

# SubVirt: Implementing malware with virtual machines

Samuel T. King, Peter M. Chen, Yi-Min Wang,  
Chad Verbowski, Helen J. Wang, Jacob R.  
Lorch.

IEEE Symposium on Security and Privacy,  
May 2006

Presented by: Ramsés Morales

# Motivation

- Attack and defense are going to lower system layers.
- Evaluate new class of rootkits: Virtual Machine Based Rootkit (**VMBR**).
  - Malicious Virtual Machine below the host OS.
- How to detect and prevent VMBRs?

# Motivation: Control

- Determined by who (attacker/defender) occupies lower layer.
- Malware on lower layer can
  - hide its state,
  - manipulate upper layer's state.
- Malware has migrated from user-level (trojans), to kernel-level [FU-rootkit].

# Motivation: Rootkit design issues

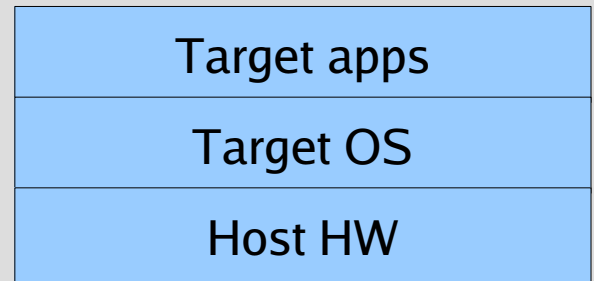
- If both malware and security software reside inside the kernel:
  - The one that better understands and anticipates the enemy, wins.
- Large feature-set in a RootKit makes it easier to detect.
  - Memory footprint.
  - Extra open ports.
  - Extra files.
- Solution: **slide malicious VMM below host OS.**

# VMBR's Design

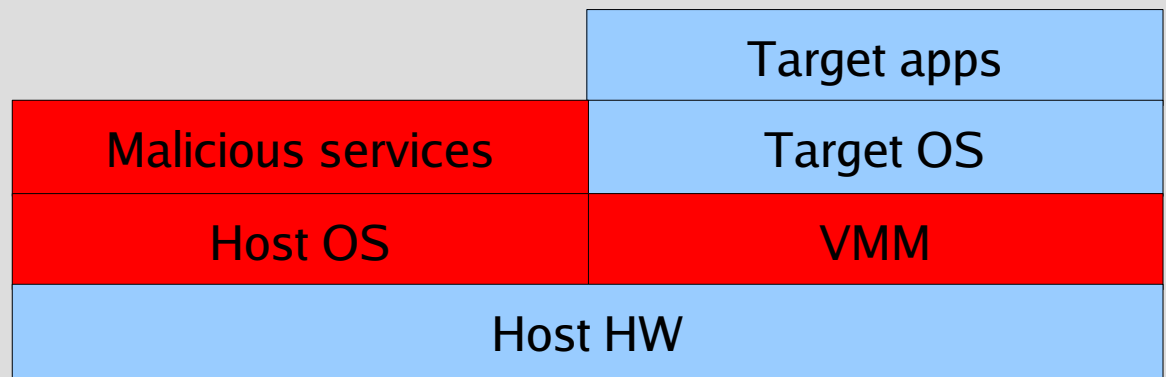


VMBR's components

Before infection →



After infection →



# How-to

- Install VMBR on host machine.
  - Gain root privileges remotely.
  - Social engineering.
  - Corruption.
  - Trojaned CD/DVD/download.
- Modify boot sequence to boot VMBR.
- VMBR boots Target OS.
- 
- 
- Installation is VMBR's main window of vulnerability.

# Implementation: Installation

- Target: **Windows XP**
  - VMBR's state on **first active partition**.
- Target: **Linux**
  - VMBR's state on **swap partition**.
  - Disable swapping.

# Implementation: boot sequence modification

- Must avoid anti-malware detection of boot modification.
  - **Manipulate blocks during final stages of shutdown.**

# Implementation: boot sequence modification

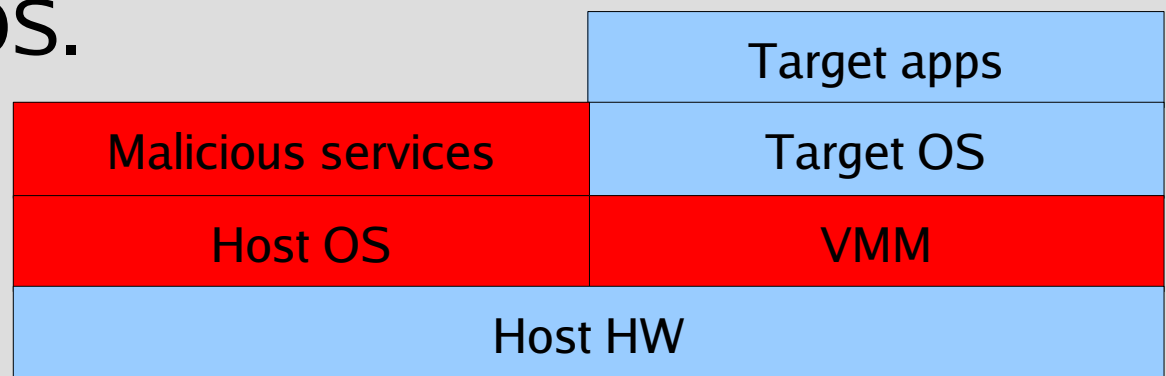
- **Windows XP:**
- Install event handler using `LastChanceShutdownNotification`.
- Use low level disk driver to write.
- Only malware can use `write` during installation.
- 
- 
- Can kernel-level malware-detector outsmart the attack before system shutdown?

# Implementation: boot sequence modification

- **Linux:**
- Add shutdown script.
- Run just before system shutdown.
- Modify boot with HD block device.
- 
- 
- 
- Can kernel-level malware-detector outsmart the attack before system shutdown?

# Malicious Services

- Easy to implement.
- Not detectable by Target OS.
- Types:
  - No interaction with Target OS.
    - Zombies, spam relay...
  - Observe Target OS.
    - Modify VMM's device emulation software.
    - Trap-based VM Introspection.
  - Tamper Target OS.



# Service: Phishing Website

- On VMware VMBR.
- thttpd.
- Forward incoming TCP requests on 8080 to phishing server.

# Service: Keystroke logging

- On Virtual-PC VMBR.
- Modification on keyboard controller emulation module.
  - 60 lines of code.
- Extracts passwords.

# Service: Sensitive data scanner

- On VMware VMBR.
- 24 line Perl script.
- Scans file system for passwords and private ssh keys.

# Service: Defense countermeasure

- On Virtual-PC VMBR.
- Avoid *redpill's* VMM detection.
- Interpose on Windows XP's code that maps an executable into process address space.
- Detect if redpill is being loaded.
- Emulate `sidt` instruction when called by redpill.

# Maintaining Control

- Avoid power-off and reboot:
- Emulate reboot by restarting virtual HW.
- Use ACPI sleep states to emulate shutdown.
  - Requires careful handling of BIOS to manipulate LEDs (if possible).
- 
- 
- 
- Power analysis to detect VMBR?

# Evaluation

(all times in seconds)	Installation	Target Boot, no VMBR	Target Boot After Emulated Reboot	Target Boot After Emulated Shutdown	Host Boot After Power-Off	Host Boot + Target Boot After Power-Off
Vmware VMBR, Linux Target, 228MB	24	53	74	96	52	52 + 93 = 145
Virtual-PC VMBR, Windows XP Target, 251MB	262	23	54	N/A	45	45 + 56 = 101

- Vmware VMBR runs on 2.8GHz PIV, 1GB RAM.
- Virtual-PC VMBR runs on 1GHz PIV, 256MB RAM.

# Evaluation

- Performance will be affected.
  - User might not notice it.
- Specialized guest drivers improve performance.
- Could be used to detect VMBR.
  - Arms race?
  - What about external detection mechanisms?

# Evaluation

- Memory allocated to VMM and attack OS: 3%.
- VMBR can present Target OS with same amount of memory as physical system.
- Paging Target OS memory can result in minimal performance impact.

# Detecting VMBR (security below it)

- Secure HW.
  - Intel's LaGrande.
  - AMD's trustworthy computing.
  - Copilot.
- Boot from secure medium.
  - CD-ROM.
  - USB.
  - Network.
- Secure VMM.
  - Can stop VMBR's boot sequence modification.
  - Attestation for boot components.
  - Can we implement VMBR profiling on sec-VMM?

# Detecting VMBR (security above VMBR)

- Take advantage of perturbations.
  - CPU overhead.
  - Memory overhead.
  - Disk overhead.
  - I/O device variety.
  - Imperfect virtualization.
- VMBR can be extended to counter some.
- Upcoming virtualization support on HW can be used to decrease perturbations, so...
- **How close can we come to the ideal VMBR with HW support?**

# Newsgroup Discussion

- External detection mechanisms?
  - Switch / NAT to detect weird packets.
  - Power analysis.
  - Wrist watch?
- VMBR profiling.
  - Secure VMM taking advantage of VMBRs?
  - Obvious arms race.
- We can detect during and after installation.
  - Does it matters?
  - Keep in mind current malware infections in spite of current security.