Hash Tree Interchange Format

The Whys and Wherefores

Chuck Lever <chuck.lever@oracle.com>

What’s Going On

Code signing, or more generally, file content attestation

- To protect file content end-to-end:
  - Attestation metadata must be created and signed just after file content is generated
  - File content must be verified just before it is used
  - The format of the attestation metadata must be independent of storage media, parse-able everywhere, and unencumbered
    - That is, it must be a standard!

Creating Attestation Metadata

The easy part

- A digest is generated on the file content
- The digest is cryptographically signed
- The signed digest is distributed with the file content

Verifying File Content

The challenging part

- The end-user's file system must use the attestation metadata to verify file content before presenting it to applications
- Applications typically read a file in small pieces (say, via `read(2)`)
- The entire file must be read into memory to verify any part of it. That makes a linear digest inefficient for verifying small portions of a file.
- Further, memory management can reclaim portions of a file not recently used, meaning the next verifying read must read the entire file again

Verifying File Content

Solving the issue

- A tree of digests enables the efficient verification of portions of a file
- However, hash trees can get large. Not all storage mechanisms have the flexibility to store boundless amounts of file metadata.
  - Legacy filesystems and storage protocols
  - Data backup
  - Software distribution schemes

First Proposal: Data Reduction

- Instead of durably storing the whole tree, store (and sign) just the root hash.
- When installing a file for use on an end system, reconstitute the tree using its root hash and the file content
  - The reconstituted tree can be maintained locally, if possible
  - Otherwise it can be cached in memory on demand

Second Proposal: Standard Format

- We want to store the metadata in a widely supported data representation format
- We want to support a broad set of digest algorithms
- Therefore, use an X.509v3 certificate
    - DER encoding
    - Standardized set of available digest algorithms
    - A cryptographic signature protects the whole thing

Technical Discussion

- Has this been done before? Let’s not duplicate it.
- Should it support ADT shapes other than binary trees?
- Is there a better approach than a Merkle tree?
- How should it handle second pre-image attacks?
  - Currently the format stores the tree height, but it might support prefixing digest values on internal nodes

Supplemental Material

Bibliography

- https://datatracker.ietf.org/doc/draft-cel-nfsv4-hash-tree-interchange-format/