# Global Information Assurance Certification Paper

# Covert Channels

*GIAC (GCIA) Gold Certification*

Author: Erik Couture, erikcouture@gmail.com
Advisor: Adam Kliarsky

Accepted: August 19, 2010

*Abstract*

*The concept of covertly passing data over a communications channel has existed for hundreds of years. The advent of interconnected computer networks employing intricate layers of protocols created a new medium through which to covertly pass data. This paper explores covert channels on computer networks and examines current and possible future threats. It assesses the risks to individuals and enterprises, and proposes countermeasures to help detect and mitigate the risks brought on by these covert channels.*

# 1. Introduction

Historically, the expression "covert channel" has broadly encompassed all communications that are hidden and communicate stealthily between endpoints. The goal of such a channel is not necessarily to obscure the data flowing through the channel, but to obscure the very fact that a channel exists. Often this data may be passed in plain sight of possible observers, but if properly engineered, may remain nearly impossible to detect. Covert channels represent a pure example of security through obscurity.

## 1.1. Definition

The term 'covert channel', when applied to computer networks, describes a mechanism for sending information without the knowledge of the network administrator or other users. Depending on the context, it has also been defined as:

- a transmission channel that may be used to transfer data in a manner that violates security policy (Van Horenbeeck, 2010).
- a means of communication not normally intended to be used for communication (Zander, Armitage & Branch, 2007).
- a mechanism for sending and receiving information data between machines without alerting any firewalls and IDSs on the network (Buetler, 2008).

Indeed any of these definitions is correct in a particular situation, but together they help paint a picture of the overall dangers that covert channels may bring to the network environment; namely a set of technologies which allows for the unobserved transmission of data over legitimate communications means.

## 1.2. It's not encryption

It is fundamental to recognize the difference between covert communications and encrypted communications. In the latter, the data stream is (ideally) unintelligible and irretrievable by an unauthorized party. Encrypted communications however do not hide the fact that a communication took place, or the identity of its originator and destination.

Encrypted data streams will often give up valuable information about themselves through their inherent characteristics which may be extracted through careful traffic analysis. Covert communications are tunneled in normal, authorized traffic using techniques that make them largely undetectable to examination by administrators and network filters. Covert channels in computer network protocols are similar but distinct to steganography, the hiding of information in audio, visual, or textual content. While steganography requires some form of content as cover, covert channels require some network protocol as carrier (Zander, Armitage & Branch, 2007).

## 1.3. Motivations

Many groups and individuals could be motivated to keep their communications secret. Criminals, hackers, nation-state and corporate spies, privacy minded individuals, sysadmins and savvy users might seek to use these channels to:

- Exfiltrate data from an otherwise secure system
- Avoid detection of unauthorized access
- Perform legitimate network management functions (Forte, Marti, Vetturi & Zambelinni, 2005)
- Install, spread or control malware on compromised systems
- Circumvent filters which may be in place limiting their freedom of speech
- Bypass firewalls for unrestricted access to the web

In the past decade, there has been an enormous increase in the popularity and proliferation of the Internet Protocol Suite (aka TCP/IP) as the primary suite of network protocols for the interconnection of computer systems. Much research has been conducted in covert channeling by use (or misuse) of its component protocols. In particular, core protocols such as HTTP, ICMP and DNS have all shown the ability to act as clandestine mediums for covert traffic. These protocols were designed well before security was a primary concern and thus have much vulnerability that allow for creative misuse, within the scope and limitations of their RFC specifications.
We will see how it is possible to covertly encapsulate blocked or controlled protocols such as peer-to-peer, chat and SSH inside other protocols expressly allowed by security

policies, thus effectively neutralizing many security restrictions enforced at the network edge (Dusi, Crotti, Gringoli & Salgarelli, 2008).

## 2. Scope

While this paper will focus specifically on covert channels through common computer network protocols, the principles are similar to those of other related areas of research, including the covert data passage over computer control and timing circuits, as described by Lampson in 1973 (Sheets & Koot, 2006), in what is generally accepted as the earliest work on computer-based covert channels.

There are a nearly infinite number of variations with which data can be encoded onto a given medium. This is particularly true of TCP/IP networks, as their intricacies provide a plethora of possible data-carrying channels. Most of the currently exploited methods exist in the Network and Application layers of the network stack, and accordingly this paper will focus primarily on those. This paper will examine several examples of existing technology that allow for the creation of covert channels. It will demonstrate how they are engineered and how they can be used to penetrate network defenses. The paper will discuss the observable characteristics of these and other similar channels in operation, and how these telltale signs can be leveraged to create effective detection and countermeasure systems.

## 3. Vulnerabilities in TCP/IP

### 3.1. Methods of covert data encoding

There are several methods of encoding data on a host communication packet stream. For clarity's sake, the following explanations will make use of the IP header (Fig 1) as an example, but the same principles are true using other protocol headers. Each packet contains a number of header fields which can be misused to traffic data instead of (or as well as) managing the network flows (Van Horenbeeck, 2006). By employing one of the following techniques, data can be transferred covertly between endpoints.

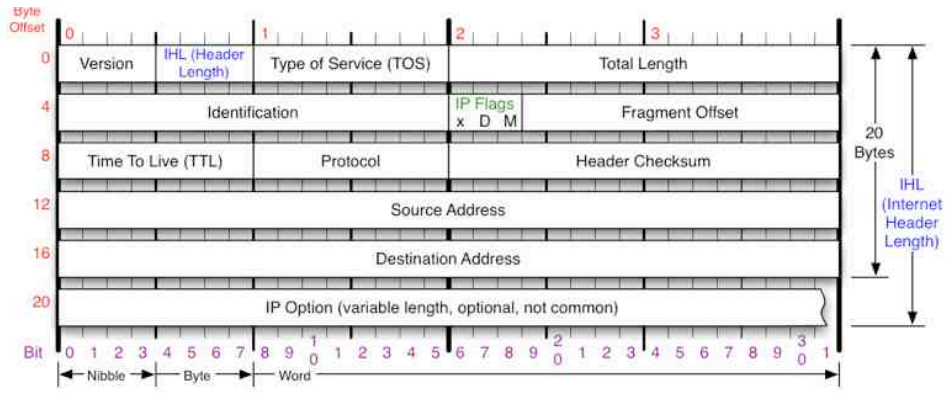



**Figure 1 – IPv4 Header (Baxter, 2008)**

### 3.1.1. Header bit modulation

Data may be stored and transmitted in a number of ways. It can be encoded in a human readable format, such as ASCII, or in hex or binary form. Binary, the simplest representation of all, allows the transmission of data using simple modulation of any single network packet characteristic; if a certain bit in the packet header is 1 or 0, the covert channel receiver interprets that data, regardless of the other characteristics or payloads of the data packets. Covert data may be encoded as binary in the least significant bit (LSB) of the IP ID or Type of Service, or any other bit which is not critical to the proper transmission of the host protocol. This type of basic binary field modulation would yield a net covert channel bandwidth of 1 bit per packet, which may be considered far too inefficient for practical use in many environments.

### 3.1.2. Header bit crafting

Better efficiency, in terms of covert bits passed per total bandwidth, can be achieved by adding additional bits per packet of legitimate network traffic. This may be accomplished by crafting header bits, or entire bytes where possible. For example, the IP ID field is randomly generated on many systems and could easily be crafted to represent an ASCII value or any other data (Zander, Armitage & Branch, 2007).

### 3.1.3. Optional header extension

Even greater gains can be achieved by employing protocols that allow for optional header fields. These fields are appended to the packet headers and are designed to carry additional data or instructions. They may be misused and made to act as an expandable

bearer for illicit traffic. It is this method which allows for the greatest amount of relative covert data per legitimate traffic, as option headers and data field can span the remainder of the unused maximum packet length (e.g. 65535 - 20 byte IP hrd = 65515 byte payload) A simple mathematical comparison of the methods above shows theoretical efficiencies.

| | Covert bytes/packet | Packet Size | Efficiency |
|---|---|---|---|
| Modulation | 1 bit | 20 | 0.63% |
| Crafting | 1 byte | 20 | 5.00% |
| Hdr Options | 100 bytes | 260 | 38.46% |
| | 65515 bytes | 65535 | 99.97% |

**Table 1 – Theoretical efficiency ratio**

It's clear that varying types of data encoding will yield drastically different throughputs, but there can be negative side effects as well, as will be explored in the Detection and Countermeasures section of this paper. Although it will be discussed in more detail in the Detection and Countermeasures section, it is worth noting that the more covert data is secreted into each legitimate data packet, the greater the chances of the channel being detected. Greater efficiency does not necessarily equal greater effectiveness.

### 3.1.4. Temporal Channels

While primarily in existence as an academic proof of concept, it has been demonstrated that temporal (time based) channels offer possibilities to channel covert data. The order which packets are sent in can be manipulated in such a way that their arrival order in itself can be encoded with data (Ahsan & Kundur, 2002). This method is somewhat less reliable as, due to network routing, there is no guarantee that packets will arrive in the order they were sent, and there is therefore a risk of out of sync packets. Additional logic would need to be added to the system to account for network congestion, latency and jitter (Sheets & Koot, 2006). On a highly reliable network, such as a LAN, with a limited number of nodes and routers this concept could prove effective.

## 3.2. Exploitable Protocols

Although nearly any protocol can be used to channel covert data, there are practical reasons to emphasize certain ones. The vast majority of research on covert channels has focused on layer 3-4 (Network and Transport) protocols such as ICMP, IP and TCP.

Layer 7 (Application) protocols such as HTTP and DNS are also commonly used. The primary factor in selection of these protocols is not only their prevalence on the Internet, but also the standard practice of allowing these protocols through network protective devices by default, in order to enable legitimate client applications.

## 3.3. Protocols on Protocols

Anyone having studied networking technology should be familiar with the concept of tunneling data over a network protocol. A common example is that of IPSEC, which encrypts the IP packet before wrapping it in an unencrypted, routable IP header. The fundamentals of covert channels are somewhat similar, but rather then passing network traffic enclosed within an encrypted payload, the covert traffic is encoded within some part of the legitimate traffic. It is only seen by a recipient who is specifically looking for that particular non-standard characteristic in the network traffic. In other terms, unless the recipient is specifically programmed to extract the encoded data in the exact way it was embedded, the traffic will be forwarded along normally and obliviously through switches and routers to the destination host. This model requires a server-client approach: a host, generally on the inside of a protected network that connects to a server on the outside (Dusi et al, 2008). Both ends must be deliberately programmed to understand the particulars of the encoding scheme or neither will be able to communicate with the other. If the server outside the protected network is not the final destination of the traffic, it can extract the embedded traffic and forward it along un-encapsulated to its destination (e.g. the internet).

This behavior, similar to a proxy, can effectively allow illicit data to egress over an unblocked host port, bound for an allowable destination port (such as 53 or 80), effectively subverting controls such as network and host-level firewalls. Covert channels may be crafted to take advantage of any protocol or packet, which is generally given access through network security devices; SYN, RST, ICMP etc. Intrusion protection devices often simply search for telltale strings or patterns in payloads, but may not recognize the subtle variations of data encoded in a header field. Content filters might

also be evaded as they commonly search for sensitive information in payload fields, but not encoded, say, in ASCII into a series of header field TCP Sequence numbers.

### 3.4. How it works in practice

One of the design challenges of tunneling a protocol encoded within another is to replicate the behavior of a fully featured network protocol on another protocol with completely different communications model and behaviors. For example, designing a covert 2-way channel riding on a connectionless protocol such as ICMP requires creative use of existing protocol standards. For effective 2-way communications, methods must be developed to handle TCP handshaking, packet retransmission and fragmentation. Relatively large allowable packet sizes in the ubiquitous Ethernet protocol, combined with high-speed interconnections create an environment capable of usefully fast data communications, if only by encoding a few bits within each packet (Zander, Armitage & Branch, 2007).

The following sections will explore specific exploits that generate reliable TCP connections over ICMP and DNS protocols.

## 4. ICMP Covert Tunnels

### 4.1. ICMP Review

Internet Control Message Protocol (ICMP) is one of the fundamental protocols in the IP suite. It communicates at layer 3, the Network layer, of the network stack and operates on a datagram basis, much like UDP. As such, it has no built in mechanisms for transmission control or guaranteed delivery (Doug, 2008). ICMP is primarily used for sending control and error messages between endpoints, but with the exception of ping and traceroute, is not generally directly employed by the end user. Rather, ICMP is usually generated as a result of a transmission failure or diagnostic effort by another network protocol.

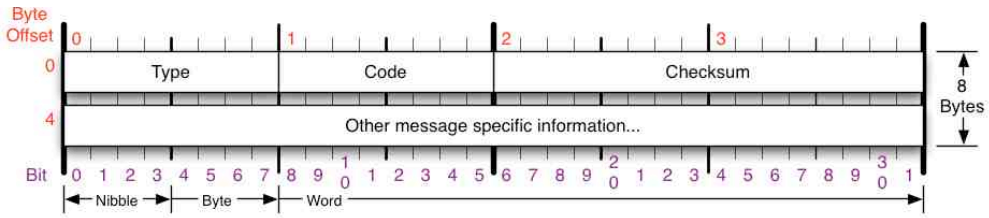



**Figure 2 – ICMP Header (Baxter, 2008)**

The key components of the ICMP header, as shown in Fig 2, are the Type, Code, Identifier and Checksum fields. Type and Code designate the purpose of the ICMP packet, the most common of which are listed in Table 2 below. The Checksum is a verification value calculated on the content of the ICMP header and payload.

| Type | Code | Purpose |
|---|---|---|
| 0 | 0 | Echo Reply |
| 3 | 1 | Host unreachable |
| 3 | 3 | Port unreachable |
| 8 | 0 | Echo Request |

**Table 2 – RFC 792**

Various ICMP messages types will use the 'other' space from bytes 4+ of the ICMP header in different ways; the Destination Unreachable message for example will append the IP header and first 64 bits of the datagram that caused the error to be generated (RFC 792). Most important for our analysis of ICMP covert tunnels is the ping packet that is laid out as per Fig 3.

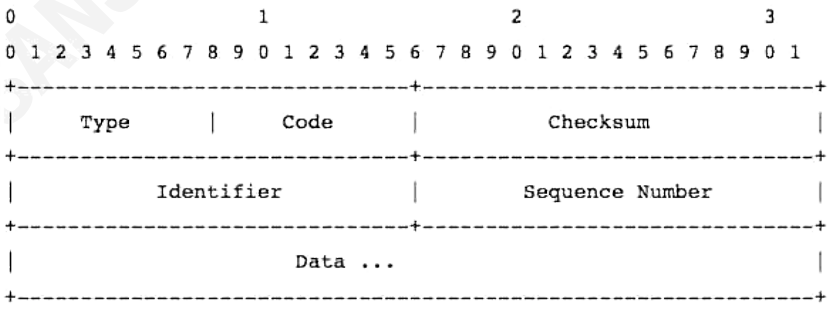



**Figure 3 – ICMP Header (Doug, 2007)**

The Identifier and Sequence fields are generated by the sender and serve to match a ICMP message to its reply. The Data field, however, carries a payload defined by the

sending operating system. In Linux (Fig 4), the ICMP Echo Request Data field is padded with 56 bytes of numeric digits, while Windows inserts 32 bytes of the alphabet (Gregg, 2007). (Fig 6) The maximum legal amount of data that can be stored in the payload field can be calculated as the maximum size of an IP packet minus the IP and ICMP headers: 65535 - 20 - 8 = 65507 bytes. Using LINUX ping, arbitrary data can be placed in the payload by simply specifying it following a –p switch.

When a host invokes the ping command, an ICMP Echo Request packet is routed to indicate the destination IP. When it arrives at the destination server, an ICMP Echo Reply packet is returned (Figs 5&7). Upon arrival at the originating host, the Sequence Numbers are matched and the time delay is displayed.

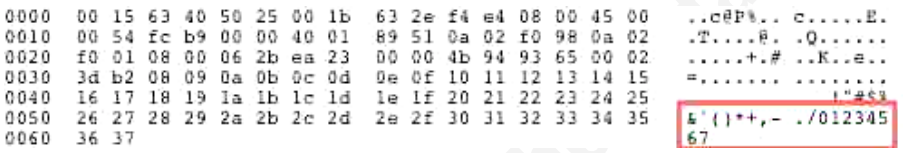



**Figure 4 - Linux Ping Request**

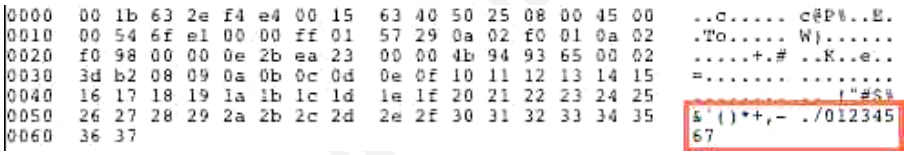



**Figure 5 - Linux Ping Reply**

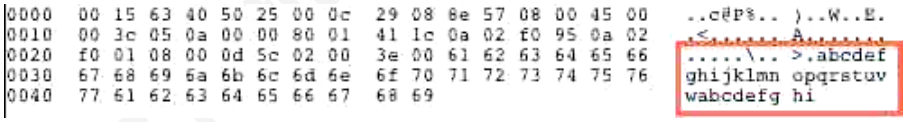



**Figure 6 - Windows Ping Request**

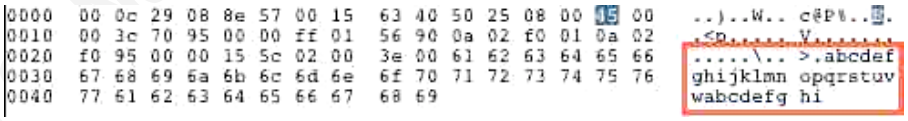



**Figure 7 - Windows Ping Reply**

As many common ICMP covert tunneling exploits rely on the ICMP echo-request/echo-reply as their attack vector, it is worthwhile having a detailed look at these packets, so as to better deconstruct the exploits themselves.

## 4.2. The Root of the Exploit

Perhaps due to the ubiquity and seemingly limited usefulness of ICMP, it is often forgotten when considering network security. Since ICMP is commonly allowed through firewalls for network troubleshooting purposes, the resultant hole provides a possible threat vector through which data can be covertly passed. Several exploits exist which leverage this vulnerability in order to circumvent firewall and proxy server controls. The essence of most existing exploits lies in the misuse and crafting of the ICMP header data field. This field is designed to return textual data in response to a network error, but the data is not verified or limited in any meaningful way. The payload can be crafted to hold any manner of data, which will piggyback on the ICMP packet (Van Eden, 2001). In order for a tunnel to be established, a receiving server needs to be installed on a selected host. This software will search for ICMP packets with specified data in the data field and interpret it in a predetermined fashion. The server can either act as an end point, or as a proxy, forwarding the traffic onward to its ultimate destination. So far, we've described the generic encapsulation method used by most ICMP covert channel software. The key difference between most of the available tools is the manner with which they embed a TCP or IP header and payload in the data field of the ICMP packet. For brevity, in the next section we will examine the traffic of a single exploit tool, ptunnel.

The client will always be using type 8 request packets for all its traffic; the proxy will always use type 0 reply packets for all its traffic. The reason for this is simple routing. Say that the client was behind a router, the router's operating system might itself reply to the proxy's echo requests without forwarding them to a host within its network that is involved in the ICMP tunnel session.

## 4.3. ptunnel Traffic Analysis

We will now explore packet captures from an ICMP tunneling tool, ptunnel. For the purposes of these analyses, a test lab was configured resembling Fig 8.

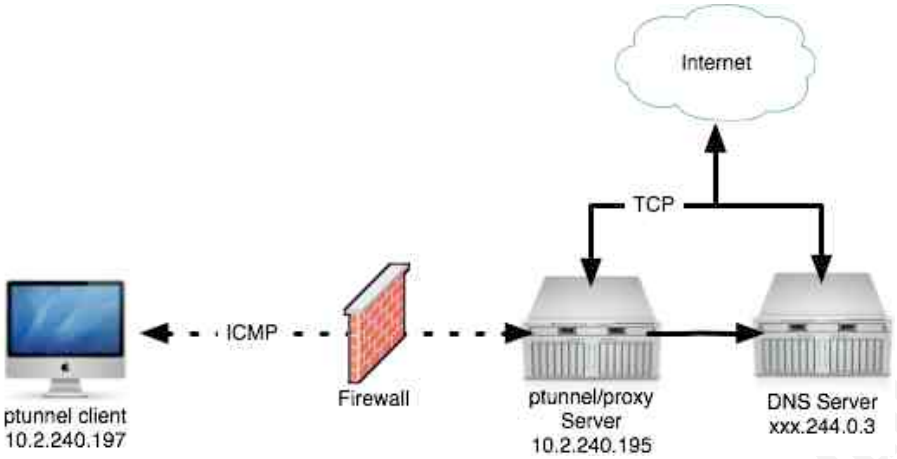



**Figure 8 – ptunnel lab setup**

The server is activated by simply running the ptunnel command from the shell prompt. Although not mandatory, a privoxy proxy service was enabled on the server in order to redirect all incoming traffic to the Internet. Should a user simply wish to access a local resource, such as an SSH server, this step could be omitted.

The tunnel was instantiated from the client with the following command:

```
# ptunnel -p 10.2.240.195 -lp 8765 -da localhost -dp 8118
```

The switches define the following parameters:

- -p 10.2.240.195 : the address of the ptunnel listener/server
- -lp 8765 : the local port on the client, through which traffic will be tunneled
- -da localhost : The loopback address on the server to redirect traffic to the privoxy web-proxy
- -dp 8118 : The server's privoxy web-proxy redirect port

Figure 9 is an IP flow graph representing the first few packets to pass through the tunnel. First we see ICMP pings and replies as the DNS request is tunneled, then an outbound DNS request to the DNS server. The requested webpage is returned from the web host and is finally returned through the firewall to the ptunnel client via ping replies.

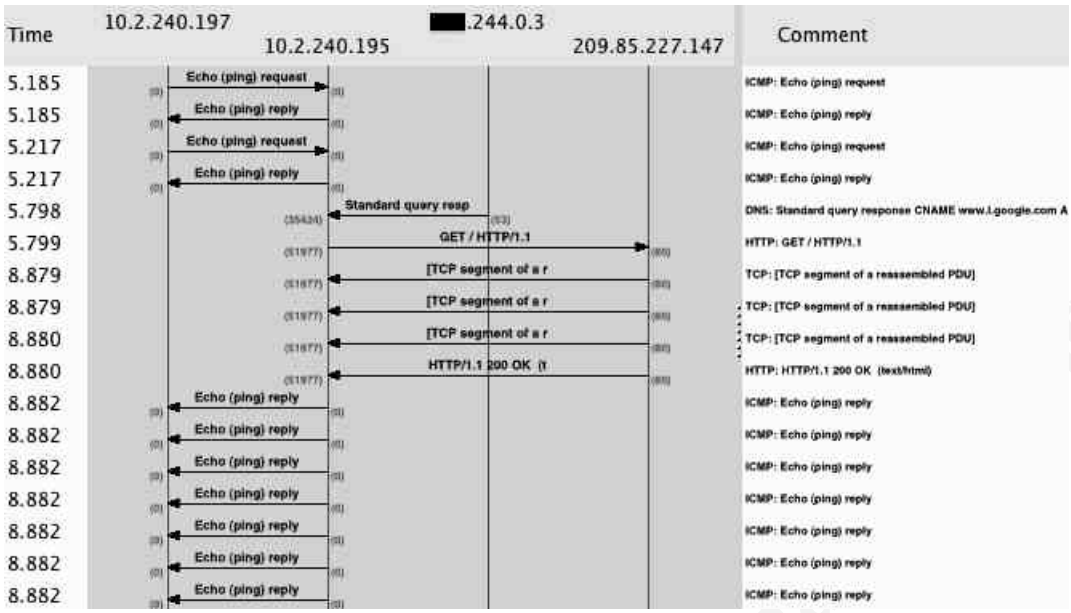



**Figure 9 – ptunnel IP flow graph**

Figure 10 depicts a Wireshark dump of the same packet stream from Figure 9. Here we can better observe the number of pings involved in transmitting the ~5k of data. We see that with the defaults settings, it took approximately 10 echo replies (#66-76) totaling ~10k to transmit the 4 packets of TCP traffic. This is in part due to the added overhead of the additional encapsulation ptunnel creates around the tunneled traffic.

| No. | Time | Source | Destination | Protocol | Info | Packet Size |
|---|---|---|---|---|---|---|
| 19 | 11:19:37.039393 | 10.2.240.197 | 10.2.240.195 | ICMP | Echo (ping) request | 770 |
| 20 | 11:19:37.039473 | 10.2.240.195 | 10.2.240.197 | ICMP | Echo (ping) reply | 770 |
| 30 | 11:19:37.071352 | 10.2.240.197 | 10.2.240.195 | ICMP | Echo (ping) request | 770 |
| 31 | 11:19:37.071399 | 10.2.240.195 | 10.2.240.197 | ICMP | Echo (ping) reply | 770 |
| 39 | 11:19:37.652033 | [redacted].244.0.3 | 10.2.240.195 | DNS | Standard query response CNAME www.l.google.com | 192 |
| 43 | 11:19:37.653062 | 10.2.240.195 | [redacted].85.227.147 | HTTP | GET / HTTP/1.1 | 738 |
| 54 | 11:19:40.733100 | [redacted].85.227.147 | 10.2.240.195 | TCP | [TCP segment of a reassembled PDU] | 1516 |
| 56 | 11:19:40.733240 | [redacted].85.227.147 | 10.2.240.195 | TCP | [TCP segment of a reassembled PDU] | 1516 |
| 58 | 11:19:40.733870 | [redacted].85.227.147 | 10.2.240.195 | TCP | [TCP segment of a reassembled PDU] | 1516 |
| 60 | 11:19:40.733941 | [redacted].85.227.147 | 10.2.240.195 | HTTP | HTTP/1.1 200 OK (text/html) | 283 |
| 66 | 11:19:40.735795 | 10.2.240.195 | 10.2.240.197 | ICMP | Echo (ping) reply | 1096 |
| 67 | 11:19:40.735906 | 10.2.240.195 | 10.2.240.197 | ICMP | Echo (ping) reply | 1096 |
| 68 | 11:19:40.735983 | 10.2.240.195 | 10.2.240.197 | ICMP | Echo (ping) reply | 1096 |
| 69 | 11:19:40.736059 | 10.2.240.195 | 10.2.240.197 | ICMP | Echo (ping) reply | 1096 |
| 70 | 11:19:40.736134 | 10.2.240.195 | 10.2.240.197 | ICMP | Echo (ping) reply | 1096 |
| 71 | 11:19:40.736209 | 10.2.240.195 | 10.2.240.197 | ICMP | Echo (ping) reply | 1096 |
| 72 | 11:19:40.736306 | 10.2.240.195 | 10.2.240.197 | ICMP | Echo (ping) reply | 1096 |
| 73 | 11:19:40.736381 | 10.2.240.195 | 10.2.240.197 | ICMP | Echo (ping) reply | 1096 |
| 74 | 11:19:40.736479 | 10.2.240.195 | 10.2.240.197 | ICMP | Echo (ping) reply | 1096 |
| 75 | 11:19:40.736552 | 10.2.240.195 | 10.2.240.197 | ICMP | Echo (ping) reply | 1096 |
| 76 | 11:19:40.736633 | 10.2.240.195 | 10.2.240.197 | ICMP | Echo (ping) reply | 1052 |
| 77 | 11:19:40.757953 | 10.2.240.197 | 10.2.240.195 | ICMP | Echo (ping) request | 850 |
| 78 | 11:19:40.758020 | 10.2.240.195 | 10.2.240.197 | ICMP | Echo (ping) reply | 850 |
| 92 | 11:19:40.778930 | 10.2.240.197 | 10.2.240.195 | ICMP | Echo (ping) request | 964 |
| 93 | 11:19:40.778952 | 10.2.240.195 | 10.2.240.197 | ICMP | Echo (ping) reply | 964 |
| 94 | 11:19:40.779003 | 10.2.240.197 | 10.2.240.195 | ICMP | Echo (ping) request | 850 |
| 95 | 11:19:40.779024 | 10.2.240.195 | 10.2.240.197 | ICMP | Echo (ping) reply | 850 |

**Figure 10 – ptunnel Wireshark Capture**

Finally, Fig 11 peers into a single packet, where we can clearly deconstruct the tunnel encapsulation. In red is the 20-byte IP header. Blue represents the 8-byte ICMP header, and the remaining bytes are ICMP payload.

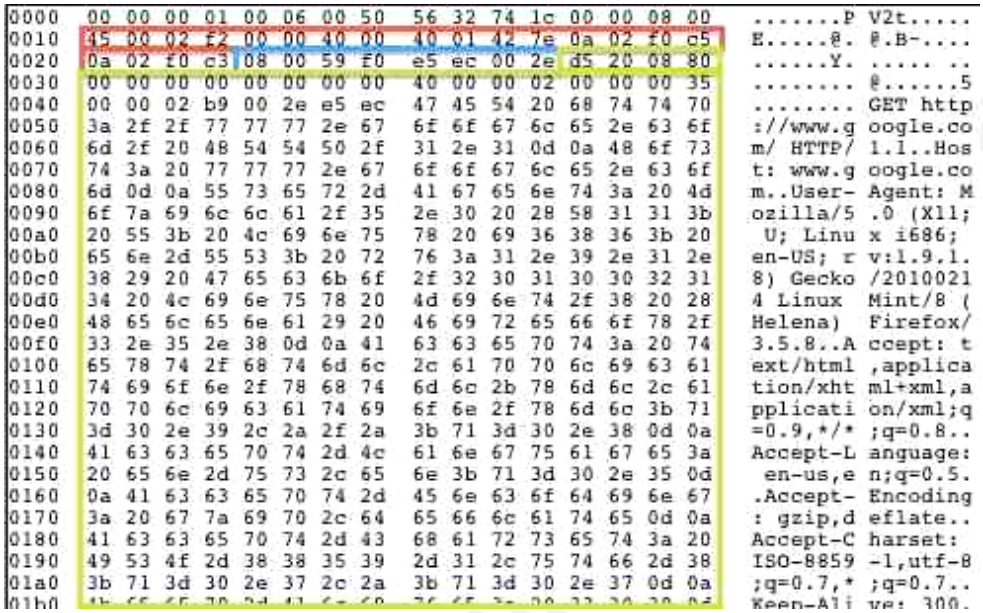



**Figure 11 – ptunnel packet analysis**

For this encapsulation to function correctly, ptunnel was designed with its own custom header (Fig 12). The packet format defines several new fields used to exchange messages between client and proxy (Stødle, 2005).

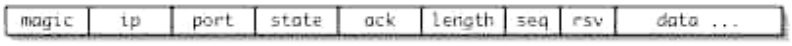



**Figure 12 – ptunnel header**

The fields employed are:

- Magic: A unique identifier that serves to separate the crafted Echo traffic from the normal Echo traffic.
- IP: Used only in traffic from client to server. Indicates destination IP.
- Port: Used only in traffic from client to server. Indicates destination port.
- State: Indicates what kind of message is being received, and who sent it.

- Ack: Similar to TCP Ack; used to acknowledge the sequence number of the next expected packet.
- Length: Length of embedded data field.
- Seq: Similar to TCP Sequence number; related to Ack; an incrementing 16-bit counter.
- Rsv: Reserved for future use.
- Data: Payload.

Figure 13 is a representation of the fields above, superimposed on a standard ICMP header.

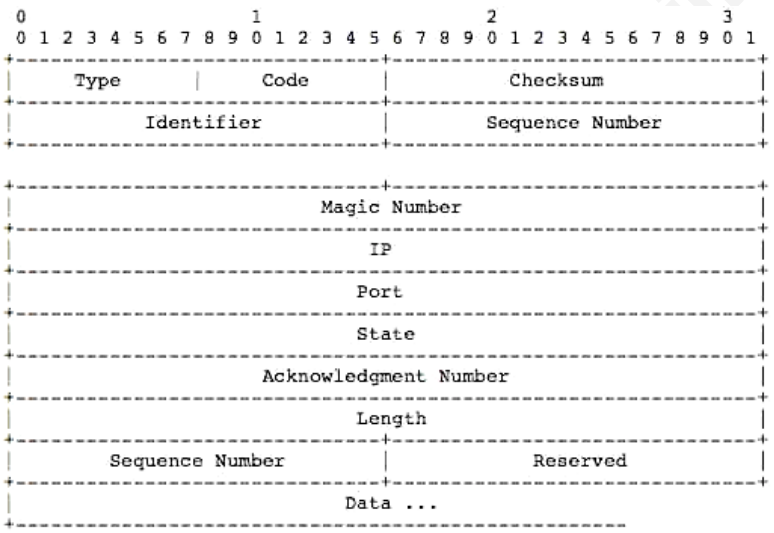



**Figure 13 – Full ICMP and ptunnel header (Doug, 2008)**

## 4.4. Tool Comparison

There are a number of ICMP tunnel exploits in the wild, which provide varying degrees of functionality and specialization.

- ptunnel : (http://www.cs.uit.no/~daniels/PingTunnel/PingTunnel-0.71.tar.gz) A well developed and documented ICMP tunneling software which supports multiple concurrent connections and password authentication. Source will compile on Unix variants and Windows.

- Loki : A venerable ICMP Backdoor program. Originally released in 1996 in the hacker magazine 'Phrack'.
- 007Shell : (http://packetstormsecurity.org/groups/s0ftpj/007shell.tgz) A tunneling package similar to Loki, which pads each packet in multiples of 64 bytes, making the tunnel appear more like legitimate traffic.
- ICMP Backdoor : (http://packetstormsecurity.org/UNIX/penetration/rootkits/icmp-backdoor.tar.gz) A rough-around-the-edges program which uses only ping reply packets. Because it doesn't pad up short messages or divide large messages, some IDS systems can easily detect traffic from this back- door tool. Compiles in several versions of Unix.
- B0CK : (http://www.s0ftpj.org/bfi/bfi7.tar.gz) A variant which uses IGMP multicast messages to improve on the work done by the authors of Loki and 007Shell. Also goes to the trouble of encoding the embedded address field for, arguably, additional covertness.
- Hans : (http://code.gerade.org/hans/) An IP (vice TCP) over ICMP solution. Employs TUN/TAP devices to, among other things, permit it to operate reliably if the firewall prevents multiple echo replies per request. Compiles for Unix/OSX and iPhone platforms (Smith, 2001).

# 5. DNS Covert Tunnels

## 5.1. DNS Review

The Domain Name System (DNS) is the protocol that translates human readable resource locators to machine readable and routable IP addresses. As such, it is commonly used to route users to resources on the Internet, most commonly to HTTP web pages. It is a bi-directional protocol that forwards queries from the originating host, out of the local network and on to a web of name servers that collaborate to resolve the appropriate IP address to a given URL.

The key components of the DNS header are primarily located in the 16$^{th}$ to 32$^{nd}$ bits of the DNS header; flags and codes (Fig 14). Among other things, the flags serve the

purpose of indicating if the DNS message being returned is the authoritative message, or if additional recursion is required to locate the answer to the query.

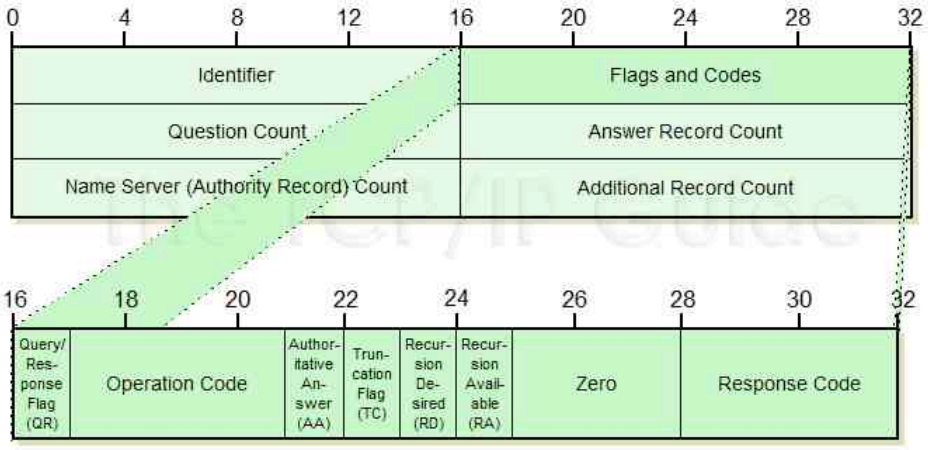



**Figure 14 – DNS header (Kozierok, 2005)**

The query will travel through the DNS system from secondary to primary and 'root' DNS servers in search of the authoritative answer to the query, which is eventually returned to the originating host as pictured in Fig 15.

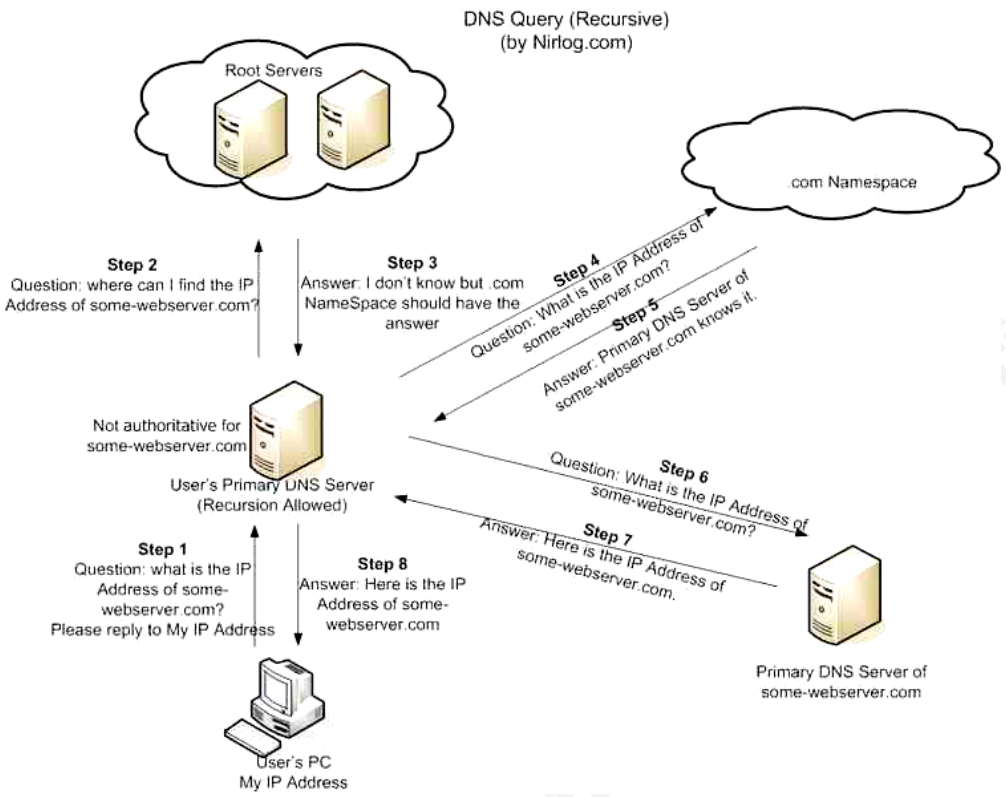



**Figure 15 – Recursive DNS query process (Nirlog.com, 2006)**

Several exploits exist which leverage DNS' bi-directionality, recursiveness and generally unsecured nature. They make it possible to exfiltrate data from within a secured network, access restricted resources or to provide a conduit for the control of malware resident on the inside.

## 5.2. The Root of the Exploit

As was the case for ICMP, DNS is pervasive and often allowed through network protective devices without scrutiny as it is assumed necessary for proper functionality of user application. Indeed, this is true, but a lack of vigilance presents a possible threat vector through which data can be covertly passed. However, due to its relative complexity, DNS tunneling only allows very low throughput (Tunnel hunter).
There are several types of DNS records; each designed to return a specific data type in response to a given request. 'A' records are authoritative records that return the IP address associated with the domain name. A 'CNAME' record returns an alias for a given host, while an NS record returns the IP of the authoritative name server for a queried

host. There are several other types, and while each serves a distinct purpose in the normal operation of DNS, the flaw exists in the fact that the maximum size of several of these field types are not strictly controlled. This design feature, which allows for long, complex URLs such as http://hostname.group.mail.domain.com, also allows the possibility for misuse in DNS requests. It is possible to send a DNS request formatted http://xxxxxxxxxxxxx.domain.com, where xxxxxxxxxxxxx is an instruction to a listening server. The embedded instruction can be engineered to carry plain text for a simple one-way data exfiltration, or a communications protocol, for tunneling TCP or other traffic. The client passes data 'up' the tunnel by encoding requests in Base32, the allowable character set for DNS queries. A request may resemble:

Dsf6tas6df5fa5df78a5d7f5adsf8a6d56a5d7.domain.com

The DNS server system will pass the query through to the authoritative DNS server for that domain (which is under our control) and it will arrive at the Ozyman server, where the embedded data is stripped off, un-encoded and forwarded to the actual requested destination. Returning traffic may be pushed back to the originating client through DNS TXT records, which allow greater flexibility in character use. The replies, encoded in Base64 can encode more data per character, and might resemble:

Jh7F9hd5dYVLhs8djFNRVU6Dgh44s7Fo

This method may be used to set up a fully functioning tunneling protocol within the DNS system. While the technology to create covert channels within DNS has existed since 1998, it was popularized by Dan Kaminsky's presentation at the 2004 DEFCON conference and his release of OzymanDNS tool.

## 5.3. OzymanDNS Traffic Analysis

We will now explore packet captures from the popular DNS tunneling tool, OzymanDNS. For the purposes of these analyses, a test lab was configured as per Fig 16.

It is important to note that you must have administrative access to the DNS server, as well as a separate server on which will run the Ozyman server software and will proxy our requests to the Internet.

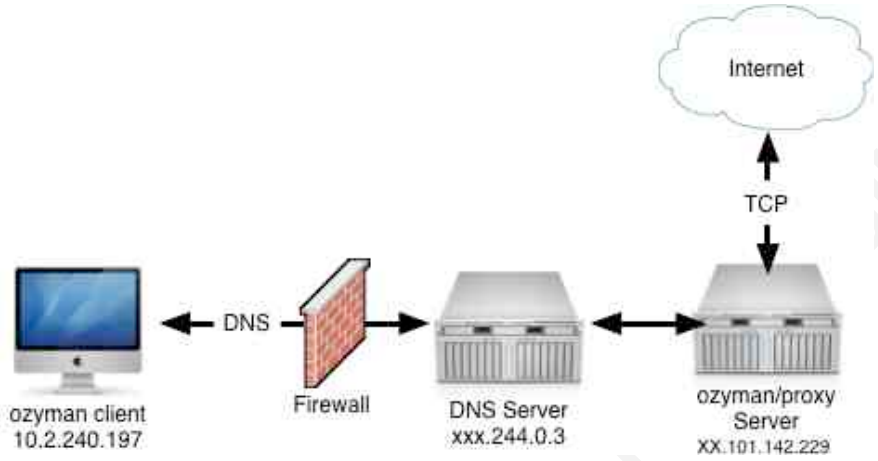



**Figure 16 – Ozyman lab setup**

The DNS server must be configured with a DNS 'NS' Record, to relay all its requests to the Ozyman server.

dnstunnel2.domain.ca IN NS XX.101.142.229

Once the DNS relay is in place, the server is activated by running the *nomde.pl* Perl script command from the server's shell prompt.

```
# ./nomde.pl -i 0.0.0.0 dnstunnel2.domain.ca
```

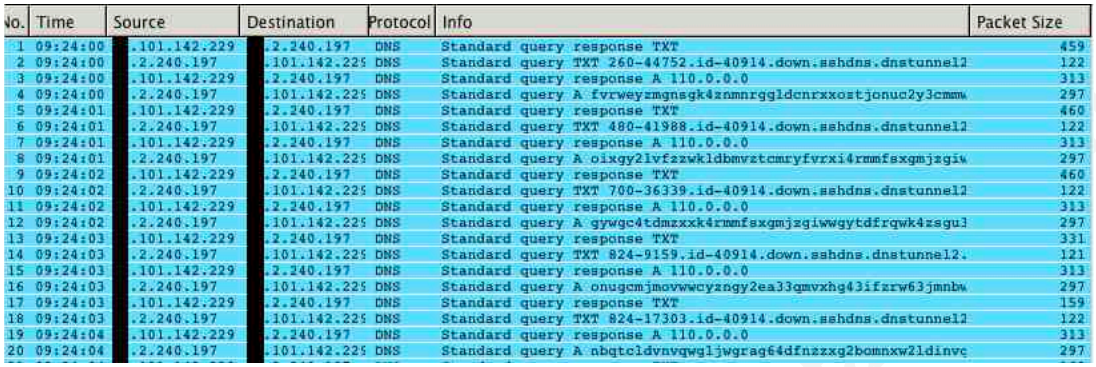
The tunnel is instantiated from the client with the following command:

```
# ssh -D 8080 -C -o ProxyCommand="droute.pl sshdns.dnstunnel2.domain.ca"
```

The switches define the following parameters:

- -D 8080 : sets up secure shell in dynamic forwarding mode.
- -C : Turn on SSH compression, for additional performance.
- -o ProxyCommand=droute… : sets the SSH option to call the Perl script and point it at the DNS server.

Figure 17 depicts a Wireshark dump of the Ozyman packet stream. We can observe the successive DNS queries, at about 122 bytes each and the returning query responses carrying payload in the 300-400 byte range.



| No. | Time | Source | Destination | Protocol | Info | Packet Size |
|---|---|---|---|---|---|---|
| 1 | 09:24:00 | .101.142.229 | .2.240.197 | DNS | Standard query response TXT | 459 |
| 2 | 09:24:00 | .2.240.197 | .101.142.225 | DNS | Standard query TXT 260-44752.id-40914.down.sshdns.dnstunnel2 | 122 |
| 3 | 09:24:00 | .101.142.229 | .2.240.197 | DNS | Standard query response A 110.0.0.0 | 313 |
| 4 | 09:24:00 | .2.240.197 | .101.142.225 | DNS | Standard query A fvrweyrmgnsgk4znmnrgqldcnrxxostjonuc2y3cmmv | 297 |
| 5 | 09:24:01 | .101.142.229 | .2.240.197 | DNS | Standard query response TXT | 460 |
| 6 | 09:24:01 | .2.240.197 | .101.142.225 | DNS | Standard query TXT 480-41988.id-40914.down.sshdns.dnstunnel2 | 122 |
| 7 | 09:24:01 | .101.142.229 | .2.240.197 | DNS | Standard query response A 110.0.0.0 | 313 |
| 8 | 09:24:01 | .2.240.197 | .101.142.225 | DNS | Standard query A oixgy2lvfzzwkldbmvztcmryfvrxi4rmmfsxgmjzgiw | 297 |
| 9 | 09:24:02 | .101.142.229 | .2.240.197 | DNS | Standard query response TXT | 460 |
| 10 | 09:24:02 | .2.240.197 | .101.142.225 | DNS | Standard query TXT 700-36339.id-40914.down.sshdns.dnstunnel2 | 122 |
| 11 | 09:24:02 | .101.142.229 | .2.240.197 | DNS | Standard query response A 110.0.0.0 | 313 |
| 12 | 09:24:02 | .2.240.197 | .101.142.225 | DNS | Standard query A gywgc4tdmzxxk4rmmfsxgmjzgiwwgytdfrqwk4zsgul | 297 |
| 13 | 09:24:03 | .101.142.229 | .2.240.197 | DNS | Standard query response TXT | 331 |
| 14 | 09:24:03 | .2.240.197 | .101.142.225 | DNS | Standard query TXT 824-9159.id-40914.down.sshdns.dnstunnel2. | 121 |
| 15 | 09:24:03 | .101.142.229 | .2.240.197 | DNS | Standard query response A 110.0.0.0 | 313 |
| 16 | 09:24:03 | .2.240.197 | .101.142.225 | DNS | Standard query A onugcmjmovwwcyzngy2ea33qmvxhg43ifzrw63jmnbw | 297 |
| 17 | 09:24:03 | .101.142.229 | .2.240.197 | DNS | Standard query response TXT | 159 |
| 18 | 09:24:03 | .2.240.197 | .101.142.225 | DNS | Standard query TXT 824-17303.id-40914.down.sshdns.dnstunnel2 | 122 |
| 19 | 09:24:04 | .101.142.229 | .2.240.197 | DNS | Standard query response A 110.0.0.0 | 313 |
| 20 | 09:24:04 | .2.240.197 | .101.142.225 | DNS | Standard query A nbgtcldvnvqwg1jwgrag64dfnzzxg2bomnxw2ldinvc | 297 |

**Figure 17 – Ozyman Wireshark capture**

An individual query coming into the Ozyman server resembles the following.

```
Writing response - done
Waiting for connections...
UDP connection from XX.206.142.26:19877
query 604: (1792-54698.id-40914.down.sshdns.dnstunnel2.domain.ca,          IN,        TXT) -
1792-54698.id-40914.down.sshdns.dnstunne2l.domain.ca    0          IN        TXT       "" ""
pending.1792-54698.id-40914.down.sshdns.dnstunnel2.domain.ca.  0          IN   A  0.0.0.0
```

The capture shown in Fig 18 shows the entire ~150 byte encoded payload of a covert DNS packet as it appears departing the client software.

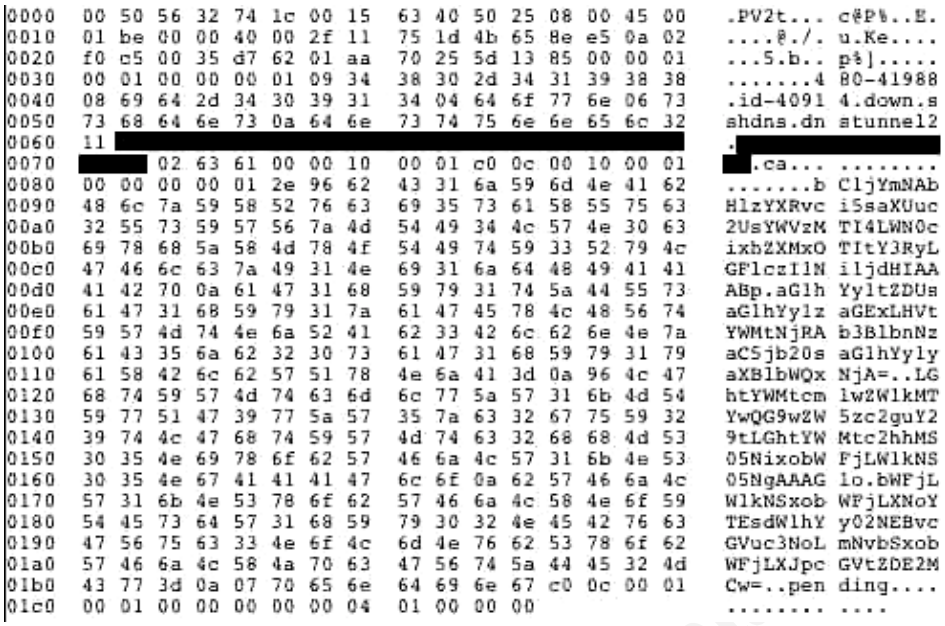



**Figure 18 – Ozyman packet capture**

The simple Base64 encoding scheme obfuscates the packet content and should prevent basic packet filtering IDS' from recognizing key traffic characteristics.

## 5.4. Tool Comparison

There are a number of DNS tunnel exploits in the wild that provide varying feature sets but similar core functionality.

- OzymanDNS: (http://web.archive.org/web/20080822170813/http://www.doxpara.com /ozymandns_src_0.1.tgz) Perl scripts written and popularized by DNS guru, Dan Kaminsky.
- NSTX: (http://savannah.nongnu.org/projects/nstx/) The 'Name Server Transfer protocol is one of the original DNS-IP tunnels. Not updated for nearly 7 years, but still very functional in its original form.
- dns2tcp: (http://www.hsc.fr/ressources/outils/dns2tcp) Quite similar to Ozyman; creates a DNS tunnel using a DNS 'IN' redirect, and allows for passage of arbitrary traffic.
- iodine: (http://code.kryo.se/iodine/) A current variant with Unix source and Windows binaries. Offers password protection and multiple concurrent user options.

- heyoka: (http://heyoka.sourceforge.net/) Windows-based DNS tunnel, focused on speed and reduced detectability.
- dnstunnel: (http://www.splitbrain.org/blog/2008-11/02-dns_tunneling_made_simple) A cleaned-up version of Kaminsky's Ozyman scripts. Fixed some bugs and formatting and removed code unnecessary for core functionality.

# 6. Risk Assessment

## 6.1. Introduction

Assessing risk in an information security context is a challenging endeavor. Several formulas and systems exist, touting solutions for identifying areas of concern and related mitigation, but in the end it is left to the security professional and the organizational stakeholders to define the inputs to these formulas. In the best cases, it is a challenging process to determine accurate assessments of possible threats and the likelihood of them occurring. Realistically, it is a game of educated guesses; what truly is the risk of a hacker or insider penetrating your network security with covert channels? It is a daunting question for which a comprehensive answer is only possible after a thorough analysis of your own organization's vulnerabilities and a number of other factors. This process is largely outside the scope of this paper, but a few relevant considerations will be reviewed below to provide a starting point.

Figure 19 (Maniscalchi, 2009), presents a model for determining threats. The salient points in this context are the 'weaknesses', 'person' and 'motivation' items. What are the weak points that might be exploited to install a covert tunnel? Who within your organization may likely want to use one to access outside resources or exfiltrate data, and why?


**Figure 19 – Threat model (Maniscalchi, 2009)**

Answering these questions will lead to a discussion of possible countermeasures, which might plug the relevant security holes and restrict the ability to do harm, from those who might most likely seek to do so.

## 6.2. Risk Assessment

With any review of potential vulnerabilities to a system, it is important to frame the assessment in terms of risk. ISO 13335 – Information Technology Security Techniques defines risk as: "The potential that a given threat will exploit vulnerabilities … and thereby cause harm to the organization." The overall risk to an information system will vary greatly depending on the organization, and ultimately be driven by the possible harm that may be caused to it by exploitation of its resources and data.

The well-known security principles of Confidentiality, Availability and Integrity may be applied as a test of the breadth of possible risks. In the case of covert channels, data confidentiality may be compromised though the prohibited passage of data in or out of a controlled network. Corporate files, client data or system information may be at risk, depending on the degree of the breach and the goal of the attacker. The availability of network resources may be affected through possible denial of service due to excessive use (flooding) of ICMP, DNS or other protocol to tunnel traffic. Network devices may not be able to cope with the unusually high throughput of these types of traffic, leading to unpredictable results. It is also possible that the tunneling of illicit network traffic could reduce the amount of bandwidth available to other users or processes and cause some reduction in normal service. System integrity may be jeopardized by the surreptitious

introduction of malware in the system, or other unauthorized system access through, for example, remote access to a system over a covert channel.

Risk is generally assessed as a function of the likelihood of a particular threat occurring. In this case, the threat has been defined in Section 1 as technology allowing the tunneling of data covertly across network security devices. In Fig 20 (Maniscalchi, 2009), risk is depicted graphically as the result of the product of the threat, the probability of it occurring and the impact to the organization.


**Figure 20 – Risk equation (Maniscalchi, 2009)**

## 6.3. Threat

The threats of unauthorized covert channels have been elaborated on in the past sections. To recap, the most likely among them include:

- Use of tunnels to circumvent network guards and gain unauthorized access to resources; either by legitimate internal users, or by hackers seeking to circumvent a user-pay or limited access system
- Exfiltration of sensitive data from within an organization, be it by an individual or by malware (Trojan horse)
- Infiltration of malware and/or related control signals into a secured network through a covert channel.

## 6.4. Probability

The likelihood of a security breach through the use of covert channels is largely based on several key factors. Primarily, the organization's security posture will determine the requirement for a motivated hacker or insider to resort to covert channels to accomplish his goals. The shortcomings of covert channels, as discussed earlier in this paper, include limited bandwidth and the requirement for a somewhat sophisticated user. While it is entirely possible that a user could employ covert channels regardless of the network security posture, it is far more likely that he would use any simpler means available through which to achieve his ends. This might include exfiltrating data on USB flash

drives or other portable media, or through legitimate but unmonitored network traffic such as email or peer-to-peer file transfer. An attacker will generally select the path of least resistance, unless perhaps, he is trying to mitigate possible future detection of a long-term 'data tap' he is emplacing.

## 6.5. Impact

The impact of data loss or misuse of network resources can be significant. The loss of corporate secrets and client personal information, as well as other types of closely held corporate data, can be devastating to an organization financially, legally and through loss of client trust. This analysis should be conducted before implementing any security measures and is equally relevant for any means though which the data may be lost, be it by covert channel, other malware or physical theft.

## 6.6. ROSI

While some might argue that security is not an investment in the normal sense of the word (it doesn't provide a financial return over time) it is an expense that will ideally pay for itself over time through the cost savings achieved by loss prevention in data "C, I and A". It is, however, 'in vogue' to attempt to associate security with the financial terminology easily understood by the executives who will inevitably be approving and paying for any costs.

Security resources, both financial and human, are limited and a security manager has to make daily decisions about best expending those resources to counter the areas of the highest risk. Depending on the presence, or lack, of firewalls, IPS', NIDs and HIDs, spending time and effort on addressing the threat of covert channels may or may not provide a wise Return on Security Investment (ROSI). Rough calculations can be performed to determine the ROSI of any given security investment using a modified ROI calculation:

$$ROSI = (Mitigated\ Risk - Cost) / Cost$$

This formula yields a curve on whichever increasing investments are met with ever diminishing security returns. This model favors implementing a few, cheap and effective

countermeasures to mitigate the majority of the risk until the point on the 'long tail' when the organization believes the residual risk is acceptable (Marcos, 2009).

# 7. Detection and Countermeasures

Detection and defeat of covert channels can, and should, be pursued with *defense in depth* techniques. Due to the nature of covert channels, discovering their existence can prove as challenging as defeating them. There is some research into IDS signatures, traffic analysis patterns and other protocol detection techniques, which only when combined may provide the requisite situational awareness and countermeasures to adequately find and defeat this type of threat.

## 7.1. Packet Signatures

Several techniques have been developed for covert channel detection using IDS signatures. The IDS signature in Table 3 displays a concept for detection of covert channels indirectly; rather then seek a specific characteristic of a covert channel itself, it focuses on known characteristics of the SSH protocol. SSH can be easily employed over many covert channels to mask traffic content, but has a characteristic signature, which can be searched for. When sought out on 'non-standard' SSH ports (if SSH is authorized at all on a network) it can give clues to secure tunnels that aren't where they should be.

| Sample Cisco IDS "SSH over Non-standard Ports" Alert<br>evIdsAlert: eventId=1183701598144406163 vendor=Cisco severity=informational originator:<br>hostId: ISE_NIPS01 appName: sensorApp appInstanceId: 384<br>time: August 30, 2007 2:39:51 AM UTC offset=60 timeZone=GMT-05:00<br>signature: description=SSH Over Non-standard Ports id=11233 version=S258<br>subsigId: 0 sigDetails: SSH Over Web Ports marsCategory: Info/Misc<br>interfaceGroup: vs0 vlan: 0 participants:<br>port: target: addr:<br>10.21.141.83 locality=Inside 4527<br>10.21.39.79 locality=Inside<br>port: 443<br>os: idSource=learned type=windows-nt-2k-xp relevance=relevant context:<br>fromTarget: 000000 53 53 48 2D 32 2E 30 2D 57 65 4F 6E 6C 79 44 6F SSH-2.0-WeOnlyDo<br>000010 20 31 2E 34 2E 30 0D 0A<br>1.4.0..<br>fromAttacker: 000000 53 53 48 2D SSH-<br>riskRatingValue: 26 targetValueRating=medium attackRelevanceRating=relevant<br>threatRatingValue: 26 interface: ge0_0 protocol: tcp |
|---|

**Table 3 - SSH over Non-standard Ports (Wippich, 2007)**

The ptunnel alert in Table 4 searches for the distinct characteristic found in *ptunnel*, Hex "\\xD5\\x20\\x08\\x80" (which can be spotted on p.15, Fig 11).

| Sample Cisco IDS "Ping Tunnel ICMP Tunneling" Alert<br>evIdsAlert: eventId=1183701598144394829 vendor=Cisco severity=high originator:<br>appName: sensorApp<br>appInstanceId: 384 time: August 27, 2007 11:34:48 PM UTC offset=60 timeZone=GMT-05:00 signature:<br>description=PingTunnel ICMP Tunneling id=5543 version=S176<br>subsigId: 0 sigDetails: \\xD5\\x20\\x08\\x80 marsCategory: Penetrate/Backdoor/CovertChannel<br>interfaceGroup: vs0 vlan: 0 participants:<br>attacker:<br>hostId: XXXX<br>addr: target: addr:<br>idSource=unknown type=unknown relevance=unknown<br>10.XXX.XXX.XXX locality=Inside<br>198.XXX.XXX.XXX locality=Inside<br>os: riskRatingValue: 80 targetValueRating=medium threatRatingValue: 80 interface: ge0_0 protocol: icmp |
|---|

**Table 4 - Ping Tunnel ICMP Tunneling (Wippich, 2007)**

Similarly, the following Snort signature was developed to detect *iodine* DNS tunnels, based, as in many Snort signatures, on specific byte content which can be found in these packet flows. It leaves to be proven how effective these types of filters are against older or newer versions of the exploits or how easily they can be avoided with any minor customization of the exploit source code.

| # detects iodine covert tunnels (over DNS<br>alert udp any any -> any 53 (content:"\|01 00 00 01 00 00 00 00 00 01\|"; offset: 2; depth: 10; content:"\|00 00 29 10 00 00 00 80 00 00 00\|"; \ msg: "covert iodine tunnel request"; threshold: type limit, track by_src, count 1, seconds 300; sid: 5619500; rev: 1;)<br>alert udp any 53 -> any any (content: "\|84 00 00 01 00 01 00 00 00 00\|"; offset: 2; depth: 10; content:"\|00 00 0a 00 01\|"; \ msg: "covert iodine tunnel response"; threshold: type limit, track by_src, count 1, seconds 300; sid: 5619501; rev: 1;) |
|---|

**Table 5 - iodine Covert Tunnels Detection (Chamberland, 2009)**

Table 6 and 7 give yet other examples of Snort rules that search for specific byte signatures.

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

**Table 6 – Potential NSTX Tunnel (Van Horenbeeck, 2010)**

| alert udp $HOME_NET any -> $EXTERNAL_NET 53 (msg:"PROXY nstx dns tunnel outbound to server"; content:"cT"; offset:12; depth:3; content:"\|00 10 00 01 00 00 29 08\|"; within:255; classtype:bad-unknown; sid:6546;) |
|---|

**Table 7 – Outbound NSTX connection (Jonkman, 2005)**

The effectiveness of these types of so-called "n-grep" or basic network traffic 'string searches' is hotly debated due to their ineffectiveness against unknown threats. Certainly, a motivated hacker could modify the code slightly and render these signatures ineffective. Still, there is little overhead in including them in your IDS' routine, particularly if you already suspect some sort of irregularity in your traffic due to some other warning sign. These filters can prove useful if a suspicious channel is found and the exact identity of the threat needs to be fingerprinted.

## 7.2. Anomaly Detection

If you suspect covert tunnels on your network, there are several methods, based on anomaly detection, which can be employed to identify the possible threat. Basic *tcpdump* or IDS rules can be employed to scan for any of the following:

- ICMP
    - An unusual spike in number of ICMP messages
    - More the one reply returning for a given Echo request
    - Unusually large ICMP messages
- DNS
    - An unusual spike in number of DNS requests
    - An unusual amount of traffic on port 53
    - Frequent, oversized DNS TXT records
    - Base 32 encoding in DNS request sub domains (Miller, 2008)

One of the difficulties of many of the above recommendations is setting the baseline; what are 'normal' parameters? This will naturally differ depending on your network and the type of traffic it carries. ICMP/DNS 'flood' filters might trigger false positives if there are specific legitimate network devices or applications which push peculiar, but valid, traffic on the network.

As shown in table 8, an ICMP flood alert can be triggered based on a certain number of ICMP packets received per period of time. This could be modified to trigger on similar, irregular 'flood' characteristics of a DNS, IGMP or other protocol-based tunnels.

| Sample Cisco IDS "ICMP Flood" Alert<br>evIdsAlert: eventId=1183701598144394830 vendor=Cisco severity=medium originator:<br>hostId: XXXXX appName: sensorApp appInstanceId: 384<br>time: August 27, 2007 11:35:27 PM UTC offset=60 timeZone=GMT-05:00 signature: description=ICMP Flood id=2152 version=S1<br>subsigId: 0<br>marsCategory: DoS/Network/ICMP interfaceGroup: vs0 vlan: 0 participants:<br>attacker: addr: 10.xxx.xxx.xxx locality=Inside<br>target: addr: 198.xxx.xxx.xxx locality=Inside os: idSource=unknown type=unknown relevance=relevant<br>riskRatingValue: 85 targetValueRating=medium attackRelevanceRating=relevant<br>threatRatingValue: 85 interface: ge0_0 protocol: icmp |
|---|

**Table 8 – ICMP Flood (Wippich, 2007)**

## 7.3. Traffic Analysis

Automated, algorithmic systems such as Web Tap and Tunnel Hunter have been developed which have made excellent inroads at detecting anomalous tunnel traffic exiting a firewalled network. These technologies attempt to avoid the shortcomings of packet filter/signature analysis and instead try to profile traffic based on human browsing patterns and known profiles of common types of application traffic. Clearly, the traffic profile of a person casually surfing the web differs greatly from that of someone chatting on IRC or downloading a Torrent; this new generation of analysis leverages these differences to detect anomalies in network traffic.

Web Tap measures browsing patterns and uses this data to determine if the traffic is coming from a legitimate user (Borders, 2004). It creates statistics based on several measurable criteria:

- Header formatting – Looks for anomalous (non-browser originating) headers.
- Delay times – Examines inter-request timings to identify automated programs operating on specific timers.
- Individual request sizes – Web Query sizes are generally quite small, this flag abnormally large requests.
- Outbound bandwidth usage – Similar characteristic as the Request Size, above.
- Request regularity – Normal traffic tends to come in short bursts, vice long, consistent periods of traffic.

- Request time of day – Searches for non-standard times of day for traffic, based on several days of observation and collection of a user data set.

Table 9 shows the cumulative number of alerts and false alarms encountered during testing, for each type of filter. The aggregate statistics show the results of all the filters running concurrently. The overall effectiveness of this technique amounts to approximately 78%. Combined with signature detection and some of the other recommendations above this technology could prove to nearly eliminate false alarms.

| Filter Name | | Number of Alerts Over 40 Days | Average Alerts Per Day | False Alarm Rate (Approximate) | False Alarm Requests |
|---|---|---|---|---|---|
| Header Format | | 240 | 6.00 | 0% (Exact) | Legitimate browser |
| Delay Time | | 118 | 2.95 | 5% | Not running on a timer |
| Individual Request Size | | 38 | 0.95 | 34% | Not upload or spyware |
| Request Regularity | 8-Hour | 132 | 3.30 | 11% | Not spyware/adware, ad server, or a timer. |
| | 48-hour | 65 | 1.63 | 7% | |
| Daily Bandwidth | 20 KB | 106 | 2.65 | 32% | Not spyware/adware, not an ad server, not a file upload, not running on a timer, and not a non-browser client |
| | 30 KB | 59 | 1.48 | 17% | |
| | 40 KB | 39 | 0.98 | 15% | |
| | 50 KB | 26 | 0.65 | 12% | |
| | 60 KB | 18 | 0.45 | 0% | |
| | 70 KB | 14 | 0.35 | 0% | |
| | 80 KB | 12 | 0.30 | 0% | |
| Time of Day | | 68 | 2.62 (Not Including | 28% | Anything when the user is present and active |
| Aggregate | | 767 | 19.2 | 12% (2.3 per day) | |

**Table 9 – WebTap statistical success rate (Borders, 2004)**

Tunnel Hunter (Dusi et al, 2008) uses a different statistical fingerprinting approach to detecting network tunnels. The researchers first analyzed a vast amount of captured operational network traffic and built mathematical fingerprints based on the behavior of an application protocol. These fingerprint are based on the IP packet size, time between the arrival of packets and the order of arrival. The fingerprint allows for successful identification of application traffic, even when channeled within an encrypted SSH tunnel.

Table 10 shows the Tunnel Hunter success rate at discovering particular types of traffic; all encrypted and tunneled within SSH tunnels.

| Protocols | Hit ratio | # sessions |
|---|---|---|
| SSH | 98.95% | 600 |
| SCP | 99.69% | 1700 |
| POP3 over SSH | 87.89% | 2360 |
| SMTP over SSH | 99.93% | 4300 |
| CHAT over SSH | 88.31% | 2100 |
| P2P over SSH | 88.77% | 1600 |

**Table 10 – Tunnel Hunter success ratio (Dusi et al, 2008)**

As the results suggest, Tunnel Hunter can detect legitimate SSH/SCP sessions with more than 98% accuracy. While these results are impressive, it leaves to be seen how this type of technology will transition from academia to turnkey security solutions.

## 7.4. Other Tactics to help Defend in Depth

- Employ the principle of Least Privilege; if users don't have the administrative rights to run unauthorized software, access a Command Prompt, send ICMP pings etc, they will be less likely to successfully deploy an covert channel.
- Developing, maintaining and enforcing a strong security policy, which properly defined misuse of network resources and outline consequences will deter less motivated individuals.
- Education. As with all threats, the more your IS security staff knows about covert channels, the more effective they can be at identifying them.
- Server hardening, as per established best practices, can help minimize exposure to all vulnerabilities, including this class of threats.
- Installing web and DNS proxies, which force all outbound DNS and HTTP requests to pass thorough them can help combat malformed requests by normalizing queries.
- Installing HIDS/NIDS which can raise flags of known Trojans, backdoors and malware which might, on additional examination have been secreted into the network through a covert channel.

- Create a solid baseline of network traffic, to better empower your analysis of various anomaly detection algorithms.
- Normalize traffic by overwriting exploitable fields and padding, which should be filled with zeros, with random noise (as long as it is non-critical to proper operation of the network), or strip unexpected or anomalous header extensions.
- Disable ICMP all together may be possible although there would be negative repercussions from network and helpdesk staff losing useful troubleshooting tool. Alternatively, host-based HIPS/firewalls could be configured to ignore inbound ICMP packets, perhaps from all IPs other then the network administrators.
- Logging. As covert channels are most likely to be discovered through second-order effects, a thorough set of logs can greatly assist in investigating the origin of a breach.
- Block, or allow only the types of ICMP and DNS messages absolutely required for proper operation.
- If covert channels cannot be entirely prevented, their effectiveness may be reduced through limiting the bandwidth, or placing restrictions of the types of traffic they exploit; a short delay added to each oversized DNS TEXT record would not substantially slow down a normal user, but could severely throttle a DNS covert channel.

There is promising research in the field of covert channel detection which recognizes the near futility of 'n-grep' type filters designed to attempt to locate common strings included in covert channels by well established exploit tools. The aim has turned instead to locating channels through use of tunnel detection algorithms and statistical fingerprinting techniques. The future promises to introduce IDS' that will identify channels based on a combination of protocol analysis, baselined usage trends, signs of tunneled protocols and traffic pattern recognition. At this time, however, security professionals are encouraged to consider covert channels in their risk analyses and to emplace what defense in depth would mitigate the perceived risk to their organization.

## 8. Conclusion

Nearly 10 years ago, Millen commented, "the one question that everyone asks about covert channels is whether they are a real threat" (Van Horenbeeck, 2006), and that remains a challenging question to answer. This paper set off with an aim of exploring covert computer network channels, examining the tactics they employ and considering options for protecting a network against the various threats they may pose. Indeed, covert channels are far from the most ubiquitous current threat to most computer networks. It is worth considering, however, that covert channels can be created, over nearly any protocol and if carefully designed may be nearly undetectable. This class of threat is unique in that sense. The presence of a covert channel may not be discovered in isolation, but its existence may be hinted at in analysis of a larger blended-threat security compromise. A degree of awareness of covert channel tactics and techniques will allow incident responders, forensic specialists and intrusion detection system engineers to consider them when analyzing the causes for a data breach or discovering the means through which a piece of malware circumvented network defenses.

As network defenses and user awareness improve in an organization, a motivated individual may resort to ever more obscure and complex means to achieve his ends. While the particular vulnerabilities presented in this paper are not exceptionally new or cutting edge, it is the understanding of the methodology of covert channels that remains the key take-away.

## 9. References

Ahsan, K., Kundur, D. (2002). Practical Data Hiding in TCP/IP. *Proceedings of ACM Workshop on Multimedia Security*

Baxter, M. (2008). IP Header Diagrams, Retrieved from: files.maunier.org/headers/

Borders, K. (2004). Web Tap - Detecting Covert Web Traffic, Retrieved from: http://www.eecs.umich.edu/ aprakash/papers/borders-prakash-ccs04.pdf .

Buetler, I. (2009). Covert Channel Attacks – Inside-out Attacks, Compass Security Retrieved from: www.csnc.ch/misc/files/publications/ covert_channel_csnc_v1.0.pdf

Chamberland, M. (2009). Snort rules for Iodine Covert DNS Tunnel Detection. Retrieved from: http://blog.securitywire.com/2009/07/26/snort-rules-for-iodine-covert-dns-tunnel-detection/

Doug. (2008). Using ICMP tunneling to steal Internet, Retrieved from: www.neverfear.org/blog/view/9/Using_ICMP_tunneling_to_steal_Internet

Dusi, M., Crotti, M., F. Gringoli, Salgarelli, L. (2008). Detection of Encrypted Tunnels across Network Boundaries, *Proceedings of the 43rd IEEE International Conference on Communications*.

Dusi, M., Crotti, M., Gingili, F., Salgarelli, L. (2009). Tunnel Hunter: Detecting Application-Layer Tunnels with Statistical Fingerprinting. *Elsevier "Computer Networks" (COMNET) Vol 53*.

Forte, D. V., Marti, C., Vetturi, M. R., Zambelli, M. SecSyslog. (2005). An Approach to Secure Logging Based on Covert Channels. *Proceedings of First International Workshop on Systematic Approaches to Digital Forensic Engineering*.

Gregg, M. (2007). Hack the Stack: Using Snort and Ethereal to Master The 8 Layers of An Insecure Network, Syngress Publishing.

Jonkman, M. (2005). NSTX DNS Tunnel Sig. Retrieved from: http://blog.gmane.org/gmane.comp.security.ids.snort.bleedingsnort/month=200511 01/page=2

Kozierok, C. M. (2005). The TCP/IP Guide. Retrieved from: www.tcpipguide.com/free/t_DNSMessage HeaderandQuestionSectionFormat.htm

Kunwar, N. (2006). DNS Amplification Attack. Retrieved from: www.nirlog.com/2006/03/28/ dns-amplification-attack.

Maniscalchi, J. (2009). Threat vs Vulnerability vs Risk. Retrieved from: www.digitalthreat.net/2009 /06/threat-vs-vulnerability-vs-risk.

Marcos, G. (2009). ROSI - Return on Security Investment Retrieved from: www.blogs.globalcrossing.com/ ?q=content/rosi-return-security-investment.

Sheets, M., Koot, M. (2001). Research Report: Covert Channels. Master's thesis, University of Amsterdam.

Smith, J.C. (2001). Covert Channels. Retrieved from: www.sans.org/infosecFAQ/covertchannels /covert_shells.htm .

Stødle, D. (2005). ptunnel - Ping Tunnel. Retrieved from: www.cs.uit.no/daniels/PingTunnel

Van Eden, L. (2001) The Truth About ICMP. Retrieved from: www.sans.org/infosecFAQ/threats/ICMP.htm

Van Horenbeeck, M. (2006). Deception on the Network: Thinking Differently About Covert Channels. *Proceedings of 7th Australian Information Warfare and Security Conference*.

Van Horenbeeck, M. (2010). DNS Tunneling. Retrieved from: http://www.daemon.be/maarten/ dnstunnel.html

Wippich, B. (2007). Detecting and Preventing Unauthorized Outbound Traffic. SANS GCIH Gold Certification Paper

Zander, S., Armitage, G., Branch, P. (2007). A Survey of Covert Channels and Countermeasures in Computer Network Protocols, *IEEE Communications Surveys and Tutorials, Vol. 9, No. 3*.

Zander, S., Armitage, G., Branch, P. (2007). Covert Channels and Countermeasures in Computer Network Protocols, *IEEE Communications Magazine, Vol. 45, No. 12*.