Journal of Advances in Developmental Research (IJAIDR)

E-ISSN: 0976-4844 ● Website: www.ijaidr.com ● Email: editor@ijaidr.com

Protecting Software Code Against Cyber Attack Through Novel Watermarking Approach

Bharati Devidas Patil[1], Prof. Dr. Monika Tripathi[2]

[1]Research Scholar, Department of Computer Science and Engineering, P. K. University, Shivpuri 473665, Madhya Pradesh, India

[2]Research Supervisor, Department of Computer Science and Engineering, P. K. University, Shivpuri 473665, Madhya Pradesh, India

Abstract

The speed at which cyberattacks are developing is unsettling. These days, ransomware assaults, malware, phishing, crypto-jacking, and data breaches are common. With more software being used in every aspect of life, the software industry is expanding in this age of cyberwarfare. This development has made it more difficult for consumers and software suppliers to stop a variety of attacks. The software detection rate of current watermark detection systems is low. This research suggests a revolutionary blind Zero code-based Watermark detection technique called KeySplitWatermark to overcome this problem and safeguard software from cyberattacks. Using the intrinsic characteristics of code, the algorithm logically incorporates a watermark and provides a reliable solution. Keywords are used by the embedding process to create code segments that are key-dependent on the watermark. This key is used by the extraction algorithms to eliminate watermarks and identify manipulation. The original program code is restored when tampering reaches a user-defined threshold, making it impervious to attacks. KeySplitWatermark produces up to 15.95 separate watermarks by evaluating tampering attacks on three different watermarks. The results demonstrate that the suggested strategy reports encouraging effects against potent and feasible cyberattacks. We used two distinct software programs to compare our proposal's performance with cutting-edge research.

Keywords: Software Watermarking, KeySpilt Approach, Watermark Embedding, Watermark Extraction

Introduction

With the 21st century's massive processing power, fast internet, Internet of Things, and Blockchain technology, Bitcoins can be used for commerce. It demonstrates how widely available digital content is across a variety of connected devices. People in the modern electronic age share information instantly, but they also have to deal with the growing threat of cyberattacks on their data, software, systems, devices, and services. Data breaches, software, malware, smart bot (DDoS) assaults, digital forgeries, and cyber fraud are all quite frequent. Software, databases, photos, music, videos, and webpages are examples of digital artefacts that be quickly developed and widely shared online. By taking use of known software flaws, hackers attempt to breach the system's security layer. They then launch attacks employing viruses, malware, Trojan horses, logic bombs, and backdoors. Malicious attackers alter program codes to produce malware and damage software systems. 99.9% of apps on third-party app marketplaces are malicious,

according to Symantec's 2018 Internet Security Threat Report [2]. Preventing cyberattacks requires protecting software and making it more resilient. Since software is the basis of all communication infrastructures, including computers and the internet, it is a crucial digital component. Software applications for communication, education, commerce, health care, cloud computing, and many more fields are being developed at a rapid pace. However, software can be quickly hacked and used for very other or harmful objectives. Typical Exposures and Vulnerabilities. In the software watermarking sector and for software cybersecurity, Figure 1 describes the process of watermarking. Using a blind code-based Zero Watermark technique, the suggested embedding and extraction procedures are detailed in the original source code. The verifier will carry out the watermark detection once the claimer first jumbles the source code and sends the watermark positions. The original watermark positions cannot be determined by the verifier if the scrambling parameters are not made available.

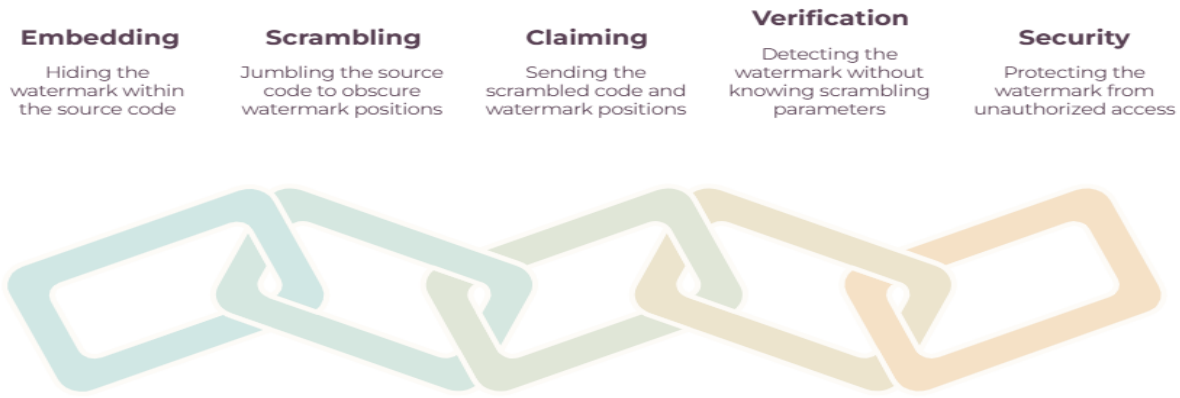

Figure 1 Blind Code based watermarking Process

The structure and watermark of the software code are used by the zero watermarking embedding algorithm [5] to create a key. The original watermark and this key need to be registered with the Certification Authority (CA). If an attack is detected based on user preferences, the extraction algorithm extracts the key from the source code of the altered software and can detect tampering by comparing the key with the CA. The original software code will be restored in the event of tampering if the payload exceeds a certain threshold. Three distinct software samples and two distinct watermarks are used to test the KeySplitWatermark. The algorithm's resilience to attacks is demonstrated by experimental results. To defend software source code from cyberattacks, we present KeySplitWatermark, a unique method based on blind zero watermarking. The algorithm, which is blind, uses the intrinsic characteristics of code to logically add a watermark and produces a reliable result. The embedding algorithm and the extraction algorithm are the two components of the algorithm. The approach uses a watermark to generate a key and is able to retrieve the software's key even after it has been compromised or attacked. If software is attacked and tampering is found, the original code can be restored, negating the impact of the attack. In particular, this work contributes in the following ways to the software watermarking and cybersecurity communities: 1) A new method based on watermarking that shields software code from attacks without changing the code to incorporate the watermark.

2) No presumptions are made regarding the length, programming language, or software code.
3) A new reactive strategy to boost software resilience in the face of cyberattacks.
4) If tampering reaches a user-defined threshold, cyberattacks on software code can be identified and the original code can be restored.

Literature Survey

community for software development and security. Since software watermarking is a developing field of study and several software companies claim to be developing secure software codes, cyberattacks on software are fairly popular these days. As more and more software vulnerabilities are revealed, a strong, effective, and durable software watermarking solution could be revolutionary. Software protection, content authentication, integrity checking, and fingerprinting can all benefit from digital watermarking. There are a number of guidelines and methods for developing secure software [3]. An overview of current procedures, standards, frameworks, life-cycle models, and approaches that either support or might support safe software development was given by Davis [5].

In order to create secure software, McGraw proposed seven touchpoint activity areas that are connected to software development artefacts. Software code vulnerabilities have also been identified using a vulnerability scanning technique based on an automated pattern-matching tool. In addition to offering a standard Secure Software Development Life Cycle, OWASP has recently recommended the top 10 security controls for web application developers and assists developers in understanding what should be taken into account or best practices at each stage of a development life cycle. A quick summary of the software protection tools that are now available is shown in Figure 3. Over the past few decades, numerous software watermarking techniques have been put forth. The proactive methods to software security that are now in use concentrate on creating safe software by adhering to all phases of the secure software development life cycle. Over the past 20 years, software watermarking has been accomplished in a number of ways, including both static and dynamic methods. Software code, FP trees, registration allocation, graph-based, dynamic path, and many more are the foundation of these methods. The following is a grouping of some of the main techniques:

A. ALGORITHMS FOR REORDERING

Re-ordering methods are static software watermarking algorithms that insert a watermark in a permutation of the existing code using transformations that preserve semantics. In 1996, Davidson and Myhrvold presented the initial block reordering algorithm. Later, Myles et al. used Sandmark to assess this algorithm's performance on Java byte code. Gong et al. presented a Java watermarking technique that rearranges the indexes to contain a watermark after analyzing the Java class file's format.

B. ALGORITHMS FOR REGISTER ALLOCATION

One method of constraint-based static software watermarking is register allocation. Based on this idea, Qu and Potkonjak proposed a QP technique that uses register allocation to solve the graph colouring problem. The QPS method was then implemented in Sand Mark by Myles and Collberg, who also conducted an empirical evaluation of the technique. Zhu and Thomborson later suggested an additional enhancement that they refer to as the QPI algorithm.

C. SPREAD-SPECTRUM METHODS

These techniques make use of concepts from radio communications using spread spectrum. The concept of adding a watermark to data's spectral components was put forth by Cox et al. A strong object watermarking technique that was more resistant to collusion assaults was later presented by Stern et al.

D. OPAQUE PREDICATE METHODS

In 1998, Collberg et al. introduced the concept of opaque constructions. Two techniques for water-marking Java programs that use opaque predicates were proposed by Arboit et al. Myles and Collberg later evaluated this approach by implementing both static and dynamic versions within the Sand-Mark framework.

E. ABSTRACT INTERPRETATION METHODS

Software verification is one application for abstract interpretation, a static analysis technique. An abstract interpretation approach that encoded the watermark in values supplied to specific integer local variables at run time was published by Cousot & Cousot. In their semantic approach to software watermarking, Preda and Pasqua modelled the attacker's capacity to recognise the signature as a completeness characteristic inside the framework of abstract interpretation.

F. ALGORITHMS FOR CODE REPLACEMENT

The concept of code replacement—that is, replacing the predetermined portion of code with the watermark value—was employed in the first patented software watermarking attempts. Menden et al. investigated the watermarking of Java programs and put forth a number of methods that involve switching byte codes within dummy methods (which are implemented as jmark). Software watermarking is one method that can be used to secure software after an attack occurs and prevent damages. Protecting software from cyberattacks is essential. None of the current approaches to software security concentrate on making software resilient after an attack has already occurred; instead, they are proactive and concentrate more on developing secure code.

Methodology

One of the most significant issues facing the digital community is safeguarding software from cyberattacks. Software protection can now be achieved by watermarking, which was originally employed for copyright protection. Software watermarking is the process of adding a watermark to software that can be used to uniquely identify the software's copy-right owner. In addition to proving ownership, software watermarking detects manipulation. An impending cyberattack may be clearly indicated if tampering reaches a particular threshold or exhibits a particular pattern. The software may become malware or a bot utilized in the next cyberattack. The original version of the software (registered with CA in the name of the copyright owner) can be substituted for the tampered version if tampering is discovered in real time. An attacker won't be able to initiate an assault in this manner. Some of the difficulties and characteristics that every software watermarking method must take into account are the availability of numerous programming languages for software, its executable form, and its dynamic interpretation and storage. It is also necessary to address the essential needs of a standard watermarking technique, such as security, capacity, imperceptibility, and robustness.

We suggest new watermarking-based strategies to defend computer programs against intrusions. KeySplit-Watermark divides the code according to the chosen keyword after first analyzing the software code to find the key words. The algorithm uses both the program code and the keywords to create a unique key. This key can be used to prove ownership in the event that a copyright issue arises in the future. The extraction algorithms are blind since they do not require a watermark as input, and the embedding algorithm does not interfere with software code to watermark it.
he KeySplitWatermark embedding algorithm requires the following inputs, as shown in Figure2:

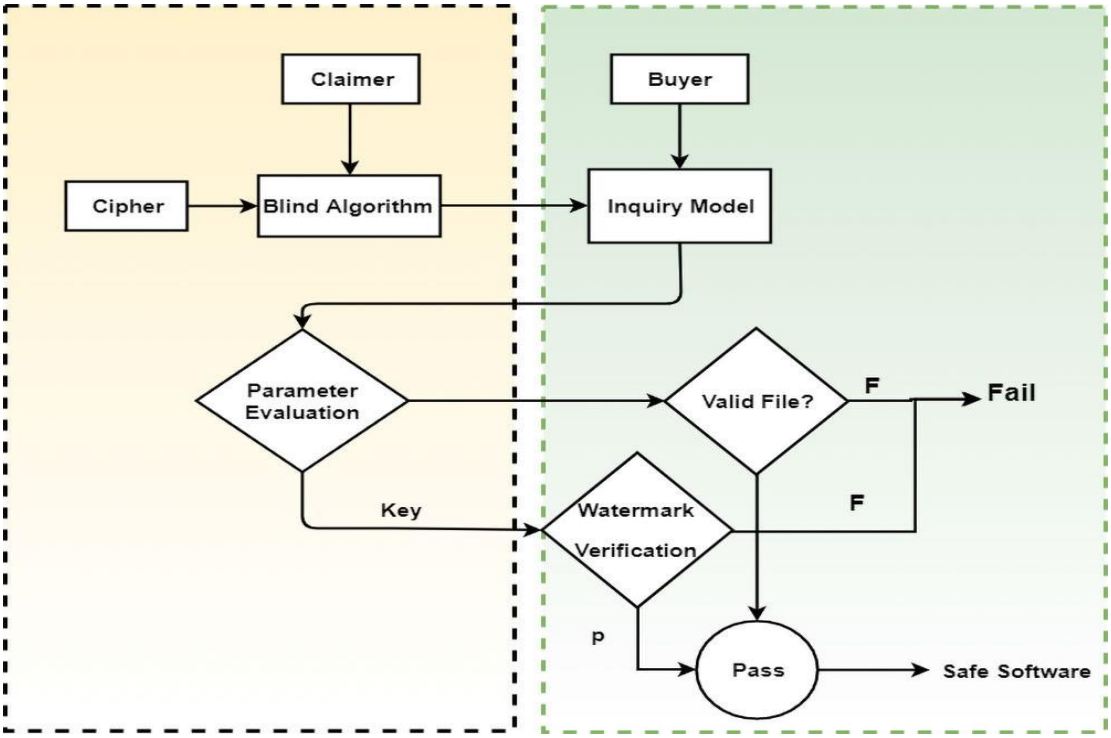

Figure 2:Watermarking algorithm using KeySpilt Approach

1) Original code: The software code that needs to be watermarked.
2) Cypher: A digital value utilized in the process of creating a key.
3) Watermark: A collection of ASCII characters.

The owner key is produced as an output by the embedding process. The CA records that key, which is subsequently used to extract the watermark (if necessary). The following inputs are fed into the extraction algorithm:
1) Attacked code file: Software code files that have been altered or used unlawfully in violation of copyright.
2) Owner key: To identify the original owner, it is obtained from the certifying authority.
This technique, which registers the contents in the name of the copyright owner, requires a reliable Certificate Authority (CA). This reliable third-party extracts watermarks whenever an attack is suspected, and if tampering is found, it offers the original software code for restoration. The altered code is swapped out for the original code, rendering the attacker's actions void. The two components of the watermarking

algorithm are watermark extraction and watermark embedding. The software's original owner embeds the watermark, and a reliable third party extracts it thereafter.

The extraction technique extracts the watermark after receiving two inputs: 1) the owner key and 2) the attacked code. In order to restore the original code, this watermark thereafter verifies the original owner's identification.

A. EMBEDDING METHODS

The embedding algorithm creates an owner key using the program's programming language keywords and inserts the watermark into the software code or program. The following is a full description of the watermark embedding algorithm: This approach initially preprocesses software code to get the top ten characters and top five keywords. The user-selected keyword is then used to divide it. The MOC list is then filled with the maximum occurring alphabetical character found in each partition. The owner key is generated using this MOC list and the supplied watermark. Every letter in the watermark is compared to every letter in the MOC list; if they match, the key is filled with the partition number (PN) and digit 0, which indicates the direct approach. The shift ciphering method (SC) is used to fill in the watermark if the letter is not present in the MOC list; however, the first digit 1 denotes an indirect method. The owner key is then created by combining the keyword with the key (OK). The owner key (OK) and original watermark (W) are registered.

Algorithm 1 Working of Embedding Algorithm (z_{26})

Represents the 26 Alphabets (a-Z)

---

1 Input C
2 Preprocess C
3 Count occurrences of all characters
4 Count occurrences of all keywords
5 Display top 10 character list and input W
6 Display top 5 keywords and input KW
7 Partition C based on KW
8 Identifying MOC from each partition and populateMOC list
9 For each letter of W, repeat step 10
10 **if** $w_j \epsilon MOC$ **then**
11 key[i]=0;
12 where j=1,2,. . . wl keyi+1=PN(MOC)
13 **else if** $w_j \ / MOC$ **then**
14 key[i]=0; where j=1,2,. . . wl
15 keyi+1=PN(MOC)
16 **else**
17 key[1+1]=$(w_j k) Mod 26$ +
18 where k is in z_{26} –
19 OK = concatenate (KW, Key)
20 Output OK

W Watermark *KW* Keyword
SC Shift Cipher *OK* Owner Key
C Software source code *PN* Partition number
MOC Maximum Occurring Character

includes an original code, date, and time from a certification authority. If necessary, this code would eventually be used to replace the tampered code.

B. EXTRACTION METHOD

When necessary, the watermark is extracted from the software code via the extraction algorithm. It extracts the watermark from the attacked program code using the watermark key that was previously generated by the embedding process. The Certification Authority, a reliable third party, is in possession of this algorithm. The following is the extraction algorithm:

This approach uses the keyword (KW) derived from the owner key (OK) to split the attacked code (Ca). The maximum occurring character (MOC) from each partition is then determined, and the MOC list is filled in. The watermark is then obtained from the source code using the contents of the watermark key (WK). To demonstrate ownership, the extracted watermark can then be compared to the original watermark that was previously registered with CA using any pattern matching measure (such as similarity index). The original software code replaces the tampered code if matching exceeds a user-defined threshold or if a specific tampering pattern is discovered.

Algorithm 2 Working of Extraction Algorithm
1 Input C_a
2 Preprocess C_a
3 Partition C based on KW (Obtained from OK)
4 Identify MOC from each partition and make MOC list
5 L_1 = length (KW), keyindex = L_1 1, L_2=length(W)
6 **while** *keyindex* < L_2 **do**
7 **if** OK ++ 0 **then**
8 W_e(I)= MOC (PN)
9 **else**
10 $W_e(I) = ReverseSC$
11 Increment I
12 Output W_e

Ca Attacked code *KW* Keyword
W Watermark *We* Extracted Watermark
OK Owner Key *SC* Shift Cipher
MOC Maximum Occurring Character

Conclusion

In order to defend software code from cyberattacks, we introduced KeySplitWatermark, a revolutionary zero watermarking technique. The approach is blind and uses the intrinsic characteristics of code to logically add a watermark while producing a reliable result. The extraction process uses the source code and the user-supplied watermark to create a unique key that is recorded with the Certification Authority (CA) and then used to recover the original software code and identify the original owner. The watermark is logically embedded in our KeySplitWatermark; only the original owner and CA are aware of its presence. The watermark cannot be removed without drastically changing the code, and any changes made to the code result in the original code being reinstated. The results demonstrate that the KeySplitWatermark is reliable, safe, and effective with little computing needs. The algorithm's performance was assessed for tampering attacks conducted on three distinct code samples. The suggested algorithm fared better in terms of execution time, capacity, and size when compared to the pertinent work. Two watermarks and a small number of samples created in two programming languages were used in this work. This work can be expanded upon and assessed in the future for application-specific software codes written in different programming languages with varying keyword sets and counts.

REFERENCES:

1. A. R. Javed, M. O. Beg, M. Asim, T. Baker, and A. H. Al-Bayatti, ‘‘AlphaLogger: Detecting motion-based side-channel attack using smart- phone keystrokes,’’ J. Ambient Intell. Humanized Comput., pp. 1–14, Feb. 2020.
2. A. K. Abdulrahman and S. Ozturk, ‘‘A novel hybrid DCT and DWT based robust watermarking algorithm for color images,’’ Multimedia Tools Appl., vol. 78, no. 12, pp. 17027–17049, Jun. 2019.
3. W. Hu, R.-G. Zhou, J. Luo, and B. Liu, ‘‘LSBs-based quantum color images watermarking algorithm in edge region,’’ Quantum Inf. Process., vol. 18, no. 1, p. 16, Jan. 2019.
4. Z. Jalil and A. M. Mirza, ‘‘An invisible text watermarking algorithm using image watermark,’’ in Innovations in Computing Sciences and Software Engineering. Dordrecht, The Netherlands: Springer, 2010, pp. 147–152.
5. N. Davis, ‘‘Secure software development life cycle processes: A technol- ogy scouting report,’’ Carnegie-Mellon Univ., Pittsburgh, PA, USA, Tech. Rep. ADA447047, 2005.
6. M. E. Kumar, G. T. Reddy, K. Sudheer, M. Reddy, R. Kaluri, D. S. Rajput, and K. Lakshmanna, ‘‘Vehicle theft identification and intimation using gsm & iot,’’ in Proc. Mater. Sci. Eng. Conf., vol. 4, 2017, Art. no. 042062.
7. G. T. Reddy, R. Kaluri, P. K. Reddy, K. Lakshmanna, S. Koppu, and
8. D. S. Rajput, ‘‘A novel approach for home surveillance system using IoT adaptive security,’’ in Proc. Int. Conf. Sustain. Comput. Sci., Technol. Manage. (SUSCOM), vol. 3. Rajasthan, India: Amity Univ. Rajasthan, Feb. 2019, pp. 1616–1625, doi: 10.2139/ssrn.3356525.