Unsupervised SOM-Based Intrusion Detection System for DNS Tunneling Attacks

Júlio F. Luz[1,2], Paulo Freitas de Araujo-Filho[1], Henrique F. Arcoverde[1,2], and Divanilson R. Campelo[1]

[1]Centro de Informática – Universidade Federal de Pernambuco (CIn - UFPE)
Av. Jorn. Aníbal Fernandes – s/n – Recife – PE – Brazil

{jcfl,pfaf,hfa,dcampelo}@cin.ufpe.br

[2]Tempest Security Intelligence
Paço Alfândega Shopping - Loja 216A, Recife, Pernambuco, 50030-030, Brazil

{julio.farias,henrique.arcoverde}@tempest.com.br

Abstract. *Although the Domain Name System (DNS) is an essential protocol for Internet operation, it may also be used for malicious activities, such as data exfiltration, through the establishment of malicious DNS tunnels. In this paper, we propose an unsupervised intrusion detection system (IDS) for detecting malicious DNS tunneling activities by leveraging self-organizing maps (SOM). Our experimental results show that our proposed solution achieved an F1-score of 0.9460, outperforming similar existing techniques in publicly available datasets, and successfully detected attacks conducted in a corporate network.*

1. Introduction

Domain Name System (DNS) refers to the distributed database implemented in a hierarchy of DNS servers and to the application layer protocol that allows users to query this database. Despite its primary purpose of translating domain names into IP addresses, malicious actors exploit this protocol for illegal purposes by establishing DNS tunnels, i.e., communication channels, between an attacker-controlled server and an infected device. Such a communication channel can then be used to transmit and receive data maliciously into the DNS protocol fields [Wang et al. 2021]. For instance, in a data exfiltration scenario, DNS queries' subdomains are used to transmit data from the victim using queries with the form "stolen-data.attacker-domain.com", where "stolen-data" represents the transmitted data using an encoding technique, such as base32, which allows the query to be resolved by the DNS infrastructure and the attacker to retrieve the exfiltrated data [Wang et al. 2021]. Many malicious actors, including OilRig and xHunt, employ this technique, with the latter having used it to attack governmental organizations [PaloAlto 2021]. Therefore, while DNS is essential to Internet operation, attackers can exploit it to steal sensitive data and cause significant financial loss to institutions.

One of the main reasons attackers rely on DNS tunneling for conducting several cyber-attacks is that it often goes unnoticed by defense mechanisms. For instance, firewalls are typically configured to allow packets on port 53, which is the default port for the DNS protocol, otherwise they would compromise the normal operation of the Internet. Hence, they cannot detect malicious activities conducted through DNS tunnels [Wang et al. 2021]. Moreover, signature-based intrusion detection systems (IDSs), which

rely on signatures of known attacks, cannot detect different malicious activities conducted through DNS tunnels [Wang et al. 2021]. In addition, most of the existing anomaly-based IDSs, which model the benign behavior of networks and systems and detect attacks by measuring deviations from those behaviors, rely on supervised learning techniques that require labeled training data. However, obtaining labeled attack data is costly and supervised IDSs usually do not perform well at detecting attacks that have not been considered in training [Nguyen et al. 2020].

To overcome such limitations, in our work, we propose an unsupervised IDS relying on self-organizing maps (SOM), which is a machine learning technique capable of reducing the dimensionality of data and that has been showing promising results at detecting intrusions [Campbell and Zincir-Heywood 2020]. In constrast to other DNS tunneling detection solutions, our proposed IDS trains the SOM using only benign DNS queries so that it does not require labelled attack data and is not biased for detecting only attacks that have been considered in training. In summary, the main contributions of our work are: (1) The proposal of an unsupervised SOM-based IDS that detects DNS tunneling data exfiltration attacks and achieves a 0.9460 F1-score; (2) Experiments evaluating our proposed IDS in a real enterprise network; (3) Experiments evaluating our proposed IDS using public datasets; (4) Comparison of the proposed method with state-of-the-art.

2. Related Works

Many authors have proposed supervised IDSs for detecting malicious DNS tunneling activities. For instance, the work in [Lambion et al. 2020] relied on convolutional neural networks (CNNs) and random forests to detect malicious activities using linguistic features acquired from DNS queries, achieving an accuracy of 96.65% and an Area Under the Receiver Operating Characteristic Curve (AUCROC) of 0.9984. However, since the detection capability of supervised methods lies within the malicious data used in training, [Lambion et al. 2020] cannot detect well unknown attacks and requires labeled attack data, which is challenging and expensive to obtain. The authors of [Campbell and Zincir-Heywood 2020] employed the SOM algorithm for DNS tunneling detection. Although they considered some experiments training with only benign data, they focused on cases where malicious instances were also present in the training set. The inclusion of malicious instances in the training data can lead to the same issues highlighted for supervised models. The authors of [Nguyen et al. 2020] used the density-based spatial clustering of applications with noise (DBSCAN) algorithm to detect anomalies in network traffic achieving an AUCROC of 0.992. However, they neither clarify how DNS tunneling activities considered were conducted nor evaluated their method in a real network.

3. Proposed Architecture

In our work, we propose an unsupervised anomaly-based IDS for detecting DNS tunneling attacks, such as data exfiltration. Our system is composed of four modules: data collector, preprocessor and feature extractor, detection agent, and notifier. The data collector acquires DNS queries from DNS traffic logs and forwards them to the preprocessor and feature extractor module, which then preprocesses and extracts features from the received DNS queries. The extracted features provide domain name characteristics, such as domain length (i.e., total number of characters of a fully qualified domain name string) and vowel count. They are forwarded to the detection agent module, which employs an

unsupervised SOM model that, after having been trained only on benign data to learn the normal behavior of DNS queries, computes an anomaly score for each new DNS query it analyzes. Finally, since high scores indicate a high likelihood of a query being an anomaly, the notifier module generates alerts for notifying security analysts of suspicious DNS queries whenever the computed anomaly detection score is higher than a threshold. Figure 1 shows our proposed architecture.

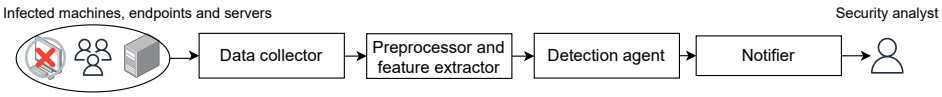

Figure 1. Proposed IDS architecture

Our detection agent employs the anomaly detection technique proposed by [Tian et al. 2014], utilizing SOM and the k-nearest neighbor (KNN) algorithm. SOM networks are composed of "nodes" or "neurons" associated with a vector of weights of the same dimension as the input data. The idea of SOM involves adjusting these neuron weights, distributed within a 2D grid, so that their weights are similar to the input data, and that neurons of similar weights are also close to each other, reflecting input data patterns in the 2D grid. Initially, SOM training initializes neuron weights randomly. Through iterative steps, SOM adjusts these weights, taking each training instance, and measuring the Euclidean distance between the instance and all neurons. The neuron with the smallest distance, called as the "best matching unit" (BMU), is selected. The BMU's weights, along with those of its neighboring neurons within the 2D grid, are adjusted and shifted closer to the input data. The process is executed for a specified number of iterations. Once the network has been trained, the neurons that were not elected as BMUs for at least a minimum number of training instances are considered to represent outlier behaviors and removed. Then, the KNN algorithm is used to identify the k-nearest BMUs to the input data so that an anomaly detection score is computed as the average Euclidian distance from the data input to its k-nearest BMUs. Finally, the computed anomaly detection score is compared to a threshold so that queries with scores higher than the threshold are reported as suspicious. Further details and equations concerning the technique applied by our detection agent can be obtained from [Tian et al. 2014].

4. Methodology and Experimental Evaluation

4.1. Datasets

To validate our proposed solution, we considered two publicly available datasets: DNS Tunneling Queries for Binary Classification dataset [Bubnov 2019] and CAIDA UCSD IPv4 Routed /24 DNS Names Dataset [CAIDA 2021]. The former contains malicious data exfiltration activities conducted through DNS tunnels using different tools, such as dns2tcp, dnscapy, iodine, and tuns. On the other hand, the latter dataset contains benign fully qualified domain names (FQDNs) obtained from studies on Internet topology and consists of tens of millions of domain names. Hence, we combined those two datasets to construct a training, a validation and a testing set. The training set contains only benign data and is used to train our proposed solution. The validation set is used to optimize the hyper-parameters of the trained models. Finally, the testing set is used to evaluate our solution's results and compare them to the results of other works. Table 1 shows the number of benign and malicious samples in the training, validation, and testing sets.

Table 1. Dataset description

Number of domains	Public dataset			Real DNS traffic dataset		
	Training set	Validation set	Testing set	Training set	Validation set	Testing set
Number of benign domains	200,000	8,000	8,000	2,345,219	128,820,871	210,136,888
Number of malicious domains	0	8,000	8,000	0	3,385	247,501

Table 2. Used features and description

Feature	Description
Entropy	Shannon's entropy for a string
Number of subdomains	Number of subdomains of a domain from the third-level domain. (i.e., "www.google.com" has 1)
Maximum label length	Maximum length of a domain level. (i.e., "www.google.com" has 6)
Length	Total number of characters of a domain
Length of continuous integer	Maximum length of continuous integer sequence (i.e., "123abcd45ef.net" has 3)
Length of continuous string	Maximum length of continuous alphabet letters sequence (i.e., "123abcd45ef.net" has 4)
Special character count	Total number of appearances of special characters (excluding dots ".")
Special character ratio	Total number of appearances of special characters (excluding dots ".") divided by the length of domain
Integer character count	Total number of appearances of integer characters
Integer character ratio	Total number of appearances of integer characters divided by the length of domain
Vowel character count	Total number of appearances of integer characters
Vowel character ratio	Total number of appearances of integer characters divided by the length of domain
Reputation value	Value extracted from the popularity of the n-grams found in the string
Reputation value per n-gram	Reputation value divided by the number of n-grams obtained for reputation value calculation.

4.2. Experimental setup

We initially developed the preprocessor and feature extractor module. The first step of this module involves applying filters to remove queries that could not be used for DNS tunneling, as they would have minimal contribution to the model training and evaluation. Following the approach in [Lambion et al. 2020], our module excludes single-level domain queries and reverse DNS queries such as "8.8.8.8.in-addr.arpa". Afterwards, the second step of the module is to extract the linguistic features that are listed in Table 2. Most of these features were inspired by the work in [Park et al. 2022]. For instance, the *reputation value* consists of a value extracted from the popularity of the n-grams found in a domain name, adding a weight $W_{N-Gram}(i) = log_2(N \times C_{N-Gram(i)})$ to each n-gram found in the string, where $W_{N-Gram}(i)$ is the weight given to the i-th n-gram of a string, N is the character size of the n-gram and $C_{N-Gram(i)}$ is how many times this n-gram appeared in the top 100,000 most accessed domains of the 1 million most accessed domains in the world, obtained from Majestic Million [Majestic 2023].

We trained our proposed SOM using a 30x30 neural network and tuned its hyperparameters such as learning rate, initial radius, number of neighbors, and minimum number of training instances per BMU. While such experiments were first conducted using the training, validation, and testing sets described in Section 4.1, we later conducted additional experiments on the internal network of Tempest Security Intelligence (TSI), which has over 500 connected endpoints generating DNS logs. We used DNS queries collected in March 2023 for constructing our training and validation sets, and DNS queries collected in April 2023 for constructing our testing set, assuming that they represented benign samples. In addition, we conducted several DNS tunneling data exfiltrations using the iodine and DNSExfiltrator tools in April 2023 for obtaining the malicious DNS queries needed for our validation and testing sets. Finally, a main constraint of our experiments on the enterprise network was to minimize false positives as we could not overwhelm analysts with numerous false alarms. Table 1 summarizes our real DNS traffic datasets.

5. Results and Discussion

We evaluated our solution's detection results by computing its accuracy, F1-Score, precision, recall, true positive rate (TPR), false positive rate (FPR), true negative rate (TNR), and false negative rate (FNR). In addition, using the same datasets as those of our methodology for training, hyperparameter tuning, and result evaluation, we compared our results to those from an implementation inspired in the work of [Campbell and Zincir-Heywood 2020], which also proposed a SOM-based IDS for detecting malicious DNS tunneling activities. Unlike ours, this method assigns a sample under evaluation to the bening or malicious class according to its closest neuron, which has been previously assigned to a class after training. In contrast, our solution does it according to an anomaly score that is computed by measuring the distances between the sample and its k closest SOM neurons. As shown in Table 3, our solution obtained better results for all considered metrics. Finally, we computed the AUCROC of our proposed IDS, obtaining a value of 0.9815, as shown in Figure 2.

Table 3. Results for experiments

Model	Accuracy	F1-Score	Precision	Recall	TPR	FPR	TNR	FNR
Proposed architecture	**94.38%**	**0.9460**	**0.9107**	**0.9841**	**98.41%**	**9.65%**	**90.35%**	**1.59%**
[Campbell and Zincir-Heywood 2020]	92.24%	0.9265	0.8800	0.9783	97.83%	13.34%	86.66%	2.16%

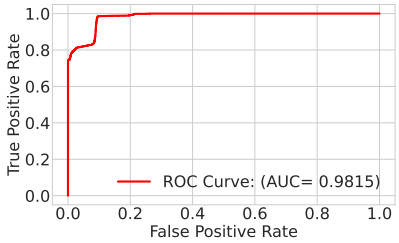

Figure 2. ROC Curve and AUC

Experiments were also conducted on real DNS traffic data using the same training, validation, and testing sets for the two approaches. [Campbell and Zincir-Heywood 2020] exhibited a relatively high false positive rate of 1.13%, resulting in over 2M false positives, making it difficult to implement in real environments. Moreover, this method does not propose the use of a threshold or any other mechanism to minimize the number of false positives. On the other hand, our proposal results show that only 121 queries were false positives, which corresponds to less than 0.0001% of FPR. We also found that when considering the count of false positives in 5-minute time windows, none of them had more than 15 false positives. Thus, we could further decrease the number of false alarms by triggering alerts only when more than 15 positive alarms have occurred in a 5-minute time window. The results regarding the detection of individual DNS queries from our proposal for real data are shown in Table 4. Since the queries were generated within a single company, it was easier for the SOM to identify query patterns than when using the public data, which was collected from multiple sources. Hence, our proposed model achieved a high TPR for a very low FPR for the data collected from TSI. Finally, we highlight that our proposed IDS effectively detected all conducted attack experiments. Since DNS tunneling data exfiltration attacks typically produce hundreds of malicious queries for each attack occurrence, and our IDS detects most of those queries, our solution successfully identified all occurrences of the conducted attacks.

Table 4. Posterior test results for real DNS traffic data

Tool	Exfiltrated file size	Accuracy	F1 Score	Precision	Recall	TPR	FPR	TNR	FNR
Iodine	1.6 MB	99.99%	0.8524	0.9982	0.7438	74.38%	0.00006%	99.99994%	25.62%
	200 KB	99.99%	0.8295	0.9899	0.7138	71.38%	0.00006%	99.99994%	28.62%
	32 KB	99.99%	0.7623	0.9745	0.6261	62.61%	0.00006%	99.99994%	37.39%
DNSExfiltrator	1.6 MB	99.99%	0.9993	0.9989	0.9998	99.98%	0.00006%	99.99994%	0.02%
	200 KB	99.99%	0.9959	0.9930	0.9989	99.89%	0.00006%	99.99994%	0.11%
	32 KB	99.99%	0.9904	0.9850	0.9959	99.59%	0.00006%	99.99994%	0.41%

6. Conclusions and Future Work

In this work, we proposed an unsupervised IDS for detecting DNS tunneling data exfiltration. Our solution eliminates the need for malicious domains in training. Thus, it reduces costs related to acquiring malicious domains, such as attack simulations and label assignments expenses. It achieved an F1-Score of 0.9460 and showed promising results for the IDS deployment in a real enterprise network. In the future, we will consider volumetric features of DNS traffic in our proposed solution to improve its detection results.

References

[Bubnov 2019] Bubnov, Y. (2019). DNS Tunneling Queries for Binary Classification. Mendeley Data.

[CAIDA 2021] CAIDA (2021). The CAIDA UCSD IPv4 Routed /24 DNS Names Dataset. https://www.caida.org/catalog/datasets/ipv4_dnsnames_dataset/.

[Campbell and Zincir-Heywood 2020] Campbell, A. J. and Zincir-Heywood, N. (2020). Exploring tunneling behaviours in malicious domains with self-organizing maps. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1419–1426.

[Lambion et al. 2020] Lambion, D., Josten, M., Olumofin, F., and De Cock, M. (2020). Malicious DNS Tunneling Detection in Real-Traffic DNS Data. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 5736–5738.

[Majestic 2023] Majestic (2023). Top 1 million websites in the world. https://majestic.com/reports/majestic-million.

[Nguyen et al. 2020] Nguyen, T. Q., Laborde, R., Benzekri, A., and Qu'hen, B. (2020). Detecting abnormal dns traffic using unsupervised machine learning. In *2020 4th Cyber Security in Networking Conference (CSNet)*, pages 1–8.

[PaloAlto 2021] PaloAlto (2021). Real-world Examples Of Emerging DNS Attacks and How We Must Adapt. https://www.paloaltonetworks.com/blog/2021/05/netsec-dns-attacks/.

[Park et al. 2022] Park, K. H., Song, H. M., Yoo, J. D., Hong, S.-Y., Cho, B., Kim, K., and Kim, H. K. (2022). Unsupervised Malicious Domain Detection with Less Labeling Effort. *Comput. Secur.*, 116(C).

[Tian et al. 2014] Tian, J., Azarian, M. H., and Pecht, M. G. (2014). Anomaly Detection Using Self-Organizing Maps-Based K-Nearest Neighbor Algorithm.

[Wang et al. 2021] Wang, Y., Zhou, A., Liao, S., Zheng, R., Hu, R., and Zhang, L. (2021). A comprehensive survey on DNS tunnel detection. *Computer Networks*, 197:108322.