Countering Kernel Rootkits with Lightweight Hook Protection

Michal Sekletar

College of Engineering and Computer Science
Orlando, FL

April 2, 2012

Acknowledgment

- Authors:
  - Zhi Wang, NC State University
  - Xuxian Jiang, NC State University
  - Weidong Cui, Microsoft Research
  - Peng Ning, NC State University
- 16th ACM Conference on Computer and Communications Security (CCS)
- November, 2009

Background – Rootkits

- Definition
  A rootkit is a stealthy type of malicious software (malware) designed to hide the existence of certain processes or programs from normal methods of detection and enables continued privileged access to a computer
- Ways of installation
  - Automated
  - Manual
- Infects target machine typically exploiting vulnerabilities in some other applications

Background – Hooking

- Definition - Hook
  Function pointers, return addresses, e.g.
  `ext3_dir_operations->readdir`
- Definition - Hooking
  Techniques used to alter or augment the behavior of an operating system, of applications, or of other software components by intercepting function calls or messages or events passed between software components.

Contributions

- Design, implementation, and evaluation of HookSafe
- Hooksafe - Hypervisor-based lightweight system that can protect thousands of kernel hooks from being hijacked by kernel rootkits.
- Efficiency of defense against rootkits using HookSafe
- Low overhead introduced using the tool

Problem Overview

- Classification of kernel rootkits
  1. Kernel Object Hooking (KOH) - hijack kernel control flow
  2. Dynamic Kernel Object Manipulation (DKOM) - modify dynamic data objects
- Majority of kernel rootkits are KOH rootkits (96%)
- Rootkit of type 1) can gain control over kernel execution
  1. Code hooks
  2. Data hooks - most common type
- Kernel hooks are scattered across kernel space
- Prior techniques are not suitable for protecting significant amount of hooks

HookSafe Architecture

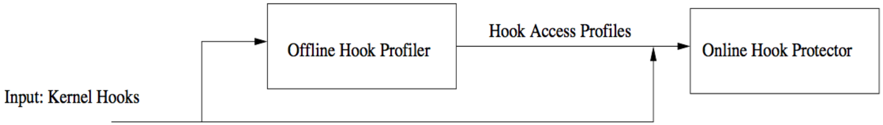
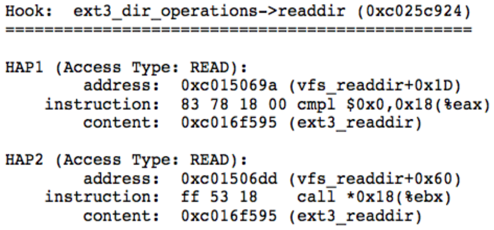


Figure: The HookSafe Architecture

- Offline hook profiler
- Online hook protector

Offline hook profiler

- Component profiles the guest kernel execution and outputs a hook access profile for each protected hook
- Dynamic analysis runs the target system on top of an emulator (e.g., QEMU) and monitors every memory access to derive the hook access instructions.
- Output of this analysis is hook access profile



```
Hook:  ext3_dir_operations->readdir (0xc025c924)
=================================================

HAP1 (Access Type: READ):
        address:  0xc015069a (vfs_readdir+0x1D)
    instruction:  83 78 18 00 cmpl $0x0,0x18(%eax)
        content:  0xc016f595 (ext3_readdir)

HAP2 (Access Type: READ):
        address:  0xc01506dd (vfs_readdir+0x60)
    instruction:  ff 53 18     call *0x18(%ebx)
        content:  0xc016f595 (ext3_readdir)
```

Figure: Hook access profile

Online hook protector

- Multiple tasks
    - Initialization
        - Creates shadow copy of kernel hooks and loads indirection code layer
        - Online patching of guest OS kernel
    - Run-Time Read/Write Indirection
        - Reads are indirected to shadow copy and returned
        - On write hypervisor will validate record and update shadow hook if request is valid
    - Run-Time tracking of dynamically allocated objects
    - Hardware register protection
- Developed on Xen hypervisor

Online hook protector - overview

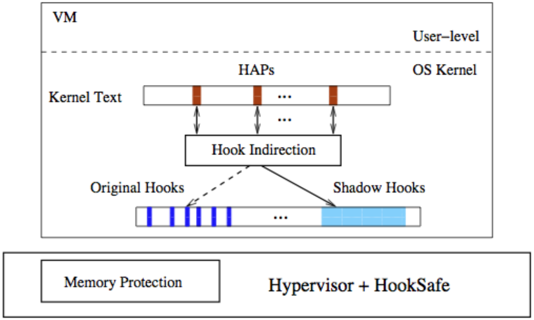


Figure: The architecture of online hook protection

Online hook protector - overview

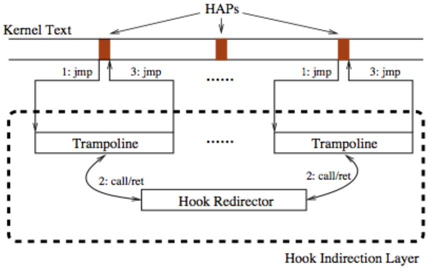


Figure: The implementation of hook indirection

Evaluation

- HookSafe tested against nine real-world rootkits
- HookSafe successfully prevented these rootkits from modifying the protected kernel hooks
- Very little overhead, around 6%

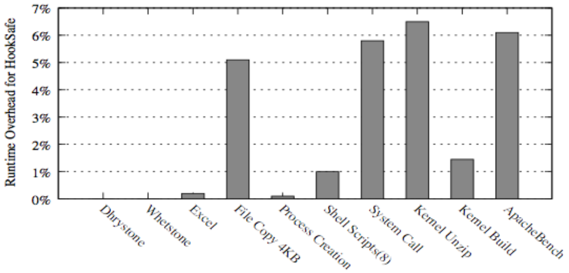


Figure: Results of benchmarks

Discussion

- Two major drawbacks
  1. Coverage of dynamic analysis done by offline hook profiler
  2. HookSafe assumes the prior knowledge of the set of kernel hooks that should be protected
- Solutions
  1. Combine dynamic analysis with complementary approach - static analysis
  2. Combine HookSafe with some other hook discovering tool, e.g. HookFinder, HookMap

Questions?