VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

Fakulta informačních technologií
Faculty of Information Technology

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

Brno, 2016 Rafael Ortiz Cáceres

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

DETEKCE DNS TUNELOVÁNÉHO PROVOZU

DETECTING DNS TUNNELING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Rafael Ortiz Cáceres
AUTHOR

VEDOUCÍ PRÁCE — Ondřej Ryšavý
SUPERVISOR

BRNO 2016

Abstrakt

Tato práce se zabývá problematikou detekce tunelovaného provozu. V práci je pospán princip tunelování provozu a uveden způsob tunelování provozu pomocí protokolu DNS. Dále je uvedena metoda pro detekci takového provozu a tato metoda je testována na několika příkladech.

Abstract

This BSc Thesis was performed during a study stay at the Faculty of Information Technology of the Brno University of Technology. This report discuss the technique of tunnels on the Internet, starting with description of virtual private networks, tunnels over internet with different protocols, focusing the DNS protocol and tunnels through the DNS protocol, The methods to detect the DNS tunnels is presented and demonstrated on several examples and tests.

Klíčová slova

Tunelovaný provoz, bezpečnost, filtrování provozu, DNS.

Keywords

Tunneling, network security, traffic filtering, DNS.

Citace

Rafael Ortiz Cáceres: DETECTING DNS TUNNELING, bakalářská práce, Brno, FIT VUT v Brně, 2016.

DETECTING DNS TUNNELING

Statement

This BSc Thesis was performed during a study stay at the Faculty of Information Technology of the Brno University of Technology. I declare that I have worked out this thesis independently under the guidance of my supervisor. I provided a complete list of all references that I cited within my report.

.....
Rafael Ortiz Cáceres
17th May, 2016

Index

1. Introduction 3
1.1 Coverts channels and covert information 3
1.2 What is tunneling? 4
1.3 Others kinds of tunneling, HTTP tunneling 6
1.4 HTTP VPN Tunnels 9
1.5 Secure Socket Layer 9
1.6 Secure Shell 10
1.7 DNS tunneling 12
1.8 Encoding and Techniques 12
Base32 Encoding 13
Base64 Encoding 13
Binary (8 bit) Encoding 13
NetBIOS Encoding 13
Hex Encoding 14
1.9 Uses of DNS tunneling 14
1.10 Known DNS tunneling applications 16
DeNiSe 16
DNS2tcp 16
DNScapy 16
DNScat (DNScat-P) 16
DNScat (DNScat-B) 16
Heyoka 16
Iodine 17
NSTX 17
OzymanDNS 17
Psudp 17
Squeeza 17
Tcp-over- DNS 17
Tuns 17
Malware using DNS 18
2. Security measures 19
2.1 Length of messages 22
2.2 In-depth analysis of DNS messages 23

2.3 Analyze network traffic 25
3. Experiments 28
3.1 Requirements to create a DNS tunnel with Iodine 29
3.2 How to install Iodine 29
4. Detecting DNS tunneling 42
4.1 Size and quantity of DNS packets and network traffic analysis 43
4.2 Analyze in depth DNS packets 52
5. Discussion and Conclusions 56
5.1 IP control and ports 56
5.2 Controlling access as admin or root 57
5.3 Recursion control the DNS server. 58
5.4 Conclusions 59

1. Introduction

1.1 Coverts channels and covert information

There are many methods and ways to create covert communication channels between 2 or more computers, in which the main purpose is to transmit hidden or encrypted information.
This document will explain some of them, but mainly focus on explaining the technique called specifically communication tunnels and explains in detail the type DNS tunnel.
In a network where communication is based on the IP protocol, you can use IP packet fields that are not normally used for communication, but they exist because at times they must be used.
For example Kundur suggests using unused bits header of an IP packet, or do not fragment bit (DF), the bits used to fragment packets in the communication, to create a covert channel.

The DF bit can be set to arbitrary values if the sender knows the Maximum Transfer Unit (MTU) size of the path to the receiver and only sends packets of less than MTU size.
Hintz suggests using bits of TCP Urgent Pointer (used to Indicate high priority data), that is unused if the URG bit is not set, to transmit hidden data.
Also header extensions and padding can be used for transmit covert information. Many protocols support extension of the standard header. Usually there are some pre-defined header extensions that allow transporting non-mandatory information on demand, but many protocols also allow header extensions to carry data not foreseen in the original specification, extending the capabilities of the protocol. Then covert information can be encoded in frame or packet padding. For example, Ethernet frames must be padded to a minimum length of 60 bytes. If the protocol standard does not enforce specific values for the padding bytes, any data can be used. Padding of the IP and TCP header to 4-byte boundaries (in case header options are present) and padding in IPv6 can also be used to transmit covert data.

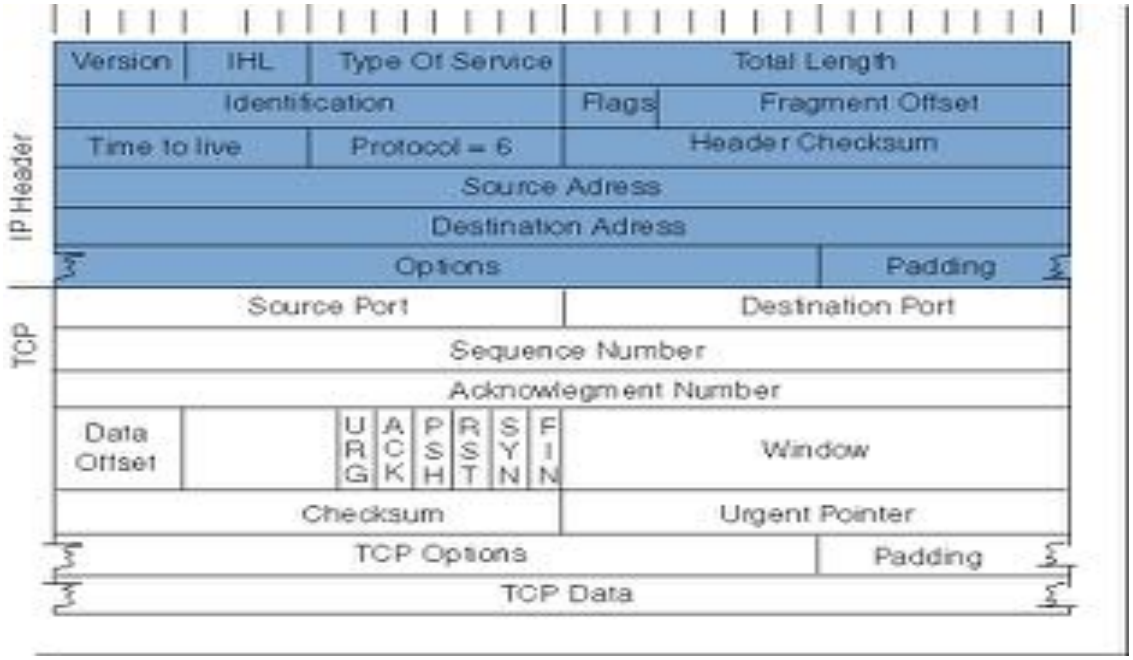

Figure 1.1: Format TCP/IP packet

http://www.cisco.com/c/dam/en_us/about/ac123/ac147/downloads/customer/internetprotocoljournal/ipj_3-2/images/figure01.gif

1.2 What is tunneling?

The tunneling technique involves encapsulating a network protocol over another (network protocol encapsulator) creating a tunnel within a computer network (internet).

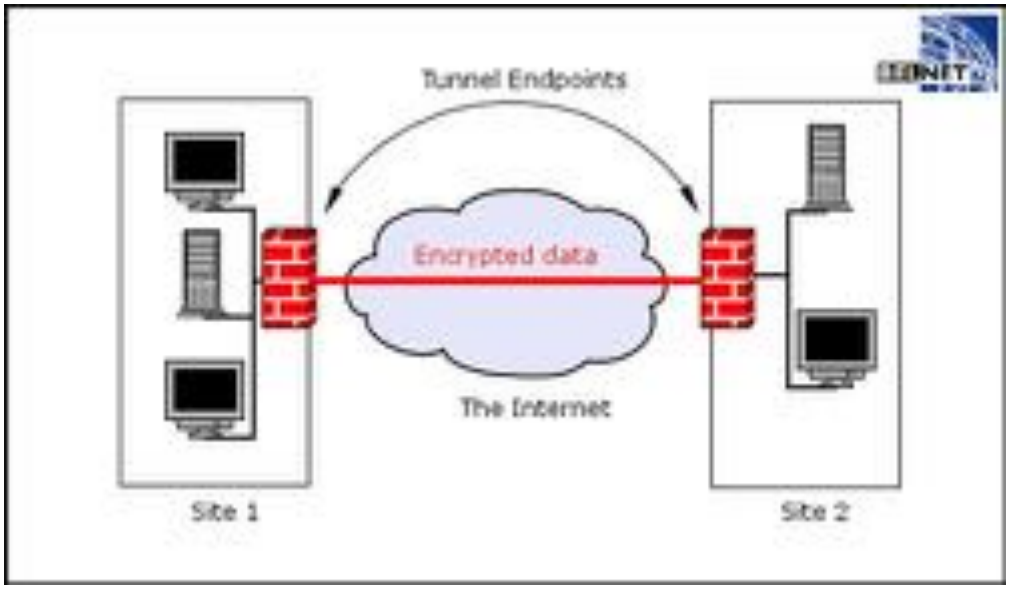

Figure 1.2: Tunnel over internet

https://toic.org/media/filer_public/1b/b4/1bb407b5-0aee-4081-8511-d2d66eb75abb/solution1-2.png

To establish a tunnel includes a PDU within a PDU, in its payload field, so that makes a communication between the two ends of the tunnel, without requiring that any intermediary interpret the PDU encapsulated within the other.

In essence, it is the transfer of a packet of information within another package that makes "wrapper".

A packet is encapsulated with another protocol, so that this is just data and is checked only by the sender and the receiver.

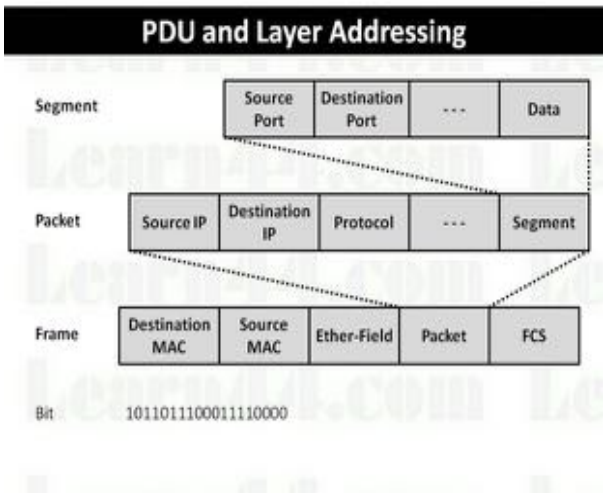

Figure 1.3: PDU and Layer addressing

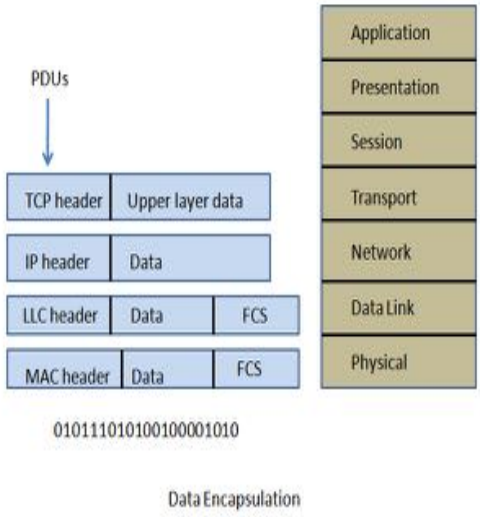

Figure 1.4: Data encapsulation model OSI

http://www.learn44.com/wp-content/uploads/2013/06/Protocol-Data-Unit-PDU-and-Layer-Addressing-in-Data-Encapsulation-Cisco-Inter-networking.jpg

Thus the intermediate nodes in the communication cannot clearly see the contents of the packet, payload, which is routed through them.

The tunnel is defined by the endpoints and the communication protocol used, among others, may be SSH.

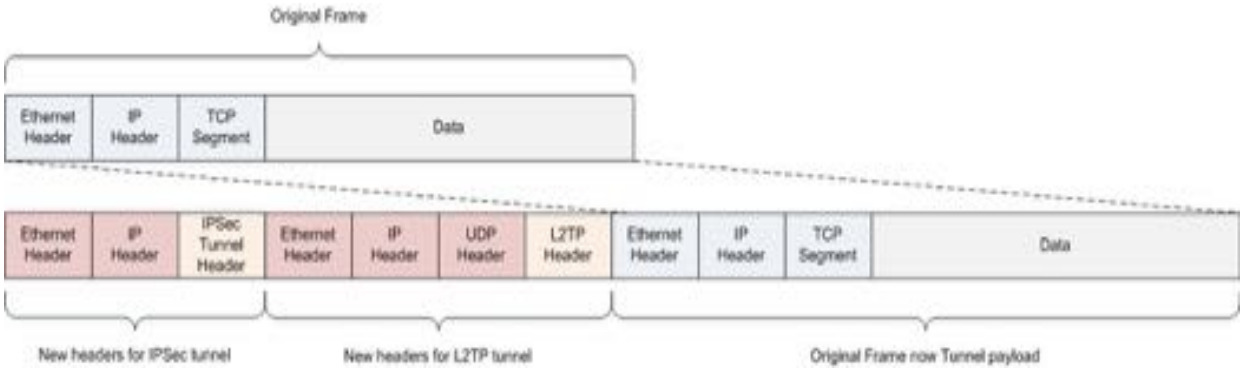

Figure 1.5 Frame encapsulate in tunnel.

https://infrastructureadventures.files.wordpress.com/2010/12/l2tp-in-ipsec1.png

For what is used tunneling?

The tunneling technique is often used to convey a particular protocol through a network that under normal conditions would not be accepted. Another use of tunneling protocols is the creation of various types of virtual private networks. The tunneling technique can also be used to avoid or circumvent a firewall. To do this, the protocol blocked at the firewall, is encapsulated within a permitted protocol, usually HTTP.

Tunneling is a technique also used for IPv6 sites to communicate through an existing IPv4 network. Overlay tunneling encapsulates IPv6 packets in IPv4 packets for delivery across an IPv4 infrastructure. This is similar to how you create a generic routing encapsulation (GRE) tunnel to transport Internetwork Packet Exchange (IPX) traffic through an IP network. At the tunnel head end, an IPv6 packet is encapsulated into IPv4 packet and sent to the remote tunnel destination. This is where the IPv4 packet header is stripped, and the original IPv6 packet is forwarded further into an IPv6 cloud.

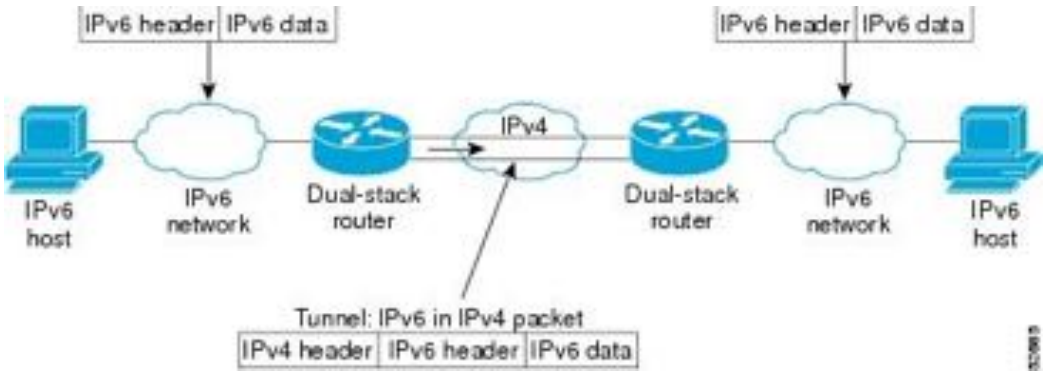

Figure 1.6: Tunnel IPv6 over IPv4

http://www.cisco.com/c/dam/en/us/td/i/000001-100000/50001-55000/52501-53000/52685.ps/_jcr_content/renditions/52685.jpg

Some of the techniques most used in tunneling are HTTP and SSH tunneling, IPv6 over IPv4, mentioned above, and VPN that can be created with different protocols, PPTP, L2TP, IPsec or SSTP, depending on the most important needs of the VPN, such as, speed, security, and ease of installation.

Kaminsky and Gil independently implemented tools to covertly tunnel IP packets over the DNS protocol. Communication takes place between a client and a fake DNS server. The client sends data to the server in DNS requests (hostname lookups) where the actual hostnames are the encoded covert information. The server sends data back to the client contained in the DNS responses.

1.3 Others kinds of tunneling, HTTP tunneling

Proxies do prevent direct inbound access to a protected machine. However, they can be bypassed using HTTP tunnels. Using HTTP tunnels an individual can create arbitrary connection into or out of

a protected network. All that is needed to tunnel through a proxy is an individual inside the network with basic web browsing access.

One technology that makes bypassing HTTP proxies so effective is encryption. While encrypting network traffic offers the client and server privacy and security for their communications, the lack of inspection can reduce the overall security for the network environment it is used in. When encapsulated network traffic is encrypted and passed though a HTTP tunnel, network connections can be created that allow arbitrary, bidirectional connections to remote destinations. When encryption technologies are used in this manner, Virtual Private Networks or VPNs can be created between internal and external machines. The VPN pushes the network perimeter of the protected network beyond the firewall, router or other network security device. This opens the protected network up to the possibility of attack or misuse SSH tunneling.

Simple HTTP tunnels are an unencrypted connection through a HTTP proxy to an arbitrary destination. The tunnel takes advantage of the HTTP CONNECT method normally used for HTTPS (secure web traffic) to connect to the destination server. A typical HTTPS connection through a proxy should look like

```
CONNECT remote-server:443 HTTP/1.0
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 4.0)
Host: remote-server
Content-Length: 0
Proxy-Connection: Keep-Alive
Pragma: no-cache
```

In the example above, a tunnel is established between the client and the remote server with a destination port of 443 or the standard SSL port. If someone wanted to make a connection to another-server on any port all that is needed is to send the following connection request instead.

```
CONNECT another-server:anyport HTTP/1.0
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 4.0)
Host: another-server
Content-Length: 0
Proxy-Connection: Keep-Alive
Pragma: no-cache
```

As shown in the above example, HTTP tunnels are not restricted to web or SSL ports. Rather, HTTP tunnels are capable of passing any outbound traffic on any TCP port as long as the client warps the appropriate HTTP CONNECT header around the data stream. Simple tunnels typically do not require control of the destination server. All that is needed is a remote server with a known listening service. Put another way, the server does not have to be modified in any way to accept a TCP connection that passes through a simple HTTP tunnel.

While the server does not need modification, some work will need to be done on the client side to properly wrap the connection with the HTTP header. The client application may have proxy support built-in and is able to directly create the tunnel. However, a bridging application may be used to allow unmodified applications to pass through the proxy.

While simple tunnels are very useful and are the basic component of all the other more advanced tunnels, they do have their limitations. One limitation to simple tunnels is that each connection to a remote TCP port requires a separate tunnel. Another limitation is that they do not encrypt the connection and pass the data in the clear. If the data needs to be encrypted, it is up to the application to only pass encrypted data through the tunnel. For example, using a simple tunnel to check a pop mail account over the Internet would pass all mail messages in the clear over the Internet. To protect the contents of the mail messages an advanced tunnel employing some form of encryption would need to be employed.

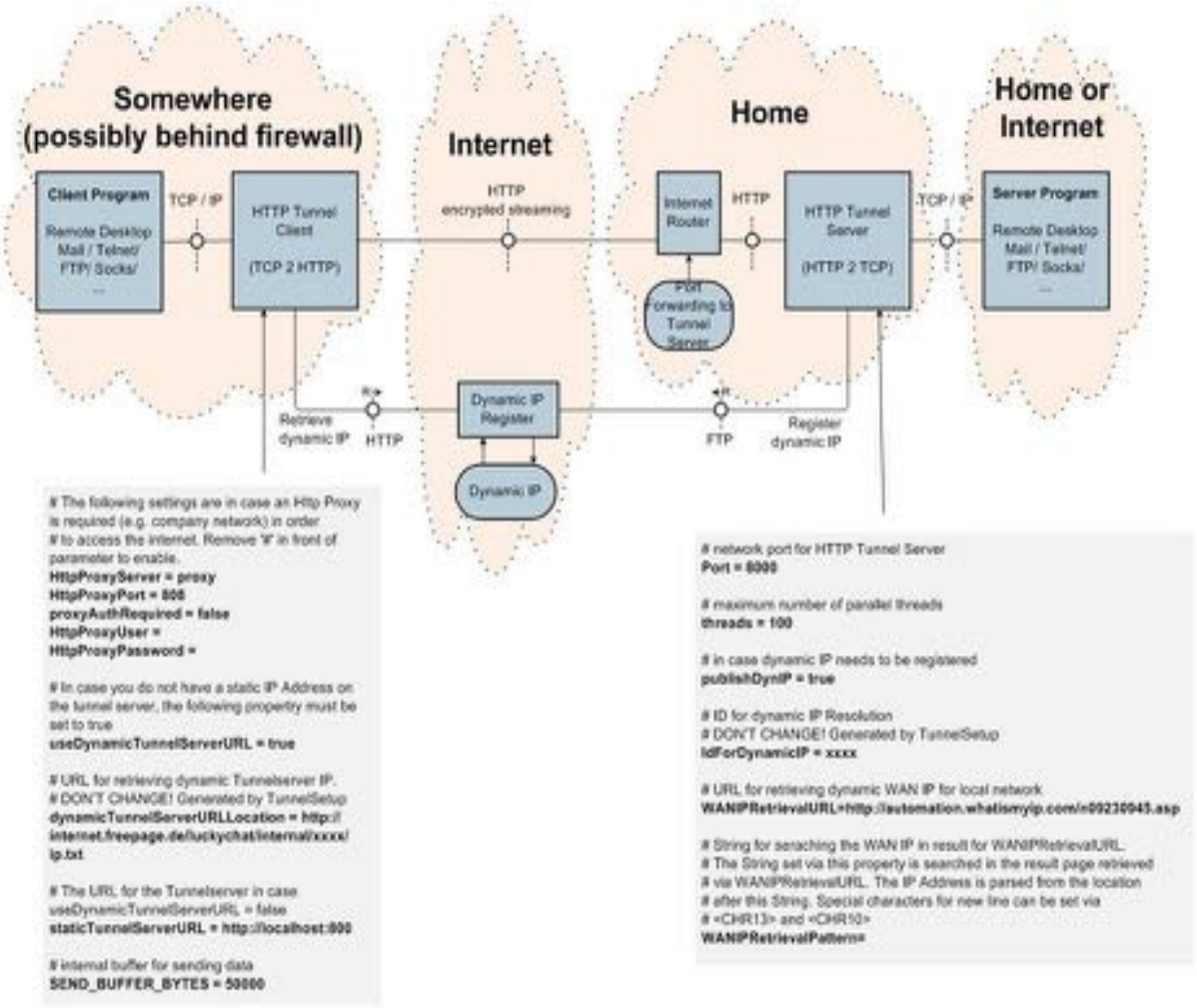

Figure 1.7: Tunnel HTTP

http://www.zenz-solutions.de/assets/img/personalhttptunnel.jpg

1.4 HTTP VPN Tunnels

VPNs or Virtual Private Networks are connection between systems over public networks, such as the Internet, that uses encryption methods to ensure privacy. VPNs can refer to single port to port encrypted communications or to protocols such as IPSEC that are designed to encapsulate all traffic between the two systems. A HTTP VPN tunnel is where any VPN technique is uses in conjunction with a simple HTTP tunnel. VPN tunnels have several advantages over simple tunnels. The biggest advantage is that the communications between the systems are encrypted. This prevents anyone whom intercepts the commutations along the network path from being able to decipher the contents. This also protects the traffic from any intrusion detection systems. Another advantage of HTTP VPN tunnels, and the largest risk, is that with VPN tunnels it is possible pass any protocol in either direction, creating a full VPN connection with a remote site.

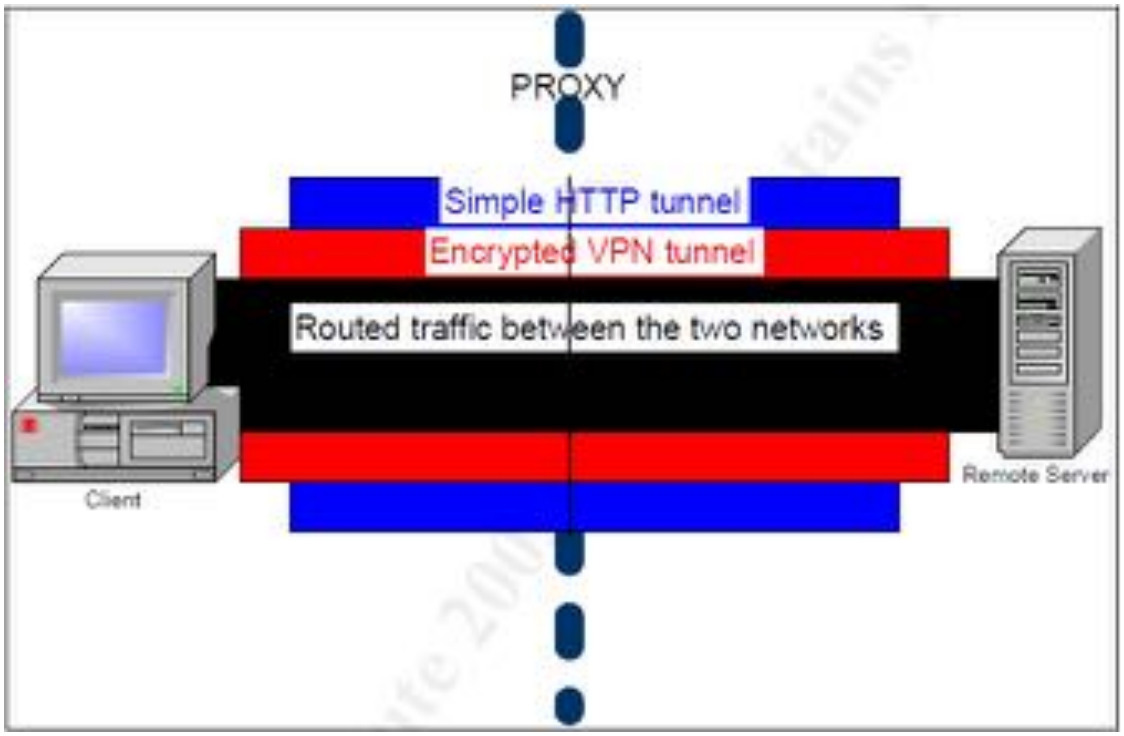

Figure 1.8: Tunnel HTTP VPN

https://www.softether.org/@api/deki/files/250/=4-4-3.png

1.5 Secure Socket Layer

Secure Sockets Layer or SSL "is a protocol developed by Netscape for transmitting private documents via the Internet. SSL works by using a private key to encrypt data that's transferred over the SSL connection " [RFC 6101]. While SSL was developed and is the standard for secure web traffic, a wide range of applications can make use of SSL. Using toolkits like OpenSSL, SSL authentication and encryption can be built into almost any type of application. Furthermore, by using a SSL wrapper program like Stunnel you can encrypt arbitrary TCP connections inside SSL.

1.6 Secure Shell

SSH (Secure Shell) was designed as a secure replacement for the UNIX “r” tools such as rsh (remote shell), rcp (remote copy), and rlogon (remote logon). All three "r" programs require a method for authenticating that you have permission to login or execute programs on the remote machine.SSH requires that the user prove his/her identity to the remote machine using public/ private key pairs, passwords, or hostname and account name information SSH main advantage is that it can “provide secure encrypted communications between two untrusted hosts over an insecure network”. One feature of SSH that goes beyond the “r” tools is its ability to create encrypted TCP tunnels between the local and remote system.

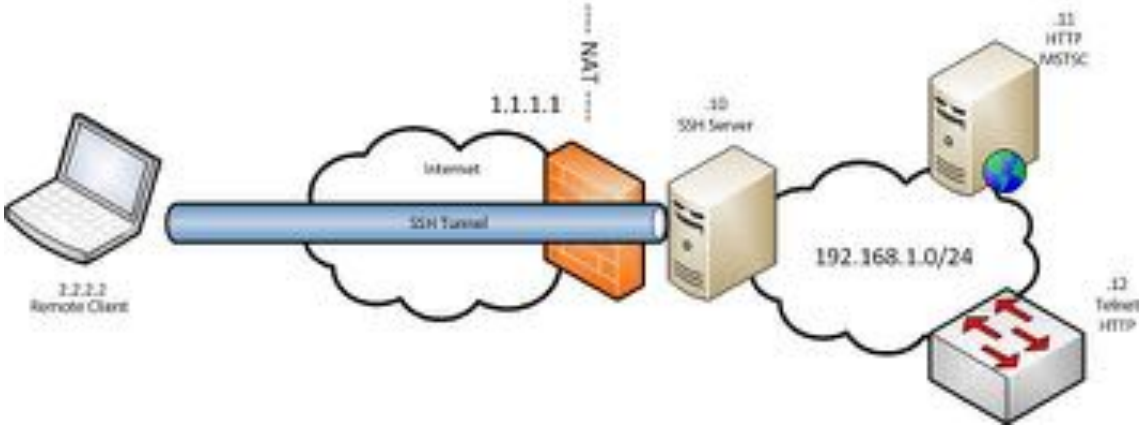

Figure 1.9: SSH tunnel

http://technologyordie.com/wp-content/uploads/2012/07/ssh_tunnel.jpg

Local tunnels listen on the local (client) machine and relay the traffic to the server. The server then delivers the traffic to its final destination. The tunnel can be set to listen on the loopback interface or 127.0.0.1 of the client machine. This is convenient when configuring a locally running application to use the tunnel. However, SSH is not limited to listing on the loopback interface. The tunnel can be set to listen on a specified port on the network IP of the client system, allowing any machine on the local network to leverage the SSH tunnel. The final destination for the tunnel can be the server’s loopback interface, network IP or even a separate system that is reachable by the server system.

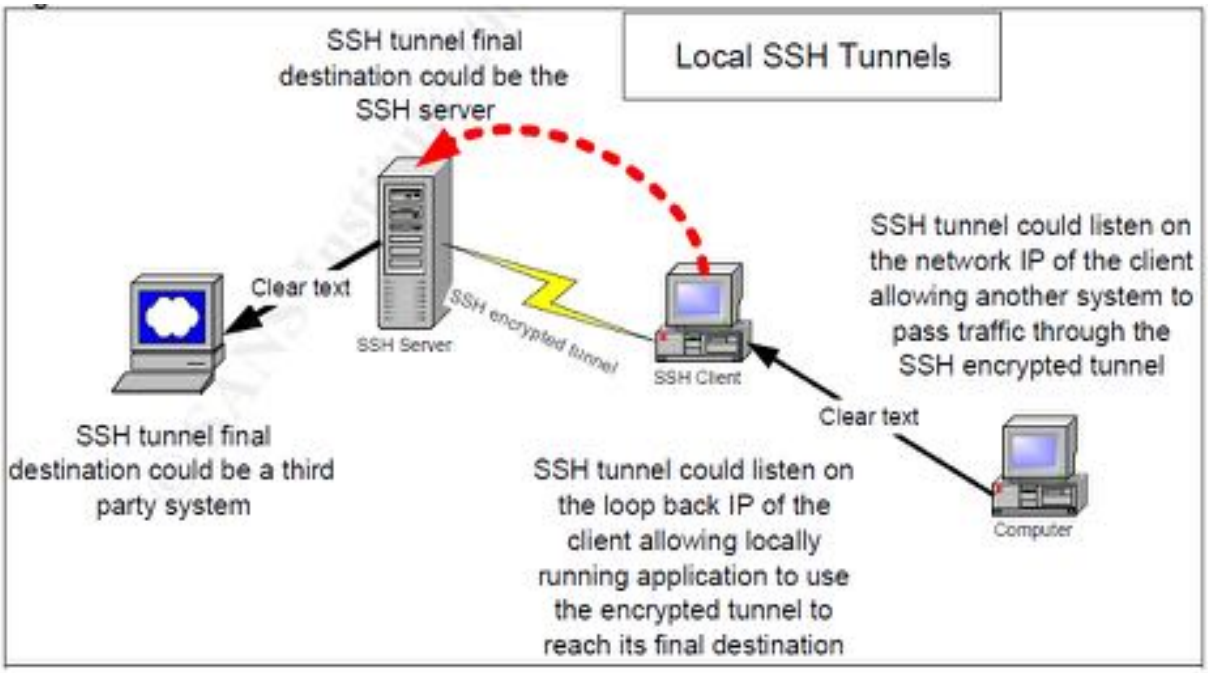

Figure 1.10: SSH tunnel 2

https://chamibuddhika.files.wordpress.com/2012/03/sshsessionforwarding.jpg

Remote tunnels differ from local tunnels by listening on the remote (server) system. Like local tunnels, the remote system can listen on the loopback interface or the network IP and relay traffic to the client machine. The final destination from the tunnel can be the client's loopback interface, network IP or even a separate system that is reachable by the client system.

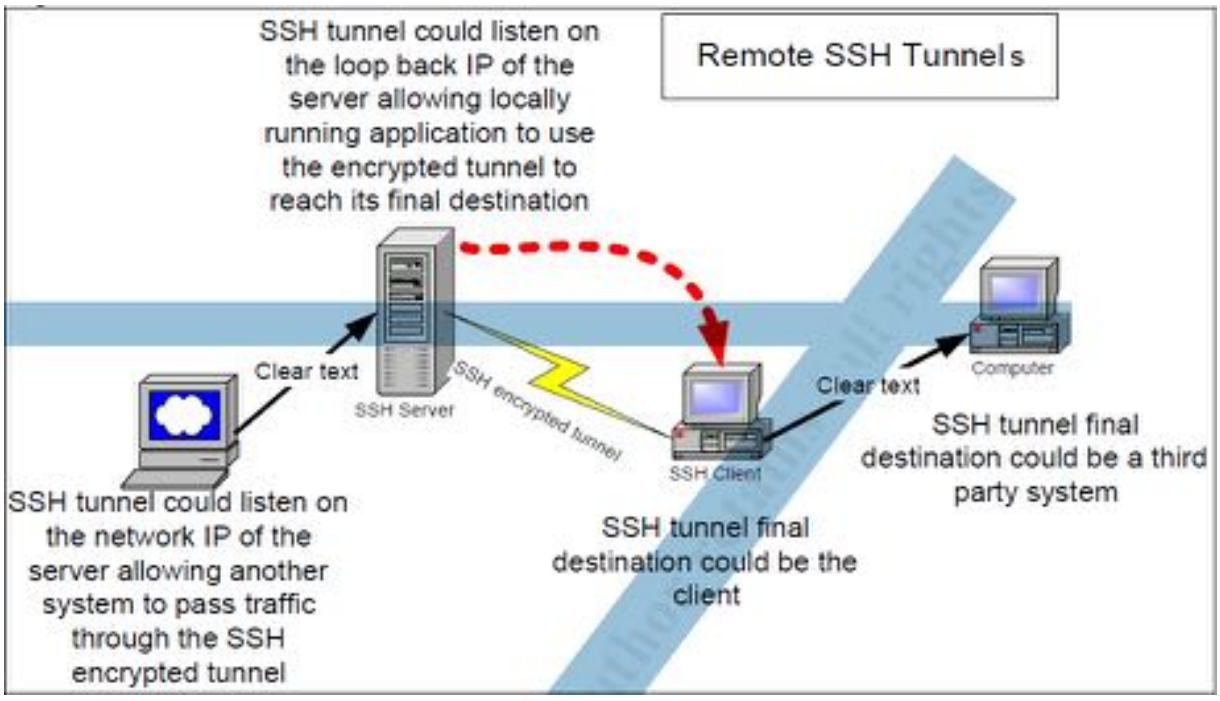

Figure 1.11: Remote SSH tunnel

https://support.zend.com/hc/article_attachments/201327426/debugging_through_tunnel.png

1.7 DNS tunneling

DNS is used to perform a forward lookup to find one or more IP addresses for that domain name. This is known as querying an A record. The user‘s network stack can then send http traffic to the destination IP address. DNS is constantly being enhanced to provide new capabilities. DNS has over 30 record types with many of the common ones being critical to core internet services. As mentioned earlier, the A record type maps a domain name to an IPv4 address. The AAAA record is used to map a domain to an IPv6 address. The CNAME record type is used to map a domain name to the canonical name. The MX record type is used to define mail servers for a domain. The NS record type is used to define authoritative name servers for a domain. The PTR or pointer record is commonly used to map an IP address to its domain name. This is commonly referred to as a reverse lookup. The TXT record type is used to return text data. This record type has been leveraged for specific purposes such as Sender Policy Framework (SPF) for anti-spam. DNS uses both UDP server port 53 and TCP server port 53 for communications. Typically, UDP is used, but TCP will be used for zone transfers or with payloads over 512 bytes.

1.8 Encoding and Techniques

The DNS tunneling utilities can make use of different DNS record types and encoding methods. In some cases, such as iodine, the utility will auto detect the best possible encoding.

The implementation details are where the various DNS tunneling utilities differ. DNS utilities vary in implementation language used, for instance, C, Java, Perl and Python to name a few. Some utilities use a TUN or TAP virtual adapter to create a local interface and IP address for the tunnel on the hosts. Other programs, as Netcat, simply create tunnels, encapsulating the information in any protocol, transmit binary data and can be used to execute commands in a remote machine or transfer files.

The encoding method including DNS record type is an area where tools have been implemented differently. Some utilities use common record types such as A records. Others use experimental types such as Null records and EDNS to improve performance. There is also the Extension Mechanisms for DNS (EDNS). If EDNS is supported by both hosts in DNS communication, then UDP payloads greater than 512 bytes can be used. EDNS is a feature that can be leveraged to improve bandwidth for DNS tunneling. One DNS tunneling utility, Heyoka will spoof the source IP addresses for requests to the server (upstream data) to lower the visibility of the client.

One technique is to encode data in DNS payloads. This is an area where the specifics of each utility vary widely. From a high level simplified point of view, the client wants to send data to server. It will encode that data in the DNS payload. For example, the client could send an A record request where the data is encoded in the hostname:

MRZGS3TLEBWW64TFEBXXMYLMORUW4ZI.t.example.com.

The server could respond with an answer as a CNAME response:

NVWW2IDPOZQWY5DJNZSQ.t.example.com.

In this way any data can be encoded and sent to the server. The server can also respond with any data. Also If there is a need for the server to initiate a communication, it cannot be done directly. Clients do not have a service listening for DNS requests and are typically behind a firewall. Server initiated communication can however be accomplished by having the client regularly poll the server. Then, if the server has data for the client it can send it as a response to the polling requests.

Base32 Encoding

Base32 or 5-bit encoding is commonly used for requests from the client. While DNS names can have upper case and lower case, the case is to be ignored which leaves 26 letters. Additionally, numbers and the ' - ' character are allowed. This provides a total of 37 unique characters. Therefore, we can take data 5 bits at a time which gives us 32 possible values. These 32 values can fit within our 37 available characters. We can then build a string of nested sub domains out of the encoded data. DNS will allow up to 255 characters in total with each sub domain (aka label) being 63 characters or less.

Base64 Encoding

Base64 or 6-bit encoding can be used for TXT record responses. A TXT record can have upper and lower case which provides 52 characters. The numbers add another 10 characters. If we add two additional characters such as - and +, then we have 64 unique values which can be used for base 64 encoding. Similar to the Base32 encoded request, the response can be encoded 6 bits at a time using a TXT response and sent back to the client.

Binary (8 bit) Encoding

Binary 8-bit encoding can be used. The authors of Heyoka [Heyoka] found that although it doesn‘t work with every DNS server, they could successfully use 8 bits per character for encoding which supports greater bandwidth through the tunnel. Additionally, for example Iodine, that is one application for do DNS tunnels, uses Null type records for responses to provide 8 bit encoding.

NetBIOS Encoding

NetBIOS encoding is another method of encoding data that has been used. For NetBIOS encoding, each byte is split in to 4 bit nibbles. Decimal 65 is added to each nibble. Each byte then is encoded in to two characters in a DNS label. This method is only used by DNScat-B

Hex Encoding

Hex encoding is another method of encoding. For hex encoding, the two characters hex values are used to represent each byte. This method is only used by DNScat-B.

1.9 Uses of DNS tunneling

Some of the most common uses of DNS tunnels are to get free Wi-Fi access for sites with a captive portal for http, but free flowing DNS. These tools can also be used for more malicious activities. A DNS tunnel can be used for as a full remote control channel for a compromised internal host. Capabilities include Operating System (OS) commands, file transfers or even a full IP tunnel. For example, data exfiltration via DNS tunneling is a method incorporated in to the squeeza penetration testing tool (Haroon, 2007). It has been shown that DNS tunneling can achieve bandwidth of 110 KB/s (Kilobytes per second) with latency of 150 ms (Van Leijenhorst, 2008).
In order to tunnel data up, so from your client on the protected network to the external server, your machine will encode the data using Base32 (due to the limited character set allowed in the queries). It will then launch a DNS request as follows:

[data].domainname.tld

What happens then, is that your client will request this to the local DNS server. It will contact .tld to obtain the name servers for "domainname.tld", after which it connects to that name server with this request. The name server at "domainname.tld" is of course under your control and instead of interpreting it as a regular query, it will store the data in the form it sees fit. The response it gives in the end, will be the downstream data to be tunneled back to you. This data is encoded using Base64, as the characters allowed in the data part of TXT records (the response) is much more liberal. This way it becomes possible for you to set up a session through DNS you send data up in the form of queries, and get data back in the form of responses. One more example, in this case we would like to tunnel out the command: "The flowers are growing in November" and you receive the response "As they were growing in May".

DNS request:
k2qa6goccrreni9ej3826plc0pp4nnte9jre6g4de38sknpe5br4mi9e.domainname.tld

DNS response:
QXMgdGhleSB3ZXJlIGdyb3dpbmcgaW4gTWF5

Using this same mechanism, not only can innocent messages be tunneled, but a complete IP tunnel can be set up, or large files can be transferred. It just consists of converting the activity you want to complete into this format, afterwards re-interpreting them at the receiving end.

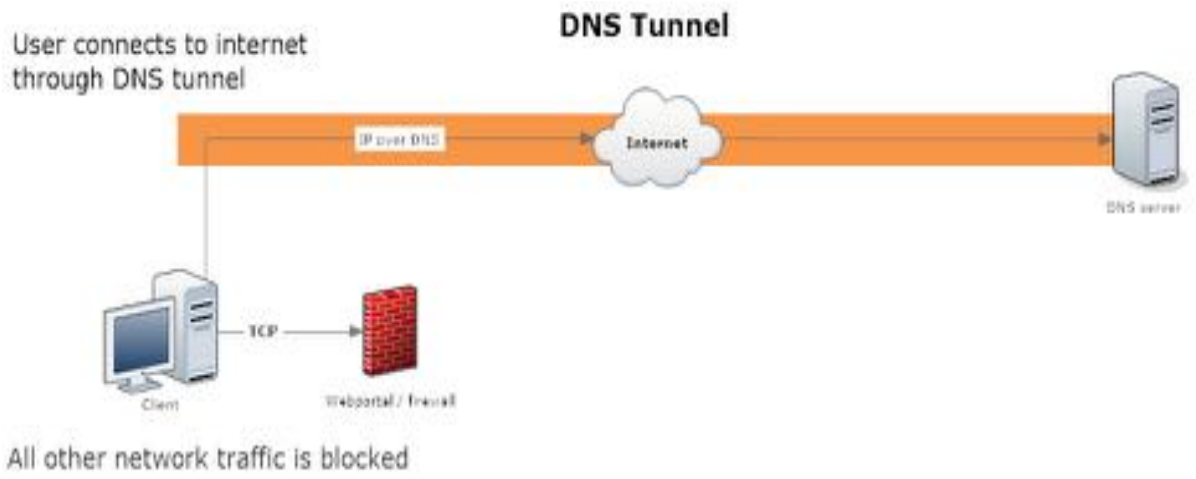

Figure 1.12: DNS tunnel

https://www.blubgoo.com/wp-content/uploads/2014/01/ip_over_dns_tunnel-e1389889549194.png

Some of the DNS tunneling utilities will create a tunnel or a TAP interface locally on the endpoint system. There will also be a TUN or a TAP device on the DNS server hosting the DNS tunneling tool. This will allow the user to tunnel IP traffic to the internet. This technique is similar to how VPN software works such as OpenVPN. There are even commercial service providers that provide the server side tunnel as a service. These services can be marketed as VPN over DNS.

All of the utilities use similar core techniques but have variation on encoding and other implementation details. The core techniques used by all DNS tunneling utilities include a controlled domain or sub domain, a server side component, a client side component and data encoded in DNS payloads. The controlled domain is used to define the authoritative name server for that domain or sub domain. The server side component will be referred to as a DNS tunnel server. The DNS tunnel server will be the authoritative name server for the controlled domain. The DNS tunnel server will typically be an internet accessible server controlled by the tunnel user. The client side component hosts the other end of the tunnel. This could be an endpoint in a security controlled enterprise environment. The tunnel could be used to communicate past the security controls and allow communication between the controlled endpoint and an arbitrary host on the internet. The client side component initiates a DNS request for which the DNS tunneling server is the authoritative name server.

1.10 Known DNS tunneling applications

There are a number of different utilities for DNS tunneling. Some of them are exposed here.

DeNiSe

DeNiSe is a proof of concept for tunneling TCP over DNS in Python. The github page for DeNiSe has six python scripting dating between 2002 and 2006 (mdornseif,2002).

DNS2tcp

dns2tcp was written by Olivier Dembour and Nicolas Collignon. It is written in C and runs on Linux. The client can run on Windows. It supports KEY and TXT request types (Dembour, 2008).

DNScapy

DNScapy was developed by Pierre Bienaime. It uses Scapy for packet generation. DNScapy supports SSH tunneling over DNS including a Socks proxy. It can be configured to use CNAME or TXT records or both randomly (Bienaime, 2011).

DNScat (DNScat-P)

DNScat (DNScat-P) was originally released in 2004 and the most recent version was released in 2005. It was written by Tadeusz Pietraszek. DNScat is presented as a swiss army knife tool with many uses involving bi-directional communication through DNS. DNScat is Java based and runs on Unix like systems. DNScat supports A record and CNAME record requests (Pietraszek, 2004). Since there are two utilities named DNScat, this one will be referred to as DNScat-P in this paper to distinguish it from the other one.

DNScat (DNScat-B)

DNScat (DNScat-B) was written by Ron Bowes. The earliest known public release was in 2010. It runs on Linux, Mac OS X and Windows. DNScat will encode requests in either NetBIOS encoding or hex encoding. DNScat can make use of A, AAAA, CNAME, NS, TXT and MX records. It provides a datagram and a stream mode. There is also a DNScat-B based Metasploit payload (Bowes, 2010).

Heyoka

Heyoka is a Proof of Concept which creates a bi-directional tunnel for data exfiltration. This tool is written in C and has been tested on Windows. Heyoka was developed by Alberto Revelli and Nico Leidecker. It uses binary data instead of 32 or 64 bit encoded data to increase bandwidth. It also uses EDNS to allow DNS messages greater than 512 bytes. Heyoka also uses source spoofing to make it appear that the requests are spread out over multiple IP addresses (Revelli, 2009).

Iodine

Iodine is a DNS tunneling program first released in 2006 with updates as recently as 2010. It was developed by Bjorn Andersson and Erik Ekman. Iodine is written in C and it runs on Linux, Mac OS X, Windows and others. Iodine has been ported to Android. It uses a tun or tap interface on the endpoint (Andersson, 2010).

NSTX

NSTX (Nameserver Transfer Protocol) From Florian Heinz and Julien Oster was released in 2000. It runs only on Linux. NSTX makes it possible to create IP tunnels using DNS (NSTX, 2002). It tunnels the traffic using either a tun or tap interface on the endpoints.

OzymanDNS

OzymanDNS is written in Perl by Dan Kaminsky in 2004. It is used to setup an SSH tunnel over DNS or for file transfer. Requests are base32 encoded and responses are base64 encoded TXT records.

Psudp

psudp was developed by Kenton Born. It injects data into existing DNS requests by modifying the IP/UDP lengths. It requires all hosts participating in the covert network to send their DNS requests to a Broker service which can hold messages for a specific host until a DNS request comes from that host. The message can then be sent in the response (Born, 2010a).

Squeeza

Squeeza is an SQL injection tool. It splits the command channel and the data exfiltration channel. The command channel can be used to create data in a database and execute other commands. It supports three data exfiltration channels: http errors, timing and DNS. For the DNS channel data is encoded in the Fully Qualified Domain Name (FQDN) used in the request (Haroon, 2007).

Tcp-over- DNS

Tcp-overdns was released in 2008. It has a Java based server and a Java based client. It runs on Windows, Linux and Solaris. It supports LZMA compression and both TCP and UDP traffic tunneling (Analogbit, 2008).

Tuns

TUNS was developed by Lucas Nussbaum. TUNS is written in Ruby. It does not use any experimental or seldom used record types. It uses only CNAME records. It adjusts the MTU used to

140 characters to match the data in a DNS request. TUNS may be harder to detect, but it comes at a performance cost (Nussbaum, 2009).

Malware using DNS

DNS has been used as a communication method by malware. Known malware using DNS include: Feederbot (Dietrich, 2011) and Moto (Mullaney, 2011). Both of these malware examples use DNS TXT records for command and control.

2. Security measures

Most of the programs or services to create DNS tunnels, are not very discreet to transmit information, so if a good analysis of DNS traffic is performed can be detected easily, but it is more difficult to know the content of what is this transmitting, because often times the information is encrypted.

Currently most companies focus their security in network traffic, HTTP, email, etc., and do not give importance to the DNS traffic, which can be a major failure. Typically, URLs or domain names FQDN (Fully qualified domain name), are easy words to remember for human familiar words that may have some number or symbol also, words that exist in the dictionary or comply with rules of writing, grammar, but as the DNS protocol is world there are many possibilities of FQDN because of the variety of languages and different characters used in them. Also keep in mind that there DGA generation algorithms name very similar to methods of coding domain name domain.

The format of an FQDN is the computer name a "." and the domain name, so an example might be, servidor1.empresa.com. The maximum length allowed for one FQDN is 255 characters (bytes), with an additional restriction to 63 bytes per label in a domain name. The FQDN labels are restricted to a limited character set: letters A-Z ASCII, digits, and the character '-', and not case-sensitive. Besides some characters more than were added in 2004 "ä, ö, ü, é, à, è ...". For this to identify tunnels DNS we will have to apply rules that determine when a DNS request is real or when trying to convey hidden information.

Basically methods to uncover tunnels DNS, DNS are to analyze the traffic and the payload of the DNS packets. Some of the important things to look for when analyzing for example the payload of DNS are the size of requests and responses, the entropy of hostnames, analyze the types of records that are not commonly used as TXT and its contents, see if some team frequently query external DNS, assuming that in an enterprise environment will have internal DNS that provide most addresses used in the company, perform statistical and grammatical analysis of the names of DNS requests, because as I mentioned above, these are usually readable and easy to remember for humans, and there are also signatures developed for certain applications that allow us to check the DNS header and payload content of a DNS package.

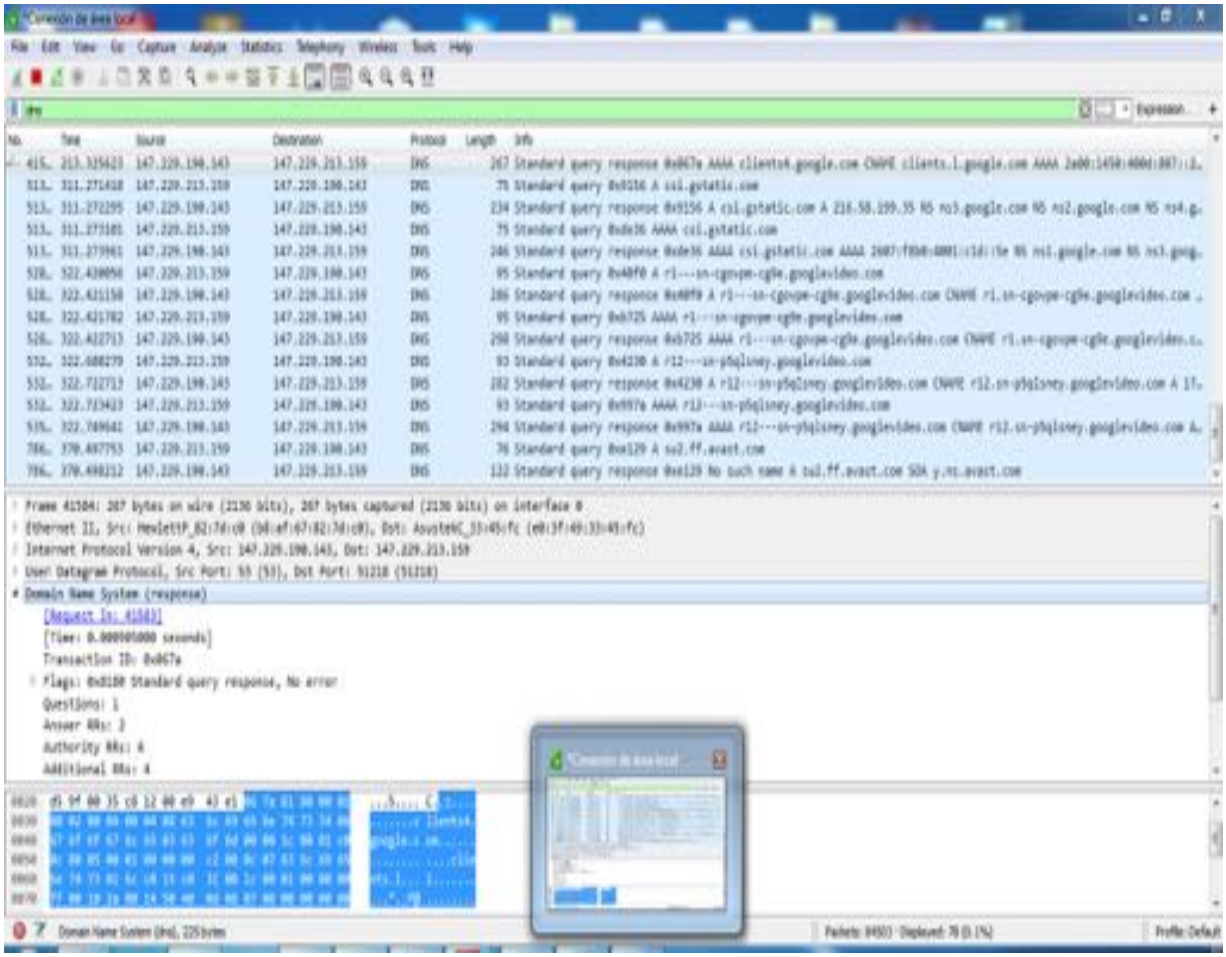

Figure 2.1: Capture DNS packets with Wireshark

As seen in the image, captured DNS packets on a network have this structure, we must analyze each field and ensure that the information they contain is logical and does not violate the normal behavior of a DNS message.

As we know there are 3 types of DNS messages, questions, answers and updates, their structures are defined in RFC 1035, RFC 2929 and RFC 2136, each a set of values for these fields specified in RFC are used. Normally the structure of the DNS message is:

Format PDU DNS

General Format

Header	12 bytes
0 or 1 issue (Q)	Variable size
0 or 1 RR responses (R)	Variable size
0 or 1 RR permitted references (A)	Variable size
0 or 1 RR additional information (I)	Variable size

Figure 2.2: Format PDU DNS

Format of the header DNS

	Bits	Field	Values
bytes 0-1		Identify	
octet 2	0	Type-PDU	0=QUERY 1=ANSWER
	2..4	Operation code	0 = Normal 1 = Reverse 2 = server status
	5	AA	1 = authoritative response
	6	TC	1 = truncated response
	7	RD	1 = requested recursion
octet 3	0	RA	1 = recursion available
	1..3	FROM	0
	5..7	Error code	0 = no error 1 = error on issue 2 = server error 3 = nonexistent name 4 = not answer 5 = Response refusal
bytes 4-5		Number of questions	
bytes 6-7		Number of responses	
bytes 8-9		Authorized number of references	
bytes 10-11		Number of additional information	

Figure 2.3: Format of header PDU DNS

Format of question DNS

NAME	Resource name	Variable size
TYPE	1=A 2=NS 5=CNAME 6=SOA 12=PTR 13=HINFO 15=MX 28=AAAA 255=*	2 bytes
CLASS	1 = Internet	2 bytes

Figure 2.4: Format question DNS

Format of a RR

NAME	Resource name	Variable size
TYPE	RR type	2 bytes
CLASS	0=Internet 1=Chaos	2 bytes
TTL	Period of validity	2 bytes
RDLENGTH	Size RDATA	2 bytes
RDATA	RR data	Variable size

Figure 2.5: Format response DNS

Then knowing the structure of a DNS message and possible values and meanings of the fields, I will explain in more detail some methods that can be used to analyze and detect messages tunneling technique.

2.1 Length of messages

Focusing on analyzing the size of requests and responses, usually in tunneling DNS as much possible data is transmitted in each message, so using some application packet capture and network filtering that interest us DNS, store information the values of DNS message fields in a database, and compare these values within normal limits. For example, if you are trying to discover if DNS tunneling that sends data out of the network, data exfiltration occurs, observe the values of the fields in which the source computer sends twice or more data than that DNS response received, also if server responses are 0 bytes, there is likely more concerned DNS tunnel. In the same way you can check if this happening infiltration data, noting that DNS requests of small size (bytes) a team receives responses from large size (bytes), in addition to further define a rule you can set a minimum value based on the average size of requests or normal DNS responses, which could be for example 80 bytes as average size for DNS requests and 180 bytes for DNS responses, as well as know the answer DNS contained within the request DNS so the size the response will be greater than the demand, also if depending on the number of responses you can make an estimate of the size that could have a reply Another feature

to consider that can help us discover a tunnel DNS is the label size, as we know that the limit for label is 63 characters (bytes), all those that approximate this amount are suspicious of tunneling so we will have to focus on analyzing them, besides those messages containing long names 255 characters, Another recommendation is to look at all hostname requests longer than 52 characters.

2.2 In-depth analysis of DNS messages

As I mentioned earlier the names of servers or domains tend to be easily readable and memorable for humans, so we can focus also on analyzing requests for DNS names with strange grammar or characters, for example usually they have few numbers, thus making analysis of hostnames, we can detect foreign hostnames, they can be with many consecutive numeric digits or interspersed with consonants, several consecutive consonants would also be suspect, or apply the rules of grammar of the language and set some benchmarks in which we allow to discover possible tunnels DNS through information hidden in hostnames, also can be compared hostnames DNS requests with the words from a dictionary or database, and it all depends if we analyze a packet or message more deeply we must take into account the time it would use the system to analyze all the packages, if we want more accurate cost analysis longer.

You can also analyze the entropy of host names to encode information because this value will be higher than normal that when writing any word in any language.

You may also analyze the hostnames and look LMS, significant longest substring, or identify the number of unique characters in a hostname, perform frequency analysis of characters, etc. You can use many analysis techniques hostnames but bear in mind also the time taken to analyze in great detail each message.

Another suspect point that can help us detect tunneling DNS is look at the types of records, if most are unusual such as TXT, MX and also the number of bytes is quite large near the allowable limits, this can be a good indicator that there DNS tunnel. Also it can be applied almost the same of analysis techniques that hostnames, statistical analysis of information containing the TXT record, analyze entropy, frequency of characters, etc, because normally the information in these records is usually readable by human, and a high rate of entropy and large amount of data on this may be a good test of DNS tunnel.

As it is known the limit of a DNS packet is 512 bytes but this can be increased to 1024 bytes with EDnS, extension packages DNS, this is not normal but can be used sometimes for DNS messages that contain a lot of information, for example, if a DNS message is about 250 bytes, 512-250 would have about 262 free bytes to hide additional information by encapsulating with a protocol such as IP or UDP, this does not mean that all DNS packets EDnS concerned insurance DNS tunnel, but it may be an indication, also when this technique is used to hide information in a DNS packet, usually hidden information is added at the end after the useful information from DNS message, which have the facility to check the contents in the header the message DNS not the total size of the DNS message

appears because the records it contains can be variable in size but can be calculated inspecting the contents of the message and consider whether to encapsulate IP or UDP after the useful content of the DNS exists grand amount of added information, it may be DNS tunnel. There is a possibility that the information does not hide this at the end of the IP or UDP frame this more difficult to detect the hidden information so serious, but not impossible.

Caution because the most sophisticated techniques to hide data in DNS packets, these do not add hidden information at the end of the packet data, in payload.

These techniques make use of pointers, adding a pointer to the end of the payload of the packet, after the hidden information, referencing valid previous DNS packet fields, so that the package looks normal, but contains the information hidden in the middle of the packet data.

This method also can be detected because it meets some features, for example contains several pointers in the packet data in the payload, and all pointers referenced previous valid package labels, and at the end of the packet data, will always be a pointer that pointed to data that match the size of the UDP packet.

With this technique programs that capture network packets in a similar way as Wireshark do not detect the package as suspicious but provide hidden information as shown in Figure 2.6.

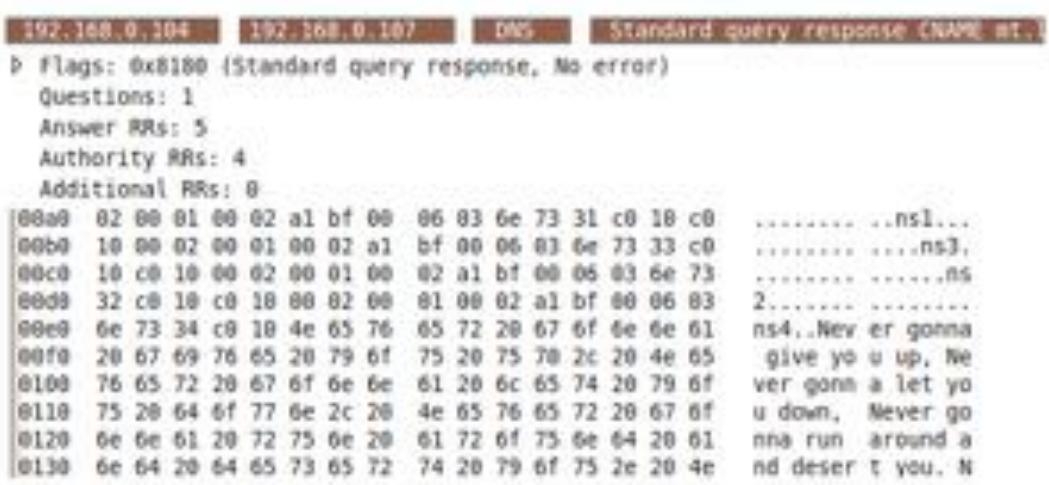

Figure 2.6: Packet tunnel DNS with injection of data in the middle

http://3.bp.blogspot.com/-xeVQApH9Sdc/TnJ1_Q74DII/AAAAAAAAABc/HlEbJD7_yMI/s1600/dns.png

Another feature that can help us identify tunneling DNS, is to control the destinations of DNS requests, because if we are in a controlled environment company, which have local DNS, then most normal DNS requests are found in the cache or local DNS of the company, so if we find that some client is constantly performing DNS queries to external servers on the Internet, it is possible to be transmitting hidden information through external DNS servers.

There are specific to detect DNS tunneling techniques in certain applications. These are called specific signatures, developed by some researchers or companies, and these can detect the method used to create DNS tunnels in some popular applications.

But the problem of these firms is that only detect certain specific types DNS tunnels developed in a concrete way.

For example, Snort signature was developed for detecting NSTX DNS tunneling (Van Horenbeeck, 2006).

```
alert udp $EXTERNAL_NET any -> $HOME_NET 53 (msg:"Potential NSTX DNS
Tunneling"; content:"|01 00|"; offset:2; within:4; content:"cT";
offset:12;    depth:3;    content:"|00    10    00    01|";    within:255;
classtype:badunknown;
sid:1000 2;)
```

To summarize, this signature is looking for a standard DNS query for a TXT resource record with text "cT"‖ near the beginning of the domain name.

Applying all these tips mentioned above have a great chance of detecting most of attempts to perform DNS tunneling. We can use any of them and best of all is when they are combined. Besides this we can also focus on analyzing the DNS packet traffic on a network, because everything mentioned so far is to analyze the content of DNS messages and not the amount or frequency of these messages. Below I will focus more on explaining methods to control and detect any DNS tunneling tips based on traffic packet and not its content. And at the very end if we use a combination of methods previously explained, and I will explain, we would get a good level of security against DNS tunnels, does not ensure that none exists, but greatly reduces the likelihood that these exist in our network.

2.3 Analyze network traffic

Other tips that can help us to discover a hidden DNS tunnel is to analyze network traffic to port 53 which is used by DNS, considering what is the origin of the requests and what is the destination. Also if you see a great number of unusual requests DNS this could be another indication. As is known, if is not used EDNS technique, extension of DNS messages, the limit of the DNS messages is 512 bytes, so for transmitting big amount information will be necessary a lot of requests and responses DNS.

Also as is explained before, if the client is receiving a big amount information from outside, from fake DNS server, and the server cannot send requests to client, the client will continually sending small false requests to server for receive the information in responses from server.

Therefore, I will explain some points more specifically to discover DNS tunneling. For example, if we observe a continuous amount of DNS requests from the same source IP to a DNS server, it might indicate a DNS tunnel. But with this we are not only safe as it can falsify or manipulate source IP so it appears that the source IP is different in each case.

Another important point is to look at the domain being queried, if detects many DNS requests, and all of them to a specific domain where only change the subdomains in the requests, and also if subdomains looks strange, exist a big possibility that will be DNS tunnel. Because the domain that is consulting can belong to fake DNS server that is controlled by evil user or unwanted user.

The part of the DNS request that belong to subdomain is the hidden information, but be careful with this because the same DNS server controlled by unwanted user, may have multiple alias or multiple domains names and the DNS requests can be targeted each time to one of these different alias or domains names.

Also if we see the same domain in which there have been many requests to unique names that have not been repeated, it can be great indication of DNS tunnel.

In a company that does not do business internationally, look for a significant number of DNS requests or responses, to parts of the world where there should be communication, control foreign origins and destinations. Further it is also advisable to observe when certain DNS records Type A or NS to our DNS servers are added if these domains should not be strange or may be an indication to detect DNS tunnels.

Another feature that can help to detect a covert DNS channel is to analyze the applications that use the DNS requests. Because a DNS request usually is related to an application like HTTP or another applications, that need to know some IP address for communication with other computer. Also note that sometimes some applications do DNS requests for security, so do not confuse this with strange applications, or sometimes one device may do reverse lookups for know one IP address.

Anti-spam solutions use DNS queries to check if IP address given is on a black list. An endpoint security product DNS queries use an encoded file hash with embedded in the FQDN to check the reputation of suspicious file.

Finally, we see that there are many characteristics that may indicate that there are DNS tunnels on a network, but it is very difficult to make a program or application to detect with 100% reliability all the techniques of DNS tunnel. However, there are many indications that can help us to detect this, so would be possible create one application that report strange traffic in the network, also with the help of a computer expert in the company, can be reduced or completely eliminate the risk of DNS tunnels.

One application that eliminates any communication that is indicative of this technique, maybe can eliminate false positives too, or restrict communications that should be allowed and this is harm for the network.

How difficult is to establish the optimal values of the parameters?, like delay of packets, size of requests and responses, or amount of DNS traffic.

This depends on each company or network according to the purposes and characteristics that these have, as the numbers of users, the type of network, etc, so cannot set static values.

With the use of signatures or programs to detect we can reduce the possibility of tunnel DNS on our network, but not 100% guarantee that this does not exist, because there are many ways to convey hidden information and any hacker could change the existing tunneling methods making the detection algorithms clueless.

Another important point to consider depends on the security and information involved in our network, is important have a good organization of our DNS servers, good control about DNS zones in our

server and which servers can consult our DNS server if it does not have one IP in his database, this is called the recursion on DNS server and is good controlling which servers are authorized for resolve this.

Also is not the same the work in one network that can allow connection any user or computer that connect in our network, this kind can have dynamic IPs and will be more difficult identification the computers if something happen, that one network where all users or computers must be known, where can be assigned statics IPs for each computer, where is not allow connection for any computer, and where can be controlled which DNS server is used for the computers.

Using all the tips mentioned in this document there is a high probability of detecting a DNS tunnel and eliminate or restrict its use, analyzing the content of DNS requests and responses or analyzing the traffic that flows through our network.

This last point to analyze network traffic, can also be taken into account to detect any abnormality in our network, either DNS or tunneling tunnel through any protocol or method, good control of network traffic can help to reduce the risks of unwanted information is transmitted.

In the following parts of this document I will explain more practical issues and demonstrate how to implement a DNS tunnel with existing tools and how to apply countermeasures for detecting this.

3. Experiments

After researching on the various existing applications available to create tunnels DNS, I think the best and most complete is **Iodine**. This program works on multiple operating systems and distributions, Linux, Mac OS X, FreeBSD, NetBSD, OpenBSD and Windows but in Windows needs a TUN / TAP device. The bandwidth is asymmetrical with limited upstream and up to 1 Mbit/s downstream. No matter that the client and server are on different operating systems, they can communicate with each other.

The program for the side of server and the side of client, it is easy to use but depends of operating system which use, normally is enough with install the program is both sides and execute one command for run, but as I said this depends of operating systems. In my experiments I used the server in Ubuntu system and the client in windows 7 system, will be explained step by step how install and run the Iodine program in these systems.

Also in internet is possible found a lot information about how install this application Iodine in different operating systems, as those named above, Mac OS X, FreeBSD, etc.

There are more programs like heyoka but there is little information about it, how to install and use, plus Iodine has more options and parameters that allow us to adapt to the network environment in which we find ourselves, allowing us to use various types of DNS records to transmit information, such as NULL, PRIVATE, TXT, SRV, MX, CNAME and A. It also allows you to use various types of character encodings, which according to the amount and/or type of information we want to convey can use one or the other. The types of encoding are as follows:

- Base32 is the lowest-grade codec and Should always work; When this is used auto detection fails.
- Base64 Provides more bandwidth, but May not work on all name servers. Base64u is equal to Base64 except in using underscore ('_') INSTEAD OF plus sign ('+'), possibly working Where Base64 does not.
- Base128 use high byte values (mostly accented letters in iso8859-1).

Other existing programs already mentioned above as Ozymandns, or DNScat, Squeeza, only allow transmitting information with a type of coding or some type of DNS single record. Others like NSTX are obsolete and no longer maintained and people who used this program now recommend using iodine.

Iodine is the most advanced program that exist at moment, it allow create DNS tunnels with different configurations, as I said before it can use all of types records of DNS packets, as NULL, PRIVATE, TXT, SRV, MX, CNAME and A, also it allow running with different kinds of codification for the

communication, as Base32, Base64 or Base 128. Also it allows use any network interface that have in the computer, as Ethernet, Wireless, etc, and configure which interface will work with tunnel or which interface we want work like normal. Besides it has options for use any port that you wish for the communication, not is obligatory used the port 53. It allows to establish the maximum MTU size, it allows force maximum downstream fragment size., maximum length of upstream hostnames, by default is 255, besides maximum interval between requests (pings) so that intermediate DNS servers will not time out. It is most used for this kind of communication also it is the most completely and it allow the most different executions using its several parameters, these will show below. For more information, consult your manual page Iodine.

3.1 Requirements to create a DNS tunnel with Iodine

Iodine 0.5.x can be downloaded from http://code.kryo.se/iodine/. If used in Windows you must first install a TUN/TAP interface, which is a part of Open VP, for instance. Open VPN can be downloaded from here: https://openvpn.net/index.php/open-source/downloads.html

DNS server is required with internet connection and public IP, is needs a domain. (If you do not have this you can use any DNS service free internet in my example, I use FreeDNS (https://freedns.afraid.org/).

3.2 How to install Iodine

In my experiments I used Windows 7 and Ubuntu, but Iodine can run in several operating systems, I will explain how to install Iodine in this systems. First know that Iodine work like client-server, so will be necessary install the program in both sides.

First of all, we will add two new entries to our DNS server, in our domain, if we don't have DNS server, use the previously mentioned FreeDNS. In my case, I used a free public domain, "chickenkiller.com". The steps to configure iodine are as follows:

- We added a record type with the name you want for our DNS server.

```
t1nn IN A 147.229.176.18
```

- We add a NS record type point to our previous server.

```
t1nnns IN NS t1nn.chickenkiller.com
```

First for run Iodine in Windows is necessary install the TUN / TAP interface in computer of server and computer of client, you can do it downloading the program OpenVPN from here.

- After this for install the server part in Windows 7, download the program from the link above. In the folder will have the application for side client and for side server, we can differentiate because

the program for the client is Iodine.exe and the program for the server is Iodined.exe, look the final "d".

- After this, open a console in windows 7, always work like admin, go to the directory where is the program and execute the follows command:
- For server side in Windows:

```
>Iodined.exe -f 192.168.1.1/24  ourdomain.com
```

Where 192.168.1.1/24 will be the IPs for our tunnel, and ourdomain.com will be our name of domain, in my experiments I used t1nn.chickenkiller.com, but can be another one, also the program will ask you for one password, write the password that you wish and remember this for after. Now we have the server running on windows, then I'll show you how to run the server on Linux, Ubuntu.

- For install Iodine in Ubuntu, open one console and write:

```
$sudo apt-get install iodine.
```

- After open the file /etc/default/iodine and edit this file:

```
$gedit /etc/default/iodine
#  Default  settings  for  iodine.  This  file  is  sourced  from
#                                                  /etc/init.d/iodined
START_IODINED="true"
IODINED_ARGS="192.168.1.1  t1nn.chikenkiller.com"
IODINED_PASSWORD="Password for the connection"
```

"The IP 192.168.1.1 can be any private IP, this will be IP for the tunnel connection."

- Save the file, and write the next commands in the console, this commands are for the server send forward all the traffic received from client, will allow the client connect to internet, eth0 must be the interface that provide internet to server:

```
$echo 1 > /proc/sys/net/ipv4/ip_forward
$iptables -A POSTROUTING -t nat -s 192.168.1.0/24 -o eth0 -j MASQUERADE
```

- Finally restart the iodine server for apply the changes, the server will be running:

```
$/etc/init.d/iodined restart
```

If everything was correct server you will be running as shown here:

Figure 3.1: Iodine running OK

The client side was in my case installed on a Windows 7, but can be installed on another Linux or any other system. Installing client in Linux is the same as the server but instead of using the executable **iodined**, the executable is **iodine**. To use iodine on Windows as I mentioned before, a TUN/TAP interface must be properly installed. It can be installed as a part of Open VPN. If you do not want to install the full program, install only the interface TUN / TAP. To do so, choose the option TAP Virtual Ethernet Adapter as shown in Figure 3.2.

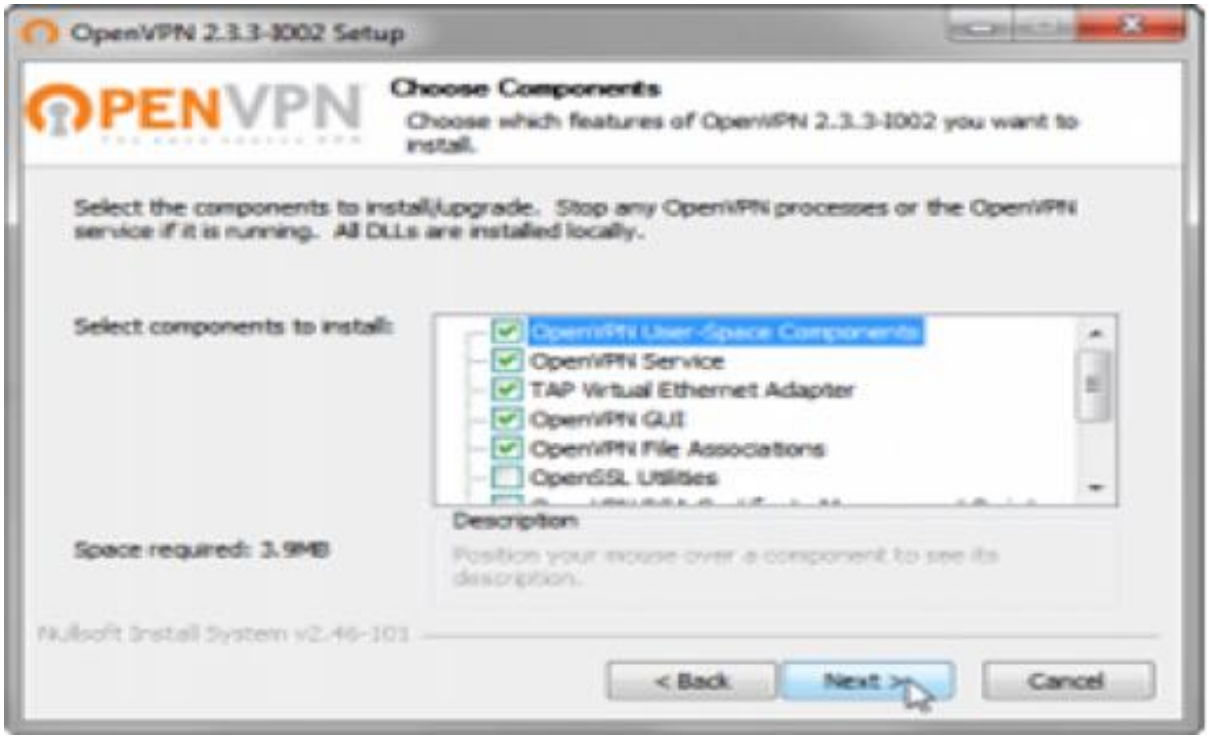

Figure 3.2: Installing TUN/TAP interface

After this installation, if we already have the iodine downloaded in our computer, open the console like root or admin and go to the directory of iodine, and run the following command in our console:

```
iodine.exe -f 147.229.176.18 t1nn.chickenkiller.com
```

If everything is correctly configured the tunnel will be created between a Linux (server) machine and Windows (Client) machine as the data discussed in this document, the IPs tunnel created will be 192.168.1.1 the server side and 192.168.1.2 the client side to verify that the tunnel is successfully created and the communication is working, try to do ping from one end to another, as shown in Figure 3.3.

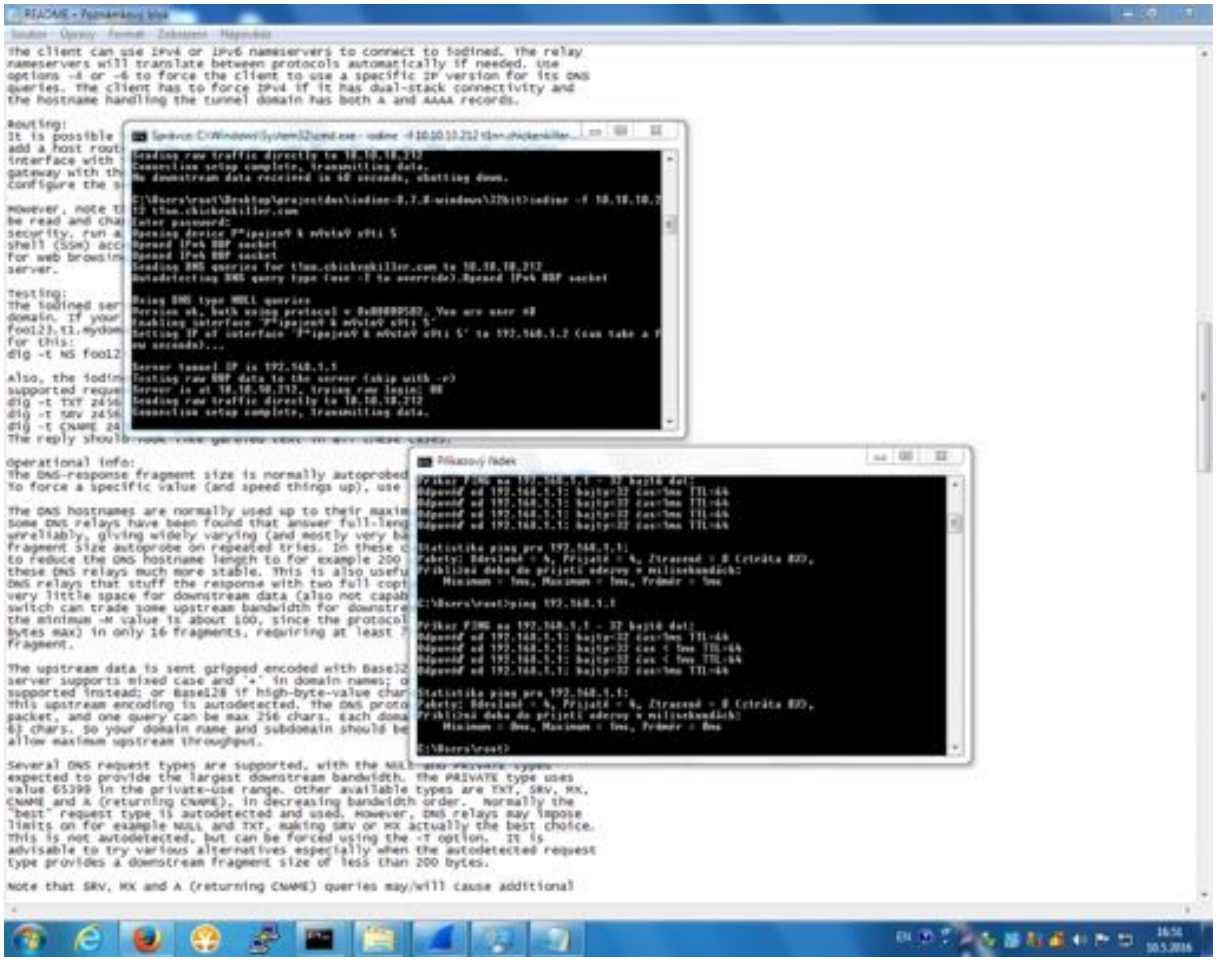

Figure 3.3: Checking connection with "ping" after create the tunnel

Now the tunnel between the two machines is created and we can transmit information. If you want the client machine send all the information through the DNS tunnel, you have to modify a client's routing table. Type the following commands in the CLI of you client's operating system (in my case Windows, but for Linux it will be similar):

```
>route  add 0.0.0.0  mask  0.0.0.0   192.168.1.1  metric  10  if  20
>route del 0.0.0.0 mask 0.0.0.0  10.10.10.1
```

Here, 192.168.1.1 is the IP of server side, the metric should be smaller that the existing default route, number 20 in my case is the number that identifies my interface TUN/TAP, and 10.10.10.1, in my case is the IP address of the existing gateway..

The client will be transmitting all information through the DNS tunnel, the server is receiving all information through the tunnel and forwarding of this information to others computers in internet or where will be necessary, also the server is providing the answers and necessary information that the client asked to server,

this can be checked by doing a tracert from the client to any address on internet and seeing that traffic is sent through the server.

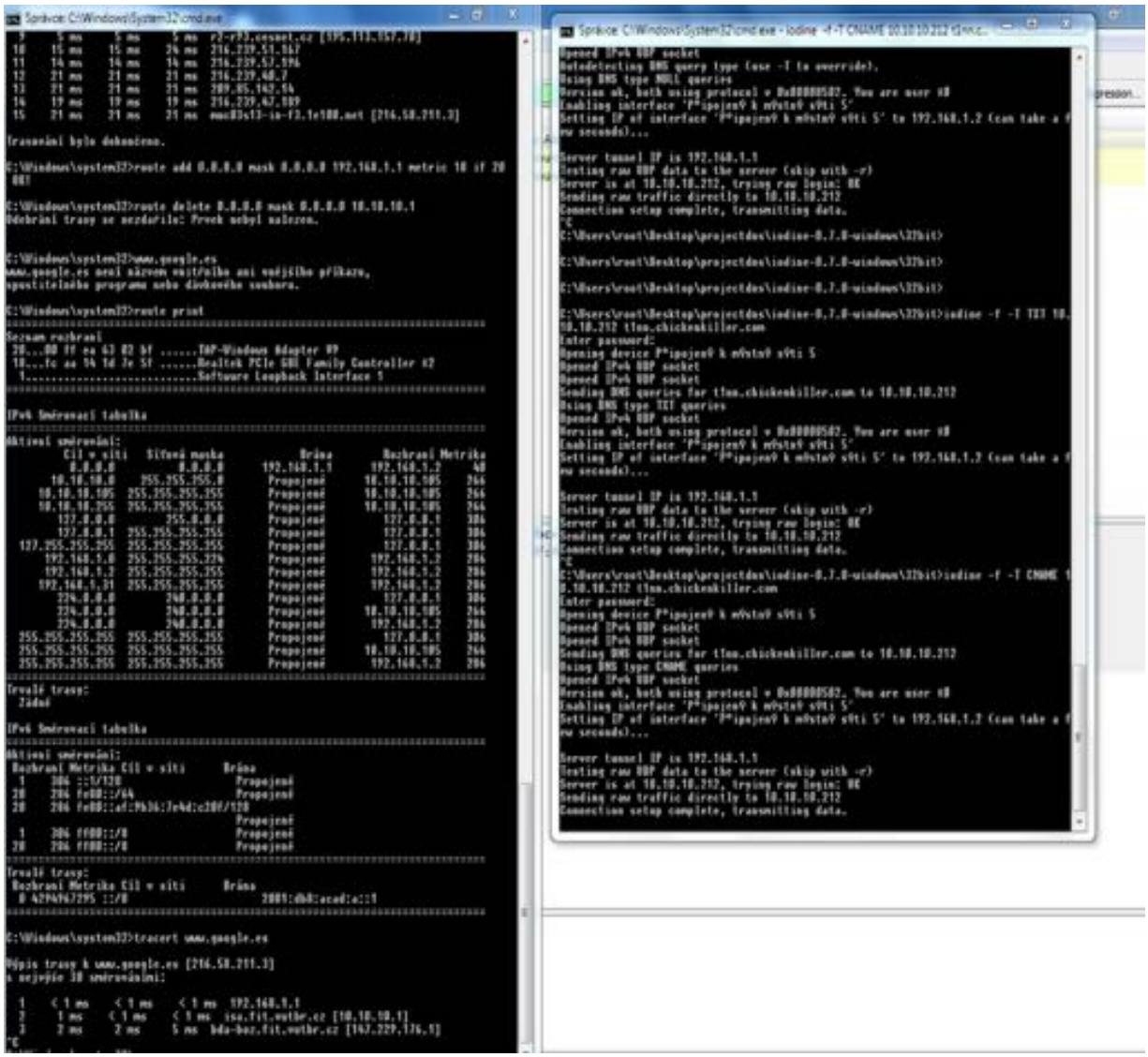

Figure 3.4: Checking the route of the packets with traceroute

To verify that the tunnel is working well and that all information transmitted through it, you can run any sniffer like Wireshark and see that all packages are being forwarded to the VPN created. In addition, the data are encrypted in the DNS packets, so routers, firewalls or other security equipment will see this traffic as DNS packets with unknown content, see Figure 3.5.

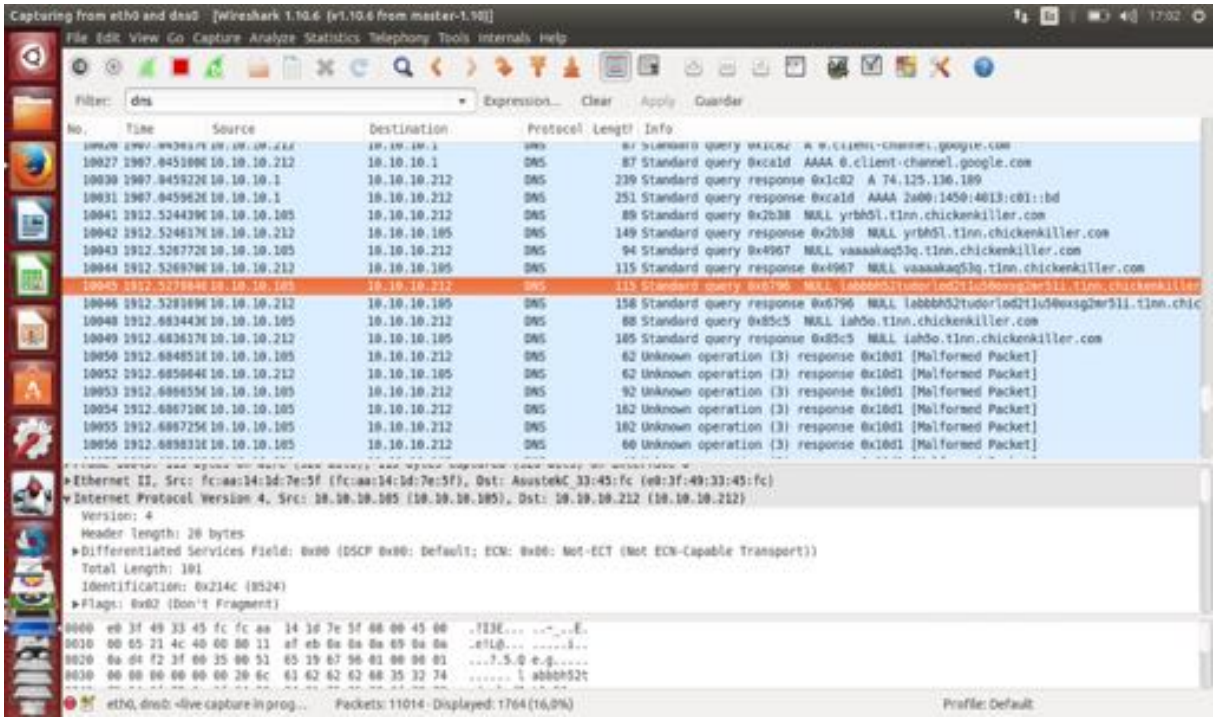

Figure 3.5: Capture of packets in DNS tunnel with wireshark

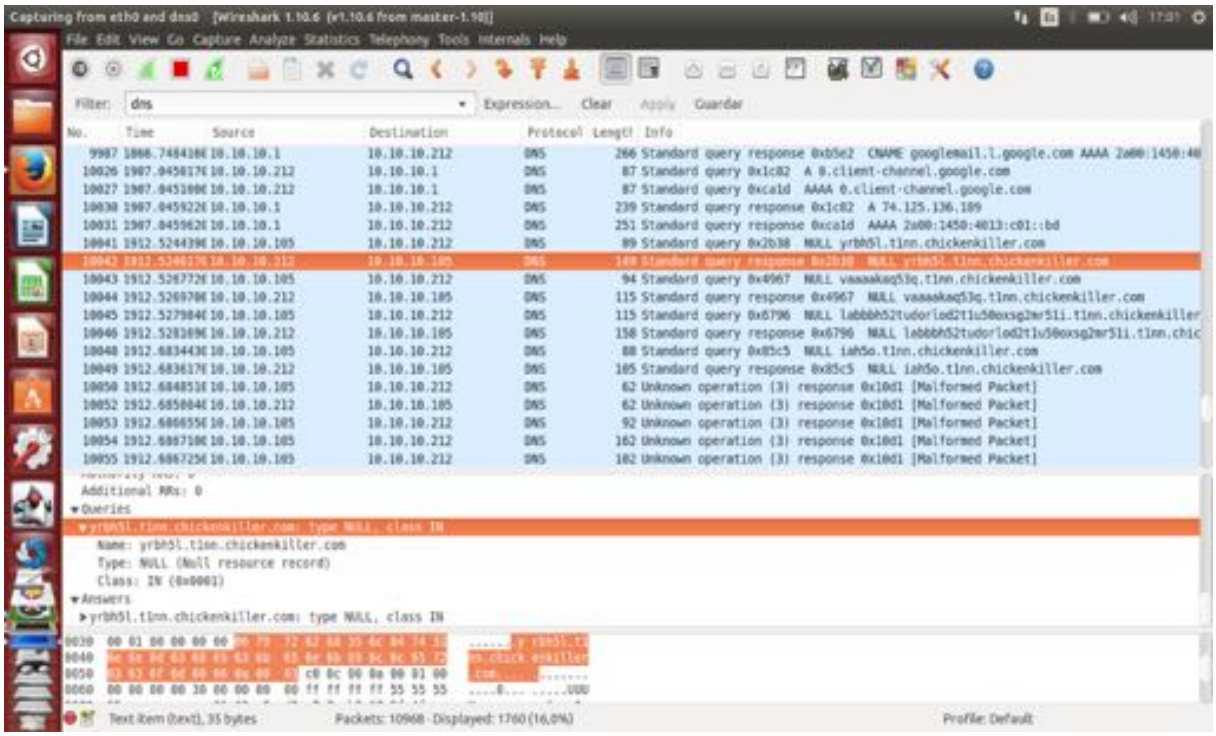

Figure 3.6: Capture of packets in DNS tunnel with wireshark

As you can see in Figure 3.5 and Figure 3.6, the first bunch of packets sent through the tunnel are recognized by Wireshark, if we analyze closer we see that the type NULL of DNS records is used to transmit information, so the first messages are recognized but the following will not, because once the

tunnel is established, the program iodine modifies packets so that they are not recognized by Wireshark but contain the hidden information.

If we look close in the first DNS packets are recognized like normal packets in DNS ,but if we see the content of DNS queries, these queries are performing request to server for strange domain name, for example that appears highlighted in orange in the first picture, also in second picture appears the query that ask for a strange domain name like show below:

-First picture, figure 3.5, asking for FQDN:

"labbh52tudorlod2tlu50oxsg2mr5li.t1nn.chickenkiller.com"

-Second picture, figure 3.6, asking for FQDN:

"yrbh5l.t1nn.chickenkiller.com"

The name of an internet address seems a little odd and complicated to remember for a person as mentioned above, the server names are used to provide people access to Internet addresses without remembering the IP. This is because iodine is encrypting the information and sending it to our request hidden in the DNS server. This is actually send IP traffic on the DNS protocol.

As mentioned above Iodine has several options to use different records and different types of coding, Figures 3.7 shows multiple runs of the program with different parameters and can be seen as Iodine uses certain types of records and transmits information.

For example with CNAME record types, using parameter -T CNAME in the command for run:

- `> iodine.exe -f -T CNAME  ipserver  domain`

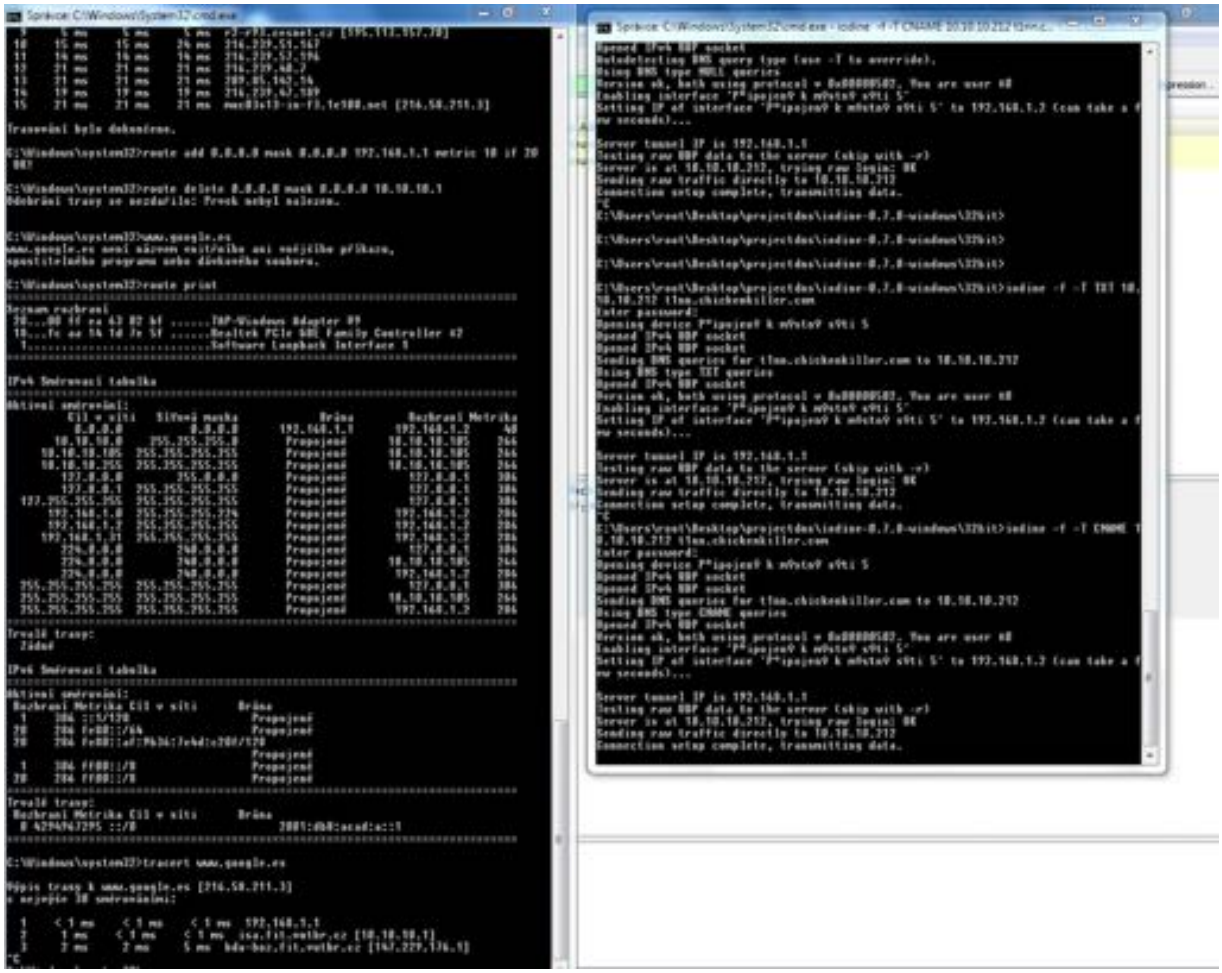

Figure 3.7: Executing Iodine with records CNAME

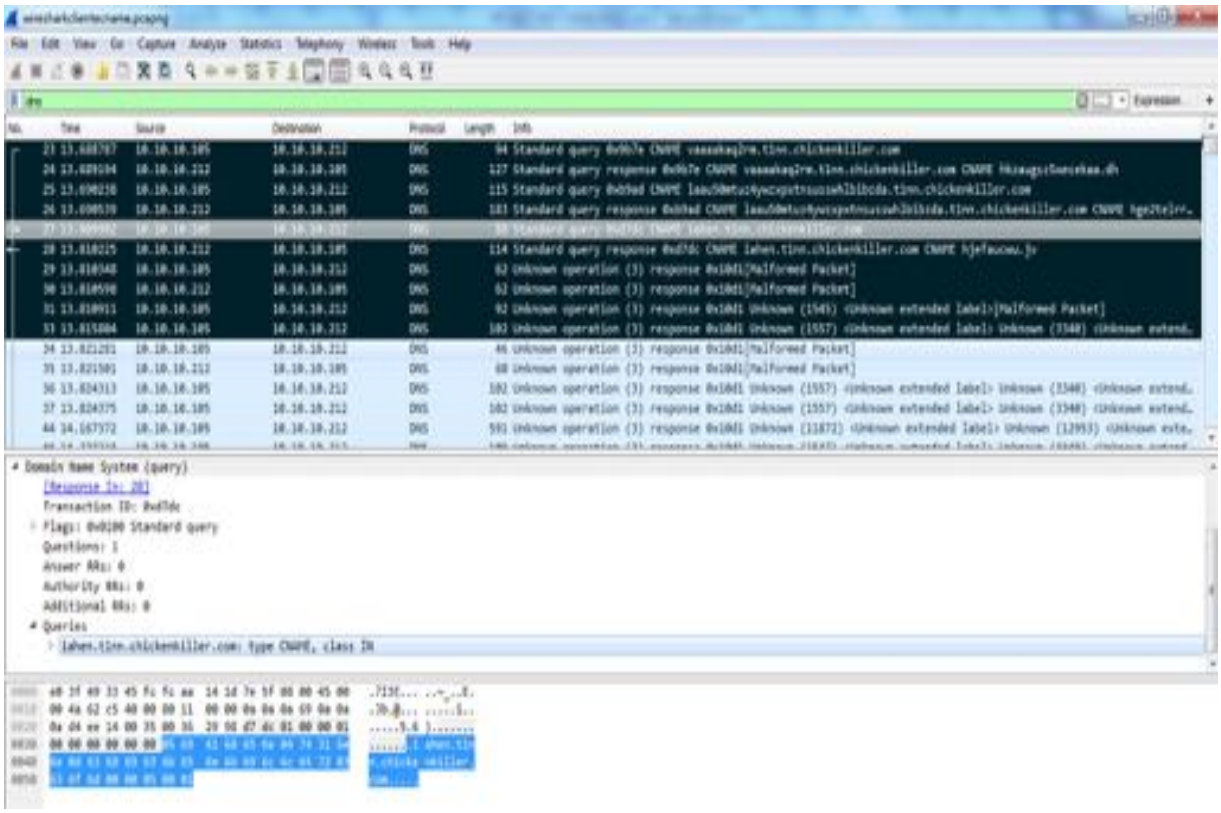

Figure 3.8: Capture of Wireshark with Iodine running with CNAME

Note that after each restart of Iodine client with different parameters the information about new default route must be refreshed.

Also in figure 3.9 Iodine running with TXT record types, using parameter -T TXT in the command for run:

- `> iodine.exe -f -T TXT ipserver domain`

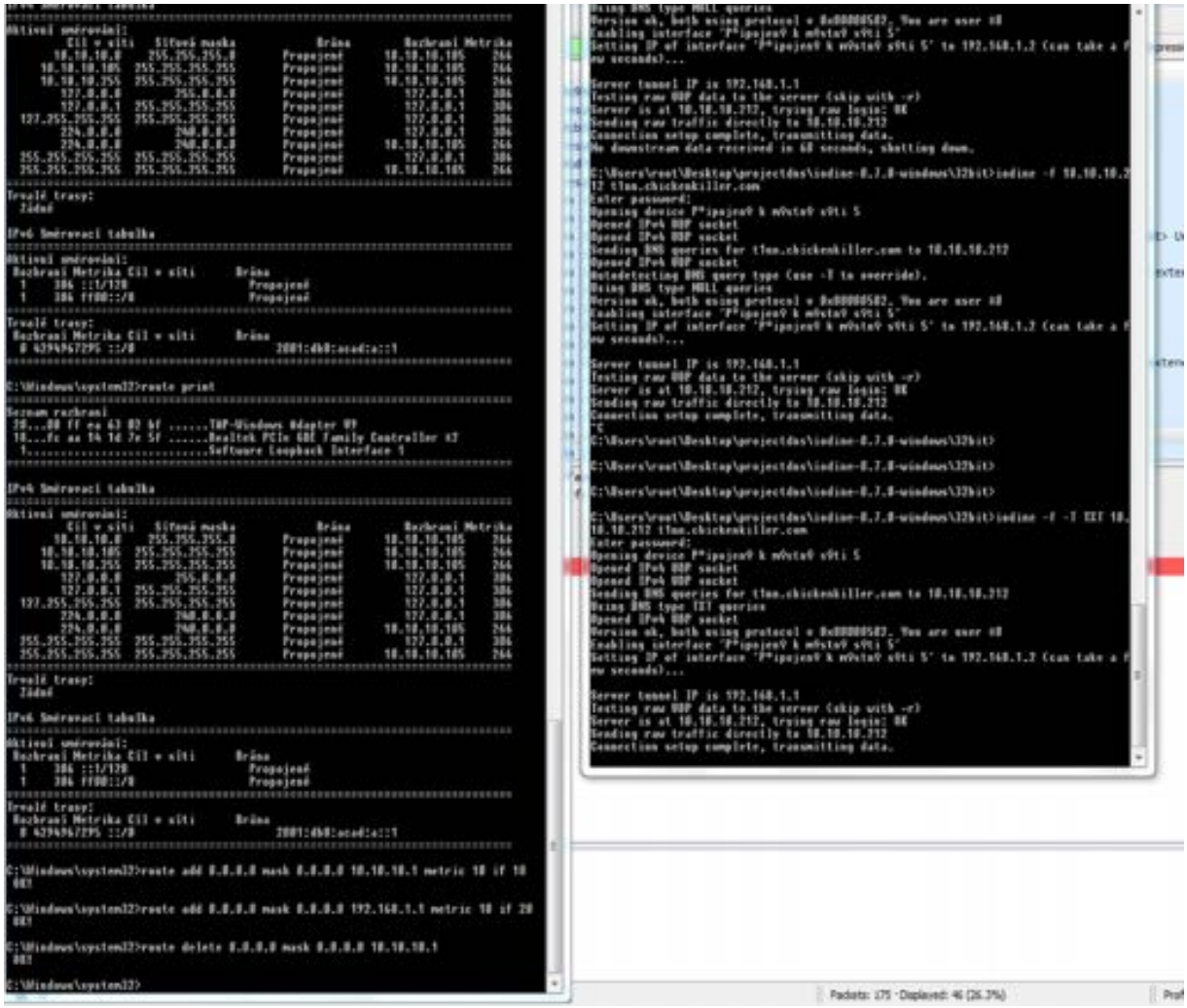

Figure 3.9: Iodine running with records TXT

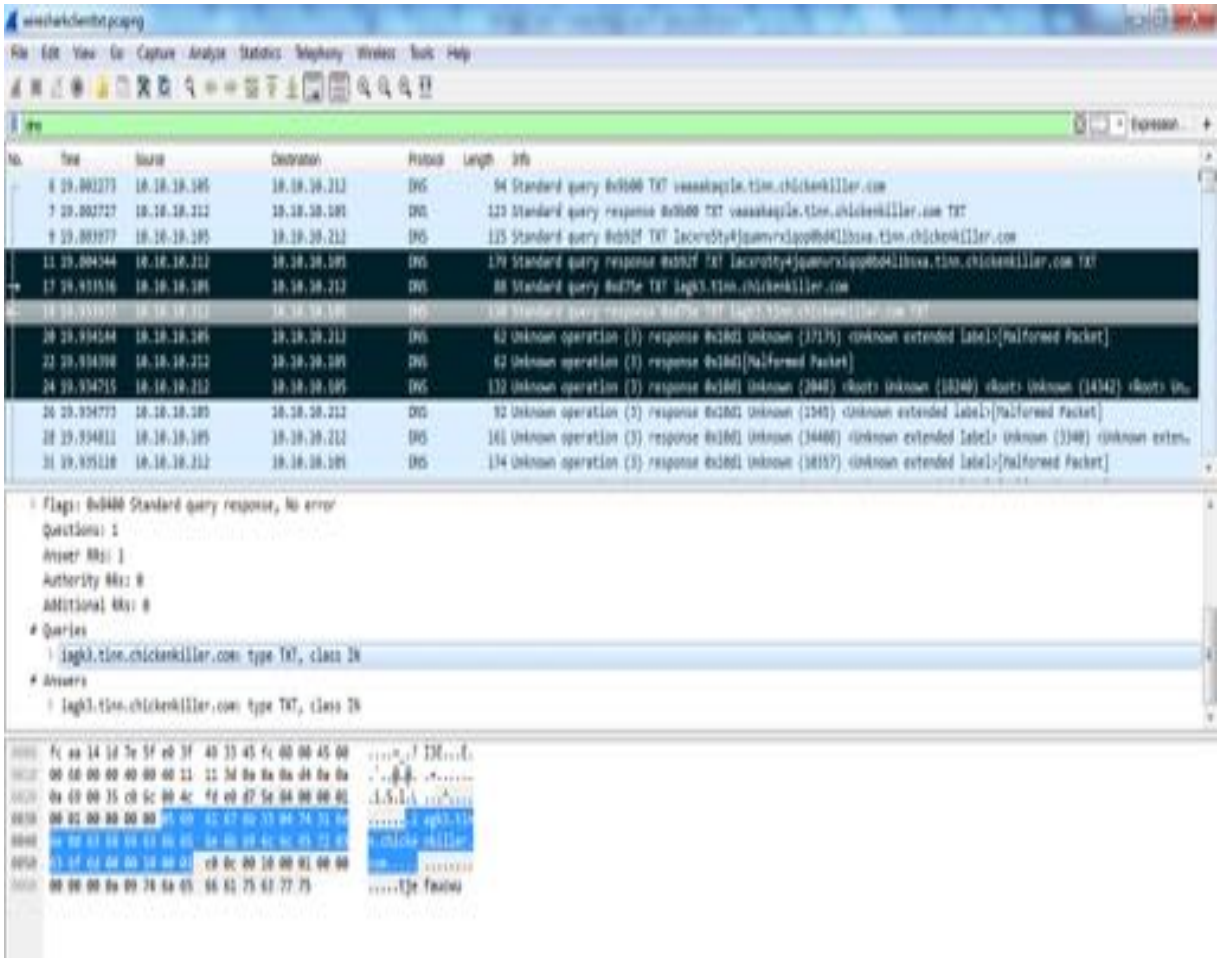

Figure 3.10: Capture of Wireshark with Iodine running with TXT

As I said earlier there are many types of records and all can be used with iodine, here are shown 3, but all can be used. This is one of the advantages provided Iodine versus other programs DNS tunnels,

Also we can choose the kind of codification for information transmitted in the tunnel with the parameter -O, we can choose between Base32, Base64 or Base128, above the benefits of each are explained.

Below one image of Iodine running with encoding Base128, figure 3.11, the command for execute Iodine with this kind of encoding is:

```
> iodine.exe -f -O Base128    ipserver    domain
```

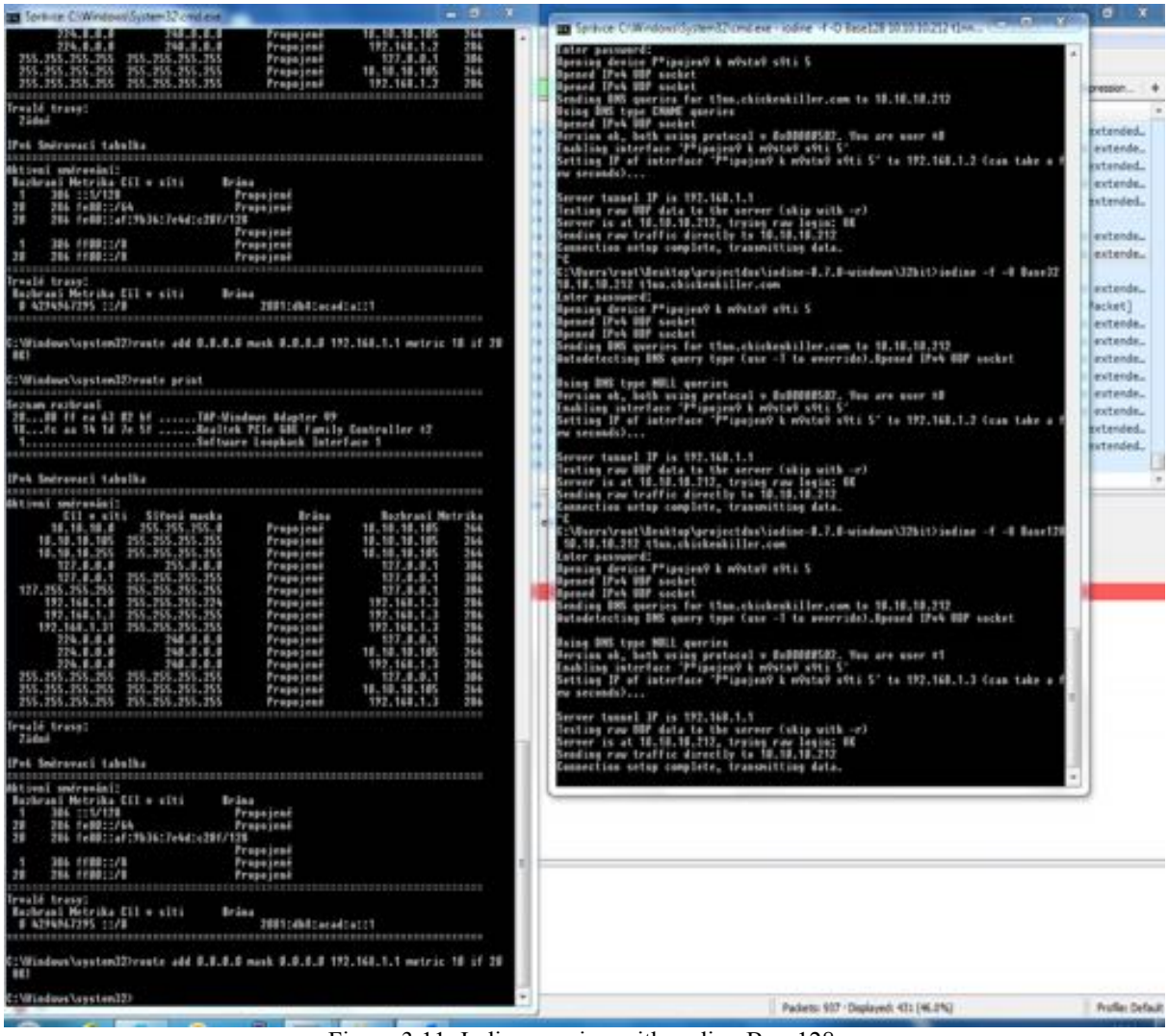

Figure 3.11: Iodine running with coding Base128

Well, as has been explained step by step above, you can create a tunnel between 2 computers and transmitting all the information hidden in DNS packets, this is a great technique of communication but if we think a little about this, depends of the intents of men which use this method DNS tunnel can be very good or can be a big problem for our network if the DNS tunnel is not wanted because with this technique we can avoid hotspots, in some cases firewalls, or communicate with any internet computer covertly to other teams, whenever we have a team with connection to a network, of course that the router or where we are connected should allow traffic on port 53 which is used for DNS, but in most networks this, even a firewall is allowed depends on the type of restrictions that apply to a computer can be avoided with this technique, because if allows us traffic on port 53 can avoid it.

Iodine also has the option to change the port through which traffic is sent, if another port that allows connection can be changed to redirect this traffic and not by 53, with the -p option, it is known that well described in the manual of iodine:

```
-p port
Make the server listen on 'port' INSTEAD OF 53 for traffic. If 'listen_ip'
does not include localhost, this 'port' can be the same as 'dnsport'.
Note: You must make sure the  DNS requests are forwarded to this port
yourself.
```

It could redirect traffic to our client in a different port using this command:

```
> iptables -t nat -A OUTPUT -p tcp -j --dport 43 DNAT --to-destination
127.0.0.0:xxx
```

Where xxx will be the port that we want use for send.

With all that explained it can be seen that the DNS tunnels are a fantastic, very useful and can be created with many different configurations and different ways, using this tool Iodine.
But the use of DNS tunnel can be harmful for our network or our company, for example if we have a private network, where we control the communication to internet and is only allowing access to internet one of our computer in all network, this computer would be identified and associated with a responsible person and with confidence in the company, but all computers are connected to the network, we can restrict access to internet by one firewall or some way that limited or control that only one computer can access to internet, but all of them have authorization for do requests DNS, the DNS tunneling can be used by one of these computer unauthorized to access internet and it sent private information of our company outside of our network without authorization.
For this reason, below we will explain how to detect DNS tunnels in our network and as prevent and eliminate them, but with everything that we have seen will not be an easy task.

4. Detecting DNS tunneling

In this chapter I will explain and perform some experiments and techniques for detecting DNS tunnel of practical way, now that have seen the different ways by create DNS tunnel, the differents types of codifications that can be used, the differents kinds of records in DNS packets that can be used to send hidden information, and after perform experiments with one of these tools to create DNS tunnel, that was Iodine, remembering all of the tips in the second chapter in this document, let to experiment and discover how detect DNS tunnel in efficient way and practical.

As explained in chapter 2 of this document, the main features to detect tunnels DNS can distinguish between:

- Size and quantity of DNS packets.
- Analyze network traffic
- Analyze in depth DNS packets

To do this we need a packet sniffer tool, in my case I used Wireshark, because Wireshark is a good tool for analyze the network traffic, Wireshark have many options for filtering the network traffic allow capture and show just the traffic that we wish and locate the DNS tunnel or another problems in the network in one way easy and fast, we can filtering the traffic by protocols, by IPs, in different interfaces, etc.

Also with this tool we can see the graphics of network traffic, size and kinds of network packets, content of the packets, the source and destination, traffic IP and UDP, etc.

In this part I will teach how use Wireshark and many of its options, but for more info you can consult here.

So with good use of this program we can detect DNS tunneling looking in the correct places as I will show below.

I tried also sniffers like tcpdump and other cloud shark, but none of them detected packets encrypted by Iodine, all of them showed the packets like Wireshark as unknown packet types.

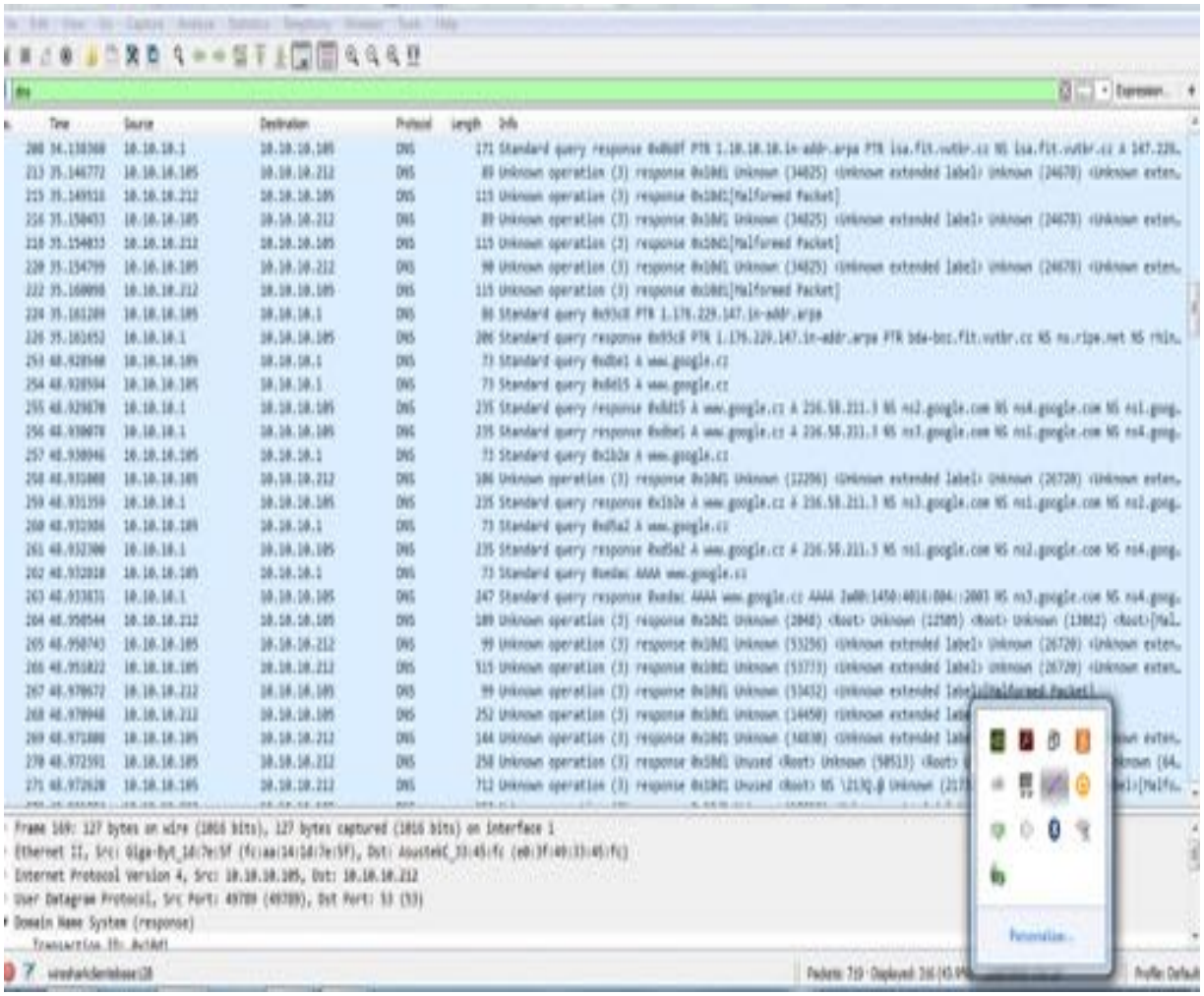

Figure 4.1: Capture Wireshark in network with DNS tunnel active

4.1 Size and quantity of DNS packets and network traffic analysis

After filtering packages just to show those belonging to DNS, as we can see in the image above, figure 4.1, we can difference between 2 kinds of packets belonging to DNS communication, the kind of packet standard query that is the normal packet that works the DNS protocol, and another type of packet is detected that it is unknown packets, this kind of packet can appears sometimes in our network because some error happened but is not normal the quantity of packets that we can see in the image before, because this image show one capture in one network that there is one DNS tunnel active, if we look there more packets of type unknown than normal packets, more than 50% and in this picture only show a little part of the capture, if we check the full file of the capture we can check that more than 80% of the packets are unknown in the network, so it is not normal in the network, just with this we can suppose that something wrong is happening in our network concretely with the

protocol DNS, also we can know where is the source of the problem like we can see in the previous figure 4.1, that the source and destination of all of these packets is same, in my case we know that all of packets the type unknown are from the communication between IP 10.10.10.105 and 10.10.10.212, because in my experiments the IP 10.10.10.105 belong to client of DNS tunnel and 10.10.10.212 belong to server of DNS tunnel, but in other network that we can try to detect DNS tunnel, we don't know if this IPs belong client or server DNS tunnel or just is some error in the communication between these computers, so is need to continue analyzing the network traffic and packets.

In the following images a comparison of two packet captures made on the same network, shown in one if the technique is working actively DNS tunneling and the other is the same network without DNS tunnels.

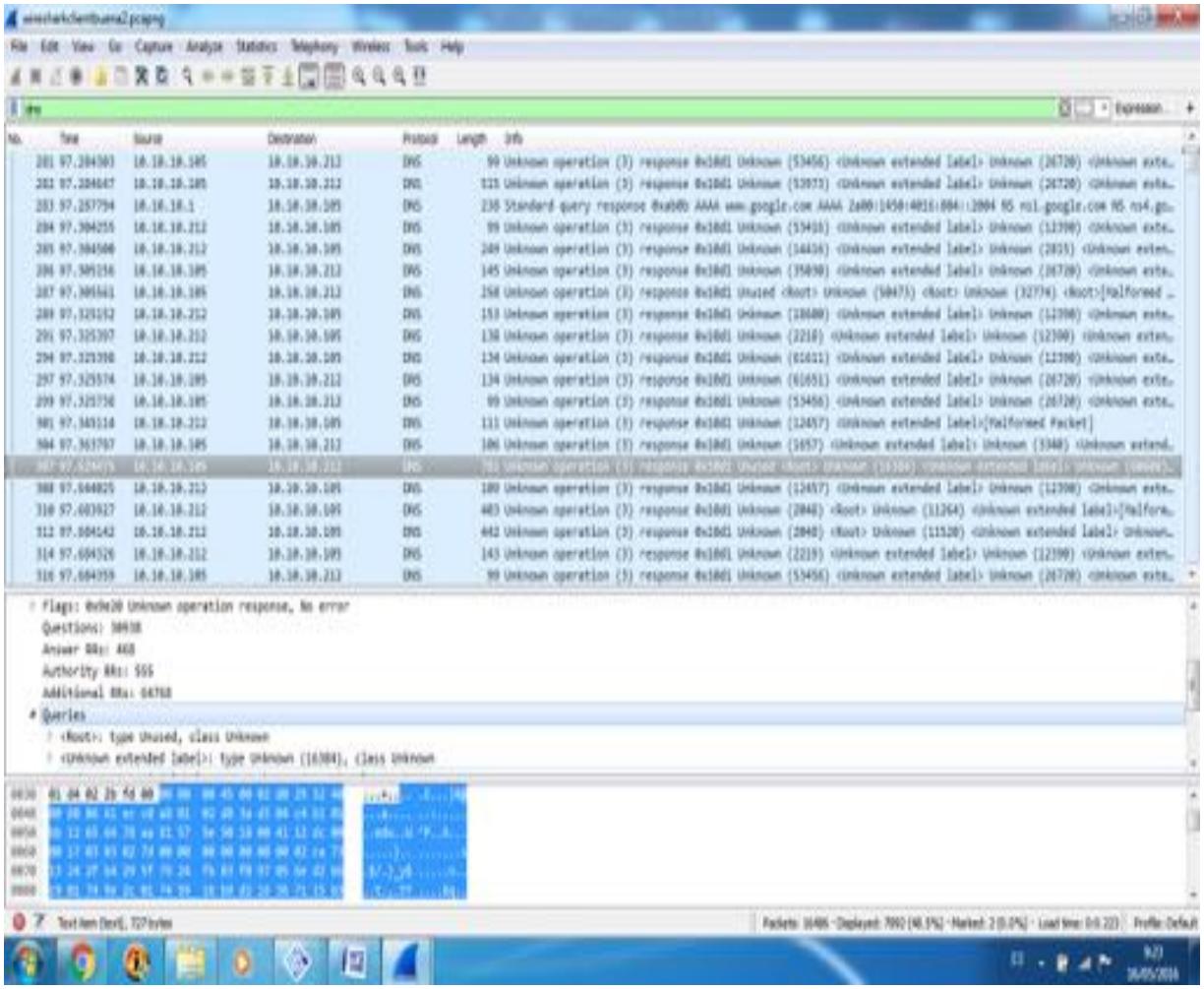

Figure 4.2:Capture Wireshark, detecting packets unknown

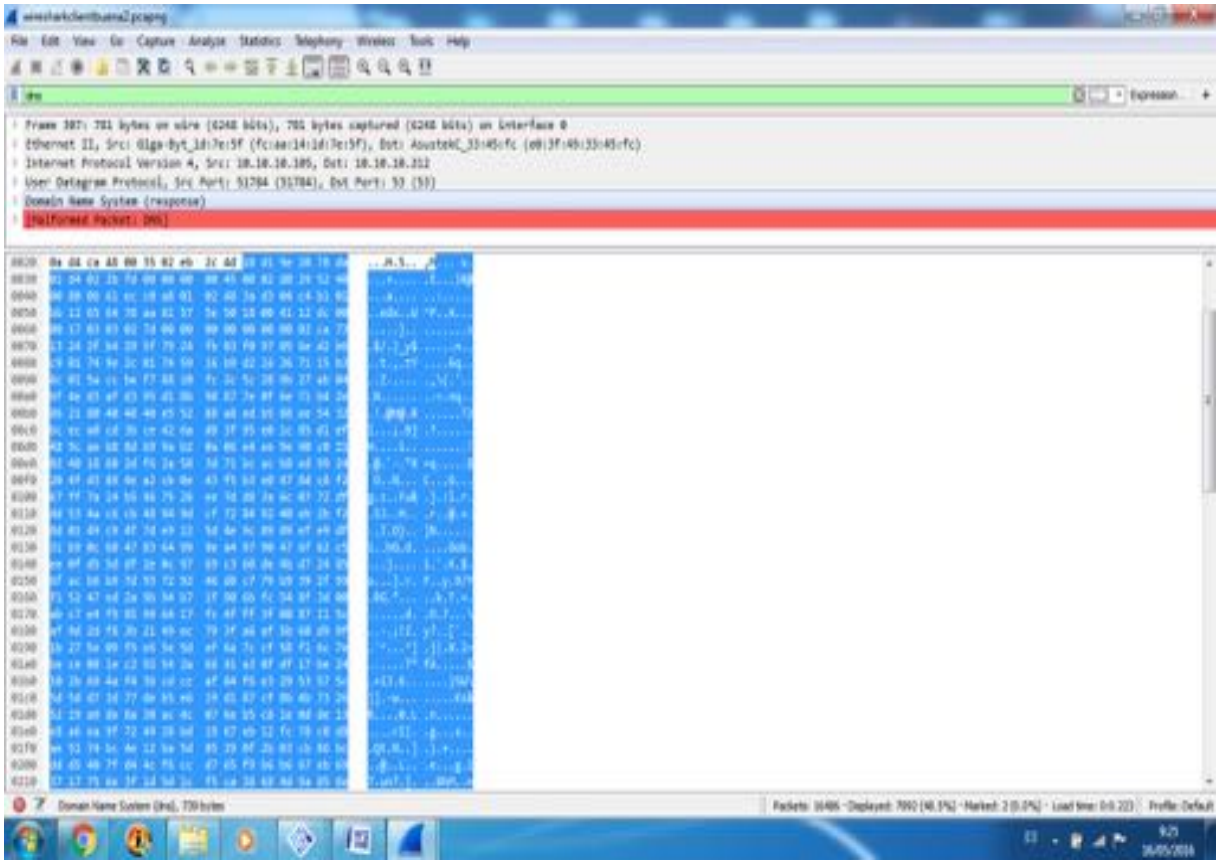

Figure 4.3: Contents of packet unknown

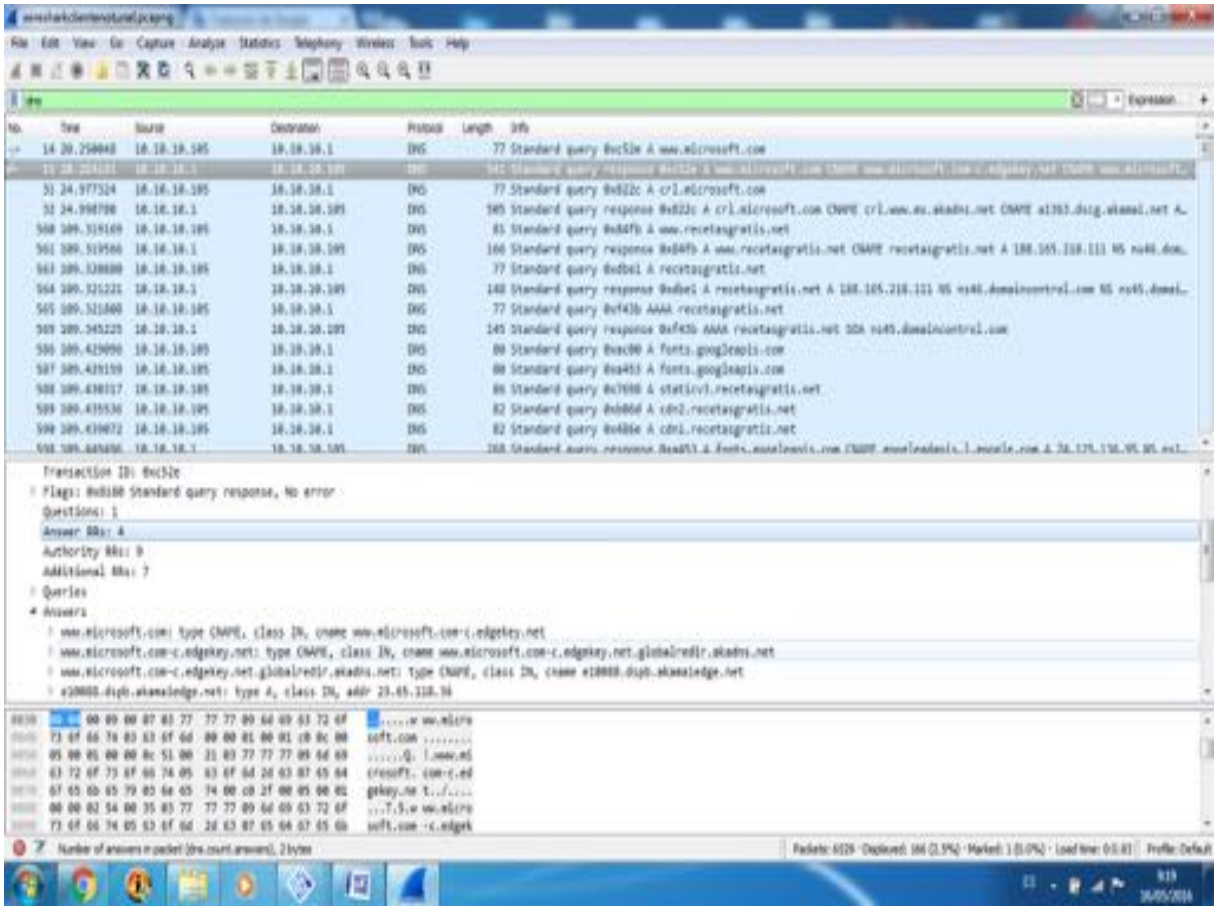

Figure 4.4: Capture Wireshark, in network without DNS tunnel.

In figure 4.2 we can see that more than 80% of DNS packets are unknown like I said above, because this picture show the capture in one network with DNS tunnel active, but in figure 4.4 that is a capture in a network without DNS tunnel the kind of packets are normal queries, so we can appreciate easily the difference.

But now we must look the size of the packets and the difference of size between the kind of packet normal and unknown, first we can look in figure 4.4 where we see that the average size of normal packets of DNS approximately, for the packets of request is around 80 bytes, and for the packets of responses, depends of numbers of responses in the packet but for example in the figure 4.4 in the packet that appears mark in grey, this packet have four answer and its size is 541 bytes and also in the same figure 4.4 more down appears some response packet and its size is 505 or more down another packets with size between 145 bytes and 170 or 200 bytes, is because this packets have less responses.

So if we compare these sizes with the size of the packets in the figure 4.2, where the DNS tunnel is active, we see that for example the packet marked in grey have a size 781 bytes and down of this we can see others packets with size around 420 bytes and also others more packets with size around 250 and other with 515, so the packets in this capture are biggest.

Also in figure 4.3 that show the expanded content of one unknown packet belonging capture of figure 4.2, and is marked in blue the content of the packet belonging to DNS protocol, the first bytes that are not marked in blue belong to another protocols like Ethernet, IP or UDP. As we can see more than 80% the content of the packet belong to DNS protocol.

Just looking at the number and size of packets that occur on a network with an active DNS tunnel we can have a good indication that something strange is happening on our network, the number of packets of unknown type which are detected besides its size is unusual, as some are smaller or normal 100 or 200 bytes but others come to reach a size of 700 or 900 bytes which is too much information to an unknown or malformed DNS packet protocol.

After detecting that our network is happening something strange, we need to see in which moment the Wireshark begin to capture the unknown packets, so in the figure 4.5 we can see the begin of capture in the network where the DNS tunnel start to run, and like is show in the picture with the packets marked in black, we need to see the query that these do.

The kind of the packet is standard query, but if we look the domain name that these consult, we can see that are so strange.

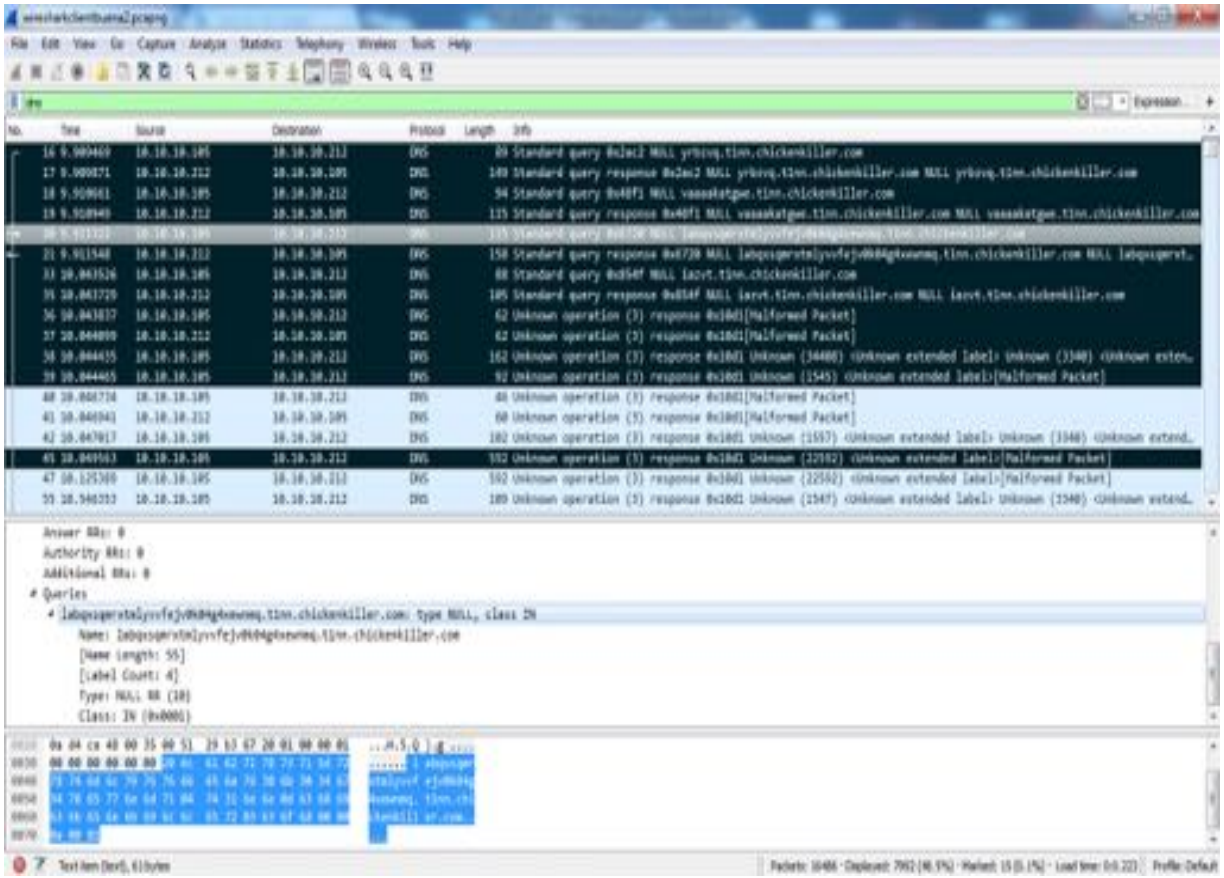

Figure 4.5: Beginning of DNS tunnel

DNS requests that are observed before starting to receive packets of unknown type are a little strange, they try to resolve these strange domain names:

```
-labqxsqmrxtmlyvvfejv0k04g4xewnmq.t1nn.chickenkiller.com: NULL type, class
IN
-yrbzvq.t1nn.chickenkiller.com: NULL type, class IN
-vaaaakatgwe.t1nn.chickenkiller.com: NULL type, class IN
```

We can see that the hostnames, the first part of the FQDN, are too strange and if we remember the tips in the second chapter of this document, where I talked about usual domains names, are remember easily by humans, and these hostnames like "yrbzvq", "labqxsqmrxtmlyvvfejv0k04g4xewnmq" and "vaaaakatgwe" don't signify nothing readable for the humans.

In addition, they all belong to the same domain that is t1nn.chickenkiller.com therefore this is strange.

In figure 4.6, we can see the result after choose the option expert information in Wireshark, in one capture of network where is working one DNS tunnel, in the first line the program show us that it count is 7470 DNS packets of type unknown, we see that there are 4 packets from PNG protocol of type unknown and 6 packets from TCP protocol and also type unknown. These packets appears for

some error or something sporadic but is not normal detect 7470 unknown packets and all of them from DNS protocol.

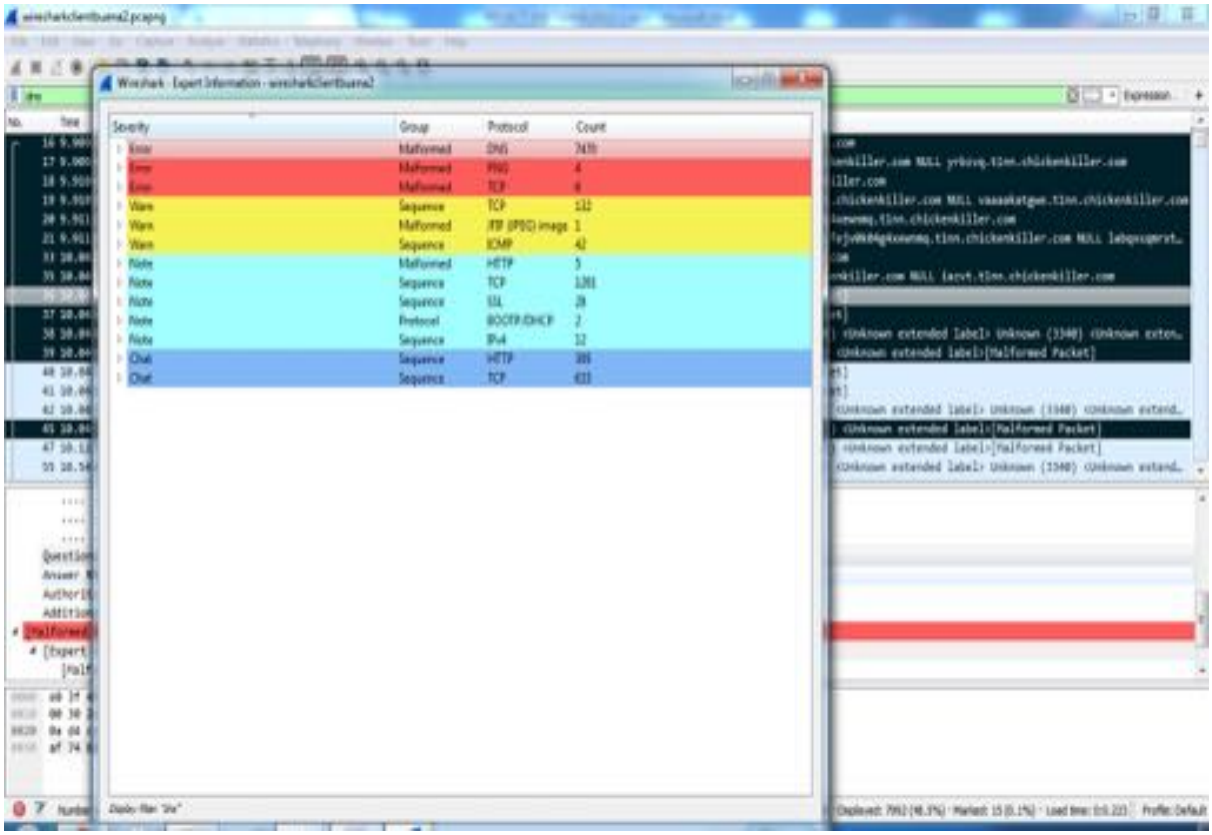

Figure 4.6: Expert Information in Wireshark, detecting amount unknown packets with DNS tunnel

By comparing this image with the same option on a network where it is not running a DNS tunnel, figure 4.7, we see we get some malformed packet which is normal, it can belong to any protocol, and some miscommunication or some error occurred specifically at some point time may receive some malformed packet, but not a lot as we have seen before 7470, in the following image the expert option information is displayed in a packet capture network where DNS is not working the tunnel.

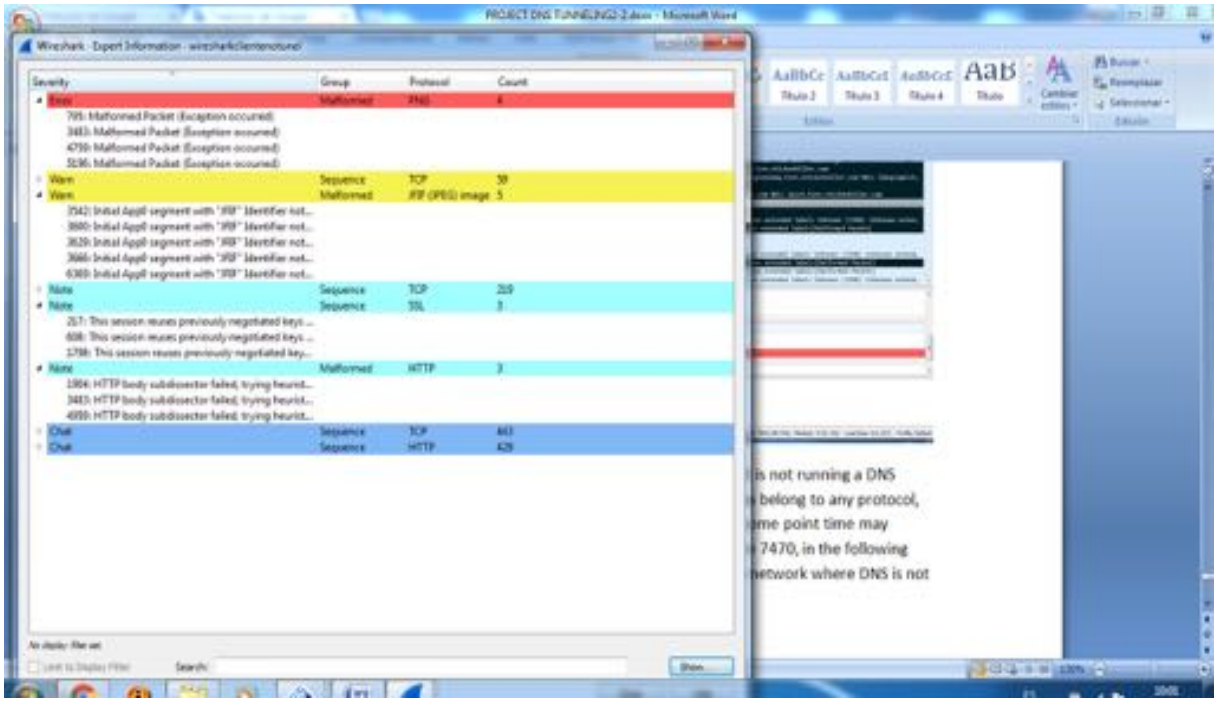

Figure 4.7: Expert Information in Wireshark, detecting amount unknown packets without DNS tunnel

So after see the figure 4.7 where is show capture of network without DNS tunnel, we see that appears some unknown packet also in a network without DNS tunnel but this is normal like I said before this can appear because some error or problem in communication, we see only 4 unknown packets and besides these are not from DNS protocol, these are from PNG protocol.

After see the 2 images, figure 4.6 and figure 4.7, and compare is easy to recognize that something wrong is happening with the DNS protocol in the network if we get information similar to figure 4.6.

So if we use the option expert information in Wireshark, the program show us important information and the program warns about the protocol DNS is not working good, following the tips mentioned in chapter 2 in this document and with our knowledge about how would be normal behavior of one normal network without DNS tunnel, with this information we begin to suppose that in our network may exist the DNS tunnel active.

To be more safer the next step is check the amount of DNS traffic and compare between one network without errors and another network suspicious or with strange DNS traffic, and this can seeing the graphics generated by Wireshark, as it will show in next images, figure 4.8, figure 4.9 and figure 4.10.

Graphs with tunnel DNS:

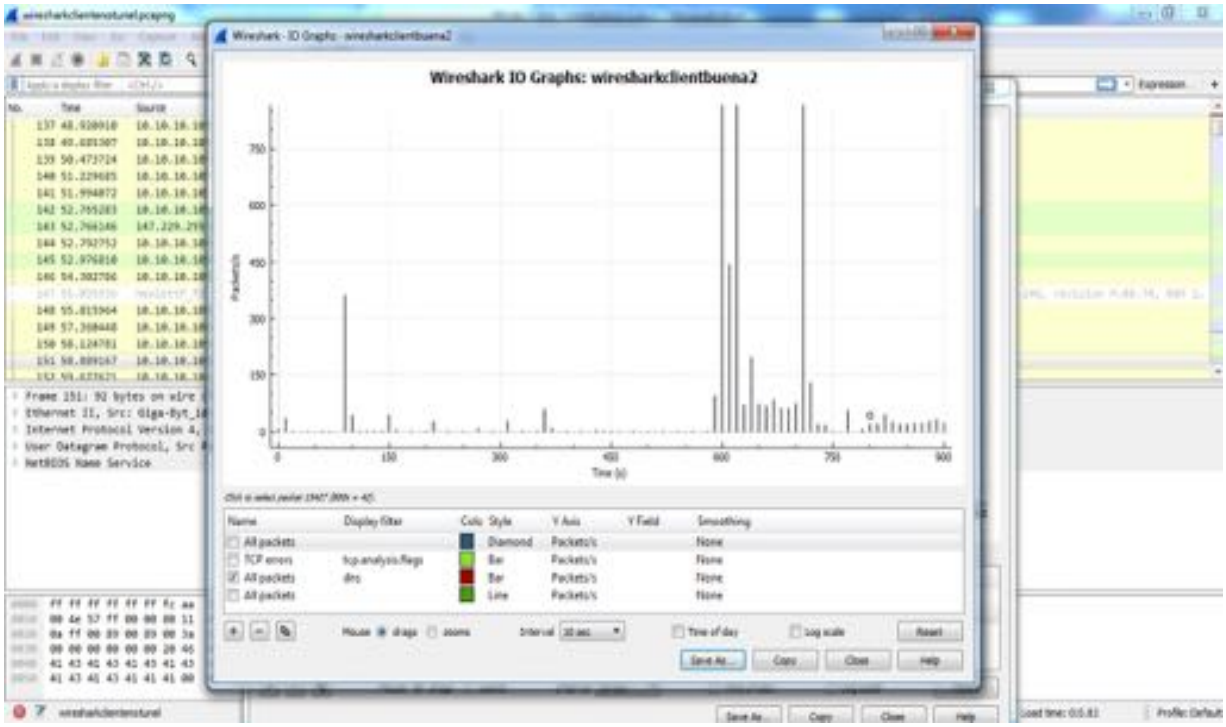

Figure 4.8: Traffic DNS with DNS tunnel

Graphs without tunnel DNS:

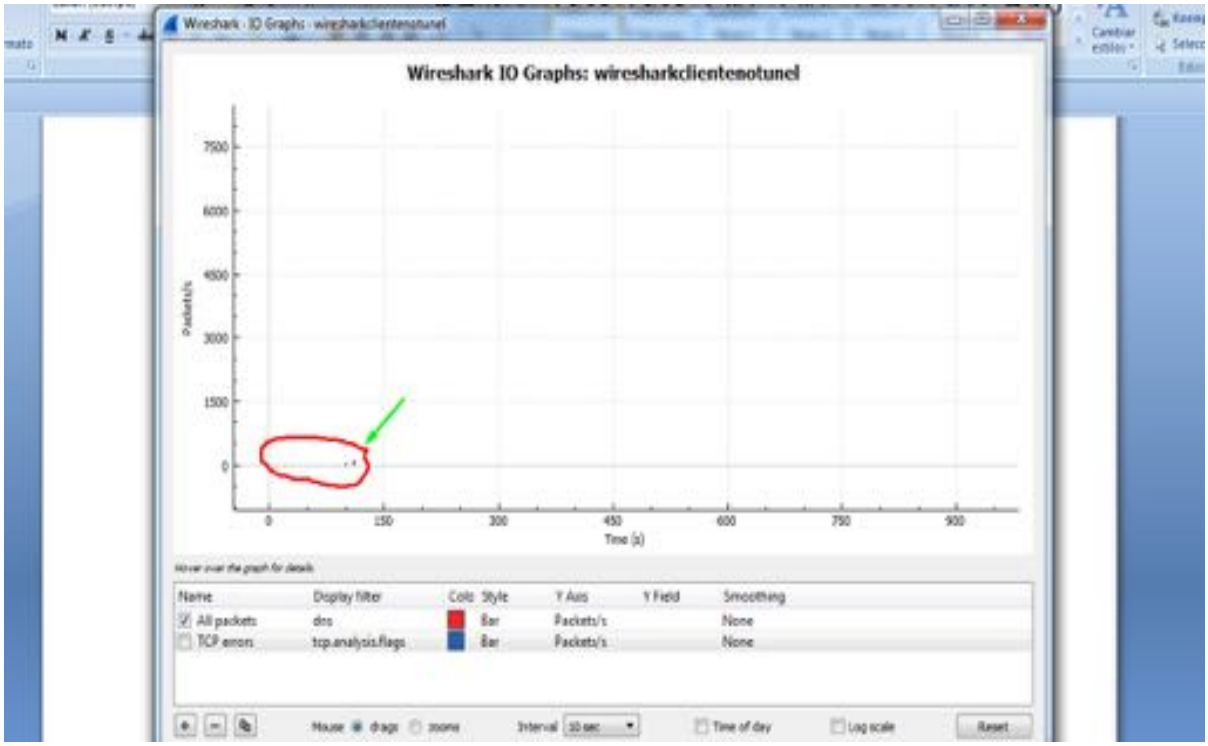

Figure 4.9: Traffic DNS without DNS tunnel

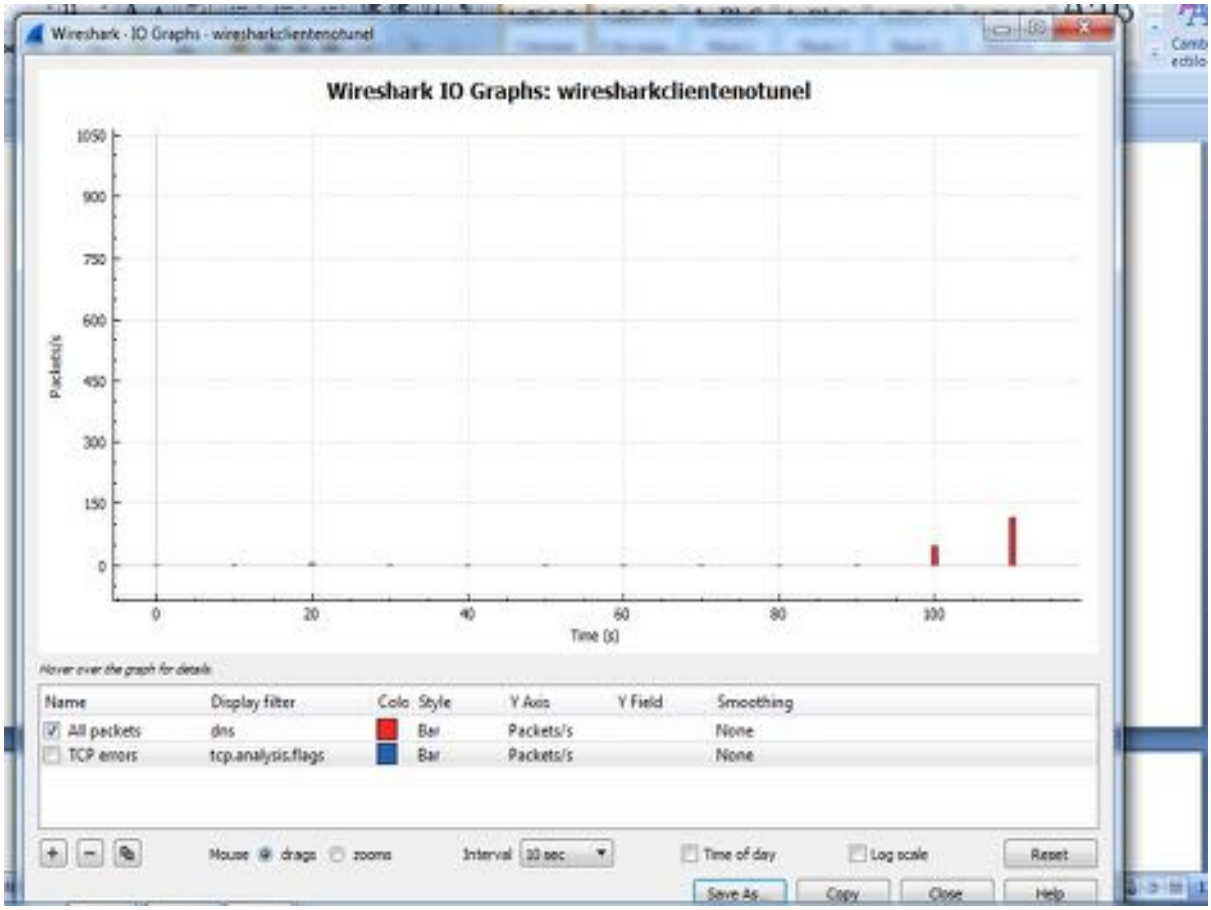

Figure 4.10: Traffic DNS without DNS tunnel, extended

The figure 4.8 and 4.9 shows the graph the same network with DNS tunnel active, and without this, measured with the same interval time and displayed each of 10 seconds information.

We can see how in just 750 seconds there are a big difference between this 2 graphs, for example in figure 4.8 network with DNS tunnel active, the amount of DNS traffic is bigger than 750 packets or even 1000 packets, in some moments see the time between 600 and 750 seconds.

Compare this with figure 4.9 where is shows the amount of DNS traffic in the same network without DNS tunnel, and in 750 seconds only near to second 100 exist a little amount of DNS traffic that is almost 150 packets.

So with this analysis we know that in our network exists:

- A big quantity the DNS traffic, only the network with DNS tunnel.
- That the most of the packets are type unknown, only the network with DNS tunnel.
- All of this unknown packets are from the communication between 2 computers that always are the same, in my case 10.10.10.105 and 10.10.10.212.
- That the size of these packets can be big, more than usual.
- Also the first packets before begin unknown packets performs strange queries .
- All of this queries ask for the same domain, in this case t1nn.chickenkiller.com, and strange hostnames.

All of this points are big evidences for exist DNS tunnel in our network, but this amount of unknown packets, we don't known what are they doing yet, so the next step is analyze the packets with more deep.

4.2 Analyze in depth DNS packets

To analyze unknown types packages, Wireshark provides an option that will be very useful, this option is "Follow UDP stream".

This menu item Brings up a separate window and displays all the UDP segments captured That are on the same UDP connection as a selected packet, which shows the contents of the package with this option analyzing several packages of unknown type will detect that finally these strange packages strangers who belong to the DNS protocol are discussed in reality communication DNS tunnel, below some pictures showing are option 2 selected packages, one is a normal package DNS traffic and the other is a package of unknown type or malformed, where you will see that belongs to the DNS tunnel.

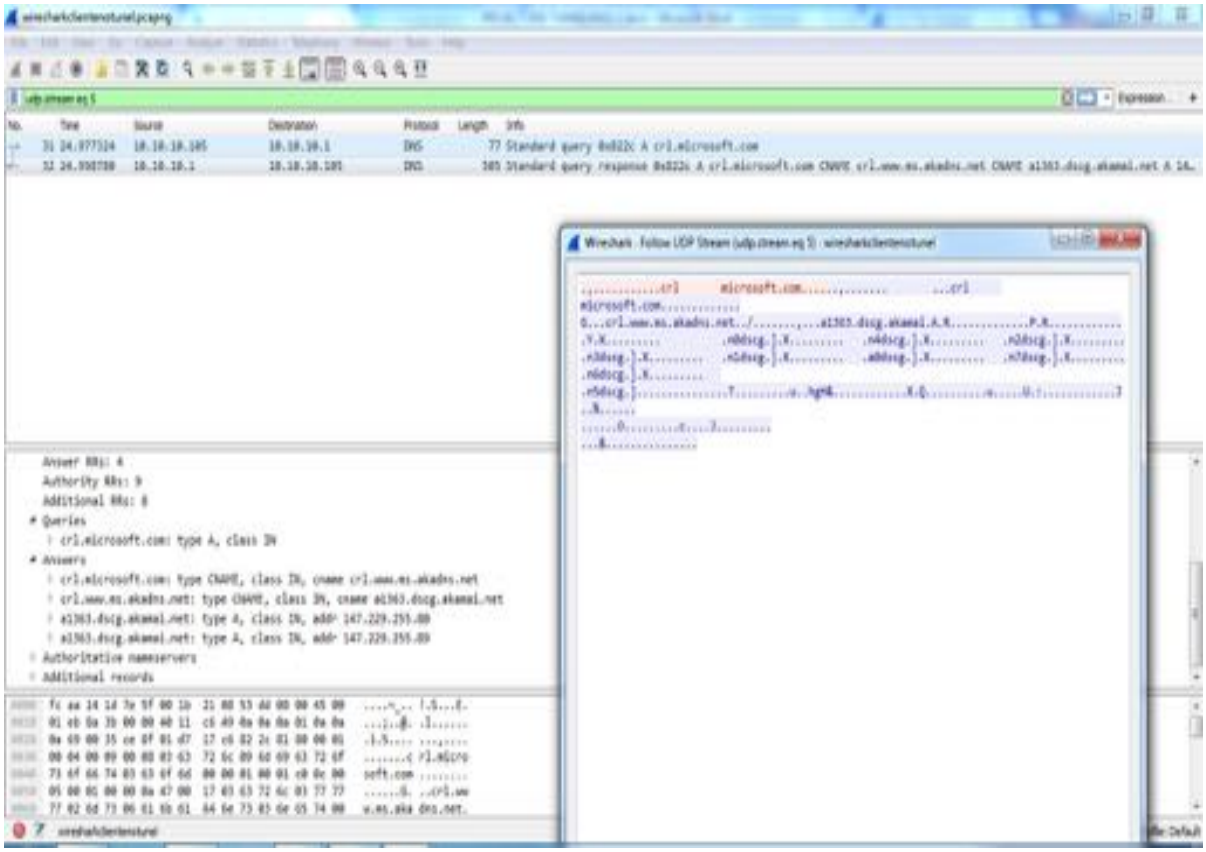

Figure 4.12: Content of normal DNS packet

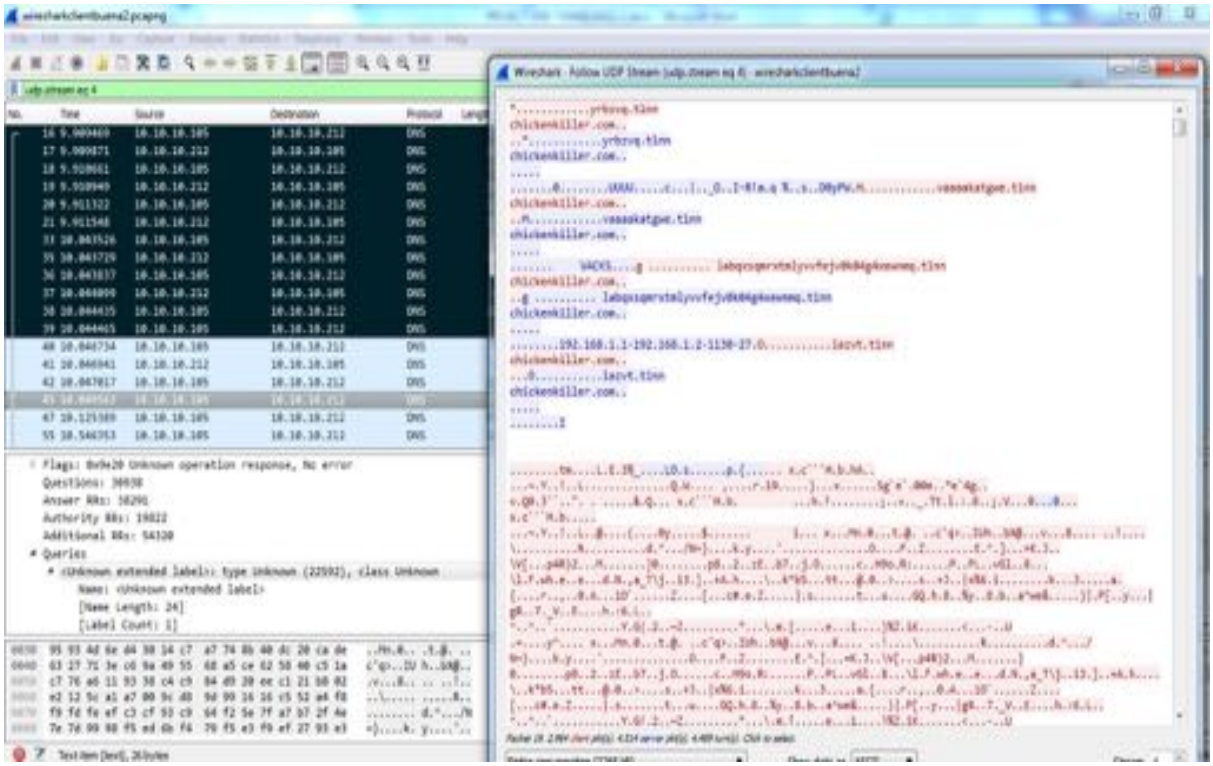

Figure 4.13: Content of unknown DNS packet, in DNS tunnel (Header)

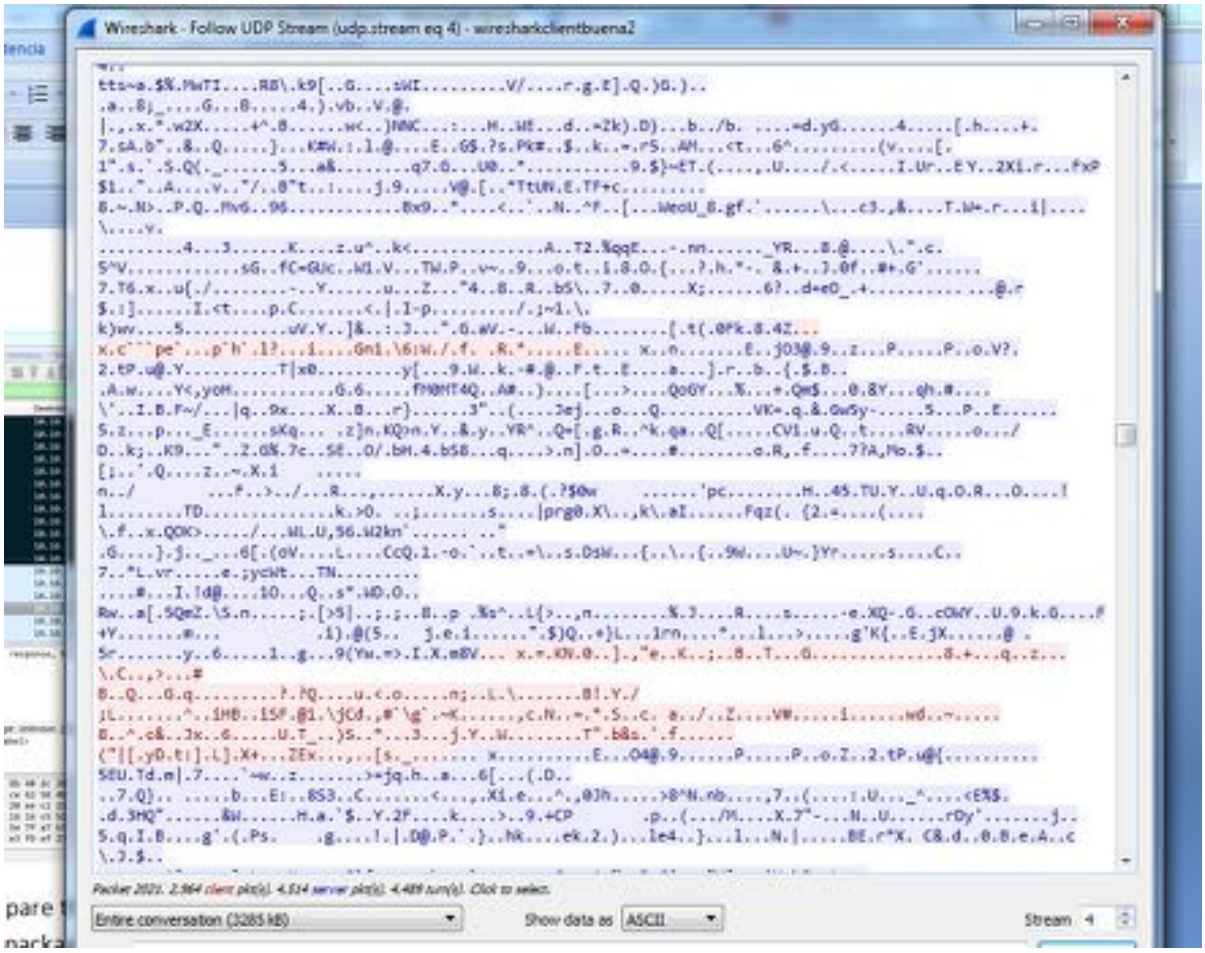

Figure 4.15: Content of unknown packet, in DNS tunnel (Load)

Finally, we can compare the contents of a normal DNS packet and a packet of unknown type of DNS, the standard package is much smaller and contains less information, also seen as queries to normal and readable domains for humans, in this case is a query referred to a server Microsoft.
But if you look at the package of unknown type, the first telltale belonging to the tunnel, in the first lines we see as are references to foreign domains that were previously identified as:

```
* ............ Yrbzvq.t1nn
chickenkiller.com ..
............ .. * Yrbzvq.t1nn
chickenkiller.com ..
.....
....... 0 ........ UUUU ..... c ... | .._ O..IR aq% s..D0yPW.H ...... ..!
...... vaaaakatgwe.t1nn
chickenkiller.com ..
............ ..h Vaaaakatgwe.t1nn
chickenkiller.com ..
.....
G .......... .... ....... VACKS labqxsqmrxtmlyvvfejv0k04g4xewnmq.t1nn
chickenkiller.com ..
.......... ..g labqxsqmrxtmlyvvfejv0k04g4xewnmq.t1nn
chickenkiller.com ..
.....
192.168.1.1-192.168.1.2-1130-27.O ........ ........... iazvt.t1nn
chickenkiller.com ..
... Or ........... iazvt.t1nn
chickenkiller.com ..
.....
........ I
```

All this we note that meets the feature of DNS tunnel, as sub domains xxxxxxxxxxxx.t1nn.chickenkiller.com query type, where the sub domain xxxxxxxxx is unreadable for a human is a meaningless word, also is always the same for all packages unknown type, which meets all these tips herein above, it looks like encrypted information and hidden nonsense is being transmitted hidden in the domain address, also if we look besides that the package is malformed contains a large unreadable amount of data, which is not normal for this type of packages, this information will be coded characters, depending on the encryption option that was used to launch the tunnel Iodine can be base32, base64 or base 128.

This last step confirms that all packets of unknown type that are transmitting refer to domain t1nn.chickenkiller.com, besides all these packages are communications between two computers or ips, which are always the same in this case 10.10.10.101 and 10.10 .10.212.

Finally we know that in our network exist one DNS tunnel active, we cannot know perfectly what are they transmitting because the information is encoded, but we know how is performed, by DNS protocol exactly with DNS tunneling , and also we know the source and destination of this communication, so let see how prevent this technique at least do more difficult the way for the unwanted user or evil user and how delete this technique in our network.

5. Discussion and Conclusions

In this chapter I will explain how prevent the DNS tunneling, how delete this technique or at least reduce the harmful for our network, because in the experiments performed here and besides all information mentioned here, as we saw too there are many ways to do the tunnel and maybe can exist others ways that wasn't mentioned here and unknown by the moment, we cannot confirm that one human may not invent others differents ways and methods for do it.

We have seen the advantages of using a DNS tunnel that allows us to communicate with the outside of our control equipment hidden from the other teams form, so this can be a problem for the network of a company in which we want to control all information our, because by this technique can be entered or sent outside of our network, important or privileged information we do not want this to be so. It depends on our internal configuration of the network of the company that this is easier to avoid or not, for example if we have a network in which all computers have a fixed IP, with which we can identify each computer, using the techniques explained earlier we can identify that team is doing the DNS tunnel, and block traffic from that IP, assuming that the client cannot obtain an IP DHCP, which would be the case. This can be done in several ways, the best way to have a secure network is to control all the computers on our network and keep them identified, this can be done through a firewall or configuring the DHCP to always assigned by the MAC the same IP to one computer. The bad thing of this method is that each time you connect a new computer to the network would have to register, but it depends if any pc that connect in our network, it can have access to our network equipment or not.

5.1 IP control and ports

For delete or prevent the DNS tunneling one of best ways is controlling of the IPs and ports in the computers in our network, because in one network that the users need internet access we cannot deny access to internet or DNS protocol for everyone, for this the best way is have a perfect control and identification of our computers.

In the moment that we have already detected the DNS tunnel, we can identify the computers involved in the tunnel.

With our firewall we can configure some rules for deny the access and communication with our network, for example with iptables some rules like these:

Following with the IPs in my experiments, ipsourcecomputer would be 10.10.10.105 and ipdestinationcomputer would be 10.10.10.212

```
iptables -A INPUT -s ipsourcecomputer -I eth1 -p tcp --dport 53 -j DROP
```

```
iptables -A INPUT -s ipdestinationcomputer -I eth1 -p tcp --dport 53 -j
DROP
iptables -A FORWARD -s ipsourcecomputer -I eth1 -p tcp --dport 53 -j DROP
iptables -A FORWARD -s ipdestinationcomputer -I eth1 -p tcp --dport 53 -j
DROP
```

But after locate the problem and add in our firewall some rules if we don't have good control about our computers, they can change their IPs, we have a trouble because can do the tunnel with another IP.

Also in the computer outside of our network we don't have control over it, it may change its IP and try again with another different, but if we blocked the another end of the tunnel, the computer inside of our local network, they cannot do the tunnel between them.

Even DNS tunnel can be created in other different port to 53 so we must be careful, but in one case that we have already detected the tunnel we know if is in different port because packets using another port and we see this with Wireshark then we can deny the traffic in other port and the tunnel will be finished.

In the worst case if they try to do the tunnel in different ports many times we can block all traffic of these computers in our firewall with the same rules without the parameter --dport 53, like I show below.

```
iptables -A INPUT -s ipsourcecomputer -p tcp -I eth1 -j DROP
iptables -A INPUT -s ipdestinationcomputer -p tcp -I eth1  -j DROP
iptables -A FORWARD -s ipsourcecomputer -p tcp -I eth1  -j DROP
iptables -A FORWARD -s ipdestinationcomputer -p tcp -I eth1 -j DROP
```

And also if we have control over the computers in our network will be better, that they don't can modify their ips, one way would be with static IPs in our local network but it may be annoying for big networks, other way may be control our DHCP server and link the IPs with the MACs, assigning the same IP for one MAC all of the time.

5.2 Controlling access as admin or root

Other point important to defend our network versus DNS tunnel, is the control in our computers like admin user or root user, as we saw above, for create a DNS tunnel is necessary install some programs, for example Iodine or even in Windows is necessary install a TUN/TAP interface, also when we execute the commands in Iodine or another program for create DNS tunnel, when the tunnel is created Iodine change some information in our computer, like the IP, or create a new virtual interface and one

private network where the information will be transmitted encapsulate with DNS protocol. But all of these operations cannot performed without privileges of admin or root.
Then if the users in our network only have access as normal users, they will can work good without any problem, and for us or the administrator of network will be better situation because less troubles will appear.
In this aspect we can't do nothing with the user outside of our network because we can't ban that he install his programs or he modify his network configuration.

5.3 Recursion control the DNS server.

If our network is a controlled environment, where the computers of our company need only consult information from some websites known or they only need access to intranet, not in all internet, we can do some configuration in DNS server from our local network for improve the security versus DNS tunneling, this will be disable the recursion in DNS server.
But if we want that this method to work, first we must meet the point above, the users can't modify the network parameters in the computers of our network or company, this can be meet restricting access like admin user o root user for every people and computers in our network, only the administrator of network or the boss of the company can have this kind of access. Even for the boss of the company this is not necessary because he can have other network configuration with access to internet but made by the network administrator. I will explain how do it.
Satisfy the point that all users can't change the IP or DNS of the computer.
We configure all computers in our network for do the queries only with the DNS servers from our network, our own DNS servers, and none more if we don't have totally control over it.
In our DNS server disable the recursion queries, with this method our DNS servers don't do queries to DNS servers in internet then will be impossible communicate with the fake DNS server for make the tunnel, but if is necessary to know some IP address of internet this will not be possible neither.
So with this configuration the users in our network and the computers only can access to IP address in our intranet because this information is from our zone and is stored in our servers, if is necessary some specific IP address to internet we can add this manually.
Also if one computer need free access to internet the network administrator can configured in the computer one DNS alternative outside of our network, or some DNS server that will be our property too, but with recursion queries active.
Then for disable the recursion in DNS server this depends of operating systems in DNS servers or the program that use these servers to provide the service DNS.
Can be different in all of them but for example for disable the recursion in one DNS server with Windows server 2008, will be this:

- Open administrator of DNS.

- In the tree of console, click with second button in the correct DNS server then open properties.

- Where?

- DNS/applicable DNS server

- Open the advanced options

- In options of server, active the box disable recursion and then accept..

Also if you want know more about the configuration one DNS server in Windows 2008 see here.

5.4 Conclusions

Finally, after seeing much related to DNS tunnels, from my point of view I can say that it is a technique very completely and complex. Is similar to another kinds of communication hidden or encoded like VPNs, if we understand good, the DNS tunneling is one VPN between two computers where the information is transmitted encapsulated with DNS protocol.

I talked too about the different protocols that can be used the similar method to perform the same purpose, like HTTP tunnels or SSL tunnels.

In my experiments and specifically with Iodine the information is encoded with some method like mentioned above as Base32,Base64 or Base128 for example with the program Iodine that it was what I use, but it was mentioned too that exist others types of encoding information, like hexadecimal encoding or binary encoding.

Exist to many ways to sent the hidden information in packets DNS in different kind of records DNS, as TXT,A, MX, NULL,etc.

Also exist many tools or programs to perform this technique, I used Iodine, but many program was mentioned in this document, for me the best was Iodine but we know that there are others programs that allow this in differents ways.

Besides in this document I explained some uses that allow this technique, as skip firewall, hotspot, with bad configurations or just that they allow the DNS traffic, that in the most of cases the DNS traffic is allowed because is one of the principal and more basics protocols for the communication over internet, because of this always is allowed the DNS traffic in networks.

Some of uses of this technique can be harmful for our network if we want have a secure network, because allow to transmit hidden information and maybe this information is confidential or very important for our company so not is good that this information go outside of our company or network without anybody know.

After know the different techniques and tools or programs, I performed some experiment and I showed how it work.

Also I explained how we can detect when DNS tunneling is happening in our network, with good analysis of the network traffics and doing comparisons between normal networks and networks with DNS tunneling active and looking the correct evidences.

Finally I explained the method for reduce the danger, or delete completely in our network DNS tunneling.

After all of this I want to say that the security in the networks is a topic very difficult and complicated, because there are many protocols, many differents ways of communication and is impossible have a network or computer 100% secure, in this document I did the focus in DNS tunneling, but as I mentioned exists HTTP tunnels, SSL tunnels and not only techniques of tunnels, there are too many methods for skip firewall, proxies, steal information, transmit hidden information, etc.

But one good network administrator can do more difficult the way by hackers or unwanted users if he keep good and secure control in his networks, also perform several and periodic analysis of network traffic and detect all of things that can be strange in one network.

References

[1] Infosec potpourri network security monitoring (NSM), Size of request and response DNS. http://blog.vorant.com/2006/05/traffic-analysis-approach-to-detecting.html

[2] DNS for Massive-Scale Command and Control Kui Xu Member, IEEE, Patrick Butler, Sudip Saha, Danfeng (Daphne) Yao Member, IEEE.

[3] HTTP Tunnels Though Proxies - SANS Institute https://www.sans.org/reading-room/whitepapers/covert/http-tunnels-proxies-1202

[4] PSUDP: A Passive Approach to Network-Wide Covert Communication, Black Hat USA 2010. https://media.blackhat.com/bh-us-10/whitepapers/Born/BlackHat-USA-2010-Born-psudp-Passive-Network-Covert-Communication-wp.pdf

[5] Reroute traffic from an IP and IP and port to another port, by usemoslinux http://blog.desdelinux.net/redireccionar-trafico-iptables/

[6] Erik Ekman <yarrick@kryo.se> and Bjorn Andersson <flex@kryo.se>. Guide Iodine http://code.kryo.se/iodine/iodine_manpage.html

[7] Alejandro Ramos, tunneled DNS, another option with Iodine 0.5.x http://www.securitybydefault.com/2010/01/tunelizando-dns-otra-opcion-con-iodine.html

[8] Manuel Jimenez, How to mount our own tunnel server always online (Part 1) http://www.hackplayers.com/2014/09/montar-nuestro-propio-servidor-de-tuneles-1.html

[9] NorfiPC, How to block and prevent access to websites and web sites. https://norfipc.com/articulos/como-bloquear-impedir-acceso-paginas-sitios-web-internet.html

[10] SANS Institute InfoSec Reading Room, Detecting DNS Tunneling. http://www.sans.org/reading-room/whitepapers/dns/detecting-dns-tunneling-34152

[11] Wikipedia , iptables https://es.wikipedia.org/wiki/Netfilter/iptables#Destinos_de_reglas

[12] Firewall with iptables http://wiki.elhacker.net/redes/administracion-de-redes-gnu-linux/firewall

[13] How work the DNS queries https://msdn.microsoft.com/es-es/library/cc775637(v=ws.10).aspx

[14] Disable recursion in DNS servers. https://technet.microsoft.com/es-es/library/cc771738(v=ws.11).aspx