

FQDN Similarity and Cache-Miss Property based DNS Tunneling Detection Technique

Madhuparna Bhowmik¹, Aastha Chowdhary² and Dr. Bhawana Rudra³

¹⁻³ NIT Karnataka, Information Technology Department, Surathkal, India
Email: ¹⁻³{madhuparnabhowmik04, aasthac67, bhawana rudra7}@gmail.com

Abstract—Although there are many effective methods to detect DNS Tunneling attacks, the attacks still happen, and the attackers can mock genuine queries to bypass such checks. However, in data exfiltration, the DNS queries are continuously changing as some part of it represents the data itself. Thus, all such queries would result in a cache miss, and therefore we can use such properties to detect DNS Tunneling attacks. However, relying on this is not enough as it will also have many false positives. To overcome the problem, we propose three criteria-based methods that consider DNS Tunneling queries' properties and use them to reduce the number of false positives and thus accurately detect DNS Tunneling traffic. We even discussed the bypassing checks in this paper, and they are both costly and require the attacker to make redundant queries.

Index Terms— DNS Tunneling, DNS Cache Server, DNS Tunneling Detection, dnscat2, FQDN, DNS Queries and Data Exfiltration.

I. INTRODUCTION

DNS Tunneling attacks are still possible and are challenging to detect. Exfiltrating the large amounts of data at once is difficult as there exist many measures to detect such DNS queries of bigger length. Some methods can detect the queries which look very different from the genuine ones, though there are still ways to mimic genuine queries. For example, a query of the form mypassword.attacker.com looks genuine as it is neither very long nor consisting of a random hash. We focus on detecting such DNS tunneling queries, which are almost indistinguishable from genuine ones. In Ishikura et al. [1] an essential property of DNS tunneling queries is used, i.e., a query would definitely result in a cache miss as to when exfiltrating data some part of the query would always be changing. However, even other queries like a genuine query made for the first time would also result in a cache miss. Therefore, we propose three criteria based on the property of DNS queries used for data exfiltration to filter out the genuine queries from the pool of queries resulting in a cache miss. We further show that the proposed method is effective because bypassing the three criteria proposed in this work would require the attacker to do redundant operations and buy many domain names, which is cost-inefficient.

II. LITERATURE SURVEY

Malicious users have found various methods to bypass security policies in recent years, and DNS tunneling is being exploited a lot lately. Data exfiltration takes place where unauthorized data transfer exploits DNS tunnelling to conceal network activity in a regular DNS traffic [2]. As DNS is an indispensable protocol and its

use can't be restricted. Attackers release their attacks by making use of DNS tunneling [1]. Many methods have been previously proposed to detect DNS tunneling attacks; machine learning methods became very common due to their reliability and flexibility. Yakov Bubnov [2] proposed a simple three-layer feedforward neural network-based method with a multilabel classification output, where each class represents a different DNS tunneling method like dns2tcp, dnscapy, etc. Ahmed Almusawi and Haleh Amintoosi [3] proposed a multilabel classification model using Support Vector Machine to detect DNS tunneling based attack. This method makes use of extracting features and perform classification on them. Features such as byte frequency, length of the packet, etc., are used as important features that are extracted from the data. In [2], the method is also able to detect not only tunneling packets but also the type of tunneling.

Jiacheng Zhang et al. [4] focused on preventing data exfiltration rather than detecting it. Time-frequency related features were taken into consideration when data exfiltration had already taken place. They made use of deep learning models like DNN, RNN, etc., along with NLP and made use of DNS query payloads as predictive variables in the model. Huiwen Bai et al. proposed that DNS tunneling is measured based on two perspectives, i.e., length and interarrival time (IAT) of the packets [6]. Also, the amount of data transmitted per packet is an important feature for the identification of different protocols. A regression model was proposed for the identification of DNS tunneling traffic. Maurizio Aiello et al. made use of supervised learning on statistical features of DNS queries [7]. They made use of the time interval and packet size of the query. Bayes classifier was adopted, which was built on the statistical features of DNS messages.

Mahmoud Sammour, Burairah Hussin, Mohd Fairuz Iskandar Othman [10] used features such as information entropy, DNS request length, IP packet sender length, IP packet response length, etc. They then made use of three different classifiers for the task of DNS tunneling detection, namely, SVM, NB, and J48, and found out SVM to have the best performance. Anirban Das, Min-Yi Shen, Madhu Shashanka, and Jisheng Wang [11] used normalized entropy, length, Ratio of the number of uppercase alphabets in the string to a total number of elements in the string, etc. as features for detecting exfiltration and then applied linear regularized logistic regression. In the case of tunneling detection, they used features such as several uppercase alphabets in the response, the number of space-like characters, etc., and then applied k-means clustering using these features.

E. Cambiaso, M. Aiello, M. Mongelli, G. Papaleo [12] exploited features such as mean, standard deviation, skewness, and kurtosis of the inter-arrival times of DNS messages and the sizes of DNS queries and answers. Then used Principal Component Analysis (PCA) to synthesize the features and used Mutual Information (MI) to elaborate discriminant separations among malicious and clean scenarios.

Bin Yu, Les Smith, Mark Three-foot, and Femi Olumofin proposed a method based on feature extraction and selection as well. They proposed that entropy metrics and human readability of domain names are good features/indicators to detect DNS tunneling. A two-tier classification was performed based on the same concept [5]. Asaf Nadler, Avi Aminov, and Asaf Shabtai [8] first transformed the DNS traffic collected into feature vectors corresponding to the domain then made use of a one-class classifier to detect domain that exchanged data over DNS.

There are mainly two types of DNS tunneling features, the payload analysis technique, which primarily aims to observe DNS payload and includes features such as the size of the request and response, the entropy of hostname, and uncommon record types, etc. The second type of DNS tunneling feature is traffic analysis, which focuses on traffic over time and includes the volume of DNS traffic per domain, the volume of DNS traffic per IP address, domain history, etc [9]. All the models discussed above work on such features and use machine learning models to detect DNS tunneling. But these features can be easily mimicked by the attackers, and the queries can look like benign ones. This can lead to a high risk of data leakage [1].

Naotake Ishikura et.al. [1] proposed a model to overcome the issue where the attacker mimics the queries such that it looks like actual DNS queries. It is observed that DNS tunneling queries is used for exfiltrating data, produces a cache miss on the DNS cache server. This acts as a trace of data leakage. They proposed two features based on the cache properties cache hit ratio (CHR) and access hit ratio (AHR). However, such methods are prone to false positives as clients making queries to a new website that they have not visited earlier would also cause a cache miss. Our paper also utilizes DNS Tunneling queries' cache-miss property and proposes three criteria to effectively detect if a DNS query causing a cache miss can be a DNS tunneling query or just a false positive.

III. MOTIVATION

The motivation of our work was to come up with ways of detecting DNS tunneling attacks and coming up with a new measure i.e. a similarity metric of FQDN which can improve the previously proposed method and help detect DNS tunneling attacks.

The domain of this paper is related to detection of DNS tunneling using the cache-miss property along with similarity metric of FQDN.

IV. METHODOLOGY

A. About The Dataset

The dataset for this experiment was generated using the dnscat2 tool. Three datasets were generated where for a certain amount of time a user randomly went about browsing different websites that were genuine and in between during random times and for random intervals, the DNS tunneling server exfiltrated data thus generating DNS tunneling traffic. Some sample queries can be seen in Figure 1.

The sample query has various fields such as the time at which the packet was generated, the source and the destination IP address of the packet, the Protocol it is following, the length of the packet and some information about the packet. This dataset was captured using wireshark along with dnscat2 as described above.

B. Proposed Methodology

DNS Tunneling queries would cause a cache miss; however, the vice-versa is not valid. Cache miss can also happen when the users try to access a website they have not previously visited. Hence, only relying on cache misses would lead to many false positives. Therefore, we developed three different criteria to eliminate most of the false positives. The three criteria are based on the query's length, the similarity of the query with the previous queries, and frequency of the same query in the near future. The first two criteria can be used on-line such that there is no information about future queries are required. However, one of them needs to know the frequency of similar queries in the future. This work's novelty lies in the second criteria that use a critical property of DNS tunneling queries. However, the first and the Third criteria are used to further increase the accuracy by covering more cases.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.001719	192.168.1.192	168.1.192	DNS	101	Standard query 0x0ac9 CNAME dnscat.e859014a4ae518df043be7001620499b2d
2	0.001719	192.168.1.192	168.1.192	DNS	156	Standard query response 0x0ac9 CNAME dnscat.e859014a4ae518df043be7001620499b2d CNAME dnscat.9e87014a4ae228b5d70d56ffff79fef54
3	0.320644	fe80::35b:ff02::fb		MDNS	102	Standard query 0x0000 PTR _pgkey-hkp._tcp.local, "QM" question
4	0.320775	192.168.1.224	0.0.0.25	MDNS	82	Standard query 0x0000 PTR _pgkey-hkp._tcp.local, "QM" question
5	0.626416	91.189.92.192	168.1.192	TLSv1.2	97	Encrypted Alert
6	0.626417	91.189.92.192	168.1.192	TCP	66	443 > 49606 [FIN, ACK] Seq=32 Ack=1 Win=258 Len=0 TSval=3848491001 TSecr=1035499621
7	0.626481	192.168.1.91	189.92.192	TCP	66	49606 > 443 [ACK] Seq=1 Ack=32 Win=523 Len=0 TSval=1035504023 TSecr=3848491000
8	0.626697	192.168.1.91	189.92.192	TLSv1.2	97	Encrypted Alert
9	0.626768	192.168.1.91	189.92.192	TCP	66	49606 > 443 [FIN, ACK] Seq=32 Ack=33 Win=523 Len=0 TSval=1035504023 TSecr=3848491001
10	0.933755	91.189.92.192	168.1.192	TCP	60	443 > 49606 [RST] Seq=33 Win=0 Len=0
11	0.933755	91.189.92.192	168.1.192	TCP	60	443 > 49606 [RST] Seq=33 Win=0 Len=0
12	1.005131	192.168.1.192	168.1.192	DNS	101	Standard query 0x08ac CNAME dnscat.21fd014a4abf649c5c74a00017cb370a04
13	1.00687	192.168.1.192	168.1.192	DNS	156	Standard query response 0x08ac CNAME dnscat.21fd014a4abf649c5c74a00017cb370a04 CNAME dnscat.dabd014a4a2d429452526cffff79fef54
14	2.011095	192.168.1.192	168.1.192	DNS	101	Standard query 0xed22 TXT dnscat.61eb014a4a0898579ce800018fb2ab880
15	2.012855	192.168.1.192	168.1.192	DNS	148	Standard query response 0xed22 TXT dnscat.61eb014a4a0898579ce800018fb2ab880 TXT
16	2.674895	140.82.11.192	168.1.192	TLSv1.2	92	Application Data
17	2.675157	192.168.1.140	82.11.192	TLSv1.2	96	Application Data
18	3.017456	192.168.1.192	168.1.192	DNS	101	Standard query 0xd12d TXT dnscat.e028014a4a9aebbfb813d001962ce92b0
19	3.019015	192.168.1.192	168.1.192	DNS	148	Standard query response 0xd12d TXT dnscat.e028014a4a9aebbfb813d001962ce92b0 TXT
20	3.084851	140.82.11.192	168.1.192	TCP	66	443 > 40528 [ACK] Seq=27 Ack=31 Win=70 Len=0 TSval=3713292442 TSecr=2887191681
21	4.023037	192.168.1.192	168.1.192	DNS	101	Standard query 0x1a5f MX dnscat.dabd014a4a5a5ac293df2e001a6a30f764
22	4.024765	192.168.1.192	168.1.192	DNS	158	Standard query response 0x1a5f MX dnscat.dabd014a4a5a5ac293df2e001a6a30f764 MX 10 dnscat.b115014a4a3872c59fa4d3ffff79fef54
23	5.029125	192.168.1.192	168.1.192	DNS	101	Standard query 0x7082 MX dnscat.8b4a014a4ab9308d05ef0d001b0a11bde5
24	5.030986	192.168.1.192	168.1.192	DNS	158	Standard query response 0x7082 MX dnscat.8b4a014a4ab9308d05ef0d001b0a11bde5 MX 10 dnscat.2f5d014a4a18317cabe684ffff79fef54
25	6.035077	192.168.1.192	168.1.192	DNS	101	Standard query 0x12ba TXT dnscat.fea1014a4a8a818e9ab707001c4b4c75ac
26	6.036709	192.168.1.192	168.1.192	DNS	148	Standard query response 0x12ba TXT dnscat.fea1014a4a8a818e9ab707001c4b4c75ac TXT
27	7.040868	192.168.1.192	168.1.192	DNS	101	Standard query 0x9052 CNAME dnscat.af0c014a4a7105d83e84e6001db088cd55
28	7.042361	192.168.1.192	168.1.192	DNS	156	Standard query response 0x9052 CNAME dnscat.af0c014a4a7105d83e84e6001db088cd55 CNAME dnscat.fcff014a4a8b018c14133ffff79fef54

Figure 1. Sample queries from the dataset generated and used

a) Query-length based similarity calculation

In this criteria, we consider the length of the query. The DNS queries that lead to a cache miss are further classified based on their length. Since the attacker is trying to exfiltrate the data, the DNS tunneling queries are usually longer than the normal DNS queries. Even though this property alone is very naive and such a check can be easily bypassed if the attacker exfiltrates very small amounts of data at a time. This method can still be very effective as very often; the attacker performs such large queries. It does not accurately classify a query as malicious or not, but instead, it is a good indication of the possibility.

b) FQDN Similarity

Consider the attacker's domain is attacker.com. All the DNS Tunneling queries would be of type habci.attacker.com. Even though attackers may frequently change the domain name used, it is expensive to buy

domain names, and at least for a period of time, like 1 hour to 1 day, they would be most likely using the same domain name, e.g., attacker.com. As a result, only some parts of the DNS query will change while exfiltrating data. E.g., two subsequent queries can be username.attacker.com and password.attacker.com. However, in case of genuine queries resulting in a cache miss, they would differ significantly from the previous queries. For example, if the user is opening github.com for the first time, it would result in a cache miss but would have very little similarity with the previous queries. Hence, we can consider a sequence of subsequent DNS tunneling query exfiltrating data as follows: hash1.attacker.com, hash2.attacker.com, hash3.attacker.com ... hash100.attacker.com. The hash part representing the data is changing and thus causing cache miss, but the rest of the query is similar to the previous queries that resulted in a cache miss. Hence the similarity of hash14.attacker.com with hash3.attacker.com is very high. But github.com is very different from attacker.com or other queries that resulted in a cache miss earlier. This shows that even though such DNS tunneling queries cause a cache miss always, they are still very similar to the previous queries. A genuine site like github.com is unlikely to use many domain names such as abc.github.com, xyz.github.com, etc. Even if such queries exist, their frequency would be pretty less. Though cases such as a user accessing docs.google.com and sheets.google.com would also result in a cache miss and high similarity with previous queries. To overcome this drawback, we use the third criteria, "access frequency," as mentioned below. This is based on the assumption that the user would most likely repeat genuine queries like docs.google.com in the future. Whereas the queries used for exfiltrating data are unlikely to be repeated in the future.

c) Access frequency

Here we consider a log of DNS queries and try to figure out if a query in the past that resulted in a cache miss was a DNS tunneling query or not. For example, if docs.google.com resulted in a cache miss but was accessed many times in the future, then it is most likely a genuine query as an attacker would not exfiltrate the same data multiple times. Moreover, often the tunneled data is not in plain text but say a hash of the data. Hence, using this criterion that DNS tunneling queries will most likely not repeat and genuine queries will repeat in the future can further filter out false positives.

Overall, these three criteria effectively filter out false positives from the pool of queries resulting in a cache miss and can accurately detect the DNS tunneling queries. Figure 2 summarizes the system's overall working and demonstrates how a query causing a cache miss is further processed and classified as a Genuine vs. Suspicious Query.

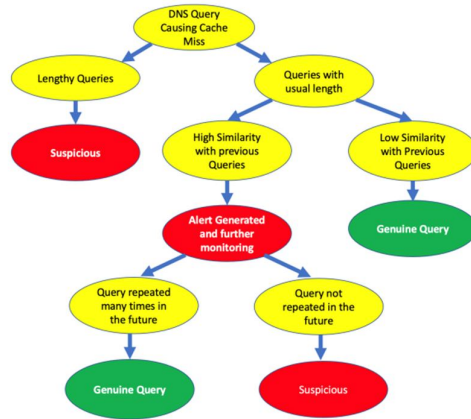


Figure 2. Decision Tree demonstrating the working of the Tool

V. EXPERIMENTAL SETUP AND RESULTS

We used an authoritative DNS server, DNS Cache server, DNS Tunneling Client, and DNS Tunneling Server to conduct the experiments. We used dnscat2 as the DNS Tunneling Client and Server. The client's data was collected for a fixed duration, and the DNS queries that resulted in a cache miss were further analyzed using a python script employing the three criteria to detect if a resulted cache-miss was due to a genuine DNS query or DNS Tunneling query. The client was continuously generating genuine traffic by visiting regular sites using the web browser, whereas the DNS tunneling client was active at some periods and exfiltrating data. We used C2C

or Command and Control from the server-side for a random amount of duration and a few times during the data collection. Multiple samples were collected using the above experimental setup and evaluated.

Using the three approaches mentioned in the methodology section, we decreased the number of false positives by approximately 85% from a system only using cache miss as a filtering criterion. This means that using the three criteria along with the cache-miss property; we were able to detect all the DNS tunneling queries along with some false positives. And using the three criteria, we were able to filter out 85% of the false positives. The following plots show the results obtained from three different datasets collected from a compromised client. Figure 3, 5, and 7 show the queries that resulted in a cache miss, and Figure 4, 6, and 8 show the queries identified as a DNS tunneling query using the three criteria. In the first set of graphs showing cache-miss property, the x-axis denotes the query number, and the y-axis is 0 for a query with cache-hit and 1 for a query that resulted in a cache-miss. So, too many blue lines in the graph show frequent cache misses. The second set of graphs (Figure 3, 5, and 7) shows the DNS Tunneling queries identified using the three criteria. In the second set of graphs, the DNS tunneling queries are marked with 1 on the y-axis, and others are marked with 0. Hence the region of frequent blue lines indicates an attack going on. It shows that many of the queries that resulted in a cache miss but were not DNS tunneling queries were identified and marked as false positives, resulting in fewer blue lines on the second set of graphs.

Next, if we consider the ways to bypass the three criteria, the attacker needs to do the following:

1. The attacker cannot use very big queries, so cannot exfiltrate a lot of data at a time. This makes it necessary for the attacker to make a lot of small queries to exfiltrate any useful amount of data.
2. Frequently change the domain name as the attacker cannot use the same domain for too long. Even if a mix of 2-3 domain names is being used, the attacker would still need to make a lot of queries with one domain name to exfiltrate a significant amount of data because of the 1st property. Using a lot of domain names is costly.
3. The attacker needs to make a lot of redundant queries because only if the query such as username.attacker.com is used a lot in the future, the discussed method will not be able to detect it as a DNS tunneling query.

This shows that bypassing the three checks would be costly for the attacker as the attacker needs to buy many domain names, make redundant queries, and cannot exfiltrate a lot of data in one query. This shows the strength of employing the three criteria proposed in this paper.

VI. CONCLUSION

We discussed a new method for detecting DNS tunneling that uses three prime features of DNS tunneling queries along with the cache-miss property. We built an effective defense against DNS tunneling attacks. In the future, we would like to build an effective firewall that can detect possible DNS tunneling queries and report the user or system administrator. Further, using a database of verified websites can also help in reducing the number of false-positives. There is still scope for improvement in this method, and in the future, we will improve accuracy by using other properties of DNS Tunneling attacks.

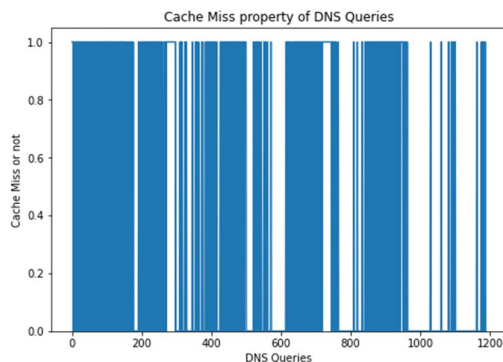


Figure 3. Dataset1: Cache Miss Property

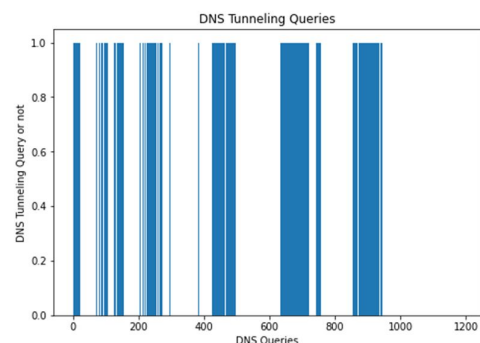


Figure 4. Dataset1: DNS Tunneling Attack

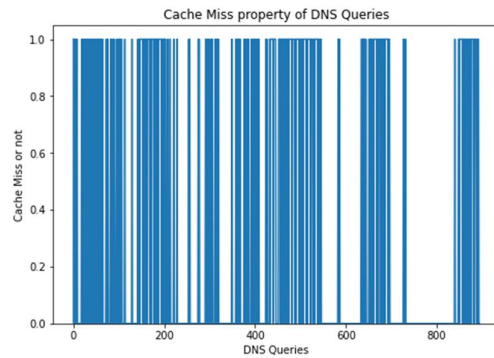


Figure 5. Dataset2: Cache Miss Property

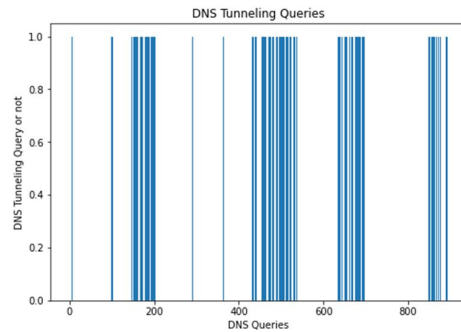


Figure 6. Dataset2: DNS Tunneling Attack

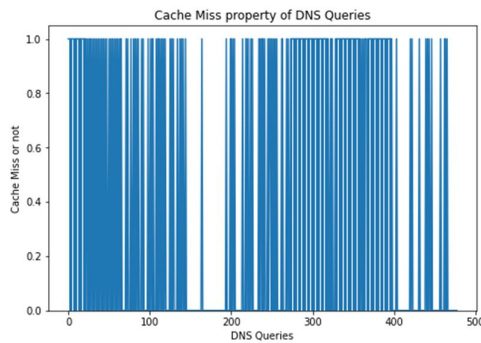


Figure 7. Dataset3: Cache Miss Property

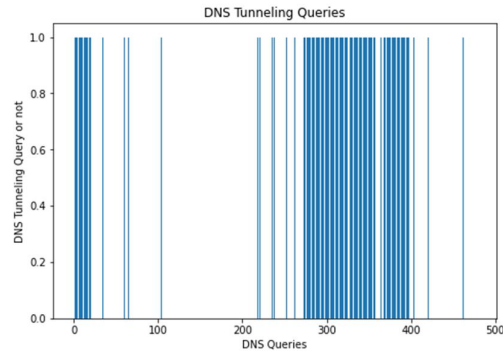


Figure 8. Dataset3: DNS Tunneling Attack

REFERENCES

- [1] Naotake Ishikura, Daishi Kondo, Iordan Iordanov, Vassilis Vassiliades and Hideki Tode - "Cache-Property-Aware Features for DNS Tunneling Detection" - 2020
- [2] Yakov Bubnov - "DNS Tunneling Detection Using Feedforward Neural Network" - EJERS, European Journal of Engineering Research and Science Vol. 3, No. 11, November 2018
- [3] Ahmed Almusawi and Haleh Amintoosi - "DNS Tunneling Detection Method Based on Multilabel Support Vector Machine" - Security and Communication Networks Volume 2018
- [4] Jiacheng Zhang, Li Yang, Shui Yu, Jianfeng Ma - "A DNS Tunneling Detection Method Based on Deep Learning Models to Prevent Data Exfiltration" - International Conference on Network and System Security (2019)
- [5] Bin Yu, Les Smith, Mark Threefoot and Femi Olumofin - "Behavior Analysis based DNS Tunneling Detection and Classification with Big Data Technologies" - International Conference on Internet of Things and Big Data (IoTBD 2016)
- [6] Huiwen Bai, Guangjie Liu, Jiangtao Zhai, Weiwei Li, Xiaopeng Ji, Luhui Yang, Yuewei Dai - "Refined identification of hybrid traffic in DNS tunnels based on regression analysis" - ETRI Journal (2020)
- [7] Maurizio Aiello, Maurizio Mongelli, Gianluca Papaleo - "Basic classifiers for DNS tunneling detection" - IEEE Symposium on Computers and Communications (ISCC) (2013)
- [8] Asaf Nadler, Avi Aminov and Asaf Shabtai - "Detection of malicious and low throughput data exfiltration over the DNS protocol" - Computers and Security Volume 80, January 2019
- [9] Mahmoud Sammour, Burairah Hussin, Mohd Fairuz Iskandar Othman, Mohamed Doheir, Basel AlShaikhdeeb, Mohammed Saad Talib - "DNS Tunneling: a Review on Features" - International Journal of Engineering and Technology (2018)
- [10] Mahmoud Sammour, Burairah Hussin, Mohd Fairuz Iskandar Othman - "Comparative Analysis for Detecting DNS Tunneling Using Machine Learning Techniques" - International Journal of Applied Engineering Research ISSN 0973-4562 Volume 12, 2017
- [11] Anirban Das, Min-Yi Shen, Madhu Shashanka and Jisheng Wang - "Detection of Exfiltration and Tunneling over DNS" - 2017 16th IEEE International Conference on Machine Learning and Applications.
- [12] E. Cambiaso, M. Aiello, M. Mongelli, G. Papaleo - "Feature Transformation and Mutual Information for DNS Tunneling Analysis" - 2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN)