www.czasopisma.pan.pl PAN POLSKA AKADEMIA NAUK www.journals.pan.pl

INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2016, VOL. 62, NO. 4, PP. 343-346
Manuscript received October 15, 2016; revised November, 2016. DOI: 10.1515/eletel-2016-0047

Network Steganography in the DNS Protocol

Michał Drzymała, Krzysztof Szczypiorski, and Marek Łukasz Urbański

***Abstract*— This paper presents possibility of using of the DNS (Domain Name System) protocol for creating a simplex communication channel between a malware-infected computer with a compromised DNS server. The proposed channel can be used to steal data or confidential enterprise information secretly.**

***Keywords*— DNS, network steganography, malware**

I. Introduction

NETWORK steganography is the family of methods that uses telecommunications protocols as carriers for hidden data. These techniques utilise modifications of the packets to perform covert communication by modification to the structure of the packet (a payload and protocol specific fields) or modification to time relations among packets (like changing the sequence of the packets or inter-packet delays). Due to the fact that the DNS protocol (*Domain Name System*) is commonly used in the Internet, it is a prime candidate for a carrier [5]. The arguments speaking in favour of it include, among others, a great volume of standard packets and considerable problems with network operation when an administrator applies too stringent rules of traffic filtering to this protocol.

With respect to the above, various attempts have been made to use the DNS for purposes other than originally intended. In 2011 Symantec announced the discovery of the W32.Morto bug, which used vulnerability in RDP (*Remote Desktop Protocol*). For communication with the C&C channel (*Command and Control*) it uses TXT records in the DNS, which are dedicated to storing content understandable to a human. W32.Morto sends a query to the DNS server about a TXT record, instead of a typical "IP domain" demand. Next, the returned text is decrypted and processed. In this manner, an electronic signature of the file and an IP address are usually provided, from which an even more malicious malware is downloaded [3].

Another idea for use is channelling between the client's machine and a substituted server, which is designated to provide a response for a previously crafted domain. Thus, it is possible to obtain access to the Internet even in a situation, when the only machine from a local network authorized to do so is a local DNS server. As demonstrated by the research [4], a channel obtained in this manner may reach the bit rate of as much as 1 Mbit/s, with delays of 150 ms. In this case, initial fragments of the URL address (*Uniform Resource Locator*) of the query are used for communication.

All authors are with the Warsaw University of Technology, Warsaw, Poland (e-mail: mdrzyma1@mion.elka.pw.edu.pl, ksz@tele.pw.edu.pl, m.l.urbanski@stud.elka.pw.edu.pl).

A popular idea in network steganography is to use fields of the packet's header. For the protocol in question, this may be an ID identifying the demand and an answer related to it. Lack of appropriate distribution of values of this field turns out to be a problem, when the field is used to carry an encrypted message. Restoration of a pseudorandom character typical for an unmodified demand is examined by researchers in [2].

An interesting idea is also hiding communication in a DDoS attack (*Distributed Denial of Service*), using DNS strengthening [6] (zombie machines generate traffic to a DNS server, but due to a replacement of IP addresses, all answers reach the victim). Information may be hidden thanks to a modification of a Zone file and a TXT-type record in a DNS server controlled by the attacker. Other proposed carriers of hidden communication include distributions of occurrences of domains in queries in a specific period of time, or of types of queries. Detection of atypical communication is more difficult because an administrator of an attacked network will deal with the attack in the first order. What is more, potentially recorded "special" packets will account for a tiny fraction of the traffic followed.

The structure of this paper is as follows: in section 2 we will describe the fundamentals of DNS service and protocol. In section 3 and 4 we will present a steganographic analysis and the model of hiding information in DNS messages. Section 5 will describe a proof of concept, which will be evaluated in Section 6. Finally, Section 7 summarizes the paper.

II. DNS Service and Protocol

DNS is a name of a service, servers of this service and a protocol for exchange of messages between clients and servers providing this service. It allows to change mnemonic (easier to remember) names of domains to IP addresses (of a network layer protocol of the ISO/OSI model). It is one of basic services that comprise the operation of the Internet today.

In order to identify a potential vulnerability of a DNS server, an analysis of formats and scenarios of exchange of messages between servers and the client has been performed [7]. This allowed to identify several potential options to hide information.

A. Format of messages

The format of DNS messages is constant, irrespective of the demand type. A message carrying an answer to a question is bigger because it uses more fields than the question. The fields for which it has not been specified otherwise are of variable length, calculated or determined elsewhere.

1) *Header* – a classic header, contains basic information allowing to send and identify messages.
2) *Question* – stores queries to the name server.
3) *Answer* – stores records of answers to queries.
4) *Authority* – indicates authority servers for a domain.
5) *Additional* – dedicated for additional information.

B. Format of header field

A header of a message contains many fields, including fields marked collectively as 'flags'. The most important elements for the solution presented in this article include:

1) *QR (Query/Response Flag)* – (one bit) has value '0' for queries, it is changed to '1' for answers.
2) *Opcode (Operation Code)* – (4 bits) specifying the query type. This is usually 0; not all 4-bit combinations are used.
3) *QDCount (Question Count)* – (2 bytes) specifies the number of queries sent in a demand.
4) *ANCount (Answer Record Count)* – (2 bytes) specifies the number of answer records. In DNS queries, the '0' value is not forced.

In the description of the *QR* field, the word "changed" has been used on purpose because the packet of answers contains the query contents in itself (it extends it by completing or modifying the existing fields).

The *count*-type fields have a function of an indicator informing the program interpreting the packet about the amount of records of a given type to be expected. Information about the length of every record (where an indicator for its end byte is calculated) is inside it.

C. Format of an answer field

A question and an answer are formed into structures which facilitate their matching. Their fields include:

1) *Name* – contains the name of the object, zone or domain which identifies the query.
2) *Type* – (2 bytes) contains the type of record. The most popular type is *record A*, that is a query about the IPv4 address of the domain specified in the *Name* field. Respectively, *AAAA* is a query about the IPv6 address.
3) Class – (2 bytes) defines a class of a query, and usually has value '1' that is *IN* (*Internet*).
4) *TTL (Time To Live)* – (2 bytes) specifies at which number of demands real queries should be sent to a DNS server, instead of using previously downloaded data (from the cache memory).
5) *Resource Data Length* – (2 bytes) specifies at what number of bytes the current record ends; this field exists to make it possible to use a common format for various types of record.
6) *Resource Data* – contains data bytes. For instance, for an A-type record (a basic query for a DNS), four bytes containing an IPv4 address are required.

D Exchange of messages

For the needs of the solution presented here, a (largely simplified) scheme of communication with a DNS server may be depicted as follows: a client wishing to find an IP address of a domain, first reaches for the address of the main DNS server recorded in the setup of the web interface. Next, it formulates a query (for instance about an A-type record) and sends it to the address obtained in the previous step. Depending on the type of the query and its content, it may be forwarded to other DNS servers until the answer finally reaches the client.

III. Steganographic Analysis

The theoretical analysis performed has been confirmed with tests carried out by sending standard and prepared queries to a Google DNS server (IP: 8.8.8.8), and by following them in *WireShark*, a program for network traffic monitoring. The following conclusions have been drawn from the analysis:

1) The DNS server processing the queries ignored distorted queries i.e. queries containing header fields completed in a non-standard manner.
2) Rare queries of the reverse query type or with unused *OPCode* distinguish themselves greatly, which adversely affects non-detectability of transmission of additional information.
3) DNS queries (messages with a *QR* flag set up to *0*) may also have responses; such a query is not treated as unprecedented or distorted; in addition, at the arrival of such a complex message, it is correctly interpreted by the server – a prepared answer is simply replaced with a correct one.
4) In the DNS answer structure, the *Resource Data Length* field informs us how many bytes the *Resource Data* field occupies; this value may, however, be predicted, for instance for an answer to the A-type record, this field will always occupy 4 bytes, which is as much as needed to record the IPv4 address; therefore, in the case of a record of this type, the protocol envisages the interpretation of only first four bytes from the *Resource Data* field – the other are ignored, in spite of determining their number in the *Resource Data Length* field.

IV. Steganographic Model

Considering the analysis, among many options the most promising seems to be the following model of hiding information in DNS messages:

1) This is a standard (*Opcode* = 0) query (*QR* = 0) about one domain (*QDCount = 1, Query[1]*).
2) Even though it is a query, the packet contains one answer (*ANCount = 1, Answer[1]*); if more than one answer is placed, the tests have shown that a real DNS server would reject the packets.

3) In the query field there is one question about the IPv4 address (*Query[1].Type = 2*) of the existing domain (for instance: *Query[1].Name = kstit2016.iitis.pl*).
4) The *Answer[1].TTL* field serves to number the sent messages. It is able to address 32,768 messages (2 bytes).
5) In the *Answer[1].IP* field the correct IPv4 address is placed, which would be provided by the DNS server to such a query.
6) **After the *Answer[1].IP* field (where, in line with the protocol specification, there should be no data) confidential information is placed.**
7) The *Answer[1].ResourceDataLength* field contains correct information on data length (4 bytes to the prepared IPv4 address + length of confidential data).

V. Proof of Concept

A Malware

The concept explained above was proven right during the implementation of malware. It contains a setup file in which the following items are defined:

1) *DnsAddr* – A public IPv4 address of a compromised DNS server – this is where all DNS queries (even the true ones) from an infected computer will go.
2) *FilePointer* – A name or path to a file, which is to be secretly sent.
3) *ChuckSize* – Maximum size of a single DNS message – when an indicated file exceeds this size, it will be cut into pieces; files with a maximum size of 32768 * *ChuckSize* are allowed, because this is the maximum amount of messages which may be sent within one session (one malware launch).
4) *IpDnsList* – Prepared list of pairs (IP address – DNS name) serving to prepare queries. In every DNS demand in which information is hidden, there is a question about a certain domain from this list, and the answer contains the IP address of this domain – in this way, the answer resembles more the one with an *Opcode = 1* code. This is another form of a security measure in the case of an analysis of DNS queries against suspicious or uncommon parameters, such as an IP address occurring too often or a private address occurring where a public one should.
5) *WaitTime* – A maximum limit of time which may elapse between sending one packet of data and another.

B From the perspective of the infected computer

1) A malware process reading start parameters from a setup file is launched on an infected computer.
2) The process replaces the address of a systemic DNS server with the one specified as a parameter, previously saving the original address.
3) The process finds the file with the required name and divides it into pieces, if necessary, and then begins sending it (generating queries with answers, using the *IpDnsList file*).
4) Sending chunks of data is randomized – before sending a consecutive packet, the malware waits for a random time from the range from 0 to *WaitTime*.

C From the perspective of the compromised server

1) A prepared DNS server process is launched.
2) It captures all the queries, which reach it and divides them into the ones, which contain hidden information and those which do not.
3) From those with information hidden data are extracted and then they are treated like other demands. These packets are not filtered out because this could be detected on an infected computer – a large number of demands without answers would occur.
4) Those without hidden data are forwarded to the real DNS server (DNS Google with the address: 8.8.8.8) with the source address replaced with the address of the compromised server; the answers to the demands returning to the server taken over are then directed to the infected computer; from its perspective it looks as if the (compromised) server under query were a real DNS server – unnoticeable delays are introduced, and all source and target addresses are set up so that they do not betray any suspicious activity.

Thanks to such a realization of a client and server application, prepared queries are very difficult to detect. Except from the fact that they contain an answer (and constantly *ANCount* is set to 1), they bear no difference from other DNS queries sent from the infected computer. Another asset is completely correct answers to prepared queries, which may also mislead a person attempting to detect suspicious traffic.

D Implementation details

The malware application has been written on the *.NET* platform in *C#*, using the *Pcap.NET* library. Such a choice was due to a good integration of the platform with the Windows operating systems, which facilitated processing of system calls.

The DNS server application was written in *Java* from the scratch due to simple network management and multithreading support. It managed a large number of queries very well, both prepared and standard ones, without introducing any noticeable delay.

The entire testing environment was launched on virtual machines, under *Windows 7* operating system control, thanks to the *Hyper-V* solution by Microsoft. Despite of a virtualization layer, the environment ran very smoothly and it allowed to perform the tests mentioned above.

The tests were conducted inside a local network under control of one router. The malware set up the router's public address as the DNS server address on the infected computer. The router was set up so that all the DNS queries (UDP packets to port 53), which reached it were directed to the

address of a substituted server on another machine inside the network. Thus, the Internet's impact on the solution's behaviour was minimized. The exchange of packets is depicted in Fig. 1.

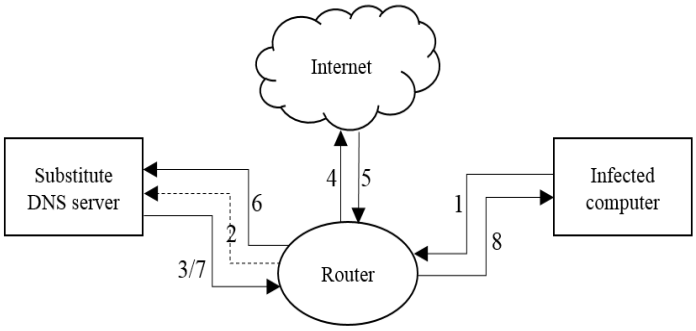

Fig. 1. Scheme of DNS message exchange: The infected computer sends a query to its DNS (1). The router directs such a query to the address of a substituted DNS server (2). The latter forwards the query to the real DNS server and waits for an answer (3, 4). The response to the request is returned (5, 6) and is directed to the original demand author (7, 8). All the steps consider the right replacement of source and target addresses and ports.

VI. Evaluation

The presented solution has been tested with the use of the following scenario (repeated 10 times):

1) A computer user browses the Internet – s/he goes to a website which s/he is interested in, makes him/herself familiar with the content presented there (and on several webpages) and then goes to another website.
2) In the course of his/her activity, malware process is launched, which sends an *s-packet* (that is a packet containing a prepared query) to a compromised server every second, on average.
3) The entire traffic is monitored by *WireShark*.

The averaged data from measurement results are presented in Table I (averaged for 10 measurements).

TABLE I
AVERAGED RESULTS OF MEASUREMENTS

Parameter	*Value*
Measurement time	600 s
Number of all packets	254621
Number of DNS packets	4838
Number of *s-packets*	590
Share of DNS packets in all packets	1.9 %
Share of *s-packets* in all packets	0.23 %
Share of *s-packets* in DNS packets	12.2 %
Volume of data hidden in one s-packet	30 B
Steganographic bit rate	29.5 B
Volume of information sent during measurement	17.3 kB

VII. Summary

Thanks to the research conducted, it was possible to find a steganographic method, which, according to the authors, is a golden mean in a triangle proposed by Jessica Fridrich [1]:

1) It ensures a very good steganographic bit rate – thanks to a large number of DNS queries, it is easy to blend into the crowd; in addition, it is possible to easily regulate the speed of sending information by introducing additional delays.
2) It ensures satisfactory undetectability – without advanced filters and dedicated software to follow anomalies in the network, it is virtually undetectable.
3) It ensures satisfactory resistance to modification – without a rule which will monitor a particular set of parameters (the number of answers vs *QR* field), packets with data may be subject to any modifications which may be applied to standard DNS packets at the attempt to detect or prevent steganography.

Furthermore, the unique method is easy to modify to obtain a two-direction communication – it is enough to cyclically send queries for instructions to the compromised server, which will be sending them in answers.

References

[1] Fridrich, J., "Applications of Data Hiding in Digital Images". Tutorial for The ISSPA'99, Brisbane, Australia (August 22-25, 1999).
[2] Altalhi A. H., Ngadi M. A., Omar S. N., Sidek Z. M., "DNS ID Covert Channel based on Lower Bound Steganography for Normal DNS ID Distribution". International Journal of Computer Science Issues (IJCSI), 8(6), 2011.
[3] Mazurczyk, W., Wendzel, S., Zander, S., Houmansadr., A., Szczypiorski, K., "Information Hiding in Communication Networks: Fundamentals, Mechanisms, Applications, and Countermeasures", Wiley-IEEE Press; 1 edition, February 2016.
[4] Van Leijenhorst, T., Kwan-Wu. C., Lowe, D., "On the viability and performance of DNS tunneling". The 5th International Conference on Information Technology and Applications (ICITA 2008), Cairns, Australia, (June 23-26 2008).
[5] Zielińska, E., Mazurczyk, W., and Szczypiorski, K. (2014). Trends in steganography. Communications of the ACM, 57(3), 86-95.
[6] Mehic, M., Voznak M., Safarik J., Partila P., Mikulec M.. 2014. "Using DNS amplification DDoS attack for hiding data". Proc. SPIE 9120, Mobile Multimedia/Image Processing, Security, and Applications 2014, 91200R (May 22 2014).
[7] The TCP/IP Guide. Accessed on: 11th on June 2016. http://www.tcpipguide.com/.