International Journal of Computer Applications (0975 – 8887)
Volume 166 – No.8, May 2017

Software Watermarking based on Return-Oriented Programming for Computer Security

Ashwag Alrehily and Vijey Thayananthan
Computer Science Department,
Faculty of Computing and Information Technology,
King Abdul Aziz University,
Jeddah 21589, Saudi Arabia

ABSTRACT

Currently, the major problem for software developers is software piracy. To protect software from piracy, many techniques are developed, and one of them is software watermark. Software watermark provides authentication and copyright protection by embedding a watermark into the software, and the owner can claim ownership of the software by watermark extraction. The software watermarking improves the computer security with a robust solution to expose the unauthorized modification or illegal copying of different kind of attacks. Now, there many techniques for embedding and extracting watermark into software and most recent one used malicious code like return-oriented programming (ROP) for good uses. Moreover, any software protection with an efficient watermarking algorithm based on ROP is a relatively new branch of computer security. Thus, in this paper, new software watermark has been designed using ROP technique that enhances the existing one. The watermark has been embeded using ROP and it has been extracted once ROP trigger is triggered. ROP trigger uses a SHA256 hash function to compare between watermark secret input and user entered key. As a result, the proposed work has strong resilience, Stealth and minimum runtime overhead.

General Terms

In this paper, ROP is considered as my general term. Throughout this research, software watermark is considered to improve the computer security.

Keywords

Computer security, Software watermark, Return-oriented programming and Secure Hash Algorithm.

1. INTRODUCTION

In computer security, ROP is one of the most popular techniques which not only prevent the code-reuse attacks but also return-to-lib(c) attacks. In this research, the ROP attacker locates specific code sequences inside the binary, then places their addresses onto the program stack, writing the appropriate return instruction itself to transfer control flow from one gadget to the next. To implement this concept, only a subset of this functionality which allows us to perform the further experiments, should be developed. Generally, ROP is used to bypass measures that prevent code injection such as data execution prevention. However, a classic code injection attack is one of the attackers' targets in the computer security issues. According to the recent Security Intelligence Report released by Microsoft, ROP played an important role in preventing the code injection attacks on Microsoft products between 2012 and 2014. Since all current major operating systems implement some form of data execution prevention mechanism, ROP is now practically required for any arbitrary code execution attack.

Computer software has become in every people's daily life, and software piracy is becoming a serious issue for enterprises. There are many techniques to protect the software, and one of them is software watermark. It is a process of embedding a secret message in the source code of the program; the secret message can be extracted to identify the information about copyright owner of the software such as author, publisher, and owner. Embedded watermark in software should not affect the flow of the program or make any redundant space that will affect the program high-quality [1]. To add watermark into software there are two important processes must perform:1) embed a watermark into software.2) extract the watermark from software. More precisely, assume that W is the watermark, P is the Program and K are the secret input; the watermark W embeds into a program P plus the secret input K to produce watermarked program Pw. The following function describes the watermark embed.

$$\text{Embed}\ (P, W, k) \rightarrow Pw.$$

And to display the copyright of the software owner the watermark W can be extracted from the watermark program Pw by watermark extractor and secret input K. The following function describe the software watermark extraction [2].

$$\text{Extract}\ (Pw\ , k) \rightarrow W.$$

There are two types of software watermark techniques: 1) Static software watermark technique which is embedded the watermark in the target application executable such as the text section and initialized data 2) Dynamic software watermark technique which is embedded the watermark in the program execution state or dynamic data that gives the program a new path to execute which contain the watermark [2]. It is proudly believed that the dynamic watermark is more reliable and secure solution because the hidden message retrieved by running and examining the specific behavior of specific path of the watermarked program [3].

One of the more resilient and stealthy over existing techniques in dynamic watermarks is software watermarking using ROP [4] which designs watermarking code to look like normal data and triggered to execute. After triggered, the hidden watermark message recovers by the pre-constructed ROP execution. Using ROP with its instructions in a program can create unexpected executions (Path) which are invisible functionally.

1.1 Motivation
Although ROP is used in the existing software watermark, following points motivate us to develop an appropriate technique for improving computer security systems. In this research, first motivation is an existing technique which has a disadvantage in the trigger function because they designed a very simple trigger function which makes attacker easy to locate and analyze watermark. The second motivation is about the uses of the malicious code where ROP had limited solutions. It means that this technique is the only existing technique that uses malicious code (ROP) into good uses. Thus, in this paper, the motivation is to enhance the computer security through the efficient technique.

1.2 Contribution
In the proposed technique, new software watermark using ROP designed and implemented with more efficient trigger function as the main contribution. To complete this research, some more points as additional contributions have been considered. They are elaborated here. Adding ROP in the program has the advantage of not creating a new data structure that makes code suspicious comparing between watermarked program and the original program. The watermark embeds into the program using ROP functions that use existing instructions which convert program normal execution path into ROP execution path. The unexpected path will not affect the normal execution of the program. The watermark extraction using a hash function in trigger function.

1.3 Paper organization
The paper is designed as follow; Section 2 includes the related work of previous research and some related idea of the proposed research. Section 3 includes proposed work. Section 4 illustrate proposed work design and implementation then section 5 includes result and discussion, and the last section is a conclusion of the proposed work.

2. LITERATURE REVIEW
When program behaves differently or crashed, that means the program has some bugs or errors. Software errors put the program in unsafe environment and lead to a security leak. There are different software errors one of them is memory error which causes software to crash in certain situation such as buffer overflow or stack overflow. Security bug or vulnerability is a defect in a system’s security which make the system open to changes and easy way for an attacker to manipulate the system to do unintended functions. These changes can affect the system in different part such as availability of the system, gain complete control of the system, change some privileges to access unintended level, and many other parts[5].

Regarding static software watermark, many types of research have been proposed. In [6] where authors have proposed it based on equation reordering. The watermark algorithm idea is to reorder the operands of the equation and hide the watermark data and be sure not change the result of the equation. The order of equation operands is changed according to the watermark information. The algorithm hides the watermark in the source code without adding any additional codes that protect the execution of form program slowdown and size change, and it is blind. However, it has limited capacity for hiding the information, but according to [7], authors have improved the efficiency of the hidden information by presented software watermark based on Coefficients of Equation. The algorithm reorders the operand coefficients of the equation if it is in a safe swappable form and it is also a blind algorithm, but the order of the operand coefficients in the equation will change according to a particular order that is from small to big. However, the existing watermark based on equation re-sort algorithm has in fact easy attacked by using the random re-sequencing technology. In [8] researchers solved this problem by proposed a new software watermark that decomposed watermark using Chinese remainder theorem and hid in the source code without any additional codes. However, the evaluation shows the robustness of the watermark; it did not affect the code length or the speed of the program, and it has better performance than [6] [7].

Regarding the software watermark in dynamic form, Collberg et al. [2] first introduced the dynamic software watermark in the form of graph based watermark. This approach stores the watermark in the form of graph structure created at execution time and then using the graph to extract the watermark. Another type of dynamic technique is the branch based watermarks as in [9] where authors proposed a new algorithm based on Shamir threshold scheme and branch based dynamic watermarking (STBDW). The watermark process goes into many phases unit embedded the watermark into the program. The watermark number divided into many pieces and the insertion of the watermark will start by execution the program with the secret sequence to find all the local variables value and to generate code to insert the watermark in the appropriate position. Moreover, the authors prove the robustness of the proposed algorithm after implementing the algorithm. At last, there are other categories of software watermark such as register-based software watermark, thread-based software watermark and obfuscation-based software watermark [10].

Another researcher uses a hash function to embed a watermark in dynamic approach such as [11]. They proposed a new software watermark algorithm based on hash function; the proposed watermark satisfy three conditions to be more secret and robust. The three conditions are the information of the watermark cannot be in the static form in the program, it must be a logical relation between the watermark and program and the information of the watermark is unique. They selected to be logical relation because if there is watermark damage, the program will behave differently. The watermarking algorithm started by segmentation the watermark. The watermark segmentation divides the watermark into smaller parts then construct the hush function after that embedding the watermark by generate code for the hash function and embed it into the program in proper position. The authors evaluate the robust of the proposed watermark under three different categories of attacks: subtractive, distortion and additive attacks and they found the algorithm resisted the attacks.

The main idea of the proposed work taken from the recent proposed dynamic software watermark in [4] they use ROP and turn it into a good use and created new software watermark. The proposed software watermark collect the watermark codes from the existing code, in the data region the ROP convert codes into ROP gadgets and build them; these codes lead to unexpected execution path. The program modified to arrange all other resource needed to follow this unexpected execution path (watermark path) on-the-fly. To extract the watermark, the trigger must happen by using the secret input to activate ROP execution after that the program control transfer to hidden ROP path. They evaluate the proposed software watermark with the previous works, and the evaluation shows the better stealth and good resilience.

3. PROPOSED WORK
In this section, the methodology of the proposed work explains in detail. watermark program develops using ROP technique, and the proposed ROP trigger is added.

3.1 Return Oriented Programming (ROP) Overview
Exploitation process or exploit is the process of taking control of program flow by changing the instruction pointer of the program. An attacker can exploit some software errors to take control of a program or to modified the software behavior. change the instruction pointer cannot be performed directly, it must be some bugs in the program. A bug in the program acts as an entry point to allow an attacker to exploit the program and provide some input to the program [12]. There are a different number of attack techniques have been created in the recent years to enable an attacker to successfully exploit the vulnerability to change the control flow such as return oriented programming ROP.

In [13] invented new technique called return oriented programming (ROP) for malicious purpose, and it becomes a most important technology today. ROP was used on many platforms such as x86 architecture, SPARC [14]. ROP based on a return to Libc attack which uses functions from the libc library. it created to works against the protection mechanisms such as W⊕X and DEP because these mechanisms work as protection for the operating system from code injection attacks, and ROP was the best fit since it has no need to inject new code.

ROP code consists of two pieces which are gadgets and Payload.

Gadget: ROP uses gadgets to build their code, and the gadgets are an orignazed unit, a small piece of instruction that located somewhere in the executables such as system libraries. Each gadget has certain value, and they should end with indirect call or jump or return example of gadgets: < pop %eax ,ret>

The gadgets sequences or called instructions perform a specific operation. ROP Gadget operation is the following:

- load and store:

There are three cases two loads and one store : load from the memory location the content to register ,load a constant to register and wrtie to the register the content of memory loaction.

- Arithmetic and Logic:

Such as add, or, not, exclusive or and shift.

Payload: Well-crafted bit string the take all selected gadgets address for selected execution.

ROP code works by linking sequence of gadgets together in the stack. Each gadget end with ret instruction that allows the attacker to chain the instruction sequence together in the stack. The stack pointer esp pointed to the first instruction of the ROP and executed it then the ret instruction invoke the second gadget to executed it and so on. Thus, the esp controlled the execution flow and worked as the program counter.

The ROP code format in a form that cannot be run directly but by its own instructions to create unexpected execution path in the program, the unexpected execution path created by ROP is invisible which makes code hard for an attacker to understand. Therefore, it is a fact that the ROP is the most famous technique used in malicious uses. And the latest research proved that the ROP could be used not only for malicious uses but also in benign uses like software watermark [4].

3.2 Watermark Program Based On ROP
The proposed work take into consideration the following issues:

- The ROP code must be not suspicious comparing the proposed watermark program with the program before adding the watermark.
- After watermark message extraction, the program should return back to the normal program execution path.

Watermark program based on ROP build by preforms four main steps which are located gadgets then create ROP payload and chain the payload in the program and finally build ROP trigger to change the execution path to the ROP path. Figure 1 shows the proposed work building steps.

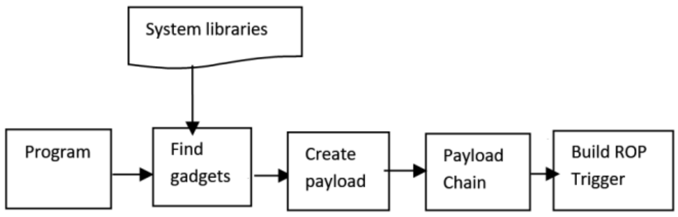


Figure 1: Proposed work building steps.

The first three steps of building ROP program responsible for embedding watermark into the program and the final step responsible for extracting the hidden watermark. In the proposed work, the watermark embed is a simple watermark and the idea of the watermark taken from [4], the proposed watermark message is “007” and stored in string S, not as integer number “007” but stored as ASCII code of the integer number “007” which is “484855”. To embed watermark, three important gadgets have been selected to control the watermark embedding and building the proposed ROP payload around it; the following points describe the watermark embed process (see Figure 2):

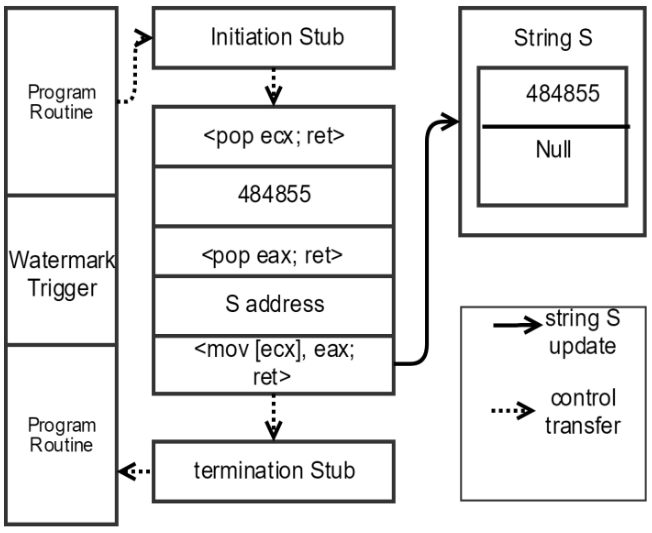


Figure 2: proposed ROP based watermark.

- First the initiation stub in charge for two functions: first, save the current stack pointer address to return to after extract watermark. second, transfer and change the current stack pointer to point to watermark code;
- The watermarking code consisting of three gadgets to complete a memory writing chain <pop ecx; ret>, <pop eax; ret> and <mov [ecx], eax; ret>;
- Give ecx register the address of string S and eax register the ASCII code constant 484855 corresponding to watermark message “007”;
- Last the termination stub in charge for return the execution to the regular program by recovering the original stack pointer.

The watermark extraction procedure triggered by ROP function which is the final step of building ROP program, the watermark extraction loads the embedded watermark ”4848455” by ROP and the address of string S into register ecx and eax, respectively, then updates the string S to the watermark message “007”.

The four main steps of building the proposed work are the following:

1- Finding watermarking gadgets:
Gadgets are series of instructions ending with return (RET) existing in the system libraries and each instruction reasonable for a specific operation. There are two steps required to finding the gadgets: first, select system libraries to search for all or specific gadgets and second, select a suitable gadget from selected system libraries that can achieve specific operation based on program need. Thus, in the method, two libraries have been picked the kernel32 and ntdll because of both libraries available and important in windows OS also they have the suitable gadgets. The kernel32 library is responsible for handling memory usage in windows operating system and without it windows cannot work properly. The ntdll library is the most important file in windows operating system. It is responsible for handling system tasks, and it has and some kernel mode function that allows (API) Windows Application Programming Interface. In the method, it only needs gadgets that perform memory writing, register loading and transferring the instruction pointer; Table 1 shows how many gadgets both libraries have, the count based on the gadgets that can complete a functional chain.

Table 1: number of gadgets in system libraries.

libraries	Memory writing	Register loading	Transfer
kernel32.dll	2	13	3
ntdll.dll	6	10	8

To locate gadgets, the existing programs Immunity Debugger program has been used with Mona.py tool to help to search for gadgets [15] [16]

2- ROP Payload:
Creating watermark in ROP need a lot of gadgets to be executed, thus after selecting all the gadgets required to embed watermark, ROP payload must build to chain all the gadgets. The payload needs to be of reasonable length, and the code constructed payload must be short because long payload code will be attracted to attackers. Also, ROP payload must be well crafted, and the selected gadgets must be executed in sequence to ensure do not affect the program execution. In the design, the ROP payload has fewer gadgets, and three dummy functions have been built to use as a distraction to attackers. Thus, the most advantage of the proposed ROP based watermark is to execute, embed a watermark and call the three dummy functions throughout the execution path which makes the proposed ROP based watermark hard to find any suspicious code in the program analysis. ROP payload could be stored in an integer array, an object of the specific class or character strings, since the proposed ROP payload has fewer gadgets. Character string has been selected to store the watermark payload in the static region, figure 3 shows how the watermark executed. As we can see, the watermark payload contains gadgets to embed watermark also the address of the three dummy; watermark payload embeds the watermark before calling the dummy functions.

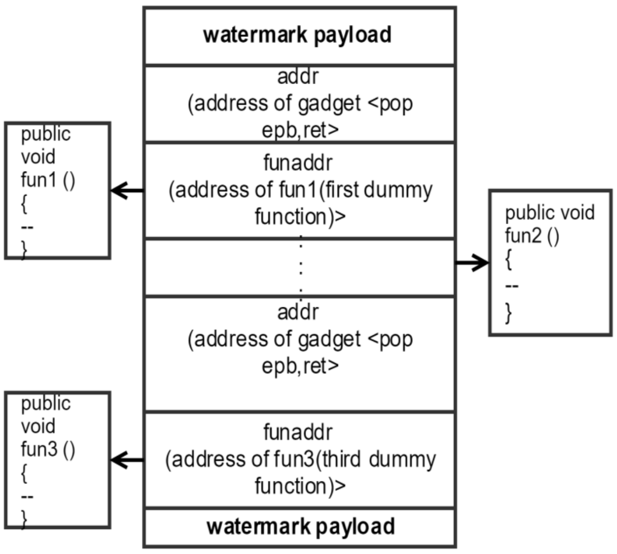

watermark payload
addr
(address of gadget <pop epb,ret>
public void fun1 () { -- }
funaddr
(address of fun1(first dummy function)>
public void fun2 () { -- }
addr
(address of gadget <pop epb,ret>
public void fun3 () { -- }
funaddr
(address of fun3(third dummy function)>
watermark payload

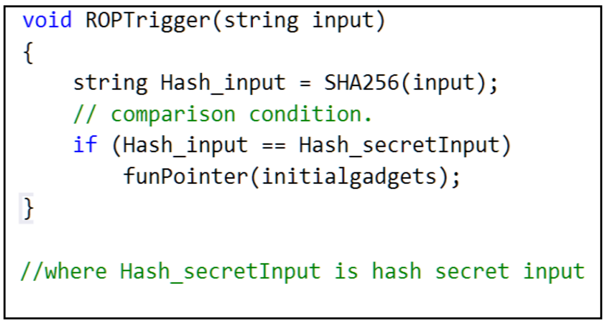
Figure 3: Watermark payload for ROP based watermark.

3-Payload Chain:
The proposed payload contains all the selected gadgets address, all the selected gadgets are in order and the payload chain and execute all the gadget based on their order. Also, because the ROP gadgets are special instructions end with return “ret”; the first gadgets of payload executed then automatically return the control to the second gadget and so on.

4- Triggering ROP via function pointer overwriting:
The final step of building ROP-based watermarking is changed the normal execution path of the program to ROP execution path to execute the hidden watermark. the ROP execution path executes the selected gadgets because of that the code must appear as a non-suspicious code. Thus, for triggering the ROP execution, a function pointer has been used. Function pointer provides a simple way of transferring the program control to the ROP gadgets by overwriting of its value.

Normally in any watermark, there is a trigger function that calls dummy function and performs some calculation that does not affect the program execution, the trigger calls transfer the normal execution of the program to the part that responsible for watermark extraction. And to trigger the trigger function done by inputting secret input. In the design the trigger function is executed by inputting the secret input. The ROP trigger calls dummy function to check if the secret input is the needed input to extract the watermark. After that, check if the secret input is the valid input then the control transfer to the payload and precisely to the first gadget. The first gadget is responsible for saving the current environment. The most important part of building the trigger function is to encode condition to test the secret input in most powerful and secure way. It is a fact the using simple block of comparisons such as if and else statement to compare between the entered input and constant secret input in the program is the simplest solution. However this solution could attract attacker attention when analyzing the program [4]. Thus, building a complex trigger function is harder to discover or analyzes by program analysis. Here a novel idea is proposed to build trigger function, see Figure 4. The obfuscation mechanism has been used to compare between the entered input and the secret input called cryptographic hash function in [17] used to obfuscation conditional code, The cryptographic hash function designed to be impossible to reverse and it has been proven that it is infeasible to reverse such function and to know the whole complete set of operations. Specifically, the SHA256 (Secure Hash Algorithm) has been used which is one of cryptographic hash functions, it is one way function, and it takes the message with any length and encrypted to unique fixed size hash of 256 bit (23 bytes) and cannot be decrypted back [18], as an example:

The message is: ‘abc.'
The encrypted SHA256 is:
‘ba7816bf8f01cfea414140de5dae2223b00361a396177a9cb41 0ff61f20015ad’
As a result, the ROP trigger can determine the secret input using SHA256 hash function to extract the watermark.



```
void ROPTrigger(string input)
{
    string Hash_input = SHA256(input);
    // comparison condition.
    if (Hash_input == Hash_secretInput)
        funPointer(initialgadgets);
}

//where Hash_secretInput is hash secret input
```

Figure 4: ROP trigger condition.

4. DESIGN AND IMPLEMENTATION

In this section, this research focuses on describing the design and implementation of the proposed work. The building block for the proposed work shown in Figure 5, to generate watermark software first the program must be re-write program code and perform all the steps mentioned above to add ROP watermark in the program.

The scheme applied on one benchmark called Sjeng from the SPECint-2006 test suite which has been used in a lot of previous researches [19][4]. Sjeng program is a chess game written in C programming language. Given a set of conditions the program attempts to find the best move in a game[20].

The Sjeng program selected as the source program and re-write the program to converted to watermark program. Thus, the Sjeng program has been modified and added the ROP

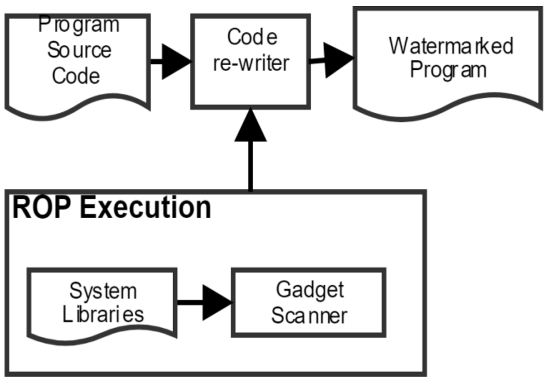


Figure 5: Building block of the proposed work.

code and the proposed watermark, the ROP code transfer the control to the gadgets when the user entered the secreted input to extract the hidden watermark message then return back to the program normal execution routine. Moreover, during the program regular use, the extra code of ROP is always executed because ROP does not change the original program execution. Therefore, the code would be less suspicious because it does not keep idle in the program.

The proposed ROP based watermarking implemented using a combination of C++ and C programming language. Thus, the Microsoft Visual Studio 2015 has been picked to implement the proposed work. It provides a lot of features such as creating software regardless of the software size and complexity, ability to set environment only once, fixing error easily and flexibility to analyze code quality and performance [21]. Figure 6 shows the pseudocode of playing Sjeng program, to start playing chess the user enter move and the Sjeng program checks if the entered move is legal to move and then the program play his move by his attempts to find the best move and so on. In each user move the program check the move possibility of being the secret input to extract watermark and based on this the program may take one of two possible paths:

1- The first path converts the secret input using hash function SHA256 and compares it with the proposed hidden method. When the move recognized as the secret watermark then the program trigger the ROP. After that, the program changes the normal execution path to the ROP execution path to extract the watermark. The ROP execution path executes all the gadgets in order and calls the three dummy functions. Also embed the watermark into the string S then extracted and display it on the screen to the user.
2- The second path, when the move recognized as regular chess move and not the secret watermark input, then the program continue the original execution path of the program without trigger the ROP.

```
START PROGRAM
WHILE IsFinalMove()
 Print "Enter your Sjeng Move "
 String input =READ(User Move)
 IF Hash(User Input) == Hash(Secret Input )
   Execute ROP( )
   Print "  Watermark "
 EndIF
 ELSE
   Sjeng_ Move Calculate()
   Print "Sjeng Move"
 ENDELSE
ENDWHILE
END PROGRAM
```

Figure 6: Sjeng program pseudocode.

5. RESULT AND DISCUSSION

In this section, evaluate the proposed software watermark using ROP. The experimental watermarking is implemented to display the watermark message on the screen. All tests were run on a laptop with a 2.60GHz Intel Core i7-6500U CPU, 8GB memory, and Windows 10 operating system. In order to gain knowledge about its practical behavior, it has to evaluate it under different assessment criteria [22]. Some of the most important criteria are correctness, robustness, stealth, Performance overhead, resistance, and credibility. Correctness is keeping the program behavior similar before and after watermark is embedded. Robustness means the watermarked program should resist attacks. Stealth means the program copy has the complexity to locate an embedded watermark. Performance overhead means the watermark program and the original program should be in same running time, memory consumption and performance [23]. Resilience measures the resistance against specific attempts at discovery or deletion. Credibility defines how accurately the watermark can be retrieved [4]. The following points are the evaluation of the proposed work under different criteria.

1- Credibility and Correctness:

The proposed software watermark based on ROP takes string “input” as input to extract the watermark message. The watermark message, in this case, is proving of executing the embedded ROP gadgets successfully. Thus, its credibility is obvious.

It is believed that the correctness of our ROP-based watermark is self- evident, and the behavior of the program still the same after embedding the watermark.

2- Stealth:

The stealth of the proposed work proved by using the dependence analysis on watermarking components to measure the stealth of the software watermark. IDA pro 5.0 tool is used to simulate a dependency analysis.

IDA pro is disassembler and debugger tool provide a lot of features to the programmer to facilitate the understanding of the program and analysis it. IDA pro support many platforms such as Windows and Linux. There are two types of dependency analysis can be generated for function and global variable: first, cross-references to a symbol (Xrefs To) display a graph with all functions and global variables that called the selected symbol. Second, cross-references from a symbol (Xrefs from) display a graph with all functions and global variables that can be called from the selected symbol [24]. For simplicity, the function that makes a comparison between the user input and watermark secret input name “ROP function” and if the condition is true, it will trigger the function name “trigger” to extract the watermark message otherwise the program will continue run without extraction of the watermark. The watermark message is inside string name “S.” The string will contain the watermark message only when the trigger function is called. Thus, the string will hold the watermark massage in the trigger function. Let assume an attacker finds the distinguishable function “ROP” and uses the IDA pro to find the dependence analysis of this function. Figure 7 shows the dependence analysis of “ROP function,” where we can see that “ROP function” called from function name “is_move” and that gives the attacker a clear path where can search for any function to find watermark code.

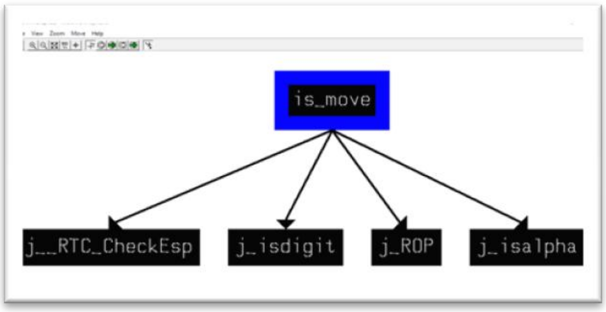


Figure 7: the dependence analysis of “ROP function.”

But the proposed watermark is hidden in the data region of the program. For clarification, If the attacker wants to find all the dependences of string “S” the graph only display “ROP” function that means function “ROP” is the only function that responsible for modifying and change the string “S,” Figure 8: shows the dependence analysis of “S” string.

This is a prove that the watermark semantics is invisible to code analysis and the attacker can not find the string that holds the watermark message even after using IDA pro to find the dependency analysis.

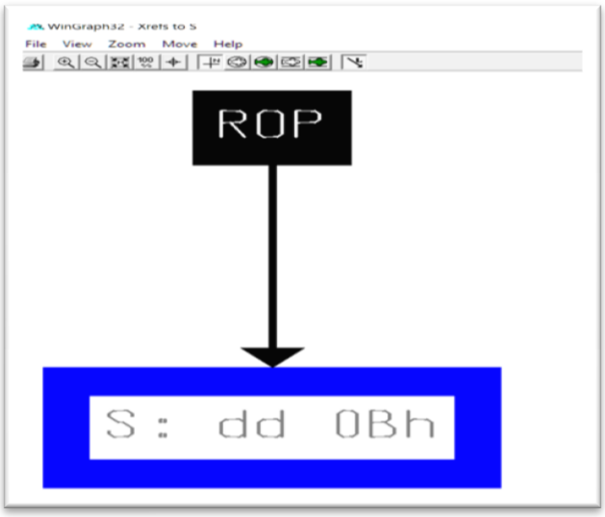


Figure 8: the dependence analysis of “S” string.

3- Resilience:

The method determines a strong resilience by test the strength of the design against distortive attacks. The distortive attack is the attacker ability to destroy the watermark code by changing the binary code of the watermark. The distortive attack can apply semantic-preserving transformations or binary obfuscation to the watermark program to change or destroy the watermark such as binary obfuscation and packing and optimizing [2]. The method tested against a selection of well-known transformation tools:

- UPX: stands for ultimate packer for executables, it provides high quality compression and fast decompression with no memory overhead [25].

- Stunnix CXX-Obfus: it performs professional obfuscation to the code to make it difficult to understand, it is very advanced cross platforms such as windows and Mac OS X, it supports many languages such as java and C/C++ [26]. In the test, the Stunnix C/C++ Obfuscator has been used.

The watermak program applied to upx and compared the watermark extraction between the orignal program and the program. After using upx, the results showed the watermark program worked successfully with compression and decompression. Thus, the results showd that after applied the above transformation tools in the propsed method the program status remain the same, and it still takes the secret input and chain the watermark and extract the watermark message correctly, this indicates that the new design has good resilience.

4- Performance Overhead:
The performance overhead evaluated on three aspects: the runtime overhead, code size increment, and the required additional heap space. Therefore, the design evaluates in comparison with the original ROP-based watermark in [4], the performances analyze of the two programs using Microsoft Visual Studio Enterprise 2015, and precisely Visual Studio Profiling Tools has been used.

The runtime overhead is the time form the program start until generates the watermark; Table 2 shows a summary of the runtime overhead and additional heap space required, Figure 9 shows the program size increment. The results showed that the runtime overhead in ms is significantly smaller than that in the original ROP- based watermark, also the proposed design has a smaller increase in the program size. The size of additional heap space in the proposed design takes only 164 bytes when constructed the gadget while the original watermark takes between 156 and 188 bytes depending on gadgets format.

Table 2: summary of the runtime overhead and additional heap space required.

Sjeng benchmark	Our ROP-based watermarking	Original ROP-based watermarking
Runtime Overhead (ms)	18	19.4
Additional heap space required	164	Between 156 or 188

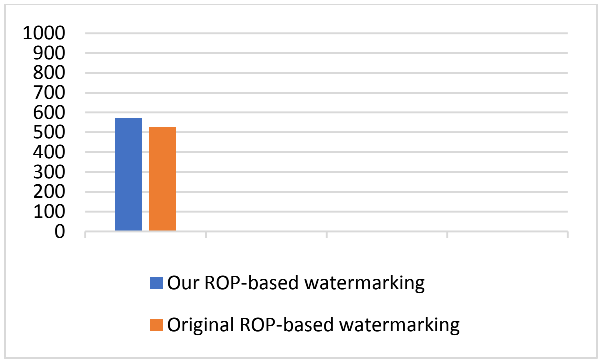


Figure 9: Increment in program size (bytes).

Heap sizes are analyzed and checked with the different gadgets where formats of the random gadgets related to Sjeng benchmark is considered as in Figure 10. This analysis provides us the future computer security which could be improved not only with the type of gadgets but also the formation of the gadgets.

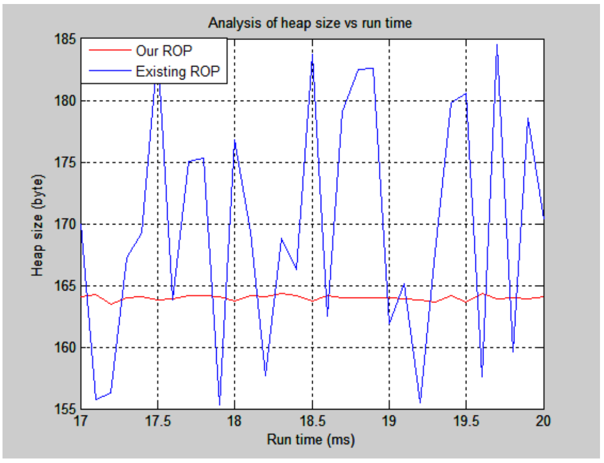


Figure 10: Heap sizes of gadgets related to Sjeng with different formats.

6. CONCLUSION

The new software watermark designed using ROP with complex trigger ROP function. In the trigger function, the hash function SHA256 used to compare between the secret input and user entered input. The proposed work evaluated with many existing criteria also compare the proposed work with an existing and only technique that used ROP to embed a watermark into software. The result showed that the proposed work hard for an attacker to understand in the program analysis and it has better resilience, Stealth and minimum runtime overhead.

The future work of software watermark using ROP is to improve it by imbedding hard watermark into the program such as adding watermark in the object of class instead of a simple string. Also, improve the efficiency of software watermark by distributing ROP payload among the program to make it harder to recognize to the attacker.

7. ACKNOWLEDGMENTS

We are glad that we completed this work successfully. This work would be impossible without the help of Dr. Vijey Thayananthan. I would like to thank him for his expert advice and usual support.

8. REFERENCES

[1] L. Chen and C. Zhang, “A Novel Algorithm for.NET Programs Watermarking based on Obfuscation” in Int Symposium on Instrumentation & Measurement, Sensor Network and Automation, (IMSN). Sanya .2012 , pp. 583 - 586.

[2] Collberg, C. and C. Thomborson (1999). Software watermarking: Models and dynamic embeddings. Proceedings of the 26th ACM SIGPLAN-SIGACT symposium on Principles of programming languages, ACM.

[3] C. Collberg and J. Nagra. Surreptitious Software — Obfuscation, Watermarking, and Tamperproofing for Software Protection. Software Security Series. Addison-Wesley, 2009

[4] H. Ma, K.Lu , X.Ma ,H. Zhang,C Jia and D.Gao." Software Watermarking using Return-Oriented Programming ".on ACM Symposium on Information, Computer and Communications Security. ASIACCS.2015.pp. 369-380.Tavel, P. 2007 Modeling and Simulation Design. AK Peters Ltd.

[5] Anley, C., et al. (2011). The shellcoder's handbook: discovering and exploiting security holes, John Wiley & Sons.

[6] M. Shirali-Shahreza and S. Shirali-Shahreza . "Software Watermarking by Equation Reordering ".on 3rd Int Conf on Information and Communication Technologies: From Theory to Applications, ICTTA .2008.pp. 1 – 4.

[7] S. Zonglu, J.Hua and X.Aicheng. " Software Watermarking Algorithm by Coefficients of Equation ".on 3rd Int Conf on Genetic and Evolutionary Computing, 2009.pp. 410 - 413.

[8] J. Hua, H. Hanlei and W.Xin . "Software Watermark Algorithm Based on Chinese Remainder Theorem".on IEEE Conf Anthology, 2013.pp. 602 – 606.

[9] Z.Jian-qi, L.Yan-heng, and Y.Ke ." A Robust Dynamic Watermarking Scheme based on STBDW ". WRI World Congress on Computer Science and Information Engineering, 2009.pp. 602 – 606.

[10] G.Gupta and J.Pieprzyk." Source Code Watermarking Based on Function Dependency-Oriented Sequencing".on Int Conf on Intelligent Information Hiding and Multimedia Signal Processing, IIHMSP.Harbin.2008.pp. 965 – 968.

[11] Zhang, X., et al. (2008). Hash function based software watermarking. Advanced Software Engineering and Its Applications, 2008. ASEA 2008, IEEE.

[12] Scut (2001). Exploiting format string vulnerabilities, Team Teso.

[13] Shacham, H. (2007). "The Geometry of Innocent Flesh on the Bone:Return-into-libc without Function Calls (on the x86)." In Proceedings of the 14th ACM conference onComputer and communications security (CCS): 552-561.

[14] Buchanan, E., et al. (2008). When good instructions go bad: Generalizing return-oriented programming to RISC. Proceedings of the 15th ACM conference on Computer and communications security, ACM.

[15] Immunity, I. Immunity debugger.

[16] Corelan (2015). Mona.

[17] Sharif, M. I., et al. (2008). Impeding Malware Analysis Using Conditional Code Obfuscation. NDSS.

[18] Standard, S. H. (2002). "FIPS PUB 180-2." National Institute of Standards and Technology.

[19] Palsberg, J., et al. (2000). Experience with software watermarking. Computer Security Applications, 2000. ACSAC'00. 16th Annual Conference, IEEE.

[20] Coffey, P. (2011). Benchmarking the amazon elastic compute cloud (ec2), Worcester Polytechnic Institute.

[21] Microsoft (2015). Visual Studio.

[22] Collberg, C., et al. (2003). Error-correcting graphs for software watermarking. Proceedings of the 29th Workshop on Graph Theoretic Concepts in Computer Science, Springer.

[23] Tang, Z. and D. Fang (2011). A tamper-proof software watermark using code encryption. Intelligence and Security Informatics (ISI), 2011 IEEE International Conference on, IEEE.

[24] Eagle, C. (2011). The IDA pro book: the unofficial guide to the world's most popular disassembler, No Starch Press.

[25] Oberhumer, M., et al. (2004). UPX: the Ultimate Packer for eXecutables.

[26] Stunnix Stunnix C/C++ Obfuscator.