Obfuscation with Turing Machine

Yan Wang[1], Dinghao Wu,[2] Shuai Wang, and Pei Wang

Pennsylvania State University, State College,PA 16801, USA,
`ybw5084@ist.psu.edu`,
WWW home page: `http://users/~iekeland/web/welcome.html`

Abstract. Software security is a fundamental research domain in this threat emerging technology world. Leveraging system vulnerbilities is one of the common ways to hack into computer system. Hackers have to understand the program control flow in order to figure out the hacking logic and technique. In this way, consealing important branch trigger condition logics are crucial for protecting softwares from being hacked. In this paper we propose a novel control flow obfuscation method with Turing machine. By entwining the original software programs with Turing machine executio, software control flow graphs could be significantly obfuscated which means it is much more harder or even impossible for hackers to embed malicious execution codes into Turing machine obfuscated programs.We implemented a obfuscation tool to gernerate obfuscated binaries using LLVM. Evaluation results demonstrate that software programs become much more complicated after Turing maching obfuscation. At current stage, we only obfuscated the interger operand instructions which are vital to the whole program.

Key words: software security,control flow obfuscation, Turing machine,LLVM

1 Introduction

Obfuscation derives from intellectual property protection. Internet brings us unprecedent convience and threat of idea plagiarism and copyright infringement at the same time. Consealing the algorithm of a software means a lot for the society especially for high-tech industry. Reverse engineering is often used by to recover source code from binaries to analyze software vulnerbilities or to steal software ideas or algorithms. Obfuscation is a technique to block or harden the process of reverse engineering.

Recently, Software security has become a bigger and bigger concern in research community because of infamous ransomware attack and severe vulnerbiliy such as the recent "Wanncry" insidence and the openssl heartbleeding bug.This incidences challenge the computer world greatly. Hackers endevors to figure out vulnerbilities inside a software program. Knowing the software architecture and program logic like control flow graph is an indispensible prerequisite for hackers to analyze the original codes. With the help of some monitoring techniques hackers could even iterate all possible paths to try to restore all important branch information along the execution paths.*Concolic testing* is an example which expoits symbolic execution given a certain input while it keeps changing input data until the code coverage proceed a threshhold[1] as shown in Figure 1. It has been proved to work in restoring the branch information in the original source codes. Hence, a lot of research focus on preventing bad guys from figuring out the essential logic. Consealing important "crossroad" in a source program. Control flow obfuscation is one of these techniques. Control flow obfuscation aims at hiding conditional transfers and complicating the execution path within a source program. Through replacing or insering extra contrlo flow graph edges, the original software logic become harder or even impossible to trace. Previous research[2] have prove the effectiveness of contral flow obfuscation.

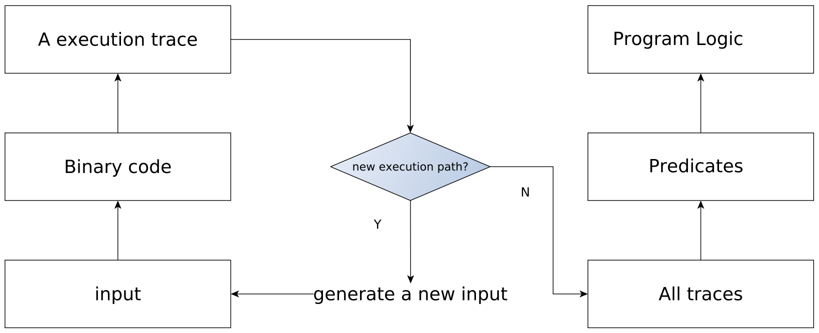

Fig. 1. Reverse engineering with concolic testing

In this paper,we propose a novel control flow obfuscation method with Turing machine embedded in the original program source codes. Turing machine is the essence of computaion so it could calculate accurately all kinds of computer operation. In this way, important branch conditions in original source code could be replaced by a Turing machine execution which yields the very identical conditional value. Besides, a Turing machine behaves like a state machine so it contains a lot of branch condition checks in Turing machine transition table. In this whole process, sensitive branch information could be successfully hided in Turing machine call graphs. In the mean time, Turing machine execution greatly introduce computational complexity to the target program so it is almost impossible to read the obfuscated programs to do reverse engineering.

Currently, we are more interested in integer operation condition instructions. Utilizing LLVM, we could turn original program source codes into intermediate representation(IR).Turing machine obfuscator take over from IR and select interested instruction candidates which would be redirected to semantic equivalent Turing machine execution. After all paths finished in the Turing machine "blackbox", the original call graph resumes from the the obfuscator interrupt and the whole program control flow graph is greatly expanded and complicated. Since LLVM is the implementation foundation, currently our proposed obfuscator could only be applied to C/C++ source programs. Inspired by previous previous works[4],we evaluate our obfuscator from four demensions which are potency,resilence, cost and stealth respectively. Results indicates that Turing machin obfuscator could effectively obfuscate target source codes with acceptable cost overhead.

This paper is organized as follows. Section 2 discusses related works on obfuscation, especially control flow obfuscation. Section 3 illustrate the idea and archetecture of Turing machine obfuscator. Obfuscator implementation is discussed in section 4. Section 5 discusses the evaluation result of our proposed obfuscator. Finally, we conclude thsi paper in section 6.

2 Related Works

Generally speaking, reverse engineering is devided into static analysis such and dynamic analysis. To compete static reverse engineering, researchers usually focus on hardening disassembling and decompiling process. To compete the dynamic reverse engineering techniques such concolic testing, conditional transfer logic must be hidden from adversaries. Control flow obfuscation has been proved effective in previous works.

In [5], the authors identified conditions that could trigger malware execution then using hash function to transform the values which could launch malware. Afterwards, correspondent conditional codes which would be ran with satisfied value were encryped with a key generated based on the instruction trigger value. By this means the obfuscation analyzer could never get a chance to get the expected "launching code" consequently planted malware could never be executed. This technology works on certain fixed trigger value but not in senarios that trigger values are intervals such as $>$ and $<$. This limitation narrows the applicable conditions greatly since a large volume of branch conditions are comparsion operations. In addition, the encrytion and decryption process in this methodology also introduce inneglectable overhead.

In [6], the authors used signals("traps") to replace the control transfers unconditional instructions like "jmp" and "call" in order to confuse disassembly operation which is the first step of reverse engineering. Dummy control transfers and junk instructions were also inserted after signal replacements. This method seems to be effective in fooling disassemblers but it can't be applied in senarios that conditional instruction logic needs to be protected from being figured out.

In [7], the authors endevors to conseal branch information leveraging a remote trusted third party environment. The idea seemed to work while it not only introduced a great network overhead, but also rely on trusted network accessiblity which can't be guranteed in reality. This drwaback means this idea can't be applied in common obfuscation cases.

In [2], the authors took advantage of neural network to replace appropriate conditional instructions in source programs in order to achieve the goal of consealing conditional instruction logic thus dynamic analysis like concolic testing could never dig useful branch information. Although the idea looks promising and the results indicates the effectiveness of this methodology to some degree, fundamentally we don't believe neural networks solution is suitable for such senarios. To the best of our knowledge, neural networks works like a blackbox. It lacks the rigorous mathmatical proof to illustrate a correct result must be generated given a input. Neural networks not only introduced complexity but also unpredictability to the original source programs. Trained models may behave very differently if given initial parameters with only some nuances, which means that it is also very hard to train a accurate enough model to simulate the conditional instruction. In addition, we noticed that neural networks consume too much memory in the evaluation experiments.

3 Turing machine obfuscation overview

We assume that H is (A_∞, B_∞)-subquadratic at infinity, for some constant symmetric matrices A_∞ and B_∞, with $B_\infty - A_\infty$ positive definite. Set:

$$\gamma := \text{smallest eigenvalue of} \quad B_\infty - A_\infty \tag{1}$$

$$\lambda := \text{largest negative eigenvalue of} \quad J\frac{d}{dt} + A_\infty \ . \tag{2}$$

Theorem 1 tells us that if $\lambda + \gamma < 0$, the boundary-value problem:

$$\begin{aligned} \dot{x} &= JH'(x) \\ x(0) &= x(T) \end{aligned} \tag{3}$$

has at least one solution $\overline{x}$, which is found by minimizing the dual action functional:

$$\psi(u) = \int_o^T \left[\frac{1}{2}\left(\Lambda_o^{-1}u, u\right) + N^*(-u)\right] dt \tag{4}$$

on the range of Λ, which is a subspace $R(\Lambda)_L^2$ with finite codimension. Here

$$N(x) := H(x) - \frac{1}{2}\left(A_\infty x, x\right) \tag{5}$$

is a convex function, and

$$N(x) \le \frac{1}{2}\left(\left(B_\infty - A_\infty\right)x, x\right) + c \quad \forall x \ . \tag{6}$$

Proposition 1. *Assume $H'(0) = 0$ and $H(0) = 0$. Set:*

$$\delta := \liminf_{x \to 0} 2N(x)\left\|x\right\|^{-2} \ . \tag{7}$$

If $\gamma < -\lambda < \delta$, the solution $\overline{u}$ is non-zero:

$$\overline{x}(t) \ne 0 \quad \forall t \ . \tag{8}$$

Proof. Condition (7) means that, for every $\delta' > \delta$, there is some $\varepsilon > 0$ such that

$$\|x\| \le \varepsilon \Rightarrow N(x) \le \frac{\delta'}{2} \|x\|^2 \ . \tag{9}$$

It is an exercise in convex analysis, into which we shall not go, to show that this implies that there is an $\eta > 0$ such that

$$f \|x\| \le \eta \Rightarrow N^*(y) \le \frac{1}{2\delta'} \|y\|^2 \ . \tag{10}$$

Fig. 2. This is the caption of the figure displaying a white eagle and a white horse on a snow field

Since u_1 is a smooth function, we will have $\|hu_1\|_\infty \le \eta$ for h small enough, and inequality (10) will hold, yielding thereby:

$$\psi(hu_1) \le \frac{h^2}{2} \frac{1}{\lambda} \|u_1\|_2^2 + \frac{h^2}{2} \frac{1}{\delta'} \|u_1\|^2 \ . \tag{11}$$

If we choose δ' close enough to δ, the quantity $\left(\frac{1}{\lambda} + \frac{1}{\delta'}\right)$ will be negative, and we end up with

$$\psi(hu_1) < 0 \qquad \text{for} \ \ h \ne 0 \ \ \text{small} \ . \tag{12}$$

On the other hand, we check directly that $\psi(0) = 0$. This shows that 0 cannot be a minimizer of ψ, not even a local one. So $\overline{u} \ne 0$ and $\overline{u} \ne \Lambda_o^{-1}(0) = 0$. □

Corollary 1. *Assume H is C^2 and (a_∞, b_∞)-subquadratic at infinity. Let ξ_1, $\dots, \xi_N$ be the equilibria, that is, the solutions of $H'(\xi) = 0$. Denote by ω_k the smallest eigenvalue of $H''(\xi_k)$, and set:*

$$\omega := \mathrm{Min}\,\{\omega_1, \dots, \omega_k\} \ . \tag{13}$$

If:

$$\frac{T}{2\pi} b_\infty < -E\left[-\frac{T}{2\pi} a_\infty\right] < \frac{T}{2\pi}\omega \tag{14}$$

then minimization of ψ yields a non-constant T-periodic solution $\overline{x}$.

We recall once more that by the integer part $E[\alpha]$ of $\alpha \in \mathbb{R}$, we mean the $a \in \mathbb{Z}$ such that $a < \alpha \le a + 1$. For instance, if we take $a_\infty = 0$, Corollary 2 tells us that $\overline{x}$ exists and is non-constant provided that:

$$\frac{T}{2\pi}b_\infty < 1 < \frac{T}{2\pi} \tag{15}$$

or

$$T \in \left(\frac{2\pi}{\omega}, \frac{2\pi}{b_\infty}\right) \ . \tag{16}$$

Proof. The spectrum of Λ is $\frac{2\pi}{T}\mathbb{Z} + a_\infty$. The largest negative eigenvalue λ is given by $\frac{2\pi}{T}k_o + a_\infty$, where

$$\frac{2\pi}{T}k_o + a_\infty < 0 \le \frac{2\pi}{T}(k_o+1) + a_\infty \ . \tag{17}$$

Hence:

$$k_o = E\left[-\frac{T}{2\pi}a_\infty\right] \ . \tag{18}$$

The condition $\gamma < -\lambda < \delta$ now becomes:

$$b_\infty - a_\infty < -\frac{2\pi}{T}k_o - a_\infty < \omega - a_\infty \tag{19}$$

which is precisely condition (14). $\square$

Lemma 1. *Assume that H is C^2 on $\mathbb{R}^{2n}\backslash\{0\}$ and that $H''(x)$ is non-degenerate for any $x \ne 0$. Then any local minimizer $\widetilde{x}$ of ψ has minimal period T.*

Proof. We know that $\widetilde{x}$, or $\widetilde{x} + \xi$ for some constant $\xi \in \mathbb{R}^{2n}$, is a T-periodic solution of the Hamiltonian system:

$$\dot{x} = JH'(x) \ . \tag{20}$$

There is no loss of generality in taking $\xi = 0$. So $\psi(x) \ge \psi(\widetilde{x})$ for all $\widetilde{x}$ in some neighbourhood of x in $W^{1,2}\left(\mathbb{R}/T\mathbb{Z};\mathbb{R}^{2n}\right)$.

But this index is precisely the index $i_T(\widetilde{x})$ of the T-periodic solution $\widetilde{x}$ over the interval $(0,T)$, as defined in Sect. 2.6. So

$$i_T(\widetilde{x}) = 0 \ . \tag{21}$$

Now if $\widetilde{x}$ has a lower period, T/k say, we would have, by Corollary 31:

$$i_T(\widetilde{x}) = i_{kT/k}(\widetilde{x}) \ge k i_{T/k}(\widetilde{x}) + k - 1 \ge k - 1 \ge 1 \ . \tag{22}$$

This would contradict (21), and thus cannot happen. $\square$

Table 1. This is the example table taken out of *The TEXbook,* p. 246

Year	World population
8000 B.C.	5,000,000
50 A.D.	200,000,000
1650 A.D.	500,000,000
1945 A.D.	2,300,000,000
1980 A.D.	4,400,000,000

Notes and Comments. The results in this section are a refined version of [8]; the minimality result of Proposition 14 was the first of its kind.

To understand the nontriviality conditions, such as the one in formula (16), one may think of a one-parameter family x_T, $T \in \left(2\pi\omega^{-1}, 2\pi b_\infty^{-1}\right)$ of periodic solutions, $x_T(0) = x_T(T)$, with x_T going away to infinity when $T \to 2\pi\omega^{-1}$, which is the period of the linearized system at 0.

Theorem 1 (Ghoussoub-Preiss). *Assume $H(t,x)$ is $(0,\varepsilon)$-subquadratic at infinity for all $\varepsilon > 0$, and T-periodic in t*

$$H(t,\cdot) \qquad \text{is convex} \quad \forall t \tag{23}$$

$$H(\cdot,x) \qquad \text{is} \quad T{-}\text{periodic} \quad \forall x \tag{24}$$

$$H(t,x) \ge n\left(\|x\|\right) \qquad \text{with} \quad n(s)s^{-1} \to \infty \quad \text{as} \quad s \to \infty \tag{25}$$

$$\forall \varepsilon > 0 \ , \quad \exists c \ : \ H(t,x) \le \frac{\varepsilon}{2}\left\|x\right\|^2 + c \ . \tag{26}$$

Assume also that H is C^2, and $H''(t,x)$ is positive definite everywhere. Then there is a sequence x_k, $k \in \mathbb{N}$, of kT-periodic solutions of the system

$$\dot{x} = JH'(t,x) \tag{27}$$

such that, for every $k \in \mathbb{N}$, there is some $p_o \in \mathbb{N}$ with:

$$p \ge p_o \Rightarrow x_{pk} \ne x_k \ . \tag{28}$$

$\square$

*Example 1 (*External forcing*).* Consider the system:

$$\dot{x} = JH'(x) + f(t) \tag{29}$$

where the Hamiltonian H is $(0,b_\infty)$-subquadratic, and the forcing term is a distribution on the circle:

$$f = \frac{d}{dt}F + f_o \qquad \text{with} \quad F \in L^2\left(\mathbb{R}/T\mathbb{Z};\mathbb{R}^{2n}\right) \ , \tag{30}$$

where $f_o := T^{-1}\int_o^T f(t)dt$. For instance,

$$f(t) = \sum_{k \in \mathbb{N}} \delta_k \xi \ , \tag{31}$$

where δ_k is the Dirac mass at $t = k$ and $\xi \in \mathbb{R}^{2n}$ is a constant, fits the prescription. This means that the system $\dot{x} = JH'(x)$ is being excited by a series of identical shocks at interval T.

Definition 1. *Let $A_\infty(t)$ and $B_\infty(t)$ be symmetric operators in $\mathbb{R}^{2n}$, depending continuously on $t \in [0,T]$, such that $A_\infty(t) \leq B_\infty(t)$ for all t.*

A Borelian function $H : [0,T] \times \mathbb{R}^{2n} \to \mathbb{R}$ is called (A_∞, B_∞)-subquadratic at infinity if there exists a function $N(t,x)$ such that:

$$H(t,x) = \frac{1}{2}\left(A_\infty(t)x, x\right) + N(t,x) \tag{32}$$

$$\forall t \ , \quad N(t,x) \quad \text{is convex with respect to} \quad x \tag{33}$$

$$N(t,x) \geq n\left(\|x\|\right) \qquad \text{with} \quad n(s)s^{-1} \to +\infty \quad \text{as} \quad s \to +\infty \tag{34}$$

$$\exists c \in \mathbb{R} \ : \quad H(t,x) \leq \frac{1}{2}\left(B_\infty(t)x, x\right) + c \quad \forall x \ . \tag{35}$$

If $A_\infty(t) = a_\infty I$ and $B_\infty(t) = b_\infty I$, with $a_\infty \leq b_\infty \in \mathbb{R}$, we shall say that H is (a_∞, b_∞)-subquadratic at infinity. As an example, the function $\|x\|^\alpha$, with $1 \leq \alpha < 2$, is $(0, \varepsilon)$-subquadratic at infinity for every $\varepsilon > 0$. Similarly, the Hamiltonian

$$H(t,x) = \frac{1}{2}k\left\|k\right\|^2 + \left\|x\right\|^\alpha \tag{36}$$

is $(k, k+\varepsilon)$-subquadratic for every $\varepsilon > 0$. Note that, if $k < 0$, it is not convex.

Notes and Comments. The first results on subharmonics were obtained by Foster and Kesselman in [10], who showed the existence of infinitely many subharmonics both in the subquadratic and superquadratic case, with suitable growth conditions on H'. Again the duality approach enabled Foster and Waterman in [12] to treat the same problem in the convex-subquadratic case, with growth conditions on H only.

Recently, Smith and Waterman (see [8] and May et al. [9]) have obtained lower bound on the number of subharmonics of period kT, based on symmetry considerations and on pinching estimates, as in Sect. 5.2 of this article.

References

1. Sen, Koushik, Darko Marinov, and Gul Agha. "CUTE: a concolic unit testing engine for C." ACM SIGSOFT Software Engineering Notes. Vol. 30. No. 5. ACM, 2005.
2. Ma, Haoyu, et al. "Control flow obfuscation using neural network to fight concolic testing." International Conference on Security and Privacy in Communication Systems. Springer International Publishing, 2014.
3. Wang, Pei, et al. "Translingual obfuscation." Security and Privacy (EuroS&P), 2016 IEEE European Symposium on. IEEE, 2016.

4. Collberg, Christian, Clark Thomborson, and Douglas Low. ”Manufacturing cheap, resilient, and stealthy opaque constructs.” Proceedings of the 25th ACM SIGPLAN-SIGACT symposium on Principles of programming languages. ACM, 1998.
5. Sharif, Monirul I., et al. ”Impeding Malware Analysis Using Conditional Code Obfuscation.” NDSS. 2008.
6. Popov, Igor V., Saumya K. Debray, and Gregory R. Andrews. ”Binary Obfuscation Using Signals.” Usenix Security. 2007.
7. Wang, Zhi, et al. ”Branch obfuscation using code mobility and signal.” Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual. IEEE, 2012.
8. Smith, T.F., Waterman, M.S.: Identification of Common Molecular Subsequences. J. Mol. Biol. 147, 195–197 (1981)
9. May, P., Ehrlich, H.C., Steinke, T.: ZIB Structure Prediction Pipeline: Composing a Complex Biological Workflow through Web Services. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 1148–1158. Springer, Heidelberg (2006)
10. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1999)
11. Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid Information Services for Distributed Resource Sharing. In: 10th IEEE International Symposium on High Performance Distributed Computing, pp. 181–184. IEEE Press, New York (2001)
12. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The Physiology of the Grid: an Open Grid Services Architecture for Distributed Systems Integration. Technical report, Global Grid Forum (2002)
13. National Center for Biotechnology Information, http://www.ncbi.nlm.nih.gov