What are you Googling? - Inferring search type information through a statistical classifier

Alfonso Iacovazzi, Andrea Baiocchi, Ludovico Bettini
DIET - Department of Information Engineering, Electronics and Telecommunications
"Sapienza" University of Rome, Rome, Italy
alfonso.iacovazzi@uniroma1.it, andrea.baiocchi@uniroma1.it, ludovico.bettini@gmail.com

***Abstract*—Privacy in communications calls primarily for information flow encryption. Packet traffic flows privacy breaches have been widely demonstrated in point-to-point communications due to information leakage from observable traffic features, like packet length, timestamp, direction. We address a point-to multipoint system, namely a Content Delivery Network, where user clients maintain and use connections with a number of servers. Specifically, we address Google search services: they are conveyed by TLS connections, by using https, either from within user accounts or even without logging as a Google services user. Https is provided to protect communications privacy. Yet, we show that by collecting the encrypted traffic and extracting simple features related to traffic activity and possibly the amount of data sent by servers to clients, effective classifiers of user activity can be realized. Specifically, we are able to distinguish which type of search a user is carrying out, among a given set of alternatives (text, images, maps, video, video on YouTube, news) with average success rates that can exceed 90%.**

***Index Terms*—google; content delivery network; side-channel information leaks; packet feature analysis.**

I. Introduction

A vast amount of research and experiments have pointed out that supposedly secure channels, i.e., encrypted connections carrying packet traffic flows, leak information through the observation of features like the sequence of packet lengths, times, addresses, enabling privacy breach, e.g., see the surveys in [1]–[6]. All of these packet features are visible in spite of encryption of payloads and they can give away information about users' activities. Such attributes are referred to as *side-channel information*. Side-channel information leaks have been extensively studied for a decade, in the context of secure shell (SSH) [7], video-streaming [8], voice-over-IP (VoIP) [9], [10], web browsing, e.g., see [11]–[13]. Already in 1996, Wagner et al. [14] have shown that the HTTP get request, encrypted using SSL, can provide information about the content of the web page exploiting the length of the URL. As a matter of example, it has even been shown that the analysis of the sequence of variable sizes of data chunks in a VoIP conversation flow allows partial transcripts of the content to be obtained [10].

The exceptional growth of social and web 2.0 services has boosted the deployment of Content Delivery Network (CDN) technology which provides contents replicated all over the globe and offers services in a collaborative way. One common approach to protect sensible data is the encryption of packet payload. This is usually accomplished in CDN by means of SSL/TLS connections among user clients and CDN servers, e.g., by using HTTPS.

In this work we study side-channel information leaks in a CDN environment. We have developed a classification system able to gather information about user activities in a CDN environment from encrypted traffic flow aggregates. Our hypothesis is that the interaction between a user and a service provider exploiting a CDN brings enough information to infer the particular service requested by the client even when the data are encrypted, and this is due to the distributed approach CDNs rely on. We target Google as the reference CDN and our task is to develop a classifier able to reveal whether a client is using Google to search for an address on a map, an image, a video on Google video, a video on Youtube, news from all over the world, or simply a word.

Privacy attacks in google search have already been discussed in [15] where Chen et al. show that it is possible to catch the history of queries submitted to a search engine, by analysing the sequence of packet lengths in ciphered wireless communication. Google services have already been considered as subject of classification techniques (along with other services such as Facebook, Microsoft Windows Update, etc.) in [16], [17], where Authors try to classify HTTP flows using only IP address of the servers hosting the corresponding content.

The contribution of this paper is twofold: i) the privacy attack scenario in a CDN, namely Google search one, is considered and it is shown that effective classification of search types can be performed by collecting observable features of the encrypted traffic; ii) a new global feature is considered in the classification algorithm, by leveraging the interaction between a client and a set of servers in a CDN.

As for the rest of the paper, Section II outlines the architecture of Google CDN. Section III gives a precise statement of the privacy problem we raise and describes the observed traffic features. In Section IV we give a brief account of the way the dataset for our experiments has been collected. Performance results are reported in Section V. Final remarks and outlook at future work is sketched in Section VI.

II. Google CDN architecture overview

A content delivery network (CDN) is an interconnected system of network elements that provides Web contents to end-users with high performance and high availability. This task is achieved by replicating contents on multiple servers

978-1-4799-1353-4/13/$31.00 ©2013 IEEE

(surrogate servers), spread all over the globe and directing them to users based on proximity.

Google can be considered as a CDN, in fact it is probably composed of more than 800,000 servers distributed worldwide (even though Google has never unveiled how many they are!). The servers are divided into six different groups each performing its own task: Web servers, document servers, index servers, spell check servers, advertisement servers, and Googlebot servers. Further details about the google' CDN architecture and the functionalities carried out by the different types of servers can be found in [18], [19].

Beyond the typical search functionality, Google provides a secure search option through the HTTPS protocol. With the protocol HTTPS, the web pages are entirely encapsulated into a SSL/TLS connection. Encryption covers HTTP headers, query parameters, requested URL, and cookies (which could contain sensitive information about the user). However, SSL/TLS cannot protect client and server IP addresses and port numbers, which are necessarily part of the TCP/IP headers. This means that an observer can still infer IP addresses and port numbers of the web servers that a client is communicating with as well as the amount of data transferred, the packet size and timing information of the communication. Moreover, in a context of CDNs in which the client communicate with multiple servers, the information observed over different SSL/TLS connections might be combined to elicit as much information about the user.

From a practical point of view, when a user goes on https://www.google.com, a certain number of TCP connections are established. Then a user can require some type of service and based on it, a number of new connections may be opened and some of the previously established flows may be closed.

In general, not all established connections are actually used to transfer data, yet most of them are kept open for a relatively long time, often up to the closing of the browser. We expect that the established connections have no easily observable relation with the user activity. However, we hypothesize that observing which connections carry how much data over a time interval yields sensitive information on user activity.

III. Privacy problem statement

Let $\mathcal{C}$ be a client and $\mathcal{S}_j$, $j = 1, \ldots, N$, servers that the client pulls data from. Connections are established between the client and each server through the public Internet. The privacy attacker $\mathcal{O}$ can collect any traffic carried by those connections. Let $m_j(t)$ be a message sent by $\mathcal{S}_j$ to $\mathcal{C}$ at time t. By message we mean packets carried through the connection with application level payload. We neglect mere control packets or retransmitted packets. As a matter of example, in case of TCP connections, a data segment from the server to the client with payload length > 0 is a message, while an ACK segment with zero payload is not considered a message, as well as a retransmitted segment (denoted by a sequence number smaller or equal to a sequence number already sent). We disregard packets other than those carrying fresh application level data, since these ones are relevant to application behavior, while others are essentially the product of internal state evolution of lower layers protocols.

The attacker $\mathcal{O}$ can collect all messages of the connections with the N servers in the time interval $[t_0-\Delta, t_0+\Delta]$ (*service time*), where t_0 is a reference time and Δ is a positive number. The choice of Δ has to strike a balance between the amount of information that can be collected (the more, the more effective the classification attack to privacy) and the complexity in terms of processing/time/memory of the attack. Formally, the traffic data base of the attacker is made up of $\mathcal{T} = \{m_j(t) | t \in [t_0 - \Delta, t_0 + \Delta], j = 1, \ldots, N\}$. From those raw traffic data, features are extracted to feed the classification algorithms.

The target of $\mathcal{O}$ is to identify the type of activity the client is doing among a set of pre-defined alternatives. Our purpose is to identify whether a user is performing a text, image, map or news search or if she is looking for a video either on youtube or on Google video. We call *search types* these alternatives, so that the classification label set is $\mathcal{A} = \{text,\ images,\ maps,\ news,\ video,,\ youtube\}$. It comprises $K = 6$ classes. Then $\mathcal{O}$ defines a classification algorithm that takes $\mathcal{T}$ as input and outputs an element of $\mathcal{A}$.

This classification is not trivial. Clients contact a large number of servers for each search, so as to speed up the response, and same servers are contacted for different search types. Let $u_{i,j}(r)$ be a binary variable equal to 1 if in the r-th search of type i server $\mathcal{S}_j$ has sent any message to the client during the observation interval. For a given search type i the utilization of server $\mathcal{S}_j$ is defined $f_i(j) = \sum_{r=1}^{R_i} u_{i,j}(r)/R_i$, where R_i is the number of queries of type i. The vectors $\mathbf{f}_i = [f_i(1), \ldots, f_i(N)]$ estimate the probability distribution of the server utilization of the i-th search type.

Figure 1 shows the utilization distribution of the most common IP addresses that have been found in our measured traffic traces. The distribution for different search types have been stacked one onto another and colored. A glimpse to the figure reveals that the distributions are not uniform, yet they are not decoupled, i.e., different search type share a large part of the servers.

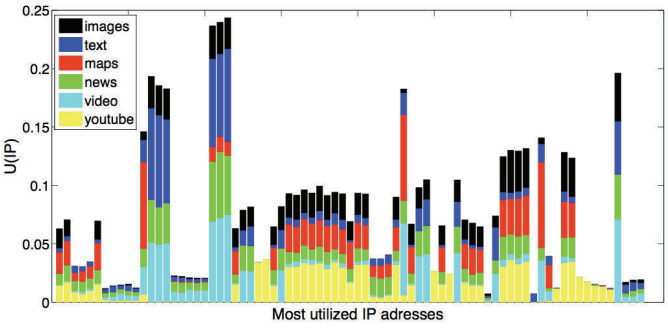

Fig. 1. Statistical distribution of IP addresses usage in Google searches.

A quantitative confirmation can be obtained by considering the correlations of the distributions, i.e., $\rho_{r,s} = \mathbf{f}_r \mathbf{f}_s^T / (\|\mathbf{f}_r\| \|\mathbf{f}_s\|)$, where a superscript T denotes transposition and $\|\mathbf{x}\|$ is the Euclidean norm of vector $\mathbf{x}$. The numerical

values obtained for the matrix formed by the $\rho_{r,s}$ from the trace we have collected are shown in Table I.

TABLE I
NORMALIZED CROSS-CORRELATIONS AMONG THE ESTIMATED DISTRIBUTIONS OF GOOGLE SERVER USAGE OF THE SIX SEARCH TYPES.

	images	*maps*	*text*	*news*	*video*	*youtube*
images	1	0.619	0.862	0.652	0.631	0.714
maps		1	0.485	0.239	0.345	0.650
text			1	0.908	0.905	0.399
news				1	0.918	0.130
video					1	0.084
youtube						1

Since the vectors $\mathbf{f}_r$ are non negative, their normalized cross-correlations range between 0 (completely decoupled distributions) and 1 (identical distributions). The values in Table I highlight that most of the search types have highly correlated server usage distributions, yet non identical. This is an indications that there is no trivially simple way of associating a partition of the set of servers to the different search types, yet some information useful to identify the type of search must lie in the server usage pattern of the clients. This is exploited systematically in this work.

The traffic descriptors associated to the n-th IP address of the Google servers, $n = 1, \ldots, N$, and to the given observation time interval are:

- The number of active traffic flows C_n. There might be many connections open, but not exchanging data. We are interested only in *active* connections, where active means sending at least one message in the observation time.
- The amount of data bytes B_n received by the clien.
- The number of messages P_n received by the client.

Statistical classification is a typical case of supervised learning, where the dataset is split in a training set Q_{tr} which is used to build the classification model, and a test set Q_{ts}, to check the performance of the model. We chose to consider four supervised machine learning: 1) the simple probabilistic classifier "Naïve Bayes" based on the Bayes' theorem which assumes a strong statistical independence among the features; 2) a classifier based on the clustering K-means algorithm; 3) a machine learning able to build and use Multinomial Logistic Regression [20]; 4) an algorithm for the construction of random forests as combination of decision trees, developed by Breiman and described in [21]. Implementations of these algorithms are provided by WEKA (Waikato Environment for Knowledge Analysis) [22], a suite of machine learning software written in Java and developed at the University of Waikato, New Zealand.

IV. DATASET CREATION

Google offers a full set of functionalities like the simple text search engine, the email client Gmail, the new Google Drive to store and share all kind of documents or the video-sharing website Youtube. It also allows searching for images, maps, videos, scholar documents and news from all over the world.

To get the final dataset that will be given as input to the machine learning for classification, three phases have been addressed: i) Traffic Generation; ii) Features extraction; iii) Dataset creation and normalization. In the following subsections the three steps are described in detail.

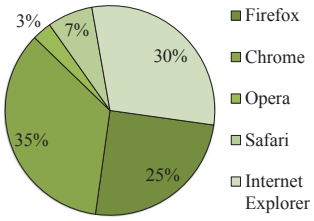

Fig. 2. Probability density function chosen for the browser selection (based on browser usage statistics at Nov 2012).

A. Traffic Generation

Since the considered traffic is ciphered it was not possible to use real traffic traces because of the lack of ground truth to train and test the classifier, so it has been artificially generated. Google searches are simulated into web browsing activity. The 5 most common browsers (Firefox, Google Chrome, Safari, Opera, and Internet Explorer) are selected in a random way based on the discrete distribution shown in Fig. 2.

A search type is randomly selected. Then, the script either types in the url bar the Google home page for maps (e.g., https://maps.google.de) or for images (https://www.google.de/imghp) or types https://www.google.de. After that, one of the icons corresponding to links on the black bar at the top of the page is selected. One or more searches (of the same type) are carried out on the same window, either on the same tab or on a new one. After a certain time, the browser is closed and a new iteration starts. The activation of the secure search option simply requires urls to start with *https* instead of *http*.

The words/senteces to be Googled are taken from a manually created database containing several lists. Details of the database are shown in Table II. Each list is relevant to a certain class, for example there is an address list, a store list and a places list for Google maps, a singer list and a song list for Google Video and Youtube and so on. Depending on the search type, each list is assigned a probability according to Figure 3. After selecting a list according to the assigned probability, the word to be Googled is randomly chosen.

To make the traces as general as possible, the domain search (e.g., .com, .it, .de, .com.br) has been randomly chosen from a domain list of 30 items at each iteration. Also the interface language and the content location has been changed for each search. The traffic has been generated from three different locations in three different European cities between June and August 2012. For each class, or search type, 2100 traces have been collected, 700 from each location. Half of the traces from each city are generated with a Google account logged in.

Traffic traces are collected by means of the packet sniffer tool tshark on the client side.

TABLE II
DESCRIPTION OF THE RESEARCH DATABASE.

List	Description
Addresses	2000 addresses from all over the world.
Celebrities	2000 names of politicians, figures of interest, actors,...
Commercial	A list of 1000 brands.
English dictionary	The dictionary available in any Linux distributions, contains more than 74000 words.
Extra	A short list of extra functions implemented by Google (e.g., cities forecast, currency change,...)
Geography	A list of 600 rivers, seas, mountains, museums, and monuments.
Places	A list of 600 cities, states, islands, and provinces.
Stores	It contains the names of restaurants, hotels, and stores from all over the world.
Singer	2000 of the famous singers of all times.
Songs	500 of the famous songs of all times.
Sport	500 entries chosen from teams, sports, coaches and players.

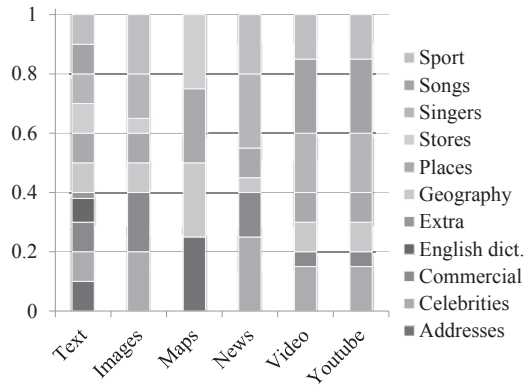

Fig. 3. Probability density function chosen for the list selection.

B. Features extraction

Starting from Google address space, we inferred, by inspecting all the traces obtained with the traffic generator, that the total number of Google IP addresses contacted by the clients is $N = 447$.

The features must be collected within the observation time $[t_0 - \Delta, t_0 + \Delta]$ where t_0 is matched to the instant in which the user types the enter key to send the search query. To be able to determine t_0, we have kept a list of these instants trace by trace during the generation of the traffic.

At the end of the feature extraction we had a dataset composed of $I = 12600$ entries, one for each query. Each entry is a vector of $3N = 3 \cdot 447$ features. The features vector extracted form the i-th query is ($i = 1, \ldots, I$):

$$\mathbf{x}_i = (C_1^{(i)}, \ldots, C_N^{(i)}, B_1^{(i)}, \ldots, B_N^{(i)}, P_1^{(i)}, \ldots, P_N^{(i)}) \quad (1)$$

C. Dataset creation and normalization

In this work several datasets have been used to classify the search type. All of them are subsets of the main dataset.

In each dataset we considered only addresses used in at least 300 instances of search, resulting in a subset of only 50 IP addresses out of the overall 447 ones. This reduction is useful to improve classifier performance as the most popular servers offer better differentiations of the search types. Moreover, classification is not feasible with all the techniques if the full dataset is used. For example the multinomial logistic classifier does not accomplish the model evaluation with so many features in a reasonable time. The full dataset can be thus reduced to a more efficient dataset with only $3{\cdot}50$ features.

Each feature vector $\mathbf{x}_i$ can be split into three subvectors, one for each feature category: number of active connections C, amount of bytes downloaded B and number of messages P. The subvectors are collected in three basic component datasets, corresponding to the three kind of features, D_C, D_B and D_P, respectively. These dataset have been normalized separately. L2 normalization has been applied instance by instance. If $\mathbf{x}_i^{(\ell)} \in D_\ell$, then we define the normalized sub vector $\mathbf{y}_i^{(\ell)} = \mathbf{x}_i^{(\ell)} / \|\mathbf{x}_i^{(\ell)}\|_2$, for $\ell = C, B, P$.

Search type classification has been tested with the algorithms presented in Section III and with all the possible combinations of the three sub-datasets D_C, D_B and D_P. Numerical results are presented in the next Section.

D. 10-fold cross-validation

In order to guarantee a rigorous evaluation of our methodology, 10-fold cross validation was performed. Cross-validation is a technique that avoids both overfitting problems and problems due to asymmetric sampling (and thus affected by bias) of the training dataset, typical of the division of the dataset into only two parts (i.e. training and test datasets). The k-fold cross-validation consist in patitioning the total dataset into k subsets. The classification experiment is repeated k times; at the i-th iteration ($i = 1, \ldots, k$), the i-th subset is selected as test set, while the other $k - 1$ are used as training set. At the end, the results of all iterations are averaged.

V. PERFORMANCE EVALUATION

Starting from the confusion matrix H, whose (i, j)-th element represents the number of instances of class i classified as belonging to class j, we can define the following parameters:

- True Positive (TP): given class k, TP_k is the ratio between the instances correctly classified as class k, and the total number of instances in that class.

$$TP_k = h_{kk} \Big/ \sum_{j=1}^{K} h_{kj}$$

- False Positive (FP): FP_k is the ratio between the instances incorrectly classified as class k, and the total number of instances of all classes except class k.

$$FP_k = \sum_{i \neq k} h_{ik} \Big/ \sum_{i \neq k} \sum_{j=1}^{K} h_{ij}$$

- Precision: given a class k, precision p_k is defined as

$$p_k = \frac{TP_k}{(TP_k + FP_k)}$$

  Precision gives us a measure of reliability of the classifier for a specific class.
- Accuracy: is the total number of correctly classified instances divided by the number of instances in the test set S_{ts}.

$$acc = \sum_{i=1}^{K} h_{ii} \Big/ \sum_{i=1}^{K} \sum_{j=1}^{K} h_{ij}$$

Two scenarios have been considered to evaluate the performance of the classifiers. In the first one the full dataset with all six classes is used (text, images, maps, news, video, and youtube). The second scenario takes into account only the four major services offered by Google (text, images, maps, and youtube). We also introduce a notation to highlight which features are used to describe each search instance. The three reference features are: number of active connections, amount of byte transferred from each server to the client and number of packets received by the client from each server. The dataset denoted with "conn-pkt-byte" is the complete one, with all features for each search instance. The dataset "conn-pkt" contains information about the number of connections and the number of packets received. Other similar notations are used to denote dataset based only on a subset of features.

A. All Classes Scenario

The performance results given in this Section are based on the full dataset comprising all six types of search (six classes).

To evaluate the contribution of each feature to the classification process all 7 possible combinations of up to three features are considered. Table III shows the results for each dataset and for each machine learning algorithm used in this work.

TABLE III
ACCURACY OVER 7 DATASETS FOR ALL 6 CLASSES.

	Accuracy (%)			
Dataset	K-Means	Naïve Bayes	Logistic	Rand. For.
conn	35.54	60.85	77.67	77.82
pkt	35.51	56.99	72.59	78.09
byte	37.40	56.63	72.19	77.82
conn-pkt	37.70	67.77	82.78	84.64
conn-byte	36.70	66.56	82.69	85.26
byte-pkt	35.62	59.90	76.51	83.87
conn-byte-pkt	36.46	67.72	81.57	87.06

K-Means is very inefficient in this scenario, since the accuracy is barely doubled over random guessing that would provide 16.67% accuracy. Also the Naïve Bayes classifier does not perform with a satisfying level of accuracy, and this is due to the low information brought by the marginal distributions of the features. The other two classifiers, Logistic and Random Forest, perform much better in terms of accuracy since they can reach 80%. This is a significant result. In fact our hypothesis is confirmed: there is some information leaking due to the distributed nature of the Google CDN. In spite of their significance, still the results point out that there is a good margin of improvement possible.

TABLE IV
PER-CLASS INFORMATION ABOUT PERFORMANCES OF LOGISTIC AND RANDOM FOREST CLASSIFIERS ON DATASET "CONNECTION".

	Logistic			Random Forest		
class	TP	FP	$Precision$	TP	FP	$Precision$
images	0.785	0.011	0.934	0.807	0.029	0.850
maps	0.891	0.016	0.917	0.908	0.017	0.913
text	0.601	0.071	0.628	0.658	0.072	0.646
news	0.829	0.073	0.694	0.759	0.052	0.744
video	0.598	0.049	0.709	0.577	0.050	0.699
youtube	0.956	0.048	0.801	0.960	0.046	0.806

For example, considering the performances of Logistic and Random Forest classifiers on dataset D_1, we can see from table IV that the classes maps and youtube are classified with high accuracy while the other classes not, and this inequality decreases the total accuracy of the classifier.

The low accuracy of some classes can be explained by looking at the confusion matrix in Table V for the Random Forest classifier. Reading the third line, we can see that almost one third of text searches is misclassified between news and video; the fifth line reports that one third of video searches are classified as text ones. While the classes news and video mingle with each other, also the classes images and news have an insufficient accuracy, but the misclassified instances are spread over all the other classes.

TABLE V
CONFUSION MATRIX FOR RANDOM FOREST CLASSIFIER ON DATASET "CONN".

Class	*images*	*maps*	*text*	*news*	*video*	*youtube*
images	1695	44	16	208	33	104
maps	58	1906	4	40	23	69
text	64	35	1382	180	370	69
news	122	39	159	1593	89	98
video	30	22	577	112	1212	147
youtube	26	41	1	9	6	2017

B. 4 Classes Scenario

A simplified scenario is obtained by considering 4 classes: text, video, maps, and youtube.

Table VI shows the results, in terms of accuracy, obtained with the 4 classification techniques applied on the new datasets. We can see that K-Means algorithm does not perform in a satisfying manner yet, leading to a 50% of the traces being correctly classified, which still represents only a factor of two improvement over random guessing (25%). On the other side, the simple Naïve Bayes classifier performs much better than on the 6 classes dataset, in fact it is now possible to reach almost a 90% accuracy considering the number of active connections combined with the number of packets or the complete dataset.

TABLE VI
ACCURACY OVER 7 DATASETS FOR 4 CLASSES SCENARIO.

	Accuracy (%)			
Dataset	K-Means	Naïve Bayes	Logistic	Rand. For.
conn	51.00	82.32	91.18	91.13
pkt	51.17	85.63	92.79	93.13
byte	52.21	85.63	92.35	93.05
conn-pkt	53.64	88.81	94.79	95.04
conn-byte	51.29	88.82	94.79	95.10
byte-pkt	51.76	88.77	94.75	95.19
conn-byte-pkt	52.28	89.76	95.23	95.81

The highest results are obtained with Logistic and Random Forest classifiers leading to a maximum accuracy of 95.8%.

TABLE VII
PER-CLASS PERFORMANCE OF LOGISTIC AND RANDOM FOREST CLASSIFIERS ON DATASET "CONN" OVER 4 CLASSES.

	Logistic			Random Forest		
class	TP	FP	$Precision$	TP	FP	$Precision$
images	0.857	0.022	0.929	0.887	0.034	0.896
maps	0.903	0.022	0.932	0.920	0.022	0.932
text	0.913	0.030	0.911	0.870	0.020	0.936
youtube	0.974	0.044	0.880	0.968	0.042	0.885

TABLE VIII
PER-CLASS PERFORMANCE OF LOGISTIC AND RANDOM FOREST CLASSIFIERS ON DATASET "CONN-BYTE-PKT" OVER 4 CLASSES.

	Logistic			Random Forest		
class	TP	FP	$Precision$	TP	FP	$Precision$
images	0.938	0.009	0.971	0.948	0.010	0.969
maps	0.950	0.011	0.967	0.970	0.011	0.967
text	0.958	0.022	0.936	0.934	0.007	0.980
youtube	0.963	0.022	0.937	0.981	0.028	0.920

Tables VII, and VIII show per-class performance of the best classifiers for two different datasets in terms of accuracy and precision. We can see how the accuracy of each class is high enough for a satisfying classification. Also the precision of each class presents value in a satisfying range $(87\% - 99\%)$, meaning that if a new unlabeled instance is classified as class c_k, there is a chance p_k that it really belongs to that class. Precision is a very important parameter, since it gives a reliability measure of classification. A class with high accuracy and low precision means that instances of that class are correctly classified, but also instances belonging to different classes are assigned to that class.

C. Varying the number of features

Since now we have only used datasets with 50, 100, or 150 attributes. To complete our analysis we have studied the performances of the classification system varying the number of features of a dataset. Given a set of features, two ranking algorithm have been used to select and sort the most relevant attributes: *Information Gain Attribute Evaluator (InfoGain)* and *Correlation-based Feature Subset Selection (CFS)* [23]. The output of these algorithms is a list of attributes sorted by the two criteria seen above. Then, one of the algorithms has produced a list of ordinated attributes, the N features with the highest rank are selected. Then the two most performing classifiers, Logistic and Random Forest, are tested varying N.

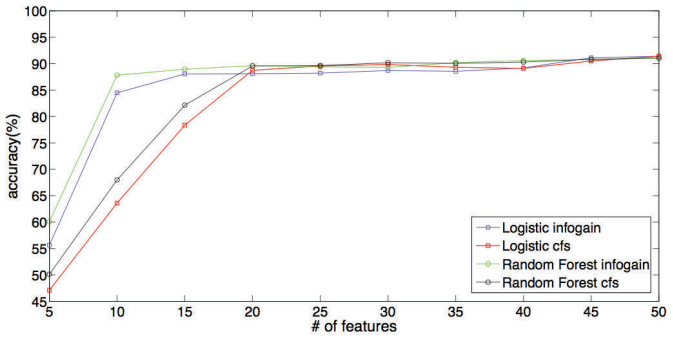

Fig. 4. Accuracy of Logistic and Random Forest classifiers over "connection" dataset varying the number of features from 5 to 50

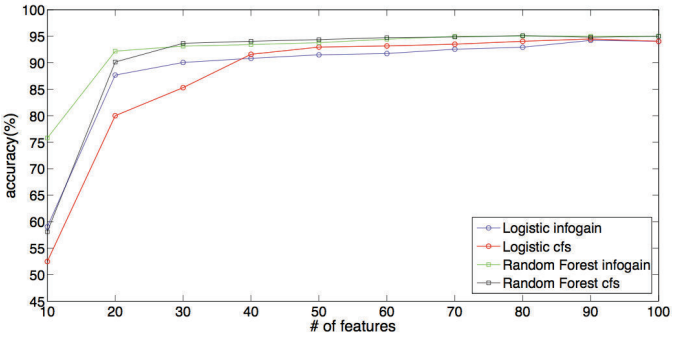

Fig. 5. Accuracy of Logistic and Random Forest classifiers over "connection-byte" dataset varying the number of features from 10 to 100

Figure 4 shows the accuracy of the two classifiers applied to the dataset "connection". We perform our test starting from the best 5 features selected by the two ranking algorithms and encasing this number by 5 until reaching 50. As we can see until 20 features are selected the accuracy is very low, especially when features are ranked with the CFS algorithm, in fact InfoGain attributes yet reach a good accuracy $(85\% - 90\%)$ with 15 features. Between 20 and 40 the accuracy is high enough but is not stable, we can also see how the two ranking algorithm select attributes giving the same accuracy results. After 40, accuracy is both high and stable. This figure tells us that 50 features assure high accuracy, but with approximately 20 features we can get high performance. This result leads to the conclusion that an attacker needs to observe only flows to a small number of servers to make a classification decision.

Figure 5 shows the accuracy of the two classifiers applied to the dataset "connection-byte". Here the number N of features

vary from 10 to 100. In this case we can see how the two ranking algorithm performs in a similar way, except for the Logistic with attributes selected with CFS until 40. Random Forest gives high accuracy starting from 60 features, while for Logistic we have to consider at least 90 attributes to obtain the same results.

VI. Conclusions

We have presented a technique for inferring search type information in the Google Content Delivery Network environment. This technique is based on statistical classification of traffic flows, by exploiting only the information brought by features observable during the search operation: number of SSL/TLS active connections, number of packets received, and amount of data downloaded.

The tests show how it is possible to classify with high accuracy traffic encrypted with the SSL protocol with a classifier exploiting multinomial logistic regression and with the decision tree based Random Forest classifier. These results demonstrate that the considered features bear an information leakage due to both the distributed nature of a CDN and the interaction between the client and more servers.

Future work aims at widening the analysis of the Google environment, by extending the number of classes to others type of user activity. For example, it would be possible to add a gmail class or a Google drive class. Also we plan to add a class "unknown" to the classification system to assess the problem of distinguishing when and if the user is performing activities considered in the classification problem as opposed to other non considered activities, encompassed under the umbrella class "unknown". Our approach could be extended in order to identify privacy breaking in other CDN environment different from Google (e.g. Akamai).

References

[1] T. Nguyen and G. Armitage, "A survey of techniques for internet traffic classification using machine learning," *Communications Surveys Tutorials, IEEE*, vol. 10, no. 4, pp. 56–76, quarter 2008.

[2] H. Kim, K. C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, and K. Lee, "Internet traffic classification demystified: myths, caveats, and the best practices," in *CoNEXT*, 2008, p. 11.

[3] A. C. Callado, C. A. Kamienski, G. Szabo, B. P. Gero, J. Kelner, S. F. L. Fernandes, and D. F. H. Sadok, "A Survey on Internet Traffic Identification," *IEEE Communications Surveys and Tutorials*, vol. 11, no. 3, pp. 37–52, 2009.

[4] M. Mellia, A. Pescapè, and L. Salgarelli, "Traffic classification and its applications to modern networks," *Computer Networks*, vol. 53, no. 6, pp. 759–760, Apr. 2009.

[5] Y. sup Lim, H. Kim, J. Jeong, C. kwon Kim, T. T. Kwon, and Y. Choi, "Internet traffic classification demystified: on the sources of the discriminative power," in *CoNEXT*, 2010, p. 9.

[6] A. Dainotti, A. Pescapè, and K. Claffy, "Issues and future directions in traffic classification," *Network, IEEE*, vol. 26, no. 1, pp. 35–40, January-February 2012.

[7] D. Song, D. Wagner, and X. Tian, "Timing analysis of keystrokes and timing attacks on ssh," in *10th USENIX Security Symposium*, vol. 2, 2001, p. 3.

[8] T. Saponas, J. Lester, C. Hartung, S. Agarwal, T. Kohno *et al.*, "Devices that tell on you: Privacy trends in consumer ubiquitous computing," in *Usenix Security*, vol. 3, no. 3.6, 2007, p. 3.

[9] C. Wright, L. Ballard, S. Coull, F. Monrose, and G. Masson, "Spot me if you can: Uncovering spoken phrases in encrypted voip conversations," in *IEEE Symposium on Security and Privacy*. IEEE, 2008, pp. 35–49.

[10] A. White, A. Matthews, K. Snow, and F. Monrose, "Phonotactic reconstruction of encrypted voip conversations: Hookt on fon-iks," in *IEEE Symposium on Security and Privacy*. IEEE, 2011, pp. 3–18.

[11] A. Hintz, "Fingerprinting websites using traffic analysis," in *Privacy Enhancing Technologies*. Springer, 2003, pp. 229–233.

[12] Q. Sun, D. R. Simon, Y.-M. Wang, W. Russell, V. N. Padmanabhan, and L. Qiu, "Statistical Identification of Encrypted Web Browsing Traffic," in *IEEE Symposium on Security and Privacy*, 2002, pp. 19–30.

[13] G. D. Bissias, M. Liberatore, D. Jensen, and B. N. Levine, "Privacy Vulnerabilities in Encrypted HTTP Streams," in *Privacy Enhancing Technologies*, 2005, pp. 1–11.

[14] D. Wagner and B. Schneier, "Analysis of the SSL 3.0 protocol," in *Proceedings of the 2nd conference on Proceedings of the Second USENIX Workshop on Electronic Commerce - Volume 2*, ser. WOEC'96. Berkeley, CA, USA: USENIX Association, 1996, pp. 4–4. [Online]. Available: http://dl.acm.org/citation.cfm?id=1267167.1267171

[15] S. Chen, R. Wang, X. F. Wang, and K. Zhang, "Side-Channel Leaks in Web Applications: A Reality Today, a Challenge Tomorrow," in *IEEE Symposium on Security and Privacy*, 2010, pp. 191–206.

[16] P. Casas, P. Fiadino, and A. Bär, "IP Mining: Extracting Knowledge from the Dynamics of the Internet Addressing Space," in *International Teletraffic Congress*, 2013.

[17] ——, "Mini-ipc: A minimalist approach for http traffic classification using ip address," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International*, 2013.

[18] L. A. Barroso, J. Dean, and U. Holzle, "Web search for a planet: The google cluster architecture," *Micro*, vol. 23, no. 2, pp. 22–28, 2003.

[19] F. Reid, *Network programming in. NET: C# & Visual Basic. NET*. Access Online via Elsevier, 2004.

[20] S. Le Cessie and J. Van Houwelingen, "Ridge estimators in logistic regression," *Applied statistics*, pp. 191–201, 1992.

[21] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[22] "Weka 3: Data Mining Software in Java," http://www.cs.waikato.ac.nz/ml/weka/.

[23] M. Hall, "Correlation-based feature selection for machine learning," Ph.D. dissertation, The University of Waikato, 1999.