Measuring the effectiveness of improved versions of sdhash similarity tool
Vitor Hugo Galhardo Moia, Marco Aurélio Amaral Henriques
{vhgmoia,marco@dca.fee.unicamp.br}

Abstract – The use of Approximate Matching (AM) functions on digital forensic investigations allow practitioners to find similar data in an efficient manner by using digests, compact representations of objects. By comparing the object digests, a score related to the level of similarity is produced. Many functions were developed over years, but one that stood out and have been the target of constant research is sdhash. In this work, we show that the open source implementation of sdhash present some bugs and issues in the digest generation process. We fixed the bugs and evaluate the impact of them in the similarity assessment. Moreover, we show that standard evaluation metrics (such as recall and precision) were not enough to highlight the benefits of the fixes, and a new metric is necessary. We finish our paper by proposing a new way of assessing the score quality of AM tools and present future work.

Keywords – Digital Forensics, Approximate matching functions, sdhash, known file filtering.

1. Introduction

Finding similar data is a hard task especially when dealing with a corpus with millions of objects. Digital forensics is one of the fields that can benefit from having new advances in figuring out how to locate similar objects in an efficient manner. Forensic practitioners dealing with even ordinary investigations has to go through lots of gigabytes of data in a limited amount of time, where besides the huge amount of data, many other issues arise, such as data heterogeneity, operating systems diversity, among others.

Solutions to identify duplicates (identical objects) are straightforward, such as the use of hash functions. By comparing the hashes of two files, one can know whether or not the objects are identical. However, hash functions are not suitable for dealing with similar content; a single bit flipped in one of the input object will change almost completely its hash. In digital forensic investigations where practitioners deal with objects that constantly change (software updates, new versions of malware, reviewed documents, etc), hashes are inefficient.

In this scenario, Approximate Matching (AM) appears as a solution for efficiently identify similarity; by creating and later comparing compact representations of objects (digest), a score is produced indicating the level of similarity, usually in the range of 0 (dissimilar) to 100 (identical or very similar). One of the most popular AM tools and target of constant research is sdhash [7]. In this work, we will focus our study in the sdhash tool for being one of the most popular tools and target of constant research because of its good detection capabilities.

2. Background

2.1. Approximate matching functions

NIST [1] defines AM functions as "a promising technology designed to identify similarities between two digital artifacts. It is used to find objects that resemble each other or to find objects that are contained in another object". This sort of functions create compact representation for objects and by comparing them, estimate the level of similarity shared between two objects. Many tools were developed in the past years, such as ssdeep [3], sdahah [7], TLSH [5], LZJD [6], among others. In this work, we will focus our study in the sdhash tool for being one of the most popular tools and target of constant research because of its good detection capabilities.

2.2. The sdhash approximate matching tool

The main purpose of sdhash is to identify statistically improbable features (unique ones) on objects to represent them. A feature is this context is a byte sequence (64 bytes) extracted from the object bytes. We can summarize the digest generation process of sdhash in three steps.

**Feature extraction:** All features are extracted from the object and have their Shannon entropy computed.

**Feature filtering / Selection:** Weak features (e.g., sequences of the same byte - 0x00, 0xFF) are removed from consideration and the ones that seem unique (based on the entropy score) are chosen to represent the object.

**Feature encoding:** Each feature chosen in the last step have is stored into fixed-size Bloom filters. Each filter holds a maximum of \( f_{max} \) features (160 by default); when a filter reaches its capacity, a new one is created. The final digest of a file is a sequence of Bloom Filters.

After creating the digest of a given object, one can compare the digest of two files to find out their similarity. The sdhash tool produces a score ranging from 0 (dissimilar) to 100 (identical or very similar). According to its inventor, sdhash score should not be interpreted as a percentage of similarity. They provide a scale to help one
3. Analyzing sdhash source code

By trying to better understand the working process of sdhash, we found out some bugs/issues in the source code released in the GitHub of sdhash latest version (v.3.4) related to the digest generation process.

We addressed all bugs/issues and presented a detailed description of them, highlighting their impact and how to fix them in our GitHub page: https://github.com/regras/AMF_sdhash. Besides, we compiled two new versions of sdhash with the bugs fixed; the two binaries differ in the bugs addressed. Tab. 1 shows the two versions with the list of bugs/issues fixed. We called them e-sdhash (enhanced sdhash) and e-sdhashCW (enhanced sdhash with circular window).

4. Initial evaluation

In this section, we present an initial attempt to measure the impact of sdhash bugs in the similarity by comparing it with its fixed version.

4.1. Data set

The data set chosen for the experiments was t5-corpus\(^3\). This is a set with 4457 objects (1.78 GiB) of different file types. We split it into two sets to simulate a digital forensics investigation, as presented in Tab. 2.

4.2. Metrics

To measure the impact of our modifications on the similarity assessment, we used standards metrics of information retrieval field. We choose recall and precision to this end, defined below:

\[
\text{recall} = \frac{tp}{tp + fn} \quad \text{and} \quad \text{precision} = \frac{tp}{tp + fp}
\]

Here, \(tp\) (true positive) is the number similar files found, \(tn\) (true negative) the number of similar files missed, and \(fp\) (false positive) the number of matches found by the tool which are not similar. In this work, we will adopt the same approach as Moia, V.H.G et al. [4] and consider as true positives the matches related to user-generated and template content, as well as false positive the ones related to application-generated content.

4.3. Results

Fig. 1 shows the initial results for sdhash and the fixed versions of it for recall and precision. Note that for recall, we had no significant improvements of the fixed versions in relation to the original one. Looking at precision, we see that the original sdhash outperformed the fixed versions, the opposite behavior that we expected; fixing the implementation bugs/issues had an overall bad performance than the original version.

4.4. Understanding what went wrong

Fixing the feature extraction and selection process of sdhash lead us to believe that with a more accurate tool, we would have a better performance with respect to recall and precision. However, we observed the opposite (at least for precision). To understand why the fixed versions had a poor performance on precision, we need to look at the features selected by the tools. It turns out that many features selected to compose the digest of the original sdhash are not the same as the ones selected by the fixed versions. Tab. 3 shows this difference by presenting some files from our data set and the amount of features in common between the three versions of sdhash (percentage).

The reason for about half of features being different in comparison for the sdhash version and fixed ones for the same file is mostly due to the "leftmost bug". As explained in the report provided in our GitHub page, the original sdhash implementation selects the rightmost lower entropy value when looking for unique features for a given object. The sdhash paper says to choose the leftmost one, the opposite of what has been done by current implementation. This is one of the main reasons for this difference in the feature set.
Table 3. Percentage of unique features in common among the different versions of sdhash.

<table>
<thead>
<tr>
<th>File</th>
<th>sdhash vs. e-*</th>
<th>e-* vs. e-CW</th>
</tr>
</thead>
<tbody>
<tr>
<td>000021.pdf</td>
<td>54.55</td>
<td>54.55</td>
</tr>
<tr>
<td>000037.xls</td>
<td>42.20</td>
<td>42.20</td>
</tr>
<tr>
<td>000612.html</td>
<td>63.57</td>
<td>63.57</td>
</tr>
<tr>
<td>001846.text</td>
<td>61.90</td>
<td>61.90</td>
</tr>
<tr>
<td>003820.doc</td>
<td>62.07</td>
<td>62.07</td>
</tr>
<tr>
<td>004393.text</td>
<td>47.53</td>
<td>47.53</td>
</tr>
<tr>
<td>004593.ppt</td>
<td>35.71</td>
<td>35.71</td>
</tr>
</tbody>
</table>

Another point we would like to highlight between the original sdhash implementation and its fixed versions is in the number of features extracted. Tab. 4 presents the total number of features per data set and per tool. We can see an increase related to the number of the features between the original and the fixed versions. Since the fixed versions have more features in general, we believe this is another reason for the decrease in precision. In our data set, we have many different objects that were considered similar due to the common features [4] (piece of data found across many different files), which are not desired in most digital forensics investigations. With more features, more common features were found, more matches considered similar, and more the precision decreased.

To support this statement, we adopted the same approach of [4] and removed the common features from sdhash digests. We adapted the original version of sdhash and the two fixed ones to identify and ignore common features. Fig. 2 shows our results for the same data set used before considering precision only. By removing common features, we can see that all versions of sdhash (including the original one) that removed the common features had significant improvements in relation to the version with no distinction of features. We can also observe that the precision of the fixed versions were, in most of the time, equal/slightly better than the original sdhash. This way, our statement about the common features stands and we can conclude that fixing sdhash will improve its detection capabilities, making it better in the similarity detection having no distinction between "good" (user-generated) or "bad" (application-generated) similarity.

An example of how the detection capabilities of e-sdhash and e-sdhashCW can better express similarity is presented in Tab. 5. Here, we can see the score produced by sdhash and its fixed versions for comparisons of non-structured data (text and most html files). The fixed versions had higher scores than original sdhash. Besides, one comparison having a 0 score for sdhash had higher scores for the other versions. By these results, we can see that other ways of evaluation are necessary, in special one taking into consideration the score, since precision and recall only can not tell the whole story.
5. The need for a new metric

Since recall and precision only provide a binary answer considering only whether or not a match is similar, other metrics that take into consideration the quality of the score are necessary. Here, we propose a new metric $M$ (see below) that considers how the score of an AM tool is close to the real similarity shared between the files.

$$M = \left| \frac{exp - score}{sim\%} \right|,$$

where $exp$ is the expected score of similarity, $score$ is the result of the AM tool, and $sim\%$ is the known similarity of a match (percentage of equal bytes). Notice that the values of $exp$ and $sim\%$ are the same, although they have different definitions. This metric measures how distant the score reported by the AM tool is from the real similarity; the shorter the distance, the better. Using this formula, we can measure the quality of the score of each tool and perform a more detailed comparison, analyzing how far from reality the tools are.

To evaluate sdhash and its fixed versions using the proposed metric, we need to know the exact amount of similarity between objects. For this reason, we will create a synthetic data set (using /dev/urandom library) where we can control the levels of similarity between objects. Then, we will test the tools by performing the single common block and random noise resistance tests [2], varying the amount of similarity of objects, their sizes, the position where the similar content is placed, etc. By changing these parameters we can see in which scenarios the fixes impacted the most.

6. Conclusion

In this work, we presented the Approximate Matching (AM) functions for dealing with the problem of finding similar data in a large corpus efficiently. We showed that one of the most popular tools (sdhash) have several implementation bugs/issues and present news versions of it where these problems are addressed. Besides, we demonstrated that the fixed versions performed worst (in general) than the original sdhash with respect to recall and precision rates. We also discussed the reasons for such behavior and the need for a new metric that better highlights the impact of the changes. We hope that adopting the proposed metric, that takes into consideration the quality of the score, we can understand the impact of the fixes and its benefits. Future work encompasses analyzing the tools under the new metric for a synthetic data set which we can control the levels of similarity between files.

References


