

Proposed Methods to Protect Built Software from Illegal Copy

Israa Shihab Ahmed¹

¹Informatics Institute for Post-Graduation Studies (IIPS Iraqi Commission for Computer & Informatics), Iraq

israa_aljebory@iips.edu.iq

Abstract

Software piracy remains a significant challenge in the digital age, threatening intellectual property and economic sustainability. This paper presents various hardware and software-based protection methods to secure proprietary software against unauthorized use and reverse engineering. The research introduces four novel protection mechanisms, including USB-dongle-based encryption, remote authentication, and encrypted software licensing. A comprehensive evaluation of these techniques highlights their effectiveness against reverse engineering attacks, providing a robust framework for software security.

1. Introduction

Software solutions play a crucial role in various industries, yet they are vulnerable to illegal reproduction and unauthorized distribution. This paper explores methods to enhance software protection by employing both hardware and software-based techniques. The proposed methods aim to increase the difficulty of cracking software, making unauthorized access economically unfeasible.

2. Related Work

Several studies have addressed software protection using encryption, license validation, and hardware-based authentication. Existing solutions, such as serial number validation, hardware dongles, and server-based authentication, demonstrate varying degrees of effectiveness. However, they remain susceptible to reverse engineering and cracking techniques, necessitating more robust approaches [1][2][3].

2.1 Data Security Principle

Data security refers to protection of data from unauthorized (accidental or intentional) modification, destruction, or disclosure.

Data security is also known as information security (IS) or computer security.

This section explains all the important related principles of data security to support software copyright.

2.1.1 Evaluation Criteria

From a security point of view, a suitable concept is the one of protected subsystems, introduced as a security requirement. It serves as a model for protecting both host-based and distributed software. This model is equivalent to the concept, requiring a higher level of difficulty for an attacker to extract special part(s) of the software (the subsystem) [36].

A subsystem is defined as a "collection of programs and databases that are encapsulated so that other executing programs can invoke only certain component programs within the protected subsystem, but are prevented from reading or writing component programs or databases, and are prevented from disrupting the intended operation of the component programs". Such a system is intended to detect modifications and disruptions by physical attacks, rather than preventing them [37].

Kent [36] also defines some criteria, such as:

Decentralization This criterion requires that the protection mechanisms must be decentralized in order to gain advantage in prevents attack.

Effectiveness to meet the security requirements over a broad spectrum of attacks, the protection mechanisms must provide a unified approach. Based on an anticipated threat environment, the protection mechanism should change only its parameters, rather than the system itself.

Generality/Flexibility a decent protection mechanism should be compatible with a wide range of applications, systems and equipment, rather than requiring to rely on a certain technology or equipment, which the system should be dependent on.

Low Cost A cost-benefit analysis should give positive result for the use of the protection mechanism: the cost of using the protection should be lower than the cost of losses caused by not using it.

Good Performance It is important for a solution to be able to perform at an acceptable time, which does not degrade the use of the software itself. A trade-off between effectiveness and performance may be usually required, giving each the necessary position.

Transparency The requirement is that the protection mechanism should be unobtrusive, so that integration with the writers of external software is made with ease and without affecting (much) the design of the external software.

2.1.2 Information Security

Information security means protecting information and information systems from unauthorized access, use, disclosure, disruption, modification, perusal, inspection, recording or destruction [38]. The terms information security, computer security and information assurance are frequently used interchangeably. These fields are interrelated often and share the common goals of protecting the confidentiality, integrity and availability of information; however, there are some subtle differences between them [39].

Principles of information security:-

2.1.2.1 Confidentiality

Confidentiality is the term used to prevent the disclosure of information to unauthorized individuals or systems. For example, a credit card transaction on the Internet requires the credit card number to be transmitted from the buyer to the merchant and from the merchant to a transaction processing network [40].

The system attempts to enforce confidentiality by encrypting the card number during transmission, by limiting the places where it might appear (in databases, log files, backups, printed receipts, and so on), and by restricting access to the places where it is

stored. If an unauthorized party obtains the card number in any way, a breach of confidentiality has occurred [40].

2.1.2.2 Integrity

In information security, integrity means that data cannot be modified undetectably [41]. Integrity is violated when a message is actively modified in transit. Information security systems typically provide message integrity in addition to data confidentiality [42].

2.1.2.3 Availability

For any information system to serve its purpose, the information must be available when it is needed. This means that the computing systems used to store and process the information, the security controls used to protect it, and the communication channels used to access it must be functioning correctly. [43].

2.1.2.4 Authenticity

In computing and information security, it is necessary to ensure that the data, transactions, communications or documents are genuine. It is also important for authenticity to validate that both parties involved are who they claim they are [44].

*Authentication System:-*Authentication is any process, through which one proves and verifies certain information, i.e. determining whether someone or something is, in fact, who or what it is declared to be. Authentication requires that the information be checked for single, previously identified entity.

2.1.3 Cryptography

In a modern branch of cryptography also known as public-key cryptography the algorithms employ a pair of keys (a public key and a private key). The many public key algorithms used and the most popular one is RSA [45].

2.1.3.1 RSA Algorithm

RSA is an algorithm for public-key cryptography that is based on the presumed difficulty of factoring large integers, the factoring problem. RSA stands for Ron Rivest,

Adi Shamir and Leonard, who first publicly described it in 1978. A user of RSA creates and then publishes the product of two large prime numbers, along with an auxiliary value, as their public key.

The prime factors must be kept secret. In RSA if one can factor modulus into its prime numbers then the private key is also detected and hence the security of the cryptosystem is broken [45]. RSA is based on the principle that some mathematical operations are easier to do in one direction but the inverse is very difficult without some additional information. In case of RSA, the idea is that it is relatively easy to multiply but much more difficult to factor.

Multiplication can be computed in polynomial time whereas factoring time can grow exponentially proportional to the size of the number [45].

2.1.3.2 Security of RSA

The security of RSA cryptosystem is also broken by two attacks based on factorization attack and chosen-cipher text attacks.

A) Factorization Attack: The security of RSA is based on the idea that the modulus is so large that it is infeasible to factor it in a reasonable time. Select p and q and calculate $n=p*q$ (two prime numbers). Although n is public, p and q are secret. For example, when Eve can factor N and obtain P and Q , Eve then can calculate $d = e^{-1} \bmod \phi(N)$ because e is public. The private exponent d is the trapdoor that Eve uses to decrypt any encrypted message [46].

B) Chosen-Cipher text Attack: A potential attack on RSA is based on the multiplicative property of RSA [46].

2.2 Reverse Engineering

Reverse engineering is taking apart an object to see how it works in order to duplicate or enhance the object. The practice, taken from older industries, is now frequently used on computer hardware and software. Software reverse engineering involves reversing a program's machine code (the string of 0s and 1s that are sent to the logic processor) back into the source code that it was written in.

Software reverse engineering is done to retrieve the source code of a program because the source code was lost, to study how the program performs certain operations,

to improve the performance of a program, to fix a bug (correct an error in the program when the source code is not available), to identify malicious content in a program such as a virus or to adapt a program written for use with one microprocessor for use with another.

Reverse engineering for the purpose of copying or duplicating programs may constitute a copyright violation. In some cases, the licensed use of software specifically prohibits reverse engineering [47].

Reverse engineering is the process of creating a blueprint of a machine to discern its rules by looking only at the machine and its behavior. At a high level, this process involves taking something that you may not completely understand technically when you start, and come to understand completely its function, its internals, and its construction. A good reverse engineer attempts to understand the details of software, which by necessity involves understanding the overall computing machinery that the software runs on functions [48]. A reverse engineer requires a deep understanding of both the hardware *and* the software, and how all work together [48].

2.2.1 Illegality of Reverse Engineering

Because reverse engineering can be used to reconstruct source code, it walks a fine line in intellectual property law. Many software license agreements strictly forbid reverse engineering. Software companies fear (and rightly so) that their trade secret algorithms and methods will be more directly revealed through reverse engineering than they are through external machine observation.

However, there is no general-purpose law against reverse engineering. Because reverse engineering is a crucial step in removing copy protection schemes, there is some confusion regarding its legality [48].

Need for Reverse Engineering because reverse engineering allows you to learn about a program's structure and its logic. Reverse engineering thus leads to critical insights regarding how a program functions. This kind of insight is extremely useful when you exploit software. There are obvious advantages to be obtained from reverse

engineering. For example, you can learn the kind of system functions a target program is using. You can learn the files the target program accesses.

The most powerful advantage to reversing is that you can change a program's structure and thus directly affect its logical flow. Reverse engineering is a powerful tool that hackers can use to compromise the security of your business [48].

2.2.2 Reverse Engineering Tools

There are many tools which have been used to break the protection[49], and the paragraphs below will describe each of the tools which are considered as important and famous:

1. Debugger

The debugger is a tool to stop execution and point to a certain task where the execution is stopped, hence, it can be used to determine where the execution of a certain code is stored or at least where to look for or what kind of code we are looking for. The debugger has many other functions like breaking the execution under certain conditions.

The debugger must be compatible with the operating system because it should use interrupts used by the system to monitor activities and message queues.

There are many available debuggers varying from open source and freeware debuggers for commercial and professional operations. The debugger used here is the PE Browse Profesional Interactive.

2. Disassembler

The disassembler is a software used to disassemble executable code to assembly language where you can see the row instructions sent to the microprocessor itself . and you can edit and modify the code without complication. A disassembler is a very important tool since it enables the user to search for pieces of code and to modify them or decode them and trace what they do.

3. PE File Viewer

A PE file viewer is an executable file viewer from the type PE where most of the 32 bit windows operating systems are capable of read. A PE viewer

can decode file header parameters for the execution and show where all the data and code parts are being stored in addition to what libraries this file is using. A generic PE editor can be used to seek the parameters for our victim executables.

4. Hex Editor

A hex editor is a viewer and an editor that enables the user to see the files as a row of 16 bytes each converted to hexadecimal values to make it easier to understand. Also it can be used to modify codes or pieces of data stored in the file's header. There are many hex editors for free on the internet although the research uses hexedit.

3. Proposed Protection Methods

on the server side user information is stored and monitored in a way to determine the number of times the serial number is used. Figure (3.2) shows block diagram of server based authentication, this method starts from user side .The user initiates the first step to authenticate his own copy of the protected software. The server should be running to complete the authentication process.

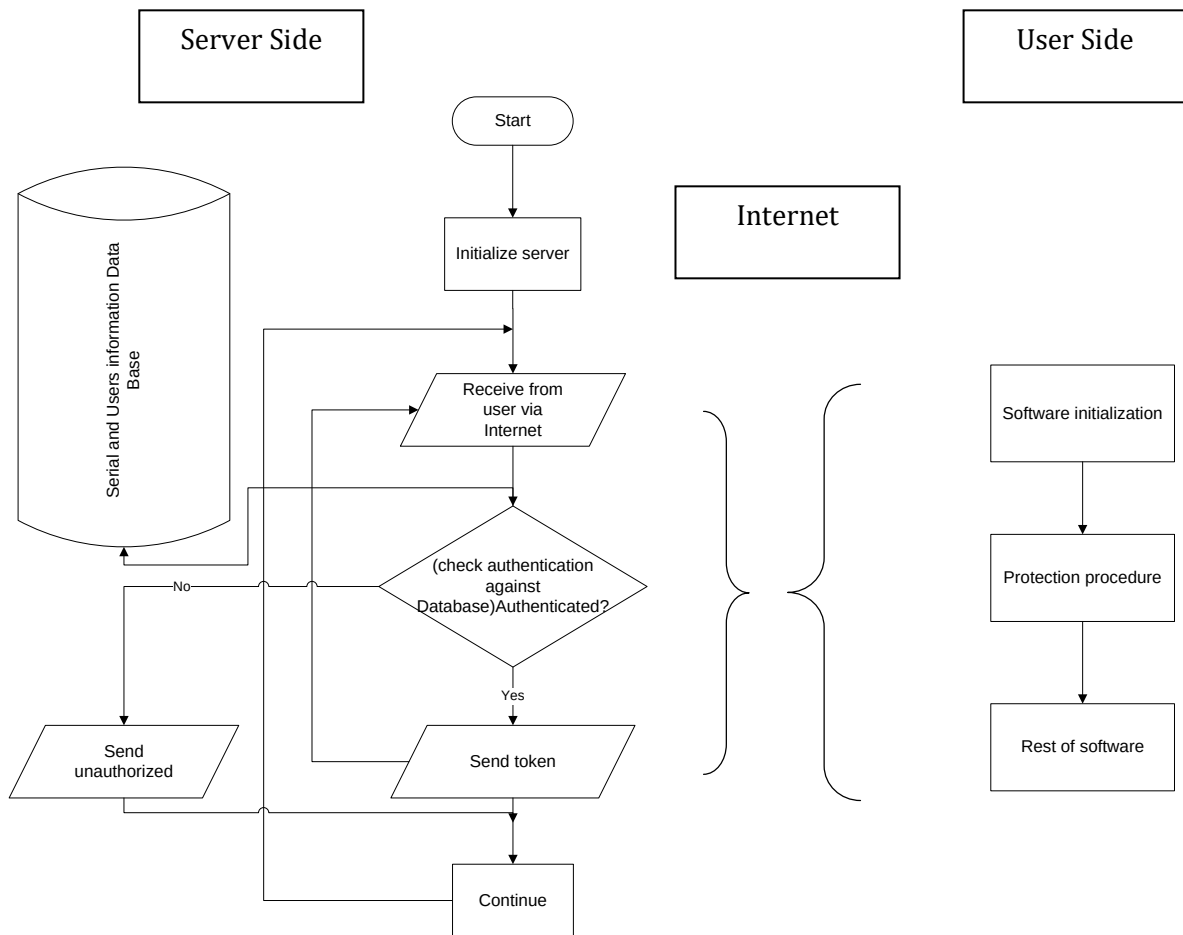


Figure (3.2) A block diagram showing a protection procedure implementation from a web server to protected software

3.1. Encrypted Key Mechanism

This method utilizes a USB-disk as a unique identifier, storing a hardware-locked encryption key. The protected software verifies the presence of the USB-disk by decrypting a key stored in its database. If the correct key is absent, the software refuses execution [4].

3.2. Encrypted Class Mechanism

A protection subroutine is stored within the USB-disk, acting as a required execution component. Without the correct USB-disk, the software cannot access necessary encryption algorithms, preventing unauthorized use [5].

3.3. Encrypted USB-Disk Information

A unique identification file is created on the USB-disk, storing encrypted information about its parameters. The software verifies this file upon execution to ensure authenticity, preventing the use of cloned or emulated disks [6].

3.4. Serial and Validation Server

A web-based authentication mechanism generates unique serial numbers linked to specific hardware identifiers. Upon installation, the software verifies the serial against a remote server, ensuring each license remains bound to a single machine [7].

4. Experimental Results

The proposed methods were evaluated using reverse engineering tools such as OllyDBG and PE Explorer. The time required to bypass each protection method was recorded, demonstrating that USB-disk-based encryption and remote authentication methods provided superior resistance to cracking attempts [8].

5. Conclusion and Future Work

The research presents four innovative software protection techniques, each offering a different level of security against unauthorized copying. Future work aims to enhance these methods by integrating AI-driven anomaly detection and blockchain-based licensing mechanisms for improved security and efficiency [9].

References

- [1] Manoharan, S. & Wu, Software licensing: A classification and case study. International Conference on the Digital Society, 2007.
- [2] Stuart Stubblebine and T.Premkumar Devanbu, "Software engineering for Software Engineering for security a roadmap", Proceeding of the conference on the future of software Engineering, ACM Press, 2000.
- [3] Min Chen, "Why We Need Software Protection". Available at <http://www.tml.hut.fi/studies/2001/papers/index.html>, 2001.
- [4] Zoeller & Renate. "Nest of Pirates", Transitions online, 2007.

- [5] Fatbardh Veseli, "Highly Secure, Intelligent Software Copy-Protection", Master of Science in Information Security, 2011.
- [6] Piazzalunga U., Salvaneschi P., Balducci F., Jacomuzzi P., Moroncelli C., "Security Strength Measurement for Dongle-Protected Software", IEEE Security & Privacy, 2007.
- [7] Olga, G, Bhagirath, N. & Rahul, S. "SPEE, A Secure Program Execution Environment tool using code integrity Checking", Journal of High-Speed Networks, 2006.
- [8] Zainab M. Fadhel and Dr. Alia Karim Abdul Hassan, "Software Piracy Types", University of Technology, 2010.
- [9] Eldad, E., "Reversing: Secrets of Reverse Engineering", John Wiley & Sons, 2005.