Constructing a Stable and Verifiable Computer Forensic System

Daniel Ayers
Elementary Solutions Ltd
Auckland, New Zealand

Open Source Digital Forensics Conference 2011
This talk is about validation of computer forensic software

- Difficulties validating and using computer forensic tools on general purpose operating systems
- What can we do with open source software, including TSK & Linux, to help?
Definitions

* **Tool** – Computer forensic software executing within a general purpose operating system

* **Positive Validation** – Ability to extrapolate from successful test(s) that tool is correct.

* **Negative Validation** – Ability to demonstrate through unsuccessful test(s) that tool is incorrect.
An Experiment

* Hypothesis – Change in OS environment can cause a correct tool to give incorrect results
* Tested – EnCase v6.18 & Linux (Debian Lenny)
* Results
  * Modification of OS TZ database broke date/time calculations (EnCase & Linux, EnCase broken anyway)
  * Modification of OS codepage/NLS definitions broke keyword searching (EnCase, Linux inconclusive)
"Correct" tool provided by vendor

Relies upon proper operation of operating system, firmware and hardware
Conclusions from Experiment

- Generic positive validation of a tool (“Tool X v1.4 works correctly”) is not possible
- A successful validation test means tool works on that particular computer or one with the same characteristics (equivalence)
- Faults can originate from
  - OS patches (e.g. US DST patch for Windows)
  - Misconfiguration
  - Security compromise (anti-forensics)
  - Changes in date and/or time
Computer Forensic System – Tool plus all hardware and software capable of influencing the behaviour of the tool.

How can you ascertain the scope of a system?
- Includes specific hardware & software
- Examine source code (for open source tools)
- strace/ptrace/Process Monitor (closed source)
But ... License Restrictions!

- The terms of the software license for most closed source tools prohibit reverse engineering and similar activities
  - It may not be legal to examine the tool in sufficient detail to identify what OS services, libraries and configuration data it relies on
  - A dead end for closed source?
A “forensic appliance”

- Based upon general purpose OS & open source software
- Automatic updates disabled
- Configuration control software (e.g. Puppet)
- Integrity verification software (e.g. Tripwire)
- Verification of hardware & firmware using diagnostics & burn-in software
- Access evidence data via Lustre, NFS, CIFS or web services.

- Clusters comprised of many appliances
Appliance Life Cycle

Version 1

BUILD  
TEST  
OPERATE

Freeze Configuration

Verify Integrity

Version 2

BUILD  
TEST  
Hardware Qualification

- Need to establish reliable operation of hardware and firmware
- Vendor diagnostic software
- Burn-in software
- Memtest86+
- IPMI/Hardware monitoring for early detection of problems
- Verify disk operation – prefer hardware RAID
* Select stable software (ad-hoc updates not possible)
* Minimal software install
* Automated configuration management (e.g. Puppet “ensure => version”)
* Freeze Configuration
  * Disable automated updates (lock file, null sources.lst)
  * Install & configure tripwire
Test Phase

* Conduct sufficient testing to support positive validation of all components of system
* Tests should compare output of software on system with known correct results
* Keep detailed records of tests and results (may be required as evidence)
Monitor integrity of system (e.g. via tripwire and IPMI/BMC/iLO/etc)
Occasional repetition of test suite (e.g. when the appliance is not required)
Maintain logs of which data is processed by what appliance
Beware of security vulnerabilities – the only way to apply patches is to restart the build, test, operate cycle!
* Want maximum “operate” for minimum “build + test”
* Key is to prove an appliance is equivalent to one that was positively validated
  * Identical hardware – qualify each unit, but build & test only once then mass deploy?
Generic validation of a tool is not possible as behaviour depends on OS correctness & configuration.

Validation tests must take into account all software & hardware factors that may influence outcome.

Necessary to obtain maximum “operation” time for minimum “build+test”.

Construction of “forensic appliances” using open source software is a convenient way to achieve this goal.