Linux Memory Analysis with Volatility

Andrew Case
Digital Forensics Solutions
Purpose of the Talk

• To highlight the Linux analysis capabilities integrated into the Volatility framework within the last year

• Some of it is implementation of previously published works

• The rest is previously never disclosed analysis techniques
  – Mostly related to advanced rootkit detection
The General Plugins

• The goal of the general plugins is to recreate the set of commands that would be run on a Linux system to investigate activity and possible compromise

• Recovery of this information has been covered in a number of publications by a wide range of authors [1]
Recovered Process Information

• Process listing (*ps aux*)
  – Command line arguments are retrieved from userland

• Memory Maps (*/proc/<pid>/maps*)
  – Can also recover (to disk) specific address ranges

• Open Files (*/proc/<pid>/fd*)
Recovered Networking Information

• Network interface information (`ifconfig`)
• Open and listening sockets (`netstat`)
• ARP tables (`arp –a`)
• Routing table (`route –n`)
• Routing cache (`route –C`)
• Queued Packets
• Netfilter NAT table (`/proc/net/nf_conntrack`)
  – Src/Dst IP, # of packets sent, and total bytes for each NAT’d connection
Recovered Misc. Information

- Kernel debug buffer (*dmesg*)
- Loaded kernel modules (*lsmod*)
- Mounted filesystems (*mount, /proc/mounts*)
- CPU Info (*/proc/cpuinfo*)
Recovering Historical Information
Recovering Historical Information

• Implemented using the `kmem_cache` analysis presented at DFRWS last year [2]

• Briefly, the `kmem_cache`:
  – Provides a consistent and fast interface to allocate objects (C structures) of the same size
  – Keep freelists of previously allocated objects for fast allocation

• Walking the freelists provides an orderly method to recover previous structures
Results of *kmem_cache* Analysis

- Can recover a number of useful structures:
  - Processes
  - Memory Maps
  - Networking Information
  - See `/proc/slabinfo`

- Two limitations:
  - The aggressiveness of the allocator (SLAB / SLUB) when removing freelists
  - Needed references being set to NULL or *freed* on deallocation
Rootkit Detection Techniques
Rootkit Detection Techniques

• Volatility, as well as other projects, has the ability to detect boring rootkits techniques:
  – Code (.text) overwriting
  – System Call/IDT hijacking

• Volatility is the only public project to be able to detect advanced, data modification-only techniques
Leveraging *kmem_cache* (again)

• We previously discussed using the freelists to find historical data

• Can also use the allocated lists to find all instances of a particular structure
  – The one caveat is SLUB without debugging on
  – Every distro checked enables SLUB debugging
  – Might be possible to find all references even with debugging off
The Idea Behind the Detection

- Dynamic-data rootkit methods work by removing structures from lists, hash tables, and other data structures.
- To detect this tampering, we can take a particular cache instance and use this as a cross-reference to other stores.
- Any structure in the kmem_cache list, but not in another, is hidden.
  - Inverse holds as well.
Live CD Analysis
Live CD Analysis

• Live CDs present a problem for investigators as they run entirely in RAM
  – No disk or filesystem(s) to do forensics analysis on

• Privacy / Anti-Forensics people are aware of this – see [5]
  – TAILs privacy live CD that forces all network through TOR and discusses avoiding disk forensics

• ... but the filesystem is memory!
  – Memory analysis can recover the entire filesystem
Live CD Filesystems

• Live CDs use a stackable filesystem that merges multiple filesystems into one view for the OS/users

• Live CDs use this to boot off a read-only CD and then store all changes in an in-memory filesystem (*tmpfs*)

• Recovery of the in-memory filesystem finds all the created & modified files since system boot
  – Immediately reveals the user’s activities
Recovering the Filesystem

- Complete details are in [4]
- The recovery plugin recovers the in-memory filesystem and writes it to disk
  - Gathers metadata such as MAC times*, owner/group, etc
- The recovered FS can be written to a disk image using loopback or to other media and then mounted read-only for normal investigation
Android Analysis
Android Analysis

• If you didn’t know, Android runs Linux
  – Means we can analyze it using similar techniques

• Volatility contains two analysis parts:
  1. Kernel Analysis
  2. Dalvik Analysis
Kernel Analysis

• Same capabilities as discussed throughout presentation

• “Porting” to Android (ARM) from Intel only required adding an “address space” for the architecture
  – Address spaces handle virtual address to physical offset translation
Dalvik Analysis

• Dalvik is the software VM for Android
  – Very similar to the JVM
  – Controls all Android applications

• I recently presented on analyzing Dalvik memory captures to gather application-specific data [6]
  – Call Information
  – Text messages
  – Emails
  – And so on...
Listing Instance Members

Source file: ComposeMessageActivity.java
Class: Lcom/android/mms/ui/ComposeMessageActivity;
Instance Fields:
  name:    m_receiver
  signature:  Landroid/content/BroadcastReceiver;

  name:    m_filter
  signature:  Landroid/content/IntentFilter;

  name:    mAppContext
  signature:  Landroid/content/Context;

  name:    mAvailableDirPath
  signature: Ljava/lang/String;
Conclusion

• Volatility Linux lives!
  – See the linux-support branch in SVN
  – Support tested on Intel Linux 2.6.9 to 2.6.3x
  – Android/ARM tested on a number of phones
    • Feel free to send me Android tablets if you want specific support for them ;)

• Hopefully will have a proper release with all the discussed features within the next few months
Questions/Comments?

• Contact:
  – andrew@digdeeply.com
  – @attrc
References