Copilot - a Coprocessor-based Kernel Runtime Integrity Monitor

Nick L. Petroni, Jr.  npetroni@cs.umd.edu
Timothy Fraser  tfraser@umiacs.umd.edu
Jesus Molina  chus@cs.umd.edu
William A. Arbaugh  waa@cs.umd.edu
Kenneth Grauel
Copilot Overview

- Protects commodity operating systems
  - Detects malicious modifications to running system
- Minimal effect on monitored system
  - Requires no change to existing host software
  - Less than 1% performance penalty
- Effective and robust
  - Has detected 12 real-world rootkits for GNU/Linux
  - Detection window of under 30 seconds
  - Operates even if host kernel is fully compromised
State of commodity OS security

• Complexity abounds
  - Commodity OS’s are already complex (and growing)
  - Placing assurance on the many parts is difficult

• Existing security tools rely on system correctness
  - All host software relies on some aspect of kernel integrity
  - This assumption is invalid: attackers modify the kernel

• Copilot provides independence from the host OS
  - Utilize direct access to system resources
  - Perform complex checks without host intervention
Copilot Monitor Experiment

Host

Monitor (PCI card)

independent communications link

Admin Station
What is a Rootkit?

Software used after system compromise to:
  • Hide the attacker’s presence
  • Provide backdoors for easy reentry

Simple rootkits:
  • Modify user programs (ls, ps)
  • Detectable by tools like tripwire

Sophisticated rootkits:
  • Modify the kernel itself
  • Hard to detect from userland
Rootkit Features

Typical rootkit implementation:
- An LKM that interposes on the system call vector:
  - Adore, rial, rkit, synapsis, modhide1, phide,kbd, linspy…

More sophisticated, more stealthy:
- SucKIT - loads via /dev/kmem instead of LKM
  - Phantasmagoria - modifies kernel text, not syscall vector

Insecurity by Obscurity:
- Taskigt - adds a hook to /proc filesystem
  - Knark - adds inet protocol handler
Limitations of Host-based Tools

Userland tools: chkrootkit, checkps, Rkscan, RootCheck…
  + Compare ps and /proc, directory link and entry counts
  - When the kernel lies, all will seem well in userland
  - Some are designed to detect only known rootkits

Kernelspace tools: KSTAT, St. Michael, Carbonite, Samhain
  + Examine kernel data structures via /dev/kmem or an LKM
  - Rootkits can make /dev/kmem and LKMs lie, too

“Arms Race”
Correctness Dependencies

- Applications
- Operating system
- CPU, cache
- PCI local bus, memory controller, main memory
- Copilot
PCI add-in card requirements

• Unrestricted access to memory
  - EBSA-285 has bus mastering capability
• Independence from host
  - EBSA-285 has a mode that ignores host commands
• Sufficient processing power, memory
  - StrongARM SA-110 CPU, 16MB RAM
• Independent communication channel for reporting
  - RS-232 serial port
Linux Virtual memory translation

virtual addresses used by kernel:

- 0xc0000000
  - kernel text, page tables
- high_memory
  - vmalloc area, module cores
- 0xfe000000

physical addresses used during DMA:

- 0x00000000
  - end of RAM
- nonlinear map via page tables
- linear map

0x00000000
end of RAM
STREAM memory throughput benchmarks

<table>
<thead>
<tr>
<th></th>
<th>copilot off</th>
<th>copilot on</th>
</tr>
</thead>
<tbody>
<tr>
<td>copy penalty</td>
<td>10%</td>
<td>10%</td>
</tr>
<tr>
<td>scale penalty</td>
<td>10%</td>
<td>10%</td>
</tr>
<tr>
<td>add penalty</td>
<td>7%</td>
<td>7%</td>
</tr>
<tr>
<td>triad penalty</td>
<td>7%</td>
<td>7%</td>
</tr>
</tbody>
</table>
WebStone HTTP throughput benchmark

Cycle (in seconds): off 30 15 5 continuous
Penalty: 0% 1% 2% 4% 14%

MB/s
Copilot Summary

• Proven effective in lab tests:
  - Detected the 12 rootkits listed on earlier slide.
  - 30-second detection window
  - Less than 1% application performance penalty

• Clear advantage over existing technologies:
  - No reliance on host software for correctness
  - Plugs into unmodified commodity host
Future

• New boards with NIC for out-of-band communications
• Integrate previous work (FS integrity monitoring)
• Privilege escalation detection
• Remote configuration, reconstitution, and forensic analysis
END
## Rootkit Taxonomy

<table>
<thead>
<tr>
<th>Rootkit:</th>
<th>Unusual methods:</th>
</tr>
</thead>
<tbody>
<tr>
<td>adore 0.42</td>
<td></td>
</tr>
<tr>
<td>knark 2.4.3</td>
<td>adds /proc, inet hooks</td>
</tr>
<tr>
<td>phantasmagoria</td>
<td>mods text, not syscall vector</td>
</tr>
<tr>
<td>rial</td>
<td></td>
</tr>
<tr>
<td>rkit 1.01</td>
<td></td>
</tr>
<tr>
<td>SucKIT 1.3b</td>
<td>loads via kmem, not LKM</td>
</tr>
<tr>
<td>synapsis 0.4</td>
<td></td>
</tr>
<tr>
<td>taskignt</td>
<td>adds /proc hook</td>
</tr>
</tbody>
</table>
But wait there’s more
Ported to a new board

- Supports *out of band* command and control, *i.e.* it has a dedicated ethernet interface.
- Supports booting from a virtual floppy, remote power cycle and reset.
- Also remote virtual terminal.
Windows Protection

• **Windows 2000**
  – SDT - Service Descriptor Tables
  – IDT - Interrupt Descriptor Table
  – GDT - Global Descriptor Table
  – Kernel Text

• **Windows XP doable (just not finished yet)**
Demo

• Windows 2000 SP4 machine with co-pilot add-in board.
  – Show how co-pilot detects the presence of SoftIce
  – Show how co-pilot detects the basic_8 rootkit
Future work

• Dynamic reconstitution and forensic reporting, e.g. transmit malicious code to central monitoring station and rebuild system.

• Deepen the monitoring capability into the process level, e.g. determine when a process has gained root level privileges without authorization.