Detection of DNS Traffic Anomalies in Large Networks

Milan Čermák, Pavel Čeleda and Jan Vykopal

Institute of Computer Science, Masaryk University, Brno
Czech Republic, {cermak|celeda|vykopal}@ics.muni.cz

Abstract. Almost every Internet communication is preceded by a translation of a DNS name to an IP address. Therefore monitoring of DNS traffic can effectively extend capabilities of current methods for network traffic anomaly detection. In order to effectively monitor this traffic, we propose a new flow metering algorithm that saves resources of a flow exporter. Next, to show benefits of the DNS traffic monitoring for anomaly detection, we introduce novel detection methods using DNS extended flows. The evaluation of these methods shows that our approach not only reveals DNS anomalies but also scales well in a campus network.

Keywords: Domain Name System, DNS, IP Flow Monitoring, IPFIX, Traffic Anomaly Detection, Internet Measurements

1 Introduction

The Domain Name System (DNS) provides fundamental functions in directing Internet traffic today. Despite the fact that DNS concepts (RFC 1034, RFC 1035) are more than three decades old, DNS remains of the utmost importance for recent network technologies. Due to the wide use of DNS we can detect not only attacks based on DNS protocol security flaws but also other attacks reflected in the DNS traffic. Since the Internet has no borders, cyber-attacks which rely on DNS or are reflected in DNS traffic may come from anywhere and at any time.

Our research is mainly motivated by valuable information carried by a DNS protocol; there is high potential to use this information for network security monitoring. In order to successfully use DNS information, it is important to find out effective ways how to gather DNS data from monitored networks. In this paper, we attempt to answer the following research questions: *(i)* How can DNS traffic be effectively analysed in large networks? *(ii)* What are the differences in the analysis of DNS traffic using standard and extended flow records? *(iii)* What are the advantages of combinating DNS traffic information with flow records for network anomaly detection?

The contribution of our work is twofold: *(i)* We proposed and evaluated new algorithm to process flows with DNS information that can significantly reduce the number of DNS flow cache entries in current flow exporters. *(ii)* We introduced novel anomaly detection methods which use extended DNS flows to enhance the detection of network threats.

The paper is organized in five sections. Section 2 describes related work. Section 3 contains a description of approaches used for flow-based DNS traffic monitoring in large networks. Section 4 proposes new DNS traffic anomaly detection methods using standard and extended flows. Finally, Section 5 concludes the paper.

2 Related Work

To detect DNS traffic anomalies, it is important to determine where and how the data are gathered. A query logging on DNS servers represents the simplest way how to monitor DNS traffic without additional monitoring infrastructure. The analysis of server logs was presented in [2,19] including the optimization of this process for a large amount of logs. The main disadvantage of this approach is its inability to monitor traffic that does not pass through the monitored servers. To avoid this problem it is necessary to use network-based monitoring approaches [3,12,23] with probes installed in the network.

Methods for analysing DNS traffic collected from networks differ from the purpose of this analysis. One of these purposes is the collection of domain characteristics and their history. This information may be used for reverse lookups with IP addresses for which no reverse DNS records exists [21], for malicious domains detection [1,3,16,23] based on time-based features, answer-based features, or abnormal TTL values. The disadvantage of this approach for large network monitoring is its focus only on domains and not for the whole DNS traffic.

Network traffic statistics may be used to get general information about a DNS network's behaviour. These statistics can be created by tools such as `dnstop` [22], `dnsgraph` [17], or `DSCng` [10] which aggregates DNS data from packets and represents them as tables or charts. With these statistics, it is possible to detect misconfigured network devices or anomalies in traffic volume but the main drawback is the focus on the whole network and the inability to analyse the DNS traffic of one specific device or domain.

To analyse behaviour of specific device, flow-based approach could be used. Although flow records provide limited information about DNS traffic, some of the DNS anomalies can be still detected. For example, [6] suggests a method for DNS tunnelling detection using statistic tests or [8] presents the detection of cache poisoning attacks.

To obtain more specific information about DNS traffic it is necessary to store all important DNS packet fields such as source and destination addresses, queried domain name, or response data. This approach is used by [4,11,12] for the detection of botnets based on the same DNS behaviour of devices, abnormal DNS traffic or malicious domain usage. This type of data can be also used for an intrusion detection system based on DNS traffic monitoring which was introduced in [18]. The drawback of monitoring only DNS traffic is that we have no information about the other network communication of a device such as information about visiting the queried domain.

Another approach to detection is an analysis and correlation of DNS traffic with other network data. This approach transforms captured packets and whole traffic into events which are then processed by network anomaly detection methods. Such methods may be implemented, for instance, using The Bro Network Security Monitor [15].

3 Flow-based DNS Traffic Monitoring

3.1 Standard Flow Monitoring

There are two basic requirements for monitoring large and high-speed networks such as campuses or ISPs. First, monitoring tools must provide near real-time data analysis and, second, the tools must not demand large storage space. To fulfil these requirements, the concept of *network flow* is used. A flow is defined in RFC 7011 as "a set of IP packets passing an observation point in the network during a certain time interval, such that all packets belonging to a particular flow have a set of common properties". The standard flow record is a vector:

$$F = (IP_{src},\ IP_{dst},\ P_{src},\ P_{dst},\ Prot,\ T_{start},\ T_{dur},\ Pckts,\ Octs,\ Flags),$$

where the flow is defined by the source and destination IP addresses IP_{src} and IP_{dst}, source and destination ports P_{src} and P_{dst}, protocol $Prot$ and the start time T_{start} with duration T_{dur}. The fields $Pckts$ and $Octs$ represent the number of transferred packets and octets, and $Flags$ TCP flags.

The flow exporter aggregates packets with common properties into one flow until the flow is terminated. This termination can be caused by the expiration of flow cache entry (active time-out, idle time-out or resource constraints), natural expiration based on packet flags indicating connection end, emergency expiration or cache flush [7]. In networks with a large volume of traffic, it is necessary to have sufficiently large and free flow cache to avoid emergency expiration or cache flush, which may cause unwanted flow records split.

Flow acquisition can be done by common network devices that support flow record export, such as routers, or by specialized network probes [7] which provides greater data accuracy and are able to effectively process a large volume of traffic. Figure 1 depicts a monitored network with the probes installed at the local network uplink and also inside the network. The probe aggregates packets and export them as flow records to the flow collector that provides tools for basic flow processing and analysis.

Although flow records do not contain information about application protocols, it is still possible to use them for monitoring DNS traffic. A DNS flow can be distinguished from others by port-based protocol identification that relies on the fact that the TCP and UDP port number 53 is assigned to the DNS protocol by IANA. This port number is by default used by DNS resolvers which listen to this port. DNS monitoring using standard flow records can reveal anomalies that affect the volume characteristics of transferred data. However, anomalies connected to DNS application data remain undetected. Another disadvantage is

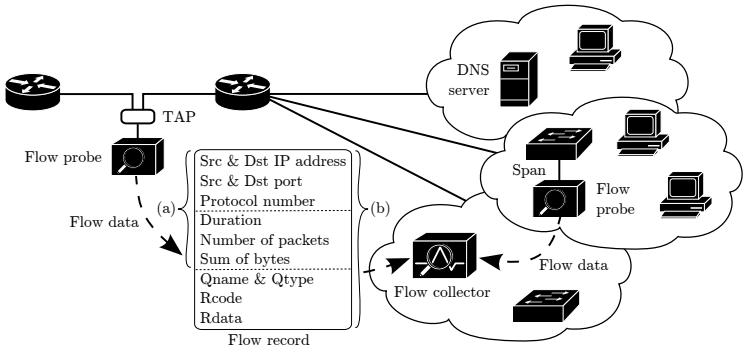

Fig. 1: Flow monitoring architecture with probes exporting (a) standard flow and (b) extended DNS flow record.

that the port 53 can also be used by other applications or protocols which may cause false positives. But this traffic usually forms only a small portion of the whole network traffic on this port.

3.2 Flows Extended by Information from a DNS Traffic

To answer the question *how can DNS traffic be effectively analysed in a large networks?*; we performed several measurements in the campus network of Masaryk University and in the Czech national research and education network, CESNET, which connects 27 Czech universities.

At first, we examined whether the flow monitoring concept is suitable for DNS traffic. Figure 2 shows the cumulative distribution function (CDF) of packets per flows which were collected in both networks over one day. Chart 2a shows that approximately 99 % of flows with the source or destination port 53 contain only one packet. This indicates that aggregation is not used. The rest of the flows carry DNS zone files, DNS tunnelling or other protocols. Through manual packet analysis, we found that one of these protocols is the BitTorrent protocol which exploits fact, that the traffic of port 53 is not restricted by network firewalls. To verify our results, we compute the same statistics depicted in chart 2b for CESNET. This network is primarily a transport network, therefore the traffic associated with port 53 contains a greater portion of other protocols than DNS. We also observed that a large amount of flows containing more than one packet with the destination port 53 are caused by attempted DNS amplification attacks which were performed almost constantly. To sum up, we observed that a typical DNS conversation consists of one packet carrying the DNS query and one packet carrying the DNS response.

Since the both DNS query and response are each represented by one flow record, it is possible to extend the standard flow record by DNS application data, such as queried domain name and type. This information does not disrupt the flow record and also does not excessively increase the flow record size.

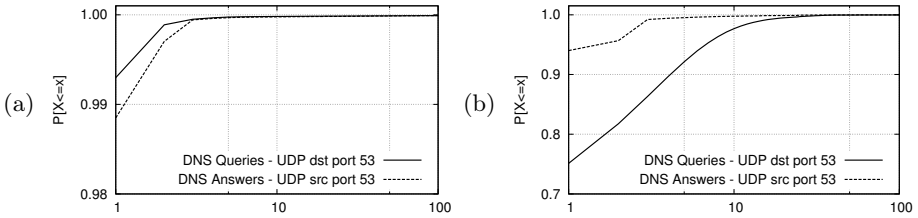

Fig. 2: CDF of DNS packets per flow observed in (a) the network of Masaryk University and (b) the Czech national research and education network, CESNET.

As a result, we could analyse DNS traffic together with other flows that can reveal traffic anomalies which otherwise would only have been detectable by deep packet inspection.

We identified that only four DNS packet fields are useful for most of the DNS traffic analysing methods: queried domain name *Qname*, queried record type *Qtype*, response return code *Rcode* and response itself *Rdata*. The others may unnecessarily increase the size of the flow record. Therefore, a DNS flow record contains the selected four fields:

$$F_{DNS} = (Qname,\ Qtype,\ Rcode,\ Rdata)$$

Because the DNS response may contain more than one answer, we recommend storing only the first answer with the same record type as a query or authoritative nameserver. For instance, in the event of a DNS query for the `A` record type, the flow record with DNS response will contain the address of the queried domain in the *Rdata* field.

3.3 Flow cache optimization using DNS extended flows

For efficient flow-based DNS traffic monitoring, we modified the standard algorithm of flow metering and export to fit the characteristics of DNS traffic. The flow cache plays a vital role in flow monitoring, but the performance of current implementations is constrained by its limited size. The translation of domain name precedes most of the network connections so we believe that DNS traffic represents a significant part of all collected flows. Storing DNS flow records in the flow cache leads to its rapid exhaustion in a very short time so we propose a modified algorithm that saves storage space in the flow cache by exporting extended DNS flow records immediately after the packet is parsed.

The algorithm checks if the packet is coming to/from the port 53 and protocol UDP. If these conditions are fulfilled, it is necessary to decide if the packet really carries a DNS payload which could be distinguished by DNS header analysis. If the packet carries a DNS payload then F_{DNS} is obtained and concatenated with a standard flow record F generated at the beginning of the algorithm. The resulting flow $F_{ext} = F \cdot F_{DNS}$ is immediately exported as an extended DNS

flow record to the collector. Otherwise, the standard flow records are stored in the flow cache.

In order to investigate the impact of the algorithm, we measured, in the network of Masaryk University and CESNET, the portion of flows using port 53 in the whole traffic. We observed that approximately 20 % and 15 % of all flows are flows possibly containing DNS traffic. In the proposed algorithm the DNS flows are not stored in the flow cache, so the algorithm saves up to 20 % of cache storage space. It can significantly help to prevent forced cache flush which causes that flow records which were originally split are now in an one record. Another advantage is that the flows extended by DNS data can be analysed in real-time because there is no need to wait for exporting timeouts. This means it is possible to detect some suspicious DNS traffic at the beginning and prevent potential damage.

4 DNS Traffic Anomaly Detection

In this section, we present several methods for the detection of DNS traffic anomalies. We first briefly discuss detection based on standard flows and then provide more details about novel methods which employ DNS extended flows. The methods were implemented as Perl scripts for an IPFIX collector and are available at [20]. In our implementation was used DNS flow data acquired by [9].

For a clear description of the proposed methods we will refer to Figure 3 which represents the general schema of our monitoring architecture, including the roles of individual devices.

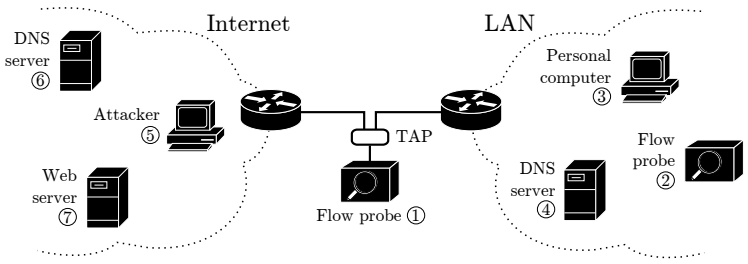

Fig. 3: General network schema with device roles.

4.1 Anomaly Detection Using Standard Flows

Although standard flow records do not contain DNS application data, it is still possible to detect some attacks targeting DNS infrastructure. The DNS amplification DDoS attack represents one of the most used network attacks involving DNS infrastructure. This attack is characterised by a large amount of same queries with spoofed IP address coming from attacker ⑤ passing to a rogue open

DNS resolver. This open resolver is a misconfigured DNS server ④ or device infected by malware ③ that acts as a rogue DNS resolver. It responds to all queries by an abnormally large packet payload that contains answers. Thus, an increasing count of flows, with high bytes-per-packet ratio and the source port 53, may indicate this type of attack.

In well-maintained networks, we can use detection techniques based on access control lists reflecting network security policy. Based on this knowledge, it is possible to use flow-based methods which report every communication from or to a DNS server out of the list. This communication may be caused by a malware-infected device ③ which operates as a rogue DNS resolver. Another example is a malicious change of device settings which causes a local DNS resolver ④ to be replaced by another ⑤ which returns incorrect answers referring to fraudulent websites.

Since flows identified only by the usage of port 53 may contain different application data than DNS, it is necessary to specify a threshold indicating when suspicious traffic could be identified as anomaly to avoid false positives. This may cause stealth DNS amplification DDoS attacks to not be detected. In Masaryk university's network we observed 8 examples of this kind of attack during three months of testing. Detection using standard flows is difficult in large and not well-maintained networks where it is hard to distinguish DNS servers from clients. In such networks, the DNS server may be incorrectly identified as open DNS resolver even if the server only correctly responds to the DNS query that contains the local domain.

4.2 Anomaly Detection Using Extended Flows

Flow records extended by DNS information can be analysed as standard flows using basic Top-N statistics of the entire local network or individual hosts. Statistics related to DNS traffic may include queried record types, return codes or, for example, most queried domain names. Any major change in these statistics may indicate abnormal behaviour. For instance, an increasing number of DNS error return codes can be caused by malfunctioning devices, or a large number of MX record queries not originating from a local e-mail server can even indicate malware-infected device that attempts to send spam.

The main advantage of using extended flow records for DNS monitoring is that these data can be analysed together with other flow records. We can search the corresponding communication in standard flows based on a returned IP address in a DNS response and confirm the visitation of a queried domain.

The combination of DNS extended flows with standard flows can be also used for tracing the originator of a query even though the DNS flow exporter ① is used only at the network edge, i. e. all DNS queries have the same source address. It is possible to use DNS responses containing the IP address of a queried domain and check if a device started communication with this address. It is very likely that it is the same device which performed the query. The disadvantage of the presented method is that device must visit the queried domain, otherwise, it is still impossible to trace the originator of the query.

Using DNS extended flows does not enable only detection of a visit to the queried domain or trace the device performing a query. DNS extended flows also enable detection of advanced network attacks and anomalies which are hardly, or not, detectable by standard flows. To show advanced examples and the *advantages of combination DNS traffic information with flow records for network anomaly detection*, we proposed several novel detection methods focusing on open resolvers, non-local DNS resolver usage, or malware domains queries. These methods are independent of the version of the IP protocol and thus it is possible to deploy them easy in IPv6 networks.

Method 1: Open DNS Resolvers Detection

Amplification DDoS attacks using open DNS resolvers are currently widely used by attackers because they can generate small packets and easily make a service inaccessible. The detection of this attack using standard flow requires a high threshold to avoid false positives. We are able to reduce the threshold by detecting the same queried domains using DNS extended flows but a threshold is still required.

The detection of an open DNS resolver can be easily done in small and documented networks by observing traffic of recognised DNS servers. In large or not well-maintained networks a list of recognised DNS servers may not exist. For this purpose, we propose a new detection method based on DNS extended flows. The method is described in the Algorithm 1. The main challenge is to distinguish an open DNS resolver ③ from a regular DNS server ④ which responds to a query containing a local domain. For this purpose, the method analyses all DNS responses observed at the network edge ① and checks if the domain is assigned to the monitored network. This check is done by requesting the local DNS resolver ④ for `ANY` record type of this domain. If the result does not contain at least one record with an IP address from the monitored network then the DNS server is reported as an open DNS resolver, otherwise the domain is added to the local domains list.

Algorithm 1 *Open DNS Resolver Detection*

function GetOpenDNSResolver (W : local domains, L : local network, F_{ext} : analysed flows)
$F_{responses} = \{F_{ext} \mid F_{ext}.IP_{src} = L \wedge F_{ext}.IP_{dst} \neq L \wedge F_{ext}.P_{src} = 53 \wedge F_{ext}.P_{dst} \neq 53 \wedge F_{ext}.Qname \neq W_1 \wedge \cdots \wedge F_{ext}.Qname \neq W_n \wedge F_{ext}.Rcode = 0\}$;
aggregate $F_{responses}$ by IP_{src} and $Qname$ to $F_{resolvers}$;
for each $F_{resolver}$ in $F_{resolvers}$ **do**
    request all information about domain $F_{resolver}.Qname$ by ANY query type;
    if domain information contain IP address from L **then**
        add $F_{resolver}.Qname$ to W ;
    else
        return "$F_{resolver}.IP_{src}$ is open DNS resolver" ;
    end if
end for

The advantage of the presented method is that we are able to detect an open DNS resolver by observing only one response. Over three months of testing in the campus network of Masaryk university, we observed 207 IP addresses operating as open DNS resolvers. In the same period, the Open Resolver Scanning Project[1] reported 76 addresses for this network. The different amount of addresses is caused by the fact that Open Resolver Scanning Project performs scans only once per day. An interesting side effect of the method is that we discovered all domains that are hosted in the campus network.

Method 2: External DNS Resolver Usage Detection

The use of an external DNS resolver ⑥ instead of the local network DNS resolver ④ may cause delay and also presents a security risk if the external DNS resolver responds with fraudulent IP addresses. In large, not well-maintained networks, it is necessary to distinguish between a client device ③ and a local DNS resolver ④, which tries to resolve a queried domain. The proposed Algorithm 2 utilizes the fact that the DNS resolver performs only queries and the client visits the queried domain. The visit is checked by finding standard flows with communication between the client and the queried domain ⑦ , which starts within approximately two seconds of the query. If the client did not visit first N selected domains then it is marked as a possible DNS server.

Algorithm 2 *External Resolver Usage Detection*

1: **function** GetClientsUsingExternalDNS (N : number of checked domains, L : local network, F_{ext} : analysed flows)
2: $F_{responses} = \{F_{ext} \mid F_{ext}.IP_{src} \neq L \wedge F_{ext}.IP_{dst} = L \wedge F_{ext}.P_{src} = 53 \ \wedge F_{ext}.P_{dst} \neq 53 \wedge F_{ext}.Rcode = 0 \wedge (F_{ext}.Qtype = \text{ A } \vee \ F_{ext}.Qtype = \text{ AAAA})\}$;
3: sort $F_{responses}$ by IP_{dst} to $F_{responses\_sorted}$;
4: **for each** $F_{response}$ in $F_{responses_sorted}$ **do**
5: $F_{com} = \{F_{response} \mid F_{ext}.IP_{src} = F_{response}.IP_{dst} \wedge F_{ext}.IP_{dst} = F_{response}.Rdata \wedge F_{ext}.T_{start} \geq F_{response}.T_{start} \wedge F_{ext}.T_{start} \leq (F_{response}.T_{start} + 2\ sec)\}$;
6: **if** number of flows $F_{com} > 0$ **then**
7: **return** "$F_{response}.IP_{dst}$ uses external resolver $F_{response}.IP_{src}$" ;
8: **end if**
9: **if** $F_{response}.IP_{dst}$ was seen N times **then**
10: go to the next $F_{response}.IP_{dst}$;
11: **end if**
12: **end for**

During the evaluation of the method in Masaryk university's network, we found that the most used DNS resolvers are public DNS resolvers operated by Google or OpenDNS. The rest were DNS resolvers of local network providers or antivirus solutions, which offer DNS resolvers as a part of user protection. We also found several malicious DNS resolvers which returned forged IP addresses of popular web pages.

[1] https://dnsscan.shadowserver.org/

Method 3: Malware Domains Query Detection

The detection of malware domains is one of the most used detection techniques based on information from DNS traffic. The detection is based on testing whether the queried domain is contained in a blacklist of known malware domains. Such inspection may be very time consuming in networks with a large amount of traffic. For these type of networks, we suggest shrinking number of checked domains only to domains queried after a device starts up. We suppose that most malware is launched automatically at the device start and attempt to immediately contact its command and control centres or download more malware.

Algorithm 3 *Malware Domains Queries Detection*

1: **function** GetMalwareAffectedDevices (N : number of checked domains, F_{ext} : analysed flows)
2: $F_{queries} = \{F_{ext} \mid F_{ext}.P_{src} \neq 53 \wedge F_{ext}.P_{dst} = 53 \wedge F_{ext}.Qname = dns.msftncsi.com \wedge (F_{ext}.Qtype = \text{ A } \vee \ F_{ext}.Qtype = \text{ AAAA})\}$;
3: aggregate $F_{queries}$ by IP_{src} to F_{starts} ;
4: **for each** F_{start} in F_{starts} **do**
5: $F_{domains} = \{F_{start} \mid F_{ext}.IP_{src} = F_{start}.IP_{src} \wedge F_{ext}.P_{src} \neq 53 \wedge F_{ext}.P_{dst} = 53 \wedge F_{ext}.T_{start} \geq F_{start}.T_{start} \wedge F_{ext}.T_{start} \leq (F_{ext}.T_{start} + 5\ minutes) \wedge F_{ext}.Qname \neq *windowsupdate.com \wedge F_{ext}.Qname \neq *msftncsi.com \wedge F_{ext}.Qname \neq *microsoft.com$;
6: select first N queried domains D from $F_{domains}$;
7: **for** all queried domains D **do**
8: exclude $D.Qname$ contained in the Alexa top domains list ;
9: check if domain $D.Qname$ is reported as malware domain ;
10: **if** $D.Qname$ is marked as malware domain **then**
11: **return** "$F_{start}.IP_{src}$ queried malware domain $D.Qname$" ;
12: **end if**
13: **end for**
14: **end for**

The device start can not be easily detected using standard flows, but with the DNS extended flows we discovered that the Windows operating systems ③ immediately query the domain `dns.msftncsi.com` to check if the configured DNS resolver ④ works. The proposed method of testing domains queried after the device startup is described in the Algorithm 3. To avoid unnecessary checks, we suggest excluding domains which are associated with Microsoft services and also the most used domains from the *Alexa Top Domains List*[2]. The rest of the domains are checked in our implementation whether they are listed on several blacklists by the *VirusTotal*[3] service.

The evaluation of the method showed that the checked domain must occur almost in 4 blacklists used by VirusTotal to avoid false positives because there were several blacklists marked by users, which are unreliable. In our campus network, we detected one device which was reported as infected by malware and also operated as an open DNS resolver.

[2] http://www.alexa.com/topsites
[3] https://www.virustotal.com/#url

5 Conclusion

We have presented an effective technique for DNS traffic monitoring in large networks based on the extension of standard flows. For these DNS extended flows we introduced examples of new anomaly detection techniques able to detect anomalies that were previously hard to detect using standard flows. To conclude our paper, we shall now summarize our research questions and answers to them.

As an answer to the research question *how can DNS traffic be effectively analysed in a large networks?*, we propose using a flow based monitoring approach. We suggest extending standard flow by four new fields from DNS application data. To gather these data we proposed a new flow exporting algorithm respecting DNS traffic to be able to effectively save space in the flow cache which plays vital role in the flow metering process. Our algorithm enables the analysis of DNS data in real-time in contrast to standard flows.

To show *differences in the analysis of DNS traffic using standard and extended flow records*, which was our second research question, we introduced novel methods of DNS traffic anomaly detection using standard and DNS extended flows. The methods using standard flows are limited by the port identification and allows only the analysis of basic flow characteristics. On the other hand, DNS extended flows enable us to clearly identify DNS traffic and use DNS application data as a basis for detections.

The DNS extended flows can be analysed together with standard flows which allows us to make detection methods more accurate. To demonstrate the *advantages of combinating DNS traffic information with flow records for network anomaly detection*, we introduced new detection methods utilizing the fact that DNS query can be combined with flows containing communication with queried domains. Thus, it is, for example, possible to check if a device really visited the queried domain.

The presented paper shows that DNS extended flows are a suitable extension of standard flows that may help in the detection of network traffic anomalies. In future work, we plan to use DNS extended flows for detecting other DNS traffic anomalies such as DNS tunnelling, or for advanced malware infected devices detection. We also plan to examine drawbacks and potential backdoors of proposed methods and provide appropriate solutions to them.

Acknowledgments. This material is based upon work supported by Cybernetic Proving Ground project (VG20132015103) funded by the Ministry of the Interior of the Czech Republic.

References

1. Antonakakis, M., Perdisci, R., Dagon, D., Lee, W., Feamster, N.: Building a Dynamic Reputation System for DNS. In: USENIX security symposium. pp. 273–290 (2010)

2. Begleiter, R., Elovici, Y., Hollander, Y., Mendelson, O., Rokach, L., Saltzman, R.: A Fast and Scalable Method for Threat Detection in Large-scale DNS Logs. In: Big Data, 2013 IEEE International Conference on. pp. 738–741 (Oct 2013)
3. Bilge, L., Sen, S., Balzarotti, D., Kirda, E., Kruegel, C.: Exposure: A Passive DNS Analysis Service to Detect and Report Malicious Domains. ACM Trans. Inf. Syst. Secur. 16(4), 14:1–14:28 (Apr 2014), http://doi.acm.org/10.1145/2584679
4. Choi, H., Lee, H.: Identifying botnets by capturing group activities in dns traffic. Comput. Netw. 56(1), 20–33 (Jan 2012)
5. Ellens, W., Żuraniewski, P., Sperotto, A., Schotanus, H., Mandjes, M., Meeuwissen, E.: FlowBbased Detection of DNS Tunnels. In: Emerging Management Mechanisms for the Future Internet, pp. 124–135. Springer (2013)
6. Hofstede, R., Čeleda, P., Trammell, B., Drago, I., Sadre, R., Sperotto, A., Pras, A.: Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX. IEEE Communications Surveys & Tutorials (2014)
7. Karasaridis, A., Meier-Hellstern, K., Hoeflin, D.: Detection of DNS Anomalies using Flow Data Analysis. In: Global Telecommunications Conference, 2006. GLOBECOM'06. IEEE. pp. 1–6. IEEE (2006)
8. Kováčik, M.: DNS plugin (2014), https://www.liberouter.org/technologies/dns-plugin/
9. Košata, B., Čermák, J., Surý, O., Filip, O.: DSCng: DNS server monitoring program (2013), http://www.dscng.cz/
10. Manasrah, A.M., Hasan, A., Abouabdalla, O.A., Ramadass, S.: Detecting Botnet Activities Based on Abnormal DNS Traffic. (IJCSIS) International Journal of Computer Science & Information Security 6(1) (Oct 2009)
11. Marchal, S., Francois, J., Wagner, C., State, R., Dulaunoy, A., Engel, T., Festor, O.: DNSSM: A Large Scale Passive DNS Security Monitoring Framework. In: Network Operations and Management Symposium (NOMS), 2012 IEEE. pp. 988–993 (Apr 2012)
12. Paxson, V.: Bro: a System for Detecting Network Intruders in Real-Time. Computer Networks 31(23-24), 2435–2463 (1999)
13. Perdisci, R., Corona, I., Giacinto, G.: Early Detection of Malicious Flux Networks via Large-Scale Passive DNS Traffic Aanalysis. Dependable and Secure Computing, IEEE Transactions on 9(5), 714–726 (2012)
14. Qu, J., Sztoch, P.: dnsgraph (2003), http://dnsgraph.sourceforge.net/
15. Schonewille, A., van Helmond, D.J.: The Domain Name Service as an IDS. Research Project for the Master System-and Network Engineering at the University of Amsterdam (2006)
16. Snyder, M., Sundaram, R., Thakur, M.: Preprocessing DNS Log Data for Effective Data Mining. In: Communications, 2009. ICC '09. IEEE International Conference on. pp. 1–5 (June 2009)
17. Čermák, M.: DNSAnomDet (2014), https://is.muni.cz/publication/1131184
18. Weimer, F.: Passive dns replication. In: FIRST Conference on Computer Security Incident (2005)
19. Wessels, D.: Dnstop: Stay on Top of Your DNS Traffic (2013), http://dns.measurement-factory.com/tools/dnstop/
20. Zdrnja, B., Brownlee, N., Wessels, D.: Passive Monitoring of DNS Anomalies. In: Detection of Intrusions and Malware, and Vulnerability Assessment, pp. 129–139. Springer (2007)