Validating a hardware write blocker

(or a similar device)

Issues

- An unsafe command* can pass-through to a drive
- An unsafe command* can be issued to a drive by a write blocker (on its own)
- A safe command can be blocked or ignored
- A device can misbehave when a faulty sector is read
- Other issues: large drives (> 2 TB), 4096-byte sectors

* – unlocking a data area protected with HPA/DCO is considered safe

Examples

AgeStar 3FBCP (firmware: unknown version), PATA/SATA-to-USB bridge
Unsafe ATA pass-through commands are not filtered in the read-only mode

Tableau TD3 (firmware: 2.0.0), forensic imager & network-based write blocker
The device writes to a drive attached to a «write blocked» port when a specific state of the Ext4 file system is encountered (no command from a host is required)

Tableau T356789iu (firmware: 1.3.0), forensic bridge
The device is blocking a host from reading sectors near the unreadable one (a read error is returned to a host even if a sector can be read: a single unreadable sector results in 128 sectors being reported as unreadable to a host)

Multiple write blockers
The device requires a USB reset when an unreadable sector is encountered

Validation directions*

Verifying that drives with faulty sectors work as expected

Run a test against a drive with an unreadable sector

Does a block device disappear in Linux? Does it appear again (later)? What about Windows?

* – repeat for each interface to a host used

Checking the issues with large drives (> 2 TB), 4096-byte sectors

- Run a test against a large (> 2 TB) drive — Are all sectors visible and readable?
- Run a test against a 4Kn drive — What is the reported sector size (as seen by a host)?

Validation directions

Sending unsafe commands and verifying execution results

- Try to write or discard (e.g. TRIM) something and check the result
- Things to consider
  - Non-typical commands: WRITE AND VERIFY (32), etc.
    - SCSI UNMAP
    - ATA PASS-THROUGH
    - Vendor-specific commands
  - Running tests against HDDs, SSDs, other types of media
  - Running tests against a drive with an unreadable sector — Is it possible to reset the write protection after trying to read an unreadable sector (if a drive is revalidated by the firmware)?
  - Rebooting a write blocker before checking if a sector has been modified — A write blocker can cache the original version of data in the «read-write-read-compare» scenario* (if a write command does not invalidate the cache)
  - Direct IO or raw read commands — An operating system can cache the original version of data in the «read-write-read-compare» scenario* (for example, when there is a mounted file system)

Testing a USB bridge?

The «UNMAP-to-TRIM» translation for SSDs

Can be used by third-party software to issue TRIM commands for SSDs and USB sticks

Not a mistake!

Checking if unsafe commands can be issued without a command from a host

- Things to consider
  - Extracting and reverse-engineering the firmware to locate weak spots
  - Running file system tests specific to Linux (if a write blocker mounts a file system) — Linux-based firmware only

Checking if safe commands work well

- Things to consider
  - Try to read all sectors and check the result
  - Include a data area hidden with HPO/DCO — If such a data area is not exposed by a write blocker, send an «unlocking» command from a host
  - Running tests against a drive with an unreadable sector — Checking the error granularity of a write blocker: are readable sectors reported (to a host) as unreadable?
  - Issuing non-typical safe commands — This is similar to existing issues with live forensic distributions

* – the «read-write-read-compare» scenario: read a sector, [try to] write something different to that sector, read the sector again, compare the data read

Would be nice to know...

1. How does a write blocker handle sectors changed by a drive itself? If such a sector is not changed, as observed by a host, then there is a cache. *Examples of such drives:*

- Old SSDs (file system-aware garbage collection);
- Old USB Flash drives (non-deterministic sectors).

2. Is it possible to boot a write blocker from attached suspect storage?

Example. The firmware is located on an SD card, but there is another slot for a suspect SD card. Is it possible to trick the boot loader, or the initial RAM file system, or something else into running the code from a suspect SD card?

Version: 01 (2018-04-25)