DNSxD: Detecting Data Exfiltration over DNS

DNSxD: Detecting Data Exfiltration Over DNS

Jacob Steadman and Sandra Scott-Hayward
Centre for Secure Information Technologies (CSIT), Queen's University Belfast, Belfast, BT3 9DT, N. Ireland
Email: jsteadman01@qub.ac.uk, s.scott-hayward@qub.ac.uk

***Abstract*—According to a 2017 SANS report, 1 in 20 organisations fall victim to data exfiltration. Data exfiltration, often the final stage of a cyber attack has damaging consequences for the victim organisation. The use of the Domain Name System (DNS) protocol for data exfiltration was first discussed in 1998. Twenty years on, this covert transmission method has become more sophisticated as malicious actors adapt to evade detection techniques. The popularity of DNS for data exfiltration is due to the essential nature of the protocol for network communication. This paper addresses the issue of DNS-based data exfiltration proposing a detection and mitigation method leveraging the Software-Defined Network (SDN) architecture. Popular DNS data exfiltration attacks and current exfiltration detection mechanisms are analysed to generate a feature-set for DNS data exfiltration detection. The DNSxD application is presented and its performance evaluated in comparison with the current exfiltration detection mechanisms.**

I. Introduction

In the 2017 SANS data protection survey [1], it is reported that 12% of organisations surveyed recorded a security breach, and 43% of those breaches involved data exfiltration. This is 1 in 20 organisations that fall victim to data exfiltration attacks. The introduction of tighter government regulations, such as the General Data Protection Regulation (GDPR) in the EU, has put renewed pressure on organisations to protect their data. The challenge that organisations face is the ability to detect and respond to data breaches within an appropriate time [2]. Threats from both external actors and insiders are becoming more sophisticated and organisations must respond with increased security measures.

One method of covert data exfiltration exploits a fundamental service of the internet; that of the Domain Name System (DNS) protocol. The purpose of the protocol is to convert human-readable domain names into IP addresses in order to initiate communication between two endpoints. To do this, a request to resolve a domain name is transmitted to a DNS resolver. This request is processed by one, or multiple, recursive DNS servers to identify the domain name-IP address mapping. As these transactions are considered normal communication within and between networks, they will not be filtered by a firewall or detected by IDPS (Intrusion Detection/Prevention System), or other security controls. The attacker can, therefore, exploit this and insert into the domain name the segments of data to be exfiltrated. The essential nature of the DNS protocol means that it is often difficult to block such attacks without negatively impacting legitimate network functionality. Therefore, many security controls are not configured to detect this covert channel. The DNS channel also enables initiation of Command & Control (C&C) channels for a malicious actor to communicate with malware on a victim's machine that is awaiting instruction.

According to a 2016 Cisco security report [3], 91% of malware used DNS in attacks and 68% of organisations failed to monitor DNS traffic as a method of attack. According to [4], the first discussion of the DNS protocol being used as an exfiltration method occurred in 1998. Since then, the attack has grown in notoriety, mainly due to its role in malware attacks. However, mitigation has not kept pace with the growth in the attack vector with the majority of networks still vulnerable to even fundamental attacks. This is both due to the service provided by DNS and due to the fact that most current detection methods could be circumvented by more sophisticated exfiltration tools. Furthermore, most current detection techniques could be evaded if the attacker had knowledge of the identifiers used in the detection process.

In order to remove the attacker advantage, we propose a solution built on the emerging software-defined network (SDN) technology. The potential to improve network security using SDN capabilities of programmability and logically centralized control has been identified [5]. With the DNSxD solution, the SDN architecture is exploited to enable detailed collection and analysis of DNS data exfiltration attack identifiers.

The paper is structured as follows: Section II describes current work related to DNS data exfiltration detection. Section III presents the DNS data exfiltration attack, popular open-source tools, and current detection methods. An analysis of potential identifying features is presented in section IV. In Section V, the DNSxD SDN-based DNS data exfiltration detection and mitigation solution is introduced. An evaluation and discussion of the performance of the proposed DNSxD solution is provided in Section VI. Finally, Section VII concludes the paper.

II. Related Work

In [11], Bromberger discusses DNS data exfiltration and potential characteristics that can be used to identify an attack. Their work has been extended by several subsequent papers. The features used are presented in Table I and discussed in Section IV.

Farnham and Atlasis [4] further develop the concepts introduced by Bromberger. They reviewed several exfiltration tools that were common at the time (2013) and developed a solution to detect exfiltration by these tools. They proposed two methods of detection; packet inspection and traffic analysis.

Jaworski [6], also built on Farnham's approach [4], proposing a method of using the Splunk data analysis engine to

TABLE I: Features used in DNS data exfiltration Detection (PI: Packet Inspection, TA: Traffic Analysis)

	Farnham [4]	Jaworski [6]	Liu [7]	Arashloo [8]	Infoblox [9]	Snort [10]	DNSxD
PI- Statistical analysis		X	X		X		X
PI- Query length			X		X	X	X
PI- Record types							X
PI- Tool signatures	X						
TA- Frequency per domain					X		X
TA- Orphan DNS requests			X				
TA- Keep-alive				X			
TA- Time interval			X				
TA- Volume of requests		X				X	X
TA- Domains per IP				X			
TA- IP per domain				X			
TA- Hosts per domain	X						
TA- Unauthorised servers		X					
TA- Blacklist servers		X				X	

analyse larger sets of potential attack identifiers to create a more robust solution. Several methods of payload and traffic analysis are employed. However, no performance evaluation is provided. More recently, Liu et al. [7] focus on the use of machine learning (ML) methods. They extract a feature set from analysis of DNS tunnels. Their ML algorithms are trained to identify attacks based on this feature set with a reported 99% success rate. However, the development of the feature set at the core of the solution is based only on high throughput tunnelling tools which may not prove effective against attacks with a lower exfiltration rate, as discussed in section III.

Arashloo et al. [8] take a slightly different approach based on the concept of a stateful data plane in SDN. They explore the ability for some detection functionality to be passed back to the switches to limit the volume of traffic forwarded to the SDN controller. The authors provide several security policy examples, including detecting malicious domains through traffic analysis. However, no performance evaluation is provided.

Kara et al. [12] discuss the use of DNS as a means of malicious payload distribution. Their solution relies on malicious DNS communication channels using uncommon DNS record types. They demonstrate a clear correlation in the distribution of record types between legitimate and malicious DNS traffic, which can be used to profile the attack. Nadler et al [13] split DNS data exfiltration attacks into two categories; high-throughput tunnelling, and low-throughput exfiltration attacks. Their solution logs specific DNS features for correlation on an hourly basis to determine if an attack has/is taking place. O'Connor et al. [14] propose PivotWall, a network security control that extends from the individual hosts into the network devices. The solution requires files to be labelled as containing sensitive data. Interaction with these files triggers a packet labelling mechanism to allow the network devices to determine if a packet contains sensitive information. Finally, Sheridan and Keane [15] discuss several methods of increasing the stealth by which DNS data exfiltration attacks operate. In order to achieve this, they propose a number of traffic blending techniques to better disguise the malicious DNS traffic with normal DNS traffic behaviour.

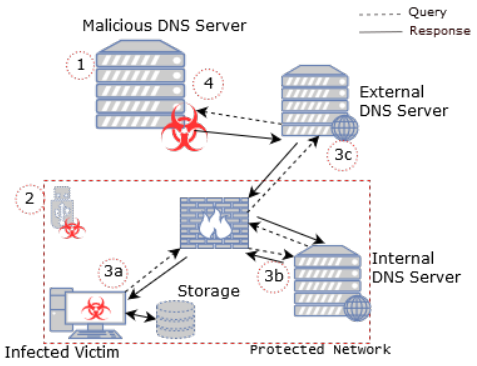


Fig. 1: Steps in the DNS data exfiltration Attack

III. DNS Data Exfiltration Techniques

There are a number of stages in a DNS data exfiltration attack and a number of elements required by the attacker. The steps in the attack are illustrated in Figure 1:

- 1: The attacker must have control of an authoritative DNS server, which will be the end point for the DNS requests transmitted from the victim.
- 2: Exfiltration tools consist of two component applications; the client on the victim machine, and the server controlled by the attacker. The victim machine must be infected with the client tool. For example, within a malware payload.
- 3a: With control of the victim's machine, the exfiltration tool is used to encapsulate selected data from the victim into DNS requests.
- 3b/c: The request is forwarded through DNS servers until it reaches the attacker. This reflects standard DNS operation i.e. if a server is unable to resolve the DNS request, it forwards it to a higher-level DNS server. Note: this process can involve many more DNS servers than shown in Figure 1.
- 4: Once the request reaches the authoritative DNS server, the server side of the exfiltration tool strips the data from the request and reconstructs it into the original format.

The attack can be separated into 3 subcategories. A range of DNS data exfiltration tools which represent each subcategory

have been selected for analysis. These are presented in Table II.

TABLE II: Features of DNS data exfiltration Tools

	Iodine [16]	**DnsCat2 [17]**	**DET [18]**	**DNSExfiltrator [19]**
Default DNS types	Specifiable	Specifiable	A	TXT
Encoded data	Base32/64/128	N/A	N/A	Base64
Encrypted data	N/A	Custom	N/A	RC4
Client polling	Yes	Yes	N/A	N/A
Query length (bytes)	255	50 polling, 227 download	67	Specifiable
Frequency (ppm)	657	102	160	Specifiable
Max Entropy	4.3	4.1	3.8	5.8

Exfiltration: Pure data exfiltration is the most basic form of the attack. Each DNS packet encapsulates data which is reconstructed at the authoritative DNS server owned by the malicious actor. The communication channel is uni-directional and does not require any specially crafted replies. The simplicity of the attack provides increased potential for variation. Specifically, this gives the attacker greater means of evading detection by limiting the amount of data transmitted in each packet, throttling the frequency of transmission, and using common query types (i.e. A, AAAA). DNSExfiltrator [19] has been selected for analysis as it is a pure exfiltration tool with the ability to apply further obfuscation to the attack. In addition to this, the Data Exfiltration Toolkit (DET) has been selected, which has the ability to spread the exfiltration across several protocols, and services.This feature is out of scope for this paper but will be considered in future work. **Tunnelling:** Unlike pure exfiltration, tunnelling allows a malicious actor to encapsulate another protocol within the DNS traffic (e.g. HTTP, FTP). This reduces the variability that can be introduced into the transmission process as the rate and volume of data in each packet is tightly coupled to the protocol being tunnelled. The high frequency and volume of traffic generated produces an easily identifiable traffic pattern. In addition to this, the most frequently occurring query types in benign DNS traffic are A and AAAA queries which translate domain names into IPv4 and IPv6 addresses. However, these do not provide a space to transmit the response from the tunnelled protocol. To navigate this problem, uncommon query types are used, which contain a space for reply data (e.g. Null). This provides another feature for identifying this category of attack. Iodine [16], has been selected to represent this subcategory. **Command & Control:** C&C channels provide an attacker with the means to communicate with their malware after it has infected a victim. It is common for DNS traffic to be used as a covert method of creating this communication channel [3]. The issue is that DNS traffic is generated from client to server, making it impossible for the malicious actor to initiate communication when they need to send their malware a command. To achieve this, the client continuously sends polling packets to the server at a predetermined interval allowing the attacker to respond with a command when needed. This polling provides an easily identifiable traffic pattern. DnsCat2 [17] provides this functionality.

A. Current Detection Methods

As described in Section II, several solutions have previously been proposed to tackle the threat of data exfiltration over DNS. Table I presents each of the features discussed in previous work. From the distribution, it is clear that a single set of features has not prevailed over the others. Furthermore, previous work such as [4] [7] focussed solely on the detection of DNS tunnels, which, as discussed in section III does not represent the most challenging version of the attack. In Section IV we present an analysis of these features highlighting the most relevant detection features for modern variants of the attack.

There are several other limitations that affect some of the most recent solutions discussed so far. In [13] Nadler et al. focus solely on the detection of low-throughput attacks by implementing a one-hour monitor window. To avoid this issue, DNSxD implements a much smaller time frame and relies on other Deep Packet Inspection (DPI) identifiers to detect low throughput attacks. In [14] O'Connor et al. suggest a solution which relies on pre-labelled files. The issue with this approach is that it may prove ineffective against insider threats, or if an attacker can copy the data from a labelled file to a new location prior to exfiltration.

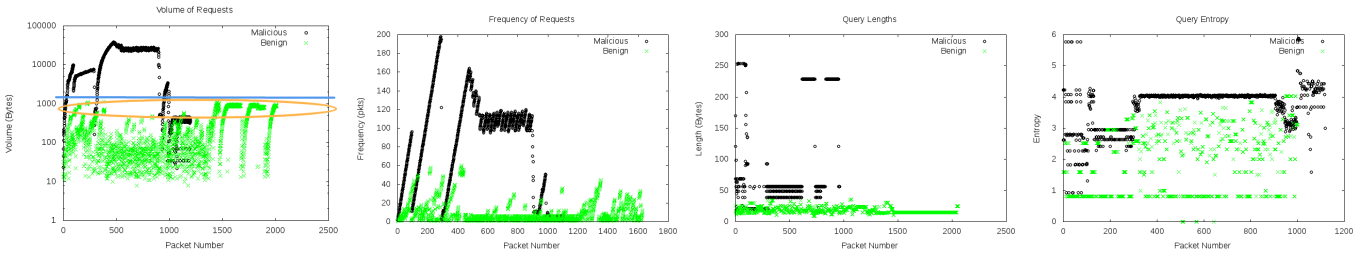
IV. Feature Analysis

From the list presented in Table I, several features have been identified as being no longer relevant to the attack, or out of scope for this application.
Tool signatures: too heavily reliant on the identification of a single existing exfiltration tool, making it easy for an attacker to avoid. **Orphan DNS requests:** require further analysis of other protocols in order to connect FQDNs with their uses, e.g. HTTP requests, which is deemed out of scope for this work. **Domains per IP:** most exfiltration tools analysed do not return domain-IP mappings. Therefore, the only IP known to DNSxD is that of the closest recursive DNS server. **Hosts per domain:** by encapsulating data in the domain name each request generates a new "host" section of the query. Therefore, this feature returns the same results as monitoring frequency. **Un-authorised servers:** a correct attack set-up should bypass this by using authorised servers. **Blacklist domains:** only applicable if malicious domains are known, and even then it is trivial for an attacker to change the domain on a regular basis to avoid detection.

DNSxD then outputs the distribution of values for each remaining feature to distinguish how malicious and benign

TABLE III: Selected Features & Threshold Values

	Threshold	**Suspicious**
Record type	Null	Not A or AAAA
Query length	55B	40-50B
Name entropy	4	3-4
Volume	1500B	300-1500B
Frequency per domain	100pkts	20-100pkts
Keep-alive	N/A	N/A
Time interval	N/A	N/A



(a) Volume per Domain (b) Frequency per Domain (c) Query Lengths (d) Query Entropy

Fig. 2: Statistical Analysis of Features

traffic patterns differ. An example of this feature analysis is depicted in Figure 2, which compares the output of known attacks against benign traffic during the same period[1]. Several of the features allowed for the creation of a threshold value, as highlighted by the horizontal line in Figure 2a. In addition to this, there are also several "suspicious" zones which legitimate traffic only occasionally enters (encircled in Figure 2a). Table III shows the final values selected for each remaining feature. A combination of several suspicious feature values can correctly identify attacks without generating false positives. This also reduces the ability for an attacker to obfuscate their attack by reducing the query size, volume, or frequency to match benign traffic. Excessive obfuscation of any feature to effectively blend with legitimate traffic will have an impact on the other features, increasing detectability via that feature. For example, reducing the size of each packet will require a greater number and/or frequency of packets.

V. DNSxD System Architecture

Based on the related work discussed in section II, there is still a need for a real-time detection mechanism for low-throughput pure data exfiltration attacks. Using the features analysed in section IV we propose DNSxD as a detection method that remains robust against more sophisticated DNS data exfiltration attacks. The solution takes advantage of the SDN architecture and has been developed as an OpenDaylight (ODL) [20] application to detect and mitigate DNS data exfiltration attacks. ODL is one of the most popular open-source controllers and has been adapted for use in production SDN deployments. On installation of the DNSxD application, a flow rule is sent to the relevant network devices (topology dependent) to direct all outbound DNS traffic to the controller. The outbound traffic, i.e. queries, provide a sufficient feature set to identify an attack, thus eliminating the requirement to mirror DNS replies as well. To ensure that the monitoring does not negatively impact legitimate traffic, two flow tables are employed. The first table contains two flow rules to redirect DNS request traffic to the controller, and direct all other traffic to the second flow table, for standard processing.

The DNSxD application processes packet-ins[2] from the network devices and applies a series of features based on the analyis presented in Section IV to maximize the detection capability, as follows:

- PI: Uncommon record type, query length, and query entropy
- TA: Volume and frequency of DNS requests

With the TA approach, a sliding monitor window is implemented to track the identifiers over a set period. There is an accuracy trade-off in the window length setting. Setting a shorter window can increase the detection rate and decrease mitigation time, and may reduce false positives (FPs) and false negatives (FNs) (i.e. triggering on benign traffic, or not triggering for malicious traffic). Although, for a slower exfiltration rate, the shorter monitor window is redundant. Conversely, a longer monitor window makes it more difficult for traffic blending to be an effective method of evading detection. However, the increase in FPs, due to the inability to distinguish accurate thresholds for DNS traffic, suggests that other identifiers may be required. For this reason, a short monitor window has been employed along with additional identifiers, such as entropy[3], to identify low throughput attacks.

Following DNSxD malicious flow detection, the data exfiltration is mitigated by the installation of a flow rule (on the relevant network node) to drop the malicious traffic. This is an immediate mitigation halting the data exfiltration, thus protecting the data and defeating the attack. From the perspective of malware and ransomware, this can prevent damage to the victim's files, and stop the malware from communicating with a C&C server, which also prevents malware propagating. Due to the recursive nature of the DNS protocol, it is not possible to install a flow rule that blocks the specific malicious server, as the network only communicates with either the internal DNS server, or the first DNS server in the chain that leads to the malicious server. This means that in mitigating the attack all DNS traffic from the infected host will be blocked until a security analyst can access the device and deal with the attack. A solution to this issue is discussed in Section VI.

In addition to performing intrusion detection and prevention, the application also performs continuous real time analysis and reporting on each feature, per host and per domain. These results, as shown in Figure 2, allow an analyst to adjust the

[1]The graphs present a subset (<2500pkts) of the full benign dataset used. The full set is discussed in section VI.

[2]UDP and DNS packet decoders were developed for ODL for the purpose of this application.

[3]The deep packet inspection necessary to extract the entropy of the domain name has a direct impact on the data privacy of network users. Detailed analysis of the data privacy impact is out of scope of this work. However, at a minimum, users should be made aware that domain names are being monitored, and the application data must be processed and stored securely.

values provided in Table III in order to account for the highly variable nature of DNS traffic in different networks.

VI. Evaluation

To evaluate the performance of the application, a test environment was created using the Mininet network virtualisation framework, as shown in Figure 3. The network consists of 5 switches, a total of 40 hosts, and an internal DNS server which acts as the first recursive DNS server for outbound DNS traffic. Each switch (Open vSwitch 2.5.2) is running OpenFlow1.3 supporting multiple flow tables. This provides a network suitable for running benign traffic and monitoring for DNS data exfiltration attacks. The network is controlled by an ODL Carbon controller [20] running the DNSxD application.

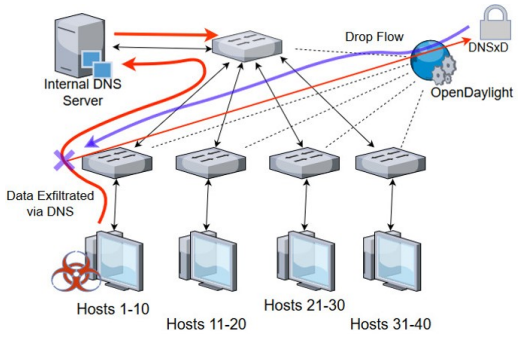


Fig. 3: Evaluation Topology

Each of the DNS data exfiltration tools was run locally to transmit a 50Kb file [4]. This generated a sample malicious DNS traffic set of 2040 packets. The benign DNS traffic set (7,074 packets) is composed of locally captured traffic, along with public datasets [21]. Legitimate DNS traffic is highly variable according to the network and each host's requirements. In order to validate the sample data set, the statistics from the sample set (e.g. Figure 2) were compared with statistics from a much larger private dataset collected by DNS OARC [22].The similarity confirmed that the benign sample provides an accurate representation of DNS traffic.

Detection: When tested against 4 open-source DNS data exfiltration tools; Iodine [16], DnsCat2 [17], DET [18], and DNSExfiltrator [19], the application was able to detect each attack without triggering FPs for benign DNS traffic. The effectiveness of each detection feature was measured individually along with the final DNSxD solution. The results are presented in Figure 5 with a comparison against Snort (v2.8.11.1), a RegEx based detection mechanism which intercepts traffic flows and manages them based on a predefined rule set. Figure 4 shows an example of the rules used to detect malicious DNS traffic. Each test was run 5 times and averaged. The attack detection rate varies based on any given identifier and the standard error increases with the throughput of the attack. This can be seen in the greater error margins for Iodine & DNSExfiltrator, the two highest throughput attacks. TA approaches, such as volume & frequency, depend heavily on well-defined thresholds (further described below). Naturally, the greater the traffic volume necessary to identify a potential attack, the longer it will take to be identified. In contrast, PI approaches, such as entropy & record type, can often detect the attack from the first malicious packet they analyse. A result of 50Kb means that the attack was successful and the full file was exfiltrated.

Mitigation Delay: In Figure 5, the number of bytes exfiltrated from the malicious file are displayed. This highlights the average delay between detecting and mitigating an attack. Depending on the data being exfiltrated this could be of little consequence. For example, if the attacker is trying to exfiltrate large quantities of corporate data. Furthermore, the polling from C&C and the initiation of protocols during tunnelling is often enough to trigger detection, meaning that no sensitive data can be exfiltrated. However, if the attacker is exfiltrating small but highly sensitive data, e.g. credit card details, through sophisticated pure exfiltration methods this could provide them with a large enough window to exfiltrate some meaningful data. An additional small delay is incurred during the process of installing the block flow rule on the relevant switches. The delay could be reduced for the volume and frequency identifiers with the penalty of an increase in FPs as benign traffic begins to fall within the threshold for a malicious attack.

Network Performance: The application has no impact on the intended traffic flow, as the traffic is mirrored (rather than redirected) to the controller allowing the traffic to flow as normal. This is one potential advantage over current IDS/IPS solutions, such as Snort, which act in-line with the traffic and potentially become a bottleneck for traffic flow to/from the network. However, the additional controller load may produce delays in the installation of flow rules, if the network was operating near peak capacity.

Thresholds: Detecting the attack based on TA, e.g. volume/frequency requires the definition of threshold values that sit beyond the limit of legitimate DNS traffic in the network. In the evaluation, the following thresholds produced the optimal results; query length $>$ 55 Bytes, volume $>$ 1500 Bytes, frequency $>$ 100 packets, Entropy score $>$ 4. Each threshold was applied over a 10 second window, specific to a domain. The closer the threshold is to legitimate traffic, the more reactive the application, limiting the amount of data that is exfiltrated prior to detection. However, this also raises concerns over generating FPs. DNS is a highly variable service, so to achieve zero FPs, only the highest throughput exfiltration attacks can be detected. Other PI identifiers must be relied on in order to detect low throughput attacks.

Accumulative Threat To increase the strength of the threshold detection mechanism, and reduce the mitigation time, combinations of suspicious levels were employed. The following ranges aided the detection without generating FPs; DNS query length $>=$ 40 Bytes, volume $>$ 1000 Bytes, frequency $>=$ 30 packets, entropy $>=$ 3.5.

[4]The features of these tools are shown in Table II.

```
reject udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"Large DNS query detected"; dsize:>110; #sid:101; rev:004;)
reject udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"DNS Frequency detected"; detection_filter:track by_src, count 50, seconds 30; sid:102; rev:004;)
```

Fig. 4: Malicious DNS Snort Rules

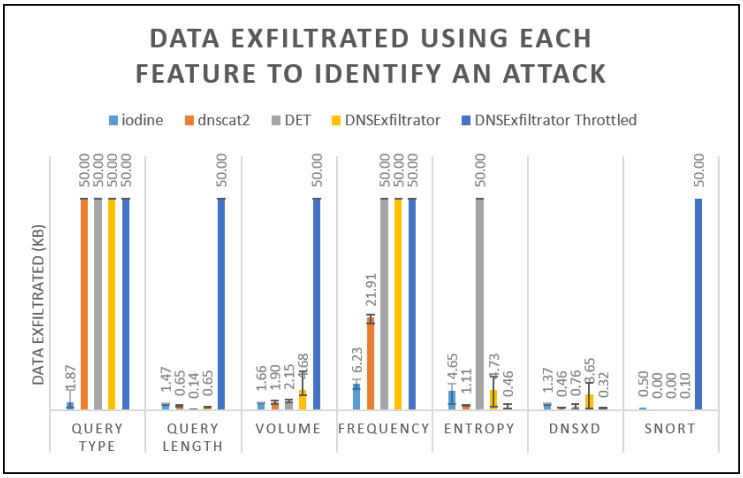


Fig. 5: Evaluation of DNSxD features, and DNSxD v Snort

A. Summary of Results

As shown in Figure 5, each detection feature had varying success depending on the attack tool. By combining each feature and accounting for suspicious feature levels, DNSxD, successfully detected each attack. Although Snort reacted to 4 out of 5 attacks faster, its inability to detect an obfuscated attack makes it a highly vulnerable solution. Furthermore, although obfuscated attacks may bypass detection mechanisms, the lower transfer rate allows mitigation measures to take effect before a substantial amount of data is lost. Higher throughput attacks however, have a greater potential to exfiltrate larger quantities of data before mitigation measures come into effect.

A number of potential improvements have been identified for this solution; reducing controller load, only blocking malicious domains once detected, and decreasing mitigation time. Protocol Independent Packet Processing (P4) has been identified as a potential solution to these 3 issues. P4 allows increased programmability within the network devices [23]. This would allow the most obvious attacks to be blocked directly at the switch, and allow any traffic which was definitely benign to bypass further analysis, i.e. trusted domains. As discussed in Section V DNSxD is unable to block specific domains. To allow the continuation of legitimate traffic, a blacklist of malicious domains could be created. Through P4 it would be possible to program a rule into the switches, which checks each DNS flow for these known malicious domains and acts accordingly. This would allow legitimate DNS traffic to continue and decrease the number of flow rules required to block malicious domains.

VII. Conclusion

The exfiltration of sensitive data remains a lucrative goal for many malicious actors. As such, data exfiltration is one of the greatest threats to corporate and private networks. In this paper, we presented a DNS data exfiltration analysis, detection, and mitigation application, DNSxD, leveraging the capabilities of the SDN architecture. DNSxD is designed based on feature selection resulting from analysis of common DNS data exfiltration tools and existing DNS data exfiltration detection mechanisms. The evaluation demonstrates the success of DNSxD in detecting exfiltration based on a range of identifiers, which in combination provide protection against even stealthy exfiltration attacks. Future work will build on DNSxD to provide an advanced data exfiltration mitigation platform.

References

[1] B. Filkins, "Sensitive Data at Risk: The SANS 2017 Data Protection Survey," *SANS Institute InfoSec Reading Room*, 2017, https://www.sans.org/reading-room/whitepapers/threats/sensitive-data-risk-2017-data-protection-survey-37950.

[2] Ponemon Institute, "Cost of Data Breach Study: Global Overview," *Research Report*, 2017, https://www.ibm.com/security/data-breach.

[3] O. Lystrup, "Majority of orgs do not monitor dns," *Cisco Security Report*, 2016, https://umbrella.cisco.com/blog/2016/01/21/cisco-security-report-more-orgs-should-be-monitoring-dns/.

[4] G. Farnham, "Detecting DNS Tunneling," *SANS Institute InfoSec Reading Room*, 2013, https://www.sans.org/reading-room/whitepapers/dns/detecting-dns-tunneling-34152.

[5] S. Scott-Hayward, S. Natarajan, and S. Sezer, "A survey of security in software defined networks," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 623–654, 2016.

[6] S. Jaworski, "Using splunk to detect dns tunneling," *SANS Institute InfoSec Reading Room*, 2016.

[7] J. Liu, S. Li, Y. Zhang, J. Xiao, P. Chang, and C. Peng, "Detecting dns tunnel through binary-classification based on behavior features," in *Trustcom/BigDataSE/ICESS, 2017 IEEE*. IEEE, 2017, pp. 339–346.

[8] M. T. Arashloo, Y. Koral, M. Greenberg, J. Rexford, and D. Walker, "Snap: Stateful network-wide abstractions for packet processing," in *Proceedings of the 2016 ACM SIGCOMM Conference*. ACM, 2016, pp. 29–43.

[9] Infoblox. Infoblox. [Online]. Available: https://www.infoblox.com/

[10] Talos. Snort IDS/IPS. Https://www.snort.org/.

[11] S. Bromberger, "DNS as a covert channel within protected networks," *National Electronic Sector Cyber Security Organization (NESCO)(Jan., 2011)*, 2011.

[12] A. M. Kara, H. Binsalleeh, M. Mannan, A. Youssef, and M. Debbabi, "Detection of malicious payload distribution channels in dns," in *Communications (ICC), 2014 IEEE International Conference on*. IEEE, 2014, pp. 853–858.

[13] A. Nadler, A. Aminov, and A. Shabtai, "Detection of Malicious and Low Throughput Data Exfiltration Over the DNS Protocol," *arXiv preprint arXiv:1709.08395*, 2017.

[14] O'Connor, Tj and Enck, William and Petullo, W Michael and Verma, Akash, "Pivotwall: Sdn-based information flow control," in *Proceedings of the Symposium on SDN Research*. ACM, 2018, p. 3.

[15] S. Sheridan and A. Keane, "Improving the stealthiness of dns-based covert communication," in *ECCWS 2017 16th European Conference on Cyber Warfare and Security*. Academic Conferences and publishing limited, 2017, p. 433.

[16] Yarrick. Iodine. [Online]. Available: https://github.com/yarrick/iodine

[17] iagox86. dnscat2. [Online]. Available: https://github.com/iagox86/dnscat2

[18] Sensepost. DET. [Online]. Available: https://github.com/sensepost/DET

[19] Arno0x. Dnsexfiltrator. [Online]. Available: https://github.com/Arno0x/DNSExfiltrator

[20] OpenDaylight. OpenDaylight Nitrogen SDN Controller. [Online]. Available: https://www.opendaylight.org/what-we-do/current-release/carbon

[21] Public DNS Packet Captures, Packet Total. https://packettotal.com/app/search?q=_type:%22dns%22#.

[22] DNS Operation Analysis and Research Center. Https://www.dns-oarc.net/.

[23] P. Bosshart, D. Daly, G. Gibb, M. Izzard, N. McKeown, J. Rexford, C. Schlesinger, D. Talayco, A. Vahdat, G. Varghese *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 87–95, 2014.