Security Workshop

Dr. Carl Pulley
c.j.pulley@hud.ac.uk

Initial Infection Vector

- How did the malware initially infect the system?
- How did the malware initially infect the organisation?
  - can we identify patient zero?
- Need to check
  - media inserted into devices
  - network/browser connections

Monday, 28 February 2011

Registry analysis can often help here.

Propagation

- How does the malware move between hosts?
- Can we identify vulnerabilities it's exploiting?

Persistence

- Registry keys (see week 21)
- Startup files (eg. services, drivers, scheduling)
- Add-on to an existing application
- Patch disk binaries
- Modify MBR
- System baselining and fuzzy hashing can aid detection here

Monday, 28 February 2011

We can perform fuzzy hashing using tools such as md5deep.

Artefacts

- We can use Locard's *exchange principle* here and look for artefacts that malware leaves behind when running
  - has the hosts file been modified?
  - any disabled or deleted security applications?
  - what about network traffic or log files?
  - etc.

Anti-Virus Software

- Ultimately a flawed idea
  - check for signatures that are indicative of an infection
    - polymorphic code changes its signature frequently (eg. koobface)
  - don't want false positives!
    - in memory malware often missed
    - staged delivery makes detection **very** difficult

Hooking

- API hooking
- Hooking of system table or exported functions
  - eg. GDT, LDT, IDT, etc.
- Can use `!chkimg` in WinDBG to aid detection
  - compares memory code with disk copy

Monday, 28 February 2011

With the Volatility framework, the apihooks plugin aids detection here.

Hooking

- Hooking unexported functions
  - aim to hook functions deep in the kernel
    - eg. Deepdoor by Joanna Rutkowska
  - detection is **very** difficult!

Process Leaching

- Inject executable code (as a thread) into a running process
  - leach CPU time from the running process
- Techniques such as reflective DLL injection means the PEB does **not** record the thread!

Monday, 28 February 2011

With the Volatility framework, the malfind2 plugin aids detection here.
We can also use thrdscan2 and pslist to count threads running under a process.
Detection is also possible by comparing the DLL list of the in-memory process with its on-disk counterpart.

DKOM Rootkits

- Unlink from EProcess double linked list
- Detection possible using cross-view detection techniques
  - list processes, threads and drivers from memory pool allocations
  - compare with data from doubly linked lists

Monday, 28 February 2011

With the volatility framework, the psscan2 and orphan_threads plugins aid detection here.
We can also use thrdscan2 and pslist to count threads running under a process.
A newer process searching plugin is csrss_pslist (this plugin searches the crss process handle DB to locate processes that have launched since crss started).