Aiello, M., Mongelli, M., Cambiaso, E., & Papaleo, G. (2016). Profiling DNS tunneling attacks with PCA and mutual information. Logic Journal of the IGPL, 24(6), 957-970. DOI: 10.1093/jigpal/jzw056
https://academic.oup.com/jigpal/article-abstract/24/6/957/2662434

Profiling DNS Tunneling Attacks with PCA and Mutual Information

MAURIZIO AIELLO, *National Research Council, CNR-IEIIT - via De Marini, 6 - 16149 - Genoa, Italy*

MAURIZIO MONGELLI, *National Research Council, CNR-IEIIT - via De Marini, 6 - 16149 - Genoa, Italy*

ENRICO CAMBIASO, *National Research Council, CNR-IEIIT - via De Marini, 6 - 16149 - Genoa, Italy; Università degli Studi di Genova, DIBRIS - via Dodecaneso, 35 - 16146 - Genoa, Italy*

GIANLUCA PAPALEO, *National Research Council, CNR-IEIIT - via De Marini, 6 - 16149 - Genoa, Italy*

Abstract

The use of covert-channel methods to bypass security policies or leak sensitive data has increased in the last years. Malicious users neutralize security restriction through protocol encapsulation, tunneling peer-to-peer, chat, or HTTP packets into allowed protocols such as DNS or HTTP. In this paper, we propose an innovative profiling system for DNS tunnels that is based on Principal Component Analysis and Mutual Information. Results from experiments conducted on a live network show that one of the introduced metric is able to characterize anomalies on small DNS servers, while the other behaves better on medium sized servers. Concerning DNS tunneling attacks, the proposed approach reveals to be an efficient tool for traffic profiling in the presence of DNS tunneling.

Keywords: tunneling, covert channel, intrusion detection, ids, characterization, dns protocol

1 Introduction

The detection of application-layer tunnels has recently received attention in the literature [18]. The case of *Domain Name Server (DNS)* tunneling has been disregarded in virtue of the inherent low throughput and the complexity of deploying an appropriate set-up in support of malicious usage of DNS data exchange. However, the recent grow of free-of-charge DNS domains overcomes the latter drawback and recent results show how appropriate settings of DNS tunneling tools can guarantee a throughput whose order of magnitude can achieve even some Mbps [6].

The principle of DNS tunneling is simple: a DNS tunneling tool embeds data in DNS queries and delivers DNS requests and responses between the tunneled client and a remote (rogue) DNS server, which forwards the received data to the real destination, hidden by DNS data (Fig. 1) [6]. Profiling has to be therefore performed by monitoring traffic of local (legitimate) DNS to understand if DNS requests of some clients are hiding tunneling. Since the packet sizes of (DNS) queries and answers are usually small, a single IP connection generating DNS tunneling without protection

FIG. 1: Entities involved in a DNS Tunnel

countermeasures could be easily identified in virtue either of the large sizes of its queries/answers or of the high number of queries in a small period of time. However, the problem here is to avoid individual sockets monitoring since it can be hardly performed in real time for a large set of clients. Our aim is to automatically identify a small portion of connections generating DNS tunneling by looking at the entire dataset of query and answers.

In this paper we introduce an innovative profiling algorithm to characterize DNS tunnels. The proposed method should not be considered a detection system, since our goal is to profile network traffic extrapolating characteristic features and identifying the pattern associated to malicious activities. The rest of the paper is structured as follows: Sect. 2 reports the proposed system, exhaustively describing the introduced profiling algorithm. Sect. 3 reports the network scenario we have considered during the tests, while obtained results are described in Sect. 4. Related work on the topic is discussed on Sect. 5. Finally, Sect. 6 reports conclusions and possible extensions to the work.

2 The Profiling Approach

In order to properly identify DNS tunnels, we propose a technique for traffic profiling based on several statistical fingerprints (features) of query and answers, acquired during the system evolution. The proposed approach is based on a features extraction phase, extrapolating selected features from captured network traffic. The profiling system is based on two metrics: the Principal Component Analysis (PCA), and Mutual Information (MI), respectively. We will now describe in detail the proposed characterization method.

2.1 Features Extraction

Let q and a be the packet sizes of a query and the corresponding answer, respectively (what answer is related to a specific query can be understood from the packet identifier) and δ the time-interval elapsing between them. Let m_a, m_q, m_δ and σ_a^2, σ_q^2 and σ_δ^2 the *averages* and *variances* of those quantities, measured over a working period of the local server. Together with averages and variances, we also exploit high order statistics, such as *skewness* and *kurtosis* of a, q and δ, denoted by s_a, s_q, s_δ and k_a, k_q, k_δ, respectively. Since s and k give a quantitative indication of the asymmetry (skewness) and heaviness of tails (kurtosis) of a probability distribution, they help improve detection inference with noisy features. An empirical analysis on the advantage of using high order statistic for the problem under investigation can be found in

[7]. In compact form, the vector of the features is: $\boldsymbol{\chi} = \{m_{a,q,\delta}, \sigma_{a,q,\delta}, s_{a,q,\delta}, k_{a,q,\delta}\}$, having 12 components, 4 statistics ($\{m, \sigma, s, k\}$) for 3 measured quantities ($\{a, q, \delta\}$).

Let n_s be the number of samples (couples of queries and answers) used to compute the mentioned statistics; n_s quantity is set to 10^3 in order to obtain quick feature generations. Features built with $n_s = 10^3$ may result very noisy. For noise considerations associated to n_s refer to [8].

2.2 Principal Component Analysis

The Principal Component Analysis (PCA) is a statistical function already known in the intrusion detection field [15]. It maps a coordinate space into a new coordinate system with axes commonly known as principal components (PCs). These axis have the property to point in the direction of maximum variance of the original data. In particular, the first PC identifies the greatest data variance in a single direction, the second one is relative to the second greatest degree of variance, and so on. The retrieved PCs are ordered by the amount of data variance they identify. Typically, the first PCs contribute most of the variance in the original data set so that we can describe them with only these PCs, neglecting the others, with minimal loss of variance. Once the PCA is computed, given a set of data and its associated coordinate space, it is possible to perform a data transformation by projecting them onto the new axes.

Considering for instance a clean DNS traffic (no attack is injected), we have extracted the inter-times features m_δ and σ_δ. Fig. 2 reports the captured data distribution. The red dotted line reported in the figure represents the first/main axis of the

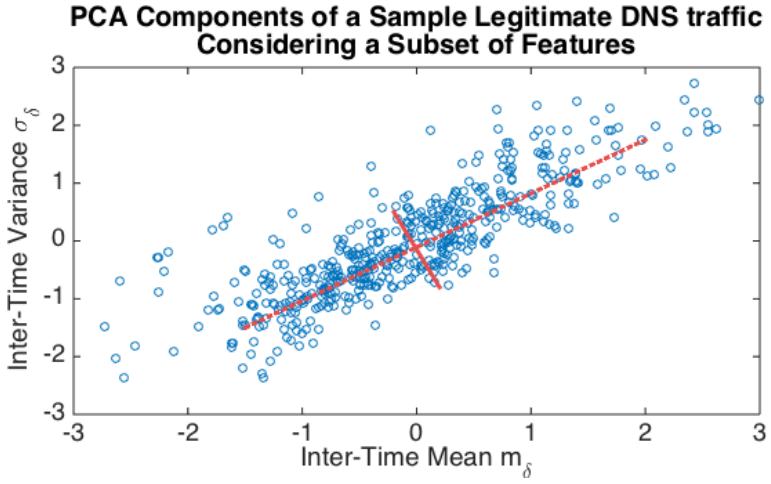

FIG. 2: Example of Variance Captured by the Component of the PCA for a Sample Traffic

PCA. It covers a wider range of data than the second axis, hence presenting a larger contribution to the information available from the data. Fig. 2 is just an example in which, for visualization purposes, the original bi-dimensional space of two features

may be synthesized by projection onto the first PCA axis; the data have a significant decrease of variance from the first PCA axis to the second PCA axis. In the rest of the paper, the PCA is computed onto the original χ vector, including the defined 12 features.

2.3 Mutual Information

Although preliminary tests reported a clear distinction between PCA results for legitimate and anomalous scenarios, this is not always true. For instance, Fig. 3 depicts the results of the PCA for the first two components, considering a legitimate and attack traffic (a DNS tunneling attack involving several download activities through the wget command line tool). The figure reports overlaps between the results related

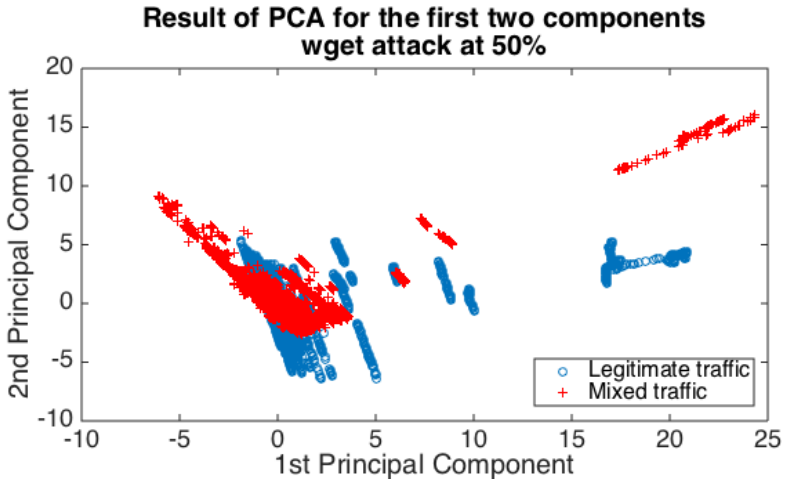

FIG. 3: PCA Results of Legitimate and Mixed Traffics, in terms of PCA_1 and PCA_2

to the two considered classes. In this case, the PCA does not help traffic characterization. In order to analyze the PCA variables to better identify anomalies, our system integrates the components of the PCA with an additional metric: the Mutual Information.

Each PCA component maps a single value we have extracted accordingly to Sect. 2.1, to a projection of that value into the PCA axis. In order to get information about a potentially anomalous traffic, our approach repeatedly applies the Mutual Information to a subset of the extracted set of samples of the features (records). In particular, we consider a fixed range of r records and we apply the metric by shifting the range over all the records. Therefore, let x be the index of the first record to consider, the PCA application we consider is relative to a subset $[x, x+r]$ of values[1], as reported in the following equation.

$$mi(x) = I(p_A^{[x,x+r]}, p_B^{[x,x+r]}) \tag{2.1}$$

[1] In case $x+r$ is greater than the size of the array including the extracted records, we restart from the beginning of the traffic.

Regarding p_A and p_B definition, a first approach we have followed has been focused on the simple mean/average metric, applied to the first component PCA_1 of the PCA.

$$m = \frac{1}{r} \sum_{i=x}^{x+r} PCA_1(i) \tag{2.2}$$

Nevertheless, this metric is not good for our context: as shown in Fig. 3, the overlapping between the clean and attack traffics leads to extremely similar values, hence making this metric ineffective.

In order to analyze the PCA variables to better identify anomalies, we have adopted a different metric. Specifically, the proposed system is based on the integration between the results obtained from the PCA analysis function and the *Mutual Information* (MI) [20], represented as $I(p_A, p_B)$. Mutual information is not new in intrusion detection [16, 10, 30, 31, 25] and may help the traffic profiling in the presence of anomalies. This is the first time it is applied in the DNS tunneling context.

Mutual Information allows us to understand the dependence between the analyzed variables. In our intrusion detection context, the first metric we consider, mi_{12}, is based on the computation of Mutual Information on the PCA results on the first two components PCA_1 and PCA_2 (in order of contribution).

$$mi_{12} = I(PCA_1^{[x,x+r]}, PCA_2^{[x,x+r]}) \tag{2.3}$$

In addition, we consider mi_{la}, a metric based on a different approach, as reported in the following equation.

$$mi_{la} = I(PCA_l^{[x,x+r]}, PCA_a^{[x,x+r]}) \tag{2.4}$$

where PCA_l identifies the PCA_1 component relative to the legitimate traffic and PCA_a is relative to the PCA_1 component computed on the potentially anomalous traffic.

Unlike mi_{12}, which only involves current/unknown traffic, the mi_{la} metric compares components belonging both to the unknown traffic and a traffic considered legitimate. Adopted legitimate traffic should be representative of a situation without attacks. For instance, this traffic may be built by analyzing multiple legitimate traffic [5]. The two different metrics thus approach the problem in different ways.

Let $mi(x)$ be the function that maps each value to the result of the Mutual Information applied on the considered PCA components (accordingly to Eq. 2.3 and Eq. 2.4), generated considering a subset of r elements consecutive to x.

Since retrieved values include noise, we apply a smoothing approach by using the *Exponential Window Moving Average (EWMA)* [33] (*smooth* in the following): the α parameter represents the weight on recent values on the smoothing operation.

$$mi'(x) = \alpha \cdot mi(x) + (1 - \alpha) \cdot mi'(x - 1) \tag{2.5}$$

In principle, we expect that an attack should induce more independence (hence lower values of MI) in the data due to its discontinuities with respect to clean traffic that should result more continuous in time. In other words, clean traffic should be characterized by higher level of similarity in time, thus leading to high values of MI. An example in the DoS context can be found in [25].

On the other hand, no theoretical reason exists to state that the correlation between the first two PCA components should achieve specific theoretical thresholds. The underlying idea proposed in this paper is rather to find a change in the correlation when anomalies begin to appear. In particular, we expect that the correlation corresponding to a clean situation exhibits an abrupt step when an anomaly is introduced.

3 Testbed

In this section of the paper we report the tests we have executed to validate the proposed protection algorithm.

3.1 Network

The following testbed has been used for performance evaluation. By taking Fig. 1 as a reference, we deployed a DNS server working in the rogue modality inside of the network of our institute. Two additional servers have been used as local servers (see Fig. 1), which receive requests from both legitimate clients and intruders. DNS traffic has been then recorded on the local servers. More details of the traffic traces are reported in Sect. 3.4. The first local server is a "small" DNS server, authoritative on a third level domain, composed by less than 10 servers accessed from the Internet. This DNS server serves about 50 clients. It generates an average of 7.8 different resolutions per second: 6% of these are related to authoritative domains, while the remaining 94% are related to clients requests.

A different situation is the "medium" server we consider. This server is authoritative on the second level domain of our research institution (more than 7000 employees), plus 50 third-level domains and about 1000 clients. In this situation, we have about 35 queries per second, and the composition of these queries is 50% of requests related to the authoritative domains and 50% for external names[2].

DNS samples were directly captured on the local servers. This is coherent with a situation in which all clients communicate with a single authorized server (this is a widely adopted security rule). In case multiple clients talk with different servers, the monitoring is performed on the firewall by listening on UDP port number 53.

A crucial choice for proper characterization consists of analyzing each DNS query, without discriminating between authoritative domains and external names or applying a finer analysis over different types of records (`A`, `AAAA`, `PTR`, `NS`, `MX`, `TXT`, etc.).

3.2 Applications

Three applications lie over the tunnel. The first one is a *wget* dump of an entire website. Even if the HTTP protocol is used, the typical on-off behavior of a real user-experience on the web does not take place here, since the executed wget operation is more similar to a data-transfer session (such as FTP). Tunneling is performed using proxy setting on the wget software, and the server side of the DNS forwards to a Squid proxy application [3].

The *ssh* protocol is used in the second case, by executing basic commands, such

[2] The mentioned statistics concerning the frequency of resolutions have been averaged over a single working day and did not change significantly in the monitored period of a week.

as directory browsing and other simple shell scripts queries. This test constitutes a typical interactive user session. Tunneling is performed using a localhost connection on the appropriate port, and the inherent connection is directly executed out of the tunnel by the rogue server.

The last experiment is conducted through a peer to peer (*p2p*) application. This case introduces high traffic burstiness. Tunneling has been set up with the usage of a socks proxy, once again using the ssh protocol, thus adding an unnecessary encryption layer. Together with p2p, ssh and wget, we also consider in the following tests a *mix* application, whose DNS trace contains samples from the three applications, extracted according to a uniform random distribution.

3.3 Tunneling tools

The *dns2tcp* tool [1] has been adopted for all the presented cases. This software implements TCP over DNS tunnels and it is composed by server-side and client-side programs. The server has got a list of resources: each resource is a local or remote service listening for TCP connections. The client listens instead on a predefined TCP port and relays each incoming connection to the final service through the server, making use of the DNS protocol. Information is encapsulated in the `TXT` field of DNS. A high-level architecture of the testbed is summarized in Fig. 1. Four components are involved: (i) tunneling client, (ii) local (honest) server, (iii) remote (rogue) tunneling server, and (iv) final destination (external) server. These components are usually located on different networks. The client executes the tunneling tool, by building queries associated to a "fake" domain. Queries are forwarded to the honest server, which forwards them to the rogue one, authoritative for the fake requested domain. The latter, in turns, performs de-encapsulation of the information in the DNS queries and rebuilds the real flow. Then, it contacts the external server and delivers the extracted information.

Other tools, such as *iodine* [2], have not been considered for our tests, since they may lead to simpler detection problems, as previously analyzed in [4, 8].

3.4 DNS traces

A DNS trace contains, in each row, the size of a given query and the size of the corresponding answer, together with the DNS response time (i.e., the time difference elapsing between them). Each row of the trace is characterized by the DNS identifier of the query. A DNS trace without tunnel is called *clean trace*. Two clean traces were considered during the tests, with respect to the small and medium servers of the network outlined above, called *small* DNS and *medium* DNS. A DNS trace with all rows corresponding to DNS messages encapsulating packets of a given tunneled application is called *tunnel trace*. A feature is generated by averaging the chosen quantity, e.g., the query size, over a number of rows, n_s , randomly chosen from the trace, by following a uniform distribution. A feature vector is associated to a tunnel case when a portion of the messages comes from a tunnel trace. Different percentages of mix are considered; for example, 90% out of the n_s are taken from a clean trace and the remaining 10% of samples are taken from a tunnel trace. In that case, the resulting dataset is called *Combined traffic (*10% *attack)* in the results.

4 Performance Evaluation

Accordingly to the description of the adopted scenario, reported in Sect. 3, we have executed tests on two DNS servers with different dimensions, executing different malicious activities: a *wget* file transfer, a *ssh* remote shell access, and the execution of a *peer-to-peer* software. In addition, accordingly to previous description in Sect. 3, we consider a *mix* attack involving all the mentioned malicious activities. We have analyzed the generated traffic and applied the algorithm described in Sect. 2, with $r = 500$ and $\alpha = 10^{-2}$.

At first, we have compared a legitimate traffic to a malicious one, injecting an attack inside of a legitimate traffic analyzed involving the communications with the two servers described above. Fig. 4 and Fig. 5 report obtained results relatively to the "small" DNS server (see Sec. 3), injecting peer-to-peer and ssh traffic at 50% and adopting the metric defined in Eq. 2.3.

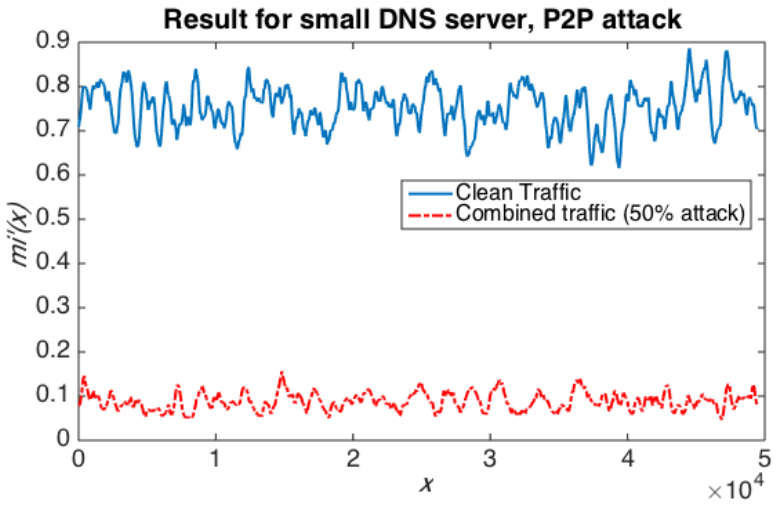

FIG. 4: Obtained $mi'(x)$ Results for Peer-to-Peer Traffic at 50% Analyzed on Small Server

We have found a clear distinction between legitimate and peer-to-peer traffic. In this case, a trivial separator/threshold may be adopted, assuming for instance anomalous each value $mi'(x) < 0.4$. Considering the mix attack (see Sect. 3.2), we have obtained results similar to peer-to-peer ones. Instead, if we consider the case of ssh, it is more difficult to define a good separator, since we noticed that the results are overlapped (see Fig. 5). Considering a wget based attack, we have obtained results similar to ssh, although with less evident superimposition.

By injecting an attack at 10% of the traffic, obtained results are similar, although the threshold grows to about 0.5. Instead, at 1%, overlapping makes us unable to properly characterize the network traffic. Such low injection of anomalies (*silent intruder*) is not representative of an attack: as stated in [4], in this case, the attack rate limit leads to a significant decrease of the exchanged payload data rate at the application level, with the consequent increase in the duration of the connection.

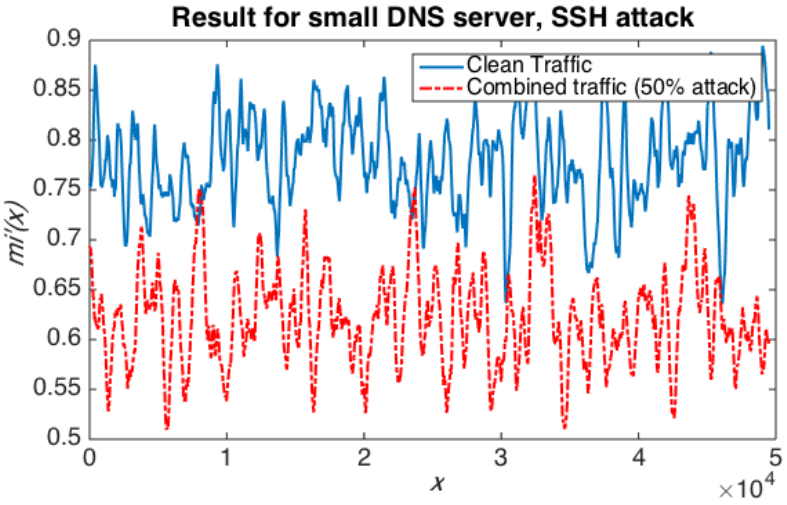

FIG. 5: Obtained $mi'(x)$ Results for SSH Traffic at 50% Analyzed on Small Server

By analyzing instead the results relative to the "medium" DNS server, for 50% of attack traffic, it is only possible to characterize the mix traffic, while single applications can't be properly characterized due to overlapping. In addition, we analyze for ssh attack that $mi'(x)$ results lead to higher values than clean traffic ones.

Until now, combined traffic has been built by injecting/mixing to the legitimate situation the traffic associated to malicious activities and for the entire duration of the traffic. Nevertheless, in a real environment, malicious traffic is injected on clean one, after the begin of the attack activities, for the entire duration of the attack. In order to represent this scenario, we now inject malicious operations on the network only for the second half of the considered time. Our aim is to identify *changes* on $mi'(x)$ results, expected to appear at half of the graph, corresponding to the begin time of the traffic injection activities. Let d_t be the difference between the means computed on the two half of the graph (left and right, l and r, respectively), for a considered traffic t, as reported in following equation.

$$d_t = |\mu_t^l - \mu_t^r| \tag{4.1}$$

Considering a clean traffic d_c and an anomalous one d_a, we define c a "sensitive change" when following equation is satisfied.

$$c \implies d_a > k \cdot d_c \tag{4.2}$$

with k fixed. In this case, we expect two different mean values between the left part of the graph (only relative to clean traffic) and the right one (including the attacks). Accordingly to Sect. 2.3, we have adopted two different metrics reported in Eq. 2.3 and Eq. 2.4. Fig. 6 report the results obtained relatively to the small DNS Server, injecting peer-to-peer traffic at 50%, at half of the capture. For simplicity, figures only plot results for combined traffics.

In both cases, we have analyzed that overlapping on legitimate traffic does not occur. Hence, a trivial *threshold* may be defined to distinguish between clean/legitimate

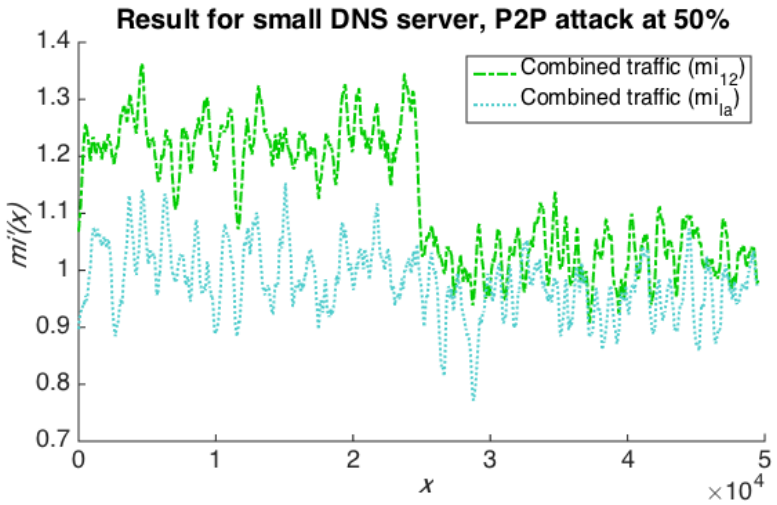

FIG. 6: Obtained $mi'(x)$ Results for Peer-to-Peer Traffic at 50% Injected at Half of the Capture and Analyzed on Small Server

and anomalous traffic, successfully identifying the anomaly. In addition, both the metrics lead to a change of $mi'(x)$ values after the attack has been injected into the traffic. Specifically, this change is visible through the mi_{12} metric, while the mi_{la} metric appears not to be good for this purpose. Nevertheless, referring to Eq. 4.2, for the mi_{12} metric we have found that $k < 62.9599$ always satisfies the equation. Similarly, $k < 16.87$ satisfies the equation for the mi_{la} metric. We can therefore state that the mi_{12} metric is preferable in this context, although the mi_{la} metric may also reveal an anomaly. This change identifies a variation on the traffic and could be adopted for detection purposes. Similar results has been retrieved by analyzing the other considered applications. By decreasing the percentage of attack injected to 10%, results are similar and a change is identified as well. Relatively to the threshold, instead, no overlapping is found only on the traffic related to the mix attack. Therefore, on the other cases, it's not possible to define a proper threshold.

Fig. 7 reports the results obtained by analyzing the traffic involving the medium DNS server. In this case, it should be evident that the mi_{la} metric is preferable to the mi_{12} one. Indeed, obtained k values are 11.8126 for the mi_{12} metric and 75.9076 for the mi_{la} metric.

Retrieved results show therefore how the proposed system is particularly effective for traffic characterization and how the two introduced metrics can be adopted to profile traffics of different nature. Hence, the proposed system could be applied to efficiently detect anomalies on a network.

5 Related Literature

Many tools for collecting DNS statistics drive detection of DNS attacks and disregard DNS tunneling. [14] analyzes DNS traffic of botnets (large groups of computers infected by trojan horses) to detect the originating traffic of the botnet manager by

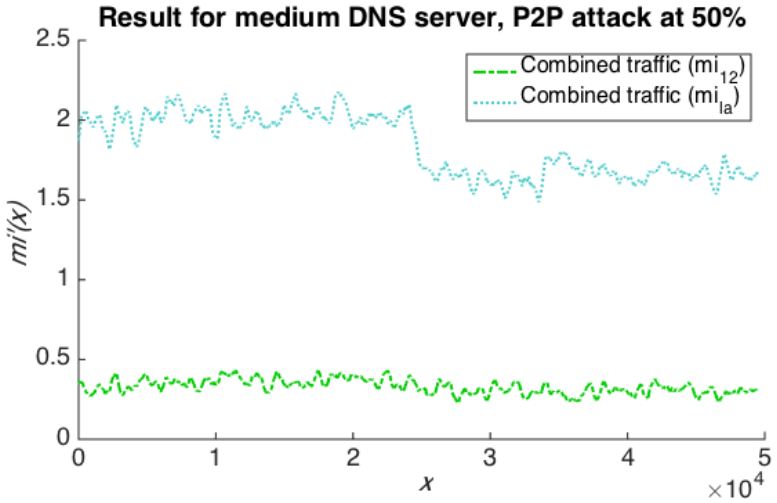

FIG. 7: Obtained $mi'(x)$ Results for Peer-to-Peer Traffic at 50% Injected at Half of the Capture and Analyzed on Medium Server

means of anomalies in the degree distribution of visited domains. [23] infers attacks to the DNS infrastructure, the *cross entropy* paradigm is used to detect significant changes in the distribution of conforming and non-conforming packet sizes. [9] uses sophisticated machine learning algorithms (e.g., decision trees under the random forest or boosting paradigms) to identify encrypted traffic tunnels with high accuracy without inspecting payload, IP addresses and port numbers. The detection of Fast-Flux Networks using DNS is proposed in [29], by discriminating the most reliable features over a large set of available features. Basic statistics of DNS queries are investigated in [26] through association with *dark* (i.e., unused) address spaces and development of *honeydns*, a tool that complements existing honeypots to prevent attackers from easily evading monitored networks.

There are few works instead relative to DNS tunneling detection and all the works are quite recent. In [19], the authors apply anomaly-based detection techniques making use of statistics of protocol messages. Anomaly-based detection is very adaptive as it simply looks at sudden changes of flows statistics [32, 11], while providing good detection rates. A tunnel is detected if some statistical threshold has been reached at a given time. Nevertheless, if silent intruders are in play, the settings of those threshold may be a hard task. The joint analysis of the anomalies at different timescales [15] or at different points of the system [17, 24] sometimes helps to improve performance. [27] proposes instead an anomaly detection system based on a different metric, the Independent Component Analysis, without focusing in particular on the DNS protocol.

Other approaches to DNS tunneling detection are based on packet inspection, namely, they exploit the content of the requested domain names. More specifically, the tool of [22] detects DNS tunneling by exploiting a neural network whose inputs include information about the used domain names. Other approaches are based instead on character frequency analysis of the domain names [12, 28] or quantitative analysis

of the traffic [13, 21]. The proposed system does not make use of packet inspection techniques: although packet inspection approaches may result more efficient, they analyze message content. Such behavior is computationally heavier than analyzing the temporal behavior of exchanged messages and it can not be adopted in case of encrypted messages (e.g. adoption of the DNSSec protocol).

Our previous works [4, 8, 7] show how basic classifiers can be exploited to build reliable and fast DNS tunneling detection, even if they showed a limit in the choose of a unique classifier for different servers and the separate detection of tunneled applications. In this work also, DNS tunneling detection is performed on the basis of the temporal behavior of DNS queries and answers, using a different approach.

6 Conclusions and Future Work

The paper has presented an innovative profiling method of DNS tunneling by exploiting Principal Component Analysis and Mutual Information. Our approach avoids packet inspection for computational reasons. The application of the anomaly-based equations of the paper may require a computation time of the order of milliseconds or less. Another reason relies on the generality of the approach that may applied to DNSSec that prevents an easy packet inspection. Moreover, the training on specific domain names may not be easily applicable to different servers and re-training should be accurately designed. The presented results show how the approach generates peaks in collected traces just after the injection of a tunnel inside legitimate traffic.

We found that in many cases, e.g., under the 50% of mixture between clean and anomalous traffic, the detection was always successful. This is true also below that threshold (e.g., under 30% of mix or less). The noise affecting the separation between clean and anomalous situations arises around 10% and below. We believe this study deserves further work since we found that detection depends on the equations parameters. Indeed, in mentioned situations, an accurate tuning of the parameters may be found to acquire anomalies with low positives and negatives detection rates. A specific effort should be devoted to find a good trade off between adaptation to traffic changes and detection reliability.

Therefore, the immediate next step of the research consists of applying the approach for detection purposes, considering for instance a larger dataset of conditions involving the variation of several parameters (e.g., number of rows of DNS trace to build a feature vector, proportion of anomalous traffic, frequency of the messages of legitimate traffic), thus also evaluating the detection time. An attack may be detected each time the trace of the MI on the first components of the PCA achieves a given threshold. A threshold could always be found in dependence of traffic and with enough anomaly packets in the server. A possible approach relies on a second level classifier that does not need training, such a "majority voting" as proposed in [8].

Other improvements are currently ongoing with respect to: profiling the tunneled applications and other clean traces, as well as further performance comparisons by using other signal processing metric, such as the *Kullback-Leibler divergence* [15].

Acknowledgements

Research activities that produced this document have been founded by Regione Liguria (Italy) in the context of the program "PAR FAS 2007-2013 - Progetto 4 - Programma triennale per la ricerca e l'innovazione: progetti integrati ad alta tecnologia" for the founding of the project "C-MES: Manufacturing Execution Systems (MES) basati su tecnologia Cloud".

References

[1] Hsc - tools - dns2tcp. `http://www.hsc.fr/ressources/outils/dns2tcp/`. Accessed: 2015-10-16.

[2] Ip tunnel over dns howto with iodine. `http://ip-dns.info`. Accessed: 2015-10-16.

[3] Squid: Optimising web delivery. `http://www.squid-cache.org`. Accessed: 2015-10-16.

[4] M Aiello, M Mongelli, and G Papaleo. Dns tunneling detection through statistical fingerprints of protocol messages and machine learning. *International Journal of Communication Systems*, 2014.

[5] Maurizio Aiello, Enrico Cambiaso, Silvia Scaglione, and Gianluca Papaleo. A similarity based approach for application dos attacks detection. In *Computers and Communications (ISCC), 2013 IEEE Symposium on*, pages 430–435. IEEE, 2013.

[6] Maurizio Aiello, Alessio Merlo, and Gianluca Papaleo. Performance assessment and analysis of dns tunneling tools. *Logic Journal of IGPL*, 21(4):592–602, 2013.

[7] Maurizio Aiello, Maurizio Mongelli, and Gianluca Papaleo. Basic classifiers for dns tunneling detection. In *Computers and Communications (ISCC), 2013 IEEE Symposium on*, pages 000880–000885. IEEE, 2013.

[8] Maurizio Aiello, Maurizio Mongelli, and Gianluca Papaleo. Supervised learning approaches with majority voting for dns tunneling detection. In *International Joint Conference SOCO'14-CISIS'14-ICEUTE'14*, pages 463–472. Springer, 2014.

[9] Riyad Alshammari and A Nur Zincir-Heywood. Can encrypted traffic be identified without port numbers, ip addresses and payload inspection? *Computer networks*, 55(6):1326–1350, 2011.

[10] Fatemeh Amiri, MohammadMahdi Rezaei Yousefi, Caro Lucas, Azadeh Shakery, and Nasser Yazdani. Mutual information-based feature selection for intrusion detection systems. *Journal of Network and Computer Applications*, 34(4):1184–1199, 2011.

[11] Tammam Benmusa, David J Parish, and Mark Sandford. Detecting and classifying delay data exceptions on communication networks using rule based algorithms. *International Journal of Communication Systems*, 18(2):159–177, 2005.

[12] Kenton Born and David Gustafson. Detecting dns tunnels using character frequency analysis. *arXiv preprint arXiv:1004.4358*, 2010.

[13] Kenton Born and David Gustafson. Ngviz: detecting dns tunnels through n-gram visualization and quantitative analysis. In *Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research*, page 47. ACM, 2010.

[14] Pieter Burghouwt, Marcel Spruit, and Henk Sips. Detection of botnet collusion by degree distribution of domains. In *Internet Technology and Secured Transactions (ICITST), 2010 International Conference for*, pages 1–8. IEEE, 2010.

[15] Christian Callegari, Loris Gazzarrini, Stefano Giordano, Michele Pagano, and Teresa Pepe. Improving pca-based anomaly detection by using multiple time scale analysis and kullback–leibler divergence. *International Journal of Communication Systems*, 27(10):1731–1751, 2014.

[16] Aki PF Chan, Wing WY Ng, Daniel S Yeung, and Eric CC Tsang. Multiple classifier system with feature grouping for intrusion detection: mutual information approach. In *Knowledge-Based Intelligent Information and Engineering Systems*, pages 141–148. Springer, 2005.

[17] Yacine Djemaiel, Slim Rekhis, and Noureddine Boudriga. Intrusion detection and tolerance: A global scheme. *International Journal of Communication Systems*, 21(2):211–230, 2008.

[18] Maurizio Dusi, Manuel Crotti, Francesco Gringoli, and Luca Salgarelli. Tunnel hunter: Detecting application-layer tunnels with statistical fingerprinting. *Computer Networks*, 53(1):81–97, 2009.

[19] Wendy Ellens, Piotr uraniewski, Anna Sperotto, Harm Schotanus, Michel Mandjes, and Erik Meeuwissen. Flow-based detection of dns tunnels. In Guillaume Doyen, Martin Waldburger, Pavel eleda, Anna Sperotto, and Burkhard Stiller, editors, *Emerging Management Mechanisms for the Future Internet*, volume 7943 of *Lecture Notes in Computer Science*, pages 124–135. Springer Berlin Heidelberg, 2013.

[20] S. Guiau. *Information Theory with New Applications*. Advanced Book Program - McGraw-Hill Book Company. MacGraw-Hill Books Company, 1977.

[21] Michael Himbeault. *A novel approach to detecting covert DNS tunnels using throughput estimation*. PhD thesis, University of Manitoba, 2014.

[22] Jarod Hind. Catching dns tunnels with a.i. *Proceedings of DefCon*, 17, 2009.

[23] Anestis Karasaridis, Kathleen Meier-Hellstern, and David Hoeflin. Nis04-2: Detection of dns anomalies using flow data analysis. In *Global Telecommunications Conference, 2006. GLOBECOM'06. IEEE*, pages 1–6. IEEE, 2006.

[24] Sándor Molnár and Marcell Perényi. On the identification and analysis of skype traffic. *International Journal of Communication Systems*, 24(1):94–117, 2011.

[25] M Mongelli, M Aiello, E Cambiaso, and G Papaleo. Detection of dos attacks through fourier transform and mutual information. In *Communications (ICC), 2015 IEEE International Conference on*, pages 7204–7209. IEEE, 2015.

[26] Jon Oberheide, Manish Karir, and Z Morley Mao. Characterizing dark dns behavior. In *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 140–156. Springer, 2007.

[27] Francesco Palmieri, Ugo Fiore, and Aniello Castiglione. A distributed approach to network anomaly detection based on independent component analysis. *Concurrency and Computation: Practice and Experience*, 26(5):1113–1129, 2014.

[28] Cheng Qi, Xiaojun Chen, Cui Xu, Jinqiao Shi, and Peipeng Liu. A bigram based real time dns tunnel detection approach. *Procedia Computer Science*, 17:852–860, 2013.

[29] Revelli and N. Leidecker. Detection of fast-flux networks using various dns feature sets. In *Shakacon 2009*, June 2009.

[30] Jingping Song, Zhiliang Zhu, and Chris Price. Feature grouping for intrusion detection system based on hierarchical clustering. In *Availability, Reliability, and Security in Information Systems*, pages 270–280. Springer, 2014.

[31] Jingping Song, Zhiliang Zhu, Peter Scully, and Chris Price. Modified mutual information-based feature selection for intrusion detection systems in decision tree learning. *Journal of computers*, 9(7):1542–1546, 2014.

[32] Anna Sperotto, Gregor Schaffrath, Ramin Sadre, Cristian Morariu, Aiko Pras, and Burkhard Stiller. An overview of ip flow-based intrusion detection. *Communications Surveys & Tutorials, IEEE*, 12(3):343–356, 2010.

[33] Dan Tang, Kai Chen, XiaoSu Chen, HuiYu Liu, and Xinhua Li. A new detection method based on aewma algorithm for ldos attacks. *Journal of Networks*, 9(11), 2014.