Volume 6, Issue 8, August 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A Framework for Preventing Software Cracking in a Distributed Network

E. E. Ogheneovo*
Department of Computer Science,
University of Port Harcourt, Nigeria

C. K. Oputeh
Department of Computer Science,
University of Port Harcourt, Nigeria

Abstract: Software cracking is a serious problem to software protection right. Software cracking is the modification of software in order to remove copyright protection from the software by manipulating the trial/demo version, serial number, hardware key, date checks, etc. Software cracking is a serious threat to commercial software developers worldwide. While deriving exact numbers is hard, it is commonly assumed that the accumulated economic damage from software copyright infringement ranges in the order of several tens of billions of dollars. In this paper, we develop a framework for preventing software cracking network. In this paper, we examine software protection through code obfuscation and encryption technique, known as one - way hashing, which resists reverse engineering attacks. The model is a code transformation technique in which functionality of original code is maintained while obfuscated and one - way hashed code is made difficult to reverse engineer. The key idea is to hide the code. The code was developed using assembly language which makes it difficult to understand thus providing effective technique for preventing cracking through encryption and obfuscation. The result shows that the code is difficult for humans to read and thereby providing effective and watertight mechanism against cracking.

Keywords: Software, software cracking, distributed network, software copyright, reverse engineering

I. INTRODUCTION

Software cracking [1] is a serious problem to software protection right. Usually, software cracking is the modification of software in order to remove copyright protection from the software by manipulating the trial/demo version, serial number, hardware key, date checks, etc. Software cracking is a serious threat to commercial software developers worldwide [2]. While deriving exact numbers is hard, it is commonly assumed that the accumulated economic damage from software copyright infringement ranges in the order of several tens of billions of dollars [3] [4]. As a consequence, software manufacturers regularly integrate technical protection mechanisms in their products that aim at preventing unauthorized copying and redistribution. Examples of such protection mechanisms include serial number or license key checks that are performed when installing the protected software products. Some manufacturers even go as far as requiring special hardware called “dongles” to be present in the user’s system to successfully run the software. While these and similar countermeasures have been established in the industry since more than three decades, a recent study covering 116 countries claims that about 42% of the globally installed software products are “pirated”.

Even considered conservatively, this indicates the relative ineffectiveness of these measures and concurrently supports the claim that software piracy is a very common phenomenon. On a technical level, hackers and criminals typically try to circumvent anti-piracy measures by creating and distributing specific software patches or even custom reverse-engineered serial number generators for commercial products, called cracks and keygens. Users who look for “cracked” software products can typically access either the unmodified original software through established file sharing systems like Bit torrent, One-Click-Hosters (OCH), or via Usenet downloads, or download bundles of the original software and cracks or keygens. Additionally, some types of cracks target legitimate evaluation copies of software products that are published by the actual manufacturer to attract potential customers. In these cases, the attacker’s aim is to emulate or unlock restricted features, thus upgrading the evaluation version to the full-priced software product [5].

Software cracking has become widespread on the Internet, where one can find a crack for almost any commercial software program available in the market. Thousands of websites dedicated to providing programs which will crack commercial software have appeared on the Internet in the last ten years, making it easier than ever before to pirate software [6]. The most common software crack is the modification of an application’s binary to cause or prevent a specific key branch in the program’s execution. This is accomplished by reverse engineering the compiled program code using a debugger such as SoftICE, OllyDbg, GDB, or MacsBug [7] until the software crack reaches the subroutine that contains the primary method of protecting the software (or by disassembling an executable file with a program such as IDA). The binary is then modified using the debugger or a hex editor in a manner that replaces a prior branching OPCODE (operation code) with its complement or a NOP OPCODE so that key branch will either always execute a specific subroutine or skip over it. Almost all common software cracks are a variation of this type.

Proprietary software developers are constantly developing techniques such as code obfuscation, encryption, and fragmentation, which are offshoots of self – modifying code, to make this modification increasingly difficult. Software cracks often contain viruses. This is done through code injection. A virus writer will find a popular crack on the

Internet, inject his virus code inside this crack, and then re-release it on the Internet for others to download [8]. Computer software comes with a licensed agreement that states how the technology can be used legally (Software and Information Industry Association [SIIA]. This is often referred to as the terms and conditions of use. To help ensure that people adhere to these conditions, copyright laws are designed to protect this form of intellectual property [9]. One of the ways this is achieved is by giving the owner of the software the exclusive rights to it. When these laws are breached people can face civil and criminal charges.

II. RELATED WORK

Cappaert et al. [10] presented a partial encryption approach depending on a code encryption approach. In order to utilize the partial encryption approach, binary codes are partitioned into small segments and encrypted. The encrypted binary codes are decrypted at runtime by users. Thus, the partial encryption overcomes the faults of illuminating all of the binary code at once as only the essential segments of the code are decrypted at runtime. Jung et al. [11] presented a code block encryption approach to protect software using a key chain. Jung's approach uses a unit block, that is, a fixed-size block, rather than a basic block, which is a variable-size block. Basic blocks refer to the segments of codes that are partitioned by control transformation operations, such as "jump" and "branch" commands, in assembly code [12]. Jung's approach is very similar to Cappaert's scheme. Jung's approach tries to solve the issue of Cappaert's approach. If a block is invoked by more than two preceding blocks, the invoked block is duplicated.

Unauthorized reverse-engineering of algorithms is a major issue for the software industry. Reverse engineers look for security holes in the program to make use of competitors' vital approaches. In order to discourage reverse-engineering, developers use a wide range of static software protections to obfuscate their programs. 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. The research by Birrer et al, [14] 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.

Kent [15] 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. The technique considered the supple 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.

Gosler [16] 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". Herzberg et al. [17] focused on the issue of software copy protection and presented a solution needing CPU encryption support (which was far less possible when presented almost 20 years ago, circa 1984-85). Cohen [18] on software diversity and obfuscation is directly concentrated to software protection and offers a wide range of algorithms.

The subsequent practical tamper resistance system of Aucsmith [19] handled similar problems by an integration of just-in-time instruction decryption, and rearranging instruction blocks at run-time to vigorously change the deals with the executing statements during program execution. Several researchers have proposed techniques on software obfuscation using automated tools and code transformations (Collberg et al. [20]. One idea would be to employ language-based tools to transform a program (most easily from source code) to a functionally equivalent program which presents greater reverse engineering barriers. If implemented in the form of a pre-compiler, the usual portability issues can be handled by the back-end of standard compilers. Collberg et al, [21] provides more information regarding software obfuscation which includes descriptions about Categorizing code transformations (e.g. control flow obfuscation, data obfuscation, layout obfuscation, preventive transformations), identification of control flow changes using opaque predicates (expressions not easy for an attacker to predict, but whose worth is recognized at compilation or obfuscation time), preliminary suggestions on metrics for code transformations, program slicing tools, and the usage of (de)aggregation of flow control or data.

Essential suggestions in software protection are done by Aucsmith [22] in combination with Graunke [23] at Intel. Aucsmith provides tamper prevention software which prevents inspection and changing of any sort on the software code. Architecture is suggested according to an Integrity Verification Kernel (IVK) that checks the reliability of vital code segments. The IVK architecture is self-decrypting and includes self-adjustment code. Software tampering prevention using self-checking code was described by Horne [24]. The integrity of segments of code is tested using some code known as testers. This can be a linear hash function and a predictable hash value. If the integrity condition is not met, suitable actions will be carried out so as to make the integrity condition satisfied. The attackers can be confused and it is difficult for them to hack the testers if more number of testers is used.

Chang and Atallah [25] presented a technique with fairly extensive capacity containing a set of guards that can be programmed to perform arbitrary processes. An illustration for this is the check sum code segments for integrity checking which provides resistance against software tamper. An additional describe guard function is repairing code (e.g. if a spoiled code segment is identified, downloading and installing a new version of the code section). The author also presents a technique for automatically keeping protections within code. Chen et al. [26] put forth oblivious hashing that engages compile-time code alterations with outcomes in the calculation of a running trace of the execution history of a complete code. In this approach a trace is considered as increasing hash values of a subset of expressions that happens inside the usual program execution.

III. METHODOLOGY

There are many reasons to protect a program’s implementation details. Commercial entities generally do so in an attempt to preserve the secrecy of algorithms and protocols that are considered to provide a competitive advantage. While crackers and cyber criminals share this goal, they also seek to confound defense systems, postpone their detection and eventual tracing.

3.1 Materials and Method

We used Windows 8 operating system, 1 GHz processor, 1 GB RAM, 128 GB hard disk, Windows Disassembler (WDASM), Hacker Disassembler, Hacker View (Hiew) text editor, Code Block (GNU GCC Compiler). The software used are: Code::Blocks 13.12, MinGW Installer, Hacker Disassembler (HDasm), and Hex Editor.

- **Code::Blocks** is an open source , cross platform IDE software which supports multiple compilers including GC and Visual C++. It is oriented towards C, C++, and Fortran.
- **MinGW Installer** is also a free and open source development environment for native Microsoft Windows applications. It includes a port of the GNU Compiler Collection (GCC), GNU Binutils for Windows. It has header files and static import libraries which enables the use of the Windows API.
- **Hacker Disassembler (Hdasm)** is a disassemble for computer software which generates assembly language source code from machine executable code. It supports a variety of executable formats for different processors and operating systems. Hdasm performs automatic code analysis, using cross-references between code sections, knowledge of parameters of API calls, and other information.
- **Hex Editor** (binary editor) is a computer program that allows for manipulation of the fundamental binary data that constitutes a computer file. The name ‘hex’ comes from ‘hexadeximal’: the standard numerical format for editing binary data.

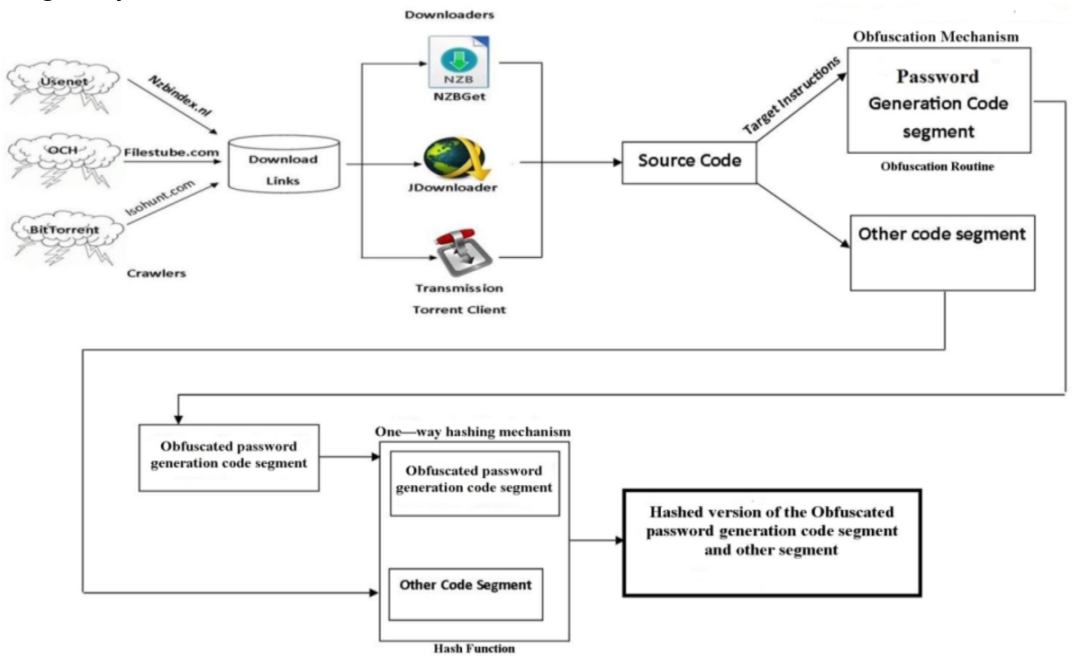

Downloaders
Usenet
OCH
BitTorrent
Nzbindex.nl
Filestube.com
isohunt.com
Crawlers
Download Links
NZB
NZBGet
JDownloader
Transmission Torrent Client
Source Code
Target Instructions
Obfuscation Mechanism
Password Generation Code segment
Obfuscation Routine
Other code segment
Obfuscated password generation code segment
One—way hashing mechanism
Obfuscated password generation code segment
Other Code Segment
Hash Function
Hashed version of the Obfuscated password generation code segment and other segment

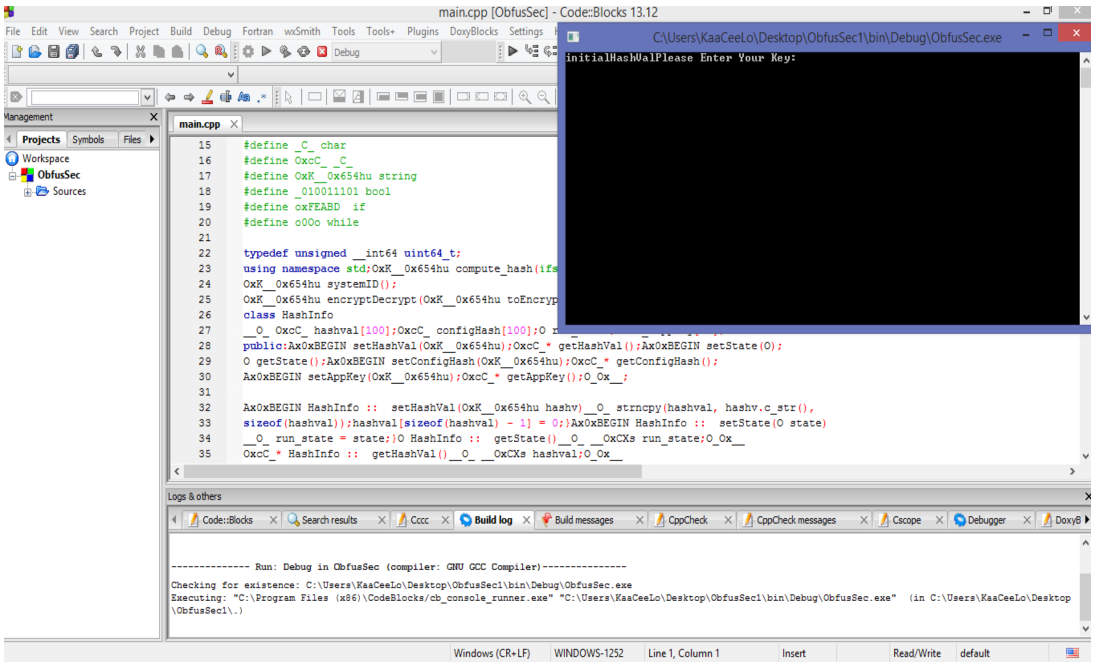
Fig. 1: Architecture of Mechanism

Figure 1 shows the architecture for preventing software cracking. Based on the figure, source code is generated from various links and converted to target instructions through obfuscation mechanism by generating password for the code segment and various other code segments are generated using one-way hashing mechanism (using hash function) to produce hashed.

IV. RESULTS AND DISCUSSION

The main idea of this technique is to protect software against illegal acts of hacking. We examine software protection through code obfuscation and encryption technique, known as one – way hashing, which resists reverse engineering attacks. Our model is a code transformation technique in which functionality of original code is maintained while obfuscated and one – way hashed code is made difficult to reverse engineer. The key idea is to hide the code. In our technique the application is transformed so that it's functionally identical to the original but it is much more difficult to understand. This is done by adding a mechanism of self-modification, known as obfuscation mechanism, to the original program, so that the original program becomes hard to be analyzed. In the binary program obtained by the proposed method, the original code fragments we want to protect are obfuscated so that the hackers would not be able to understand the real source code.

Figure 2 shows an obfuscated code of the assembly language used for the implementation of the research. It contains the


main.cpp [ObfusSec] - Code::Blocks 13.12
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings
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
Windows (CR+LF) WINDOWS-1252 Line 1, Column 1 Insert Read/Write default
C:\Users\KaaCeeLo\Desktop\ObfusSec1\bin\Debug\ObfusSec.exe

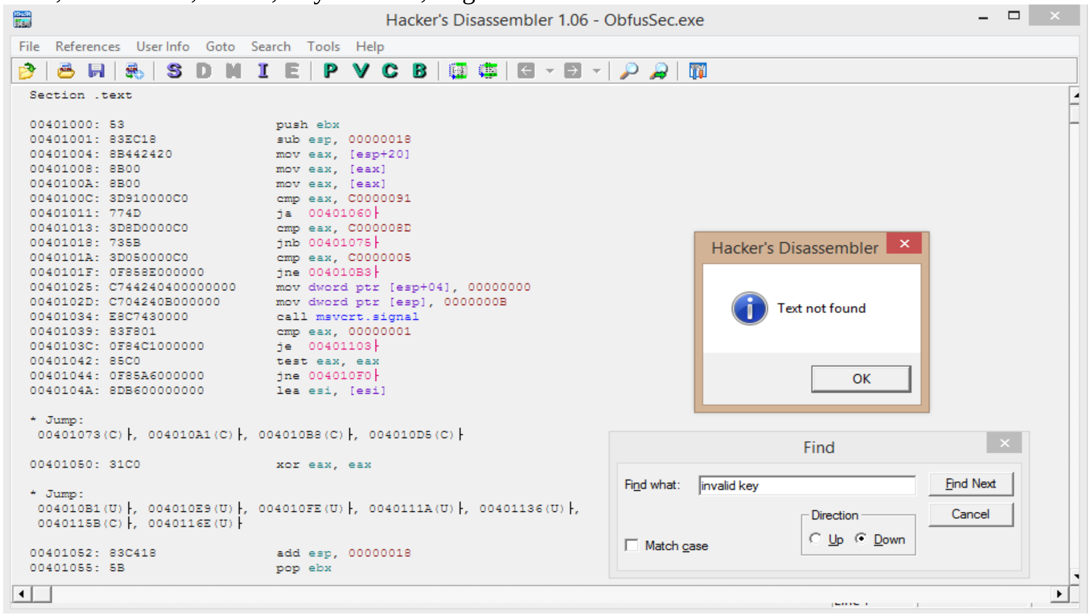
```
#define _C_ char
#define OxcC_ _C_
#define OxK__0x654hu string
#define _010011101 bool
#define oxFEABD  if
#define o0Oo while

typedef unsigned __int64 uint64_t;
using namespace std;OxK__0x654hu compute_hash(ifs
OxK__0x654hu systemID();
OxK__0x654hu encryptDecrypt(OxK__0x654hu toEncryp
class HashInfo
__O_ OxcC_ hashval[100];OxcC_ configHash[100];O r
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__
```

```
initialHashValPlease Enter Your Key:
```

```
-------------- Run: Debug in ObfusSec (compiler: GNU GCC Compiler)---------------
Checking for existence: C:\Users\KaaCeeLo\Desktop\ObfusSec1\bin\Debug\ObfusSec.exe
Executing: "C:\Program Files (x86)\CodeBlocks/cb_console_runner.exe" "C:\Users\KaaCeeLo\Desktop\ObfusSec1\bin\Debug\ObfusSec.exe"  (in C:\Users\KaaCeeLo\Desktop
\ObfusSec1\.)
```

Fig. 2: Screen display to enter key

Finally, figure 3 shows the screen display of the output after running the source code. It contains the VirtualAddress, VirtualSize, Offset, PhysicalSize, flags and the code itself.



```
Section .text

00401000: 53                           push ebx
00401001: 83EC18                       sub esp, 00000018
00401004: 8B442420                     mov eax, [esp+20]
00401008: 8B00                         mov eax, [eax]
0040100A: 8B00                         mov eax, [eax]
0040100C: 3D910000C0                   cmp eax, C0000091
00401011: 774D                         ja  00401060├
00401013: 3D8D0000C0                   cmp eax, C000008D
00401018: 735B                         jnb 00401075├
0040101A: 3D050000C0                   cmp eax, C0000005
0040101F: 0F858E000000                 jne 004010B3├
00401025: C744240400000000             mov dword ptr [esp+04], 00000000
0040102D: C704240B000000               mov dword ptr [esp], 0000000B
00401034: E8C7430000                   call msvcrt.signal
00401039: 83F801                       cmp eax, 00000001
0040103C: 0F84C1000000                 je  00401103├
00401042: 85C0                         test eax, eax
00401044: 0F85A6000000                 jne 004010F0├
0040104A: 8DB600000000                 lea esi, [esi]

* Jump:
 00401073(C)├, 004010A1(C)├, 004010B8(C)├, 004010D5(C)├

00401050: 31C0                         xor eax, eax

* Jump:
 004010B1(U)├, 004010E9(U)├, 004010FE(U)├, 0040111A(U)├, 00401136(U)├,
 0040115B(C)├, 0040116E(U)├

00401052: 83C418                       add esp, 00000018
00401055: 5B                           pop ebx
```

Find what: invalid key

Fig. 3: Screen display of the output

V. CONCLUSIONS

In this paper, we examine software protection through code obfuscation and encryption technique, known as one - way hashing, which resists reverse engineering attacks. The model is a code transformation technique in which functionality of original code is maintained while obfuscated and one - way hashed code is made difficult to reverse engineer. The key idea is to hide the code. The code was developed using assembly language which makes it difficult to understand thus providing effective technique for preventing cracking through encryption and obfuscation. The result shows that the code is difficult for humans to read and thereby providing effective and watertight mechanism against cracking. There are many reasons to protect a program's implementation details. Commercial entities generally do so in an attempt to preserve the secrecy of algorithms and protocols that are considered to provide a competitive advantage. While crackers and cyber criminals share this goal, they also seek to confound defense systems, postpone their detection and eventual tracing. We examine software protection through code obfuscation and encryption technique, known as one - way hashing, which resists reverse engineering attacks. The model is a code transformation technique in which functionality of original code is maintained while obfuscated and one - way hashed code is made difficult to reverse engineer. The key idea is to hide the code. In this technique, the application is transformed so that it's functionally identical to the original but it is much more difficult to understand. This is done by adding a mechanism of self-modification, known as obfuscation mechanism, to the original program, so that the original program becomes hard to be analyzed. In the binary program obtained by the proposed method, the original code fragments we want to protect are obfuscated so that the hackers would not be able to understand the real source code. Then, we use an encryption technique, known as one - way hashing, to generate our application licenses. The technique is able to protect software against illegal acts of hacking.

REFERENCES

[1] C. K. Oputeh and E. E. Ogheneovo, "Overcoming Trial Version Software Cracking Using a Hybridized Self-Modifying Mechanism," IOSR Journal of Computer Engineering (IOSR-JCE), Vol. 16, Issue 3, pp. 13-21, 2014.

[2] E. E. Ogheneovo. and C. K. Oputeh, "Source Code Obfuscation: A Technique for Checkmating Software Reverse Engineering," *Int'l Journal of Engineering and Science Invention*, vol. 3, Issue 5, pp. 1-10, May 2014

[3] Business Software Alliance, "What is Software Piracy?" *Fourth Annual Business Software Alliance and IDC Global Software Piracy Study* 2007. Retrieved March 2, 2008, http://www.BSA.ORG/GLOBALSTUDY, Accessed 20th April, 2013.

[4] R. Gopal and G. Sanders, "International Software Piracy: Analysis of Key Issues and Impacts, Info. Sys. Research, Vol. 9, No. 4, pp. 380 - 397l, 1998.

[5] P. Oorschot ," P.C.: Revisiting Software Protection," *In ISC 2003*. LNCS, Springer, pp. 1 – 13, 2013.

[6] M. Kammerstetter, C. Platzer and G. Wondracek, "Vanity, Cracks and Malware: Insights into the Anti-Copy Protection Ecosystem**,"** *Proc. of the 2012 ACM Conference on Computer and Communications Security*, pp. 809-820, 2012.

[7] P. Gutmann, "An Open-source Cryptographic Coprocessor," *Proc. 2000 USENIX Security Symposium*, 2000.

[9] M. Traphagan and A. Griffith, "Software Piracy and Global Competitiveness: Report on Global Software Piracy. *Int'l Review of Law," Computers and Technology*, vol. 12, pp. 431-451, 1998.

[10] J. Cappaert, N. Kisserli, D. Schellekens and B. Preneel, Toward Tamper Resistant Code Encryption, *Practice and Experience*, LNCS, Vol. 4991, pp. 86-100.

[11] D. Jung, H. Kim and J. Park, "A Code Block Cipher Method to Protect Application Programs From Reverse Engineering," Korea Inst. Inf. Security Cryptology, vol. 18, no. 2, pp. 85-96, 2008.

[12] J. Cappaert, N. Kisserli, D. Schellekens and B. Preneel, "Self-Encrypting Code to Protect Against Analysis and Tampering," 1st Benelux Workshop Inf. Syst. Security, 2006.

[13] B. Birrer, R., Raines, R Baldwin, B. Mullinsand R. Bennington, "Program Fragmentation as a Metamorphic Software Protection," 3rd *International Symposium on Information Assurance and Security*, pp. 369 – 374, 2007.

[14] S. Kent , "Protecting Externally Supplied Software in Small Computers," Ph.D. Thesis, M.I.T., September 1980.

[15] J. Gosler, "Software Protection: Myth or Reality? *Advances in Cryptology," CRYPTO'85*, Springer-Verlag LNCS 218, pp.140–157, 1985.

[16] A. Herzberg and S. Pinter, Public Protection of Software, *ACM Trans. Computer Systems (Earlier Version in Crypto'85),*Vol.5, No.4, pp.371–391

[17] F. Cohen, "Operating System Protection through Program Evolution, *Computers and Security,"* Vol. 12, No 6, pp. 565–584.

[18] D. Aucsmith, (1996). Tamper-Resistance Software: An Implementation. In Ross Anderson, Editor, *Information Hiding, Proc. of the 1st International Workshop*, Vol. 1174 of LNCS, pp. 317–333, 1996.

[19] C. Collberg, C. Thomborson and D. Low (1998). Breaking Abstractions and Unstructuring Data Structures, *IEEE International Conf. Computer Languages* (ICCL'98), Chicago, IL (USA), 1998.

[20] C. Collberg, C. Thomborson and D. Low (1997). A Taxonomy of Obfuscating Transformations, Technical Report 148, Dept. Computer Science, University of Auckland.

[21] B. Horne, L. Matheson, C. Sheehan and R. Tarjan, “Dynamic Self-Checking Techniques for Improved Tamper Resistance,” Proc. 1st ACM Workshop on Digital Rights Management (DRM 2001), Springer LNCS 2320, pp.141–159, 2002.

[22] H. Chang and M. Atallah,(2002). Protecting Software Code by Guards. In Proc. 1st ACM Workshop on Digital Rights Management (DRM 2001), Springer LNCS 2320, pp.160–175.

[23] Y. Chen, R. Venkatesan, M’ Cary, R. Pang, S. Sinha, and M. Jakubowski (2002). Oblivious Hashing: A Stealthy Software Integrity Verification Primitive. Proc. 5th Information Hiding Workshop (IHW), Netherland, Springer LNCS 2578, pp. 440 – 414.

APPENDIX: Part of the assembly language code used for the study

```
* Call:
 00401290, 004012B0

00401000: 53                    push ebx
00401001: 83EC38                sub esp, 00000038
00401004: A1C0A24700            mov eax, [0047A2C0]
00401009: 85C0                  test eax, eax
0040100B: 741C                  je  00401029
0040100D: C744240800000000      mov dword ptr [esp+08],
00000000
00401015: C744240402000000      mov dword ptr [esp+04], 00000002
0040101D: C7042400000000        mov dword ptr [esp], 00000000
00401024: FFD0                  call eax
00401026: 83EC0C                sub esp, 0000000C

* Jump:
 0040100B(C)

00401029: C7042410114000        mov dword ptr [esp], 00401110
00401030: E817140200            call
KERNEL32.SetUnhandledExceptionFilter
00401035: 83EC04                sub esp, 00000004
00401038: E823750100            call 00418560
0040103D: E8FE750100            call 00418640
00401042: 8D44242C              lea eax, [esp+2C]
00401046: 89442410              mov [esp+10], eax
0040104A: A1E0604700            mov eax, [004760E0]
0040104F: C744240400104800      mov dword ptr [esp+04],
00481000
00401057: C7042404104800        mov dword ptr [esp], 00481004
0040105E: C744242C00000000      mov dword ptr [esp+2C],
00000000
00401066: 8944240C              mov [esp+0C], eax
0040106A: 8D442428              lea eax, [esp+28]
0040106E: 89442408              mov [esp+08], eax
00401072: E8D9110200            call msvcrt.__getmainargs
00401077: A134704800            mov eax, [00487034]
0040107C: 85C0                  test eax, eax
0040107E: 7442                  je  004010C2
00401080: 8B1DD8824800          mov ebx, msvcrt._iob
00401086: A3E4604700            mov [004760E4], eax
0040108B: 89442404              mov [esp+04], eax
0040108F: 8B4310                mov eax, [ebx+10]
00401092: 890424                mov [esp], eax
00401095: E8BE110200            call msvcrt._setmode
0040109A: A134704800            mov eax, [00487034]
0040109F: 89442404              mov [esp+04], eax
004010A3: 8B4330                mov eax, [ebx+30]
004010A6: 890424                mov [esp], eax
004010A9: E8AA110200            call msvcrt._setmode
004010AE: A134704800            mov eax, [00487034]
004010B3: 89442404              mov [esp+04], eax
004010B7: 8B4350                mov eax, [ebx+50]
004010BA: 890424                mov [esp], eax
004010BD: E896110200            call msvcrt._setmode

* Jump:
 0040107E(C)

004010C2: E899110200            call msvcrt.__p__fmode
004010C7: 8B15E4604700          mov edx, [004760E4]
004010CD: 8910                  mov [eax], edx
004010CF: E8EC760100            call 004187C0
004010D4: 83E4F0                and esp, FFFFFFF0
004010D7: E844790100            call 00418A20
004010DC: E887110200            call msvcrt.__p__environ
004010E1: 8B00                  mov eax, [eax]
004010E3: 89442408              mov [esp+08], eax
004010E7: A100104800            mov eax, [00481000]
004010EC: 89442404              mov [esp+04], eax
004010F0: A104104800            mov eax, [00481004]
004010F5: 890424                mov [esp], eax
004010F8: E859030000            call 00401456
004010FD: 89C3                  mov ebx, eax
004010FF: E86C110200            call msvcrt._cexit
00401104: 891C24                mov [esp], ebx
00401107: E848130200            call KERNEL32.ExitProcess
```

```
0040110C: 8D742600              lea esi, [esi]
00401110: 53                    push ebx
00401111: 83EC28                sub esp, 00000028
00401114: 8B442430              mov eax, [esp+30]
00401118: 8B00                  mov eax, [eax]
0040111A: 8B00                  mov eax, [eax]
0040111C: 3D910000C0            cmp eax, C0000091
00401121: 773D                  ja  00401160
00401123: 3D8D0000C0            cmp eax, C000008D
00401128: 724D                  jb  00401177

* Jump:
 0040117C(C)

004011C0: C744240400000000      mov dword ptr [esp+04],
00000000
004011C8: C704240B000000        mov dword ptr [esp], 0000000B
004011CF: E8A4100200            call msvcrt.signal
004011D4: 83F801                cmp eax, 00000001
004011D7: 744A                  je  00401223
004011D9: 85C0                  test eax, eax
004011DB: 0F8473FFFFFF          je  00401154
004011E1: C704240B000000        mov dword ptr [esp], 0000000B
004011E8: FFD0                  call eax
004011EA: B8FFFFFFFF            mov eax, FFFFFFFF
004011EF: E962FFFFFF            jmp 00401156

* Jump:
 004011D7(C)

00401223: C744240401000000      mov dword ptr [esp+04], 00000001
0040122B: C704240B000000        mov dword ptr [esp], 0000000B
00401232: E841100200            call msvcrt.signal
00401237: 83C8FF                or eax, FFFFFFFF
0040123A: E917FFFFFF            jmp 00401156
0040123F: 90                    nop

* Jump:
 00401146(C)

00401240: C744240401000000      mov dword ptr [esp+04], 00000001
00401248: C7042408000000        mov dword ptr [esp], 00000008
0040124F: E824100200            call msvcrt.signal
00401254: 85DB                  test ebx, ebx
00401256: B8FFFFFFFF            mov eax, FFFFFFFF
0040125B: 0F84F5FEFFFF          je  00401156
00401261: 8944241C              mov [esp+1C], eax
00401265: E8D6730100            call 00418640
0040126A: 8B44241C              mov eax, [esp+1C]
0040126E: E9E3FEFFFF            jmp 00401156
00401273: 8DB600000000          lea esi, [esi]
00401279: 8DBC2700000000        lea edi, [edi]


00401334: 55                    push ebp
00401335: 89E5                  mov ebp, esp
00401337: 83EC28                sub esp, 00000028
0040133A: 894DF4                mov [ebp-0C], ecx
0040133D: 8B4D08                mov ecx, [ebp+08]
00401340: E87F390200            call 00424CC4
00401345: 89C2                  mov edx, eax
00401347: 8B45F4                mov eax, [ebp-0C]
0040134A: C744240864000000      mov dword ptr [esp+08],
00000064 ;'d'
00401352: 89542404              mov [esp+04], edx
00401356: 890424                mov [esp], eax
00401359: E82A0F0200            call msvcrt.strncpy
0040135E: 8B45F4                mov eax, [ebp-0C]
00401361: C6406300              mov byte ptr [eax+63], 00
00401365: C9                    leave
00401366: C20400                ret 0004

00401369: 90                    nop

* Call:
 00401A52
```

```
0040136A: 55                 push ebp
0040136B: 89E5               mov ebp, esp
0040136D: 83EC04             sub esp, 00000004
00401370: 894DFC             mov [ebp-04], ecx
00401373: 8B45FC             mov eax, [ebp-04]
00401376: 8B5508             mov edx, [ebp+08]
00401379: 8990C8000000       mov [eax+000000C8], edx
0040137F: C9                 leave
00401380: C20400             ret 0004

00401383: 90                 nop

* Call:
 004010F8

00401456: 8D4C2404           lea ecx, [esp+04]
0040145A: 83E4F0             and esp, FFFFFFF0
0040145D: FF71FC             push dword ptr [ecx-04]
00401460: 55                 push ebp
00401461: 89E5               mov ebp, esp
00401463: 57                 push edi
00401464: 56                 push esi
00401465: 53                 push ebx
00401466: 51                 push ecx
00401467: 81EC38050000       sub esp, 00000538
0040146D: 898DC8FAFFFF       mov [ebp-00000538], ecx
00401473: C7857CFBFFFFFC3B4000 mov dword ptr [ebp-
00000484], 00403BFC
0040147D: C78580FBFFFF543D4700 mov dword ptr [ebp-
00000480], 00473D54
00401487: 8D8584FBFFFF       lea eax, [ebp-0000047C]
0040148D: 8D55E8             lea edx, [ebp-18]
00401490: 8910               mov [eax], edx
00401492: BA1C294000         mov edx, 0040291C
00401497: 895004             mov [eax+04], edx
0040149A: 896008             mov [eax+08], esp
0040149D: 8D8564FBFFFF       lea eax, [ebp-0000049C]
004014A3: 890424             mov [esp], eax
004014A6: E80D7B0100         call 00418FB8
004014AB: E870750100         call 00418A20
004014B0: C645E700           mov byte ptr [ebp-19], 00
004014B4: 8D8570FDFFFF       lea eax, [ebp-00000290]
004014BA: C744240408000000   mov dword ptr [esp+04],
00000008

* Jump:
 00401F40(C)

004014F7: 8D8570FDFFFF       lea eax, [ebp-00000290]
004014FD: C7442404E0000000   mov dword ptr [esp+04],
000000E0 ;'à'
00401505: 8D9598FBFFFF       lea edx, [ebp-00000468]
0040150B: 891424             mov [esp], edx
0040150E: C78568FBFFFF02000000 mov dword ptr [ebp-
00000498], 00000002
00401518: 89C1               mov ecx, eax
0040151A: E829710400         call 00448648
0040151F: 83EC08             sub esp, 00000008
00401522: 8D8570FDFFFF       lea eax, [ebp-00000290]
00401528: 83C074             add eax, 00000074 ;'t'
0040152B: 89C1               mov ecx, eax
0040152D: E8A2B40300         call 0043C9D4
00401532: 83F001             xor eax, 00000001
00401535: 84C0               test al, al
00401537: 0F84E4090000       je  00401F21
0040153D: 8D4594             lea eax, [ebp-6C]
00401540: C78568FBFFFF03000000 mov dword ptr [ebp-
00000498], 00000003
0040154A: 89C1               mov ecx, eax
0040154C: E89BD50400         call 0044EAEC
00401551: 8D8598FBFFFF       lea eax, [ebp-00000468]
00401557: 89C1               mov ecx, eax
00401559: E826FEFFFF         call 00401384
0040155E: 85C0               test eax, eax
00401560: 0F94C0             sete  al
00401563: 84C0               test al, al
00401565: 0F847D060000       je  00401BE8
0040156B: 8D4590             lea eax, [ebp-70]
0040156E: C78568FBFFFF04000000 mov dword ptr [ebp-
00000498], 00000004
00401578: 89C1               mov ecx, eax
0040157A: E86DD50400         call 0044EAEC
0040157F: C744240408000000   mov dword ptr [esp+04],
00000008
00401587: C7042410000000     mov dword ptr [esp], 00000010
0040158E: E829100700         call 004725BC
00401593: 8D9578FCFFFF       lea edx, [ebp-00000388]
00401599: 890424             mov [esp], eax
0040159C: C78568FBFFFF05000000 mov dword ptr [ebp-
00000498], 00000005
004015A6: 89D1               mov ecx, edx
004015A8: E8E3740600         call 00468A90
004015AD: 83EC04             sub esp, 00000004
004015B0: 8D8598FBFFFF       lea eax, [ebp-00000468]
004015B6: 89C1               mov ecx, eax
004015B8: E8DBFDFFFF         call 00401398
004015BD: 89442404           mov [esp+04], eax
004015C1: C7042460184800     mov dword ptr [esp], 00481860
004015C8: C78568FBFFFF06000000 mov dword ptr [ebp-
00000498], 00000006
004015D2: E839080700         call 00471E10
004015D7: 8D459D             lea eax, [ebp-63]
004015DA: 89C1               mov ecx, eax
004015DC: E81B240400         call 004439FC
004015E1: 8D8598FBFFFF       lea eax, [ebp-00000468]
004015E7: 89C1               mov ecx, eax
004015E9: E8AAFDFFFF         call 00401398
004015EE: 8D558C             lea edx, [ebp-74]
004015F1: 8D4D9D             lea ecx, [ebp-63]
004015F4: 894C2404           mov [esp+04], ecx
004015F8: 890424             mov [esp], eax
004015FB: C78568FBFFFF07000000 mov dword ptr [ebp-
00000498], 00000007
00401605: 89D1               mov ecx, edx
00401607: E8A0D20400         call 0044E8AC
0040160C: 83EC08             sub esp, 00000008
0040160F: 8D459D             lea eax, [ebp-63]
00401612: 89C1               mov ecx, eax
00401614: E8EF230400         call 00443A08
00401A5A: C7442404B8704700   mov dword ptr [esp+04],
004770B8
00401A62: C7042460184800     mov dword ptr [esp], 00481860
00401A69: E8A2030700         call 00471E10
00401A6E: C645E701           mov byte ptr [ebp-19], 01
00401A72: 8D4580             lea eax, [ebp-80]
00401A75: C78568FBFFFF16000000 mov dword ptr [ebp-
00000498], 00000016
00401A7F: 89C1               mov ecx, eax
00401A81: E882D40400         call 0044EF08
00401A86: 8D856CFEFFFF       lea eax, [ebp-00000194]
00401A8C: C78568FBFFFF0C000000 mov dword ptr [ebp-
00000498], 0000000C
00401A96: 89C1               mov ecx, eax
00401A98: E8BBC70500         call 0045E258
00401A9D: 8D4584             lea eax, [ebp-7C]
00401AA0: C78568FBFFFF0A000000 mov dword ptr [ebp-
00000498], 0000000A
00401AAA: 89C1               mov ecx, eax
00401AAC: E857D40400         call 0044EF08
00401AB1: 8D4588             lea eax, [ebp-78]
00401AB4: C78568FBFFFF09000000 mov dword ptr [ebp-
00000498], 00000009
00401ABE: 89C1               mov ecx, eax
00401AC0: E843D40400         call 0044EF08
00401AC5: 8D458C             lea eax, [ebp-74]
00401AC8: C78568FBFFFF06000000 mov dword ptr [ebp-
00000498], 00000006
00401AD2: 89C1               mov ecx, eax
00401AD4: E82FD40400         call 0044EF08
00401AD9: E9DA000000         jmp 00401BB8

* Jump:
 00401642(C)

00401ADE: 8D45BF             lea eax, [ebp-41]
00401AE1: 89C1               mov ecx, eax
00401AE3: E8141F0400         call 004439FC
```

```
00401AE8: 8D8574FFFFFF        lea eax, [ebp-0000008C]
00401AEE: 8D55BF              lea edx, [ebp-41]
00401AF1: 89542404            mov [esp+04], edx
00401AF5: C70424EC704700      mov dword ptr [esp], 004770EC
00401AFC: C78568FBFFFF1A000000 mov dword ptr [ebp-
00000498], 0000001A
00401B06: 89C1                mov ecx, eax
00401B08: E89FCD0400          call 0044E8AC
00401B0D: 83EC08              sub esp, 00000008
00401B10: 8D45BF              lea eax, [ebp-41]
00401B13: 89C1                mov ecx, eax
00401B15: E8EE1E0400          call 00443A08
00401B1A: 8D45C4              lea eax, [ebp-3C]
00401B1D: 8D9574FFFFFF        lea edx, [ebp-0000008C]
00401B23: 891424              mov [esp], edx
00401B26: C78568FBFFFF1B000000 mov dword ptr [ebp-
00000498], 0000001B
00401B30: 89C1                mov ecx, eax
00401B32: E821CE0400          call 0044E958
00401B37: 83EC04              sub esp, 00000004
00401B3A: 8D45C0              lea eax, [ebp-40]
00401B3D: 8D55C4              lea edx, [ebp-3C]
00401B40: 89542404            mov [esp+04], edx
00401B44: 890424              mov [esp], eax
00401B47: C78568FBFFFF1C000000 mov dword ptr [ebp-
00000498], 0000001C
00401B51: E8C41B0000          call 0040371A
00401B56: 8D45C0              lea eax, [ebp-40]
00401B59: 89442404            mov [esp+04], eax
00401B5D: C7042460184800      mov dword ptr [esp], 00481860
00401B64: C78568FBFFFF1D000000 mov dword ptr [ebp-
00000498], 0000001D
00401B6E: E8D9040700          call 0047204C
00401B73: C7042498094700      mov dword ptr [esp], 00470998
00401B7A: 89C1                mov ecx, eax
00401B7C: E8EFAE0400          call 0044CA70
00401B81: 83EC04              sub esp, 00000004
00401B84: 8D45C0              lea eax, [ebp-40]
00401B87: C78568FBFFFF1C000000 mov dword ptr [ebp-
00000498], 0000001C
00401B91: 89C1                mov ecx, eax
00401B93: E870D30400          call 0044EF08
00401B98: 8D45C4              lea eax, [ebp-3C]
00401B9B: C78568FBFFFF1B000000 mov dword ptr [ebp-
00000498], 0000001B
00401BA5: 89C1                mov ecx, eax
00401BA7: E85CD30400          call 0044EF08
00401BAC: C7042400000000      mov dword ptr [esp], 00000000
00401BB3: E8E0060200          call msvcrt.exit

* Jump:
 00401AD9(U)

00401BB8: 8D8578FCFFFF        lea eax, [ebp-00000388]
00401BBE: C78568FBFFFF05000000 mov dword ptr [ebp-
00000498], 00000005
00401BC8: 89C1                mov ecx, eax
00401BCA: E8B1750600          call 00469180
00401BCF: 8D4590              lea eax, [ebp-70]
00401BD2: C78568FBFFFF04000000 mov dword ptr [ebp-
00000498], 00000004
00401BDC: 89C1                mov ecx, eax
00401BDE: E825D30400          call 0044EF08
00401BE3: E9FE020000          jmp 00401EE6

* Jump:
 00401565(C)

00401BE8: 8D8598FBFFFF        lea eax, [ebp-00000468]
00401BEE: 89C1                mov ecx, eax
00401BF0: E88FF7FFFF          call 00401384
00401BF5: 83F801              cmp eax, 00000001
00401BF8: 0F94C0              sete  al
00401BFB: 84C0                test al, al
00401BFD: 0F84E3020000        je  00401EE6
00401C03: 8D856CFEFFFF        lea eax, [ebp-00000194]
00401C09: C78568FBFFFF1F000000 mov dword ptr [ebp-
00000498], 0000001F
00401C13: 89C1                mov ecx, eax
00401C15: E862BF0500          call 0045DB7C
00401C1A: C744240404000000    mov dword ptr [esp+04],
00000004
00401C22: C7042408000000      mov dword ptr [esp], 00000008
00401C29: E88E090700          call 004725BC
00401C2E: C744240402000000    mov dword ptr [esp+04],
00000002
00401C36: 890424              mov [esp], eax
00401C39: E87E090700          call 004725BC
00401C3E: 8B8DC8FAFFFF        mov ecx, [ebp-00000538]
00401C44: 8B5104              mov edx, [ecx+04]
00401C47: 8B0A                mov ecx, [edx]
00401C49: 8D956CFEFFFF        lea edx, [ebp-00000194]
00401C4F: 89442404            mov [esp+04], eax
00401C53: 890C24              mov [esp], ecx
00401C56: C78568FBFFFF20000000 mov dword ptr [ebp-
00000498], 00000020 ;' '
00401C60: 89D1                mov ecx, edx
00401C62: E86DBA0500          call 0045D6D4
00401C67: 83EC08              sub esp, 00000008
00401C6A: 8D856CFEFFFF        lea eax, [ebp-00000194]
00401C70: 89C1                mov ecx, eax
00401C72: E851BB0500          call 0045D7C8
00401C77: 83F001              xor eax, 00000001
00401C7A: 84C0                test al, al
00401C7C: 743D                je  00401CBB

* String: " Error opening main executable file 2
        "

00401C7E: C74424042C714700    mov dword ptr [esp+04],
0047712C
00401C86: C70424C0174800      mov dword ptr [esp], 004817C0
00401C8D: E87E010700          call 00471E10
00401C92: C7042498094700      mov dword ptr [esp], 00470998
00401C99: 89C1                mov ecx, eax
00401C9B: E8D0AD0400          call 0044CA70
00401CA0: 83EC04              sub esp, 00000004

* String: "pause"

00401CA3: C7042480704700      mov dword ptr [esp], 00477080
00401CAA: E8E1050200          call msvcrt.system
00401CAF: C7042401000000      mov dword ptr [esp], 00000001
00401CB6: E8DD050200          call msvcrt.exit

* Jump:
 00401C7C(C)

00401CBB: 8D45C8              lea eax, [ebp-38]
00401CBE: 8D956CFEFFFF        lea edx, [ebp-00000194]
00401CC4: 89542404            mov [esp+04], edx
00401CC8: 890424              mov [esp], eax
00401CCB: C78568FBFFFF20000000 mov dword ptr [ebp-
00000498], 00000020 ;' '
00401CD5: E8FE130000          call 004030D8
00401CDA: 8D4594              lea eax, [ebp-6C]
00401CDD: 8D55C8              lea edx, [ebp-38]
00401CE0: 891424              mov [esp], edx
00401CE3: C78568FBFFFF21000000 mov dword ptr [ebp-
00000498], 00000021 ;'!'
00401CED: 89C1                mov ecx, eax
00401CEF: E814D30400          call 0044F008
00401CF4: 83EC04              sub esp, 00000004
00401CF7: 8D45C8              lea eax, [ebp-38]
00401CFA: C78568FBFFFF20000000 mov dword ptr [ebp-
00000498], 00000020 ;' '
00401D04: 89C1                mov ecx, eax
00401D06: E8FDD10400          call 0044EF08
00401D0B: 8D856CFEFFFF        lea eax, [ebp-00000194]
00401D11: 89C1                mov ecx, eax
00401D13: E87CBA0500          call 0045D794
00401D18: 8D8570FFFFFF        lea eax, [ebp-00000090]
00401D1E: 8D5594              lea edx, [ebp-6C]
00401D21: 891424              mov [esp], edx
00401D24: 89C1                mov ecx, eax
00401D26: E82DCC0400          call 0044E958
00401D2B: 83EC04              sub esp, 00000004
00401D2E: 8D45CE              lea eax, [ebp-32]
```

```
00401D31: 89C1              mov ecx, eax
00401D33: E8C41C0400        call 004439FC
00401D38: 8D8598FBFFFF      lea eax, [ebp-00000468]
00401D3E: 89C1              mov ecx, eax
00401D40: E853F6FFFF        call 00401398
00401D45: 8D956CFFFFFF      lea edx, [ebp-00000094]
00401D4B: 8D4DCE            lea ecx, [ebp-32]
00401D4E: 894C2404          mov [esp+04], ecx
00401D52: 890424            mov [esp], eax
00401D55: C78568FBFFFF22000000 mov dword ptr [ebp-
00000498], 00000022 ;'"'
00401D5F: 89D1              mov ecx, edx
00401D61: E846CB0400        call 0044E8AC
00401D66: 83EC08            sub esp, 00000008
00401D69: 8D45CE            lea eax, [ebp-32]
00401D6C: 89C1              mov ecx, eax
00401D6E: E8951C0400        call 00443A08
00401D73: 8D8570FFFFFF      lea eax, [ebp-00000090]
00401D79: 89442404          mov [esp+04], eax
00401D7D: 8D856CFFFFFF      lea eax, [ebp-00000094]
00401D83: 890424            mov [esp], eax
00401D86: C78568FBFFFF24000000 mov dword ptr [ebp-
00000498], 00000024 ;'$'
00401D90: E8A7070700        call 0047253C
00401D95: 84C0              test al, al
00401D97: 0F84E6000000      je  00401E83
00401D9D: 8D45CF            lea eax, [ebp-31]
00401DA0: 89C1              mov ecx, eax
00401DA2: E8551C0400        call 004439FC
00401DA7: 8D8568FFFFFF      lea eax, [ebp-00000098]
00401DAD: 8D55CF            lea edx, [ebp-31]
00401DB0: 89542404          mov [esp+04], edx
00401DB4: C7042454714700    mov dword ptr [esp], 00477154
00401DBB: C78568FBFFFF25000000 mov dword ptr [ebp-
00000498], 00000025 ;'%'
00401DC5: 89C1              mov ecx, eax
00401DC7: E8E0CA0400        call 0044E8AC
00401DCC: 83EC08            sub esp, 00000008
```

```
00401DCF: 8D45CF            lea eax, [ebp-31]
00401DD2: 89C1              mov ecx, eax
00401DD4: E82F1C0400        call 00443A08
00401DD9: 8D45D4            lea eax, [ebp-2C]
00401DDC: 8D9568FFFFFF      lea edx, [ebp-00000098]
00401DE2: 891424            mov [esp], edx
00401DE5: C78568FBFFFF26000000 mov dword ptr [ebp-
00000498], 00000026 ;'&
00401DEF: 89C1              mov ecx, eax
00401DF1: E862CB0400        call 0044E958
00401DF6: 83EC04            sub esp, 00000004
00401DF9: 8D45D0            lea eax, [ebp-30]
00401DFC: 8D55D4            lea edx, [ebp-2C]
00401DFF: 89542404          mov [esp+04], edx
00401E03: 890424            mov [esp], eax
00401E06: C78568FBFFFF27000000 mov dword ptr [ebp-
00000498], 00000027 ;'''
00401E10: E805190000        call 0040371A
00401E15: 8D45D0            lea eax, [ebp-30]
00401E18: 89442404          mov [esp+04], eax
00401E1C: C7042460184800    mov dword ptr [esp], 00481860
00401E23: C78568FBFFFF28000000 mov dword ptr [ebp-
00000498], 00000028 ;'('
00401E2D: E81A020700        call 0047204C
00401E32: C7042498094700    mov dword ptr [esp], 00470998
00401E39: 89C1              mov ecx, eax
00401E3B: E830AC0400        call 0044CA70
00401E40: 83EC04            sub esp, 00000004
00401E43: 8D45D0            lea eax, [ebp-30]
00401E46: C78568FBFFFF27000000 mov dword ptr [ebp-
00000498], 00000027 ;'''
00401E50: 89C1              mov ecx, eax
00401E52: E8B1D00400        call 0044EF08
00401E57: 8D45D4            lea eax, [ebp-2C]
00401E5A: C78568FBFFFF26000000 mov dword ptr [ebp-
00000498], 00000026 ;'&'
00401E64: 89C1              mov ecx, eax
00401E66: E89DD00400        call 0044EF08
```