Detection of DNS tunneling in mobile networks using machine learning

Van Thuan Do[1], Paal Engelstad[2], Boning Feng[2] ,Thanh van Do[3,2]

[1] Wolffia AS, Martin Linges vei 15, 1364 Fornebu, Norway
[2] Oslo and Akershus University College of Applied Sciences,
Pilestredet 46, 0167 Oslo, Norway
[3] Telenor ASA. Snarøyveien 30 1331 Fornebu, Norway
vt.do@wollfia.no
{paal.engelstad, boning.feng}@hioa.no
thanh-van.do@telenor.com

Abstract. Lately, costly and threatening DNS tunnels on the mobile networks bypassing the mobile operator's Policy and Charging Enforcement Function (PCEF), has shown the vulnerability of the mobile networks caused by the Domain Name System (DNS) which calls for protection solutions. Unfortunately there is currently no really adequate solution. This paper proposes to use machine learning techniques in the detection and mitigation of a DNS tunneling in mobile networks. Two machine learning techniques, namely One Class Support Vector Machine (OCSVM) and K-Means are experimented and the results prove that machine learning techniques could yield quite efficient detection solutions. The paper starts with a comprehensive introduction to DNS tunneling in mobile networks. Next the challenges in DNS tunneling detections are reviewed. The main part of the paper is the description of proposed DNS tunneling detection using machine learning.

Keywords: Mobile network security, mobile fraud, mobile privacy, cyber security, cyber attacks, mobile vulnerability, machine learning

1 Introduction

The emergence of fancy, powerful but user-friendly mobile devices such as smartphones, tablets, etc. combined with the deployment of mobile wireless broadband access like 3G/4G have made mobile wireless Internet access the most popular Internet usage form surpassing by far the fixed Internet access. However, although affordable for the majority, the mobile wireless access to the Internet is not free of charge and the user is charged based on the used data volume. Indeed, mobile operators usually offer flat fees for different data volumes per month. Further, mobile data usage while roaming on foreign mobile network is very expensive and unaffordable for most of regular non-business users. This explains some individual's motivations and efforts to bypass the mobile operator's charging function and to get free Internet

access. But, most difficult to accept is the behavior of a few dishonest mobile operators who equip their customer's smartphones with apps that enable the evasion of the visited operator's charging function when roaming. These apps use quite often DNS (Domain Name System) [1] [2] tunneling, a method which is motivated by the need of Internet access at Wifi hotspots while evading the fees. DNS tunneling causes obviously revenue losses to mobile operators. But, most importantly, it could be used by any attack that requires firewall evasion i.e. an attacker can send and receive commands and data bypassing the firewall. The simplest but not less serious attack is the theft of confidential information such as personal data, health care data, credit card numbers, payroll, etc. that has financial value. For mobile operators DNS tunneling is not only causing loss of revenues but also deteriorating the quality of service of the overall wireless access and damaging their reputation. A solution preventing DNS is urgently needed.

One obvious solution to prevent DNS tunneling abuses is to block all malicious DNS queries and responses. Unfortunately, this is not a trivial task because differentiating malicious traffic from legitimate one is very challenging if not impossible while blocking the entire DNS traffic is not an inadmissible option. Actually, there is currently no really efficient prevention solution that is mobile operators can use.

In this paper we propose to use machine learning techniques to detect and mitigate DNS tunneling. The paper starts with a state-of-the-art detection and prevention of DNS tunneling, which is followed by a comprehensive introduction to DNS tunneling in the mobile network. Next the challenges of DNS tunneling detection are analyzed. A brief introduction of machine learning and clarifications on how it can be useful in the DNS tunneling detection are then given. The main part of the paper is the description of the proposed DNS tunneling detection using machine learning. The paper concludes with some suggestion for further works.

2 State-of-the-art detection and prevention of DNS tunneling

Actually DNS tunneling is a known vulnerability that has been known for many years now [3] [4] and there were a lot of works on detection and prevention of DNS tunneling both in academia and in industry. The prevention tools can be classified as following:

Firewalls

All firewalls allows the definition of rules to prevent IP spoofing and to deny DNS queries from IP addresses outside the defined numbers space to prevent the name resolver from being exploited as an open reflector in DDoS attacks. They also enable inspection of DNS traffic for suspicious byte patterns or anomalous DNS traffic to block DNS tunneling. Popular firewalls such as Palo Alto Networks, Cisco Systems, WatchGuard, etc. can detect and block certain DNS tunneling traffic. Unfortunately, they are only efficient against known DNS tunneling methods but are not usable when it comes to the unknown ones.

Intrusion detection systems

Intrusion detection systems (IDS) like Snort, Suricata or OSSEC allow the composition of rules to report DNS request from unauthorized clients, to count DNS queries and responses, DNS queries made using TCP, DNS queries to nonstandard ports, suspiciously large DNS queries, any value in any field of the DNS query, etc. However, theses IDS can only detect the known attacks.

Traffic analyzers

Passive traffic analyzers i.e. analyzers that monitor traffic without injecting traffic into the network or modify the traffic that is already on the network, can be used in the identification of DNS tunneling. Unfortunately, these analyzers rely on the knowledge of the traffic amount and patterns.

Passive DNS replication

Replication of every DNS queries and responses enables analysis that could identify malware using domain name generated by Domain Generation Algorithm (DGA). Passive DNS replication can be used together with IDS to block known malicious domains but again are not usable for the unknown ones.

The DNS tunneling in the mobile network poses additional challenges in terms of processing capability and real time response because of the much larger number of users and considerable number of unknown visiting users but so far according to our knowledge there is not yet any detection work dedicated especially for mobile DNS tunneling.

3 Brief introduction to DNS tunneling in the mobile network

To introduce DNS tunneling in the mobile network, it is necessary to explain the Internet access from the mobile network. The mobile network is actually a complex network consisting of several mobile networks e.g. 2G, 3G and 4G with a multitude of network elements having different functions. However, since our focus is on the access to the Internet, it is sufficient to consider a simplified representation of the mobile network as shown in Figure 1.

Figure 1 Mobile wireless access to the Internet

As the user attempts to browse and visit a certain web site http://www.website.com a Packet Data service is started. A Packet Data Protocol (PDP) Context Activation is initiated. It establishes a bearer between the mobile phone through the SGSN (Serving GPRS Support Node) and the GGSN (Gateway GPRS Support Node) to the Internet. A Packet Data Protocol (PDP) context is established with the GGSN. The PDP context is a data structure that contains the subscriber's session information, such as IP

address, International Mobile Subscriber Identity (IMSI), and Mobile Station International Subscriber Directory Number (MSISDN). A tunnel is established between the SGSN and the GGSN. User traffic is encapsulated using GTP-U protocol. At the GGSN it is decapsulated and sent to the PDN through the Gi interface (or the Gp in case of roaming). PDP establishment and termination occurs through the GPRS Tunneling Protocol GTP-C protocol.

On 4G/LTE networks the functionality of the GGSN has been replaced by the Serving Gateway (SGW) and the PDN Gateway (PGW).

Before reaching the Internet the data traffic passes through the Policy and Charging Enforcement Function (PCEF), which is quite often integrated within the GGSN or the PGW. It can block the traffic when the user has exceeded the data quota and may also be equipped with Deep Packet Inspection (DPI) functionality.

The Domain Name System (DNS) is a hierarchical decentralized naming system for computers, services, or any resource connected to the Internet or a private network, which allows the translation of human recognizable domain names into numeric IP addresses and enables servers and computers to look up and communicate with each other. The DNS is defined by the IETF (Internet Engineering Task Force) RFC (Request for Comments) 1034 [1] and RFC 1035 [2].

DNS tunneling in the mobile network in the same way generic DNS tunneling in IP network exploits the fact that most operators allows all DNS traffic out and also through port 53 without charging to set up a tunnel for IP traffic bypassing firewall and charging functions.

In fact, it is quite simple to bypass the PCEF by establishing a modified name server on the internet and by creating a special client that is capable of encoding information in the DNS packets. As shown in Figure 2 the client might send a chunk of data as an “A” or “AAAA” record which may look something like this “nslookup VGhlIHgbWFrZSB1cCB0aGUgNjQgY2hhcmFjdGVycyByZXF1aXJlZCBmb3IgYmFzZ.myDNSTunnel.com" which may encapsulate personal information about John Doe as shown in Figure 2.

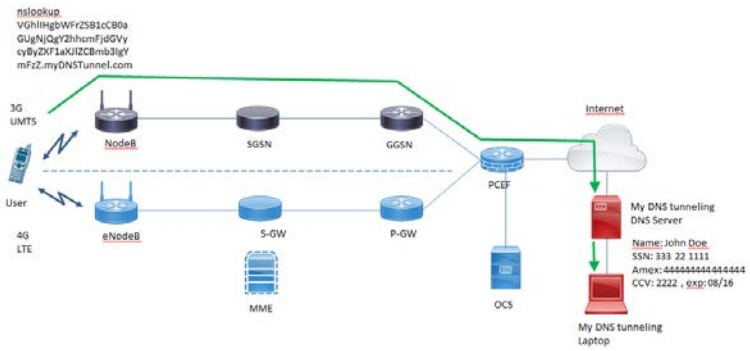

Figure 2 A simplified DNS tunneling in mobile network

Once this query arrives at the modified DNS server, the server can send any data that is waiting for the client by responding to the A query with a CNAME record - "CNAME:JllHRyYWRpdGlvbmFsbHkgbm9NsZWFuLiBGb3IgZ.myDNSTunnel.com".

Another way to do this involves using DNS TXT or EDNS type records, which allow large unstructured strings to be sent. Reverse lookups can also be used to fetch the data responses.

There are currently a variety of DNS tunneling tools for both PC and Android phones using different forms for data encoding such as OzymanDNS [5], Dns2tcp [6], Iodine [7], Heyoka [8], DNSCat [9], MagicTunnel [10], Element53 [11], VPN-over-DNS [12], etc.

4 Challenges in detection of DNS tunneling

In this section, the different DNS tunneling detection techniques are reviewed and their limitations are identified and analyzed. As stated in [13] [14] the detection techniques can be classified into two categories:

Payload analysis: This analysis category can again be divided into sub-categories as follows:

- ***Size of request and response:*** This technique focuses on the size of the request and response, e.g. length of DNS queries and responses [16], ratio of the source and destination bytes [15], size of host name request, etc. The difficulty of this technique type is to find size thresholds that are optimal against all the tunneling methods.
- ***Hostnames entropy:*** This technique is based on the assumption that encoded name have higher entropy than legitimate DNS names. Unfortunately, some tunneling methods do not create high entropy hostnames and some content delivery networks do use hostnames with high entropy to represent some type of information.
- ***Statistical analysis:*** Tunneling can be detected by looking at specific character makeup of DNS names, e.g. percentage of numerical characters in domain names, number of unique characters, percentage of the length of the Longest Meaningful Substring (LMS), number of repeated consonants, etc. The challenge is to determine the threshold value for these specific character makeups. Further, an intelligent DNS tunneling tool will be able to abandon these makeups when their traffic is blocked.
- ***Uncommon record types:*** Not commonly used record types e.g. "TXT" could be used in the detection. Unfortunately, this technique is not decisive.
- ***Specific signatures:*** Each DNS tunneling tool does have specific way of using the attributes in a DNS header, which can be used as signature in the detection. This technique can only be used for known DNS tunneling.

Traffic analysis: This type of analysis considers multiple queries and response pairs over time to detect tunneling:

- ***Volume of DNS traffic per IP address:*** The amount of DNS traffic generated by a specific client IP address [16] can be used in the detection because tunneled data is typically limited to 512 bytes per request and a large number of requests are need for communication. Unfortunately, advanced DNS tunneling tools can spoof the source IP address and spread the requests within a larger range of IP addresses to avoid detection.
- ***Volume of DNS traffic per domain:*** Large amounts of traffic to a specific domain can be used to detect tunneling because DNS tunnel utilities are quite often set up to tunnel the data using a specific domain name. Unfortunately, sophisticated tunneling tools can make use of multiple domain names and hence avoid detection.
- ***Number of hostnames per domain:*** The number of hostnames for a given domain can be an indicator for tunneling. However, it is not trivial to determine the optimal threshold because different tunneling methods have different numbers of hostnames.
- ***Geographic location of DNS server:*** Large amounts of DNS traffic to different parts f\of the world may be an indicator for tunneling. Unfortunately, in the mobile network there will be users coming from all over the word that may have request to NDS resolvers from their country of origin.
- ***Domain history:*** Domain history can also be an indicator for detection of DNS tunneling. By checking when an A record or NS record is added because a domain could be acquired only recently for DNS tunneling. However, not all newly acquired domains are used in tunneling and one cannot trust every old domain.
- **Orphan DNS requests:** Orphan DNS requests can be also used to detect DNS tunneling because they are the ones that do not have a corresponding request by another application such as http. However, orphan DNS requests may be legitimately used by security devices and program for IP address lookups.

In brief, there is so far no DNS tunneling detection that is really satisfactory for mobile network [17].

5 How can machine learning help

As in [18] which proposes the usage of machine learning in the protection of mobile networks, Tom Mitchell' s definition of machine learning [19] is adopted in this paper as follows:

"The field of machine learning is concerned with the question of how to construct computer programs that automatically improve with experience"

He provides also a short formalism as follows:

"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E."

The machine's ability to learn and improve its solutions to problems is hence central in machine learning.

As shown in section 4, the reviewed DNS tunneling methods do not provide firm criteria but rather non-conclusive indicators. In order to build an efficient DNS tunneling detection it is necessary to use a combination of a large number of both payload and traffic analysis method. Such an approach is not suitable for mobile network because with the huge number of users coming from all over the world it is time consuming and requires huge processing capability.

In this challenging situation, Machine Learning can come to rescue by providing a sound way to define normal behavior of the mobile network when there is no tunneling. Upon the emergence of a DNS tunnel, machine learning techniques will detect anomalies which indicate the presence of the DNS tunnel. At the beginning there will be false positives, i.e. anomalies that are not due the presence of a DNS tunnel but the machine will receive the feedback, learn it and get better for the next time.

For the detection of DNS tunneling, we experiment 2 machine learning methods, namely One Class Support Vector Machine (OCSVM) and K-Means as follows:

One Class Support Vector Machine (OCSVM):

OCSVM [20] belongs to a supervised learning class called Support Vector Machine (SVM) that divides the input spaces into two regions, separated by a linear boundary and classifies input either inlier, i.e. falling into one category such as normal or outlier, i.e. falling outside such as abnormal. Although the training of SVM is performed on both positive and negative data the OCSVM extension makes it possible to use only positive data in the training process.

K-Means

k-means clustering, an unsupervised method [21] aims at partitioning n observations into k clusters in which each observation belongs to the cluster with the nearest mean.

6 DNS tunneling detection using machine learning

In order to test and verify the proposed DNS tunneling detection using machine learning we need DNS traffic data which are both benign and malicious. A testbed is created with four clients: one malicious DNS tunneling client on mobile phone and 3 regular browsers.

- The mobile phone client is realized by an Android 4.4 running on a Virtual Machine (VM) and hosting Slow DNS, a DNS tunneling app.
- The other three clients are realized by a Ubuntu 14.04 running on virtual machines and hosting regular browser.

The 3 browser are regularly accessing the World Wide Web and generating both DNS traffic and http traffic. Before initiating browsing from the mobile phone client a Slow DNS tunnel is established.

All the DNS traffic are then gathered and captured by Wireshark. The DNS packets were filtered out in Wireshark and saved as a comma-separated values (csv) file. Each line in the csv file contained meta data for one packet, with the features No., Time, Source, Destination, Protocol, Length, Info. The raw data had to be reformatted to be used for the machine learning models. To do this a python script is developed and

used, which went through the csv file to find the response to each request and creating a new csv file.

The features in the new file were Time, Source, Destination, Protocol, LengthUp, LengthDown, Info, Label.

The Time feature now was the time between the request and response, not the time since the capture started. Table 1 shows how two lines of the new csv file looks like, one line with regular DNS traffic and one with malicious traffic.

Time	Source	Destination	Protocol	LengthUp	LengthDown	Info	Label
0.021829999999997796	192.168.1.60	192.168.1.1	DNS	89	276	Standard query 0x0ef7 A safebrowsing-cache.google.com	1
0.5176109999999987	192.168.1.14	192.168.1.1	DNS	209	274	Standard query 0x3a8c NULL 149N2546851188122-246-109-MHoQF3dk88nOvvChaPdyeLvknsPKAAAAdgAbh.u4KADQAMgAOAA0AGQALAAwAGAAJAAoAFgAXAAgABgAHABQAFQAEAAUAEgATA.AEAAgADAA8ie.tg16.m7q.in	-1

Table 1 A line with regular DNS traffic and one with malicious traffic

To test and verify the two proposed machine learning methods for DNS detection, SciKit-Learn, a library for Python which contains functions to create machine learning classifiers and support for training and testing is used. The metrics class in SciKit-Learn contains many functions to evaluate classifiers. For an outlier and categorization classification the normal way to determine the success is by measuring precision, recall and F-score. The precision of a classifier determines the percentage of the elements selected that are true positives. The recall determines the percentage of the relevant elements was selected. The F-score measurement is derived from both precision and recall and gives a result which better represents the overall character of the classifier. The closer to 1 the higher are both the precision and the recall; and classifier is working well.

DNS tunneling detection using OCSVM

Since the SciKit-Learn OCSVM has four kernels linear, polynomial, Radial Basis Function (RBF) and sigmoid, experiments are carried out with each kernel.

		precision	recall	f1-score	support
Kernel = rbf	Outlier	0.40	1.00	0.57	1124
	Inlier	1.00	0.51	0.67	3394
	avg / total	0.85	0.63	0.65	4518
Kernel = sigmoid	Outlier	0.25	1.00	0.40	1124
	Inlier	0.00	0.00	0.00	3394
	avg / total	0.06	0.25	0.10	4518
Kernel = linear	Outlier	0.24	0.94	0.38	1124
	Inlier	0.57	0.03	0.05	3394
	avg / total	0.49	0.25	0.14	4518
Kernel = poly	Outlier	0.86	0.94	0.90	1124
	Inlier	0.98	0.95	0.96	3394
	avg / total	0.95	0.95	0.95	4518

Table 2 Classification report for OCSVM with different kernels

As shown in Table 2, the poly kernel has best result. However, after some adjustment of the nu and gamma parameters, the RBF kernel obtains an f1of 96% which is higher than the poly kernel.

DNS tunneling detection using K-Means

The K-means classifier is tested with three different initiation methods, namely k-means++, random and ndarray. The results of the experiment are shown in Table 3.

All the initiation methods have a quite even total f1-score, but a closer look of each line may identify a weakness. With init set to random the model is almost not able to predict any outlier, with recall at 1% and both precision and f1-score at 0%.

		precision	recall	f1-score	support
init = ndarray	Outlier	0.14	0.99	0.24	286
	Inlier	1.00	0.48	0.65	3441
	avg / total	0.93	0.52	0.62	3727
init = k-means++	Outlier	0.14	0.99	0.25	286
	Inlier	1.00	0.50	0.66	3441
	avg / total	0.93	0.53	0.63	3727
init = random	Outlier	0.00	0.01	0.00	286
	Inlier	0.86	0.50	0.64	3441
	avg / total	0.79	0.47	0.59	3727

Table 3 Classification report for K-means models with different initiation

Evaluation

The experiments show that the DNS detection using OCSVM is superior to the one using K-means. This is not surprising since K-means is a cluster classifier and work best when the clusters are even, which is not the case of DNS tunneling where only a minor part of the traffic data is malicious. Further, the experimented data set is too small for K-means.

OCSVM gave great results with the poly kernel with default parameters, and with the RBF kernel when the gamma and nu parameters are tuned. As the poly kernel only seemed to work with the default parameters and with two features from the dataset, it seems to be quite unstable and might not be the best to use in a real implementation. The RBF kernel had a recall of close to 100% on the outliers in most of the tests, which means it was able to categorize all the outliers correctly. This is important for a DNS tunneling detection. The weakness of the method is the precision of outliers and recall of inliers, which means it produces some false positives. By working with the initiation parameter of the model it is possible to reduce the number of false positives down. The OCSVM with RBF kernel is a good method to implement DNS tunneling detection.

7 Conclusion

In this paper, we propose to use machine learning techniques in the detection of DNS tunneling, which so far does not have any really efficient solutions. Two machine learning methods, namely OCSVM and K-means have been selected for the experiments. A testbed able to generate and collect both regular and malicious DNS traffic is established. Experiments have been carried out and the results prove that machine learning is a feasible technique that could be used in the detection of DNS tunneling. However, the efficiency depends heavily on the machine learning method in use and on some degree the fine-tuning of their respective parameters. The experiments have many limitations. First, the dataset used in the experiment is too small and is not representative for the huge DNS traffic in the mobile network. Next, the dataset is generated only by 4 clients, one malicious and three benign and hence does not contain sufficient variations in terms of IP addresses, domains, DNS tunneling meth-

ods, etc. Third, the performance and the scalability of the detection have not been evaluated due to the small size of the dataset. As further work, it is quite interesting to carry out the experiments using real DNS traffic data both benign and malicious collected from the Telenor mobile networks. It might be also quite relevant to make use of a machine learning technique called the Deep Learning Auto-Encoder (DL-AE) [23].

8 References

1. IETF: RFC 1034 Domain names – concepts and facilities, Internet standard, Nov 1987
2. IETF: RFC 1035 Domain names - Implementation and specification - Internet standard, Nov 1987
3. Pure Hacking: Reverse DNS Tunneling – Staged Loading Shellcode, Ty Miller, Blackhat 2008
4. Ayaya: Black Ops of DNS, Dan Kaminsky, Blackhat 2004.
5. OzymanDNS – Dan Kaminsky, 2004 - https://dankaminsky.com/2004/07/29/51/
6. Dns2tcp - Hervé Schauer Consultants - http://www.hsc.fr/ressources/outils/dns2tcp/
7. Iodine - http://code.kryo.se/iodine/
8. Heyoka - http://heyoka.sourceforge.net/
9. DNScat - http://tadek.pietraszek.org/projects/DNScat/
10. MagicTunnel - http://www.magictunnel.net/
11. Element53 – Sander Nijhof - https://nijhof.biz/element53/
12. VPN over DNS - https://www.vpnoverdns.com/
13. SANS Institute: Data Charging Bypass - How your IDS can help, Hassan Mourad, Sept 2014.
14. SANS Institute: Detecting DNS Tunneling, Greg Farnham, Feb 2013.
15. Bianco, D.: A traffic-analysis approach to detecting dns tunnels. (2006, May 3) Retrieved from http://blog.vorant.com/2006/05/traffic-analysis-approach-to-detecting.html
16. Pietraszek, T.: Dnscat. (2004, October 31)Retrieved from http://tadek.pietraszek.org/projects/DNScat/
17. Heavy Reading: DNS Security for Service Providers: An Active Approach at L7 – White Paper – Patrick Donegan, Oct 2015
18. Do, van Thuan, Engelstad, Paal, Feng, Boning & Do, Thanh van: Strengthening mobile network security using machine learning, LNCS –13th International Conference on Mobile Web and Intelligent Information Systems, Vienna, Austria, 22-24 August 2016 – ISBN 978-3-319-44214-3, ISBN 978-3-319-44215-0 (ebook), pp 173-183.
19. Mitchell, Tom M.: Machine Learning, ISBN-0-47-042807-7, Mcgraw-Hill Companies, Inc. 1997
20. Larry M Manevitz and Malik Yousef. One-class svms for document classification.- the Journal of machine Learning research, 2:139–154, 2002
21. MacQueen, J. B. (1967). Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press. pp. 281–297. MR 0214227. Zbl 0214.46201. Retrieved 2009-04-07.
22. SlowDNS: a free VPN over DNS Tunneling Tool - http://slowdns.com/
23. Bengio, Y. (2009). "Learning Deep Architectures for AI" (PDF). Foundations and Trends in Machine Learning 2. doi:10.1561/2200000006.