Tracking HTTPS Service Use and Impact on the Private Life's Users: an Application to an Image Search Engine

Charlotte ASSELIN-BOULLÉ
Guillaume ROZIER

TELECOM Nancy
Villers-lès-Nancy, 54600 FRANCE
{charlotte.asselin-boulle, guillaume.rozier}@telecomnancy.eu

Pierre-Olivier BRISSAUD
Jérôme FRANÇOIS

Inria (Loria)
Villers-lès-Nancy, 54600 FRANCE
{pierre-olivier.brissaud, jerome.francois}@inria.fr

***Abstract*—Because of the development of encryption to protect user privacy, it has been more difficult to monitor traffic for control purposes.**

However, the main purpose of our research was to add some monitoring on HTTPS service in order to enable a company to detect the non legitimate behaviour on his network by using a blacklist, to filter some results on different HTTPS services. Nevertheless, the goal is also to keep a notion of employees' privacy.

The method proposed in this paper is, regarding the encrypted sizes of HTTP objects loaded, to use kernel density estimation to create a single signature, and applying a classification method.

The main idea of this paper is to regroup the blacklisted keywords to reduce the number of signatures and at the same time prevent the classifier to guess the exact blacklisted words raised when a detection occurred, without losing too much accuracy by comparison with the 99% of true positives identification in the case of a single signature for each single word.

May 17, 2018

INTRODUCTION

As Cloud Computing is growing, every day more and more people and companies are using web tools instead of native software. With recent public scandals about privacy, and because brand image is very important for companies, the latter started to rely on encryption techniques to protect both users and companies, like HTTPS. Today (may 2018), around 82% of browsing time in the most popular web browser Google Chrome in the USA is encrypted [5].
Even if protecting user privacy is very important, we also need to detect and prevent bad behaviours like illegal activities through the Internet, especially in a company network.
The first solution would be to decrypt HTTPS data using a proxy server, but the goal is not to track every user, but only to prevent illegal usages on some services. The second solution, which is to block the access to a service is not better because a service can be used for both legal and illegal purposes. So, we propose a fast and transparent solution to allow monitoring a web service in order to prevent an inappropriate use of the service without any decryption of the traffic, tracking every user or totally shutting down of the web service.
Our solution is passive (as no intermediate is required) and transparent as no specific software is required to be installed by the users. The solution has two different parts: the first one is the generation of a signature's database, and the second one is the screening of the user's requests on web services.
First, our system will intercept encrypted HTTPS network packets and generate a signature for each web service request (thanks to Kernel Density Estimation (KDE) [3], using the frequency of occurrences and size of each packet). Then, when the learning phase is complete, the system will be able to analyse user encrypted traffic and eventually raise an alert: for each user's request, the system extracts thumbnails' sizes[1] as we implemented our method on an images search engine and compares its to the signatures' database.

More details on this solution can be found in the paper written by P.-O. Brissaud & al. [1].

In this paper, we show that we can build cluster of signatures, so that the filtering part may protect users' privacy and may go faster, without increasing significantly the non recognition rate and the error rate. We tested our work with Google Images, as this web service is really popular and should not be totally blocked as it can be used for both legitimate and illegitimate purposes.

In this project, we tried to evaluate the impact of our monitoring system on the user privacy and find new solutions in order to make our analyse less intrusive.
The first step has been to collect data for each word requested by the user, and making signatures depending on the HTTPS traffic of each word, stored in a database. Then, we tried the simplest way for clustering databases' signatures: randomly assemble groups. Our second approach was to use more

[1]The encrypted sizes of packet of HTTP, in our case the packet contains images, so thumbnails.

advanced clustering methods, like K-medoids algorithm. In this case, we tried to put together the closest words, i.e. words that have the closest signature. Firstly, we tested our algorithms in Closed World, that means the words used for the learning phase were the same that them used for the filtering phase. Then we tested its in Open World, thus new words were used for the filtering phase.

The paper is organised as follow. Section I presents related work. Our model is defined in Section II, and results are given and explained in Section III. Finally, Section IV discuss the results and Section V concludes the document.

I. Related Work

A. Tracking engine

The major concept of the tracking engine was raised by the article [1] co-writen by the search supervisors P.-O. Brissaud and J. François. Indeed the present work is the continuation of the work presented in this paper.
To recap, the main purpose was to show a method of tracking inappropriate uses of HTTPS service. Notably, in the process (mapped in Figure 1), there are two stages: learning and monitoring.

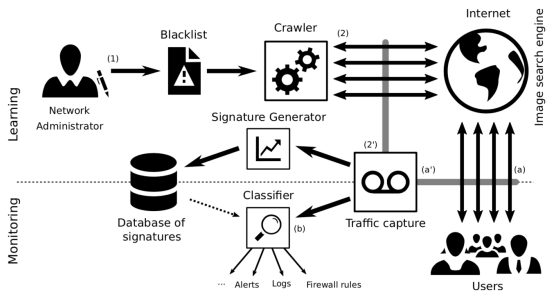

Figure 1. Global architecture to track HTTPS application use [1].

The learning phase is required at least one time before monitoring the user traffic. This first part can be made only once for a same blacklist, even if it is better to repeat it over time for best results, because of the variability over time of Google Images search results. It consists in the generation of a database of signatures where each signature is assigned to the trace of a web page loading. In this case it's a keyword search on Google Images. Each page load correspond to a keyword in a blacklist provided by the network administrator. The crawler requests each keyword of the blacklist multiple times on Google Images and captures traffic. Then, a signature is generated thanks to all the traces stored related to a keyword.

The second phase is the monitoring one. Each time a user will search a keyword on Google Images, the system will intercept the encrypted traffic and store thumbnails' sizes associated with this keyword. The system then compares these signature with the signatures' database and figures out whether it corresponds or not with a blacklisted word. In this case, network administrators can decide for example to block this user or raise an alert.

B. K-medoids clustering

In order to make different groups of words, we tried two methods. The first one is randomly and the second one is to apply a clustering algorithm. In our researches we used K-medoids clustering algorithm. Our studies of this process is mainly based on the class supports of Mr. R. Rakotomalala [2].

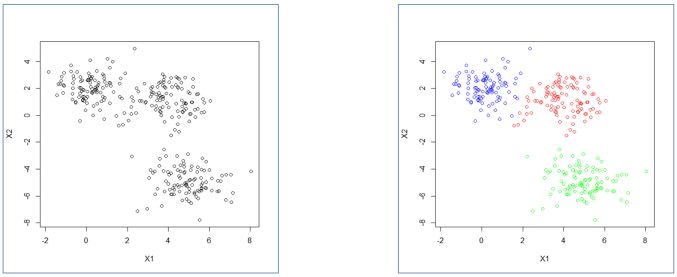

Figure 2. K-medoids made three clusters on this dataset [2].

K-medoids is a clustering algorithm related to the popular K-means clustering algorithm. Both of them are partitional (the main goal is breaking the database up into different groups), and both or them attempt to reduce the distance between every point of a cluster and the center of this cluster. The main difference between them, which makes the K-medoids more robust for abusive values, is also here: while K-means uses mean value as center point, K-medoids uses medoids which is the most centrally located object of each cluster. The K-medoids algorithm is rather simple and has been chosen for two main reasons: (1) it is more robust than K-means, (2) it only needs a distances matrix (as we work with one thousand variables, it is not possible to have the coordinates of each point).

The most popular implementation of K-medoids algorithm is Partional Around Medoids (PAM) [algorithm 1]. Assuming i is an object of the dataset and M_k one of the K medoids of the dataset, the PAM algorithm objective is to minimise the average dissimilarity:

$$E = \sum_{k=1}^{K} \sum_{i=1}^{N_k} \mathbf{d}(\mathbf{i} - \mathbf{M_k}).$$

Algorithm 1 K-medoids PAM

input: K = number of clusters ; M = distances matrix
• Select randomly K points as K-medoids called M_k

while Changes are made **do**
• Assigning each object to the closest medoid M_k
• Calculate the E average dissimilarity
• Randomly select a non medoid object O_i
• Calculate the new average E_{new} dissimilarity of swapping M_k and O_i

if $E_{new} < E$ **then**
swap P_i with M_k
end if

end while

output: dataset divided into K clusters

II. Methodology

A. Overview

The main purpose is to recognise if a keyword associated with an HTTPS request to Google's servers corresponds to a blacklisted one or not. We implemented our solution on Google Search Image service because of its huge database and its popularity. A study on packets' sizes allowed having a significant value used to discriminate each request without decryption of HTTPS protocol.
We also wanted to evaluate the impact of our monitoring system on the user privacy and find new solutions in order to make our analyse less intrusive. In this optic, we introduce clusters of words to mask the blacklisted word pointed out, in the groups where it was included.

The first step was to implement an operation engine to build the database of signatures with the tool KDE [3] and to implement an algorithm to regain a keyword from its signature.
Then, the second major stage was creating groups of words with different methods and adjust the initial algorithm in order to identify the relevant elements and raise the best alternatives.

The main objective was to find an adaptive method, as quick and simple as possible and the most effective. It was also adding, if possible, a privacy notion which does not reduce performances.

B. Initial operation engine

Our system is made of two different parts. This section explains quickly the practical application of these parts.

1) Signature generation

As main data, we had JSon files that contain, for each word of a set of ten thousand keywords from Oxford dictionary, the timestamp of the request and a list of sizes corresponding to the encrypted packet length. We use two of these sizes: req_size which is a request's length, and $resp\_size$ which is the size of an encrypted packet sent to user (Figure 3).

```
{
"Sympathetic":
    {
    "1497870700.22":
        {
        "count": 78,
        "data": [
            {
            "timestamp": 1497870722.18,
            "req_size": 601,
            "packet_number": 1,
            "req_count": 1,
            "resp_count": 272,
            "resp_size": 344288
            },

            ...

            {
            "timestamp": 1497870724.13,
            "req_size": 874,
            "packet_number": 1369,
            "req_count": 1,
            "resp_count": 1,
            "resp_size": 273
            } ]
        }
    }
    ...
}
```

Figure 3. JSON example. In this extract, the data corresponds to the keyword "Sympathetic".

First, we filtered this data in order to delete every packet unrelated to things that does depend to a specific keyword request on Google Images (like Google's logo, searchbar, buttons...). The only things on a Google Images page which are directly related to word search are thumbnails. As said in [1], packets which contain thumbnails' data have, most of the time, a request's size equal to one of these values:

$$req\_size \in \{366, 367, 369, 374, 375, 512, 513, 515, 520, 521\}.$$

The next step is the signature generation. For each keyword, it is necessary to have a signature that depends to its traffic, so that it is almost unique, at least we need to find back the keyword knowing its signature.

The signature function can't be just a list of the thumbnails' sizes because, in this case, it is really difficult to compare

two signatures as each one is a discrete function. In fact, each thumbnails' size can differ a lot between two requests of the same keyword, as Google Images results can be slightly different over time. Thus the result of the comparison of two signatures would be very low and insignificant. So, in our method we use Kernel Density Estimation (KDE) technique which thus allows us to make the signature function continuous, and to smooth the results. Therefore, thanks to KDE it is possible to compare two signatures. Assuming a distribution for the thumbnails' sizes corresponding to a keyword, KDE estimates the density function. This technique creates a continuous function, and it gives us an average of different distributions of sizes for one same keyword. Let $X = (x_1, ..., x_n)$ be a random distribution, K a kernel function and h the bandwidth parameter of the kernel, the KDE density functions is calculated like this:

$$\sigma_i(y) = \frac{1}{nh} \sum_{i=1}^{n} K(\frac{y - x_i}{h}).$$

In our case, n is the number of thumbnails, K is the kernel function, here a top-hat function (it works also with a Gaussian function), h is the width of the kernel function. So the higher h is, the less precise the signature is but also the more smoother the signature is.

K is defined as follow: $K(u) = \frac{1}{2}$ if $u \in \{-1, 1\}$, else 0.

Assuming σ is a signature, so it is the signature of the sum of each trace, the database is composed with every σ elements.

2) Classification engine

The classification engine is the opposite way. Given a list of thumbnails' sizes (stored thanks to a filter between the user and the internet), it should returns the keyword searched by the user on Google Images if the latter is blacklisted, or $not\_in\_blacklist$ if it is not.

For each signature in the database $\{\sigma\}$, a score sc is calculated thanks to Kernel Density Estimation. This score is done by a function considering the correlation between the thumbnails' sizes and some of properties like shape and distribution of the density function. The easiest way would be to associate the thumbnails' sizes list with the keyword associated with the σ_i corresponding to the highest score sc, evaluated thanks to its signatures. However, the classification engine should also decide whether the word is present or not in the blacklist database.

The classification engine should be as accurate as possible and return a blacklisted word only if the score related is much higher than other scores. In our system, the highest score is considered as relevant only if reaching a threshold t depending of the mean and standard deviation of scores calculated as follow:

$$t = mean(L_s) + \alpha \times std(L_s)$$

with L_s the list of calculated scores given a capture s and α the filtering parameter. Its impact on the results is studied in the next chapter (III-C1).

C. Groups

After implementing the method to recognize if a word is in the blacklist and can say exactly which one is, we chose to add a permissive notion. Indeed the purpose of the tracking is not exactly to know which keyword was requested when it's a blacklisted one but only if it's a blacklisted word or a neutral word. In this way, a privacy aspect could be regained. So, the target of this part was to find a way to pool several words in a single signature (generated with the same method as a signature for a single word) without loosing the good classification result.

1) Random groups

The first solution explored is the most intuitive: the idea is to consider all the sizes (*req_size*) in a packet sent to the users corresponding to images (*resp_size*) in all the traces for training not for one word but for a number *N* of words. *N* was a parameter that we changed in order to see the incidence. So, the signature of each group is the same signature generated for each word, but it is generated with the sum of every trace for each word it contains.

In other words, all the signatures of all the traces used for the learning phase of N words were considered as the signatures of the same "keyword": the number of the group.

With this method, when an agreement with the blacklist is raised, the group considered can be known but the access to the particular word can be limited.

2) Dissimilarity scores : K-medoids' groups

The second solution thought was to considerate the similarity between words and build groups regarding to this criterion. First, the linguistics link or the semantic link seems to be a good trail. Nevertheless, we observed that those aspects do not influence the size of the packets, so, there is no relation between those links and similar signatures.

In fact, on Google Image, for two searches of words semantically close the part of mutual thumbnails is substantially low and often of zero. So the probability of same sizes of thumbnails between two words semantically linked is as lower as two words semantically distant.

An relevant process to evaluate similarity was thought. As explained a signature is a density function, so mathematics tools can be used: the integral. Indeed calculating the dissimilarity score between two signatures mentioned in the precedent article [1] (Section VI.A), is calculating the area between the two curves (corresponding of the two signatures), so the "difference":

$$\phi\left(k_1, k_2\right) = \int_0^{20000} |\sigma_{k_1}(x) - \sigma_{k_2}(x)| dx$$

with k_1 and k_2 the words and σ the signature function.

We can check that the dissimilarity score of a word's signature with itself is rightfully 0.

In practice, Kernel Density is a continuous function, so we

used its kernel parameter *tophat* to quantify the density function. In this way, the integral did not need to be discretized, as it can be the sum on each point of sizes sampled (in integer range $[\![0, 20000]\!]$ where the maximum limit was determined experimentally). Thus, the scores were the difference between the integral of two different keywords. Besides, we exploited the exponential of the dissimilarity scores to spread values. Once chosen the way to compared the signatures, a relevant balanced repartition was wanted, reflecting the dissimilarity one by one. As referenced in the Related Work I-B we decided to use the K-medoids method to build the different distributions: the algorithm considers a matrix of distance (the dissimilarity scores) to establish the number of clusters (groups) wanted.

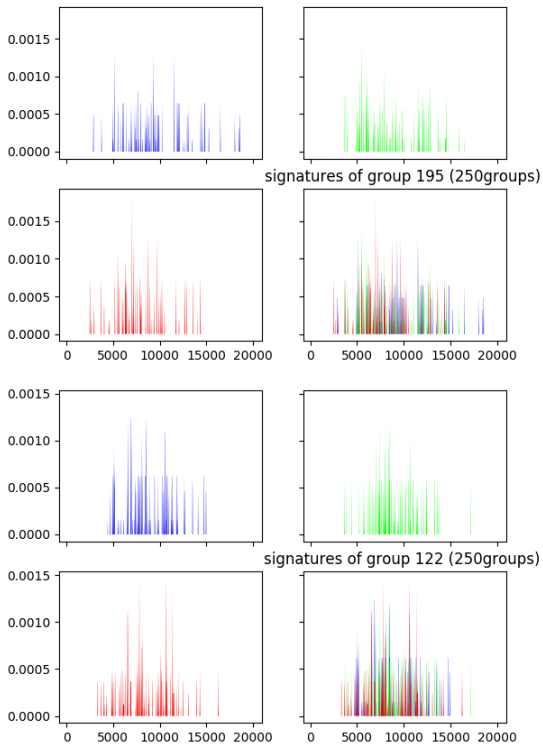

Figure 4. Two Examples of the function signature of 3 words classified in the same K-medoids' group.

The figure 4 bring visual consistency out. Indeed, even if the similarity is not totally relevant, a coherence can be anyway observed: in the two examples the global aspect is distinct and within the same group the 3 words seems, as a matter of fact, to have a concomitant aspect. Besides that, we remind that the signatures are density function which is hard to represent accurately.

III. Results and Interpretations

A. Sample

For creating our databse and evaluating our results, we used two different words databases. In the first case, that is called *Closed World*, we used the same words for creating the dababase and obtaining the results. In the second case, called *Open world*, only less than 10% of the words used were in the blacklist database.

1) Closed World

In Closed World, data samples contains 5 traces of 1,000 words from the Oxford Dictionary. 4 of the 5 traces have been used for training purpose, in order to create the signature's database. The last trace, for each keyword, has been used for testing classification engine. Every graph in Section III-B has been obtained with the 5th trace tested with a signatures' database obtained with the first 4 traces. Therefore, every word tested is part of the blacklist.

2) Open World

Concerning Open World, every tested word is not part of the blacklist. First of all, the signatures' database is the same as for Closed World, so it contains 1,000 signatures from the first 4 traces of each word. To test classification engine, we used the same data sample as for Closed World (the 5th trace of each 1,000 world), but we added 10,000 other traces related to as much new words (not already in blacklist). Thus we finally have 1,000 blacklisted keywords, within the global 11,000 testing keywords.

B. Results in Closed World

This section presents the results of the analyse step, with a signatures' database generated in *Closed World*. Words are clustered into groups with two different methods: firstly randomly, and secondly with K-medoids algorithm.

1) With Random groups

As described in section II-C1, our initial approach was to find the impact of clustering signatures into groups in the database. So the natural path is to randomly generate groups (each group has the same length, plus or minus 1) of different sizes, to create a new signatures database from those groups and finally to test the classification engine with this new database. As the database is made of 1,000 words, we have 1,000 different possibilities: from 1 group (that contains the 1,000 words) to 1,000 groups (each group contains 1 word). As 500 groups represents 2 words per group, we focused on 1 to 500 groups (between 501 and 1000 groups, some groups contains only 1 word which is not very interesting). Our algorithm generates randomly those groups: each time, the words' database is mixed, and then as long as there are words left, each word is associated with a group. The signature of each group is the sum of every trace for each word it contains.

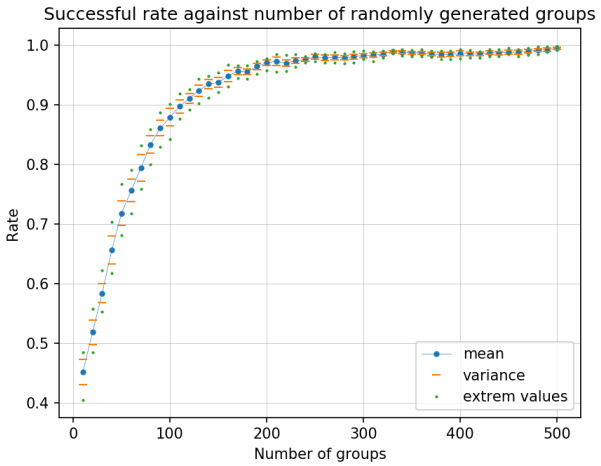

Figure 5. Plot of successful rates against number of groups (generated randomly). Close world.

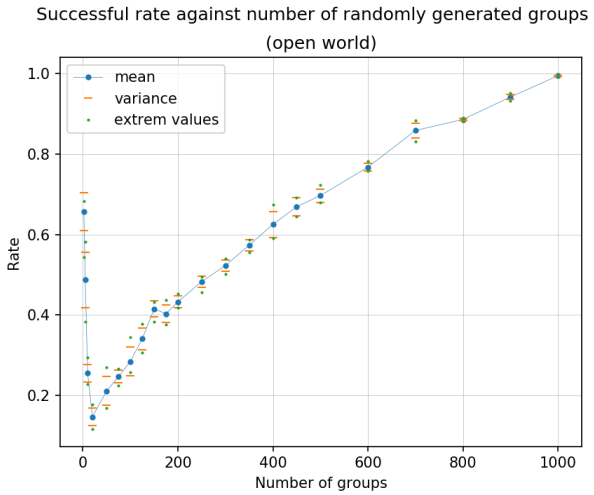

Figure 6. Plot of successful rate against number of groups (generated with K-medoids method). Close World.

The results are very promising. To quantify the completion of the method, the rate of successful classification was introduced. It represents the ratio between the number of words classified in the right group and the total number of tested words. The experiment was iterated several times to be more precise by using the mean of the results for each size of groups. On Figure 5, which represents the successful rate against the number of groups, the curve increases rapidly with the number of groups, below around 150 groups. Near 500 groups, the curve is tendings toward `1.0`. That means that it is possible to decrease the number of groups without observing a bad successful rate. For example, with around 330 groups (so about 3 words in each group), the successful rate is above 99 %.

2) With K-medoids' groups

Now we know that we can merge signatures in the database into groups without deteriorate the classification engine performances, we wonder how to make better groups. We wanted to know how to have the best successful rate for every number of groups (we wanted to reach every upper green point on the Figure 5, as it represents the top values). So the K-medoids algorithm looked like to be the best solution as it allowed us to create groups where signatures are as close as possible as described in II-C2 section.

As can be seen on the Figure 6, the results are not as good as expected. The curve drops quickly with the size of each group. For example, with 500 groups (it correspond to only 2 words in each group) the successful rate is under 70%: more than 30% of blacklisted words were not detected. With 330 groups (remember that represents a very high score with randomly generated groups), the successful rate is barely above 50%, meaning that almost half of the testing traces were miss classified.

C. Results in Open World

Even though the results in Closed World with random method was satisfying, the possible applications of this method are never in Closed World. In fact, the main objective is to detect if a word is in the blacklist or not. So the words' database used for detection should be bigger than the one used for the learning phase.

In this perspective, an Open World vision needs to be considered.

First and foremost, the table figure 7 itemise the exact vocabulary of each classification metrics. The true positive words are and should be on the blacklist, the true negatives ones are and should be neutral. The false ones are when the algorithm be mistaken. The wrong classified are very marginals and negligible: it is when the algorithm classified the world as a blacklisted one but not the right one. Follow-up the terms "rates of successful" will rally the true positives and the true negatives where the "errors" will the term to rally the false positives, the false negatives and the wrong classifieds.

Reality / Classification	Blacklisted word	Neutral word
Word noticed as a blacklisted one	True Positive / Wrong Classification	False Positive
Word noticed as a neutral one	False Negative	True Negative

Figure 7. Table of the different specific metrics in Open World.

1) Parameters evaluation (alpha)

As explained in section II-B2, to determinate the classification in Open World we need to introduce a threshold. Indeed in our system, the highest score is considered as relevant only if it's reaching the threshold :

$$t = mean(L_s) + \alpha \times std(L_s)$$

with L_s the list of calculated scores.
The determination of the filtering parameter α was empirical. In previous experiments [1], alpha was fixed to **4.1** but, because of the inclusion of new parameters, we need to check the concordance to keep (or not) the same value.

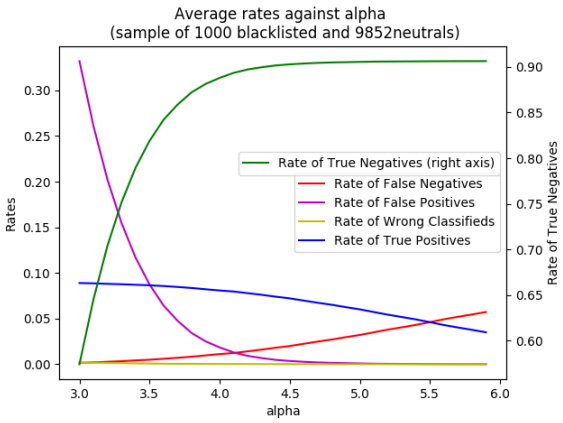

Figure 8. Plot of all the metrics against the value of alpha. Blacklist of 200 groups of 5 words.

The first approach was to consider all the metrics of classification appeared in Open World against alpha. It's the purpose of the graphic 8 established with the configuration of the 1,000 blacklisted words (regrouped in random groups of 5) and the 11,000 words for monitoring. Although the different configurations, the dynamics of the curves of the true negatives and the false positives are the same, which was easy to figure out. Actually if alpha is too low, the engine classified the word if there are a few of sameless, and if alpha is to height, the engine does not forge sameless. So alpha helps to know if the highest score stands out the mean score. If it is, the score is relevant, if not, it could be a random concomitance.

The most relevant point was the intersection between the false positives and the false negatives: the target is to have least ones of the two errors' rates. Nevertheless one of the metrics could be promoted, at the expense of the other ones: if the administrator prefere to do not have false positive for example, alpha will be chosen higher. Besides, in Open World approach, the number of true positives is low, because the proportion of blacklisted words is low: optimally the method could classify rightfully only the number of words in the blacklist, so only 10%.
The second approach was to considerate the ratio between the size of the blacklist and the size of the noise (the words not blacklisted) impact on the determination of alpha: if there is a little rate of no blacklisted (we are close to the Closed World situation), the alpha could be low, reverse, the more we have no blacklisted words possibilities, the more alpha is relevantly high. This behaviour is brought out in the figure 9.

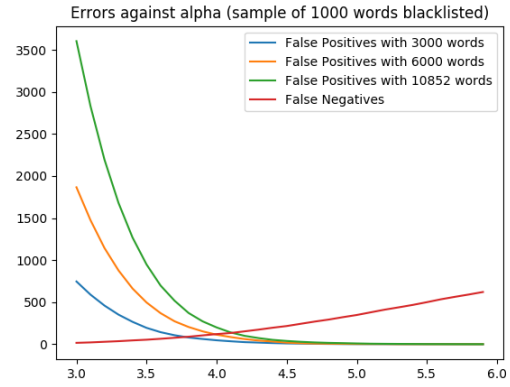

Figure 9. Plot of the false positives and negatives classification against the value of alpha for different size of confrontation's sets. Blacklist of 200 groups of 5words.

In order to confirm the choice of alpha, it seems appropriate to also show its role and impact although the ROC curve 10. This curve corresponds to the popular way of using metrics in Open World. Thanks to these three results that the chose of $\alpha = 4.1$ is pertinent: the best agreement for a good classification.

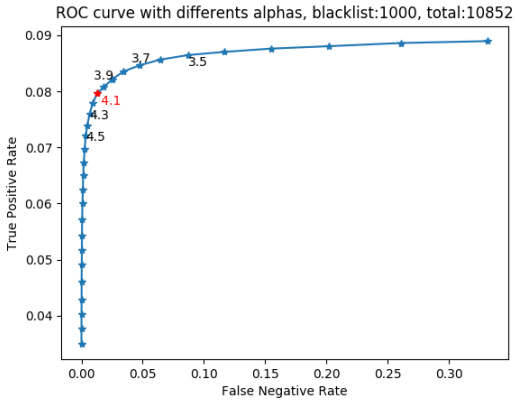

Figure 10. The ROC curve : true positives against false negatives with alpha changed. Blacklist of 200 groups of 5 words

2) With random groups

Therefore, now we know which value of α is the best, we tested our classification engine in Open World with randomly generated groups. As a reminder, false negatives corresponds to keywords mistakenly non-blacklisted by our classification engine and false positives match those mistakenly blacklisted by our algorithm. The results shaped in figure 11 show the global rate is between 90% and 99%. Nevertheless, the first part (the increasing one) is not relevant: indeed the "rate of successful" is only constitute to the true negatives. The relevant aspect is that the best rates were around 200 groups, which is corresponding to the situation of 5 words in each group. As a reminder, we chose the alpha with groups of 5 words. So, the

best configuration we find here is the one which corresponds to the tests session.

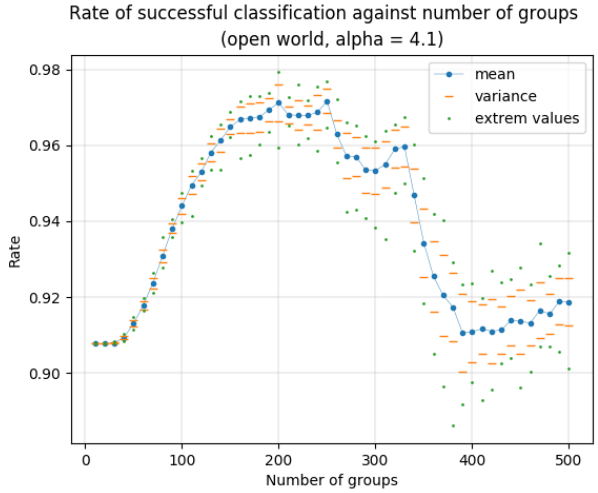

Figure 11. Rates of successful classification against number of groups. Open World. Randomly generated groups.

To corroborate this interpretation, the number of false negatives was studied. As we can see on Figure 12, the number of false negatives rationally increases with the number of groups. But there is a big step around 350 groups: if the main requirement of the system is to miss as little as possible blacklisted words, the number of groups shouldn't go over 350 groups. This curve cross-analyzed with one of the false positives Figure 13 reflect that the best condition is when the alpha constrain the number of false positives but does not blow up the false negatives. So the circumstances where it was chosen.

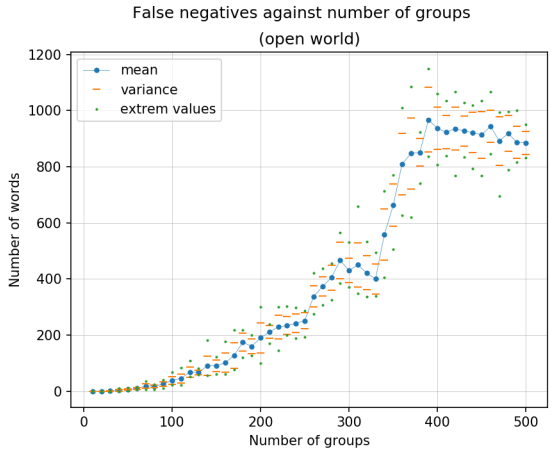

Figure 12. False negatives against number of groups. Open World. Randomly generated groups.

To conclude, the best number of groups depends on the requirements of the system. Most of the time, this number should be between 250 and 450 groups for 1,000 words. If the number of groups lower it become less relevant because there are too much wrong classification, even in Closed World.

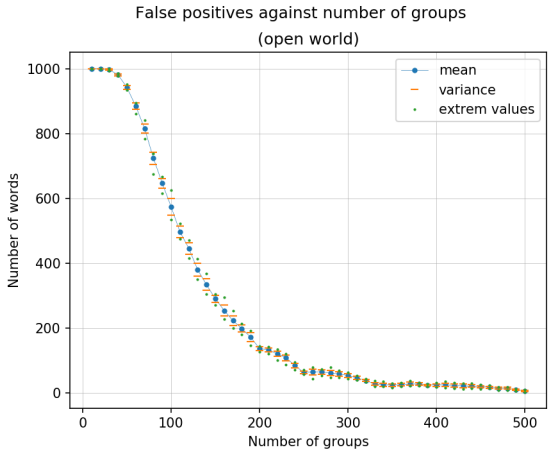

Figure 13. False positives against number of groups. Open world. Randomly generated groups.

If the number of groups is higher there are some groups with only 1 words, so there is not anymore a privacy aspect. So if the network administrator does not want to miss a blacklisted word, the number of groups should be around 500. If he can accept some omissions, he never wants to block a request that shouldn't have been blocked, a lower number of groups should be chosen. However, whatever his choice, the alpha needs to be adapted to the choice of the size of the groups.
In the case of 200 groups of 5 words, so 1,000 words in the blacklist, and 10,952 words for the monitoring step, we found, with an alpha of `4.1` a mean of rate of true negatives of 96.55% and true positives of 1.7% so a mean rate of classification of 98.23% and a mean of false positives of 0.21% and of false negatives of 0.62%. So the average rate of successful classification is good but the number of blacklisted words not classified is not sufficient if a company wants to know each time an illegitimate page load was done.

IV. Discussion

A. *Groups : privacy & efficiency*

The results have shown that regrouping words do not widely impact the comparison's score in Closed World for groups of three or four words in each group. Nevertheless, in Open World this method needs to be adapted. Indeed, regrouping words in Open World give rates classification less convincing if the parameter alpha is badly determined: where initial operation engine have rates of true positives and negatives above 99%, the regrouping engine raises rates of 97% if the alpha is chosen regarding the number of words in the group but the good classification is more due to the true negatives than the true positives.
In this way, we add a notion of privacy but we lose the device efficiency to raise all the exception if there are a blacklisted page load. Nevertheless this method could be apply with more strictly parameters. Whatever the company could not know

the exact searched keywords but could detect futures requests. However, the protocol needs to be applied with the attention of chosen parameters, particularly alpha.

B. A highly variable Alpha

As specified the chose of the parameters alpha is very important and significant regarding the classification. It depends on the size of the database against the size of the possible encounter world. It depends also on the number of the words in each groups. Indeed the results are relevant only if the alpha was chosen knowing the number of words in groups and the global size of the blacklist.
This aspect could be binding if the database needs to change regularly, but, normally it is not the goal of this tracking engine.

C. Elaboration of the groups

The Elaboration of the groups need to be done carefully.
As seems in III-B2, the successful rate when groups are elaborated with the K-medoids algorithm was particularly bad. Indeed, even in Closed World, the successful rate of classification is barely above 50% with groups of 3 words (when in the same conditions rate of the random creation was above 99%).
Those results are not at all intuitive: thought that considering the similarity will raise the performances was legitimate. Mainly the grouping of similar signatures which could be apparent to add some trace for the learning. However, the experimentation is clearly remaining a loophole in the way of thinking this approach.
Assessing the situation bring a probable explication. An overall inequality repartition of the number of words in each group was noticed. For example, when 500 groups were constituted, a huge number of them contains a single word, some of them two or three and a couple of groups have above 15 or 20 words into a unique group. Trying to split the big groups by dichotomy lead to a similar result. This pattern is intuitively a reason of the bad classification: a word in a bit groups can't be detected correctly because the "noise" of the other signatures is too height.
The future question is determining if it's possible to constitute groups considering the similarity but imposing an equal size, in this configuration, the results are, or not, better than a random method elaboration. It could also be insightful to try to created groups the most disparate as possible, maybe it could be a good way to merge without losing the performance: the noise can be easily ignored when we can identify it.

D. Sustainability of signatures

One of the biggest problem that we can face is the stability of signatures over time. With trends and news, Google Images' result pages can be different day after day, old thumbnails can be deleted and new ones can appear. Therefore, it is probably more difficult for the filtering engine to well recognise words over time.
In order to study the life expectancy of these signatures, we elaborated a list of more than 8,000 words to send to a web crawler in order to estimate the rate of change and the impact on the monitoring. This list contains many names of famous peoples, more prone to many enhancements of thumbnail regarding current events, and some common nouns from the French dictionary.
This aspect is not relevant today nevertheless the long-term goal is to do a real-time application and we will need to know the best frequency to rebuild the database of signatures of blacklisted words.

V. Conclusion

This paper details an implementation of a new framework to filter HTTPS traffic and an application in the case of an images search service. This system, which is simple as it does not require the user to install a software (or a proxy server), can be used by a network administrator in order to monitor its traffic and detect illegitimate usage. Our solution relies on web objects' sizes and a signature generation thanks to Kernel Density Estimation.
We also developed a procedure which adds a privacy aspect based on clustering words in groups. Two different methods have been explored: a random one and another based on K-medoids algorithm. The first one is relevantly effective as the successful classification rate in Closed World remains high at about 99% for three words per group and 97% in Open World with a relevant choice of parameters. However, the second method (based on K-medoids clustering) is less convincing.
In future work, we will focus on a real-time solution that will be adaptable to the variability of signatures.

Acknowledgment

We would like to thank P.O. Brissaud and J. François for their help, work and management.

We also would like to thank TELECOM Nancy (Université de Lorraine) for giving us the opportunity to do research work such as this one.

References

[1] *Tracking HTTPS Service Use: an Application to an Image Search Engine*, Pierre-Olivier BRISSAUD, Jérôme FRANÇOIS, Isabelle CHRISMENT, Thibault CHOLEZ and Olivier BETTAN

[2] *Algorithme des K-Medoïdes, Classification Automatique - Méthode de réallocation*, Ricco RAKOTOMALALA
http://eric.univ-lyon2.fr/~ricco/cours/slides/classif_k_medoides.pdf

[3] *Kernel Density Estimation*
https://en.wikipedia.org/wiki/Kernel_density_estimation

[4] *K-medoids algorithm*
https://en.wikipedia.org/wiki/K-medoids

[5] *HTTPS encryption on the web*
https://transparencyreport.google.com/https/overview?hl=en