Covert Channel using the IP Timestamp Option of an IPv4 Packet

Hassan A. Alsaffar, Graduate Student, Hassan.Alsaffar@rit.edu
and Daryl Johnson, Associate Professor, Daryl.Johnson@rit.edu
Computing Security Department
B. Thomas Golisano College of Computing and Information Science
Rochester Institute of Technology
Rochester, New York 14623

ABSTRACT

The TCP/IP protocol suite is known to be lacking security in many aspects, which may allow an attacker to leverage them using some techniques in the form of covert communications. Secret messages could be embedded into one of the TCP/IP header fields and possibly not get detected by various firewalls or intrusion detection systems (IDS's). This paper proposes a novel covert channel utilizing the IP timestamp option inside an IPv4 packet header to exchange a secret message with another party.

KEYWORDS

Covert Communication, Storage-based Covert Channel, System Security, Network Security, Steganography, Encryption, TCP/IP, IPv4, Network Monitoring, Intrusion Detection System.

1 INTRODUCTION

Over the past few decades, security threats, like computer viruses, trojan horses, and other exploits have been of increasing concern to system and network security professionals. However, few of these professionals investigate the risk of covert channels while most have overlooked and deemed them as a low priority risk compared to other threats until recently. Covert channels do not attract the attention of security analysts as much as viruses and other exploits. Taking advantage of covert channels, illicit users have been utilizing available technologies to transmit their own secret messages to other parties without the system/ network administrators' knowledge. According to the SANS Institute, a private U.S. company specialized in information security and cybersecurity, "covert channels techniques have been evolving from the 1970s until today, trying to avoid protections used by network administrators" [1]. Covert channels are a rising threat due to the increasing use of technology that might lack security and could be leveraged for malicious activities.

In this paper, an example of a network-based covert channel is demonstrated to show that the threat of covert channels is real and practical. This proposed covert channel utilizes the IP timestamp option of an Internet Protocol version 4 (IPv4) packet to exchange a secret message. This covert channel will use the Hypertext Transfer Protocol (HTTP) which usually has a lot of traffic associated with it. This traffic could thus provide an excellent means for concealing the covert channel. However, this covert channel could be used with any protocol that is part of the internet suite of communication protocols including but not limited to HTTPS, SSH, FTP.

1.1 Covert Channels

Covert channels were defined for the first time by Butler Lampson in 1973 as a communication channel not designed for any kind of information transfer [2]. Lampson's definition of a covert channel applies to system-based channels, but the concept is still relevant in any shared environment, including network-based environments. Network-based covert channels were first introduced in 1987 to prove that such a concept is possible and can be applied over the network as well [3]. Such a type of covert channel takes advantage of unused bits and fields from the Transmission Control Protocol/ Internet Protocol (TCP/IP) headers or padding data using certain techniques to transmit information among two or more parties. Those covert channels may be limited, but they do provide realistic channels through legitimate and regular TCP/IP traffic moves.

1.2 IP Timestamp Option

IP Timestamp is an optional extension to the IPv4 header that allows the sender to request timestamp values from any machine which handles the packet by specifying its IP address. According to RFC781, the IP timestamp option is a right-justified, 32-bit timestamp in milliseconds since midnight UT [4]. It is primarily used for diagnostics purposes specifically to measure the network delay time between gateways.

2 COVERT CHANNEL CHARACTERISTICS

The objective of the covert channel of this study is to exchange secret messages in the presence of firewalls and IDSs. The secret communication thus will be hidden and encrypted, so that an attacker cannot demonstrate its existence without knowing the technique used to hide the message as well as the secret key used for the encryption. Along with those objectives, below are the major characteristics that this covert channel has:

- **Type**: traditionally, covert channels were classified into storage and timing channels [3]. The type of this covert channel is storage-based, since this channel involves the writing of object values by sender into the IP timestamp option field of a packet.
- **Bandwidth**: the bandwidth is measured in bits per second of how much information can be transmitted through a covert channel. Due to the fact that the length of the timestamp option is 32 bit, one can send a secret message with up to 12 characters per packet. Based on the experiment implemented using this covert channel, it took 72.3 milliseconds for the HTTP Get Request packet to travel from the client to the simulated web server. Such a low bandwidth will reduce the observability of the channel by statistical means.
- **Detection**: due to the fact that "IP packets very rarely contain the timestamp option" [5], the potential for this covert channel to be detected is high. Therefore, an encryption mechanism is utilized to ensure the exchanged messages are not readable by anyone unless he/she has the decryption key. Further details about the encryption method used are explained later.
- **Prevention**: firewalls and IDSs are capable of detecting the activity of this covert channel due to the rare use of the IP timestamp option these days. However, most of those firewalls and IDS's rules and signatures are based on public list of malicious URLs or IPs. Thus, unless custom rules are created to detect the use of the IP timestamp option, this covert channel would maintain a good undetectability level and covert rate.
- **Permissibility**: "Packets with the option (IP timestamp option) present can travel at most 20 hops, so it is of little use in the open Internet" [6]. That being said, this covert channel is more suitable for small-to-medium size networks.

3 METHODOLOGY

Under a Linux environment, two Python (Version No. 2.7.8) scripts were written for the purpose of demonstrating this covert channel. One script is on the client, which is used for establishing the connection, encrypting, and sending the covert message to the simulated web server. The second script is on the server, which is used for completing the connection, receiving, and decrypting the covert message. Those scripts utilize Python libraries such as

Scapy (Version No. 2.2), a well-known network packet manipulation tool [7], to generate ordinary HTTP transaction packets that will be utilized to carry the covert message between the client and the server. Knowing that HTTP data rides above the TCP protocol, which guarantees reliability of delivery and breaks down large data requests and responses into network-manageable chunks, it is important to ensure that the covert channel scripts establish the TCP three-way handshake connection before sending the covert message over the HTTP transaction packets (Get Request and Response packets).

3.1 Embedding within the IP Timestamp Option

As described earlier, the length of the IP timestamp option is 32 bits (4 Bytes). The first byte would be used to specify the indicator of the timestamp option in the packet using Scapy, and the other 3 Bytes are to be used to carry the secret message (3 Bytes of data can carry up-to 12 characters).

3.2 Encryption

Due to the length of the secret message (24 bits, 3 Bytes), Caesar cipher is used to encrypt/decrypt the covert messages. Caesar cipher is a type of substitution cipher that shifts the characters *N* number of places. To make this encryption technique stronger, the shifter, the encryption/decryption key, will be incremented by 1 every time it is used. 25 is the maximum number the shifter can be equal to, so techniques to ensure that the shifter does not go bigger than 25 is implemented on both the encryption and decryption keys.

3.3 Topology

Two separate hosts on two different networks were utilized to analyze this experiment and capture the network traffic. One machine served as a web server while the other served as a client. Snort, which is one of the most commonly used lightweight intrusion detection for networks, was also used to detect and create alert notification for various types of attacks on the web server network. The latest version of Snort with all the current rule sets was kept up and running during this experiment (Snort v2.8 Community-Rules). In addition, tcpdump was used to capture all packets entering and leaving the client and server machines. Further details about the topology used are shown in Figure 1. Because this activity can be a very onerous task on a busy server like this experiment simulated web server, not to mention the large amounts of storage required, a reasonable approach was implemented to capture HTTP traffic from only those addresses that were expected to be sending covert messages.

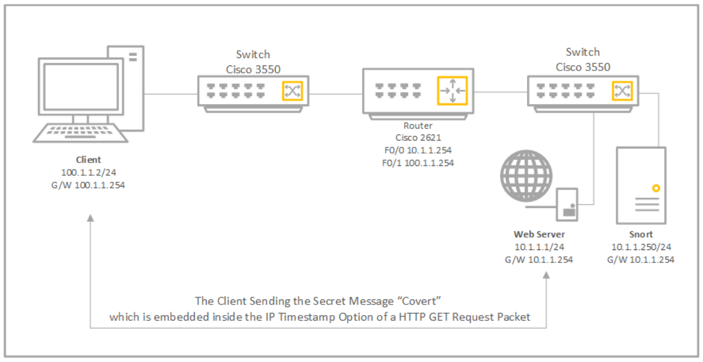
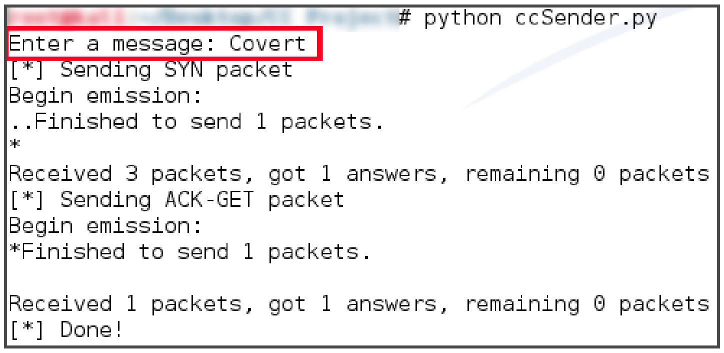


Figure 1. Experiment Topology.

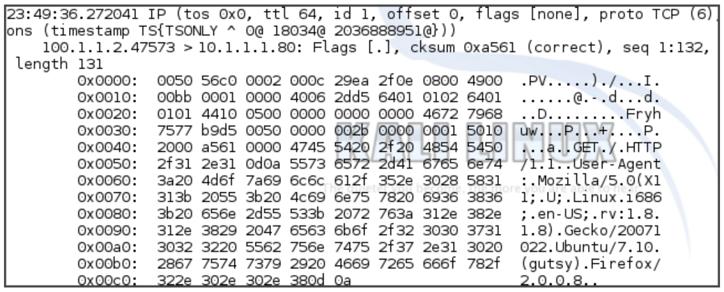
4 RESULTS

With the web server up and running, the client machine established the connection to it on port 80 (HTTP). A string of characters, "*Covert,*" was sent from the client to the server. None of the rule sets on Snort reported any alerts due to the generated traffic from this connection and the covert communication activity. All the relevant packets captured by tcpdump are shown below that prove the covert channel was carried over the network to the server without any issues. The server was able to decode the covert message and displays the covert message as it was intended.



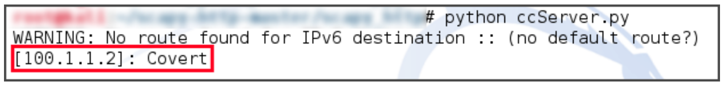
```
[redacted]# python ccSender.py
Enter a message: Covert
[*] Sending SYN packet
Begin emission:
..Finished to send 1 packets.
*
Received 3 packets, got 1 answers, remaining 0 packets
[*] Sending ACK-GET packet
Begin emission:
*Finished to send 1 packets.

Received 1 packets, got 1 answers, remaining 0 packets
[*] Done!
```

Figure 2. The Client Sends Out the Secret Message



```
23:49:36.272041 IP (tos 0x0, ttl 64, id 1, offset 0, flags [none], proto TCP (6)
ons (timestamp TS{TSONLY ^ 0@ 18034@ 2036888951@}))
    100.1.1.2.47573 > 10.1.1.1.80: Flags [.], cksum 0xa561 (correct), seq 1:132,
 length 131
        0x0000:  0050 56c0 0002 000c 29ea 2f0e 0800 4900  .PV.....)./...I.
        0x0010:  00bb 0001 0000 4006 2dd5 6401 0102 6401  ......@.-.d...d.
        0x0020:  0101 4410 0500 0000 0000 0000 4672 7968  ..D.........Fryh
        0x0030:  7577 b9d5 0050 0000 002b 0000 0001 5010  uw...P...+....P.
        0x0040:  2000 a561 0000 4745 5420 2f20 4854 5450  ...a..GET./.HTTP
        0x0050:  2f31 2e31 0d0a 5573 6572 2d41 6765 6e74  /1.1..User-Agent
        0x0060:  3a20 4d6f 7a69 6c6c 612f 352e 3028 5831  :.Mozilla/5.0(X1
        0x0070:  313b 2055 3b20 4c69 6e75 7820 6936 3836  1;.U;.Linux.i686
        0x0080:  3b20 656e 2d55 533b 2072 763a 312e 382e  ;.en-US;.rv:1.8.
        0x0090:  312e 3829 2047 6563 6b6f 2f32 3030 3731  1.8).Gecko/20071
        0x00a0:  3032 3220 5562 756e 7475 2f37 2e31 3020  022.Ubuntu/7.10.
        0x00b0:  2867 7574 7379 2920 4669 7265 666f 782f  (gutsy).Firefox/
        0x00c0:  322e 302e 302e 380d 0a                   2.0.0.8..
```

Figure 3. Tcpdump: HTTP Get Request Packet Carrying the Secret Message.



```
[redacted]# python ccServer.py
WARNING: No route found for IPv6 destination :: (no default route?)
[100.1.1.2]: Covert
```

Figure 4. The Server Correctly Reads the Secret Message.

5 FUTURE WORK

To properly secure the covert message, both the connection and the message will be encrypted with strong encryption methods. An encrypted protocol such as HTTPS will be used for the connection between the client and server instead of HTTP. Also, it is worth noting that if the Caesar Cipher is known to be used for the message encryption, then it is fairly simple to break the encrypted code by a brute force attack. This can be done using a trial and error approach to attack the cipher. Therefore, a stronger encryption technique will be developed to replace the current applied encryption method. Moreover, this covert channel currently offers a one-way communication from the client to the server; thus, more work will be done to ensure that this covert channel features a two-way communication.

6 CONCLUSION

This paper presented a covert channel and the threat that can be presented to any system if used. This covert channel has been demonstrated and analyzed in a "real-life" setting and the results clearly establish the effectiveness and successfulness of exchanging a message by evading stateless IDS tools. The results establish very clearly the effectiveness of tools such as tcpdump in identifying network traffic patterns generated as a result of covert channel activity.

REFERENCES

[1] J. Selvi and R. VandenBrink, "Covert Channels over Social Networks," The SANS Institute (2012):n. pag. Print.
[2] B. W. Lampson, "A note on the confinement problem," Communications of the ACM 16.10 (1973): 613-615.
[3] J. Giffin, R. Greenstadt, P. Litwack, and R. Tibbetts, "Covert messaging through TCP timestamps," Privacy Enhancing Technologies, Springer Berlin Heidelberg, 2003.
[4] Z. Su, "Specification of the Internet Protocol (IP) timestamp option," RFC781 (1981).
[5] S. J. Murdoch and S. Lewis, "Embedding covert channels into TCP/IP," Information Hiding, Springer Berlin Heidelberg (2005).
[6] M. Wolf, "Covert channels in LAN protocols." Local Area Network Security, Springer Berlin Heidelberg (1989), 89-101.
[7] Snort, (2014), Intrusion Detection System, [online] https://www.snort.org (Accessed: 16 December 2014).