IJITKM 0973-4414 Volume-8, Number-2 Jan-Jun 2015 pp. 229-235 **Impact Factor 2.8** available online at www.csjournalss.com

Server-Side Algorithms for WHLK Framework

Shubhnandan S. Jamwal, Nishant Gupta
jamwalsnj@gmail.com

Abstract: Software security and piracy is becoming more and more important issue in the age of Internet. Software piracy has been a direct threat to the revenue of software vendors, therefore, the need to employ effective and efficient techniques for detecting and preventing software piracy must be reevaluated. One of very important technique is software watermarking and using registration keys. This paper proposes the Server-side algorithms for registration and embedding the watermark into the software using WHLK approach [13]. We have tested the algorithms and the analysis of the proposed algorithms proves that these registration algorithms are more reliable and efficient in comparison to other techniques.
Keywords: Software watermarking, algorithms, software piracy, server-side registration.

INTRODUCTION

Software industry is making progress by leaps and bounds. In today's global marketplace, the way it is evolving not only leads neophytes to new opportunities but introduces commencing threats to software vendors. The curse of software piracy hampers software industry in software development process. Initially the main focus of software companies is to develop new and intuitive software. Now, they are spending most of the time, money and resources in researching and developing techniques to protect their intellectual property. Because of the increasing amount of attention directed towards piracy prevention, it is important for companies to know what can be done to stop the average user from attempting to pirate software.

According to Business Software Alliance (BSA), the amount of financial loss caused by software piracy in 2011 was more than 63,456 million U.S. dollars [1]. In addition, the rate of loss is growing with market size expansion. BSA estimated that worldwide software piracy rate went up to 42 percent in 2011. Total losses due to software piracy in India stood at a staggering figure of about INR 13,783 crores in 2011 showing about 63 per cent piracy rate in India [2]. Thus, many studies to protect software from piracy have been promoted and their achievements have emerged as hardware-based and software-based approaches. Some of the common software protection measures are:

Intellectual property protection: The objective is to link the software with information about its author by using techniques such as watermarking [3].

Protection against function analysis: The objective here is to prevent a malicious host from discovering what function is being computed by a software element. Techniques such as code obfuscation or function hiding are used, sometimes complemented by the use of hardware tokens.

Software use-control: This is aimed at guaranteeing that only authorized users can run the software according to some contractual conditions. Our paper implemented new algorithms comprising of integration of software watermarking and use of registration code.

LITERATURE REVIEW

Numerous studies have been conducted in the past few years regarding global software piracy.

In 2005, William Zhu et. al. [3] in his paper gives a brief overview of software watermarking. This paper also describes the taxonomy, attack models, and algorithms of software watermarking.

Qiang Liu [4] in his paper focused on some special protective approaches using some exterior components (such as smart cards, hardware security unit and dongles) to make more difficult for unauthorized use of software. The paper also compared these approaches and offer some insight to the different approaches adopted. But it needed to bind the software with exterior component (hardware). The cost of exterior component has to be added to the cost of software which seems to be unreasonable.

Yawei Zhang et.al [5] proposes a software-splitting technique which put the split contents on the client instead of the remote trust server. This new technique would encrypt the extracted contents from the software by a key relating to the hardware characteristics, and then decrypt them dynamically during the main program running. In spite of the usefulness of this technique, it is not stimulating for some explanation programming languages such as VB, JAVA because it's nearly impossible to directly manipulate in the memory for these programming languages.

Petar Djekic and Claudia Loebbecke [6] in their paper studied the influence of technical copy protections on application software piracy. Scientific research has been empirically investigated to what extent technical copy protections avoid illegal copying. The research findings can not confirm to be also applicable to similar industry-specific software like graphics application software.

Vineet Kumar Sharma et. al. [7] identifies the flaws in the existing defense mechanisms and examines social and technical challenges associated with handling software piracy prevention. The paper also characterized the currently used static licensing structure in software distribution along with its drawback and proposes a dynamic software licensing scheme which comprises software serial keys integrated with hardware token to provide an ethical optimal utilization of software to customer organization and initiates to stop software piracy.

Ibrahim Kamel, Qutaiba Albluwi [8] advocates protecting software copyright through hiding watermarks in various data structures used by the code, e.g., Bþ-trees, R-trees, linked lists, etc. The paper provides a detailed security analysis and performance evaluation to show that the embedded watermarks are robust and can withstand various types of attacks. This technique can be applied to other data structures that do not put restriction on the order of the data; however, it is not suitable for data structures that are sensitive to the order of the data like, Bþ-trees, Quad-trees, k-d tree, etc. in addition to memory-resident data structures like stacks, linked lists, arrays, binary trees, etc.

Aniket Kulkarni, Sachin Lodha [9] examined software protection through code obfuscation technique which resists reverse engineering attacks. Although the scheme requires high development efforts to construct obfuscated code, it can be useful to obfuscate code which implements secret functionality such as license checking mechanism.

Various strategies have been employed to make unauthorized duplication and use of software more difficult. One such approach by Ireneusz J. Jozwiak and Krzysztof Marczak [10] is to provide a hardware "key" which is typically installed in the parallel port or USB of the computer to provide a software interlock. If the key is not in place, the software will not execute. This method is relatively expensive for the developer and cumbersome for the authorized user while remaining vulnerable to the theft conducting piracy by duplication of the hardware key. Another approach requires the user to enter a serial number or customer identification number during the installation of the software. Missing or invalid registration information prevents the installation of the software. This approach is easily defeated by transferring the serial number or customer identification number to one or more unauthorized users. Yet another approach requires registering the software with the manufacturer or distributor to obtain an operational code or password necessary for the installation of the software. Again, once the operational code or password is obtained, it may be perpetually transferred along with pirated copies to numerous unauthorized users.

Ajay Nehra et. al. [11] proposed a SMS gateway based technique, checking the authenticity of the Software at every fixed time interval giving it advantage to identify fake unauthorized users. Also, blocking such installed software save software companies from huge loss. But there is a need to find an automation process as a manual response for each software on the client machine will be a tedious task.

OBSERVATION

WHLK Model [12] is developed and implemented on the basis of analysis of findings from a survey conducted on number of stakeholders involved in academic as well as industrial sector. The results of this exploratory study clearly show that users are well aware of software piracy. Even though the pirated software does not fulfill requirements of the user but the involvement in software piracy is highly influential. A very high percentage of users are acquiring software through their respective vendors who load the pirated copies of software in their machines. This harddisk loading type of piracy shows the highest percentage among all types of piracy. Even the core factors like risk, market, morality and education are not able to affect the software pirates.

The findings clearly show that most of the software is acquired through piracy instead of purchasing them. The queries generated in the questionnaire related to software section resulted in very high percentage of software acquiring through different types of software piracy. Most of the software under different categories i.e. operating system, computer languages, office, databases, utilities, and educational are acquired by the users through one type of piracy or the other. Results shows that 98% of software is pirated which give a different picture of piracy rates in India as compared to BSA reports of 63% piracy rates in 2011.

In contrast to other techniques, our paper shows the implementation of the server-side algorithms during the purchase and registration of software.

WHLK MODEL

The implementation and testing of WHLK Model [13], which introduced an integration of Software Watermarking, Hardware Parameters and License Key, has been proposed that can greatly reduce unauthorized use of software. Noting the weakness in existing approaches of software registration, we incorporated the new processes in the process of software registration. First, instead of preventing from illicit copying, we force the software to be only used by an authorized running environment. In order to accomplish this task, the user is required to submit his identification details which are being applied through algorithms to generate random unique key (string) and this key is embedded as a watermark in the program. Also, an automated program fetches the uniquely identifying characteristics of running environment such as the physical sequence numbers of the CPU, the main board, processor and other hardware information. These hardware characteristics and the License key are integrated to be used as parameters. All of these parameters are encrypted using a pre-defined code generating a new random unique number called as Registration Code (RGCN) which is acknowledged to the user. This model provides an integration of the privacy of users, security of information and license key, together for the control of the piracy. It enjoys a modular design and can be implemented by any machine with flexible configurations and windows X operating systems. Furthermore, the proposal allows flexible registration information definition. This Model not only makes it harder to create an additional available copy based on diversity, but also prevents illegal uses on the copy.

SERVER SIDE REGISTRATION ALGORITHMS

A. SOFTWARE WATERMARKING ALGORITHM

Software Watermarking Algorithm 1.0 is implemented at vendor's side when user purchases the software. The personal information of user viz. name, affiliation and identification number are input as parameters for generating a unique key. This unique key is used to embed a watermark in the code of software. ***CT (Collberg Thomborson)*** algorithm is applied on this unique key and embeds the watermark in the class file. After successfully applying this algorithm, another algorithm namely ***StringEncodeObfuscator*** is applied to avoid reverse engineering. The watermark embedded in the class file is secure and cannot be altered or tampered due to obfuscated file.

Algorithm 1.0:	**(Software Watermark Algorithm) Input the User name N, affiliation A and identification number I as string variables, this algorithm generates a unique key UK and embed watermark in the software using UK.**
Step 1.	**Read: N, A, I from the user.**
Step 2.	**Generate UK.**
Step 3.	**Embed watermark using CT algorithm.**
Step 4.	**String Encode Obfuscator algorithm is applied.**
Step 5.	**Exit.**

B. NEW REGISTRATION ALGORITHM

Registration Algorithm is designed for user to register the software first time on new machine. While registering the software on computer machine, the ***New Registration Algorithm 1.1*** fetches the hardware characteristics of machine. These characteristics include the CPU ID, harddisk ID, MAC ID, and Processor Serial Number. These values are concatenated and stored in a string. This string variable is verified by the server in the database. If this variable exists in database, ***Algorithm 1.4*** is applied else RNGCryptoServiceProvider is applied for generating unique key. A unique key is generated by the server. Then the user inputs his E-Mail ID and License key which are being submitted to the server. This license key is concatenated with unique key and the result is stored in another string variable. The concatenated string is the registration code. Then the system date has been generated. The registration code is stored in the database and also acknowledged to the e-mail id of the user. Lastly, the database has been updated and new registration process completes its cycle.

Algorithm 1.1: **(NewRegistration Algorithm) Fetch the CPU ID (CID), Harddisk ID (HDID), MAC ID (MID) and Processor Serial Number (PSN) of the machine, this algorithm generates the unique key UK and concatenates it with License Key (LK) to update registration code database RGCNDB and send to client.**

Step 1. Read CID, HDID, MID, PSN from the machine.
Step 2. HWD : = Concatenate (CID,HDID,MID,PSN).
Step 3. Submit HWD to server for verification.
Step 4. If HWD exists, then:
Apply Re-registration algorithm 1.4.
Go To Step 5.
Else:
RNGCrptoServiceProvider is applied.
Server generates UK.
Input EID and LK.
RGCN : = Concatenate (UK,LK).
Read: SysDate : = DateTime().
Update RGCNDB.
ENDIF.

C. RE-REGISTRATION ALGORITHM IF HARDWARE CHANGES

Another algorithm is designed to manage the changes in the hardware of machine of the user. If the user has changed the hardware of the machine, he needs to re-register the software and in this case ***Re-registration Algorithm 1.2*** is applied. Three parameters user name, affiliation and ID are concatenated and stored in a string variable. This string variable is verified by the server in the database. If the variable does not exist, ***NewRegistration algorithm 1.1*** is applied else an algorithm is applied which fetches the hardware characteristics of machine. These characteristics include the CPU ID, harddisk ID, MAC ID, and Processor Serial Number. These values are concatenated and stored in a string variable. This string variable is verified by the server in the database. If this variable exists in the specified field of database, ***Algorithm 1.4*** is applied else RNGCryptoServiceProvider is applied for generating unique key. A unique key is generated by the server. Then the user inputs his E-Mail ID and License key which are being submitted to the server. This license key is concatenated with unique key and the result is stored in another string variable. The concatenated string is the registration code. Then the system date has been generated. The registration code is stored in the database and also acknowledged to the e-mail id of the user. Lastly, the database has been updated and re-registration process completes its cycle.

Algorithm 1.2: **(Re-registration Algorithm if machine hardware changes)**
Input the User name N, affiliation A and identification number I as string variables, this algorithm fetches the CPU ID (CID), Harddisk ID (HDID), MAC ID (MID) and Processor Serial Number (PSN) of the machine generating the unique key UK and concatenates it with License Key (LK) to update registration code database RGCNDB and send to the client.

Step 1. Read N, A, I from the user.
Step 2. Set X : = Concatenate (N,A,I).
Step 3. Submit X to server for verification.
Step 4. If X do not exists, then:
Apply NewRegistration algorithm 1.1.
Go To Step 5.
Else:
Read CID, HDID, MID, PSN from the machine.
HWD= Concatenate (CID,HDID,MID,PSN).
Submit HWD to server for verification.
If HWD exists, then:
Go To Step 5.
Else:
RNGCrptoServiceProvider is applied on HWD.
Server generates UK.
Input EID and LK.
RGCN : = Concatenate (UK,LK).
Read SysDate : = DateTime().
Update RGCNDB.
ENDIF.
ENDIF.
Step 5. Exit.

RNGCryptoServiceProvider is applied for generating unique key. A unique key is generated by the server. Then the user inputs his E-Mail ID and License key which are being submitted to the server. This license key is concatenated with unique key and the result is stored in another string variable. The concatenated string is the registration code. Then the system date has been generated. The registration code is stored in the database and also acknowledged to the e-mail id of the user. Lastly, the database has been updated and re-registration process completes its cycle.

```
Algorithm 1.3:   (Re-registration Algorithm for same machine in case user does not have
                 details) Input the User name N, affiliation A and identification number I as
                 string variables, this algorithm fetches the CPU ID (CID), Harddisk ID
                 (HDID), MAC ID (MID) and Processor Serial Number (PSN) of the
                 machine generating the unique key UK and concatenates it with License
                 Key (LK) to update registration code database RGCNDB and send to client.
Step 1.          Read: N, A, I from the user.
Step 2.          Set X : = Concatenate (N,A,I).
Step 3.          Submit X to server for verification.
Step 4.          If X do not exists, then:
                                 Apply NewRegistration algorithm 1.1.
                                 Go To Step 5.
                 Else:
                         Read: CID, HDID, MID, PSN from the machine.
                         HWD= Concatenate (CID,HDID,MID,PSN).
                         Submit HWD to server for verification.
                         If HWD do not exists, then:
                                 Apply algorithm 1.4.
                                 Go To Step 5.
                         Else:
                                 RNGCrptoServiceProvider is applied.
                                 Server generates UK.
                                 Input EID and LK.
                                 RGCN : = Concatenate (UK,LK).
                                 Read SysDate : = DateTime().
                                 Update RGCNDB in the database.
                         ENDIF.
                 ENDIF.
Step 5.          Exit.
```

E. RE-REGISTRATION ALGORITHM FOR SAME MACHINE IF DETAILS AVAILABLE

This algorithm is designed to manage the re-registration process of software on same machine if details are available with the user. In case of availability of registration code with the user while re-registering the software, **Re-registration algorithm 1.4** is applied. User inputs the registration code and server verifies the code for its existence. If the code does not exist ***Re-registration Algorithm 1.3*** is applied else a System Date is generated. Database is updated and re-registration process completes its cycle.

```
Algorithm 1.4:   (Re-registration Algorithm for same machine in case user has details) This
                 algorithm inputs the registration code RGCN and re-register the software till
                 next update.
Step 1.          Input RGCN.
Step 2.          Submit RGCN to server for verification.
Step 3.          If RGCN do not exists, Then:
                         Apply Algorithm 1.3
                         Go To Step 4.
                 Else:
                         Read: SysDate : =
                         DateTime().
                         Update RGCNDB.
                 ENDIF.
Step 4.          Exit.
```

```
Algorithm 1.5:   (Time-frame Algorithm) This algorithm reads current System Date
                 DBDate and finds the re-verification time by finding difference DE
                 between SysDate and DBDate.
Step 1.          Read: SysDate from RGCNDB.
Step 2.          Read: DBDate=DateTime().
Step 3.          TimeSpan ts= DBDate – SysDate.
Step 4.          Int DE = ts.Days.
Step 5.          String dt = DE.ToString().
Step 6.          If dt.StartsWith("-"), Then:
                         Process.kill().
                         Go To Step 7.
                 Else
                         Submit update.
                         DE=DE + 30.
                 ENDIF.
Step 5.          Exit.
```

F. TIME-FRAME ALGORITHM

The time-frame algorithm is designed to block the software on the basis of time span to secure the software from being used elsewhere. The **time-frame algorithm 1.5** is applied for blocking the software if time span between the system date of registration ***SysDate*** and current system date ***DBDate*** is more than 30. This is done by calculating the time span between SysDate and DBDate. If the time span is showing the positive value less than 30, it means that the system date is correct and user has still days to update the software. In case, the time span is showing the negative value meaning that the system date is incorrect and needs to be corrected. If it is corrected, user can update the software but if the time span is still showing the negative value, the process generated kills the process itself after some stipulated time meaning that software terminates itself. Software does not run but will be blocked till the user re-registers again with the correct details.

CONCLUSION

WHLK Model has been implemented on machines with variant configurations. Client-side and server-side phases of this model have been put to test on these machines and results observed are justifying the objectives of our research. This model not only makes it harder to create an additional available copy based on diversity, but also prevents illegal uses on the copy.

The Server-side verification algorithms were designed and implemented on various machines. It was observed that the newly designed algorithm at server side woks accurately and the chances of pirating the software copy are reduced to nil.

REFERENCES

[1] Business Software Alliance, 2012. Ninth Annual BSA Global Survey of PC User Attitudes.[Online]. Available: (http://globalstudy.bsa.org/2011/downloads/study_pdf/2011_BSA_Piracy_Study-Standard.pdf)

[2] Business Software Alliance, 2012. Ninth Annual BSA Global Survey of PC User Attitudes.[Online]. Available: (http://globalstudy.bsa.org/2011/downloads/study_pdf/pr_india_en.pdf)

[3] William Zhu, Clark Thomborson, and Fei-Yue Wang, "A Survey of Software Watermarking", *in Proc. Of IEEE Int. Conf. on Intelligence and Security Informatics,* Springer-Verlag Berlin Heidelberg, Germany. LNCS 3495, pp. 454-458, 2005.

[4] Qiang Liu," Techniques Using Exterior Component against Software Piracy", Department of Computer Science, University of Auckland, New Zealand.

[5] Yawei Zhang, Lei Jin, Xiaojun Ye, and Dongqing Chen," Software Piracy Prevention: Splitting on Client", *in Proc. of Int. Conf. on Security Technology*, IEEE Computer Society, Washington, DC, USA, pp. 62-65, 2008.

[6] Petar Djekic and Claudia Loebbecke," Preventing application software piracy: An empirical investigation of technical copy protections", in Journal of Strategic Information Systems, Elsevier, vol. 16, pp. 173-186, 2007.

[7] Vineet Kumar Sharma, S.A.M. Rizvi, S.Zeeshan Hussain, and Anurag Singh Chauhan, "Dynamic Software License Key Management Using Smart Cards", *in Proc. Of Int. Conf. on Advances in Computer Engineering*, IEEE Computer Society, pp. 244-246, 2010.

[8] Ibrahim Kamel, Qutaiba Albluwi, "A robust software watermarking for copyright protection", Computers and Security, Elsevier, vol. 28, no.6, pp. 395-409, 2009.

[9] Aniket Kulkarni, Sachin Lodha," Software Protection through Code Obfuscation. Project Report. Tata Research Development and Design Centre, Pune (2012).

[10] Ireneusz J. Jozwiak and Krzysztof Marczak, "A Hardware-Based Software Protection Systems – Analysis of Security Dongles with Time Meters", *in Proc. Of 2nd Int. Conf. on Dependability of Computer Systems*, IEEE Computer Society Washington, DC USA, pp. 254-261, 2007.

[11] A. Nehra, R. Meena, D. Sohu, and O.P. Rishi, "A Robust Approach to Prevent Software Piracy, *in Proc. Students Conference on Engineering and Systems*, pp.1-3, IEEE, March 2012.

[12] Nishant Gupta, Shubhnandan S. Jamwal, Devanand Padha, " Watermark, Hardware Parameters and License Key: An Integrated Approach of Software Protection", in International Journal of Advanced Research in Computer Science and Software Engineering, vol. 3, no. 5, pp. 1285-1289, May 2013.

[13] Nishant Gupta, Shubhnandan S. Jamwal, Devanand Padha, "WHLK: Framework for Software Authentication and Protection", in IEEE African Journal of Computing & ICT, IEEE, Nigeria, vol. 7, no. 1, pp. 69-80, 2014.