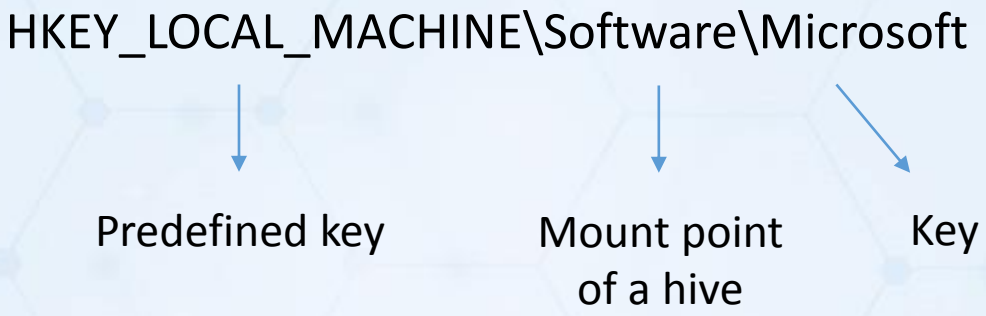
BI.ZONE

In-depth forensic analysis of Windows registry files

Maxim Suhanov

Registry basics

A typical registry path:



This path is different in a kernel:

- \Registry\Machine\Software\Microsoft

Some hives do not have a visible mount point!

This hive is non-volatile (stored on a disk), backing files:

- C:\Windows\System32\config\SOFTWARE
- C:\Windows\System32\config\SOFTWARE.LOG (+ .LOG1/.LOG2)

Hive format (NT family)

Seven versions of the format:

from 1.0 (pre-release versions of Windows NT 3.1)

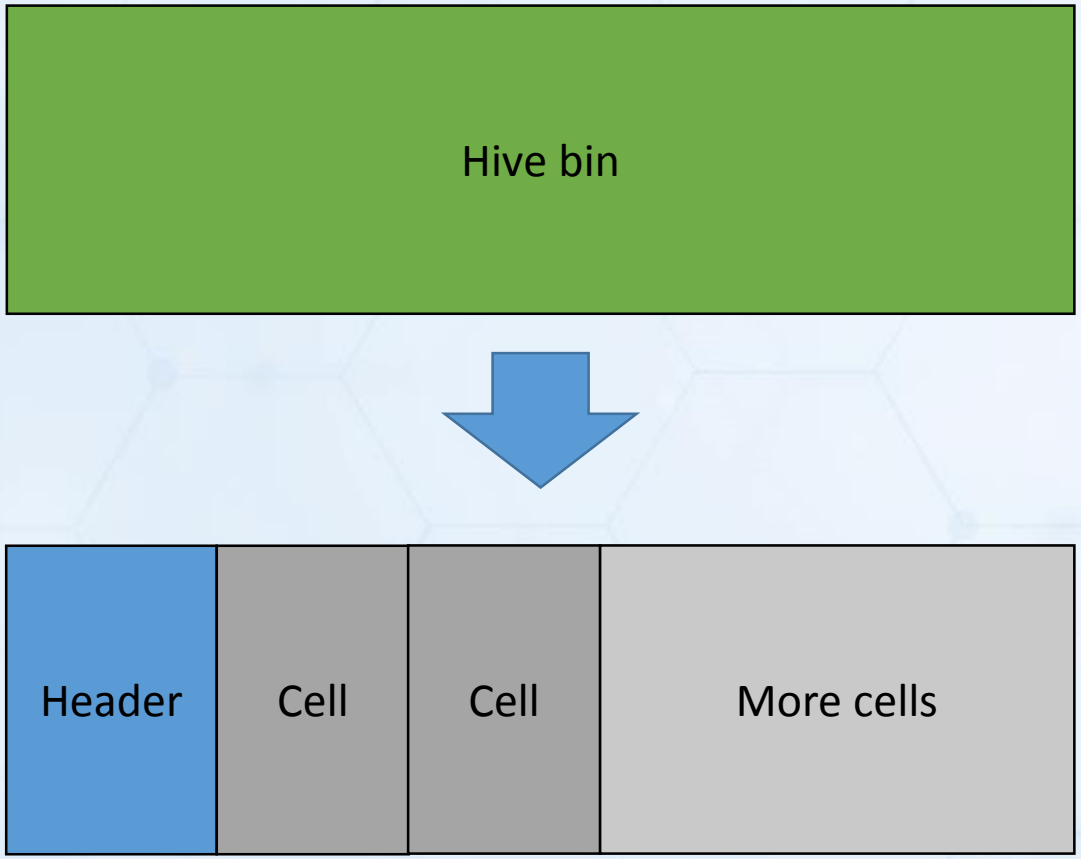
to 1.6 (introduced in Windows 10 “Redstone 1”, not used yet)

The structure of a hive file:

The base block contains the size of all [allocated] hive bins.

Hive format (NT family)

A hive bin contains a header and cells.



The header contains the size of this hive bin.

A cell may contain:

- Key node (nk)
- Key value (vk)
- Key security (sk)
- List of subkeys (li, lf, lh)
- List of subkeys lists (ri)
- List of values
- Value data
- Big data records (db)
- List of segments
- Value data segments

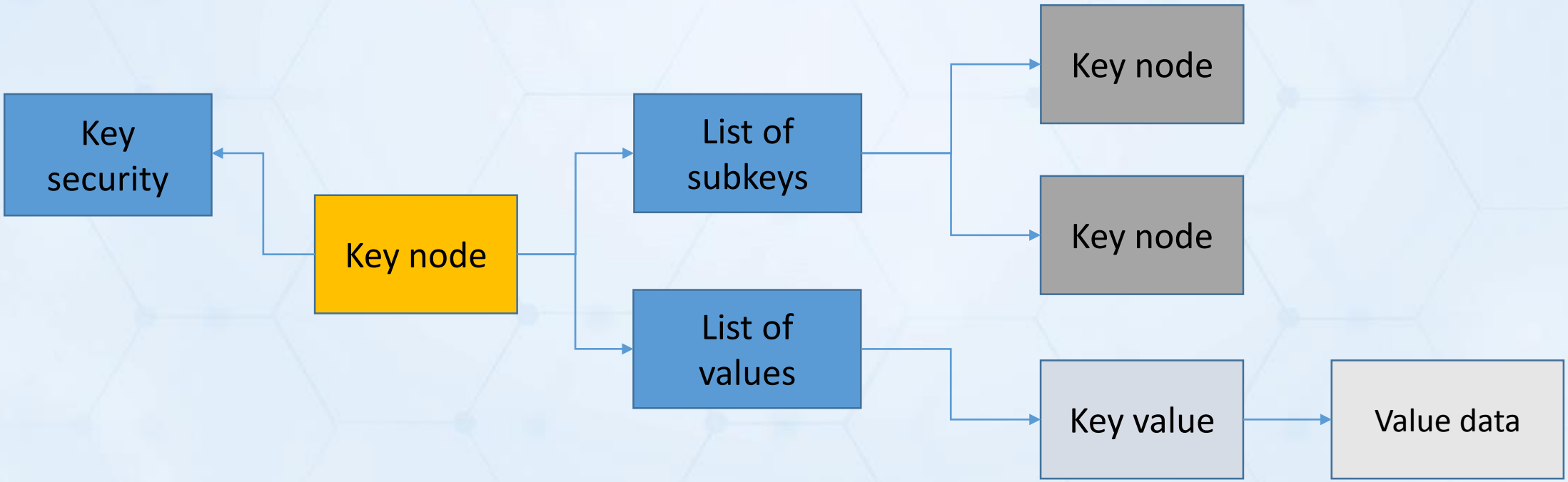
The first 4 bytes of a cell are used to record the size of this cell (positive value: unallocated cell, negative value: allocated cell)

Hive format (NT family)

Records (entities) point to other records (entities) using a relative offset of a cell. A base block points to a root key node.

File offset of a cell = Length of a base block + Relative offset of a cell

File offset of a cell = 4096 + Relative offset of a cell



Hive format (NT family)

All registry structures are documented here:

https://github.com/msuhanov/regf/

Transaction log files (NT family)

- Before writing dirty data to a primary file, this data is stored in a transaction log file.
- If a system crash occurs when writing to a transaction log file, a primary file will be intact.
- If a system crash occurs when writing to a primary file, a transaction log file will be used to repeat the write operation.

The old format (before Windows 8.1):

Transaction log files (NT family)

- Every bit of the bitmap corresponds to a single 512-byte sector of the hive bins data in a primary file.
- If set, a sector is dirty (and modified contents are in the transaction log file).

Relative offset of a dirty sector in a transaction log file = Index of a bit in the bitmap * 512

Zero-based, unset bits not counted.

Relative from the start of dirty sectors (pages).

As of Windows XP, a single run of dirty data is not smaller than a page (4096 bytes).

Transaction log files (NT family)

- A bad sector in a primary file results in the following:
  - Dirty data cannot be written to a primary file (to the location with a bad sector).
  - We cannot overwrite dirty data in an existing transaction log file (otherwise a system crash may leave us without a good copy of dirty data).
  - We cannot mark mounted hives as read-only.
  - No further writes (after a failed one) to a primary file will be allowed (new dirty data will be discarded).

Transaction log files (NT family)

- Solution: the dual-logging scheme (starting from Windows Vista).
  - After a failed write to a primary file, switch the log file being used (.LOG1 -> .LOG2, .LOG2 -> .LOG1), and try the write operation again.
  - Repeat this until a bad sector is gone.

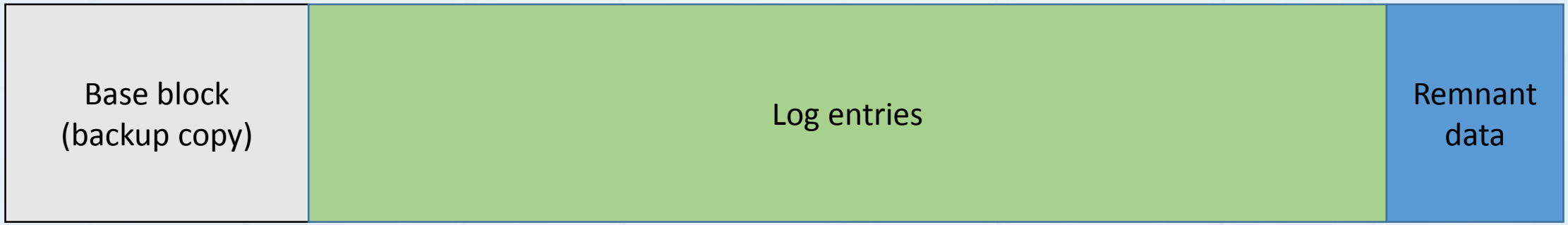
Microsoft left the CmpFailPrimarySave kernel variable used to simulate failed writes!

Transaction log files (NT family)

- The old format requires dirty data to be written to a disk twice.
- The new format is used to stash dirty pages in a transaction log file without writing them to a primary file. These pages will be written to a primary file later.

  - When all users are inactive.
  - During the full shutdown.
  - After 3600 seconds since the latest write.

The new format (as of Windows 8.1):



Also, each log entry has a checksum for dirty data (Marvin32).

Deleted data (NT family)

- When a key or a value is deleted, all corresponding cells are marked as unallocated.
- A cell will be coalesced with an adjacent unallocated cell.
- A single unallocated cell may contain multiple deleted records (entities).

Deleted data (NT family)

Well, not all cells are marked as unallocated when a corresponding record (entity) is deleted…

- In recent releases of Windows 10, renaming a key will leave an old key node in an allocated cell.
- This key node will be present until the hive is defragmented.

Thus,

Cells with deleted data = All cells – Referenced cells

Deleted data (NT family)

Also, deleted records (entities) can be found in:

- Remnant data at the end of a primary file.
- Slack space of allocated and referenced records (especially, in cells with subkeys/values lists).
- Transaction log files (old log entries, remnant data, gaps).

Preallocated space.

Distribution of recoverable deleted keys and values in primary files:

- Unallocated cells: 97.9%
- Remnant data at the end of a file: 0.6%
- Slack space in allocated and referenced cells: 1.5%
- Allocated but unreferenced cells: 0% (only 1 key was found)

Caveats

- Most registry viewers ignore data in transaction log files.
  - *Registry Explorer, Windows Registry Recovery, Registry Recon, libregf, reglookup.*
  - Latest changes to registry keys and values (made in Windows 8.1 and later versions of Windows) may be invisible to such tools.
  - Malicious programs can hide data from offline registry viewers by manipulating the CmpFailPrimarySave kernel variable (modified keys and values will be stored outside of a primary file).

Example:
A laptop with a USB cellular modem, building the timeline for the Software hive, looking for timestamps related to the activity of the modem.

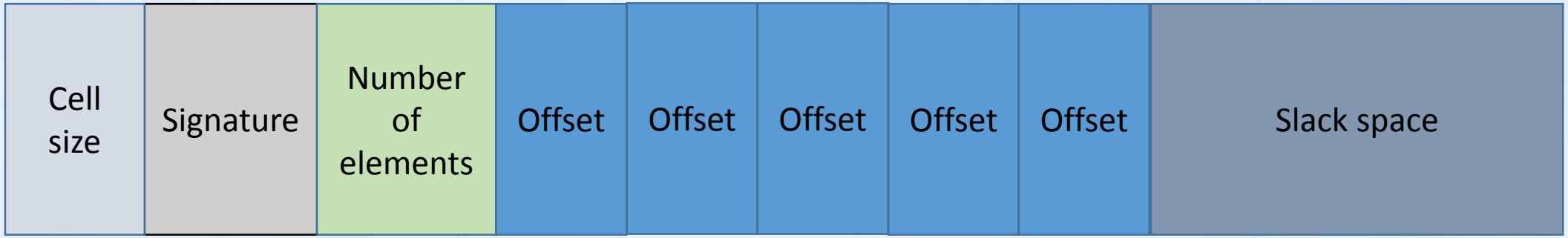
With transaction log files: 13 different key modification timestamps were found for a single registry key.
Without transaction log files: 1 timestamp for that key.

Caveats

- Offline registry libraries in antivirus software ignore transaction log files too.
  - Kaspersky Rescue Disk 10 (based on Linux) will delete malicious keys/values from a primary file only, without applying dirty data from a transaction log file.
  - When running Windows 8.1 and later versions of Windows, it is possible to create a malicious autorun entry that will not be deleted when performing an antivirus scan from Kaspersky Rescue Disk 10.
  - The same is also possible with older versions of Windows by exploiting the CmpFailPrimarySave kernel variable.

Caveats

- When recovering deleted data from primary files, many tools skip the slack space of allocated records (entities).
- 1.5% of recoverable keys and values are not recovered!



A cell with a subkeys list

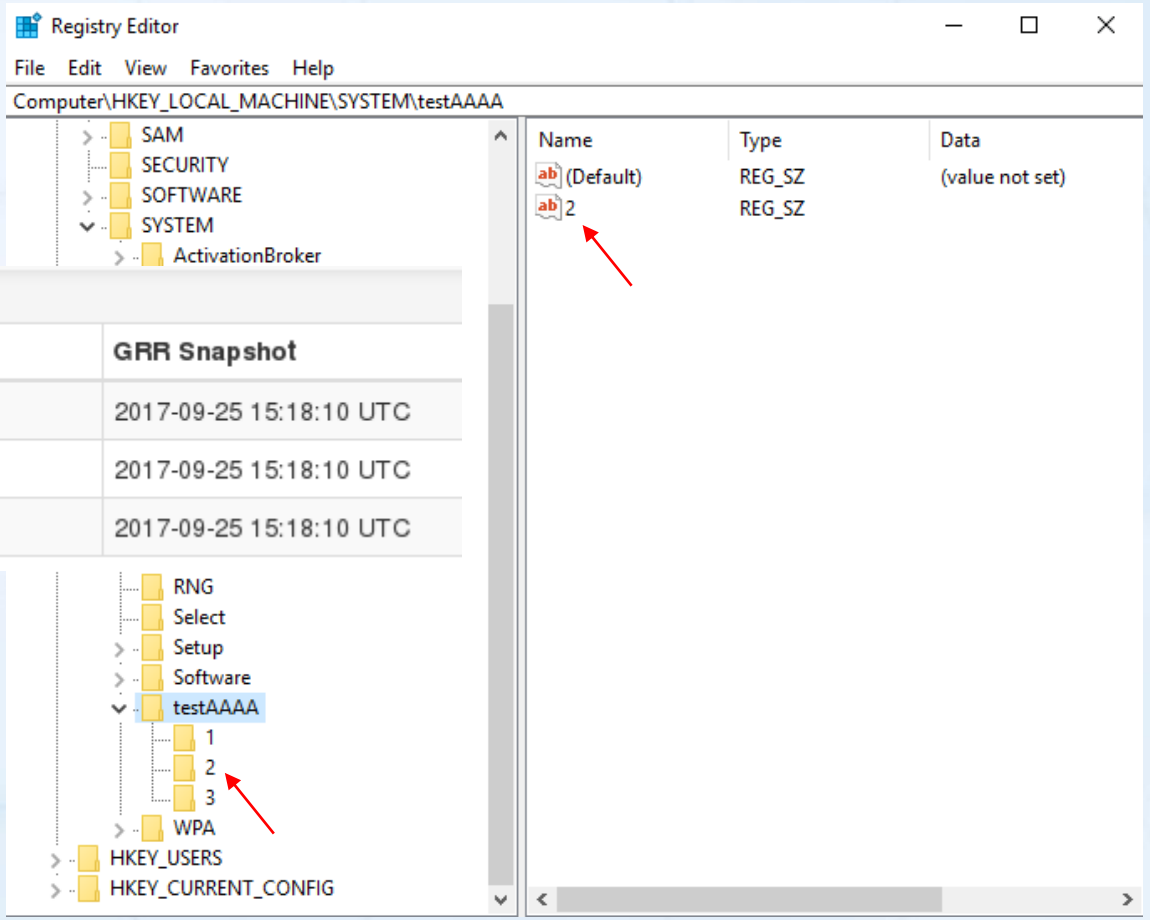
Caveats

- Treating the registry as a typical file system is dangerous too!
  - A subkey and a value of a single key can share the same name.
  - A name of a key or a value could be “.” or “..”.
  - Also: null byte, “/”, and “\”.

registry > HKEY_LOCAL_MACHINE > SYSTEM > testAAAA

Icon	Name	st_size	st_mtime	st_ctime	GRR Snapshot
	1	4			2017-09-25 15:18:10 UTC
	2	0			2017-09-25 15:18:10 UTC
	3	4			2017-09-25 15:18:10 UTC

GRR: a value shadows a key with the same name



Yet another registry parser (yarp)

https://github.com/msuhanov/yarp/

(library & tools, Python 3)

- Parse Windows registry files in a proper way (with forensics in mind).
- Expose values of all fields of underlying registry structures.
- Support for truncated registry files and registry fragments.
- Support for recovering deleted keys and values.
- Support for carving of registry hives.
- Support for transaction log files.

?