Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

A kernel stack protection model against attacks from kernel execution units

Wangtong Liu [*], *Senlin Luo, Yu Liu, Limin Pan, Qamas Gul Khan Safi*

Information System & Security and Countermeasures Experiments Center, Beijing Institute of Technology, Beijing 100081, PR China

ARTICLE INFO

Article history:
Received 21 October 2016
Received in revised form 7 September 2017
Accepted 11 September 2017
Available online 20 September 2017

Keywords:
Virtualization
Rootkit detection
Control flow integrity
Kernel stack integrity
Ret-to-sched rootkit

ABSTRACT

Many defensive approaches have been proposed to protect the integrity of the operating system kernel stack. However, some types of attacks, such as the "return-to-schedule" rootkit, pose a serious threat to these approaches. In this paper, we present a kernel stack protection model to protect the integrity of the kernel stack. It adopts a synchronous design strategy to bind the execution unit with its kernel stack using virtualization technology, and allows the execution unit to write its own current kernel stack with legal kernel codes. To test the model, we propose three kinds of potential attacks which extend the "return-to-schedule" rootkit. The experimental results show that the prototype of the model can be effective against all attack methods, and introduces a performance cost of only 2%. Therefore, it effectively protects all types of data on the kernel stack with a small performance overhead.

1. Introduction

Kernel-level rootkits pose grave threats to the integrity of the operating system (OS). With the highest privileges of the OS, they are able to modify any kernel data and take complete control of the OS. The integrity of an OS has two aspects: code integrity and data integrity. Code integrity is easier to defend with W⊕X protection (Seshadri et al., 2007); however, data integrity is hard to protect due to its variability. Typical attacks that can undermine data integrity include Kernel Object Hooking (KOH) (Wang et al., 2009), Dynamic Kernel Object Manipulation (DKOM) (Butler and Hoglund, 2004), and the "return-to-schedule" (ret-to-sched) rootkit (Hofmann et al., 2011). The first two types of attack target the data in kernel heaps, while the last type of attack is aimed at the data on kernel stacks.

Currently, virtualization is a significant technology for preventing these attacks, since it provides a higher privilege than the OS. Several defensive methods based on virtualization technology have been proposed to counter the attack methods mentioned above. Systems such as HookSafe (Zhi et al., 2009) keep track of kernel objects of interest to prevent a KOH attack. In addition, systems such as KernelGuard (Junghwan et al., 2009) and VMDetector (Ying et al., 2011) try to detect DKOM attacks by identifying invariants through code analysis or multi-view comparison. However, all of these defensive methods focus on the data in kernel heaps, while leaving the data on kernel stacks dangerously exposed.

The ret-to-sched rootkit (Hofmann et al., 2011) was recently proposed at the well-known ASPLOS academic conference. It is aimed at kernel stacks, and attacks the return addresses on kernel stacks in order to transfer kernel control to the rootkit. It can completely subvert the integrity of the OS

[*] *Corresponding author.*
E-mail address: lwt1231234@126.com (W. Liu).

https://doi.org/10.1016/j.cose.2017.09.008
0167-4048/© 2017 Elsevier Ltd. All rights reserved.

and possesses high-secluded performance. Since kernel stacks are large regions of untyped memory and are resistant to traversal, it is hard to protect the data on kernel stacks.

To the best of our knowledge, OSck (Hofmann et al., 2011) is the first and the only system to handle this kind of threat. It periodically detects the ret-to-sched rootkit by ensuring that the return addresses on the stacks of the scheduled processes correspond to valid kernel code regions. However, the protection it offers for kernel stacks is not comprehensive; its asynchronous design and lack of protection for the other types of data on the kernel stacks may put the operating system at risk. It also fails to take into account some other potential attack methods that may be adopted by an improved ret-to-sched rootkit.

Overall, current integrity protection methods can effectively protect the data in kernel heaps, but cannot completely prevent attacks on the data on kernel stacks. This paper addresses the problem of kernel stack safety to ensure the integrity of the OS kernel. The kernel stack protection (KSP) model is proposed to protect the data on kernel stacks. The key point of the model is that it allows the execution unit (in this paper, an execution unit means a piece of code with its own stack) to write to its own current kernel stack with legal kernel codes. Its synchronous design strategy can defeat a transient attack and guarantee the legitimacy of all kinds of data on kernel stacks. In addition to the ret-to-sched rootkit, it can also defeat some other potential attacks. Finally, a prototype is developed which demonstrates the effective implementation of the proposed method.

Our work offers the following contributions:

- We present the kernel stack protection model to protect the integrity of kernel stacks; it can protect all types of data on kernel stacks.
- We propose three kinds of potential attacks to improve the "return-to-schedule" rootkit: rewriting the kernel stacks by another execution unit, hijacking kernel codes or data to indirectly attack the current kernel stack and modifying the kernel stack pointers.
- Based on this model, we implement a prototype and provide an experimental evaluation of its security and performance.

The rest of this paper is organized as follows. Section 2 describes related work; Section 3 presents the KSP model; Section 4 describes the overall design of the prototype and its implementation; Section 5 evaluates its security and performance; Section 6 discusses the model; and Section 7 presents the conclusion.

2. Related work

There are several defensive methods which focus on the protection of stack data. Some of these protect data from vulnerabilities by introducing additional data onto stacks, and require the help of compilers to add detection codes and data. Some adopt virtualization technology to detect data on stacks; these can make use of asynchronous detection to obtain periodical snapshots, and then analyze their legality. Alternatively, they can utilize synchronous design to intercept some events, and then judge whether or not the data has changed.

OSck (Hofmann et al., 2011) is a system that discovers kernel rootkits by detecting malicious modifications to OS data using virtualization technology. In order to focus on the safety of kernel stacks, we describe only its stack protection measures. To detect the ret-to-sched rootkit, OSck obtains return addresses on the kernel stacks of unscheduled processes through periodical snapshots and ensures that these return addresses correspond to valid kernel code regions. As the author of the OSck has said, the rootkit may construct mechanisms such as the return-oriented program (ROP) attack (Liu et al., 2011; Shacham, 2007) to avoid this detection. OSck therefore needs to be extended with the information of the valid kernel call graph to prevent this kind of attack. However, the call graph is hard to construct precisely for a non-open source OS, and OSck may not therefore work well for a non-open source OS. In addition, the author assumes that "an attacker cannot accurately predict when an OSck check will run". The implication is that if an attacker can accurately predict the time of the check carried out by OSck, the rootkit has a chance to avoid detection. In addition, there are also several other potential attack methods that may frustrate detection by OSck, and these are described in Section 3.

In a similar way to OSck, StackSafe (Liao and Luo, 2015) detects rootkits by periodically scanning the kernel space of the guest OS. StackSafe is remarkably efficient because of its improved scanning strategy. However, its protection of the kernel stack is not comprehensive, since it is not designed to defend against ret-to-sched rootkits. For example, StackSafe provides no protection for the kernel stack pointers, and thus ret-to-sched rootkits can attack the return address through hijacking the kernel stack.

VFEP (Tian et al., 2014) makes use of virtualization technology to detect the return address on stacks, preserving the return addresses for dangerous functions that may be attacked by vulnerabilities. Whenever these functions return, VFEP checks them through trapping into the hypervisor, which introduces a high performance cost. However, ret-to-sched rootkits can tamper with any return address (not just those of dangerous functions) on the kernel stacks, and this would be prohibitively expensive for the VFEP to defend against.

In conclusion, only OSck is designed to defend against the ret-to-sched rootkit; the other types of method are not suitable for defending against this rootkit. However, OSck is inadequate to protect the integrity of kernel stacks. This paper proposes a kernel stack protection model to guard the integrity of kernel stacks.

3. Kernel stack threat model and protection model

In this section, a brief background and introduction to existing kernel stack attack methods is given, and the basic defensive model of KSP is proposed in order to defend against these methods of attack and to protect the integrity of the kernel stack.