IJIREEICE ISSN (O) 2321-2004, ISSN (P) 2321-5526

International Journal of Innovative Research in Electrical, Electronics, Instrumentation and Control Engineering

Impact Factor 8.021 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 13, Issue 3, March 2025

DOI: 10.17148/IJIREEICE.2025.13338

ANTI PIRACY SOFTWARE PROTECTION TOOL

BHUVANESH. J[1], Dr. R. PRABA[2]

UG Student, Department of Information Technology, Dr. N.G.P Arts and Science College, Coimbatore[1]

Associate Professor, Department of Information Technology, Dr. N.G.P Arts and Science College, Coimbatore[2]

Abstract: This project is designed to protect software from piracy and ensure that it can only be accessed by authorized users. With software piracy being such a huge issue on the internet, it poses a significant threat to software companies. Hackers can use malicious programs to break into systems, steal software, and use it illegally. That’s why there’s a need for a system that protects both the software and the users from theft and misuse. The purpose of this project is to create a Software License Management System that works over the internet or intranet. It helps prevent hackers from using pirated software by generating and verifying serial keys remotely. The system keeps track of important details like the number of licenses sold, the type of license, the license number, serial keys, number of users, and the validity of each license. This system makes sure that only legitimate, paid users can install and use the software. It verifies the license at the time of installation and stops the installation process if the software is pirated. All information about sold software and license transactions is securely stored in a centralized database for easy verification. Additionally, the system allows users to register their software and product details online. These details are securely stored in the database, and the serial keys are encrypted to prevent hackers from stealing them. Users also get email alerts and reminders to keep them informed about software updates and license renewals

1.INTRODUCTION

The primary objective of this project is to develop a robust Anti-Piracy Software Protection Tool that ensures the secure distribution and usage of licensed software, safeguarding it from unauthorized access and misuse. This tool is designed to protect the intellectual property rights of software developers and companies by implementing a comprehensive Software License Management System. Through functionalities like serial key generation, verification, and centralized license management, the tool aims to prevent piracy, detect unauthorized installations, and ensure that only authenticated users can access the software. By integrating encryption techniques for storing serial keys and leveraging a centralized database for tracking license details, the project seeks to establish a secure and scalable system for managing software licenses.

Additionally, the tool is intended to enhance user engagement and convenience through automated features such as email alerts and periodic reminders for software updates. By facilitating user registration and securely maintaining product and license details, the project aims to streamline the process of license verification and enforcement. With ASP.Net as the front-end and SQL Server as the back-end, this project seeks to deliver an efficient, user-friendly, and secure solution for combating software piracy, ultimately supporting the integrity and sustainability product businesses.

LITERATURE REVIEW: ANTI-PIRACY SOFTWARE PROTECTION TOOLS

Software piracy is a major problem in today’s digital world, leading to substantial losses for software companies. Hackers and malicious software allow people to illegally copy, distribute, and use programs without paying for them. This not only harms businesses financially but also threaten the security of the products users rely on. To tackle this issue, software companies have developed a range of tools and techniques aimed at preventing piracy. This literature review will highlight the different methods used in anti-piracy software protection, focusing on the evolution of these tools and current trends.

1. The Impact of Software Piracy

Piracy occurs when software is used without a valid license, often through unauthorized copying or distribution. The damage caused by piracy is immense. According to the Business Software Alliance (BSA), piracy costs software companies billions of dollars every year (BSA, 2020). Beyond the financial losses, piracy also leads to decreased software security, as pirated versions often contain malware or vulnerabilities (Koggel, 2014). This makes it crucial for developers to adopt strategies that prevent software misuse.

2. Techniques Used in Anti-Piracy Protection

To protect software from illegal copying and usage, several anti-piracy protection techniques are commonly used. These can be broken down into three major categories: serial key management, hardware-based protection, and online authentication.

2.1 Serial Key Management

Serial keys are one of the simplest and most widely used forms of software protection. When users purchase software, they are given a unique serial key that must be entered during installation. The software then checks the key against a database to ensure that it's valid. While this method is effective to some degree, it's not foolproof. Hackers often develop tools (called keygens) that can generate these serial keys, bypassing the protection (Eichhorn et al., 2015). Nevertheless, using encryption to store serial keys can still help make it harder for hackers to tamper with the system (Farooq & Khan, 2018).

2.2 Hardware-Based Protection

Another approach to preventing software piracy involves using physical hardware, like USB dongles or hardware tokens. These devices must be plugged into the computer in order for the software to run. If the device is removed or absent, the software won't work. While this method can be highly secure, it also has its downsides, such as the risk of losing the dongle or the added cost of producing and distributing these physical devices (Mundt & Ziegler, 2019). Despite these challenges, this form of protection is still commonly used for valuable software, like CAD programs or video games (Gupta & Lee, 2017).

2.3 Online Authentication and Licensing

With the growth of cloud-based systems, online authentication has become a popular method for protecting software. This technique involves verifying the user's serial key or license information through an internet connection to a centralized server. Many major software products, like Microsoft Office and Adobe, use this method. This ensures that users are connected to the server to confirm their legitimacy whenever they use the software (Li & Liu, 2020). However, online authentication can still be bypassed through proxies or unauthorized sharing, so extra measures like encryption or digital signatures are often used to protect the communication (Luo et al., 2019).

3. Additional Anti-Piracy Strategies

Beyond these common methods, there are additional strategies software companies use to deter piracy.

3.1 Obfuscation and Code Protection

Obfuscation involves making the software code difficult for hackers to understand or reverse engineer. By altering code structure, renaming variables, or encrypting certain parts of the code, it becomes more difficult for unauthorized individuals to tamper with the software (Huang & Wang, 2016). When used alongside other protection methods, code obfuscation helps increase the overall security of the software.

3.2 Watermarking

Watermarking involves embedding hidden information into the software that identifies the legitimate buyer. If a pirated version of the software is found, the watermark helps track the source of the leak. While this can help identify the origins of piracy, it's not foolproof since hackers can sometimes remove or modify the watermark (Ahsan et al., 2017). Still, it remains a useful tool in combination with other anti-piracy measures.

4. Challenges and Limitations

Despite the advancements in anti-piracy tools, several challenges remain. One of the biggest obstacles is cracking and reverse engineering, where hackers find ways to bypass protection systems (Cai et al., 2019). Once a weakness is found, it is often shared online, allowing pirated versions of the software to circulate freely.

Another challenge is the user experience. Some anti-piracy methods, such as online authentication and hardware-based solutions, can frustrate legitimate users, especially if they face connectivity issues or lose their hardware dongle (Santos et al., 2018). It's important to find a balance between security and convenience for the end user.

Lastly, the **cost** of implementing advanced anti-piracy measures, like hardware protection or complex encryption, can be prohibitive for smaller companies (Choudhury & Mukherjee, 2021). For independent developers and smaller businesses, these costs can outweigh the benefits.

5. The Future of Anti-Piracy Protection

Looking ahead, it's likely that emerging technologies like Artificial Intelligence (AI) and Machine Learning (ML) will play a significant role in improving anti-piracy efforts. AI can help detect unusual patterns of software usage, allowing companies to spot potential piracy in real time (Khan et al., 2020).

Additionally, blockchain technology is gaining traction as a means of securely managing software licenses and preventing piracy, offering a decentralized and tamper-proof way to record transactions (Zhou et al., 2018).

Cloud-based software and subscription models are also gaining popularity. These methods make it easier for software companies to control who has access to their products, reducing the risk of piracy. Users will be required to stay connected to the internet, ensuring that software is always authenticated and updated.

METHODOLOGIES

System testing is state of implementation, which is aimed at ensuring that the system works accurately and efficiently as expect before live operation commences. It certifies that the whole set of programs hang together. System testing requires a test plan that consists of several key activities and step for run program, string, system and user acceptance testing. The implementation of newly designed package is important in adopting a successful new system. Testing is the important stage in software development. the system test in implementation stage in software development process. The system testing implementation should be confirmation that all is correct and an opportunity to show the users that the system works as expected. It accounts the largest percentage of technical effort in the software development process. Testing phase in the development cycle validates the code against the functional specification testing is vital to achievement of the system goals. The objective of the testing is to discover errors to fulfill this objective a series of test step unit, integration. validation and system tests were planned and executed the test steps are:

Unit Testing

Unit testing is all about checking the smallest pieces of the software to ensure they work as expected. Think of it as “module testing” — it’s done during the programming phase itself. In this stage, each part of the software, or "module," is tested to make sure it produces the correct output. The goal is to catch any issues early when each individual component is being developed, ensuring that each piece of the puzzle fits together perfectly.

Integration Testing

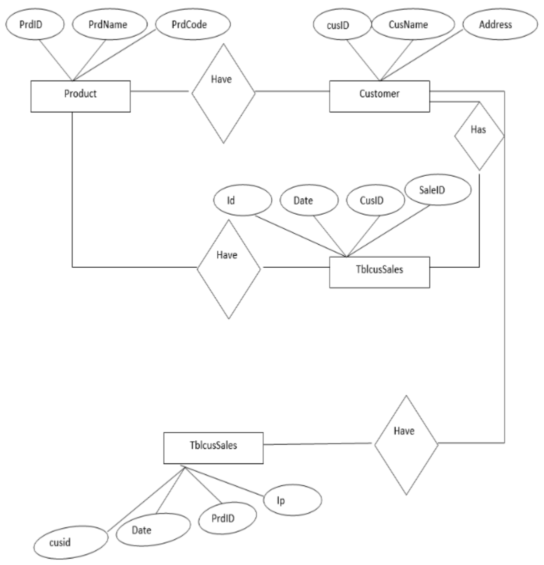
Once the individual modules are working, we move on to integration testing. This type of testing ensures that all the different modules of the software work well together when combined. Sometimes, data can get lost between modules, or one module might interfere with another, causing problems. Integration testing helps uncover these issues. The process involves testing the combined modules to make sure everything interacts smoothly. If there are any issues, they’re addressed, though fixing them can be tricky since it might be difficult to pinpoint the exact cause in a complex system. But this step is crucial for ensuring that the software as a whole works as expected.

Validation Testing

After integration testing, the software is nearly complete, with all modules combined and all major errors fixed. Now it’s time for validation testing, which focuses on ensuring that the software behaves as the user expects. In simple terms, validation testing checks if the software does what it was designed to do and if it meets the user’s needs. After this test, we typically end up with one of two outcomes: either the software works well and is ready to move forward, or further tweaks are needed.

ASP.NET Architecture

ASP.NET is based on the fundamental architecture of .NET Framework. Visual studio provide a uniform way to combine the various features of this Architecture.



The architecture of the .NET Framework can be understood by looking at it from bottom to top:

Common Language Runtime (CLR)
At the very base of the architecture is the **Common Language Runtime** (CLR). Think of it as the engine that runs your code. It sits on top of the operating system and is responsible for managing the execution of your program. When you write code for .NET, that code is known as "managed code," meaning it's handled by the CLR. This provides important features like memory management, error handling, and security. One key benefit of the CLR is that it allows programs written in different programming languages to work together seamlessly.

.NET Framework Class Libraries
Above the CLR is a huge set of **.NET Framework Class Libraries**. These libraries are like a toolbox that comes with pre-built code to help you accomplish common tasks. For example, there are libraries for networking, reading and writing files, and interacting with databases. There are also tools for debugging your code. All of these libraries are organized under the **Services Framework**, which ties everything together and makes it easy for developers to access and use them.

ADO.NET
One important part of the .NET Framework is **ADO.NET**, which is Microsoft's tool for working with data. ADO.NET is a modernized version of the older ADO (ActiveX Data Objects) but with a fresh approach. It's designed specifically for building web applications and helps you interact with different types of data, such as databases or XML files. ADO.NET allows developers to manipulate and access data efficiently, which is especially important in today's web-driven world. The "Software License Maker and Copyrights Monitor" has been designed to meet the requirements of the user. Need is stressed for detailed analysis of the existing manual system and a careful design so that further modification can be done in an efficient manner with minimum disruption to the system.

1. The Software License Maker helps the software company to protect the misuse of software from hacker. It consists of User accounts, Serial key database and updates. When the software installed on computer it automatically checks the License accounts database and breaks the software installation process, if the software is hacked not bought. With copyright relevant information are stored and maintained in centralized database for License verification when the software is sold.

- The system is a user-friendly editor to make the user comfortable to use.
- The system satisfies the needs of the user in reasonable time.
- Care is taken to reduce the user in inputting data.

CONCLUSION

It also provides the facility to manage product key details and License information in easiest way. The "Software License Maker and Copyrights Monitor" has been designed to meet the requirements of the user. Need is stressed for detailed analysis of the existing manual system and a careful design so that further modification can be done in an efficient manner with minimum disruption to the system.

DOI: 10.17148/IJIREEICE.2025.13338

The Software License Maker helps the software company to protect the misuse of software from hacker. It consists of User accounts, Serial key database and updates. When the software installed on computer it automatically checks the License accounts database and breaks the software installation process, if the software is hacked not bought.

REFERENCES

[1] Kathleen Reavis Conner, Richard P. Rumelt, (1991) Software Piracy: An Analysis of Protection Strategies. Management Science 37(2):125-139.
[2] Michael Stolpe (2000) Protection Against Software Piracy: A Study Of Technology Adoption For The Enforcement Of Intellectual Property Rights, Economics of Innovation and New Technology, 9:1, 25-52,DOI: 10.1080/10438590000000002.
[3] Nehra, R. Meena, D. Sohu and O. P. Rishi, "A robust approach to prevent software piracy," 2012 Students Conference on Engineerin and Systems, Allahabad, India, 2012, pp. 1-3, doi: 10.1109/SCES.2012.6199075.
[4] Sen, M., Dutt, A., Agarwal, S. and Nath, A., 2013, April. Issues of privacy and security in the role of software in smart cities. In 2013 International Conference on Communication Systems and Network Technologies (pp. 518-523). IEEE.
[5] Barbosa, P., Brito, A. and Almeida, H., 2020. Privacy by Evidence: A Methodology to develop privacy-friendly software applications. Information Sciences, 527, pp.294-310.
[6] Hatzivasilis, G., Papaefstathiou, I. and Manifavas, C., 2016. Software security, privacy, and dependability: Metrics and measurement. IEEE Software, 33(4), pp.46-54.
[7] Baldassarre, M.T., Barletta, V.S., Caivano, D. and Scalera, M., 2019. Privacy oriented software development. In Quality of Information and Communications Technology: 12th International Conference, QUATIC 2019, Ciudad Real, Spain, September 11–13, 2019, Proceedings 12 (pp. 18-32). Springer International Publishing.
[8] Hadar, I., Hasson, T., Ayalon, O., Toch, E., Birnhack, M., Sherman, S. and Balissa, A., 2018. Privacy by designers: software developers’ privacy mindset. Empirical Software Engineering, 23, pp.259-289.
[9] A literature study on privacy patterns research. In 2017 43rd Euromicro Conference on Software Engineering and Advanced Applications (SEAA) (pp.194-201). IEEE.SDN): A Systematic Literature Review. Electronics, 12(14), 3077.
[10] Kemmer, F., Reich, C., Knahl, M. and Clarke, N., 2016, April. Software defines privacy. In 2016 IEEE International Conference on Cloud Engineering Workshop (IC2EW) (pp. 25-29). IEEE