Implementation of an ICMP-based covert channel for file and message transfer

Conference Paper · October 2008

DOI: 10.1109/ICECS.2008.4674998 · Source: IEEE Xplore

3 authors, including:

Zouheir Trabelsi
United Arab Emirates University
139 PUBLICATIONS **1,337** CITATIONS
SEE PROFILE

Wassim El-Hajj
American University of Beirut
123 PUBLICATIONS **3,256** CITATIONS
SEE PROFILE

Implementation of an ICMP-Based Covert Channel for File and Message Transfer

Zouheir Trabelsi, Wassim El-Hajj, and Safuat Hamdy
College of Information Technology
UAE University
Al Ain, United Arab Emirates
trabelsi@uaeu.ac.ae, welhajj@uaeu.ac.ae, s_hamdy@uaeu.ac.ae

Abstract— **Covert channels exist in most communications systems and allow individuals to communicate truly undetectable. However, covert channels are rarely used due to their complexity. This article presents a novel covert channel based on record route IP header options for downloading and uploading text files and sending short messages. The presence of hidden texts and messages exchanged over the proposed covert channel will be unnoticed unless the IP header options are inspected. A tool has been implemented based on the proposed covert channel. It uses a client/server technology to offer a hidden version of the well known File Transfer Protocol (FTP) service. This service is called Hidden File Transfer Protocol (HFTP). In addition, the tool offers a hidden SMS service (HSMS) allowing to send hidden short messages.**

Keywords: **Covert channel, record route IP header option, Hidden File Transfer Protocol (HFTP), Hidden SMS (HSMS).**

I. Introduction

In order to protect information that is sent over a network from unwanted disclosure, one usually uses cryptography. However, an encrypted channel between two remote entities may allow an attacker to detect that there is information exchanged between the two entities. Therefore, the attacker may proceed further and attempt to decrypt the captured traffic.

Instead of hiding the information of a message one could alternatively hide the very existence of the traffic; in other words, instead of cryptography one resorts to steganography. The sheer volume of Internet traffic provides a high bandwidth vehicle for covert communications that leads to a plethora of applications.

In this article, we propose a covert channel; our implementation offers two services: concealed file download and upload and concealed short messages. The implementation demonstrates that our proposal is practical and efficient.

II. Related Work

The concept of a covert channel was first introduced by Lampson [1] as a channel that is used for information transmission, but that is not designed nor intended for communication. Girling [2] has analyzed covert channels in a network environment; his work focuses on local area networks (LANs) in which three *obvious* covert channels (two storage channels and one timing channel) are identified. The first uses the bits reserved for addresses, the second uses the bits reserved for the length and the third uses the time difference between the packets.

Wolf [3] presents results applied to LAN protocols. He highlights the relationship between covert storage channels and protocol format, and the link between covert timing channels and protocol procedure elements taking into account the frame layouts of the LAN protocols. Covert storage channels utilize the *reserved fields, pad fields* and *undefined fields* of the frames.

A more specific approach is adopted by Rowland [4]. Focusing on the IP and TCP headers of the TCP/IP protocol suite, Rowland devises proper encoding and decoding techniques by utilizing the *IP identification field*, the TCP *initial sequence number* and *acknowledge sequence number fields.*

Katzenbeisser and Petitcolas [5] have also observed the potential for data hiding in the TCP/IP protocol suite. The significance of using TCP/IP stems from the sheer volume of covert communication that can be attained since TCP/IP packets are used to transport vast amounts of Internet packets. Katzenbeisser and Petitcolas use the term *Internet Steganography* for this potential scenario and indicate that the ongoing research work includes the embedding, recovering and detecting information in TCP/IP packet headers.

Compared to the current covert channels, our proposed covert channels is more practical and robust since it combines the simplicity of ICMP tunnelling and the undetectability of the TCP/IP covert channel. In fact, the ordinary ICMP covert channel involves storing hidden data and messages in the data fields of the ICMP packets, mainly in Ping ICMP packets. The existence of hidden data and messages in such data fields can be easily identified, since ICMP packets usually do not carry data inside their data fields. However, our proposed covert channel does not use the data fields of the ICMP packets. It uses the fields of the IP header option.

III. THE FIELDS OF THE IP HEADER OPTION

In the IP header of an IP packet, the IP options field is not required in every datagram. Options are included primarily for network testing or debugging. Options processing is an integral part of the IP protocol, and any conforming implementation must include it.

The IP header option field has the following structure for all the types of options:

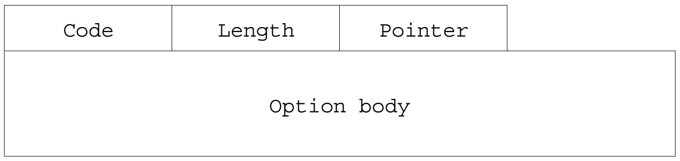

Figure 1. The fields of the IP header option

There are eight possible options that can accompany an IP datagram; most options are used for control purposes.

The routing and timestamp options are most interesting because they provide a way to monitor or control how internet gateways route datagrams. The record route option allows the source to create an empty list of IP addresses and arrange for each gateway that handles the datagram to add its IP address to the list.

The `Code` field contains the option number and option class (7 for record route). The `Length` field specifies the total length of the option as it appears in the IP datagram, including the first three octets. The fields starting with one labeled `First IP Address` comprise the area reserved for recording internet addresses. The `Pointer` field specifies the offset within the option of the next available slot.

IV. COVERT CHANNELS BASED ON THE RECORD ROUTE IP OPTION

When the IP header option designates a record route, the fields `Code` and `Pointer` should be set to the values 7 and 4, respectively. The maximum value in the field `Length` should be 39. On its way to the destination any packet with such IP header option would ask each router it passes to write its IP address into the 4-bytes field pointed to by `Pointer` (Figure 2.) and to increment `Pointer` by 4. The next router would write its IP address into the next 4-bytes field of the IP header option. However, if the value of `Pointer` becomes greater than the value of `Length`, then no more routers can store their IP addresses in the IP header option.

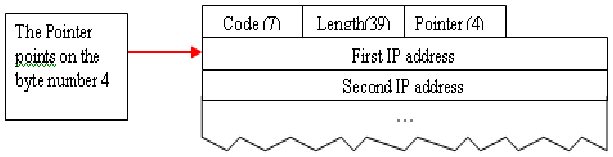

Figure 2. A normal record route option header Therefore, we may establish a covert channel if the initial value of `Pointer` is greater than the value of `Length` (Figure 3. a), or just greater than the length of the hidden message (Figure 3. b).

If we set the initial value of `Pointer` greater than the value of `Length`, then no router can write its IP address into the header option, and we can use up to 36 bytes of the IP header option to store hidden data. However, if we set the initial value of `Pointer` just greater than the length of the hidden message, then some routers can write their IP addresses into the remaining bytes of the IP header option.

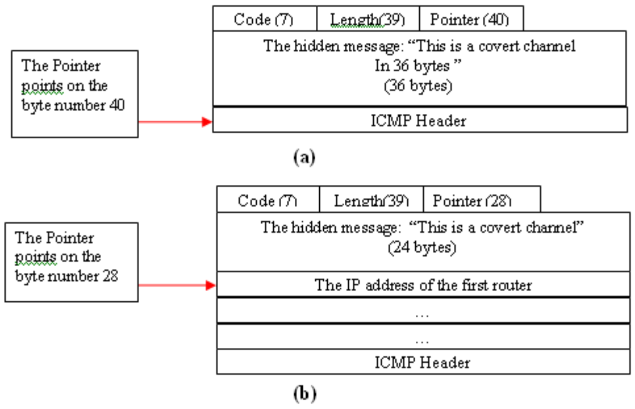
The Pointer points on the byte number 40
Code (7)
Length(39)
Pointer (40)
The hidden message: "This is a covert channel In 36 bytes " (36 bytes)
ICMP Header
(a)
The Pointer points on the byte number 28
Code (7)
Length(39)
Pointer (28)
The hidden message: "This is a covert channel" (24 bytes)
The IP address of the first router
...
...
ICMP Header
(b)

Figure 3. The different values of the Pointer field used for the covert channel

V. A HIDDEN FILE TRANSFER PROTOCOL – HFTP

As an application of the proposed covert channel, we defined a hidden file transfer protocol (HFTP) that allows downloading and uploading hidden text files and sending hidden SMS messages. Therefore, a client/server application can be built based on this HFTP protocol.

The previous section describes the structure of a normal record route IP header option. For the purpose of the HFTP protocol, we modify that structure by adding new fields that will be used mainly:

- To open a connection by the HFTP client with the HFTP server
- To acknowledge a packet
- To terminate a session.
- To control the packet's sequence numbers

The new structure of the record route IP header option used by the HFTP Protocol is shown in Figure 4.

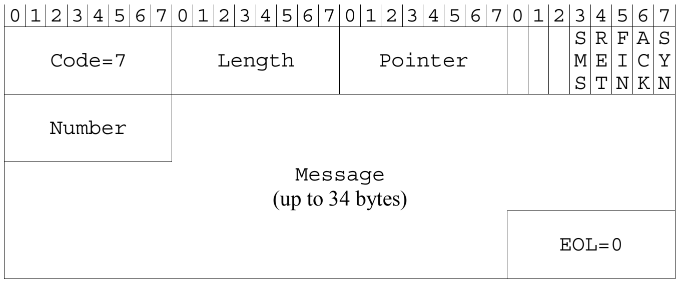

Figure 4. The new record route IP header option structure used by the HFTP protocol

It is important to indicate that all the fields in this new structure, except the field `Hidden Message`, constitute the header of the HFTP protocol. These fields will be used to control the session between the HFTP client and HFTP server; the field `Hidden Message` is the space available for hidden data.

A. The fields of the HFTP protocol header

The function of each field in the new structure is as follows:

1. The fields `Code`, `Length` and `Pointer` have the same use as in a normal record route IP header option.
2. The flags `SYN`, `ACK`, `FIN`, and `SMS`:
   - `SYN`, `ACK` and `FIN` are similar to the ones used by the TCP protocol.
   - a set `RET` flag indicates that the packet is a retransmission of a lost packet (for lost packet control).
   - a set `SMS` flag indicates that the packet carries an indiviual hidden SMS message and is not a part of a hidden file transfer.
3. `EOL` (end of list): must be zero, and it is a padding field.
4. The field `Number` (one byte) has four uses which are discussed in the next section.

B. The Field `Number`

The field `Number` has four uses depending on the type of the packets exchanged between the server and the client.

1. The `Number` field is used by the HFTP server to inform the HFTP client about the total packets that will be sent.
2. The field `Number` is used by the HTFP client to inform the HFTP server about the total lost packets.
3. The `Number` field is used by the HFTP server or the HFTP client to indicate the packets sequence numbers, during the downloading phase or uploading phase, respectively.
4. The `Number` field is used by the HFTP server to inform the HFTP client about the type of allowed operations or provided privilege. There are four types of operations or privilege, namely: view list (list of files to be downloaded), download files, upload files, and send SMS.

Figure 5. shows the structure of the `Number` field when it is used to indicate the type of operations or privileges.

0	1	2	3	4	5	6	7
unused				send SMS	upload file	download file	view list

Figure 5. The structure of the `Number` field when it is used to indicate the type of privilege

When the HFTP server offers a particular privilege to an HFTP client, the corresponding bit in the `Number` field is set to 1.

Figure 6. specifies an example of the privileges offered by the HFTP server to a particular HFTP client. The HFTP client is not allowed to view the list of the hidden texts files that can be downloaded. However, the client is allowed to covertly upload files and covertly send SMS messages. Therefore, the decimal value of the `Number` field is 12.

0	1	2	3	4	5	6	7
unused				send SMS	upload file	download file	view list
0000				1	1	0	0

Figure 6. An example of privileges offered by the HFTP server to an HFTP client

VI. A SCENARIO FOR ESTABLISHING A CONNECTION WITH THE HFTP SERVER

The usual FTP service is designed to be encapsulated in TCP packets. Operations like establishing a connection, retransmission of packets, flow control etc. are treated by the TCP protocol during a FTP session. However, the proposed covert channel does not use TCP; the encapulating protocol is ICMP. Therefore, the HFTP protocol should provide means for establishing and terminating connections, and controlling the traffic. For that reason we inserted some flags similar to ones used by the TCP protocol in the HFTP header.

All packets exchanged between the HFTP server and the clients are ICMP packets. To establish a connection with the HFTP server, a HFTP client sends first to the server an ICMP packet with an ICMP header and an IP header with a record route option where the fields `Code`, `Length`, and `Pointer` are set to the following values: `Code` = 7, `Length` = 39, and `Pointer` = 40.

In the sequel we describe how a client establishes a connection to a HFTP server:

1. In the first packet used to establish the connection with the HFTP server, the bit `SYN` of the `Flags` field in the record route option is set to 1.
2. The HFTP server will send a packet to the client asking also for establishing a connection (the bit `SYN` = 1) and acknowledging the received packet (the bit `ACK` = 1). The reply `220` is used to indicate that the HFTP server is asking for a username and password of the client. We actually use the same commands and replies used by the normal FTP protocol.
3. The HFTP client will send its username and password to the HFTP server.
4. The HFTP server will send to the client its corresponding privilege. If the privilege value is zero, then an authentication error has occurred and the `Message` field is empty. If the privilege indicates that the client can not

see the list, then the `Message` field is empty but no error message will be shown to the client. In all the remaining cases, the `Message` fields contains the reply `331` as defined by the normal FTP protocol.

5. This packet is the acknowledgement packet of the Packet 4.

A. *File Download: The HFTP server command `get`*

Once the connection is established with the HFTP server, the client will attempt to download a text file. The following packets exchanged are:

1. client: a `SYN` packet that carries the name of the file selected to be downloaded.

The server then sends the following packets:

2. An initial `SYN-ACK` packet that carries the total number of packets that will be sent (this number is obtained by dividing the file size by 34).
3. the following packets: packets that carry the actual contents of the file.
4. an optional packet: total number of lost packets and their corresponding sequence numbers.
5. optional: retransmission of any lost packets.

The client then sends

6. The final packet: `FIN-ACK` packet sent by the client to indicate that the file is downloaded successfully.

FTP offers a variety of services that use large number of specific commands. Our proposed HFTP does not need this large number of commands since it is used for simple download and upload of files; operation like directory creation and change, binary transfer, removing files and so on are not necessary. For these reasons HFTP includes only the two commands `get` and `put`.

B. *Retransmission of Lost Packets*

TCP uses a window for controlling the packet flow. HFTP does not provide this feature; doing so would further decrease the already limited bandwidth. We use the following simple process for the retransmission of lost packets:

The client starts a download by sending the command `get` *`file`*. The server calculates the total number of packets that needs to be sent. After receiving the total number of packets that will be sent, the client waits for the first packet. When the first packet is received, the client waits for the second one. If the client does not receive a packet within a timeout, the client considers that packet as a lost packet. Hence, the client starts waiting for the packet that normally follows the lost packet.

When a client receives the `FIN` packet or the total number of received packets reaches the expected number, the client sends an `ACK` packet carrying the number of lost packets. This process is repeated until there are no lost packets.

C. *File Upload: The HFTP server command `put`*

The upload process is similar to the download process, but the file is located at the client side. Hence, the packet containing the total number of the packets that will be sent to the HFTP server is generated by the HFTP client.

VII. Security Considerations and Limitations

The bandwidth of the proposed covert channel is rather low: up to 34 data bytes may be sent in one packet. In order to send a large message or file, one would need to send quite a number of ICMP packets. However, an unusual number of such packets might arouse suspicion.

In order to mitigate such suspicion the covert channel uses many types of ICMP packets, such as echo ICMP packets and timestamp ICMP packets.

We point at some issued that need to be addressed: First, a firewall on the path could be configured to eliminate traffic of that kind at all; in this case our covert channel cannot be used. Second, an intrusion detection system might flag such traffic as attack and raise an alarm; in this case the channel no longer covert. Third, an attacker who is prepared to watch specifically for our covert channel will detect its presence; in that case further obfuscation is necessary such as non-standard encryption. Then the data will appear to be (pseudo)random data. This might arouse suspicion as well: the entries in the source routing option header are not random; with little extra effort it can be verified whether they are meaningful or not.

Finally, we also point out that our proposal does not provide any kind of integrity checking (mostly in order to save bandwidth); this is currently left to the application of the channel.

VIII. Conclusion

We have introduced a novel protocol for covert channels, the Hidden File Transfer Protocol (HFTP). This protocol uses the record route IP header options for conceiled download and upload of files and conceiled transport of short messages (SMS). Typical sniffers do not inspect the record route IP options; such a sniffer will not notice the presence of the covert channel. In addition, to avoid the detection of the packets flow between the HFTP server and client, the packets exchanged do not have the same ICMP packet's type. A GUI client/server tool has been developed to demonstrate the practical efficiency of the proposed covert channel.

In the proposed method, the information in the covert channel is packaged in the form of IP addresses. However, it is possible for one to verify the validity of these IP addresses in the connection path which immediately offers a means for steganalysis. New technologies are being developed to protect the scheme from such potential analysis.

References:

[1] B.W. Lampson. A Note on the Confinment Problem. In *Communications of the ACM* 16(10), pp. 613–615, October 1973.

[2] C.G. Girling. Covert channels in LAN's. In *IEEE Transactions on Software Engineering* 13(2), pp. 292–296, February 1987.

[3] M. Wolf. Covert channels in LAN protocols. In *Proceedings of the Workshop on Local Area NetworK Security – LANSEC'89* (1989), T.A. Berson and T. Beth (eds.), vol. 396 of LNCS, Springer, pp. 91–101.

[4] C. H. Rowland. Covert channels in the TCP/IP Protocol Suite. In *First Monday*, 2(5), May 1997. Available from http://www.firstmonday.org/ISSUES/issue2_5 .

[5] S. Katzenbeisser and F. Petitcolas. Information Hiding Techniques for Steganography and Digital Watermarking. Artech House Books, 1999.