



Confidence in a connected world.

External Monitoring of Endpoint Configuration Compliance

Darrell Kienzle, Matthew Elder, and Ryan Persaud
Symantec Research Labs
April 14, 2009

EC-CAM Overview

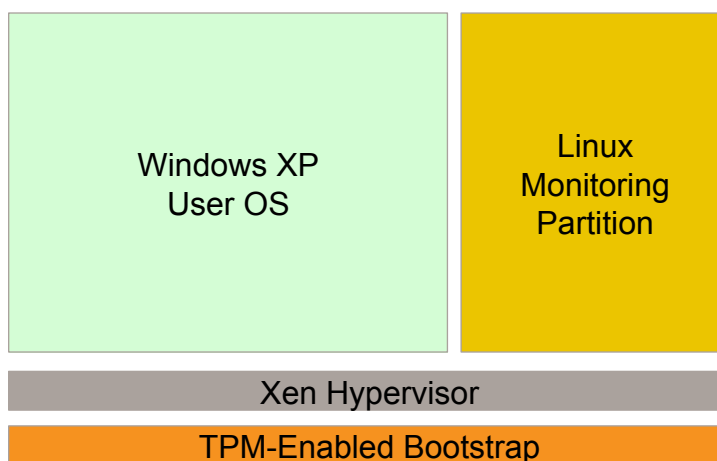


- *Enterprise Configuration Compliance Administration Manager*
 - Assess the overall enterprise risk
 - Focus on configuration compliance
 - Ensure that systems meet policy
 - Not equivalent to real-time blocking
- Compliance from inside the OS is difficult
 - Traditional agents execute in the context of the OS they monitor
 - Rootkits can create a *hall of mirrors*
- The EC-CAM Agent
 - Take advantage of virtualization hardware and software
 - Observe each system from outside a virtual machine
 - Much harder to spoof, emulate, or circumvent

The Semantic Gap

- Virtualization introduces a *semantic gap*
 - Policy is expressed in high-level constructs
 - Enforcement occurs at low-level interfaces
- We have perfect visibility into the Virtual Machine
 - Very high confidence that the bits and bytes are reported accurately
 - How confident are we that we know what they mean?
- Alternately
 - We are not dependent on untrustworthy OS APIs
 - But we can no longer depend on OS APIs

Endpoint Architecture



Analysis Overview



- Analysis
 - Pause the virtual machine (e.g., nightly check)
 - Inputs: raw memory, swapfile, file system
- Periodic checks
 - Executables in file system
 - Registry settings
 - Running process code
 - *Adding* stored CPU state
- Real-time checks
 - Monitor hardware changes

Checking the File System



- File system compliance
 - Are any disallowed or unknown applications present on disk?
 - Disk may be virtualized or raw block device
- Process
 - ntfs-3g presents raw disk as a *read-only* file system
 - Walk the file system, looking for Portable Executable (PE) headers
 - Hash contents of PE files
 - Check hashes against a whitelist of allowable applications

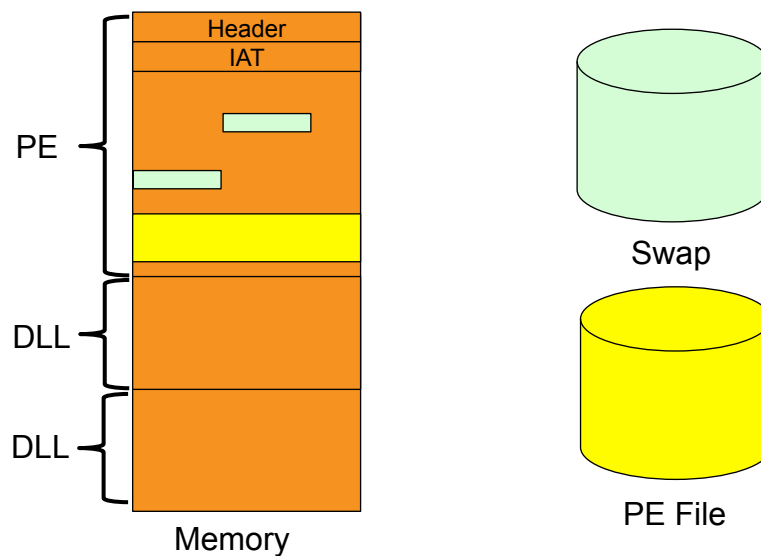
Checking the Registry

- Windows Registry compliance
 - System and application configuration database
 - Stored in undocumented “hive” files
 - Accessed exclusively through APIs (inaccessible to us)
- Process
 - Access the registry hive files through ntfs-3g
 - Use ntreg library to get read-only access
 - Examine specific security-relevant keys, installed software

Checking Running Processes

- Running Process compliance
 - Are necessary security applications running?
 - Are any unknown or disallowed programs running?
 - Do any running programs or drivers show signs of tampering?
- Process
 - Modified Xen *dump-core* creates flat file of VM memory image
 - ntfs-g provides access to swap file, program files
 - Check that each executing program is consistent with disk image
 - Report hashes of executing programs

Checking a Running Process



Observations: Consistency

- Concern about accessing memory/files beneath APIs
 - Structural integrity
 - Uncommitted cache data
- Integrity has not been a problem
 - The Windows registry is (now) quite resilient
 - Memory structures are fairly stable
 - Tools have proven fairly robust
- Uncommitted changes have also not been a problem
 - Disk changes settle fairly quickly
 - Compliance-relevant changes are rare
 - Worst case, changes will be detected in the next pass

Other Observations

- Windows XP
 - Trial and error (“Operationally specified”)
 - Some compatibility issues across Service Pack versions
 - No hotfix has broken any of our code
- Generality
 - Successfully applied to VMware Workstation & Fusion snapshots
 - Should be applicable to hibernated system
- False Positives
 - Legitimate modifications to NTOSKRNL, HAL
 - Changes to several Microsoft DLL

Related Work

- Virtual Machine Introspection
 - Garfinkel and Rosenblum, *A Virtual Machine Introspection Based Architecture for Intrusion Detection*, 2003
 - Jiang, Wang, and Xu, *Stealthy Malware Detection through VMM-based “Out-of-the-Box” Semantic View Reconstruction*, 2007
 - Litty, Lagar-Cavilla, and Lie, *Hypervisor Support for Identifying Covertly Executing Binaries*, 2008
- Memory forensics
 - Kornblum, *Using Every Part of the Buffalo in Windows Memory Analysis*, 2007
- Rootkit detection
 - Rutkowska, *System Virginity Verifier*, 2005

Discussion

- Malware advances
 - We have observed malware unlink itself from process lists
 - Could malware construct shadow copies of Windows data structures
 - Could it “unlink” an entire core?
- Live Forensics
 - Most Windows structures appear fairly stable
 - Could sufficiently resilient code work without pausing the OS?
 - Preliminary experimentation has been promising
- JIT Compilers / Interpreters
 - How to distinguish inserted code from JIT-compiled code?
 - What are the implications of whitelisting an interpreter?



Confidence in a connected world.

Thank You!

Darrell Kienzle
darrell_kienzle@symantec.com