The International Journal Of Engineering And Science (IJES)
|| Volume || 5 || Issue || 8 || Pages || PP -18-24 || 2016 ||
ISSN (e): 2319 – 1813 ISSN (p): 2319 – 1805

# Combating Software Piracy Using Code Encryption Technique

[1]Ogheneovo, E. E., [2]Japheth, R. B.
[1]*Senior Lecturer, Department of Computer Science, University of Port Harcourt, Port Harcourt, Nigeria*
[2]*Lecturer, Department of Mathematics/Computer Science, Niger Delta University, Yenagoa, Nigeria*

**ABSTRACT**
*Computer security is of great concern to users and corporate bodies now ever than before due to activities of criminals and hackers on the Internet. Software piracy and the breach of the copyright laws, intentionally or unintentionally is very common these days. Software piracy is a menace to software developers and computer users all over the world. Software hackers have become nuisance to many organizations, corporate bodies and government alike. Pirating software has caused lost of several billions US Dollars and the problem continued unabated. There have been a lot of security threats in recent past due to the activities of hackers. Several financial organizations and national securities have been threatened and even some have been compromised. In this paper, we proposed the code encryption technique for combating software piracy. Using C++ programming language to develop the code, the technique converts plain code to an encrypted form that cannot be understood by the hacker or intended hacker unless he has the key to encrypt or decode the encrypted data. Our result shows that using this technique, it will be difficult to pirate software after it has been released to intended user(s)..*

***Keywords*:** Software piracy, code encryption, computer software, computer security, software developers

Date of Submission: 17 May 2016 Date of Accepted: 05 July 2016

## I. INTRODUCTION

Software piracy is a generic term for the illicit duplication of copyrighted computer software [1]. Software piracy involves the use, reproduction or distribution of software without the permission of the software author [2]. In 2009 alone, piracy costs software industry about $51 billion US Dollars [3]. Software piracy continues to be a major economic concern for businesses and organizations. Given the high cost of producing software, development of technology for prevention of software piracy is important for the software industry. Software piracy can be defined as "copying and using commercial software purchased by someone else". Software piracy is illegal. Each pirated piece of software takes away from company profits, reducing funds for further software development initiatives. The roots of software piracy may lie in the early 1960s, when computer programs were freely distributed with mainframe hardware by hardware manufacturers (e.g. AT&T, Chase Manhattan Bank, General Electric and General Motors). In the late 1960s, manufacturers began selling their software separately from the required hardware [4][5].

Software piracy and the breach of the copyright laws, intentionally or unintentionally, can occur in numerous ways. The Business Software Alliance and International Data Corporation, BSA/IDC [6] identified ten forms of software piracy that are not necessarily mutually exclusive categories: softlifting, unrestricted client access, hard-disc loading, OEM Piracy/Unbundling, Commercial Use of Noncommercial Software, Counterfeiting, CD-R piracy, Internet Piracy, Manufacturing Plant Sale of Overruns and 'Scraps' and Renting. While the specific terms and conditions of the use of software are program specific, these categories are only effective when the person's behaviour breaches the software's license agreement. Software Piracy directly harms the firms producing the software [7]. This is because it acts as a disincentive for people to produce innovative technology since they are not guaranteed to benefit from their hard work [8][9]. This then impacts on the customers as the reduction in profits is passed onto the consumer in the form of higher prices. Not only does it hamper the development of software, it also reduces the exportation of the products [10]. This has a negative impact on the wealth of a country since the Software Industry can act as an economic driver. For instance it provides jobs, business opportunities and tax revenues.

The software industry also contributes to the world economy by advancing society through technological innovations (Business Software Alliance and International Data Corporation [BSA/IDC]) [11]. The negative impact software piracy has on peoples' lives has often led researchers to classify this form of behaviour as an immoral and illegal act [12]. Computer software is easy to pirate because it is relatively easy to copy and does not result in a degradation of the quality of the product. Researchers also hypothesize that people pirate computer software because of the high number of personal computers now available. Current illegal software in the US accounts for 25 - 50% of the software in use. Other countries often have levels of piracy well beyond that of the US. The different categories of software piracy are reflective of the fact that software piracy varies in terms of its degrees of intensity. The categories indicate that piracy can vary from one extreme of sharing the software with a friend (soft lifting) to the other extreme of duplicating and selling the unauthorized copies under the disguise that they are legal copies (counterfeiting). Therefore for the purpose of this research, software piracy is defined as an act that occurs when people make copies of the computer software without permission or they load the computer software onto more machines than the licensed agreement says they can. Examples of computer software are databases, security packages, PC Games and reference software [13].

Software piracy is an important issue to be tackled in the development of any software. Software companies have responded by either placing preventative or deterrent measures in place. Preventative measures are aimed at wearing the pirates down by putting in measures that make it hard to pirate such as coder cards and hardware locks. These controls are aimed at wearing the pirates down to reduce its appeal. Deterrent controls try to encourage people not to pirate software by threatening legal sanctions. Gopal and Sanders [14] found that only deterrent measures help save a company's profits. Al-Rafee and Cronan [15] state that it is evident that these two measures are not effective in combating software piracy since the respective companies are still facing increasing loses. In this paper, we proposed the code encryption technique for combating software piracy. The technique converts plain code to an encrypted form that cannot be understood by the hacker or intended hacker unless he has the key to encrypt or decode the encrypted data. Our result shows that using this technique, it will be difficult to pirate software after it has been released to intended user(s). This technique makes code harder to understand by intended hackers.

## II. RELATED WORK

Birrer et al. [16] evaluates the performance overhead of a program fragmentation engine and offers examination of its efficiency against reverse-engineering approaches. The experimental results show that program fragmentation has low overhead and is an effective approach to obscure disassembly of programs through two common disassembler/debugger tools. Metamorphic software protections include another layer of protection to conventional static obfuscation approaches, forcing reverse engineers to alter their attacks as the protection changes. Program fragmentation incorporates two obfuscation approaches, over viewing and obfuscated jump tables, into a novel, metamorphic protection. Segments of code are eliminated from the chief program flow and placed throughout memory, minimizing the locality of the program. These fragments move and are called using obfuscated jump tables which makes program execution hard.

Zeng et al. [17] considered the supply manufacturing venture networks data security and software protection and proposed an enterprise classified data security and software protection solution, to describe the enterprise data storage, transmission and application software installation authorization, license and so on, presented a time and machine code depending on MD5, AES encryption algorithm dynamic secret key the encryption approach, to protect the enterprise data confidentiality, integrity and availability, to attain the software installation restrictions and using restrictions. Kent (1980) proposed a software protection technique which deals with the security needs of software vendors like protection from software copying and modification (e.g. physical attacks by users, or program-based attacks). Techniques proposed to handle these requirements include physical Tamper- Resistant Modules (TRMs) and cryptographic techniques. One approach comprises of using encrypted programs, with instructions decrypted immediately preceding to execution. Kent also observed the dual of this issue like user needs that externally supplied software be confined in its access to local resources.

Gosler's software protection survey [18] investigates circa-1985 protection technologies which comprise of hardware security tools (e.g. dongles), floppy disc signatures (magnetic and physical), analysis denial approaches (e.g. anti-debug approaches, checksums, encrypted code) and slowing down interactive dynamic analysis. The main goal is on software copy prevention, but Gosler observed that the potency of resisting copying should be balanced by the potency of resisting software analysis (e.g. reverse engineering to study where to alter software and for protecting proprietary approaches) and that of software modification (to bypass security checks). Useful tampering is generally headed by reverse engineering. Gosler also described that one should anticipate that an opponent can execute dynamic analysis on the target software without discovery (e.g. using in-circuit emulators and simulators) and that in such scenario, due to repeated experiments, one should anticipate the opponent to win. Thus, the main goal of practical resistance is to construct such experiments "enormously arduous". Another proposal Gosler [19] is cycling software (e.g. through some forced obsolescence) at a rate faster than an opponent can wreak it; this expects the model of forced software renewal, who suggested hopeless pirates via forced updates and software aging). This technique is suitable where protection from attacks for a restricted time period suffices.

Kim [20] proposed the stochastic maintenance approach for software protection through the closed queuing system with the untrustworthy backups. The technique shows the theoretical software protection approach in the security viewpoint. If software application modules are denoted as backups under proposed structured design, the system can be overcome through the stochastic maintenance model with chief untrustworthy and random auxiliary spare resources with replacement strategies. Additionally, the practical approach of technology improvement in software engineering through the technology innovation tool called TRUZ.

## III. METHODOLOGY

The encryption technique which is also referred to as the hashing mechanism has a pre-stored hashed value of any software it is integrated to. Every time the software is accessed, it computes a hash value and compares it with the existing pre-stored hash value. If both hash values are the same, the software runs if not it terminates. Apart from this feature, it also converts all characters of the serial number to hexadecimal mashed data so it is very difficult to know the serial number needed to access the software. figure 1 shows an architecture of the encryption mechanism in which a plain program that is easy to understand after development by the developers is converted to encrypted form which is hard to understand and can only be decoded by someone having the decryption key.

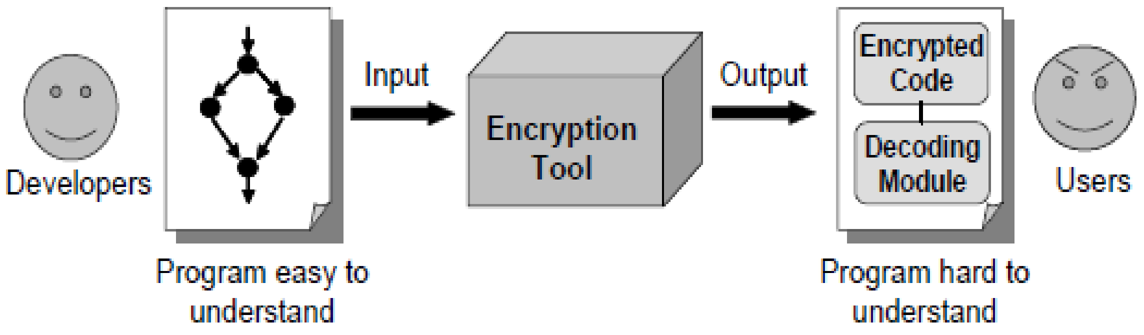


**Fig. 1:** Architecture of the encryption mechanism

The technique has a pre-stored hash value of any software it is integrated to. Every time the software is accessed, it computes a hash value and compares it with the existing pre-stored hash value. If both has values are the same, the software runs otherwise it terminates. Apart from this feature, it also converts all characters of the serial number to hexadecimal or mashed data so it is very difficult to know the serial number needed to access the software. Figure 1 shows the architecture of the model. It contains the following parts.

**Plain Code:** The plain code is the normal code written in English-like form using any programming language. In this research, the code was written using C++; which of course is easy to understand even by non-expert programmers.

**Encryption Tool:** This is the tool we used to perform the encryption of the program. The encryption tool converts the plain code to an encrypted form that is more difficult and harder for a cracker to understand.

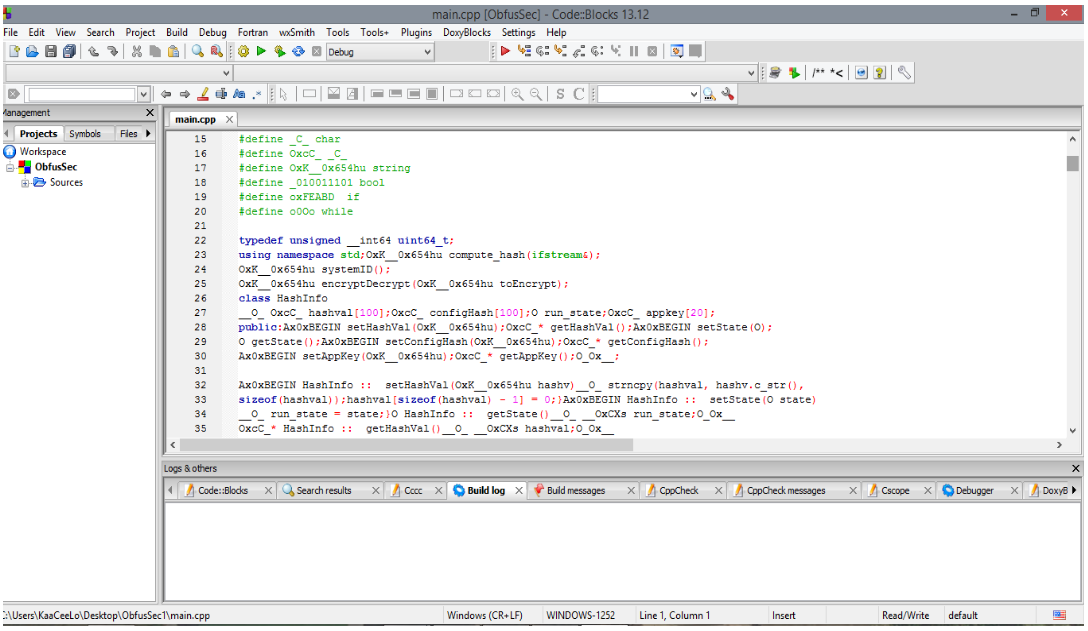
**Encrypted Code:** Using our encryption tool, the plain code is converted to encrypted code which is difficult to read even by an expert in programming. The encrypted code can only be read and understood when the right key is used to decode the encryption. Usually, it is difficult for a cracker to understand encrypted program. Once a cracker finds it difficult to understand the encrypted program, it means the chances of cracking such a program are very unlikely.

**Decoding Module:** In this module, the encrypted code is converted to the form that is easy to read using the decrypted key. The decoded form of the code is now converted back to the plain form which is now easy to understand and read.

Then, we use an encryption technique, known as one - way hashing, to generate our application licenses. The serial number generation is hidden using encryption. It is a code transformation technique where the functionality of the code is maintained while the code is mangled such that it cannot easily be understood by a software cracker. For the first installation, the software will require that a serial key is entered into the software via the interactive section. If the valid key is entered, the game grants you access and you can play it. However, if the serial number entered is invalid, the game will not grant access. The serial number code segment was compiled and an executable file known as keygen.exe was generated. When double-clicked, the keygen.exe, it automatically generates the serial number.

## IV. RESULTS AND DISCUSSION

We implemented the work using the Windows 8 operating system of 64-bit, 2 GB RAM, 1 gigahertz (GHz) processor, 64 GB hard disk, Windows disassemble (WDASM), Hacker disassemble, Hacker View (Hiew), Code Block (GNU GCC compiler), an open source and cross platform IDE compiler.,



```
#define _C_ char
#define OxcC_ _C_
#define OxK__0x654hu string
#define _010011101 bool
#define oxFEABD  if
#define o0Oo while

typedef unsigned __int64 uint64_t;
using namespace std;OxK__0x654hu compute_hash(ifstream&);
OxK__0x654hu systemID();
OxK__0x654hu encryptDecrypt(OxK__0x654hu toEncrypt);
class HashInfo
__O_ OxcC_ hashval[100];OxcC_ configHash[100];O run_state;OxcC_ appkey[20];
public:Ax0xBEGIN setHashVal(OxK__0x654hu);OxcC_* getHashVal();Ax0xBEGIN setState(O);
O getState();Ax0xBEGIN setConfigHash(OxK__0x654hu);OxcC_* getConfigHash();
Ax0xBEGIN setAppKey(OxK__0x654hu);OxcC_* getAppKey();O_Ox__;

Ax0xBEGIN HashInfo ::  setHashVal(OxK__0x654hu hashv)__O_ strncpy(hashval, hashv.c_str(),
sizeof(hashval));hashval[sizeof(hashval) - 1] = 0;}Ax0xBEGIN HashInfo ::  setState(O state)
__O_ run_state = state;}O HashInfo ::  getState()__O_ __OxCXs run_state;O_Ox__
OxcC_* HashInfo ::  getHashVal()__O_ __OxCXs hashval;O_Ox__
```

main.cpp [ObfusSec] - Code::Blocks 13.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
Management
Projects Symbols Files
Workspace
ObfusSec
Sources
main.cpp
Logs & others
Code::Blocks Search results Cccc Build log Build messages CppCheck CppCheck messages Cscope Debugger DoxyB
:\Users\KaaCeeLo\Desktop\ObfusSec1\main.cpp
Windows (CR+LF) WINDOWS-1252 Line 1, Column 1 Insert Read/Write default

**Fig. 2:** Snapshot of the conversion of our C++ program to an encrypted code

A study of some hash exchange function using Mozilla Firefox Web browser was conducted using our encryption tool. The result is as shown in table 1.

**Table 1:** Hex-Hash Exchange for Mozilla Firefox Web browser

| Hash Exchange List | Found Hashes | Left Hashes | Total | Percentage |
|---|---|---|---|---|
| airsoft | 7,006 | 141 | 7,147 | 98.03% |
| carders | 2, 214 | 6, 211 | 8, 425 | 26.28% |
| .fcolimpia | 3, 726 | 630 | 4,346 | 85.54% |
| freehackforumall | 25, 335 | 16,390 | 41,725 | 60.72% |
| gamepassion | 2,944 | 38 | 2,982 | 98.72% |
| gawkers | 674, 690 | 69,174 | 743, 864 | 90.7% |
| hellbound | 5, 949 | 277 | 6,225 | 95.55% |
| inbookmark | 2,101 | 229 | 2,330 | 90.17% |
| jaillords | 2,348 | 94 | 2,442 | 96.16% |
| phpbb | 186,127 | 3,540 | 189,667 | 98.13% |
| rootkit | 54,372 | 4,303 | 58,675 | 92.67% |
| scrollwars | 11, 570 | 369 | 11,939 | 96.91% |
| tntvillage | 36,154 | 1,681 | 37,835 | 95.56% |
| Twilightdisorder | 5,079 | 98 | 5,177 | 98.11% |
| wsgifts | 2,798 | 70 | 2,868 | 97.56% |
| xiaozhi | 1,068,680 | 728,782 | 1,798,462 | 59.42% |

Table 1 shows a hex-hash exchange for Mozilla Firefox Web browser which was used to test our encryption tool. Using the hash exchange list, we were able to determine the total number of hashes, the hashes found, and the number not found. Based on these figures, were then determine the percentage of the found hash. This is graphically illustrated in figure 3.

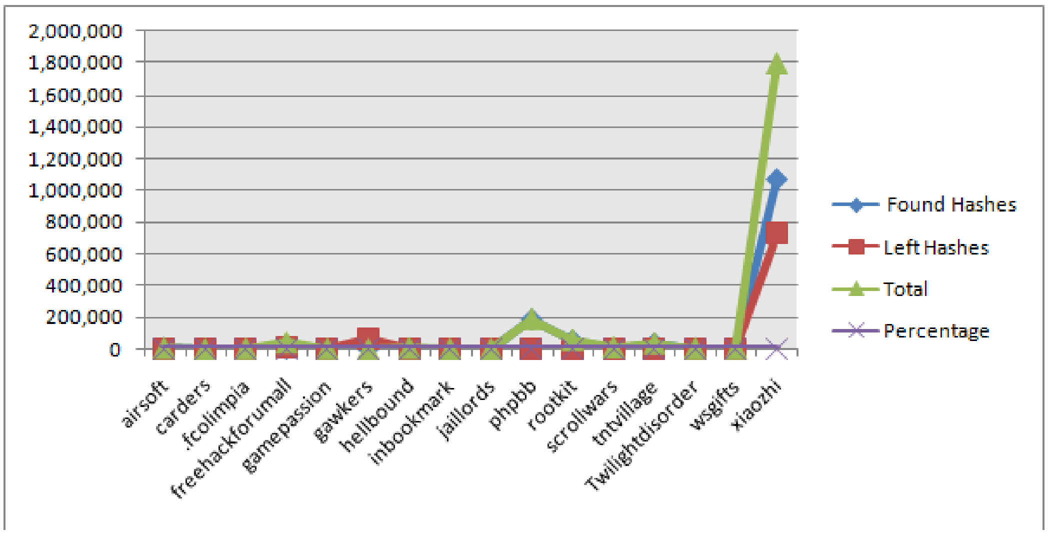


**Fig. 3:** Hex-Hash Exchange for Mozilla Firefox Web browser

## V. CONCLUSION

Software piracy and the breach of the copyright laws, intentionally or unintentionally is very common these days. Software piracy is a menace to software developers and computer users all over the world. Software hackers have become nuisance to many organizations, corporate bodies and government alike. Pirating software has caused lost of several billions US Dollars in revenue and the problem continued unabated. There have been a lot of security threats in recent past due to the activities of hackers. Several financial organizations and national securities have been threatened and even some have been compromised. In this paper, we proposed the code encryption technique for combating software piracy. The technique converts plain code to an encrypted for that cannot be understood by the hacker or intended hacker unless he has the key to encrypt or decode the encrypted data. Our result shows that using this technique, it will be difficult to pirate software after it has been released to intended user(s).

## REFERENCES

[1] Hornik, D. M. (1994). Combating Software Piracy; The Softlifting Problem, Harvard Journal of Law & Technology, Vol. 7, No. 2, pp. 377-417.

[2] Palanissamy, A. (20110. The Future of Copyright In India—A Special Reference to Software Piracy, Its Challenge and Proposal for Reform. In Proceedings of the Int’l Conference on Software and Computer Applications (IPCSIT’11), Singapore, Vol. 9, pp. 102-107.

[3] Kamble, S. (2013). Software piracy Prevention Using Image Splitting, Int’l Journal of Information and Computer technology, Vol. 3, No. 8, pp. 833-840.

[4] Hongxia, J. and Lotspiech, J. (2003). Forensic Analysis for Tamper Resistant Softwar14th International Symposium on Software Reliability Engineering, ISSRE 2003, http://www.bsa.org/Piracy%20Portal.aspx, Accessed 20 March 2014.

[5] Gu, Y., Wyseur, B. and Preneel, B. (2011). Software-Based Protection Is Moving To The Mainstream. IEEE Software, Special Issue on Software Protection, Vol. 28, No. 2, pp. 56–59.

[6] Business Software Alliance and IDC (2010). The Risks of Obtaining and Using Pirated Software. Eighth Annual Business Software Alliance and IDC Global Software Piracy Study.

[7] Conner, K. and Rumelt, R. (1991). Software Piracy: An Analysis of Protection Strategies, Management Science, Vol. 37, pp. 125-139.

[8] Steidlmeier, P. (1993). The Moral Legitimacy of Intellectual Property Claims: American Business and Developing Country Perspectives, journal of Business Ethics, Vol. 12, pp. 157-164.

[9] Kent, S. (1980). Protecting Externally Supplied Software in Small Computers, Unpublished PhD Thesis, . I. T., September, 1980.

[10] Traphagan, M. and Griffith, A. (1998). Software Piracy and Global Competitiveness: Report on Global Software Piracy. International Review of Law, Computers and Technology, Vol. 12, pp. 431-451.

[11] Business Software Alliance (2007). What is Software Piracy? Available from: Business Software Alliance and IDC. (2007). Fourth annual Business Software Alliance and IDC Global Software Piracy Study 2007. Retrieved April 20, 2014, from http://www.BSA.ORG/GLOBALSTUDY.

[12] Kini, R., Rominger, A. and Vijayaraman, B. (2000). An Empirical Study of Software Piracy and Moral Intensity Among University Students. The Journal of Computer Information Systems, Vol. 40, pp. 62-72.

[13] Tang, J. and Farn, C. (2005). The Effect of Interpersonal Influence on Shoplifting Intention and Behavior, Journal of Business Ethics, Vol. 56, pp. 149-161.

[14] Gopal, R. and Sanders, G. (1997), Preventative and Deterrent Controls for Software Piracy. Journal of Management Information Systems, Vol. 13, pp. 29-47.

[15] Al-Rafee, S. and Cronan, T. (2006). Digital Piracy: Factors that Influence Attitude Towards behavior, Journal of Business Ethics, Vol. 63, pp. 237-259.

[16] Birrer, B., Raines, R. Baldwin, R. Mullis, B. and Bennington, R. (2007). Program Fragmentation as a Metammorphic Software protection, 3rd Int’l Symposium on Information Assurance and Security, pp. 369-374.

[17] Zeng, M., Qiong-Mei, L. and Cheng, W. (2010). Practices of Agile Manufacturing Enterprise Data Security and Software Protection. In Proceedings of the 2nd Int'l Conference on Industrial Mechatronics and Automation (ICIMA'10).
[18] Gosler, J. (1985). Software Protection: Myth or Reality? Advances in Cryptology. In Proceedings of CRYPTO'85, Springer-Verlag LNCS 218, pp. 140-157.
[19] Jakobson, M. and Reiter, M. (2002). Discouraging Software Piracy Using Software Aging. In Proceedings of the 1st ACM Workshop on Digital Rights Management (DRM'01), Springer LNCS 2320, pp. 1-12.
[20] Kim, S.-K., (2009). Design of Enhanced Software Protection Architecture by Using Theory of Inventive Problem Solving, IEEE Int'l Conference on Industrial Engineering and Engineering Management, IEEM'09.

**Appendix : The encrypted code**

```
#include <stdlib.h>
#include <stdio.h>
#include <iostream>
#include<iostream>
#include<fstream>
#include <iomanip>
#include <sstream>
#include <windows.h>
#define O int
#define __OxCXs return
#define O_Ox__ }
#define __O_ {
#define _C_ char
#define OxcC_ _C_
#define OxK__0x654hu string
#define _010011101 bool
#define oxFEABD  if
#define o0 cout
typedef unsigned __int64 uint64_t;
using namespace std;OxK__0x654hu Ox11(ifstream&);
OxK__0x654hu _0010();
OxK__0x654hu _O__(OxK__0x654hu toEncrypt);
class HashInfo
__O_ OxcC_ hashval[100];OxcC_ configHash[100];O run_state;OxcC_ appkey[20];
public:Ax0xBEGIN setHashVal(OxK__0x654hu);OxcC_* getHashVal();Ax0xBEGIN setState(O);
O getState();Ax0xBEGIN setConfigHash(OxK__0x654hu);OxcC_* getConfigHash();
Ax0xBEGIN setAppKey(OxK__0x654hu);OxcC_* getAppKey();O_Ox__;
Ax0xBEGIN HashInfo ::  setHashVal(OxK__0x654hu hashv)__O_ strncpy(hashval, hashv.c_str(),
sizeof(hashval));hashval[sizeof(hashval) - 1] = 0;}Ax0xBEGIN HashInfo ::  setState(O state)
__O_ run_state = state;}O HashInfo ::  getState()__O_ __OxCXs run_state;O_Ox__
system("cls");o0 << "\n\n\tTic Tac Toe\n\n";
o0 << "Player 1 (X)  -  Player 2 (O)" << endl << endl;o0 << endl;
_O1(5,' ');_O1(1,'|');_O1(5,' ');_O1(1,'|');_O1(5,' ');
_O1(1,'\n');o0 << "  " << ox11L[0xE0-0xDF] << "  |  " << ox11L[0xDF-0xDD] << "  |  " << ox11L[0x4D-0x4A]
<< endl;_O1(5,'_');
_O1(1,'|');_O1(5,'_');_O1(1,'|');_O1(5,'_');_O1(1,'\n');_O1(5,' ');_O1(1,'|');_O1(5,' ');_O1(1,'|');_O1(5,' ');

_O1(2,' ');o0 << ox11L[0x2B-0x23];_O1(2,' ');_O1(1,'|');_O1(2,' ');
o0 << ox11L[0xAA-0xA1] << endl;_O1(5,' ');_O1(1,'|');_O1(5,' ');
_O1(1,'|');_O1(5,' ');_O1(1,'\n');O_Ox__ _O_ _O1(O __o,_C_ o__)
__O_ for(O _o=0; _o<__o; _o++) __O_ printf ("%c",o__);O_Ox__
O_Ox__ OxK__0x654hu Ox11(ifstream& f)__O_ uint64_t hash, fsize;f.seekg(0, ios::end);fsize = f.tellg();
f.seekg(0, ios::beg);hash = fsize;for (uint64_t tmp = 0, i = 0; i < 65536 / sizeof (tmp)
&& f.read((OxcC_*) &tmp, sizeof (tmp)); i++, hash += tmp);f.seekg(MAX(0, (uint64_t) fsize - 65536),
ios::beg);
for (uint64_t tmp = 0, i = 0; i < 65536 / sizeof (tmp) && f.read((OxcC_*) &tmp, sizeof (tmp)); i++, hash +=
tmp);
stringstream cst;cst << setw(16) << setfill('0') << hex << hash;__OxCXs cst.str()+_0010();
O_Ox__ OxK__0x654hu _0010() __O_ HW_PROFILE_INFO hwProfileInfo;
oxFEABD  (GetCurrentHwProfileA(&hwProfileInfo) != 0) __O_ OxK__0x654hu
_10011 = hwProfileInfo.szHwProfileGuid;
```

```
_10011.erase(0, 1); _10011 = _10011.substr(0, _10011.size() - 1);
__OxCXs _10011; O_Ox__ else __O_ cerr << " Cannot get system unique id";
exit(0); O_Ox__ O_Ox__ OxK__0x654hu _O__(OxK__0x654hu Ox)
__O_ OxcC_ key = 'K';OxK__0x654hu output = Ox;
for (O i = 0; i < Ox.size(); i++)output[i] = Ox[i] ^ key; __OxCXs output;O_Ox__
```