Forensic Extensions for VirtualBox

Chris Lockfort

Overview

- Introduction
- Changes
- Demo
- Difficulties
- End state of the project
- Questions+Heckling

Specific Added Features / Changes to Existing Features

- Integration with vboxshell application to interact with running virtual machines
- Ability to instantly dump memory/disk from a live box
  - Live analysis!
    - Much better than just a disk image after-the-fact.
    - Memory-based rootkits, etc, can't hide.
  - No need to disturb a critical server from doing its job just to take a look at it.
  - Better than live dump of memory from kernel (i.e. MS COFEE, windd, WindowsSCOPE) because it cannot be affected by smart rootkits.

Added Features / Changes (Cont'd)

- Presents memory/disk image to the forensic examiner in a non-proprietary format that is easy to use other tools with.
  - Previously, snapshots were in an undocumented format that was a mishmash of current disk state and memory state in one file. Also this disk state format represents only the disk state that we *don't* want, the portion of disk changes occurring *after* the snapshot.
  - Now separate raw disk and ELF/core format memory images, easy to manipulate with any tools.
- Creates and records multiple-format (MD5+SHA1) hashes of the images and records timestamps for **all** files created.

Unexpected Free Beer

- For code simplification/deduplication reasons, the current save function (*VMR3Sav(...)*) is internally using the brand new VM teleportation functions (*vmR3SaveTeleport()*), but simply to the local machine.
- For free we get an extremely powerful tool that can magically:
  - Connect to a remote VM Host machine
  - Dump memory/RAM from running VM guest system while it is still running
  - Securely have the images shipped back to you.
  - Separate/ standardize disk/ memory format / hash

Demo

A Fairly Unavoidable But Serious Issue

- The code attempts to check for this issue, but you may need a ludicrous amount of disk space in order to fit, temporarily, before the in-between stages are deleted (assuming a local-host local-dump situation):
  - VM running disk
  - VM forensic duplicate of disk+memory(in proprietary format)
  - VM forensic duplicate of disk (separated)
  - VM forensic duplicate of memory (separated, raw format)

  For instance, a 120GB HDD + 12GB RAM server could need up to 384GB of disk space.

Partial Fix

- This can be vastly improved by not using raw disk images; if your forensic utilities support QCOW2/VDI images, this would be an option.
- Additionally, you can be saved by sparse or transparent-compression filesystem functionality (i.e. EXT4, NTFS, BtrFS, ZFS, etc)
  - Notably, only 'modern' filesystem without sparse feature is HFS+
    - Could possibly use transparent compression or built-in mounted-loopback-disk-grow mac-specific functionality.

Other Issues

- Oracle's version of “Open Source Software” makes it fairly hard to actually push code to them
  - Can't ever actually push code unless you work for Oracle; have to get someone from Oracle to vet your patches and submit them into the repository.
- Some parts of API/SDK are being rewritten and are currently broken.
  - They simply return “not yet implemented” errors

End State of the Project

- Python modules/command line utility; integrates with VirtualBox's 'VBoxShell'
  - https://github.com/clockfort/vbox-save
- Found some vboxshell weirdness where it was not clear at all what box (remote/local/other remote?) you were running API calls on, and successfully upstreamed a patch to fix it into VirtualBox proper