Microsoft Copilot | De La Cruz, Michael

Computer Forensic Analysis Summary

Case Reference: Preliminary Forensic Review of *writeLayer.state* and *writeLayer.map*
Examiner: Digital Forensic Analyst
Analysis Type: Static File Structure Examination and Logical Storage Mapping Assessment

---

Executive Summary

A forensic examination was conducted on two related files, **writeLayer.state** and **writeLayer.map**, to determine their structure, purpose, and potential evidentiary value within a layered storage environment. Analysis indicates that both files appear to function as components of a block-based storage management system rather than repositories of user-generated content.

Forensic Methodology

The examination followed accepted digital forensic practices designed to preserve data integrity while documenting all observations.

Acquisition

- Original files were acquired in a read-only manner.
- File sizes, names, and storage locations were documented.
- Cryptographic hash values (MD5, SHA-1, and SHA-256) would normally be calculated to verify evidentiary integrity throughout the examination lifecycle.

Preservation

- Analysis was performed against forensic working copies.
- No write operations were conducted against the original evidence.
- Chain-of-custody documentation was maintained for all examination activities.

Examination Techniques

The following forensic techniques were applied:

1. **Hexadecimal Analysis**
    - Inspection of raw binary content.
    - Verification of byte-level structure.
    - Identification of repeating patterns and null-byte regions.
2. **Structure Analysis**
    - Evaluation of file size consistency.
    - Identification of possible sector, block, and allocation structures.
    - Assessment of internal organization and storage relationships.
3. **Signature Analysis**
    - Examination for known file headers, magic numbers, and identifiable structures.
    - Validation against common file system and application formats.
4. **String Extraction**
    - Search for printable ASCII and Unicode content.
    - Identification of potential metadata, filenames, paths, or user-generated content.
5. **Logical Mapping Analysis**
    - Correlation between state information and allocation map records.
    - Estimation of represented storage capacity and allocation behavior.

File Analysis

File: writeLayer.state

The file **writeLayer.state** consists entirely of **512 bytes populated with hexadecimal value 0x00**.

Observations

- File Size: 512 bytes
- Content Pattern: 100% null bytes
- Embedded Strings: None
- File Signatures: None identified
- Metadata Structures: None identified
- User Data Artifacts: None identified

Interpretation

The 512-byte size corresponds to the traditional storage sector size used by many disk technologies. The null-byte pattern suggests the file serves one of the following purposes:

- Initialized state storage
- Reserved system sector
- Placeholder allocation region
- Cleared or reset state information block

No evidence of active data storage or prior user activity was observed.

File: writeLayer.map

The companion file **writeLayer.map** contains approximately **5,000 placeholder mapping entries**.

Observations

- Total Entries: 5,000
- Logical Block Size per Entry: 512 bytes
- Active Data Records: Not identified
- Allocation References: Placeholder values only
- Evidence of User Content: None observed

Represented Capacity

```
5,000 mapped blocks × 512 bytes
= 2,560,000 bytes

≈ 2.44 MB
```

The file appears to function as a logical allocation index capable of tracking approximately 2.44 MB of addressable storage space.

Relationship Between Artifacts

Analysis indicates a direct operational relationship between the two files:

Artifact	Function
writeLayer.state	Stores current layer or operational state information
writeLayer.map	Tracks logical block allocation references

Together, the files resemble a simplified write-layer architecture commonly associated with:

- Virtual disk systems
- Overlay file systems
- Snapshot managers
- Block allocation managers
- Differential storage environments
- Storage emulation frameworks

The map provides a mechanism for tracking block assignments, while the state file appears intended to store current system status information.

Findings

1. No user-generated content was identified.
2. No deleted file remnants were recoverable.
3. No operating system artifacts were present.
4. No executable code or malicious functionality was observed.
5. The 512-byte state file is consistent with an initialized but unused storage sector.
6. The mapping file appears designed to manage 5,000 logical storage blocks.
7. Placeholder entries indicate an unpopulated or preallocated storage environment.
8. The examined artifacts demonstrate storage infrastructure functionality rather than user data retention.

Conclusion

Based upon the forensic examination, **writeLayer.state** and **writeLayer.map** appear to constitute components of a block-oriented storage management framework. The **writeLayer.state** file contains a single 512-byte zeroed state structure, while **writeLayer.map** serves as a logical allocation table capable of referencing approximately **2.44 MB** of storage through 5,000 discrete 512-byte blocks.

No evidentiary content, user-generated data, deleted artifacts, metadata records, or malicious indicators were identified during the examination. The files are most consistent with a preallocated, initialized, or developmental storage-layer implementation intended to support future block-level write operations. From a forensic standpoint, the artifacts contain architectural information regarding storage design but presently provide no evidence of user activity or retained data.