# Flow-Based Detection of DNS Tunnels

Wendy Ellens[1], Piotr Żuraniewski[1,4,⋆], Anna Sperotto[2], Harm Schotanus[1], Michel Mandjes[3], and Erik Meeuwissen[1]

[1] TNO, The Netherlands
[2] University of Twente, The Netherlands
[3] University of Amsterdam, The Netherlands
[4] AGH University, Poland

**Abstract.** DNS tunnels allow circumventing access and security policies in firewalled networks. Such a security breach can be misused for activities like free web browsing, but also for command & control traffic or cyber espionage, thus motivating the search for effective automated DNS tunnel detection techniques. In this paper we develop such a technique, based on the monitoring and analysis of network flows. Our methodology combines flow information with statistical methods for anomaly detection. The contribution of our paper is twofold. Firstly, based on flow-derived variables that we identified as indicative of DNS tunnelling activities, we identify and evaluate a set of non-parametrical statistical tests that are particularly useful in this context. Secondly, the efficacy of the resulting tests is demonstrated by extensive validation experiments in an operational environment, covering many different usage scenarios.

**Keywords:** network flows, DNS tunneling, anomaly detection, cyber security.

## 1 Introduction

Tunneling data over DNS may be used as a way to circumvent access and security policies in firewalled networks. A typical example is to illegally browse the web when an access fee is requested, as it may happen in hotels or airports. DNS tunneling is possible because DNS requests are almost never filtered at the firewall, effectively opening a security breach. The fact that information bypasses a network first line security mechanism makes DNS tunneling very attractive also in contexts other than free web browsing. Key examples are *command and control* and *data exfiltration* in cyber-espionage attacks, where it is fundamental for an attacker to have an available but inconspicuous communication channel.

DNS tunneling works by encapsulating data into DNS packets. Typically, the tunnel client encapsulates the data to be sent in a query for a specific domain name. The DNS resolver treats the tunnel traffic as a *regular* request by starting the lookup process for the requested domain name, possibly recursively consulting other DNS resolvers, as in Figure 1. At the end of this operation, the request

⋆ Part of this work was done while the author was also at the University of Amsterdam.

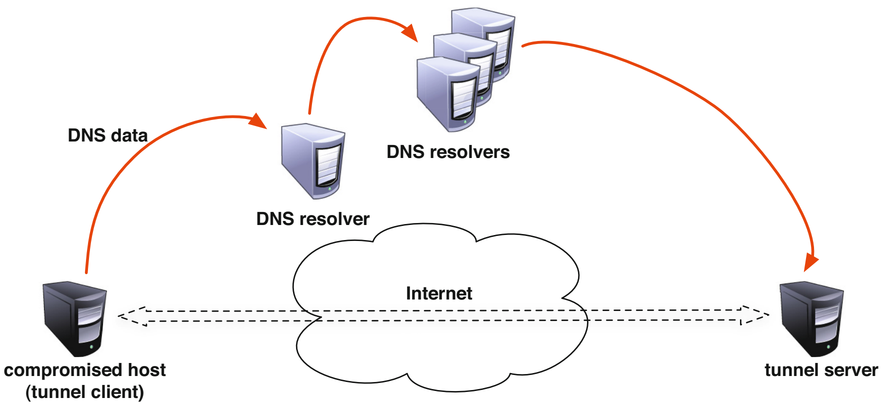


**Fig. 1.** General setup of DNS tunneling

is processed by the tunnel server. The server retrieves the encapsulated data and responds to DNS queries by enclosing tunnel data in the answer section of the DNS response message.

This paper describes a novel approach to the automated detection of DNS tunnels at the network boundaries, specifically targeting web browsing, data exfiltration and command & control traffic. The novelty of our approach is in combining network flows [1] with statistical tests, using data collected in a production network. Our procedure has been tested in operational settings with injected DNS tunnel traffic and during normal network operations. A flow is defined as "a set of IP packets passing an observation point in the network during a certain time interval and having a set of common properties" [2]. Since they are by now vastly deployed and less resource intensive than deep packet inspection techniques, network flows have stirred the interest of the research community in applying them for network security [3]. Our procedure has been tested in operational settings with actual DNS tunnel traffic and it shows promising results in detecting anomalous DNS traffic. However, the approach described in this paper is quite general and can be used to detect other types of network anomalies, caused by e.g. intrusions [4], faults, or changing user behavior [5].

Tunneled traffic and related detection mechanisms have received attention in the research community over the last years. The works in [6] and [7] focus on the analysis of the performance of data transmission over tunnels. In [8,9], the authors propose a statistical classification mechanism, based on basic features as packet size and packet inter-arrival time, for detecting HTTP and SSH tunnel traffic in IP flows. The authors of [10] address the topic of passive DNS security monitoring, in which a data-mining approach to the detection of DNS tunnels and fast-flux is presented. Differently from these contributions, we provide a set of possible statistical detection mechanisms specifically targeting DNS tunnels at the flow level. The work in [11] proposes a flow-based detection of DNS server cache poisoning and tunneling attacks over DNS, the latter focusing on deviations of the size of DNS packets. In our work we monitor a larger set of variables, aiming to detect abuse of the DNS protocol (tunneling) which can include various types of malicious activities, such as taking control over an attacked host or data exfiltration. Besides, the fact that we have ground truth information facilitates obtaining deep insight into the detector's performance. In this way we evaluated the usefulness of several detectors, each tracking several variables, using both raw and time-binned data.

This paper is organized as follows. We start with a description of the experiments we carried out to obtain flow data for DNS tunnels in Section 2. Section 3 presents a detailed analysis of how DNS tunnel traffic can be characterized based on flow data. This analysis forms the basis for defining the set of flow-based DNS tunnel detectors that we present in Section 4. In Section 5 we present our results in detecting DNS tunnel traffic. Finally, we draw our conclusions in Section 6.

## 2 Experimental Setup

This section presents the setup used in our experiments. We introduce the architecture and the methodology used for the data collection, and we present the collected data sets that we use for the analysis of the tunnel characteristics and for the validation of our detection approach.

We have set up a typical DNS tunnel architecture. A subnetwork of a university campus (i.e., a part of the production network used on a regular basis by approximately 300 people) mimics a compromised network. This setup allows conducting experiments in the presence of representative background traffic. A host in this subnetwork acts as compromised machine, i.e., the *tunnel client*. This is the host that a hypothetical attacker uses to exfiltrate data out of the compromised network. Outside the campus network, a second host plays the role of the second tunnel end point, namely the *tunnel server*. The tunnel server is the receiver of the exfiltrated data. To tunnel traffic over DNS we run the tool *Iodine*[1] on both the tunnel client and the tunnel server. *Iodine* allows users to specify to which DNS resolver the tunnel DNS requests should be sent. We considered the options described below.

We tested our scenarios by using three ordinary DNS resolvers, which we indicate as **non-local resolvers**, namely the DNS in the network of the tunnel client; a Norton open DNS resolver; and a resolver on the tunnel server machine. In all these cases, a stream of packets with the same source port will be sent from the client to the designated DNS resolver. In addition, since the way *Iodine* handles the stream of packets to the resolvers can be considered in itself an indication of irregular DNS activity, we also tested an alternative setup. In this case the tunnel client uses the host on which it is running as a **local resolver**. The host forwards the DNS request to an external DNS resolver using the Linux tool `bind`, which instantiates a new source port for each issued DNS request, as would happen in the case of regular DNS traffic.

[1] `http://code.kryo.se/iodine`

Our experiments aim to generate realistic examples of DNS tunnel usage regarding data exfiltration, command & control and web-browsing. We have identified three attack scenarios. Each of the corresponding tests is repeated for all four resolvers.

– Data exfiltration is represented by means of a **file transfer**. The file transfer operation has been implemented using `nc`. We performed transfers for files of 1 and 2MB.
– To generate command and control traffic, we set up an **interactive session** over the tunnel. We mimic this interactive shell session behavior by issuing a series of random bash commands over an ssh connection at randomly chosen time intervals. We conducted interactive sessions with a duration of 5 and 10 minutes.
– In addition, we consider a third scenario, in which the tunnel is used for **web browsing**. Web browsing is not directly related to data exfiltration or command and control, but it certainly represents the most widely-known use of DNS tunneling. In this case, the tunnel is not necessarily a mean for hiding information but a way to circumvent web access restrictions. We generate web browsing sessions by requesting random pages from a pool of urls at random time intervals. The web browsing sessions have a duration of 10 and 15 minutes.

The identified scenarios allow us to study several aspects of DNS tunnel usage. For example, a file transfer is likely to be, relatively speaking, bandwidth-aggressive, even though, considering both the characteristics of the DNS protocol and the encapsulation overhead for transferring data over the tunnel, a DNS tunnel typically shows limited throughput. Differently, an interactive session is largely human-controlled, therefore, it is not likely to create regular usage patterns and it can be carried out with minimal data transfer. The web browsing scenario complements the previous scenarios because it includes both the human component as well as the data transfer component (download of the page content).

In addition to the above-described tests, we collected a data set of normal DNS traffic. For each experiment, network flows have been created using the IPFIX software probe *YAF*[2], by monitoring the traffic of the subnetwork where the tunnel client resides. This allowed us to have a realistic mixture of tunnel and normal traffic. For the flow creation, we considered a flow active timeout of 60 seconds and an inactive timeout of 15 seconds. Our experiments led to the collection of 4 data sets, summarized in Table 1. For each of the three data sets containing tunnel traffic (file transfer, interactive session and web browsing) two consecutive attacks have been carried out using four different DNS resolvers in the following order: Norton DNS resolver, local resolver, a resolver in the network of the tunnel client, and the resolver at the tunnel server. While the percentage of tunnel flows ranges from 10% to 30%, it does not imply that the detection is trivial. In fact, more than 99% of tunnel flows are due to the local resolver and these flows essentially do not differ from legitimate DNS flows.

[2] YAF: Yet Another Flowmeter, `http://tools.netsa.cert.org/yaf/`

**Table 1.** Data sets collected during the experiments

| **Data Set** | **Flows** | **Tunnel flows (total)** | **Local resolver flows** | **Duration (hrs)** |
|---|---|---|---|---|
| *File transfer* | 379 355 | 113 635 | 113 527 | 1.27 |
| *Interactive session* | 353 075 | 36 389 | 36 265 | 1.55 |
| *Web browsing* | 339 108 | 56 799 | 56 641 | 1.3 |
| *Normal traffic* | 3 739 138 | 0 | 0 | 32.36 |

## 3 Data Analysis

In this section, we first identify which variables, based on flow data, may provide meaningful information for the detection of DNS tunnels. Secondly, we show, by means of the collected data sets, how such variables are affected by tunnel traffic.

### 3.1 Data Selection

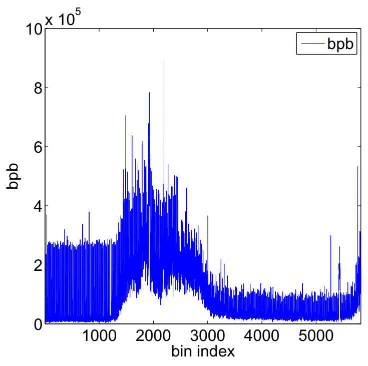
Our analysis is based both on *raw data*, i.e., the flows without further processing, as well as on preprocessed data in form of *time series*. In particular, we consider the time series of the number of flows, packets and bytes per bin. The time series are created by taking the duration of a flow into account, therefore by proportionally dividing the number of packets and bytes in a flow over the bins in which the flow was active. The timeseries are labeled: a bin is marked as "tunnel active" if it contains at least one tunnel flow. We create time bins of lengths of 1 second, 5 seconds and 20 seconds.

Given the information in the raw flow data and the binned data we have selected for the further evaluation eight **variables** that may indicate the presence of tunnel traffic. For the raw flows we monitor the *bytes per flow* (bpf), the *packets per flow* (ppf), the *bytes per packet per flow* (bpppf, the average bytes in a packet calculated per flow) and the *flow duration*. For the binned data the variables that we monitor are *bytes per bin* (bpb), *packets per bin* (ppb), *flows per bin* (fpb) and *bytes per packet per bin* (bpppb, the average bytes in a packet calculated per bin).

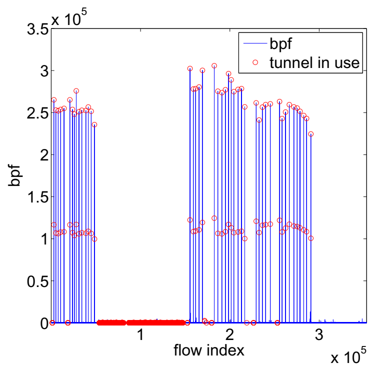
### 3.2 Analysis of DNS Flows

We analyze the eight variables of Section 3.1 for the four data sets of Table 1. Figure 2 shows several several examples of this analysis. In each of the figures, the flows created by the DNS tunnel or the bins containing tunnel traffic are marked (if any tunnel traffic is present). Our analysis showed that bytes per flow gives results very similar to packets per flow. The same holds for bytes and packets per bin. In the rest of this paper we therefore do not consider the number of packets per flow/bin.

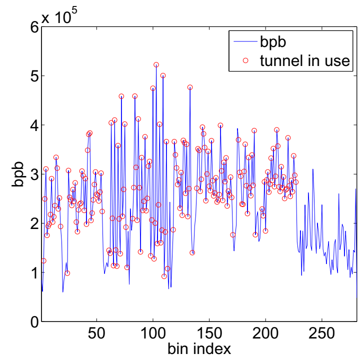
A first observation is that the normal DNS traffic shows a clear day-night pattern when considering the number of bytes (see Figure 2(a)), packets and flows per bin. For the bytes per packet per bin, bytes per flow, packets per flow



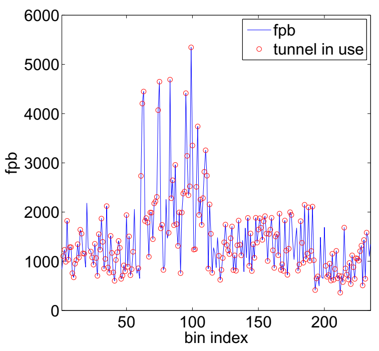
(a) Bytes per 20 sec bin (normal traffic)



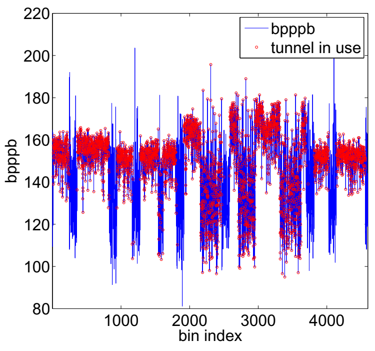
(b) Bytes per flow (interactive session)



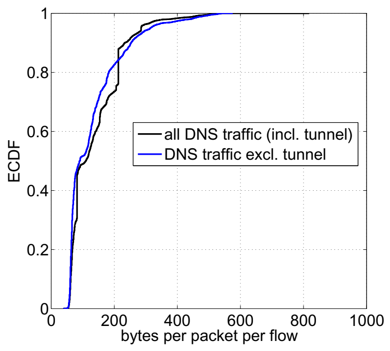
(c) Bytes per 20s bin (interactive session)



(d) Flows per 20 sec bin (web browsing)



(e) Bytes per packet per 1s bin (file)



(f) Bpppf ECDF for local resolver

**Fig. 2.** Plots of several monitored variables and several session types. DNS tunnel activity is marked (if present).

and bytes per packet per flow such a day-night pattern is not clearly visible. As a consequence, if a the threshold on the number of bytes/packets/flows per bin is used, this should be time-of-the-day-dependent.

When tunnel traffic is injected, its characteristics depend on the type of resolver is in use. In Figure 2(b) we see particularly well a distinction between the traffic generated for the local resolver — which uses a new port number for each request, as does normal DNS traffic — and for the non-local resolvers — which reuse the same port number, causing different requests to be aggregated to a one large flow. A local resolver activity can also be easily spotted on Figures 2(c) and 2(d) – look for the periods of both higher values and variability.

As expected, an increase in the bytes per flow/bin is visible when the DNS tunnel is active (see Figures 2(b) and 2(c)). The increase is much more pronounced in bytes per flow than bytes per bin. However, it can be seen in the figures that bytes per flow/bin are not effective to detect the tunnel in the case of a local resolver, i.e, for flows with index interval $0.5 \cdot 10^5 - 1.5 \cdot 10^5$ and for bins in the interval 60 – 120. To detect DNS tunnels in the case of a local resolver, flows per bin might be monitored (see Figure 2(d)).

A variable that shows an increase due to tunnel activity for all resolvers is the average number of bytes per packet per bin. A difference between bins with and without tunnel activity is especially well visible when looking at Figure 2(e). Finally, the tunnel traffic does not affect the number of bytes per packet per flow time series. However, when the tunnel is active, we observed a variation in the distribution of the packet size, in particular an increase in the number of packets of approximately 200 bytes. The empirical cumulative distribution function (ECDF) of the normal traffic for 12 non-overlapping intervals of normal traffic shows that the distribution is stationary over time, so the distribution-based method may also be suitable for this variable. Figure 2(f) shows a clear distinction between the ECDFs of the normal traffic and the traffic including the tunnel flows. In this figure the tunnel uses the local resolver. These ECDFs do not show any significative difference when non-local resolvers are considered.

## 4 Flow-Based Anomaly Detectors

The analysis in Section 3.2 has shown that tunnel traffic affects normal DNS traffic in several ways and that different variables can be used for detection. In this section, we investigate and define a set of flow-based detection methods based on the previously identified variables.

### 4.1 Anomaly Detection Methods

There are several methods to find *anomalies* (irregularities, in our case a sudden increase, also called a changepoint) in the values of the aforementioned variables [12]. Our approach to detect anomalies in DNS traffic consists of three methods. Each method addresses a different aspect of tunnel traffic, i.e. we aim at detecting anomalies causing ii) peaks in the traffic time series; ii) changes in the average amount of traffic and iii) changes in the underlying traffic distribution. We explain and discuss these methods here, indicating also how we tailored them to our needs:

- **Threshold method**: An alarm is raised if a certain threshold is exceeded. The threshold is chosen in such a way that the false positives in the normal data are limited, for example by setting a threshold equals to the $p^{th}$ percentile of the normal traffic distribution for a considered variable. For data that shows a day-night pattern we consider a *time-of-the-day-dependent threshold.* This means that the threshold changes at regular intervals based on normal data captured at the same moment of the day.
- **Brodsky-Darkhovsky method** (BD): when the tunnel activity causes an increase in the average value of one of the variables, but it does not create clearly identifiable peaks, one may use the BD-method [13]. This method calculates the average over a number of observations and compares this to the average of an earlier period. The BD-method searches for evidence of an anomaly by identifying periods of observations with changes in mean. In our study we have focused on upward shifts in mean. The advantage of the BD-method is that it considers periods of different lengths. The method works as follows. A window $(x_i, \ldots, x_{N+i})$ of $N$ past observations is taken, this window is split in two and if the increase in average between the first part $(x_i, \ldots, x_{i+k-1})$ and the second part $(x_{i+k}, \ldots, x_{i+N})$ exceeds a threshold, an alarm is raised. This procedure is repeated for all possible $k$ such that both parts contain at least $n$ observations. When an alarm is raised, the method gives the offset $k$ in the considered window at which the increase in average took place.
- **Distribution-based method**: if a variable does not show peaks or an increased average when the tunnel is active, a change may be visible in the distribution of its values. For those cases we compare the ECDF of the observed values in some window with the ECDF of the values in normal traffic. We then perform the *Kolmogorov-Smirnov test* (KS) [14], which means that we calculate the maximum distance between the two ECDFs and raise an alarm if this difference exceeds a certain threshold. This threshold can be based on theoretical results or a training set.

We remark that the BD-method and the distribution-based method are window-based and introduce some delay in the detection.

### 4.2 Five Flow-Based Detectors for DNS Tunnels

The analysis of DNS tunnel flows in Section 3.2 provided us with ideas about how to combine the monitoring variables (based on raw or binned flows) with anomaly detection methods to a DNS tunneling detector for either local resolvers, non-local resolvers, or both. By combining the detectors for local and non-local resolvers, we have implemented five flow-based detectors for DNS tunnels:

- **Detector 1** uses raw flows as an input. To detect non-local resolver tunnels it puts a threshold on the bytes per flow. To reduce the number of false positives a threshold on the flow duration is added. An alarm is raised if the number of bytes per flow exceeds 5000 and the flow duration is larger than 55 seconds. This detector component is combined with a component that detects local resolver tunnels. This component monitors changes in the KS-test value created by the presence of local resolver tunnels flow, and it raises an alarm if the maximum difference between the observed ECDF and the baseline ECDF for the bytes per packet per flow is more than 0.15.[3]

- **Detector 2 and 3** both combine binned flows with a time-dependent threshold. Detector 2 uses bytes per bin for non-local resolvers and flows per bin for local resolvers. Detector 3 monitors the average number of bytes per packet per bin. Both detectors use thresholds that generate a maximum of 1% of false positives in the normal data of the same hour. Bin sizes of 1, 5 and 20 seconds have been tested.
- **Detector 4 and 5** use binned flows and the BD-method to detect an increase in the average of the monitored variable.[4] Again one detector uses bytes per bin for non-local resolvers and flows per bin for local resolvers and the other detector uses bytes per packet per bin.[5]

## 5 Detection Performance

To evaluate the detection performance of the proposed detectors, we consider the two metrics. The first detection metric is the *detection rate*, i.e. the fraction of attacks that is detected. In our case, we consider as an attack each of the file transfers, interactive sessions or web browsing sessions performed during the experiments. An attack is considered as detected if at last one alarm is raised in the time frame of the attack. The second detection metric that we consider is the *number of false positives per session*. For practical purposes, since verifying a false positive can potentially be a costly operation, it is important that the number of false positives per hour/day is limited.

The calculation of the detection rate and the false positives is straightforward for the threshold method. The BD-method and the distribution-based method are both based on a window of flows/bins, how to mark the single flow/bin as normal or anomalous (i.e. we report them as tunnel activity) is therefore less obvious. The BD-method gives us the moment at which the average has increased and the moment at which we notice this increase. We mark all flows/bins between

[3] This threshold is determined based on training and is higher than the standard critical value of 0.016 for a significance level of 5%. Every 2500 flows this Kolmogorov-Smirnov test is performed on an ECDF based on 10000 flows. A moving average of size 2 is applied before calculating the ECDFs, because of the correlation in the data due to request-response patterns.

[4] For bin sizes of 1, 5 and 20 second respectively, the window sizes $N$ are 180, 50, 30 bins, with a minimum number of observations $n$ equal to 18, 5 and 3.

[5] The thresholds for the increase in average are 10000, 35000, 80000 for bytes per bin, 100, 300, 700 for flows per bin, 20, 15, 10 for bytes per packet per bin, for bin sizes of 1, 5, 20 seconds, respectively.

**Table 2.** Detection performance for the proposed detectors

| **Detector nr. - resolver** | **Variable** | **Detection rate** | **False positives** |
|---|---|---|---|
| *File transfer* | | | |
| *1 - non-local* | bpf | 1 | 2 |
| *1 - local* | bpppf | 1 | 3 |
| *Interactive session* | | | |
| *1 - non-local* | bpf | 1 | 0 |
| *1 - local* | bpppf | 1 | 4 |
| *Web browsing* | | | |
| *1 - non-local* | bpf | 1 | 2 |
| *1 - local* | bpppf | 1 | 1 |

| | | **Bin size (s)** | | | **Bin size (s)** | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 5 | 20 | 1 | 5 | 20 |
| *File transfer* | | | | | | | |
| *2 - non-local* | bpb | 1 | 0.667 | 1 | 4 | 0 | 0 |
| *2 - local* | fpb | 1 | 0 | 1 | 5 | 0 | 0 |
| *3 - all* | bpppb | 0.375 | 0.5 | 1 | 13 | 4 | 0 |
| *Interactive session* | | | | | | | |
| *2 - non-local* | bpb | 0.167 | 0.167 | 0 | 1 | 0 | 0 |
| *2 - local* | fpb | 0 | 0.5 | 0 | 0 | 0 | 0 |
| *3 - all* | bpppb | 0.5 | 0.75 | 1 | 32 | 6 | 0 |
| *Web browsing* | | | | | | | |
| *2 - non-local* | bpb | 0.5 | 0.333 | 0.5 | 1 | 0 | 0 |
| *2 - local* | fpb | 0.5 | 0.5 | 0.5 | 1 | 0 | 0 |
| *3 - all* | bpppb | 0.5 | 0.75 | 1 | 10 | 1 | 0 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| *File transfer* | | | | | | | |
| *4 - non-local* | bpb | 1 | 0.667 | 0.833 | 129 | 93 | 28 |
| *4 - local* | fpb | 1 | 0 | 1 | 9 | 31 | 11 |
| *5 - all* | bpppb | 0.75 | 0.5 | 0.875 | 152 | 0 | 24 |
| *Interactive session* | | | | | | | |
| *4 - non-local* | bpb | 0.333 | 0.833 | 1 | 88 | 52 | 22 |
| *4 - local* | fpb | 0.5 | 1 | 1 | 24 | 21 | 8 |
| *5 - all* | bpppb | 0.5 | 0.625 | 0.5 | 51 | 17 | 11 |
| *Web browsing* | | | | | | | |
| *4 - non-local* | bpb | 1 | 1 | 1 | 270 | 65 | 38 |
| *4 - local* | fpb | 1 | 1 | 1 | 1 | 26 | 11 |
| *5 - all* | bpppb | 0.75 | 0.625 | 0.625 | 116 | 39 | 17 |

these two moments as anomalous. For the distribution-based method we mark as anomalous the moment at which the alarm is raised.

We have applied the five detectors of Section 4.2 to the three attack data sets of Table 1. The detection results have been summarized in Table 2.

*Detector 1* detects all DNS tunnel usages at the expense of 12 false positives in total (with 8 of them being a result of the aforementioned detection delay effect). For binned data, *detector 3* outperforms *detector 2* if the bin size is set to 20s. *Detector 4* generally has a better detection rate than *detector 2*. *Detector 5* has a somewhat worse detection rate than *detector 3* for 5 and 20 second bins. For 1 second bins *detector 5* detects more attacks than *detector 3*, but the results are not as good as those of *detector 3* with 20 seconds bins. The false positives, although decreasing with the bin size, are high for *detectors 4* and *5*. They are an inherent property of the detection method (detection delay).

Our analysis identified several detection mechanisms that can detect DNS tunnel traffic, each one of them based on different input data (raw flows and time series) and detection approaches. The choice of which one of the proposed detection mechanism, or a combination of them, should be applied may depend on specific network characteristics and requirements. For example, *detector 1* would be a good choice for a solution based on raw flow data, while *detector 3* can be directly applied to time series. It is therefore important that the algorithms are tuned for the application to specific environments.

## 6 Conclusions

In this paper, we have proposed a technique for the detection of DNS tunnel traffic. In particular, we have targeted several tunnel usage scenarios, ranging from the traditional web browsing to cyber-espionage attacks, like command & control channels and data exfiltration. Our approach is based on the combination of flow data and statistical tests for anomaly detection able to capture the variability of DNS tunnel traffic. Our analysis of the characteristics at the flow level of DNS tunnel traffic has highlighted that flow data offer suitable indicators for the presence of tunnel traffic. Appropriate metrics in this respect are, for example, bytes per flow or the number of flows over time.

Our contribution is twofold. Firstly, based on an extensive data analysis, we have identified both i) relevant flow-derived variables that are indicative of tunnel activities and ii) a set of non-parametric statistical tests suitable for effective detecting DNS tunnel usage. Our detection approach targets different temporal and statistical characteristics of malicious traffic, namely peaks in traffic volume, changes in means and changes in distribution. Secondly, we have extensively tested our approach in an operational environment using different datasets for training and evaluation. Our validation has shown that we are able to detect diversified tunnel usage scenarios with a high detection rate. In a follow-up research we recommend to apply proposed detectors in alternative environments. This would only require training of the relevant parameters (e.g., thresholds, optimal bin size).

**Acknowledgements.** This research was performed within the Cyber Attack Detector (CAD) project which was a joint effort of the organizations authoring this paper, Fox-IT and the Netherlands National Police Department (KLPD).

The CAD project was partially funded by the Dutch Ministry of Economic Affairs, Agriculture and Innovation as part of the Maatschappelijke Innovatie Agenda Veiligheid (IMV1100032).

## References

1. Cisco.com: Cisco ios netflow configuration guide, release 12.4. (September 2010), `http://www.cisco.com`
2. Quittek, J., Zseby, T., Claise, B., Zander, S.: Requirements for IP Flow Information Export (IPFIX). RFC 3917, Informational (2004)
3. Sperotto, A., Schaffrath, G., Sadre, R., Morariu, C., Pras, A., Stiller, B.: An overview of ip flow-based intrusion detection. IEEE Communications Surveys & Tutorials 12(3), 343–356 (2010)
4. Sperotto, A., Mandjes, M.R.H., Sadre, R., de Boer, P.T., Pras, A.: Autonomic parameter tuning of anomaly-based idss: an ssh case study. IEEE Transactions on Network and Service Management 9, 128–141 (2012)
5. Mandjes, M., Żuraniewski, P.: $M/G/\infty$ transience, and its applications to overload detection. Performance Evaluation 68(6), 507–527 (2011)
6. Nussbaum, L., Neyron, P., Richard, O.: On Robust Covert Channels Inside DNS. In: Gritzalis, D., Lopez, J. (eds.) SEC 2009. IFIP AICT, vol. 297, pp. 51–62. Springer, Heidelberg (2009)
7. Aiello, M., Merlo, A., Papaleo, G.: Performance assessment and analysis of DNS tunneling tools. Logic Journal of IGPL (2012)
8. Crotti, M., Dusi, M., Gringoli, F., Salgarelli, L.: Detecting HTTP Tunnels with Statistical Mechanisms. In: IEEE International Conference on Communications (ICC 2007), pp. 6162–6168 (June 2007)
9. Dusi, M., Crotti, M., Gringoli, F., Salgarelli, L.: Tunnel Hunter: Detecting application-layer tunnels with statistical fingerprinting. Computer Networks 53(1), 81–97 (2009)
10. Marchal, S., Francois, J., Wagner, C., State, R., Dulaunoy, A., Engel, T., Festor, O.: DNSSM: A large scale passive DNS security monitoring framework. In: IEEE Network Operations and Management Symposium (NOMS 2012), pp. 988–993 (2012)
11. Karasaridis, A., Meier-Hellstern, K., Hoeflin, D.: NIS04-2: Detection of DNS Anomalies using Flow Data Analysis. In: IEEE International Conference on Global Telecommunications Conference (GLOBECOM 2006), pp. 1–6 (December 2006)
12. Callegari, C., Coluccia, A., D'Alconzo, A., Ellens, W., Giordano, S., Mandjes, M., Pagano, M., Pepe, T., Ricciato, F., Żuraniewski, P.: A methodological overview on anomaly detection. COST-TMA Book chapter (to appear, 2013)
13. Brodsky, B., Darkhovsky, B.: Nonparametric methods in change-point problems. Mathematics and Its Applications, vol. 243. Springer (1993)
14. Kolmogorov, A.: On the empirical determination of a distribution law (1933). In: Shiryayev, A. (ed.) Selected Works of A.N. Kolmogorov. Probability Theory and Mathematical Statistics, vol. II, pp. 139–146. Springer Netherlands (1992)