification center for the use of the KeySplitWatermark algorithm and its testing.

Table 1

Comparative Results for Increase in the size of the Watermarked Code and in Execution Time for Crptoencryption With 31KB File

Watermark length (bit)	Increase in program (KB)	Increase in program KeySplitWatermark	Execution time(ms)	Execution time KeySplitWatermark
128	18	0	23	18
256	34	0	40	32
512	67	0	45	39
1024	130	0	123	105

Table 2

Attacks and results

Tool	Attack Mode	Extraction	Extraction KeySplit Watermark
ASProtect	Encrypts program	100%	100%
UPX	Conducts code compression	100%	100%
Aspack	Used to shell the program	100%	100%

6. Conclusions

This paper provides a brief overview of methods for protecting software code from modification and distribution. One such method is digital watermarks. This method has many disadvantages, but they have been eliminated with the advent of a new type of digital watermarks - zero digital watermarks.

One of the promising methods of zero digital sign is KeySplitWatermark. To improve the characteristics, its modernization and further research are proposed. It is also proposed to study and use it together with national algorithms (DSTU 7624, DSTU 4145, DSTU 7564) and certification authority.

7. References

[1] Business Software alliance, Software Management: security imperative, business opportunity, 2018.

[2] Christian S. Collberg, Clark Thomborson Watermarking, Temper-Proofing, and Obfuscation – Tools for Software Protection, 2000.

[3] Чернов А. В., Анализ запутывающих преобразований программ, 2003, URL: http://citforum.ru/security/articles/analysis/.

[4] James Hamilton, Sebastian Danicic Department of Computing, Goldsmiths, University of London United Kingdom, A Survey of Static Software Watermarking, URL: https://www.researchgate.net /publication/224229798_A_survey_of_static _software_watermarking.

[6] Aleš Roček, corresponding author Michal Javorník, Karel Slavíček, and Otto Dostál, Zero Watermarking: Critical Analysis of Its Role in Current Medical Imaging, URL: https://www.ncbi.nlm.nih.gov/pmc/articles/ PMC7886926/.

[7] Zulfiqar Ali, Muhammad Imran, Mansour Alsulaiman, Tanveer Zia, Muhammad Shoaib, A Zero-Watermarking Algorithm for Privacy Protection in Biomedical Signals.

[8] Celestine Iwendi, Zunera Jalil, KeySplitWatermark: Zero Watermarking Algorithm for Software Protection Against Cyber-Attacks, 2020, URL:

https://ieeexplore.ieee.org/document/9068217/references#references.

[9] Aihab Khan, Syed Afaq Husain, A Fragile Zero Watermarking Scheme to Detect and Characterize Malicious Modifications in Database Relations, 2013, URL:https://hindawi.com/journals/tswj/2013/796726/.

[10] National standard of Ukraine, Information technologies. Cryptographic information protection. Symmetric block transformation algorithm DSTU 7624: 2014.

[11] National standard of Ukraine, Cryptographic information protection, Based digital signature on elliptical curves. formation and verification DSTU 4144-2002.

[12] National standard of Ukraine, Cryptographic information protection. Hashing function DSTU 7564: 2014.