Malicious DNS Tunneling Detection in Real-Traffic DNS Data

Danielle Lambion*, Michael Josten*, Femi Olumofin†, Martine De Cock*
* *School of Engineering and Technology, University of Washington, Tacoma, USA*
Email: {dlambion, mjosten, mdecock}@uw.edu
† *Infoblox, Santa Clara, USA, Email: folumofin@infoblox.com*

***Abstract*—While originally not intended for data transfer, the Domain Name System (DNS) is currently used to this end anyway, in a process called DNS tunneling (DNST). Malicious users exploit DNST for data exfiltration from infected machines, posing a critical security threat. We train and evaluate state-of-the-art convolutional neural network, random forest, and ensemble classifiers to detect tunneling in DNS traffic. Finally, we assess the classifiers' performance and robustness by exposing them to one day of real-traffic data.**

***Keywords*-DNS tunneling, random forest, CNN**

I. Introduction

DNS servers resolve domain names to IP addresses to route connections on the internet. Since the DNS protocol is not meant for data transfer, it is often overlooked by security administrators when hunting for illegitimate data transfers. Malicious users exploit such knowledge to extract confidential data from compromised computers by encoding the data to be tunneled within a DNS query name, while avoiding firewalls and other network security monitoring mechanisms. So-called DNS tunneling (DNST) is used for benign purposes as well, such as a computer sending a document to a wireless printer or an antivirus software sending updates to a computer. This mixed use of DNST for legitimate and illegitimate purposes makes detecting malicious DNST in real traffic very challenging.

We developed state-of-the-art machine learning (ML) classifiers for the detection of DNST attacks, using a combination of random forest (RF) and convolutional neural network (CNN) models. The CNN is trained on the query names themselves as input, while for the RF we leverage domain expertise from the literature on informative features to extract from DNS queries, and the grouping of DNS queries in batches for more accurate classification [1], [2]. While previous work has mostly been applied to synthetic data that is generated with known tunneling software, we train and evaluate our classifiers on real traffic data.

II. Methods

Dataset. We created a dataset from a real-time stream of passive DNS data, collected from subscribers including Internet Service Providers, schools, and businesses. For each instance, the dataset contains the query name, query type, IP address, and query time and date. A fictitious example of a DNS query name is `ata000g10289.badsite.com`, where `.com` is the top-level domain (TLD), `badsite.com` is the second-level domain (SLD), and `ata000g10289` is the prefix. In DNST, the exfiltrated information (payload) is contained in such prefixes.

`257b700cx.360tls.com`	`257b700c`
`261e1ccax.360tls.com`	`261e1cca`
`26580a1ax.360tls.com`	`26580a1a`

Figure 1. Query names before (left) and after (right) removal of the common suffix `x.360tls.com`.

Each instance in the dataset has one of three labels, shown in the left column of Table I. "Non-tunneling traffic" refers to instances flagged as DNST traffic by a deployed DNST detector, and deemed to be false positives by security analysts. *Our goal is to train a classifier that can distinguish malicious traffic, i.e. (3), from benign traffic, i.e. (1)-(2).* Since only a limited amount of data can be tunneled within each query name, data exfiltration is typically done over multiple queries combined. To detect DNST behavior more reliably, we therefore grouped queries with the same SLD, day, and IP address. Similar grouping methods have been proposed before [1], [2], [3]. In addition to grouping, we removed instances not considered to be beneficial for training and testing a DNST detector, as explained in the curation steps below:

1) From the raw dataset, we removed all query names that have `.arpa` as TLD. This TLD is exclusively used for internet infrastructure and can never be a tunnel.
2) We grouped query names by the defined grouping criteria: SLD, day, and IP address.
3) We removed any groups of size one. These are assumed to be non-tunnels as a single query is insufficient to tunnel information.
4) We retrieved the prefix portion by removing the common suffix from the query names within a group. For DNST traffic, this portion would be the payload. The suffix for some groups surpasses the SLD, as illustrated in Fig. 1. In the remainder of this paper, by "prefix" we mean the string that is left after removal of the common suffix.
5) We removed query names with an empty prefix after common suffix removal. These are assumed to be non-tunnels as information cannot be passed without a prefix. Any groups with only 0 or 1 query name left are dropped.

M. De Cock is a Guest Professor at Ghent University

978-1-7281-6251-5/20/$31.00 ©2020 IEEE

Table I
TRAIN AND TEST DATASET STATISTICS

		Train		Test	
		# instances	**# groups**	**# instances**	**# groups**
negative	(1) Normal Resolved	20,779	3,363	21,073	8,455
	(2) Non-Tunnel	484,472	3,363	471,001	4,524
positive	(3) DNS Tunnel (DNST)	58,743	6,726	2,820	587

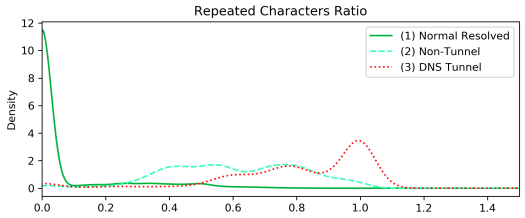

Figure 2. Feature analysis for feature 6 in Table II.

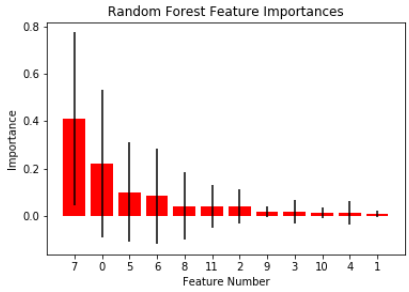

Figure 3. Feature importance ranking for the RF+CNN model trained with the training data in Table I using the features in Table II.

The curated dataset contains 7,313 groups of DNST traffic. We randomly selected 90% of the SLDs occurring in this DNS Tunnel traffic and assigned the corresponding 6,726 groups to the train data, while reserving the remaining 587 groups as test data. These are examples of the *positive* class for our binary classifiers. For the *negative* examples in the training data, we selected 3,363 groups of categories (1) and (2) each, to balance out the 6,726 groups of category (3). In initial experiments, we found such a balanced train set to yield better results. As evident from the last column in Table I, the distribution of the categories in the test data is not balanced at all, to more closely reflect a real traffic situation. As described in Sec. III, after training and testing our classifiers on the data from Table I, we applied them to one day of real-traffic data to assess their usefulness for practical deployment in a DNST detection system.

Convolutional Neural Network (CNN) Classifier. We trained a CNN to classify *individual* DNS query names, and subsequently use majority voting to label *groups* of DNS queries. We used the multichannel CNN architecture proposed by Saxe and Berlin [6] that has been applied successfully before for the related problem of detection of domain generation algorithms [7]. In this architecture, an embedding layer is followed by four parallel convolutional layers with incrementing kernel sizes of 2, 3, 4, and 5, each with 256 filters. Each convolutional layer learns to detect the soft presence of soft character n-grams (with $n = 2, 3, 4, 5$). This is reminiscent of the use of bigrams to detect DNS tunneling domains by Qi et al. [8]. The outputs of the parallel convolutional layers are fed into two hidden layers, each with 1024 nodes. The results of the hidden layers reach a single node output layer with a sigmoid activation function. Similar to [9], we use integer encoding with padding on the prefix as input to the CNN model.

Random Forest (RF) Classifier. In addition to the CNN, we trained a RF using 100 trees, and entropy as the criterion to select features for the tree nodes. Unlike the CNN, the RF is trained directly to classify entire groups of DNS queries. To create a feature set, we studied the existing literature on ML approaches to domain name classification and DNS threat detection [1], [2], [3], [4]. After extensive feature analysis on the data from Table I, for the RF we retained the 12 features in Table II. In the right column in Table II we cite references where similar features were used. As illustrated in Fig. 2, many of the features allow to distinguish normal resolved instances quite well from the two other categories in Table I, while non-tunnel and DNS tunnel instances are much closer in terms of feature values.

Many of the features in the literature are defined for individual query names of prefixes; to convert them into group features, we aggregate the feature values of the prefixes in a group by taking the average over the group (cfr. feature 0–8 in Table II). Feature 9 differs from feature 8 in that for feature 9, we concatenate all the prefixes from the group into one string and compute entropy over that entire string. The inclusion of feature 7, namely the CNN probability, enables a straightforward mechanism to leverage both the CNN and RF approaches simultaneously. In Sec. III, we use the notation "RF+CNN" to refer to the RF model trained on all 12 features from Table II, and "RF" for the corresponding model trained on all features except feature 7. Fig. 3 shows a ranking of the features according to their importance in the RF+CNN model. As we might expect, the probabilistic likelihoods from the CNN classifier is the most important feature. Features 0, 5, and 6 follow in importance ranking order. These are likely due to tunneling behaviors having longer prefixes to contain an attacker's payload, with a character distribution that deviates from natural language.

Table II
GROUP FEATURES USED IN RANDOM FOREST TRAINING AND CLASSIFICATION

No.	Feature	Description	Reference
0	Avg. Length	The average character length of the prefixes in the group	[1], [2], [4]
1	Avg. Unique Characters Count	The average count of unique characters in the prefixes in the group	[4]
2	Avg. Label Count	The average count of labels in the query names in the group	[1], [3], [5]
3	Avg. Gini Index	The average Gini value of the characters in the prefixes in the group	[1], [4]
4	Avg. Classification Error	The average classification error of characters in the prefixes in the group	[1], [4]
5	Avg. Digit Count	The average count of digits in the prefixes in the group	[4]
6	Avg. Ratio of Repeated Characters	The average ratio of characters repeated in the prefix to the number of unique characters in the prefixes in the group	[4]
7	Avg. CNN Probability	The average probability likelihood of the prefixes in the group being a DNS Tunnel according to the CNN classifier	our work
8	Avg. Entropy	The average normalized entropy of characters in the prefixes in the group	[1], [4], [5]
9	Concatenated Entropy	The normalized entropy of characters in the concatenated prefixes of the group	[1], [4], [5]
10	Unique Prefix Count	The count of unique prefixes within the group	[2]
11	Group Size	The count of prefixes within the group	our work

Table III
RESULTS ON TEST DATA

Model	Acc	AUC	**AUC@1%FPR**	**TPR@1%FPR**
RF	96.04%	99.84%	**91.90%**	**100%**
CNN	99.30%	99.41%	**83.62%**	**97.45%**
RF+CNN	96.65%	99.84%	**92.34%**	**99.49%**

Acc=Accuracy, AUC=Area Under Curve, FPR=False Positive Rate, TPR=True Positive Rate

III. RESULTS

A clear goal of the classifiers is to accurately detect DNS tunnels from network traffic while also maintaining a sufficiently small false positive rate (FPR). In network security scenarios, a detection of a DNS tunnel will likely result in an expensive network investigation, rendering false positives very costly. Results of all classifiers are presented in Table III; the results are for the classification of the groups of DNS traffic in the test data from Table I. As the last column indicates, all classifiers are able to detect (almost) all DNST traffic at a small FPR of 1%. The difference in predictive performance between the CNN model on one hand, and the RF and RF-CNN models on the other hand, may be due to the latter models' ability to leverage meaningful group features, while the CNN approach relies on a simple majority voting aggregation strategy to label each group based on the labels assigned to the prefixes within the group.

To assess the utility of the classifiers outside of the development setting leading up to the results in Table III, we applied them on an entire day of network traffic, consisting of ≈ 50 million queries. On this data, we applied the same grouping methodology as described in Sec. II, omitting "day" as a grouping criterion since all DNS queries originated in the same day. The one day of real-traffic data contains DNS query type NULL records, which are commonly used for DNS tunneling activity using the Iodine DNS tunneling tool [2]. We leveraged this knowledge to tune the classification threshold of the RF at 0.8. 35,088 (10.94%) groups, accounting for 16,664,953 (34.32%) queries, passed this threshold and were flagged as suspected DNST tunnels.

While only 30% (2,754 out of 9,039) SLDs in the flagged traffic are known whitelist candidates, they account for an overwhelming 92% of the total number of queries, leaving the remaining 8% as suspected DNST traffic. Upon further analysis of the suspected DNST queries, we found evidence of malicious activity. For example, the domain `easywbdesign.com` was flagged by our DNST classifier and it belongs to the Glupteba malware family as a command-and-control domain [10].

Acknowledgement. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

REFERENCES

[1] B. Yu, L. Smith, M. Threefoot, and F. Olumofin, "Behavior analysis based DNS tunneling detection and classification with big data technologies," in *IoTBD*, 2016, pp. 284–290.

[2] A. Nadler, A. Aminov, and A. Shabtai, "Detection of malicious and low throughput data exfiltration over the DNS protocol," *Computers & Security*, vol. 80, pp. 36–53, 2019.

[3] Z. Yang, Y. Hongzhi, L. Lingzi, H. Cheng, and Z. Tao, "Detecting DNS tunnels using session behavior and random forest method," in *2020 IEEE Fifth International Conference on Data Science in Cyberspace (DSC)*, 2020, pp. 45–52.

[4] R. Sivaguru, J. Peck, F. Olumofin, A. Nascimento, and M. De Cock, "Inline detection of DGA domains using side information," *IEEE Access*, vol. 8, pp. 141 910–141 922, 2020.

[5] P. Yang, Y. Li, and Y. Zang, "Detecting DNS covert channels using stacking model," *China Communications*, vol. 17, no. 10, pp. 183–194, 2020.

[6] J. Saxe and K. Berlin, "eXpose: A character-level convolutional neural network with embeddings for detecting malicious URLs, file paths and registry keys," *arXiv preprint arXiv:1702.08568*, 2017.

[7] B. Yu, J. Pan, D. Gray, J. Hu, C. Choudhary, A. C. Nascimento, and M. De Cock, "Weakly supervised deep learning for the detection of domain generation algorithms," *IEEE Access*, vol. 7, pp. 51 542–51 556, 2019.

[8] C. Qi, X. Chen, C. Xu, J. Shi, and P. Liu, "A bigram based real time DNS tunnel detection approach," *Procedia Computer Science*, vol. 17, pp. 852–860, 2013.

[9] J. Zhang, L. Yang, S. Yu, and J. Ma, "A DNS tunneling detection method based on deep learning models to prevent data exfiltration," in *International Conference on Network and System Security*, 2019, pp. 520–535.

[10] L. Nagy, "Glupteba: Hidden malware delivery in plain sight: Inside a self-concealing malware distribution framework with a security-resistant ecosystem," *SophosLabs*, 2020.