Relevant Change Detection

A Framework for the Precise Extraction of Modified and Novel Web-based Content as a Filtering Technique for Analysis Engines

Kevin Borgolte, Christopher Kruegel, Giovanni Vigna

Department of Computer Science
University of California, Santa Barbara
Santa Barbara, California, United States of America

kevinbo,chris,vigna@cs.ucsb.edu

ABSTRACT

Tracking the evolution of websites has become fundamental to the understanding of today's Internet. The automatic reasoning of how and why websites change has become essential to developers and businesses alike, in particular because the manual reasoning has become impractical due to the sheer number of modifications that websites undergo during their operational lifetime, including but not limited to rotating advertisements, personalized content, insertion of new content, or removal of old content.

Prior work in the area of change detection, such as XyDiff [3], X-Diff [8] or AT&T's internet difference engine [4], focused mainly on "diffing" XML-encoded literary documents or XML-encoded databases. Only some previous work investigated the differences that must be taken into account to accurately extract the difference between HTML documents for which the markup language does not necessarily describe the content but is used to describe how the content is displayed instead. Additionally, prior work identifies all changes to a website, even those that might not be relevant to the overall analysis goal, in turn, they unnecessarily burden the analysis engine with additional workload.

In this paper, we introduce a novel analysis framework, the Delta framework, that works by (i) extracting the modifications between two versions of the same website using a fuzzy tree difference algorithm, and (ii) using a machine-learning algorithm to derive a model of relevant website changes that can be used to cluster similar modifications to reduce the overall workload imposed on an analysis engine. Based on this model for example, the tracked content changes can be used to identify ongoing or even inactive web-based malware campaigns, or to automatically learn semantic translations of sentences or paragraphs by analyzing websites that are available in multiple languages.

In prior work, we showed the effectiveness of the Delta framework by applying it to the detection and automatic identification of web-based malware campaigns [2] on a data set of over 26 million pairs of websites that were crawled over a time span of four months. During this time, the system based on our framework successfully identified previously unknown web-based malware campaigns, such as a targeted campaign infecting installations of the Discuz!X Internet forum software.

WWW'14, April 7–11, 2014, Seoul, Korea.
ACM 978-1-4503-2745-9/14/04.
http://dx.doi.org/10.1145/2567948.2578039.

Categories and Subject Descriptors

D.2.13 [**Software Engineering**]: Reusable Software; H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*Clustering, Information filtering, Selection process*; I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving—*Uncertainty, "fuzzy", probabilistic reasoning*

Keywords

near-duplicate detection; change detection; change extraction; modified content extraction; analysis pipeline optimization;

1 INTRODUCTION

Identifying and automatically tracking changes made to websites has become essential to the understanding of today's Internet for developers and business alike as they leverage web-based content more and more. Existing work in the area of web evolution [1, 5–7] suggests that websites do not simply change randomly or only with new content; instead, they evolve constantly through many small changes. If one takes into account the trend toward personalization of websites, such as through personalized advertisements, a change is even likely to happen at each visit, which, in turn, makes it necessary to check if any relevant changes were made on each single visit. Very often, the relevancy of a change is determined by computationally costly analysis engines. Overall, these results suggest that it becomes increasingly expensive for developers and business to reason about the novelty of crawled web-based content, i.e., if it has been seen yet (possibly in a slightly different, personalized way), and it also becomes increasingly hard to filter modifications that might not be relevant for a given analysis goal.

In this short paper, we propose a solution to this problem by introducing a framework to (i) identify and extract modified relevant elements more accurately while ignoring irrelevant changes, and (ii) to reduce the time spend to analyze changes that are similar to previously observed modifications.

The main contributions of this short paper are:

- We introduce the Delta framework, a framework based on a novel extension of existing change detection algorithms to detect and identify relevant changes in websites more accurately, to group similar changes, and to reduce the workload of an analysis engine.
- We introduce a tree difference algorithm that is resistant to tiny changes, for instance the correction of typographical

mistakes or evolutionary steps a website normally undergoes that are not significant or relevant to the overall analysis goal.

2 DESIGN

The Delta framework's analysis process consists of four simple steps that are responsible for extracting the relevant changes made to a website and filtering out duplicates. For each of these steps, some parameters must be chosen that depend on how the relevant changes manifest themselves in a website and in what way they need to be identified and can be tracked. For instance, those parameters include threshold values, evaluation functions and the clustering algorithm that is being used to group similar modifications. The framework's four simple steps are:

1. Retrieval of the current version of the website and normalization of the DOM-tree.
2. Similarity measurement with respect to a base version.
3. Clustering of similar modifications to reduce the submissions that are being send to the analysis engine.
4. Analysis of the identified and novel modifications with a potentially computationally costly analysis engine.

In the general setting of analyzing a website, one -clearly- has to retrieve it first. Once the current version of a website was retrieved, in the setting of the Delta framework, we perform some basic normalization steps on the DOM-tree. These normalization steps take care of simple semantic equivalences that can be represented in syntactically different ways, for instance, such as different quotation marks for attribute values of HTML tags or the order in which attributes are listed.

In the second step, since the Delta framework leverages the changes made to a website to reduce the analysis workload, we require a base version of a website to which we can compare the current version and based on which we can extract the modified content. Such a base version can be (a) stored in a locally kept database that caches a previously fetched version (potentially selected or filtered according to some criteria), or (b) retrieved from an online archive like the Internet Archive[1] or a cache provided by a search engine. In our framework, the difference between the base and the current version is then extracted with an unordered fuzzy tree difference algorithm that we describe in more detail in Section 3.

Upon extraction of the modified content, the framework leverages a machine learning algorithm, which uses specific features to model the relevance of a modification, to discard irrelevant changes and to cluster relevant and similar changes together. For instance, in the case of detecting a web-based malware campaigns, one want to use features that model the same behavior and one also want to use a density-based clustering method that includes outlier detection to recognize early on when a new campaign starts. Thanks to this clustering step, we can filter out irrelevant changes as well as changes that are similar to already observed modifications even before the final, potentially computationally costly analysis occurs. In this step, in sum, we are preventing the submission of already analyzed content to the analysis engine.

Ultimately, only novel modifications that have not been observed and analyzed yet are passed to the analysis engine that then determines if the modifications are relevant or not, for instance, if a website is malicious or if an English article was translated to Spanish or French.

[1]http://www.archive.org

3 FUZZY TREE DIFFERENCE

A fundamental requirement to measuring the similarity between two websites is an understanding of the notion of difference, i.e., what does it mean that two websites are different? In case of the Delta framework, we are also only interested in significant or relevant changes made to a website in respect to our overall analysis goal. For example, a change is generally not going to be relevant if simply the checksum of a website changes, but instead one might want to track the modifications that would make a site behave maliciously and infect visitors with malware, or, one might only be interested in the changes where the language changed from English to French or Spanish. The applications of the framework and the different notions of relevant modifications are endless.

This restriction to only relevant changes in terms of our analysis goal makes it impossible to generally define the term of a relevant modification. Since our goal is to provide a general analysis framework regardless, we simply require that any relevant difference can be expressed as a change between the normalized DOM trees of the two websites, i.e., relevant changes must survive the normalization step or a modified normalization method must be employed.

To extract the changes made to a website, we leverage existing work on change detection [3, 4, 8] and we employ a tree difference algorithm that we generalize to a fuzzy notion to remove irrelevant modifications as early on as possible. Our fuzzy tree difference algorithm is shown in Algorithm 1 where the arguments T_1 and T_2 correspond to the normalized DOM-trees of base and current version, and where t_r is the similarity threshold. Ultimately, to the end of not discarding relevant modifications, for a specific use of the framework one must understand and define what changes are relevant and select suitable threshold values as well as a suitable similarity measure between two strings and a suitable (fuzzy) hash function.

While the reasoning behind coloring nodes in our algorithm might not be obvious in the first place, one can imagine leveraging the color of a node to analyze and keep track of the propagation of a template or part of a website that is repeated multiple times. For instance, to detect a matching asymmetry if a new blog post based on the same template was published while all prior posts are still being kept on the website.

Generally speaking, using a cryptographic hash function when setting the threshold value of 1 corresponds to a standard unordered tree-to-tree algorithms. On the contrary, setting the threshold to 0 independent of the hash function is equivalent to comparing all element with each other and therefore computationally impractical for any modern website because of the extremely large number of possible combinations, which is why reducing the number of potential match pairs through a similarity measure is fundamental to leveraging computationally more expensive analysis techniques.

For an example difference computation between two web sites based on the algorithm, we refer to our prior work [2] that discusses the nuances of the fuzzy tree difference algorithm including a difference example in greater detail.

4 EFFECTIVENESS OF THE FUZZY TREE DIFFERENCE ALGORITHM

In the previous section, we introduced the fuzzy tree difference algorithm, a generalization of existing tree difference algorithms to a fuzzy notion. The algorithm is a cornerstone of the Delta framework since it is responsible for the identification and removal of irrelevant modifications. To measure its effectiveness, we now provide an overview on the reduction

Algorithm 1 Fuzzy Tree Difference

```
1  function FuzzyTreeDifference(T1, T2, tr)
2      G ← Graph
3      for all n ∈ T1.nodes do
4          G ← G.insert_node(n)
5      for all n ∈ T2.nodes do
6          for all m ∈ T1.nodes do
7              if path(m) = path(n) then
8                  d(m,n) ← similarity(hash(m), hash(n))
9                  if d(m,n) ≥ tr then
10                     G.insert_node(n)
11                     m.color ← blue
12                     n.color ← blue
13                     G.insert_edge(m, n, d(m,n))
14     M ← max_weight_matching(G)
15     for all (m, n) ∈ M do
16         T1.remove_node(m)
17         T2.remove_node(n)
18     return T1, T2
```

in the number of tags that were extracted from a website pair, and which we ultimately have to analyze to determine if they are relevant changes.

For our work on detecting infection campaigns [2], we gathered a data set of over 26 million unique pairs of websites where for each pair the normalized DOM-trees are different in their SHA256 checksum once flattened. For a more in-depth discussion of the characteristics of our data set, we refer to our prior work.

To evaluate the reduction capabilities of our framework, we investigated to what degree we were able identify unmatched tags, i.e., tags which were removed or inserted into the new version of a website that are different from tags we observed previously. Figure 1 provides an overview about the number of tags that did not match any other tag. In detail, the introduced framework can, even for very high threshold values ($t_r = 0.99$), efficiently reduce the number of relevant tags that are extracted from the distinct pairs of websites where a standard tree-to-tree algorithm would identify vastly more tags. In turn, this reduction allows us to use more computationally costly analysis methods in later steps while still keeping the total amount of time that is being spend analyzing constant.

For instance, in the case of the Delta-system, for over 80% of all pairs, we eliminated all potential matches in which a standard tree difference algorithm would have identified at least one tag as being different and were additional analysis time would have been spend unnecessarily if not for the fuzzy nature of our approach. If the work per modification is constant, the reduction observed corresponds to a reduction in analysis time of at least 80%.

5 CONCLUSION

In this paper, we introduced the Delta framework, a more general notion of the Delta-system [2], i.e., a novel, light-weight framework to identify, extract and filter relevant changes made to websites before passing them on for further analysis. To support the accurate extraction of the relevant modifications made to a website, the framework leverages a fuzzy tree difference algorithm that extracts DOM-tree nodes that were more heavily modified than others, i.e., it can discard changes in single characters or words, and can even discard legitimate evolutions of a website that would not yield any further insight after analysis. In addition, to keep the overall analysis

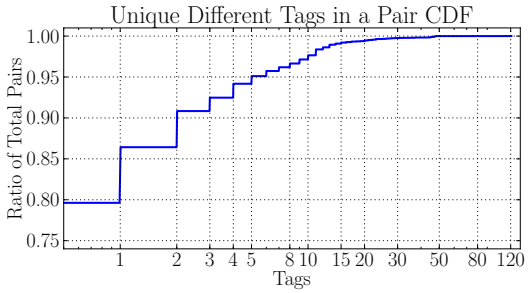

Figure 1: Overview of website pairs that have at most x tags unmatched tags (removed and inserted tags are combined).

workload low, the framework's clustering step groups together similar modifications, which allows to transfer analysis results within a group of the same change without requiring analysis.

As an example use case of the Delta framework, we referred to the Delta-system that successfully detects previously unknown web-based malware campaigns in the wild. Additionally, we support our claim of the reduction capabilities of the Delta framework by investigating the number of extracted relevant modifications in comparison to a standard tree-to-tree comparison algorithm. Ultimately, the reduction capabilities of the framework combined with its clustering step support the use of more elaborate and computationally more expensive analysis techniques that can be more precise.

Acknowledgment

This work was supported by the Office of Naval Research (ONR) under grant N000140911042, the Army Research Office (ARO) under grant W911NF0910553, the National Science Foundation (NSF) under grants CNS-0845559 and CNS-0905537, and Secure Business Austria.

6 REFERENCES

[1] R. Baeza-Yates, C. Castillo, and F. Saint-Jean. Web Dynamics, Structure, and Page Quality. In *Web Dynamics*, pages 93–109. Springer-Verlag, 2004.

[2] K. Borgolte, C. Kruegel, and G. Vigna. Delta: Automatic Identification of Unknown Web-based Infection Campaigns. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, pages 109–120. ACM, 2013.

[3] G. Cobena, S. Abiteboul, and A. Marian. Detecting changes in XML documents. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 41–52. IEEE, 2002.

[4] F. Douglis, T. Ball, Y.-F. Chen, and E. Koutsofios. The AT&T Internet Difference Engine: Tracking and viewing changes on the web. *World Wide Web*, 1(1):27–44, 1998.

[5] F. Douglis, A. Feldmann, B. Krishnamurthy, and J. Mogul. Rate of Change and other Metrics: a Live Study of the World Wide Web. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, volume 119. USENIX Association, 1997.

[6] D. Fetterly, M. Manasse, M. Najork, and J. Wiener. A large-scale study of the evolution of web pages. In *Proceedings of the 12th International Conference on World Wide Web*, WWW '03, pages 669–678. ACM, 2003.

[7] B. A. Huberman and L. A. Adamic. Evolutionary Dynamics of the world wide web. *Condensed Matter*, January 1999.

[8] Y. Wang, D. J. DeWitt, and J.-Y. Cai. X-Diff: An effective change detection algorithm for XML documents. In *Proceedings of the 19th International Conference on Data Engineering*, ICDE '03, pages 519–530. IEEE, 2003.