Enhance virtual-machine-based code obfuscation security through dynamic bytecode scheduling

Kaiyuan Kuang a, Zhanyong Tang a,*, Xiaoqing Gong a, Dingyi Fang a, Xiaojiang Chen a, Zheng Wang b

a School of Information Science and Technology, Northwest University, China
b MetaLab, School of Computing and Communications, Lancaster University, UK

Article history:
Received 29 March 2017
Received in revised form 1 September 2017
Accepted 6 January 2018
Available online

ABSTRACT

Code virtualization built upon virtual machine (VM) technologies is emerging as a viable method for implementing code obfuscation to protect programs against unauthorized analysis. State-of-the-art VM-based protection approaches use a fixed scheduling structure where the program always follows a single, deterministic execution path for the same input. Such approaches, however, are vulnerable in certain scenarios where the attacker can reuse knowledge extracted from previously seen software to crack applications protected with the same obfuscation scheme. This paper presents DSVMP, a novel VM-based code obfuscation approach for software protection. DSVMP brings together two techniques to provide stronger code protection than prior VM-based approaches. Firstly, it uses a dynamic instruction scheduler to randomly direct the program to execute different paths without violating the correctness across different runs. By randomly choosing the program execution path, the application exposes diverse behavior, making it much more difficult for an attacker to reuse the knowledge collected from previous runs or similar applications to launch an attack. Secondly, it employs multiple VMs to further obfuscate the mapping from VM opcode to native machine instructions, so that the same opcode could be mapped to different native instructions at runtime, making code analysis even harder. We have implemented DSVMP in a prototype system and evaluated it using a set of widely used applications. Experimental results show that DSVMP provides stronger protection with comparable runtime overhead and code size, when it is compared to two commercial VM-based code obfuscation tools.

© 2018 The Author(s). Published by Elsevier Ltd. This is an open access article under the CC BY -NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).
1. Introduction

Unauthorized code analysis and modification based on reverse engineering is a major concern for the software industry. Such attacks can lead to a number of undesired outcomes, including cheating in online games, unauthorized use of software, pirated pay-tv etc. Industry is looking for solutions for this issue to deter reverse engineering of software systems. By making sensitive code difficult to be traced or analyzed, code obfuscation is a potential solution for the problem.

Code virtualization based on a virtual machine (VM) is emerging as a promising way for implementing code obfuscation (Fang et al., 2011; Oreans-Technology, 2015, 2016; VMProtect-Software, 2015; Wang et al., 2013, 2014; Yang and Huang, 2011). The underlying principal of VM-based protection is to replace the program instructions with virtual instructions which attackers are unfamiliar with. These virtual instructions will then be translated into native machine code at runtime to be executed on the underlying hardware platform. Using a VM-based scheme, the execution path of the obfuscated code is controlled by a virtual instruction scheduler. A typical scheduler consists of two components: a dispatcher that determines which instruction is ready for execution, and a set of bytecode handlers that first decode the bytecode and then translate it into native machine code. This process replaces the original program instructions with bespoke bytecode, allowing developers to conceal the purpose or logic of sensitive code regions.

Prior work on VM-based software protection primarily focuses on making a single set of bytecodes more complex, and uses one single virtual instruction scheduler. This is based on the assumption that the scheduler and the bytecode instruction set are difficult to be analyzed in most practical runtime environments. However, research has shown that this is an unreliable assumption (Falliere et al., 2009) in certain scenarios where an adversary can easily reuse knowledge obtained from other applications protected with the same scheme to perform reverse engineering (termed cumulative attacks in this work). To protect software against cumulative attacks, it is important to have a certain degree of non-determinism and diversity during program execution (Collberg, 2011).

This paper presents DSVMP (dynamic scheduling for VM-based code protection), a novel VM-based code protection scheme to address the problem of cumulative attacks. Our key insight is that it will be more difficult for the attacker to track the application logic if sensitive code regions behave differently in different runs. DSVMP achieves this by introducing rich non-determinism and diversity to program execution. To do so, it exploits a flexible, multi-dispatched scheme for code scheduling and interpretation. Unlike prior work where a program always follows a single, fixed execution path for the same input across different runs, the DSVMP scheduler directs the program to execute a randomly selected path for each protected code region. As a result, the program follows different execution paths in different runs and exposes a non-deterministic behavior. Our carefully designed scheme ensures that the program will produce a consistent output for the same input despite the execution paths might look differently from the attacker’s perspective. To analyze software protected under DSVMP, the adversary is forced to use a large number of trail runs to understand the logic of the program. This significantly increases the cost of code reverse engineering.

Dynamic instruction scheduling in DSVMP is achieved through a combination of two techniques. Firstly, DSVMP provides a rich set of bytecode handlers, each of which has a unique control flow, to translate a bytecode instruction to native code. Handlers for a particular bytecode opcode all generate identical native machine instructions for the same input, but their execution paths and data accessing patterns are different from each other. During runtime, our VM instruction scheduler randomly selects a bytecode handler to translate a bytecode to the corresponding native machine code. Since the choice of handlers is randomly determined at runtime for each bytecode instruction and the implementation of different handlers are different, the dynamic program execution path is likely to be different across different executions. Secondly, DSVMP employs a multi-VM scheme so that various code regions can be protected using different bytecode instruction sets and VM implementations. This further increases the diversity of the program, making it even harder for an adversary to analyze the software behavior or to reuse knowledge extracted from other software products. This is because different products are likely to be protected using different bytecode forms and VM implementations.

The whole is greater than the sum of the parts. These techniques, put together, enable DSVMP to provide stronger code protection than any of the VM-based techniques seen so far. We have evaluated DSVMP on four applications that implement some of the widely used algorithms: “md5”, “aescrypt”, “bcrypt” and “gzip”. Experimental results show that DSVMP provides stronger protection with comparable runtime overhead and code size when compared to two commercial VM-based code obfuscation tools: Code Virtualizer (Oreans-Technology, 2015) and VMProtect (VMProtect-Software, 2015).

This paper makes the following contributions:

- It presents a dynamic scheduling structure for VM-based code obfuscation to protect software against dynamic cumulative attacks.
- It is the first to apply multiple VMs to enhance diversity of code obfuscation.
- It demonstrates that the proposed scheme is effective in protecting real-world software applications.

The rest of this paper is organized as follows. Section 2 introduces the principle of classical VM-based code obfuscation techniques and cumulative attacks scenario. Section 3 describes the VM reverse attacking approach. Section 4 gives an overview of DSVMP, which is followed by a detailed description of the design in Sections 5 and 6. Section 7 uses a case study to demonstrate protection scheme provided by DSVMP. Evaluation results are presented in Sections 8 and 9 before we discuss the related work in Section 10. Finally, Section 11 presents our work conclusions.

---

1 A bytecode is the binary form of a virtual instruction.
دریافت فوری متن کامل مقاله

امکان دانلود نسخه تمام متن مقالات انگلیسی
امکان دانلود نسخه ترجمه شده مقالات
پذیرش سفارش ترجمه تخصصی
امکان جستجو در آرشیو جامعی از صدها موضوع و هزاران مقاله
امکان دانلود رایگان ۲ صفحه اول هر مقاله
امکان پرداخت اینترنتی با کلیه کارت های عضو شتاب
دانلود فوری مقاله پس از پرداخت آنلاین
پشتیبانی کامل خرید با بهره مندی از سیستم هوشمند رهگیری سفارشات