Utilizing Reverse Engineering in Tracking Software to diagnose its weaknesses

Rana Jumaa Surayh Al-janabi
Al-Qadesya University- College of Medicine

Hawraa Adil Nori
Babylon University- Computer Center

Abstract

In this research, Video Cutter Software was analyzed and that software uses a well known protection method called "User Name and Registration Number". This software uses kind of complicated key generating algorithm. The analyzer hits upon the protection routine of interested software and adds suitable code to them. In other words, he discovers where in memory the entering data is stored and then to find out what is done with it. In this research, a merge technique was adopted between code injection and key generation i.e. (code is added to the execution file) in order to make the software itself browse message that contains registration number and this is in turn would consume less time than expected in analyzing (for key generation algorithm). Also, this research discusses how to support software protection by using anti-reverse engineering techniques to prevent crackers from license code stealing.

الخلاصة

في هذا البحث, تم تحليل برنامج تقطيع ملفات الفديو والذي يستخدم أسلوب حماية معروف (اسم المستخدم وكلمة المرور).هذا البرنامج يستخدم خوارزمية توليد مفتاح نوعاً ما معقدة. المحلل اكتشف برنامج الحماية الفرعي للبرنامج الرئيسي وقام باضافة شفرة مناسبة لهم. بمعنى اخر ان المحلل اكتشف مكان البيانات المدخلة (اسم المستخدم وكلمة المرور)في الذاكرة وماهي العمليات التي تمت عليها. في هذا البحث, تقنية الدمج بين طريقة حقن الشفرة ومولد المفاتيح استخدمت اي (تم اضافة شفرة الى الملف التنفيذي) لجعل البرنامج يعرض رسالة تحتوي على كلمة المرور الخاصة به وهذا بدوره سوف يستهلك وقت اقل من المتوقع في التحليل(لخوارزمية توليد المفتاح). ايضا, هذا البحث ناقش طرق لتعزيز حماية البرنامج من خلال استخدام تقنيات مقاومة الهندسة العكسية لمنع المخترقين من سرقة مفتاح الترخيص.

Introduction

Reverse engineering is the art that taking apart an object to see how it works in order to duplicate or enhance the object. This practice is taken from older industries but now, it is frequently used on computer hardware and software.[1]

Hardware reverse engineering involves taking apart a device to see how it works. For example, if a processor manufacturer wants to see how a competitor's processor works, they can purchase a competitor's processor, disassemble it, and then make a processor similar to it. In general, hardware reverse engineering requires a great deal of expertise and is quite expensive.[1]

Software reverse engineering is the process of understanding the intricacies of designer's program or even commercial software at a lower level than the compiler. It involves reversing a program's machine code (the string of 0s and 1s that are sent to the logic processor) back into the source code that it was written in, using programming language statements. Software reverse engineering is very important since it can be used to retrieve the source code of lost projects, to study how the program performs certain operations, to improve the performance of a program, to fix a bug (correct an error in the program when the source code is not available), to identify malicious content in a program such as a virus , to adapt a program written for use with one microprocessor for use with another, or to try to break the protection of software to determine weaknesses in that software in order to reinforce its protection.[1,13]

Software Revere engineering theories may be used to inject code in software or build a key generator. By code injection the breaker can insert code into software to change the course of execution, while key generator is a program written for specific software, so the

user can enter any name and then have the registration code for that name. In case Code Injection is used to attack, the result can be disastrous, for instance, it's used by some malicious software to propagate. [12]

In this research, Reverse engineering is used to analyze video cutter software and explain how to break it in order to clarify its weaknesses and suggest a method to support its protection and this method of protection can be explained in details in the next researched because it's very complicated and need another research.

Tools & Method

1- Tools: In this research, the following tools were used:

a. OllyDbg: is the most widely used program for the debugging purposes, so it's a debugger that emphasizes binary code analysis, which is useful when source code is not available. It traces registers, recognizes procedures, API calls, constants and strings, as well as locates routines from object files and libraries. The software is free of cost, OllyDbg is downloaded from http://www.ollydbg.de/odbg110.zip. Let's take a look at the whole program by showing the windows compromising it as illustrated in figure (1) :**-[8]**

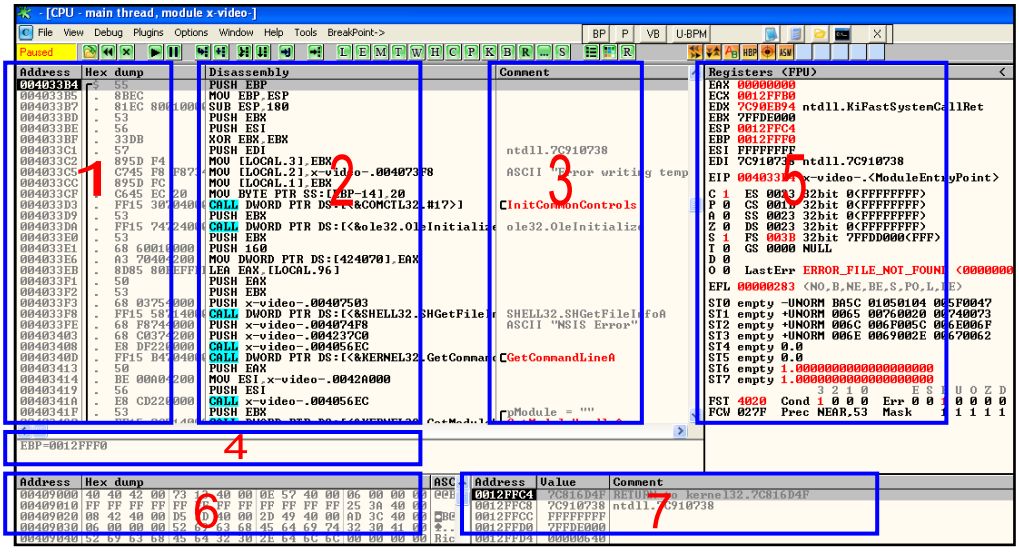

Figure1: Main OllyDbg Window

1: addresses of assembly instructions window
2: commands and instructions in assembly language
3: comments window
4: information window
5: registers window: involves general purpose registers, EIP register which always points to instruction currently executed, segment registers, flags register and
another types of registers.
6: dump memory window: which contain the addresses, hexa representation and the ASCII corresponded to them.
7: Stack window

b. PEiD : It detects most common packers, cryptors and compilers for PE files. This program is used for examining, as it reveals whether or not the program is protected, if protected, it will determine the type of protection method and if not, it will determine the programming language used to write program. PEiD is downloaded from http://www.peid.info/files/PEiD-0.94-20060510.zip. [11]

c. **Video Cutter Software**: is the attacked program. Video Cutter is a powerful software that could assist user to select and cut video segments of his favorite video file, and cut out the segments he dislike. It is downloaded from http://www.xilisoft.com/video-cutter.html

2. Method: It includes two stages as explained in figure (2):-

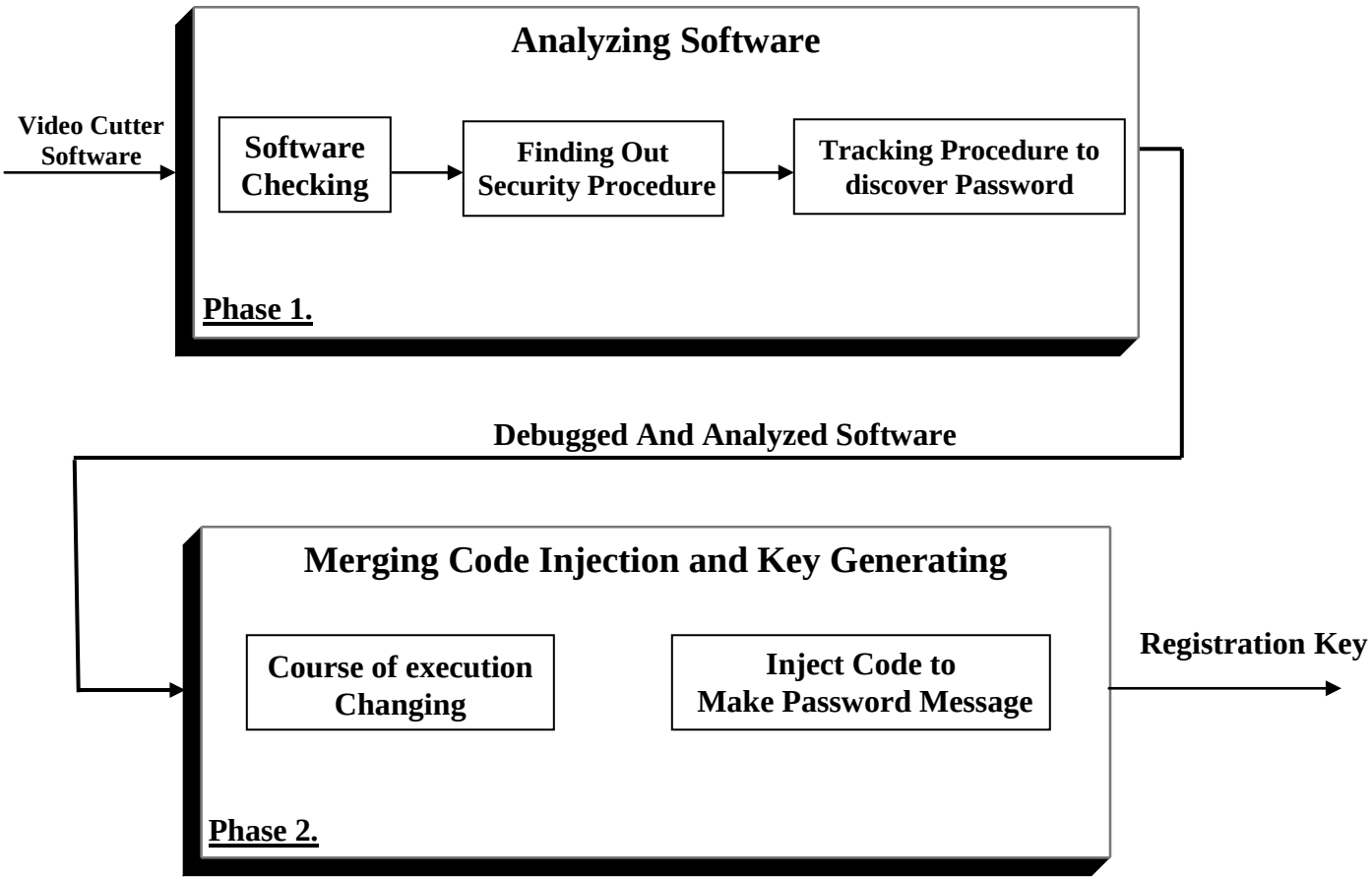

Figure (2):- Block Diagram that explains the main stages for tracking software to obtain registration key

a. **Analyzing software:** In this phase, Analyzers concern with problem definition, requirements gathering and analysis.[3,9]

1. checking video cutter software with PEiD program as in figure (3). The analyzer finds out that the software is written in visual C++7.0 and the software is unpacked.

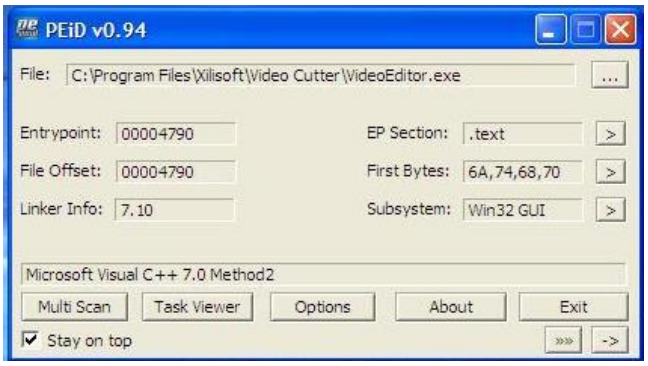

Figure3: Main PEiD Window

2. Opening video cutter software using OllyDbg, as illustrated in figure (4).

Figure 4: Executable file opening in OllyDbg

3. Through debugging software, by entering wrong (user name and license code) then register them, we face a message that tell us the registration is wrong. Analyzer pause the debugger then choose view → call stack to give him the ability to know from which procedure that message appear

4. depending on step(3) , the analyzer discovered that the module (UILib8-M) is responsible for key generation function and founds that the size of serial number that he entered it will be compared with 27hex (39 in decimal) and so the analyzer choose that module as illustrated in figure (5):

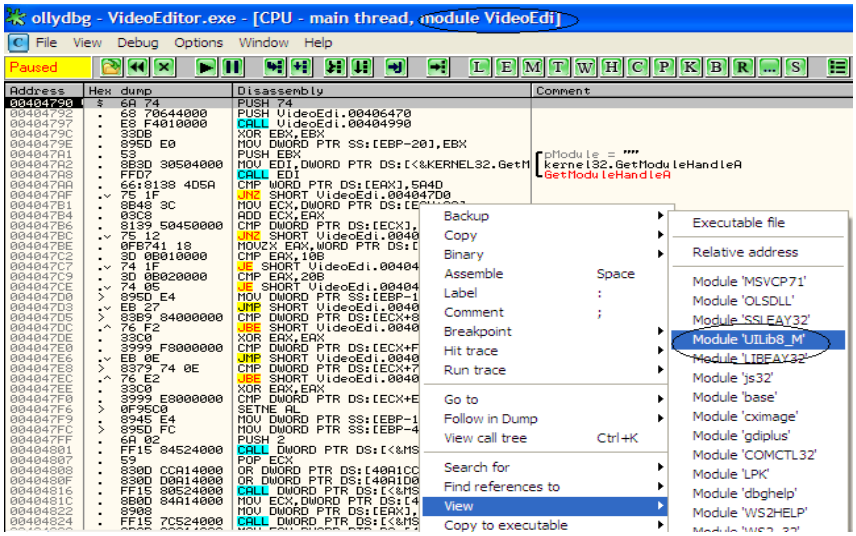

Figure (5) Explain transferring to module UILib8-M

5. According to what is mentioned in step 4. the analyzer choose compare instruction (cmp eax, 27) and put breakpoint on it, as illustrated in figure (6).

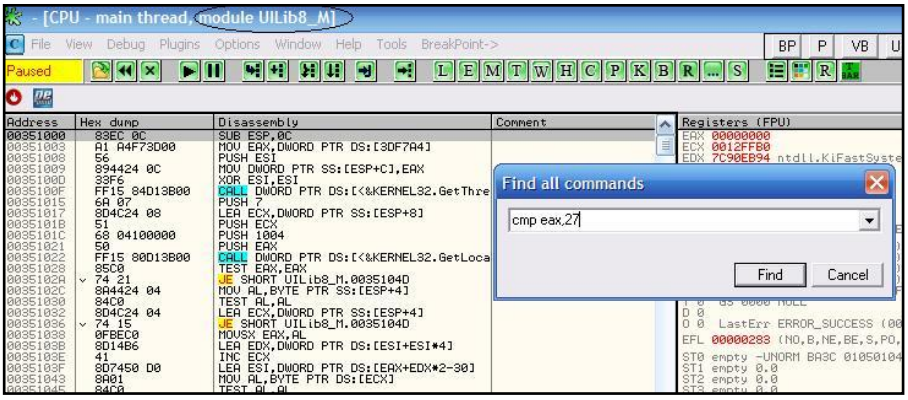

Figure (6) : how to search for instruction

6. To put break point, the analyzer presses F2 to each instruction that he selects and after that double click on any of these instructions to return to (UILib8-M) module, as illustrated in figure (7).

Figure (7) :How to put breakpoint

7. As the same way that illustrated in step (4). Analyzer choose compare instruction (cmp esi, 20) put breakpoint on it. To calculate license code

8. The analyzer runs the software using F9 and then uses F8 until reach to software interface. As it is mentioned in step (4) he enter (39 character), as illustrated in figure (8).

Figure (8): wrong password but it should be (39 char)

9. using F8 until analyzer reaches to password that is shown in stack window as illustrated in figure (9)

Figure (9) : password was appeared

b. Merge code injection and key generating: in this phase, the analyzer uses an easy way to add code to video cutter software which browses a Message Box that shows password to user. The method involved these steps:- [2,4]

1. After password was appeared in previous phase, the analyzer finds (JMP instruction) and changes it to refer to main module (VideoEdi) at address (00404f96) using assemble command as illustrated in figure (10).

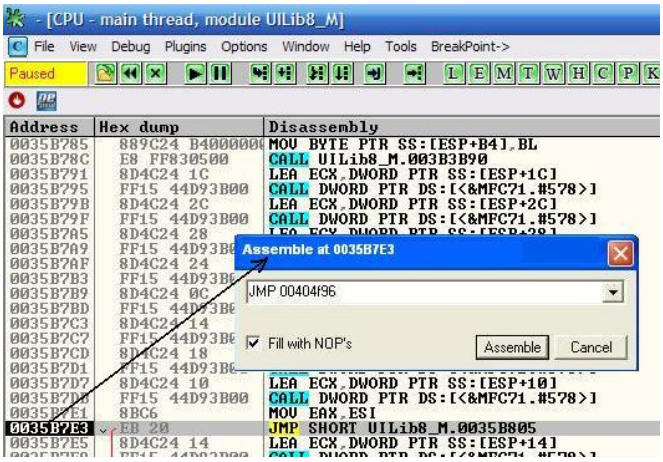

Figure (10) Using assemble command code can be changed

2. Each message box has a title. There is a way to add that title by selecting an empty spaces (contains zero's) in memory dump window and press space key so windows of edit data is appeared then the title can be written in (ASCII field) as explained in figure (11)

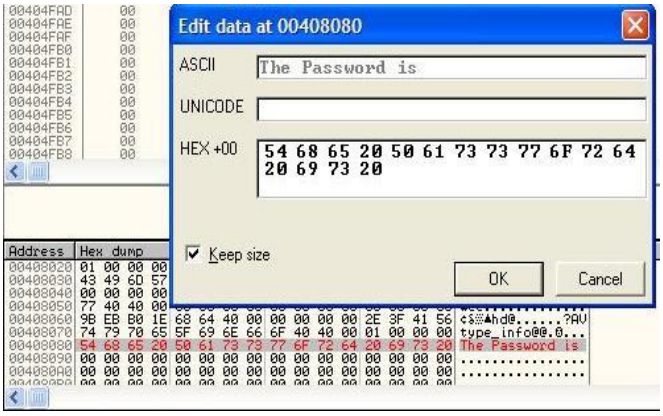

Figure (11) :How to insert data

3. The next step was to transfer control to main module (VideoEdi) at address (00404F96) in order to add the following code using assemble command too, as illustrated in figure (12).

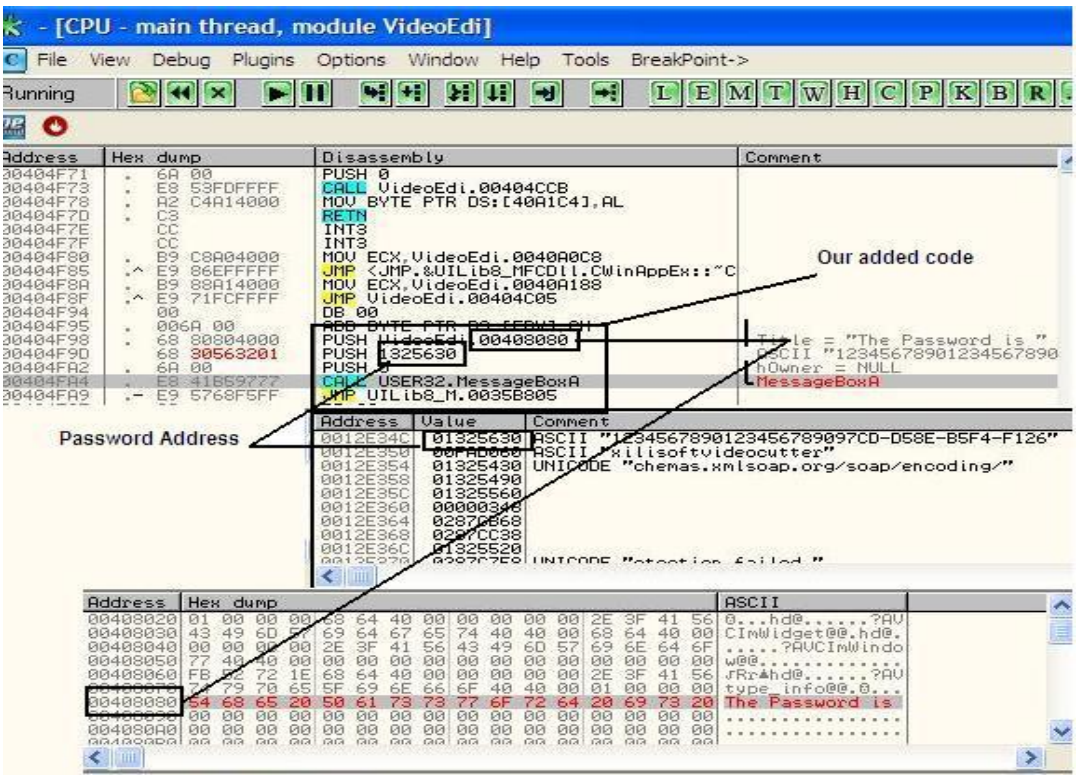

Figure (12): adding our code

4. By running software the password message Box was appeared as illustrated in figure(13)

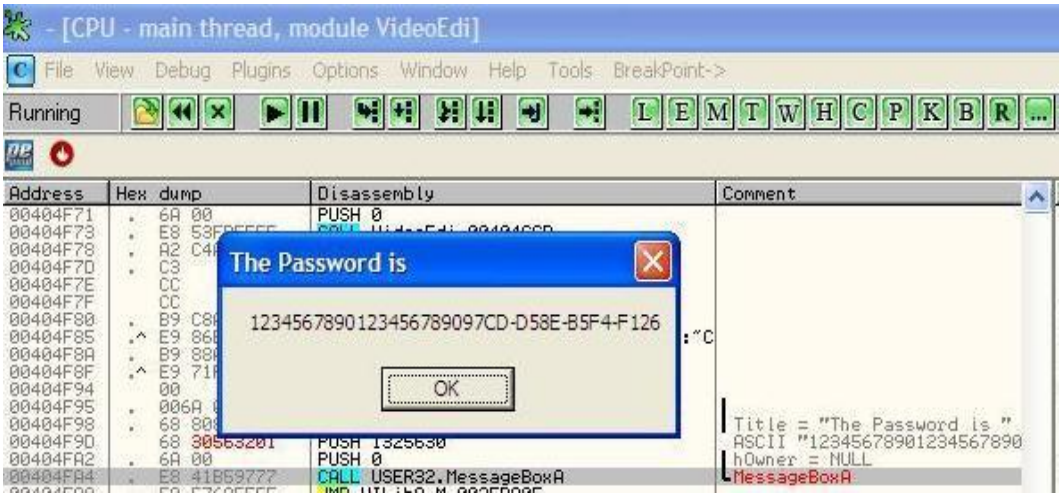

Figure (13): Password is appeared in a Message

Result

In this research, code injection technique is used to discover a password in an easier way and without needing to analyze the details of key generator as well as, this research explains the general steps to track software that doesn't have source code.

Actually, by utilizing user name and password in original software (video cutter). It is registered as illustrated below in figure (14)

Figure (14): Software was registered to user name (reverse)

In fact, there is no correlation between user name and license code because of using the same license code with different user name the software was registered as explained in figure (15). So, software designers use kind of different way in key generating because it doesn't depend on user name.

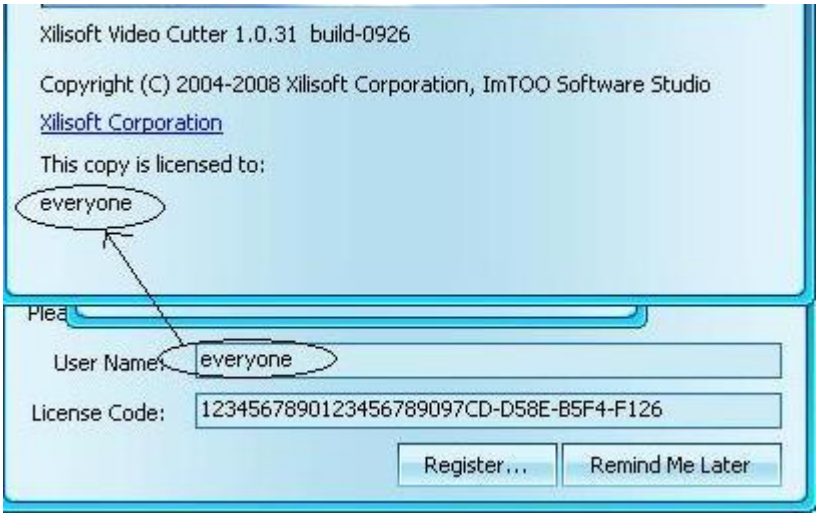

Figure (15): using the same license code software was registered to user name (everyone)

Suggestions for Obtaining More Secure software

Software security is first and foremost about identifying and managing risks. One of the most effective ways to identify and manage risk for an application is to iteratively review its code throughout the development cycle. [10]

In fact, there is a question that always repeated" What can we do to secure our software completely". Unfortunately, there is no single or easy answer to that question. Application security must be dealt with and considered in every phase of the software development cycle (SDLC). Security in the SDLC is essential, so that Microsoft has bestowed upon it a completely new name: the Secure Development Lifecycle, or SDL.[5]

The Secure Development Lifecycle attempts to marry the pillars of the original SDLC with fundamental secure practices throughout the lifecycle. This practice will create an application that has a secure core and can better withstand attacks and protect against reverse engineering. To achieve that protection, many protection techniques are used such as anti-debug, anti-dump, and encryption layers, and also the ability to choose which protector the executable should appear as in PEiD and other signature-based scanners. Thus, packers and protectors try to slow down attackers as long as possible. [5,6]

In fact, there is an ongoing battle between the coders who develop programs that protect against cracking, reverse engineering and the engineers themselves. Every time the protectors release a new technique, the engineers find a way around that specific method. This is the driving force behind the cracking "scene" and anti-reverse engineering fields. Here are some of protection techniques researchers suggested to make video cutter software more secure, these techniques are almost used by protectors:- [7,14]

1. **Encryption Layers**
   To protect applications against analysis, packers and protectors often use encryption layers. Usually, in a manner similar to viruses, polymorphic engines

are employed to generate a different crypt/decrypt algorithm for each protected application. Two different kinds of encryption are usually observed:

A. **Loader encryption**

The protection code resides in the loader. To protect against static analysis and modifications of the underlying code and protections, the loader is encrypted, usually many times. Therefore, it is not possible to directly patch the code underneath. The loader can be split into many parts, each of them encrypted by many layers.

B. Application encryption

Like the loader, the application is also encrypted to prevent disassembly and modifications. Although the application can be encrypted with many layers, most of the time it has only one or two layers. On the other hand, the loader may vary from a couple of layers to a few hundred. After parts of the loader have been executed, they can be re-encrypted or destroyed, so that a fully decrypted loader is never in memory at any time.

2. **Obfuscation Techniques**

One of the first tricks that appeared in packers was code obfuscation, designed to slow down analysis. Techniques are used to scramble the code, making it hard to read, follow, and debug. Many techniques exist such as junk code (as its name suggests, it utilizes code that is junk or not needed to confuse a reverse engineer as to what the current code is actually trying to accomplish. When the junk code that is inserted into a routine is convincing and successfully manages to confuse a reverse engineer).

3. **Anti Debugging Techniques**

Using a debugger, it is possible to single-step through applications, and inspects their code in real time. This is obviously a problem for packers and protectors, since it enables an analyst to reverse-engineer them. To counteract this, anti-debugging tricks are used.

A. **IsDebuggerPresent**

Despite being inefficient, the IsDebuggerPresent API function was very common in the first packers and protectors, and some of them are still using it as a first-stage check.

B. **BreakPoint Detection**

Another common technique is the detection of software breakpoint.

C. **Timing Attacks**

The theory behind timing attacks is that executing a section of code, especially a small section, should only take a miniscule amount of time. Therefore, if a timed section of code takes a greater amount of time than a certain set limit, then there is most likely a debugger attached, and someone is stepping through the code.

4. Anti-Dump Techniques

Anti-dump refers to protections preventing process dumping or techniques used to render the dumped executable unusable. Such protection is done either at runtime or protection time.

Actually anti –reverse engineering techniques can not be covered in detailed here, it needs another research. And finally, it is worth to mention that reinforcing security of the software is really necessary issue but also software coders should compromise between the cost of software security and security itself. As well as software coders even

if they try to protect their software as much as they can, that software can be considered more secure (not easy to break) but it is not completely secure.

Conclusion

Actually, this research discusses two issues. The first is how crackers can exploit software weaknesses to attack software, so the analyzer focus on the specific code that he really need for his own purpose because the entire software analysis requires too much time and efforts consuming. So, the merge mechanism that the researchers use in this paper has the benefit of avoiding analysis of complete key generator algorithm. Thus researchers explain how crackers can attack software without needing to analyze the entire software.

Second, using of reverse engineering by crackers can cause complicated problems to software production companies by stealing their license code, so it is necessary for those companies to use anti- reverse engineering techniques to make their software more secure and also the companies themselves should use reverse engineering in all phase of software development cycle to test their software security. So reverse engineering can be used to support and also attack security.

References

1. Arleigh Crawford," Reverse Engineering", http://searchcio-midmarket.techtarget.com/sDefinition/0,,sid183_gci507015,00.html , 2007.

2. Crudd, "Code Injection", http://www.reteam.org/papers/e19.pdf, 2002.

3. George N., Glafkos C., "Reverse Engineering: Anti-Cracking Techniques", http://www.reteam.org/papers/e19.pdf, 2008.

4. Jame's, "Cracking and Patching With OllyDbg", http://www.youtube.com/watch?v=wGXh 2kozjMA&feature=related, 2008.

5. Joe Basirico," Application security shouldn't involve duct tape, Band-Aids or bubble gum", http://searchsoftwarequality.techtarget.com/news/article/0,289142,sid92_gci1254902,00.html, 2007.

6. Joren McReynolds, Packer Detection and Generic Unpacking Techniques, http://securitylabs.websense.com/content/Blogs/2927.aspx, 2008.

7. **Josh Jackson**, " An Anti-Reverse Engineering Guide", http://www.codeproject.com/KB/security/AntiReverseEngineering.aspx, 2008.

8. Master, "How to use OllyDBG", http://otfans.net/showthread.php?p=755360, 2007.

9. Oharan, "How to crack a simple serial", http://www.youtube.com/watch?v=vr41J1-nVKs, 2007.

10. Steven Lavenhar,"Code Analysis" , https://buildsecurityin.us-cert.gov/daisy/bsi/articles/best-practices/code/214-BSI.html, 2008.

11. Toralf, "PEiD - detect common packers, cryptors and compilers", http://www.autohotkey.com/forum/topic17879.html, 2007.

12. Wikipedia, "Code Injection", http://en.wikipedia.org/wiki/Code_injection, 2008.

13. **Wikibooks**, Reverse Engineering/ Introduction, http://en.wikibooks.org/wiki/Reverse_Engineering/Introduction, 2009.

14. Win32 Portable Executable Packing Uncovered, http://securitylabs.websense.com/content/Assets/HistoryofPackingTechnology.pdf.