Advanced Computer Forensics

Ruiyang Liu 85C576

Conference Paper Summary - Group #1

Paper selected: md5bloom - Forensic filesystem hashing revisited

Paper author: Vassil Roussev*, Yixin Chen, Timothy Bourg, Golden G. Richard III

Word count: 537 words

By comparing the hash value, one can identify known data and verify the integrity of significant data. Hashing can assist us in answering yes-no questions and filtering out the hash values we require in the computer file system. To increase the efficiency and accuracy of the forensic filesystem, the author of this paper presents md5bloom, a bloom filter for forensics.

In forensics, similarity discovery and fine-grained change detection are hot subjects. Multiple target correlations may overwhelm the workstation's capability. As a result, a well-designed and highly efficient algorithm is required to address this problem. Additionally, detecting fine-grained changes is difficult. Because the reference files are not static, they may become out of date very fast. The bloom filter replaces the typical single hash with a collection of hashes, which aids in resolving scaling concerns. Alternatively, the bloom filter can be applied to big data sets.

The author introduces md5bloom in this work with the goal of developing an open-source tool for digital forensics. Md5bloom offers three filter functions: query evaluation, filter generation, and filter comparison. Users can directly control it via the command line. Additionally, the author conducts various experiments to validate the md5bloom. These studies include determining the rate of MD5 false positives, comparing bit ratios to PDFs, identifying hash manipulation, and detecting object versioning.

The author generates streams of data from various devices. Then he uses the command to construct a filter from the random devices' blocks. Another sanity check is to ensure that the recognition rate is 100% for the same output.

MBR (matching bit ratio): Bloom filters with two random files are predicted to have a matching bit ratio of approximately 33%. And the filter achieves a 45 percent match rate with only a 20% overlap in the original hash files. The amount of matched bits and the number of overlapping records are linearly related.

Detecting hash tampering: The author provides simulation results to help explain the system, confirming the conjecture that the only way to see a matching bits number to the left of the bell curve is to introduce non-randomness purposefully. The author experimented with two approaches: increasing the number of bits set to one and increasing the number of bits set to zero.

Detecting object versioning: The author chooses three pairs of Linux libraries at random that have names that imply version connections. Following that, the author prepares MD5 for each .text file in the libraries and compares them.

To summarize, the author enhances the use of Bloom filters by creating a program called md5bloom. Additionally, the author describes and interprets the first probabilistic framework and conducts simulations to validate the framework using hash sets. Additionally, the author conducts three different experiments to demonstrate the utility of bloom filters. At the conclusion of the article, the author discusses his future work on md5bloom. md5bloom beta will be distributed as an open-source program in the near future. Furthermore, additional bloom filters will be made available and supported in the future.