Coffee forensics — Reconstructing data in IoT devices running Contiki OS

By:
Jens-Petter Sandvik (National Criminal Investigation Service (Kripos) and NTNU), Katrin Franke (Norwegian University of Science and Technology (NTNU), Habtamu Abie (Norwegian Computing Centre), and Andre Årnes (NTNU and Telenor Group)

From the proceedings of
The Digital Forensic Research Conference
DFRWS USA 2021
July 12-15, 2021

NTNU | Norwegian University of Science and Technology

COFFEE FORENSICS

Reconstructing data in IoT devices running Contiki OS

Jens-Petter Sandvik, Katrine Franke, Habtamu Abie, Andre Årnes

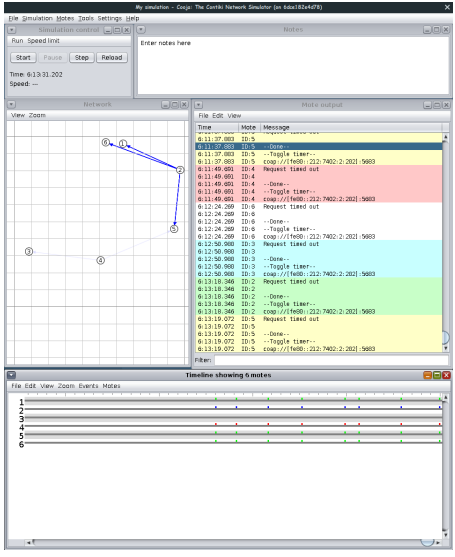
2021-07-12 – 2021-07-14

Introduction - Background

- IoT systems includes a variety of subsystems
  - Focus on resource constrained devices
- Many different operating systems and file systems
  - Linux, Windows, FreeRTOS, Contiki, QNX, ARM MBed OS, TinyOS, RiotOS, VxWorks, etc.
  - FAT, ext4, YAFFS, Reliance Edge/Nitro, Coffee, HRFS, LittleFS, etc.
- Contiki-NG, based on Contiki
  - Minimum requirements: 10 kB RAM, 100 kB ROM
  - 6LoWPAN, CoAP, MQTT, RPL
  - …but not Matter/Thread
  - Used in the Thingsquare platform
  - Have been used by 5% of IoT developers

Introduction - Cooja

- A network simulator
- A device emulator
- Simple to set up using docker image
- Log status messages from nodes
- Can change network during simulation



Coffee File System - Coffee File System

- Designed for flash memory
- Low resource usage
- No $MFT, FAT table or Superblock
- Have to scan all pages to build up internal view of FS
- File headers contain all information
- FS structures held in RAM, not written to disk
- No folders
- File name is unique identifier

Coffee File System - Coffee File System nomenclature

- File system consist of **sectors** which consist of **pages**
- A whole page must be written, can only **program** bits from “1” to “0”
- A whole sector must be **erased**, reset all bits to “1”
  - Done during **garbage collection**

Active: Pages that have been allocated and belong to existing files.

Obsolete: Pages that have been active but currently not in use. They are available for garbage collection.

Isolated: Pages that have been active and belonging to a file starting in the previous sector. They exist as file fragments without a header at the start of a sector.

Deleted: Pages that are obsolete or isolated.

Unused: Pages that have not been written to after being erased by the garbage collector.

Coffee File System - Coffee File System structures

Size (bytes)	Field
2	LOG_PAGE
2	LOG_RECORDS
2	LOG_RECORD_SIZE
2	MAX_PAGES
1	DEPRECATED_EOF_HINT
1	FLAGS
16/40	NAME

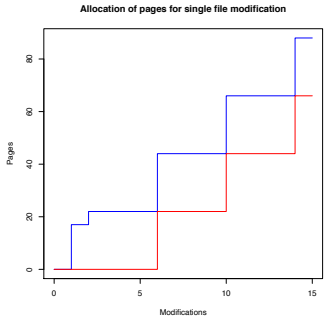
Value	Flag
0x01	HDR_FLAG_VALID
0x02	HDR_FLAG_ALLOCATED
0x04	HDR_FLAG_OBSOLETE
0x08	HDR_FLAG_MODIFIED
0x10	HDR_FLAG_LOG
0x20	HDR_FLAG_ISOLATED

```
00032600  37 02 00 00 00 00 11 00 00 0f 66 69 6c 65 30 30  7.........file00
00032610  33 2e 74 78 74 00 00 00 00 00 46 69 6c 65 33 20  3.txt.....File3

00033700  00 00 00 00 00 00 05 00 00 17 66 69 6c 65 30 30  ..........file00
00033710  33 2e 74 78 74 00 00 00 00 00 05 00 05 00 03 00  3.txt...........
00033720  03 00 46 69 6c 65 33 20 20 42 23 30 30 30 30 30  ..File3  B#00000
```

Coffee File System - File creation and modification

- Upon creation 17 pages are allocated for file
- Appends just added to end, as no already written bits are changed
- File size doubled when in need of more pages
- Any other modification adds a 5-page logfile
  - Contain a list indicating which pages are modified
  - A list of the modified pages
- New file/logfile created when log is full
  - Last version copied to new file, changes to log



Coffee File System - Coffee file system dynamics

- Files are written end-to-end
- Extended wear-leveling
- When allocation fails, garbage collection starts
  - No consecutive pages to allocate
- No reshuffling of pages to free up more sectors
- Deleted pages in sectors with live files survive
- Pages belonging to deleted files at start of sector are “isolated pages”
  - No information about header

Coffee File System - File system tests

- File operations tested in four tests: two for file writing and two for GC
- File writing testing file creation, deletion, appends and changes
  - Appends don't cause any log file to be written
  - Exceeding allocated length doubles allocation size
  - One writing operation to one page often caused two writes to log file
  - Last version of a base+log file compound is copied to the next base file
- Garbage collection
  - Garbage collection starts when file reaches end of file system
  - No sorting and rearranging of pages
  - When all sectors contain active pages, FS can't free up any sectors
  - Huge files spanning more than one sector no different from small files

Recreating file version history - File version history

- Files in same sector is written consecutively
- Files in different sectors can be from different GC runs
- Two approaches:
  - File content differences
  - Sector offset analysis
- Both methods have their limitations

Recreating file version history - File offset analysis

- Files are written consecutively, sector size is in general not divisible by the file size
- Assumption that all files have the same length
- A sequence of offsets for the first file in each consecutive sectors

Offset of first file

$$A_0 = 0 \tag{1}$$

$$A_n = \left[\left(A_{n-1} + \left\lceil \frac{S_S}{S_F} \right\rceil \right) \pmod{S_S} \right] \pmod{S_F} \tag{2}$$

- Setting S_S to 256 and F_S to 22:
  $0 \to 8 \to 16 \to 2 \to 10 \to 18 \to 4 \to 12 \to 20 \to 6 \to 14 \to 0$

Recreating file version history - File offset analysis results

- Created a mote that wrote two files: 1 s/w, 100 s/w
- Worked for most part…
- After GC, offset start at 0
- Assumption on same file size don't hold
- If file size is even, why an odd offset?

Starting file offsets

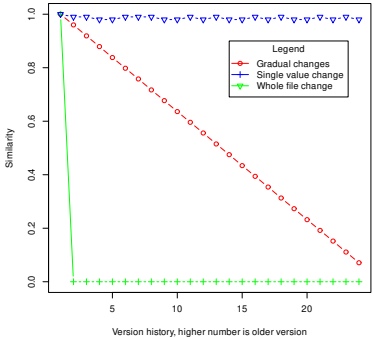
Sector	First file
0x00	0x00
0x01	0x08
0x06	0x04
0x0d	0x01

Recreating file version history - File content analysis

- For gradually changing files
  - Ring buffers
- Not good for files where the same part(s) of the file changes every time
- Several existing diff algorithms
  - Levenstein distance/ Edit distance
  - Myers' diff algorithm
- Knowledge about the writing pattern for selecting comparison algorithm
- In this study we used Myers diff algorithm

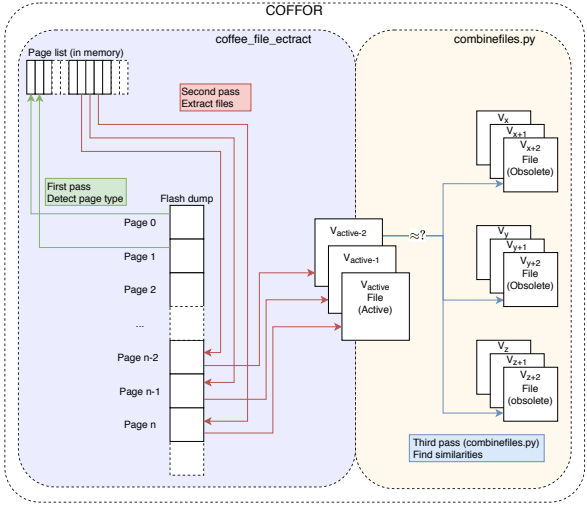
Recreating file version history - File content analysis results

- Testing 3 types of file changes between versions:
  1. New location changed
  2. A single value changed
  3. Whole file changed
- Only the gradually changed file can be reconstructed
- Ordering not dependent on distance between versions
  - …Until whole file has changed



COFFOR - COFFOR

- **COF**fee **FOR**ensics
- 2 passes:
  1. Page type detection
     - Active/del base/log
     - File fragment
     - Zero-page
     - Isolated page
  2. File extraction
     - name_A/D+page_ver
- Comparing files:
  - Comparing last versions w/ given version
  - Radare2's radiff



COFFOR - Results

- All pages are correctly classified by COFFOR
- The file versions are correctly saved
- Should be tested on a wider variety of flash images
- radiff works well
- But only for a subset of files

Final thoughts - Conclusion and way forward

- Wrap-up
  - The artifacts of the Coffee File System
  - File version history ordering
  - COFFOR tool
- Do we really need to collect data from the devices?
- Can the version history be used for other types of data?
- `https://github.com/jenspets/coffor`

Thank you for your attention