Real-Time Detection of DNS Exfiltration and Tunneling from Enterprise Networks

Jawad Ahmed*,†, Hassan Habibi Gharakheili*, Qasim Raza*, Craig Russell†, and Vijay Sivaraman*
*Electrical Engineering and Telecommunications, University of New South Wales, Sydney, Australia.
†CSIRO Data61, Sydney, Australia.
Emails: {j.ahmed, h.habibi}@unsw.edu.au, q.raza@student.unsw.edu.au, craig.russell@data61.csiro.au, vijay@unsw.edu.au

***Abstract*—Enterprise networks constantly face the threat of valuable and sensitive data being stolen by cyber-attackers. Sophisticated attackers are increasingly exploiting the Domain Name System (DNS) channel for exfiltrating data as well as maintaining tunneled command and control communications for malware. This is because DNS traffic is usually allowed to pass through enterprise firewalls without deep inspection or state maintenance, thereby providing a covert channel for attackers to encode low volumes of data without fear of detection.**

This paper develops and evaluates a real-time mechanism for detecting exfiltration and tunneling of data over DNS. Unlike prior solutions that operate off-line or in the network core, ours works in real-time at the enterprise edge. Our first contribution is to develop, tune, and train a machine learning algorithm to detect anomalies in DNS queries using a benign dataset of top rank primary domains from two enterprise networks. Our second contribution is to implement our scheme on live 10 Gbps traffic streams from the network borders of the two organizations, inject more than a million malicious DNS queries generated via an exfiltration tool, and show that our solution is able to identify them with high accuracy. Our tools and datasets are made available to the public for validation and further research.

I. Introduction

The Domain Name System (DNS) is used for converting domain names (*e.g.,* `google.com`) into IP addresses and as such constitutes a mission-critical service. However, DNS communication is relatively poorly policed by organizations (compared to services like email, FTP, and HTTP) and has been exploited by cyber-criminals to maintain covert communication channels with compromised hosts. The resulting damages can be huge, amounting to several million dollars in a single attack [1]. Several high-profile data exfiltration breaches have been reported recently: the Sally Beauty breach (a theft of 25K credit cards) [2] and FrameworkPOS malware (a theft of 56M credit cards from Home Depot) [3] in 2014, BernhardPOS malware [4] in 2015, MULTIGRAIN malware [5] in 2016, Win32.Backdoor.Denis [6] in 2017, and UDPoS Malware [7] in 2018. In addition, there have been a number of DNS tunneling incidents in which malware actors used their DNS servers to send and receive the command and control commands to and from compromised hosts; examples include Feederbot [8] and botmaster [9], Morto worm [10], and Wekby pisloader [11].

978-3-903176-15-7 © 2019 IFIP

One way for the attacker to exploit DNS is to register a domain (*e.g.,* `foo.com`) so that the attacker's malware in a host victim can then encode valuable private information (such as credit card numbers, login passwords or intellectual property) into a DNS request of the form `arbitrary-string.foo.com`. This DNS request gets forwarded by resolvers in the global domain name system to the authoritative server for the `foo.com` domain (under the attacker's control), which in turn sends a response to the host victim. This provides the attacker with a low-rate but covert two-way communication channel between a host victim and their command-and-control center.

Interestingly, enterprise firewalls are typically configured to allow all packets on UDP port 53 (used by DNS) since DNS is such a crucial service for virtually all applications. Some firewalls do offer enhanced DNS protection but these require deep packet inspection of DNS messages to identify the covert channel and then isolate domains that contain encoded data. The significant resources required for this capability [12], and the resulting impact on firewall forwarding performance, usually results in enterprise network operators disabling such features. This ability to transit firewalls gives attackers a covert channel, albeit a low-rate one, by which to exfiltrate private data and to maintain communication with malware by tunneling other protocols (*e.g.,* SSH, FTP) to command-and-control centers. As one example, the remote access trojan DNSMessenger [13] discovered in 2017 used DNS queries and responses to execute malicious powerShell commands on compromised hosts.

In this paper we develop and validate a mechanism for real-time detection of DNS exfiltration and tunneling in two operational networks – a large University and a mid-sized Government Research Institute. Our **first** contribution is to develop, tune, and train a machine learning algorithm to detect anomalous DNS queries using a known dataset of benign domains as ground truth. For our **second** contribution we implement our scheme on live 10 Gbps traffic streams from borders of the two organizations, inject more than a million malicious DNS queries generated using an exfiltration tool and show that our scheme is able to identify such malicious activity with high accuracy. We make our tools and datasets available to the public to facilitate further research into this area.

II. Related Work

DNS traffic has been analyzed to identify malicious network activities [14]. Work in [15] proposed a method to find the maximum information that can be encoded in a sub-domain portion of a DNS query name to detect whether the query contains encoded data or not. Authors used an information theoretic approach, namely the use of Kolmogorov complexity. The authors established an upper bound on the volume of surreptitious communication by investigating inter-query time and query record type. In [16] Das et al. employed supervised learning based models with logistic regression to classify queries into normal and exfiltration. We believe that classification (signature-based) approach is not sufficient for addressing the new and growing security issues, and specifically obtaining "ground truth" on malicious queries in order to train the classifier is difficult. Similar to our approach, Asaf et al. [17] proposed an anomaly-based solution to detect low throughput data exfiltration over DNS. The authors maintain states of several attributes for each primary domain over last n hours (*e.g.,* rate of A and AAAA records, average length of query name). In [18]–[20], authors have proposed DNS tunnel detection using character frequency analysis. However, the detection criteria is based on the threshold value and attackers may trick these systems easily.

Our focus is on attributes of fully qualified domain names that can be extracted in "real-time", without a need for states (*i.e.,* "stateless") – we assume that DNS traffic in not encrypted over TLS. In our scheme, we look for anomalies of query names indicative of deviation from normal behavior, since anomaly detection holds promise as a way of detecting new and unknown threats pattern. Our scheme can be extended by collecting states only for those hosts that generate anomalous queries, and ultimately mitigate malicious DNS tunneling/exfiltration – such mitigation is beyond the scope of this paper.

III. Detection of Anomalous DNS Queries

In this section we first briefly look at characteristics of DNS query names, and then develop a machine learning technique to determine if a DNS query of an enterprise host is normal or not (*i.e.,* "anomaly detection"). Our objective is to achieve a real-time detection of anomalous query names with accuracy comparable to or better than techniques that require temporal states related to characteristics of queried domains or DNS activity of hosts.

A. DNS Queries and Attributes

We have collected DNS traffic from the border of two enterprise networks, a medium-size research institute and a large University campus. In both instances, the IT department of the enterprise provisioned a full mirror (both inbound and outbound) of their Internet traffic (each on a 10 Gbps interface) to our data collection system from their border routers (outside of the firewall), and we obtained appropriate ethics clearances for this study (UNSW Human Research Ethics Advisory Panel approval number HC17499, and CSIRO Data61 Ethics approval number 115/17). We extracted DNS packets from each of the enterprise Internet traffic streams in real-time by configuring rules to match incoming/outgoing IPv4 and IPv6 UDP packets on port 53 in an OpenFlow switch. The study here considers data collected over a one-week period from 30-Jul-2018 to 5-Aug-2018.

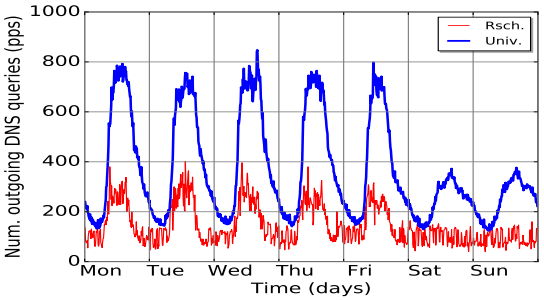

Fig. 1. Real-time number of queries.

Considering the load of DNS queries generated by enterprise hosts, shown in Fig. 1, we see that the number of packets-per-second in the research network varies between 50 to 400 depending on the day of the week and peak/off-peak hours. For the University network, on the other hand, a larger variation is observed – *i.e.,* 150 to more than 800 pps.

We now identify attributes for the query name (i.e. FQDN: Fully Qualified Domain Name) generated by enterprise hosts that are relevant to differentiating benign and malicious DNS queries. Our aim is to use only "Stateless" attributes which can be derived from individual DNS query packets, independent of time-series characteristics of queried domains or hosts DNS activity – there is no overhead in computing these attributes in real-time. We define our attributes by three main categories namely characters count, entropy (an indication of randomness) of string, and length of discrete labels in the query name.

Total count of characters in FQDN is an important attribute since more characters imply that the query name probably carries embedded information for an outside host. Since the exfiltrated (or Command & Control) message is carried by the sub-domain portion of an FQDN, we use the **count of characters in sub-domain** as our second attribute. Additionally, we use the **count of uppercase characters** and **count of numerical characters** in a query name to determine if it is benign or malicious. This is because the fraction of uppercase and numerical characters becomes high in encrypted/ encoded data [16] – however, not all encrypted data is malicious. Random ("not-readable") sub-domains are common in DNS exfiltration/tunneling queries due to use of encryption and/or encoding [17]. **Entropy** is a measure to determine the degree of non-readability (or strength of encryption) and uncertainty in a string. We use the **number of labels** as our sixth attribute. This is because DNS exfiltration/tunneling traffic tend to use certain patterns of labels in their query names. For example, in the query name `www.scholar.google.com`, there are four labels separated by dots. Also, **maximum label length** and **average label length** in the FQDN are the last two attributes for the machine learning model.

B. Machine Training

We train our anomaly detection machine with benign data from four days of our dataset – we keep the remaining three days worth of data for testing. Ground truth of benign domains in the literature is largely drawn from highly ranked popular domains [21]. For example, Alexa top-ranked domains are commonly used. Since Alexa no longer publishes free top one million sites, we use its alternative, Majestic Million [22] that releases a free dataset of top 1M domains and updates it on a daily basis. Majestic ranks sites by the number of subnets linking to that site. However, Alexa ranking is based on the browsing behavior of Internet users (*i.e.,* estimate of global traffic to a domain). We note that some malicious domains may appear among top K Alexa domains due to a burst of requests from a high number of infected clients querying them. For the benign training instances, we only use top 10,000 primary domains. We also include FQDNs for "`sophosxl.net`" domain (related to a benign anti-virus application) which is not among the top 10K Majestic dataset.

C. Algorithms and Tuning Parameters

The objective is to maximize detection of anomalous queries while reducing the rate of false alarms (*i.e.,* incorrectly detecting a normal query as anomalous, or vice versa). Many of supervised machine-learning algorithms for detecting anomalies such as one-class SVM and Replicator Neural Network suffer from high false alarms since they are optimized for profiling the inlier behavior rather than detecting anomalies. We employ "Isolation Forest (*iForest*)" [23] which is an effective algorithm in detecting anomalous instances in high-dimensional datasets with minimal memory and time complexities.

The iForest algorithm "isolates" observations by randomly selecting an attribute and then randomly selecting a split value in the range of values (*i.e.,* between min and max) for the selected attribute. Since recursive partitioning can be represented by a tree structure, the number of splittings required to isolate an instance is equivalent to the path length from the root node to the terminating node. This path length, averaged over a forest of such random trees, is a measure of normality and the decision function (*i.e.,* normal instances are expected to have a fairly large path length in random partitioning). Therefore, when a forest of random trees collectively produces shorter path lengths for a particular instance, it is highly likely to be an anomaly.

Algorithm Tuning: We used `scikit-learn` and its APIs, an open-source machine-learning package written in Python, to train and test our machine. We have used three tuning parameters for iForest during training phase namely number of trees (*n_estimators*), height limit of trees (*max_samples*), and contamination rate. We tune the value of each parameter while fixing the other two parameters and validate the accuracy of our machine for both benign and malicious instances (that we have the ground truth) in both organizations. The default value for the number of trees is 100, the height limit of trees is set to "auto" (implying 8 given the size of our dataset), and contamination rate is 10%.

TABLE I
DETECTION ACCURACY OF GROUND-TRUTH INSTANCES AFTER TUNING.

	Benign	**Malicious**
Research Institute	98.44%	95.07%
University Campus	97.99%	98.49%

TABLE II
PERFORMANCE OF OUR MACHINE FOR TRUSTED DOMAINS.

	Research institute				**University campus**			
primary domain	**normal**	**anomalous**	**Avg. query length**	**false-rate (%)**	**normal**	**anomalous**	**Avg. query length**	**false-rate (%)**
`akadns.net`	2.6M	24K	38	0.91	7.6M	191K	38	2.4
`googleapis.com`	165K	1.6K	76	0.96	526K	15K	76	2.7
`gstatic.com`	207K	362	69	0.17	835K	986	76	0.11
`in-addr.arpa`	3.7M	49K	26	1.32	9.2M	1.1M	26	10.7
`mcafee.com`	1.9M	735	84	0.03	635K	13K	88	2.01
`onmicrosoft.com`	22K	1.6K	51	6.55	201K	1537	53	0.75
`senderbase.org`	1.1M	14K	66	1.32	2.2M	2816	66	0.12
`sophosxl.net`	138K	6.5K	103	4.44	2.5M	394K	119	13.7
`spamhaus.org`	12K	597	31	4.7	947K	7.7K	32	0.81
`spotify.com`	579	31	45	5.08	468K	1.2K	168	0.25
Top 100 domains (*e.g.,* `google`, `apple`)	7.9M	135K	20	1.68	24M	351K	20	1.41

For ground-truth malicious instances, we have generated DNS exfiltration queries by our open source tool, forked from an open source project called "DNS Exfiltration Toolkit" (DET) [24]. We ran our tool on a machine inside the University network that exfiltrates the content of a CSV file containing 1000 samples of random credit card details (obtained from [25]) to an authoritative name server under our control located in the Research network. DET employs AES-256 encryption and uses two tuning parameters namely max length of query name (*i.e.,* 50 to 218 characters) and max length of labels (*i.e.,* 30 to 63 characters) to diversify our synthetic malicious queries. We generated a total of 1.4M exfiltration queries which are publicly available at [26] in form of a CSV file.

We found that setting the number of trees equal to 2 results in a high accuracy of more than 91% for benign and 63% for malicious instances – increasing this parameter does not enhance the accuracy but increases the model size and prediction time. Having fixed the number of trees to 2 and the contamination rate to 10%, we varied the height of trees from 1 to 20. The detection performance rises by increasing the height limit of trees and gets stabilized at the value of 18 with the best accuracy of more than 90% and 98% for ground-truth benign and malicious instances respectively. We then fixed the number of trees to 2 and height limit of isolation trees to 18 to quantify the impact of contamination rate. Decreasing the contamination rate from 10% to 2% improved the performance of our model for both organizations as shown in Table I, with the accuracy of more than 97% for benign instances and more than 95% for malicious instances.

To summarize, we found the optimal value of tuning parameters equal to 2, 18, and 2% respectively for the number of trees, the height limit of trees, and the contamination rate. For optimal tuning parameters the iForest algorithm sets the threshold value of anomaly score to 0.54, distinguishing normal and anomalous instances.

Table II shows the performance of our machine (after tunning) for selected benign instances – for cross validation. It can be seen that the rate of false alarms is mostly less than 5% in both organizations, though we see a higher false rate (*i.e.,* more than 10%) for `in-addr.arpa` and `sophosxl.net` domains in the University network. We, therefore, pre-filter instances for

TABLE III
ANOMALY DETECTION FOR RESEARCH INSTITUTE.

Input	Output	Days 1-4	Days 5-7
Benign domains	normal	98.44%	98.30%
	anomalous	1.56%	1.70%
Others	normal	78.43%	77.55%
	anomalous	21.57%	22.45%

TABLE IV
ANOMALY DETECTION FOR UNIVERSITY CAMPUS.

Input	Output	Days 1-4	Days 5-7
Benign domains	normal	97.99%	97.99%
	anomalous	2.01%	2.01%
Others	normal	70.57%	70.59%
	anomalous	29.43%	29.41%

domains that are highly trusted (*i.e.,* certainly benign) without passing them to the anomaly detection machine.

IV. PERFORMANCE EVALUATION

In this section, we evaluate the efficacy of our scheme by: (a) cross-validating and testing the accuracy of the trained model for benign instances, (b) testing the detection rate for malicious DNS queries that we generate using our tool, and (c) quantifying the performance in real-time on live 10 Gbps traffic streams from the two organizations.

Accuracy: As mentioned in the previous section, we trained our model with benign instances from 4 days' worth of our data (*i.e.,* Days 1-4), and tested with all instances from Days 5-7 in addition to remaining instances from Days 1-4 that were not used for training (*i.e.,* "Others"). Tables III and IV show the rate of detection (*i.e.,* normal versus anomalous) for the benign and Others instances in the two networks. It can be seen that 98% of benign instances are correctly detected as normal during both cross-validation (*i.e.,* Days 1-4) and testing (*i.e.,* Days 5-7) phases. We note that our machine raises a false alarm for about 2% of benign domains, as highlighted in red texts. To address this, we populate a whitelist of domains that are highly trusted. Our whitelist comprises top 100 domains from the Majestic ranking dataset (*e.g.,* `google.com`, `bbc.com`, `amazonaws.com`) as well as a number of popular legitimate (*e.g.,* `akadns.net`, `in-addr.arpa`, `spotify.com`) and security services (*e.g.,* `spamhaus.org`, `senderbase.org`). Note that these security services are using disposable domains (*i.e.,* "single-time use") for the purpose of signaling over DNS queries (*e.g.,* `0.0.0.0.1.0.0.4e.135jg5e1pd7s4735ftrqweufm5.avqs.mcafee.com` [27]). Additionally, the average anomaly score for instances classified as normal and anomalous is 0.44 and 0.59 respectively in both organizations.

Real-Time Performance: In terms of real-time performance, we have quantified the average time for extracting eight attributes and anomaly detection (via running prediction against the trained model) by testing more than 300 million DNS queries in our dataset from the two enterprise networks – our attributes extraction and anomaly detection engines run on a virtual machine with 4 cores of CPU, 6GB of memory, and storage of 50GB. As shown in Table V, on average it takes 800 μsec to determine if a DNS query is normal or not. This mean that our scheme can process about 1250 DNS queries per second which is well above the actual rate of DNS queries in both organizations, as shown in Fig. 1.

TABLE V
AVG. TIME COMPLEXITY OF OUR SCHEME.

extracting attributes	54 μsec
detecting anomalies	746 μsec
Total time per each query name	800 μsec

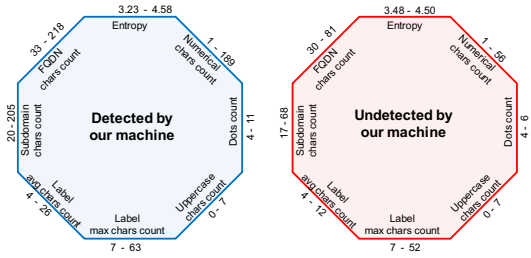

Fig. 2. Attributes of DNS exfiltration query names: detected vs. undetected

Known DNS Exfiltration: Lastly, we evaluate the efficacy of our scheme with known DNS exfiltration queries. As explained in previous section in Table I, our machines for the Research Institute and the University Campus respectively were able to correctly detect 95.07% and 98.49% of exfiltration queries (generated by our DET tool) as anomalous instances.

In Fig. 2, we show the value of attributes for detected instances (on the left) versus undetected instances (on the right) using the machine for the Research Institute. Even though undetected instances were shorter both in total length and average label length, it is important to note that there is a fair overlap of value range comparing detected and undetected instances across all attributes – suggesting that the collection of attributes would determine the output of our machine. Additionally, our machine was able to detect 10 samples of DNS queries from known real malware reported on various forums [4], [13], [28].

V. CONCLUSION

Enterprise networks are potential targets of cyber-attackers for stealing valuable and sensitive data over DNS channels. We have developed and validated a mechanism for real-time detection of DNS exfiltration and tunneling from enterprise networks. We have developed, tuned, and trained a machine learning algorithm to detect anomalies in DNS queries using a known dataset of benign domains as ground truth. We have then evaluated the efficacy of our scheme on live 10 Gbps traffic streams from the borders of the two organizations networks by injecting more than a million malicious DNS queries via an exfiltration tool. We have also made our tools and dataset publicly available.

VI. ACKNOWLEDGMENTS

This work was completed in collaboration with the Australian Defence Science and Technology Group.

REFERENCES

[1] Efficient iP, "The Global DNS Threat Survey," Tech. Rep., 2017.

[2] B. Krebs. Deconstructing the 2014 Sally Beauty Breach. [Online]. Available: https://krebsonsecurity.com/2015/05/deconstructing-the-2014-sally-beauty-breach/

[3] P. Rascagneres. New FrameworkPOS variant exfiltrates data via DNS requests. [Online]. Available: https://bit.ly/2VpfVd5

[4] B. Allen. (2015) BernhardPOS - New POS Malware Discovered By Booz Allen. [Online]. Available: https://bit.ly/2R1Az4B

[5] C. Lynch, D. Andonov, and C. Teodorescu. (2016) MULTIGRAIN – Point of Sale Attackers Make an Unhealthy Addition to the Pantry. [Online]. Available: https://bit.ly/1pgvJxn

[6] A. Shulmin and S. Yunakovsky. (2017) Use of DNS Tunneling for C&C Communications. https://bit.ly/2SwS1KY.

[7] R. Neumann and L. Somerville. (2018) UDPoS – exfiltrating credit card data via DNS. https://bit.ly/2GSEfAP.

[8] R. J. Dietrich. Feederbot Botnet Using DNS as Carrier for Command and Control (C2). [Online]. Available: https://chrisdietri.ch/post/feederbot-botnet-using-dns-command-and-control/

[9] C. J. Dietrich, C. Rossow, F. C. Freiling, H. Bos, M. Van Steen, and N. Pohlmann, "On Botnets that use DNS for Command and Control," in *Proc. Computer Network Defense*, Gothenburg, Sweden, Sept. 2011.

[10] C. Mullaney. Morto worm sets a (DNS) record. [Online]. Available: https://www.symantec.com/connect/blogs/morto-worm-sets-dns-record

[11] T. Spring. Wekby apt gang using DNS tunneling for command and control. [Online]. Available: https://threatpost.com/wekby-apt-gang-using-dns-tunneling-for-command-and-control/118303/

[12] S. Kathuria. (2015) DNS Firewall is not a Next Generation Firewall. [Online]. Available: https://bit.ly/1NbDIRp

[13] E. Brumaghin and C. Grady. (2017) Covert Channels and Poor Decisions: The Tale of DNSMessenger. [Online]. Available: https://bit.ly/2R6OSEM

[14] M. Lyu, H. Habibi Gharakheili, C. Russell, and V. Sivaraman, "Mapping an Enterprise Network by Analyzing DNS Traffic," in *Proc. Passive and Active Measurement (PAM)*, Puerto Varas, Chile, Mar 2019.

[27] Y. Chen, M. Antonakakis, R. Perdisci, Y. Nadji, D. Dagon, and W. Lee, "DNS noise: Measuring the pervasiveness of disposable domains in modern DNS traffic," in *Proc. Dependable Systems and Networks (DSN)*, Atlanta, GA, USA, June 2014.

[15] V. Paxson, M. Christodorescu, M. Javed, J. R. Rao, R. Sailer, D. L. Schales, M. P. Stoecklin, K. Thomas, W. Venema, and N. Weaver, "Practical Comprehensive Bounds on Surreptitious Communication over DNS," in *Proc. USENIX Security*, Washington, DC, USA, Aug. 2013.

[16] A. Das, M.-Y. Shen, M. Shashanka, and J. Wang, "Detection of exfiltration and tunneling over dns," in *Proc. Machine Learning and Applications*, Cancun, Mexico, Dec. 2017.

[17] A. Nadler, A. Aminov, and A. Shabtai, "Detection of Malicious and Low Throughput Data Exfiltration Over the DNS Protocol," *CoRR*, 2017. [Online]. Available: http://arxiv.org/abs/1709.08395

[18] K. Born and D. Gustafson, "NgViz: Detecting DNS Tunnels Through N-gram Visualization and Quantitative Analysis," in *Proc. ACM CSIIRW*, Oak Ridge, Tennessee, USA, April 2010.

[19] K. Born and D. Gustafson, "Detecting DNS Tunnels using Character Frequency Analysis," in *Proc. Annual Security Conference*, Las Vegas, USA, Apr 2010.

[20] C. Qi, X. Chen, C. Xu, J. Shi, and P. Liu, "A Bigram based Real Time DNS Tunnel Detection Approach," *Proc. Procedia Computer Science*, 2013.

[21] Y. Zhauniarovich, I. Khalil, T. Yu, and M. Dacier, "A Survey on Malicious Domains Detection Through DNS Data Analysis," *ACM Comput. Surv.*, vol. 51, no. 4, July 2018.

[22] T. M. Million. (2018) Top 1 million website in the world. [Online]. Available: http://downloads.majesticseo.com/majestic_million.csv

[23] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in *Proc. Data Mining*, 2008.

[24] R. Qasim. (2018) DET (extensible) Data Exfiltration Toolkit. [Online]. Available: https://github.com/qasimraz/DET

[25] (2018) MasterCard Credit Card Generator. [Online]. Available: https://www.getcreditcardinfo.com/masterbulkgenerator.php

[26] (2018) DNS EXfiltration Dataset. [Online]. Available: https://nozzle-data.sdn.unsw.edu.au/dns-exfiltration-dataset/

[28] L. Mendieta. (2016) Three Month FrameworkPOS Malware Campaign Nabs 43,000 Credit Cards from Point of Sale Systems. [Online]. Available: https://bit.ly/2BSw166