

Improving web traffic inference using page level embedding information[†]

Olivier Paul
GET/INT
9 rue Charles Fourier
Evry, France
Olivier.Paul@int-evry.fr

Abstract—This paper presents a new technique to improve the accuracy of a web traffic analysis tool. This technique is based on embedding relations existing between objects in a web site. We show how such relations can be used to improve the understanding a tool can get from network level measures.

Keywords: *Traffic analysis.*

I. INTRODUCTION

Over the past few years, there has been a growing interest in techniques allowing application level information to be obtained from network level measures ([1], [2], [3]). These techniques usually associate the usage of application level protocols or application level operations with network level signatures in order to recognize such events without requiring expensive changes in existing network devices. In [4] we present a tool called RequIn, allowing such analysis to be carried on web traffic. In this paper we present an improvement to this tool allowing an improved accuracy in the recognition of operations. The paper is organized as follows. Section II presents RequIn as well as its limitations. Section III presents our proposal. Section IV presents its implementation as well as some preliminary test results. Section V compares our proposal to existing alternatives. Finally section VI concludes this paper.

II. WEB TRAFFIC INFERENCE

The core of our architecture is based on RequIn [4]. RequIn is a web traffic inference tool that uses network level measures on flows (timestamps, number of bytes exchanged in both directions, number of packets, source and destination addresses and ports) to infer application level information.

The amount of information inferred from network level measures depends on the type of request performed (GET, HEAD, POST...) and the corresponding result code (200, 404, 304...). These two parameters are themselves inferred from network level measures. TABLE I. provides the relation between inferred result code, method, URI and object size.

As described in [4], RequIn suffers from a few limitations. For example since URI inference is partly based on the volume of information transferred URI information can only be obtained when objects are actually transferred from the server

to the client. As a result requests generating erroneous (404) or unmodified (304) responses cannot be analyzed as precisely as correct (200) answers. Another limitation is the proportion of correctly inferred requests which is around 70% for a medium size web server.

TABLE I. INFERRED HTTP HEADERS DEPENDING ON HTTP RESULT CODE AND METHOD.

Result code	Method	URI	Object Size
200	GET	Yes	Yes
	HEAD/ TRACE	No	No
	PUT/POST	No	Yes
404	GET/HEAD	No	No
	PUT/POST	No	Yes
	TRACE	No	No
304	GET/HEAD/ TRACE	No	No
	PUT/POST	No	Yes
Other	Any	No	No

III. USING PAGE LEVEL INFORMATION

The idea behind our proposal is simple and based on the notion of page.

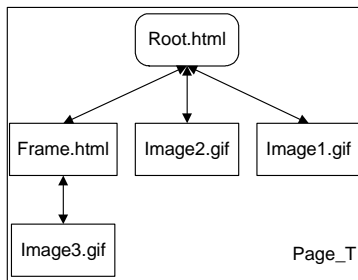
Individual object requests are often performed to web servers as a part of a page request. Each page includes a root document to which a set of objects (e.g. images, sounds, other html documents ...) are connected. Connected objects are most of the time downloaded automatically by browsers when the root document is received. The transfer of this set of objects (including the root and connected objects) is usually called a page transfer ([5], [6]). A browsing session usually includes a set of page transfers from the time a user starts browsing a web site until he stops for a significant amount of time. Page transfers are separated from each other by a minimum amount

[†] This work was funded through the FP6 IST DIADEM Project

of thinking time. A typical minimum value for this thinking time is one second ([5], [6]).

By taking into account the notion of page, we can learn about embedding relations between objects. We believe these relations can be used to improve or correct inference operations performed in [4]. For example assuming request to a page constructed as presented in Figure 1. and assuming that "Image2.gif" is already in the browser cache (because it was downloaded through a different page some time ago), the server would answer with a "304" (not modified) response thus preventing our inference software from obtaining the identity of the requested object. If we use page level information and are able to guess that "Page_T" was the page requested during browsing operations, we would be therefore able to associate this "304" response with the appropriate object.

FIGURE 1. PAGE COMPOSITION EXAMPLE



In a second scenario, let's assume that the request for "Frame.html" was incorrectly guessed as "Doc.txt" by our inference software. If we succeed to guess that this request was part of a request to the "Page_T" page we can deduce that "Doc.txt" was unlikely to be transferred while downloading this page. This could help us switching from "Doc.txt" to a more likely answer.

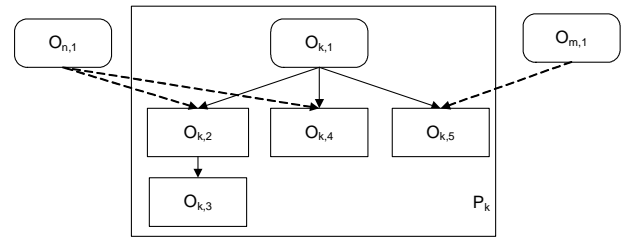
Obtaining information about embedding relations can be performed by analyzing server logs using clients addresses and timing information (as well as reference information when the server is configured to record it). It can also be done by analyzing servers using page analysis tools. Both approaches have advantages and drawbacks as each might miss some kind of relations. For instance server logs might not include references to pages that have never been requested by a user while automated analysis tools can miss pages that are unconnected to other objects. The first approach was selected in this paper due to its easy implementation.

Using embedding relations is however not very easy. As mentioned earlier due to objects missing or already in cache, any member of the "Page_T" transfer might be missing including the root document. Additionally because browsers often employ multiple TCP connections to transfer objects, request and responses might be received in any order. Finally objects embedded in one page might also be embedded in other documents as presented in Figure 2.

Our inference page level technique is a based on a few heuristics that we explain using the page example presented in Figure 2. As presented in this diagram the root $O_{k,1}$ of page P_k

refers to several objects among which a few objects are also referred by roots belonging to other pages.

FIGURE 2. RELATIONS BETWEEN OBJECTS



When a request to page $P_k(O_{k,1}, O_{k,2}, \dots, O_{k,p(k)})$ is performed, a set of flow records $F=(F_1, \dots, F_f)$ representing the transfers of objects between the server and the web client is received by our inference software. A general objective for our page level inference technique is to match these flow records to existing objects. In order to do so we use the inference function developed in [4]. This function, in accordance with the restrictions mentioned in TABLE I. provides us for each flow (or object request) F_i with:

- The operation performed (C_i).
- The result code (R_i).
- A set of possible object identifiers with their respective likelihood $S = ((O'_{i,1}; P_{i,1}), (O'_{i,2}; P_{i,2}) \dots (O'_{i,n(i)}; P_{i,n(i)}))$.

Using object identifiers we would like to find the page $P_k(O_{k,1}, O_{k,2}, \dots, O_{k,p(k)})$ that generated F : a set of objects/likelihood couples $(O'_{i,j}; P_{i,j})$ such that:

$$\forall i, 1 \leq i \leq f; \exists j, 1 \leq j \leq n(i);$$

$$(1) \exists k, 1 \leq k \leq t; \exists l, 1 \leq l \leq p(k)$$

$$/ O'_{i,j} = O_{k,l}$$

$$(2) \sum_{i=1}^f P_{i,j} \text{ is maximized.}$$

However a straightforward implementation of (1) and (2) would not work well for several reasons: Since the inference function provides probabilistic results an inferred object $(O'_{i,j}; P_{i,j})$ might not match any requested object $O_{k,l}$. Reciprocally a requested object $O_{k,l}$ might not match any inferred object $(O'_{i,j}; P_{i,j})$ for example when such object is in a cache. Moreover a naïve implementation of such a scheme would require on average $O(f \cdot t \cdot \bar{n} \cdot \bar{p})$ comparisons for each page request where \bar{n} and \bar{p} represent respectively the average number of inferred objects per flow and the average number of objects embedded in a page. This can result in a slow implementation.

Consequently we propose a different procedure based on a heuristic where we attribute scores to potential objects generated through a request. Points are attributed to an object $O_{i,j}$ in two main cases.

- $(O_{i,j}; P_{i,j})$ has been inferred (i.e. $(O_{i,j}; P_{i,j}) \in S$).

- $O_{i,j}$ is a referrer for an inferred object ($O'_{i,j}$; $P'_{i,j}$) (i.e. ($O'_{i,j}$; $P'_{i,j}$) $\in S$ is referred by $O_{i,j}$). This second case is meant to solve the situation where the root would be missing, would have been incorrectly inferred or would have been inferred with a low likelihood score.

The number of points attributed depends on the likelihood $P_{i,j}$ (or $P'_{i,j}$). In addition to these points an object can get additional points when the number of referred objects matches the number of flows generated by a client or when the object is a root node (refers other objects). Our assumption is that when a sufficient number of referred objects are received, the score of the root will increase in such a way that it will become easily distinguishable from other potential roots.

Once a root object is identified we can match referred objects to existing flows using flow sizes, operations and response code results as clues. Requests to standalone objects (e.g. pdf/zipped/MS word files) can be treated the same way, the only difference being that only the direct likelihood is used.

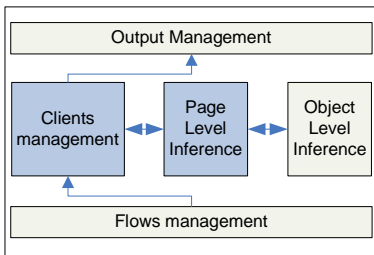
Since several page requests can be performed to a server by a single client over a period of one second (our minimum thinking time interval). This process might have to be executed several times until every flow is matched to an object.

In order to simplify computations, points are maintained using a table indexed using objects identifiers. Using such a table points attributions necessitate on average $O(f \cdot \bar{r} \cdot \bar{n})$ computations per page request. This is usually much better than the naive $O(f \cdot t \cdot \bar{n} \cdot \bar{p})$ as $\bar{r} \ll t$.

IV. IMPLEMENTATION AND EARLY RESULTS

Page level operations act as an additional module to the software that was developed in [4] and uses existing object level inference functions and flows and output management modules as shown in Figure 3.

FIGURE 3. TOOL ARCHITECTURE



When request/response records are received, they are gathered according to their source IP address by the clients management module. The list of clients that have not generated any requests for more than the chosen thinking time is obtained regularly and passed to the page level inference module which sequentially examines the set of flows generated by each client. To do so, this module implements the scheme mentioned in section III. Once every flow is matched to an object it is passed to the output management module which formats the results.

In order to test our module we obtained logs from an existing departmental server including roughly 15k objects.

The logs, including 309k entries, were divided in a training set consisting of 240k entries and two testing sets. The first one (Set1) was made of 10k entries randomly selected and a second testing set (Set2) was made including one page request for every page found in the remainder of the 70k entries log file. The training set was used to build the information used by the object level inference and page level inference processes. Testing records were used to generate a script automatically downloading pages using a widely used commercial browser with a few seconds interval between downloads. After running the script, the web server log file was recovered and new request records were extracted (server logs #2). The inferred results provided by the output management were stored in another file (inference results).

Test results are presented in TABLE II. The correctly inferred column indicates the proportion of requests found in the “server logs #2” file that could also be found in the “inference results” file at the time the request was performed. Reversely, the incorrectly inferred column indicates the proportion of requests found in the “server logs #2” file that could not be found in the “inference results”. In both cases we consider that two requests match when they apply the same operation to the same object.

Jaccard based results which are based on an alternative approach presented in [8] will be explained in the next section. Overall the page level approach improves significantly inference results.

Performances were measured in both cases on a Pentium xeon 2.6Ghz with 1Gb of memory. Results are given in TABLE II. Overall the performance is 5 to 10 times slower. In its current state, our software is able to analyze between 50k and 100k requests per second. The difference in term of performance between Set1 and Set2 is mainly explained by the composition of the two sets in term of proportion of pages and standalone objects. The identity of a standalone object can be obtained much faster than the identities of objects belonging to pages since no referral relation needs to be used.

TABLE II. INFERENCE AND PERFORMANCE RESULTS

Method, Set	Correctly Inferred	Incorrectly Inferred	Time per request
Jaccard Based, Set1	49%-71%	51%-29%	/
Object Based, Set1	64%	36%	1.9us
Page Based, Set1	82%	18%	9.7us
Jaccard Based, Set2	45%-63%	55%-37%	/
Object Based, Set2	57%	43%	2.0us
Page Based, Set2	81%	19%	21us

V. RELATED WORK

As of today, web traffic analysis is mostly performed using HTTP proxies or using web server logs. Although both approaches are able to provide accurate results they also carry a few drawbacks. HTTP proxies are usually only able to serve a few thousands requests per second making them expensive to

deploy in an operator network if one wishes to analyze any possible exchange. Moreover proxies can sometimes be insufficiently transparent in particular in the case encrypted traffic or when cookies are used. On the other hand using web server logs requires cooperation between the monitor and the server monitored. It also assumes that server logs cannot be tampered with.

Web traffic analysis using packet level measures has been an active field for some time. Hernández-Campos and al. ([10], [5]) introduces the idea of using packet trains to delimitate HTTP operations. The paper also introduces a set of rules to classify HTTP result codes based on the size of packet trains. In [4] we build on this seminal work and introduce a technique to identify individual objects from flow records using a more complex statistical model based on a Bayesian network. Such model allows using a larger number of flow descriptors to obtain more accurate inference results than those that might have been obtained solely using server responses sizes.

After submitting this paper we found a couple of papers ([7][8]) using an idea somehow similar to ours in order to perform TLS/SSL encrypted web traffic analysis. In both papers the influence of SSL is considered as a constant overhead and traffic analysis tests were carried over non SSL traffic. A main difference between our approach and theirs is that they do not attempt to use individual objects identities but instead use the notion of group of requests sizes. As a result objects to pages relationships cannot be exploited.

More precisely, in [8] the authors only distinguish between the root document request and the set of requests used to gather referenced objects. They use the amount of data transported from the server to the client for both type of requests to build signatures that can later be matched to packet trains. In [7] Qixiang and al. identify individual objects transfers using a technique similar to the one in [10]. They later use the sizes of individual objects transfers to create a set of sizes for each page request. These sets are used as signatures to identify page transfers from web servers. Signatures and transfers are matched using a similarity measure based on the Jaccard coefficient. The Jaccard coefficient of two sets U and V is defined as:

$$J(U, V) = \frac{|U \cap V|}{|U \cup V|}.$$

Performance comparison with these approaches is not obvious. Both exhibit results that are similar to our proposal. However the content of the testing sets can have a large impact on results. For instance servers used in [8] only include a few hundreds pages when ours includes a few thousands. In [7] the authors use a testing set of 100k pages originating from several web sites. However the median number of objects referenced per page in their testing set is 11 when ours is 5. Finally both proposals do not take into account the effect of aging on the composition of pages which might affect inference accuracy unfavorably as shown in [4]. As [8] appears to be the most complete proposal, we implemented the matching technique used in [8] in order to perform a comparison using a single

testing set. We however did not compare the time per request as we did not try to optimize the similarity comparison implementation. Results are the first set of numbers reported in the "Jaccard based" line in TABLE II. Overall our implementation performs significantly better. Apart from the differences mentioned above another explanation could be that [8] does not try very hard to obtain correct object sizes by only considering the number of bytes transported in packets (the authors claim a small difference does not have a large impact on inference results). Instead we perform [4] additional computations based on the expected sizes of HTTP headers. However even when using these additional computations the method proposed in [8] provided results that were still lower (15% on average) than our page based proposal (Results are the second set of numbers reported in the "Jaccard based" line in TABLE II.).

VI. CONCLUSION

In this paper we show how page level relations can be used to significantly improve the accuracy of our web traffic analysis software. We believe this work can impact two different fields. One of them is network management as the ability to infer users' activities can provide network operators with tools to differentiate traffics more accurately. The other field is the privacy of communications. As shown here, even synthetic, flow based information can lead to a good understanding of operations performed by users. Interestingly, as demonstrated in [7], [8] as well as more recent work [9], TLS/SSL encryption might not provide much help in term of privacy for web exchanges.

VII. REFERENCES

- [1] T Karagiannis & al., BLINC: Multilevel traffic classification in the dark, ACM SIGCOMM 2005 conference, 2005.
- [2] Kuai Xu & al., Profiling Internet Backbone Traffic: Behavior Models and Applications, ACM SIGCOMM 2005 conference, 2005.
- [3] J. Herman, al., Internet Traffic Identification using Machine Learning, IEEE GLOBECOM 2006 conference, 2006.
- [4] Olivier Paul, Jean Etienne Kiba. RequIn, a tool for fast web traffic inference, IEEE GLOBECOM 2005 conference, 2005.
- [5] Félix Hernández-Campos & al., Tracking the Evolution of Web Traffic: 1995-2003, ACM/IEEE MASCOTS 03 conference, 2003.
- [6] P. Barford and M. E. Crovella, Generating Representative Web Workloads for Network and Server Performance Evaluation, ACM SIGMETRICS '98 conference, 1998.
- [7] Statistical Identification of Encrypted Web Browsing Traffic, Qixiang Sun, Daniel R. Simon, Yi-Min Wang, Will Russell, Venkata N. Padmanabhan, Lili Qiu, In Proceedings of IEEE Symposium on Security and Privacy, Oakland, CA, USA, May 2002.
- [8] Traffic analysis of SSL encrypted Web Browsing, Heyning Cheng and Ron Avnur, 1998.
- [9] sMonitor: A Non-Intrusive Client-Perceived End-to-End Performance Monitor of Secured Internet Services, Jianbin Wei and Cheng-Zhong Xu, USENIX Annual Technical Conference, 2006.
- [10] What TCP/IP Protocol Headers Can Tell Us About the Web, F. Donelson Smith, Felix Hernandez-Campos, Kevin Jeffay, David Ott, ACM SIGMETRICS/Performance, 2001.