Detection of Covert Channels Over ICMP Protocol

Sirine Sayadi, Tarek Abbes, Adel Bouhoula

Detection of Covert Channels over ICMP Protocol

Sirine Sayadi
National School of Electronics and Telecommunications of Sfax
University of Sfax, Tunisia
Email: sirine.sayadi@yahoo.fr

Tarek Abbes
National School of Electronics and Telecommunications of Sfax
University of Sfax, Tunisia
Email: tarek.abbes@enetcom.usf.tn

Adel Bouhoula
Higher School of Communication of Tunis
2083 Cité El Ghazala, Tunisia
Email: adel.bouhoula@supcom.rnu.tn

***Abstract*—With the growing complexity of networks and communications protocols that become increasingly enormous and extensive, we are confronted with the problem of covert channel that affects the confidentiality and integrity of data sent in the network. Covert channels also known as hidden channels can elude basic security systems such as Intrusion Detection Systems (IDS) and firewalls. We propose in this work a method to monitor and detect the presence of hidden channels that are based on an essential monitoring protocol "Internet Control Message Protocol" (ICMP). We undergo the network traffic with a set of verifications ranging from simple fields verification to more complex pattern matching operations. To validate our idea, we have installed Ptunnel, a tool that allows to tunnel TCP connections to a remote host using ICMP echo request and reply packets. Our experimental results show the possibility to discover such malicious traffic with high performance.**

Keywords*—*Network Security; Covert Channel; Storage Channel; Traffic analysis; ICMP protocol, ICMP Tunneling; Tunneling Detection.

I. Introduction

The evolution of computer networks in recent years has led to the development of new technologies and services but also the emergence of new threats that destroy our data. According to Lampson [2], attacking covert channels occurs when the communication of information between two parties is secretly done via a chain which is not intended for the sending of information. The purpose of using hidden channels is to send data in the network while ensuring that the sending is unnoticed by a third party and without alerting any firewalls or Intrusion Detection Systems (IDS) on the network. This is done for several purposes such as (a) using the resources of others to send secret data, (b) data leakage, (c) installation, distribution and control of malicious software, (d) bypass security devices such as firewall, etc.

In the literature, covert channels can be classified into two categories, timing and storage channels [4]:

- A Timing Channel is a covert channel in which a process reports information to another process while manipulating the time it takes to perform a given operation. The other process can observe to understand what happened.
- Storage Channels are based on shared data storage areas such as Unused or optional protocol fields. They include all means that would allow the direct or indirect writing at the storage area by a process and the direct or indirect reading of it by another one.

With the increased use of hidden channels that violates security policies, it has become necessary to find counter measures to detect this type of attacks. Several research works were done to defeat covert channels. They are mainly two countermeasure approaches. The first one relies on actively modifying optional and unused protocol fields whether there is or not a covert channel. The method has the drawbacks to change even safe traffic, introduces some delays and consumes computing resources. The second approach passively supervises network traffic and undertakes some controls looking for suspicious activities. In this case, it notifies the administrator about the presence of covert channels. In this work, we are interested in storage covert channels and we apply the passive monitoring approach to tackle this problem. We conduct many verification controls:

- Check ICMP messages size.
- Check the number of Echo Reply messages for a given Echo Request.
- Check if there is a spike in the number of ICMP messages.
- Check patterns.

The remainder of this paper is structured as follows: Section II describes some related works. Section III explains ICMP protocol and presents our detection method. Section IV discusses the experimental results. We devote Section V to assess the detection performance of our method. We draw in Section VI the conclusion and we mention some future works.

II. Related Works

Several researchers have oriented their research axes to detect covert channel attacks using multiple methods and techniques. Covert channels can be divided into Timing covert channels and Storage covert channels. In a timing covert channel, the bits of the covert information are represented by varying the time interval between the sent packets. S. Gianvecchio and H.Wang in [1] propose a detection system that measures changes in entropy and in corrected conditional entropy (CCE) within the network to determine if timing channels are being used. A change in the entropy of a process provides a critical clue for covert timing channel detection. P. L. Shrestha et al [6] provide a framework based on Support Vector Machine (SVM) classifier to detect covert timing channel. They derive four popular fingerprints (K-S test, regularity score, entropy, and corrected conditional entropy) from the traffic under investigation to form the SVM classifier.

By mounting storage covert channels, the hidden data is written in optional or unset protocol fields. The research works can be grouped according to the exploited network protocol. Using TCP protocol, E. Tumoian and M. Anikeev analyzed the accuracy of a neural network to detect the covert channel in the TCP Initial Sequence Number (ISN) field [9]. First, the neural network was formed to predict successive ISNs for different operating systems. Then, they monitored the ISNs used and compared them to the prediction models by measuring their statistical deviance. If an ISN sequence did not match any pattern indicating a normal deviation of the generated ISN then it may be a covert channel. H. Song and X. Li calculated an anomaly score that corresponds to the sum of weighted frequencies on all flag combinations in one second [7]. A high score indicates a suspect traffic. Then, the authors monitored and tested the decrease of TCP sequence numbers, considered as abnormal. They completed their traffic analysis by calculating the variance of TCP sequence numbers within a moving time window then compared it against the variance baseline for a normal traffic. A deviation may reveal the presence of hidden channels.

Using ICMP protocol as a medium for a covert channel, Sohn, et al. [5] proposed a detection method based on Support Vector Machine (SVM) to learn ICMP packets payload characteristics. They used Loki2 to generate abnormal ICMP packets and Tcpdump to capture network traffic. They preprocessed collected packets to extract the characteristic of ICMP packets headers and payloads. After a training phase, they built an SVM classifier able to discern suspect ICMP packets. In [8] K. Stokes et al. tested the efficiency of firewalls and IDS to detect ICMP based hidden channels. They concluded that covert channels can easily defeat many modern security appliances if ICMP protocol is permitted. In [3] D. N. Muchene et al. transformed the covert channels to a secure reporting system of security alerts. They considered that this covert network communication combined with sophisticated insider threat monitoring tools, will improve security level and raise the costs and risks for intruders.

Existing research works consider that security technologies such as Firewalls and IDS cannot detect exactly the attacks of covert channels. Besides, exploited protocols are fundamental, so they cannot be deactivated. Regarding the capital importance of these protocols and the fact that they are very much used, we propose a detection method of hidden channels that rely on ICMP protocol.

III. Protocol Design

A. ICMP Overview

Internet Control Message Protocol (ICMP) is one of fundamental protocols of TCP/IP stack. It operates in the network layer and is mainly used to send control and error messages between endpoints. An ICMP packet is usually generated as a result of a diagnosis by another network protocol or a failure of transmission. As shown in Figure 1, ICMP header is composed of several fields that are Type, Code and Checksum. Other fields may be defined in the Data part such as the Identifier, sequence numbers in ICMP echo request (type 8) and reply (type 0).

The size of fields “type” and “code” is one byte. These latter indicate the purpose of the ICMP packet. The Checksum field is useful to check the integrity of the ICMP message. The identifier number and the sequence number are encoded with two bytes. They allow to match the ICMP echo reply messages to the echo request messages. Finally, the content of the Data can be filled by fixed bytes padding that depends on the operating system.

For Linux, the ICMP Echo Request payload contains 56 bytes, among them 8 bytes are used as a timestamp and the remainder match a fixed pattern. For the Windows operating system, the ICMP payload contains 32 bytes of sorted alphabets.

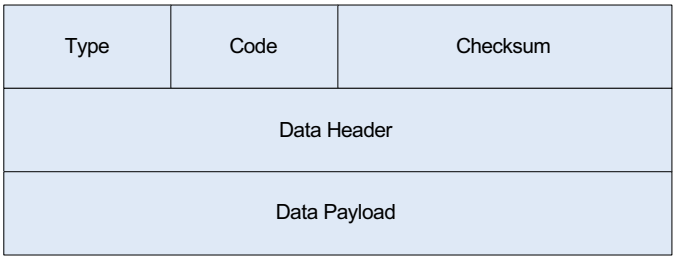

Fig. 1: ICMP frame format

B. Proposed Method

In order to discover covert channels, we apply two categories of verification. In the first step, we control the shape of ICMP messages. If suspicious activity is revealed, we pursue our inspection through pattern matching operations, which are considered greedier in term of computing resources. Figure 2 resumes the different verification controls that we will perform.

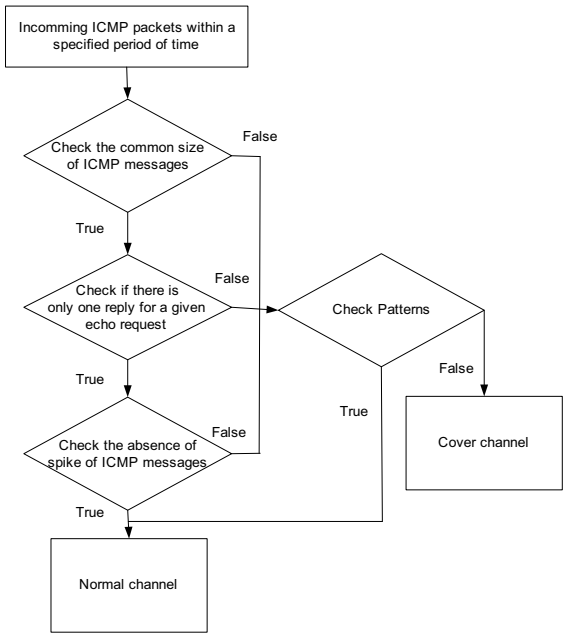

Fig. 2: Proposed Method

1) Stage 1 - Analysis of ICMP messages shape:

During the stage 1 of our inspection, we start the verification with the size of ICMP data field. Mostly, this size must be equal to 32 bytes (Windows) or 56 bytes (Linux). If it is not the case, we trigger the second stage of inspection. Otherwise, we execute a second verification control at stage 1. During this test, we compute the number of ICMP echo replies associated to the same echo request. Association is ensured if there is a correspondence between the identifier and the sequence number. Usually, it must be only one association. If it is not the case, we start Stage 2 with pattern matching operations. Otherwise, we execute the last verification control in stage 1. In this case, we calculate the rate of ICMP messages regarding the rest of network traffic in a fixed size sliding window. If we notice a large increase if this rate, we start stage 2 of inspection. Else, we consider that the ICMP message is normal.

We use the sliding window principle to calculate the ICMP message rate with respect to the whole traffic. Fig. 3 shows the computing inside the sliding window. Each window is composed of fixed number of time intervals (IT). Detection is only triggered with the reception of an ICMP packet. Besides, we use the previous ICMP packet to calculate the window's shift. For example, the interarrival time between two consecutives ICMP packets is about 3 interval times in Fig. 3. In this case, we have to update only IT0, IT1 and IT2 and keep IT3 to compute the ICMP messages rate at window 2.

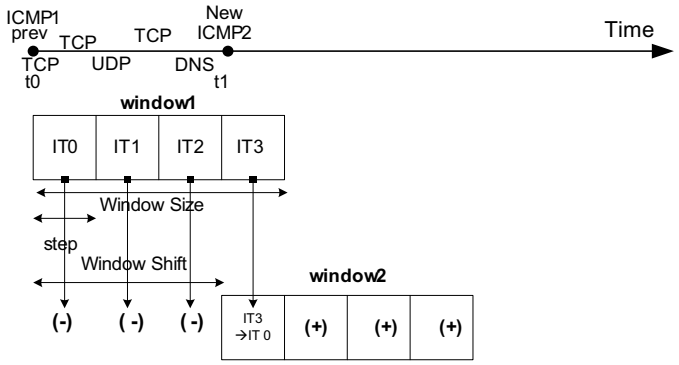

Fig. 3: Sliding Window

We present in Algorithm 1 the different tests achieved during the stage 1. We define the two functions "count_icmp_pkt" and "count_all_pkt" to compute respectively the number of ICMP messages and all received packets. Besides, the function "count_icmp_same_addr_Id" calculates the number of ICMP messages that have the same identifier or sequence number as the current analyzed packet "p". If we succeed in one test of stage 1, then we use the function "phase2_inspection(p)" to start stage 2.

2) Stage 2 - Analysis of ICMP messages content:

We are looking in this stage at the content of ICMP messages. Mostly, the operating systems use a constant payload to build the ICMP message. For example, Microsoft Windows uses the pattern "0x61 0x62 ... 0x77" while Linux employs the pattern "0x08 0x09 ... 0x37". We can inspect

```
foreach packet p in logfile do
    X=count_all_pkt(p, prev, Win.size);
    if p.proto=ICMP then
        Y=count_icmp_pkt(p, prev, Win.size);
        Z=count_icmp_same_addr_Id(p,prev,Win.size);
        if (p.size≠32) and (p.size≠56) then
            stage2_inspection(p);
        else
            if Z > 2 then
                stage2_inspection(p);
            else
                if Y / X > Threshold then
                    stage2_inspection(p);
                else
                    p is normal;
                end
            end
        end
    end
end
```

Algorithm 1: Stage 1

all bytes in ICMP data field. However, such action will consume a lot of memory and time. To resolve this problem, we introduce two heuristics to ensure the verification. Our first strategy is to compare N successive random bytes from a random position pos. We present in Algorithm 2 such verification.

```
Let pattern_Linux the constant content of an ICMP
message sent by a Linux machine ;
Let pattern_Win the constant content of an ICMP
message sent by a Windows machine;

N=rand mod 8+1;
Pos=rand mod 24 +1;
Sub_pattern_Linux= substring(pattern_Linux,N,pos);
Sub_pattern_Win= substring(pattern_Windows,N,pos);
Sub_pkt_Linux =substring (p.data+8,N,pos);
Sub_pkt_win=substring (p.data,N,pos);
Find_Linux=match(Sub_pkt_Linux,Sub_pattern_Linux);
Find_Win=match(Sub_pkt_Win,Sub_pattern_Win) ;

if Find_Linux or Find_Win then
    Then p is normal;
else
    p is a covert channel at p.timestamp;
end
```

Algorithm 2: Stage 2 using random pattern matching

We define another content verification using the hash function MD5. Indeed, we compute for each pattern its hash value on 128 bits. Then, for each analyzed packet, we check if the hash of the packet content is identical to one of the two possible hash values. In case of equality, we decide that the packet is legitimate, else we mark the packet as a part of a

covert channel. We describe this verification in Algorithm 3.

Let $pattern\_Linux$ the constant content of an ICMP message sent by a Linux machine ;
Let $pattern\_Win$ the constant content of an ICMP message sent by a Windows machine;

Hash_Linux=MD5(P.data+8);
Hash_Win=MD5(P.data);
Find_Linux=compare(Hash_Linux,Hash_pattern_Linux);
Find_Win=compare(Hash_Win,Hash_pattern_Win);

if *Find_Linux or Find_Win* **then**
    Then p is normal;
else
    p is a covert channel at p.timestamp;
end

Algorithm 3: Stage 2 using MD5 hashing

IV. Experimentation

A. Experimental environnment

In order to mount our experiment, we setup two virtual machines on VMWAR. Besides, we install an attack tool called Ptunnel on two machines to serve as a Ptunnel client and a Ptunnel server (See Fig. 4). Ptunnel allows to reliably tunnel TCP connections to a remote host through ICMP packets. Traffic can be analyzed on an intrusion detection system deployed between the machines. It's also possible to retrieve traffic at the destination machine (Ptunnel server) as the latter can be considered as the target of the attack.

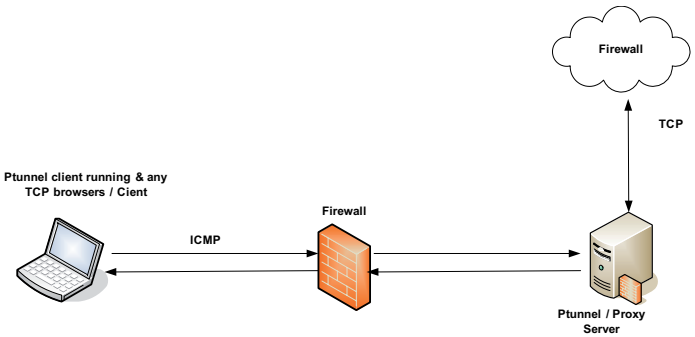

Fig. 4: Ptunnel Setup

The configuration of the Ptunnel server is easy and requires only to indicate the reception network interface. Hence, it suffices to tape this command "Ptunnel -c interface" .

The configuration of the Ptunnel client varies regarding the intended traffic to vehiculate. For example, if we would like to tunnel http traffic, we have to reconfigure the browser proxy on the client machine to localhost:8000. Besides, we execute the command " Ptunnel -p addr_serv -lp 8000 -da dest_addr -dp dest_port" where lp, da and dp represent respectively the listen port, the destination address after reaching the proxy and the port that the proxy should use to tunnel the TCP connection. Fig. 5 gives the result of such execution when we would like to reach the url "www.google.tn". We can notice that the number of echo reply messages exceeds the one of echo requests. Besides, the ICMP payload contains the string "HTTP 1.0".

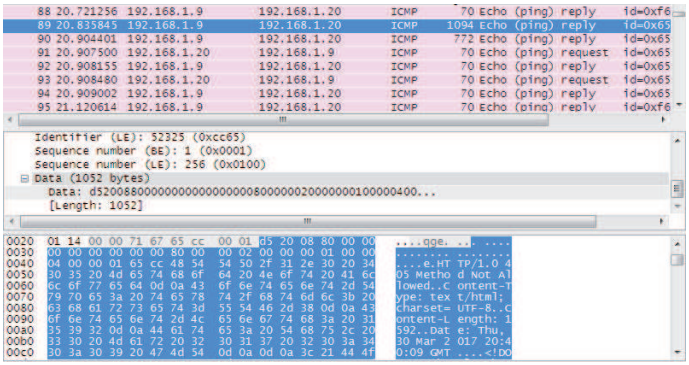

Fig. 5: Captured packets after the execution of ptunnel

We can also tunnel other types of traffic. For example, the command " Ptunnel -p addr_serv -lp 8000 -da ssh_dest_addr -dp 22 " permits to vehiculate ssh traffic through ICMP messages.

To discern the existence of hidden channels, we have to capture network traffic and to conduct our analysis. The inspection is done on 2 stages. First of all, we look for unusual ICMP messages size, abundance of ICMP echo reply messages with the same identifier and a spike of the number of ICMP packets. Then, we confirm our inspection by filtering the ICMP data field searching for habitual patterns.

During our experiments, we execute Wireshark to capture network traffic. We export collected packets into PDML file. PDML is a structured XML document that contains details about packets dissection. We develop our proper program to parse XML using DOM (Document Object Model) programming API.

B. Exeprimental results and discussion

We carried out a set of experiences in presence of normal ICMP as well as covert channels. The hidden channel can be the result of interaction with a Web server or a SSH server. We compare in the first experiment the size of Data field of the ICMP messages. As shown in Fig. 6, the normal ICMP traffic has two possible sizes: 32 bytes in case of a Windows operating system and 56 bytes when dealing with a Linux machine. A covert channel interacting with a Web server modifies the common size of ICMP data field. Indeed, when tunneling the protocol HTTP, the ICMP data field has the size 28 bytes if it carries HTTP requests, and different bigger sizes (above 500 bytes in our case) if it encapsulates HTTP replies.

Fig. 6 displays also the effect of tunneling a SSH session inside an ICMP based covert channel. The size of ICMP data field varies from 28 bytes to 200 bytes. Small packets are explained by the interactive nature of SSH application. Accumulation of sent bytes is possible thanks to Nagle Algorithm.

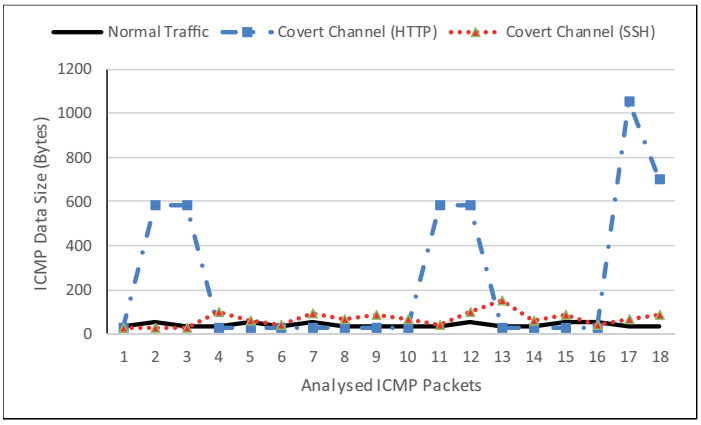

Fig. 6: ICMP Data Size in Normal Traffic and Covert Channels

Our second experiment verifies the occurrence of the same ICMP sequence number. We notice in Fig. 7 that in normal traffic each sequence number figures twice, one for the echo-request message and another for the echo-reply message. However, the number of utilization becomes variable in presence of covert channels interacting either with HTTP or SSH servers.

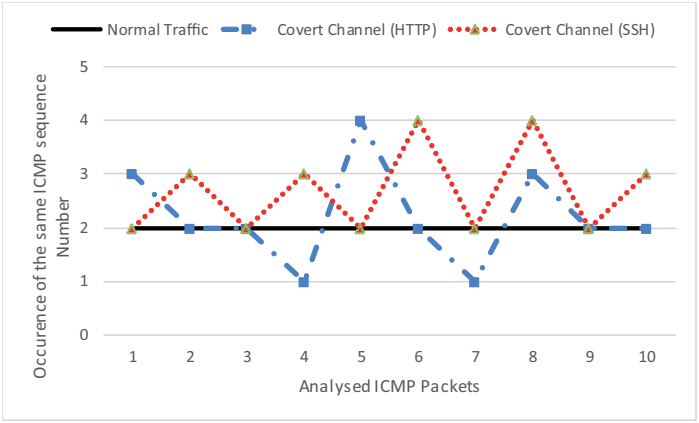

Fig. 7: Number of Utilization of the Same ICMP Sequence Number

Eventually, we evaluate the rate of ICMP messages with respect to the rest of traffic. The computation is done within a window size of 4 seconds with a step equals to 1 second. Fig. 8 shows that in normal conditions, ICMP traffic is not too present, and never exceeded in our experimentation 0.37. Hence, we choose this value as a threshold to mark an abnormal ICMP traffic. Covert channel toward a HTTP server leads the ICMP messages rate to reach 0.9, whereas the interaction with a SSH server, can bring this rate up to 0.7.

DoS attacks can be misinterpreted as hidden channels since they also cause a spike of ICMP packets. To avoid this false alarm, we verify in Phase 2 of the detection process, the content of ICMP messages.

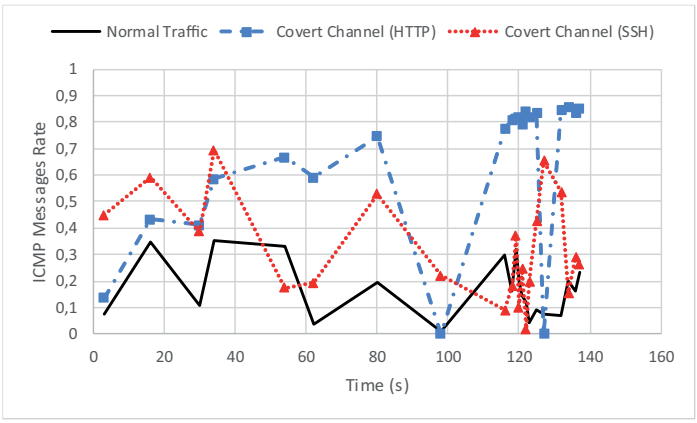

Fig. 8: ICMP messages rate

V. PERFORMANCE EVALUATION

We applied our detection method on a traffic generated by standard ping, personalized ping and the Ptunnel tool. Standard pings are issued from Windows and Linux operating systems. Besides, under Linux, we employ two options that modify the standard ping, the packet size (-s *packetsize*) and the payload (-p *pattern*). Ptunnel is used to communicate with a HTTP server and a SSH server. In total, we have 1266 ICMP packets, scattered in the network traffic. There are 751 ICMP messages that belong to covert channels, the other part (515 ICMP packets) is normal ICMP exchanges.

We show in Table I the confusion matrix to depict our detection results. In this execution, we choose to rely on the hashing test to ensure the phase 2 of our inspection. We succeed to reveal all malicious ICMP packets (751 messages), without generating any false negatives. False negatives are still possible since we could have collision with MD5 hashing, but this is very scarce. However, we detect 8 false positives due to changing the size of ICMP messages.

TABLE I: Confusion Matrix using Hashing Function

Prediction → Actual traffic ↓	Covert channel	Normal traffic
Covert channel	751	0
Normal traffic	8	507

We draw in Table II our detection results when using random pattern matching. In this execution, we detect only 745 malicious ICMP packets. The obtained 6 false negatives are due to a tricky modification of the ICMP messages pattern. In fact, the intruder inserts a secret message, just after the common pattern used by the operating system. Besides, we get 7 false positives because a legal modification of the ICMP data field.

TABLE II: Confusion Matrix using random Pattern Matching

Prediction → Actual traffic ↓	Covert channel	Normal traffic
Covert channel	745	6
Normal traffic	7	508

We compare in Table III the performance of the two ICMP content analysis methods. We give successively the True Positive Rate (TPR) or precision, the True Negative Rate (TNR), the False Positive Rate (FPR), the False Negative Rate (FNR), the accuracy and the time execution. The two methods present almost the same performance. Random Pattern Matching generates more false negatives but it is faster than hashing function. Since covert channels may require many packets to transfer information, we can rely on Random Pattern matching to discover them. Occasionally we execute MD5 hashing to prevent possible false negatives.

TABLE III: Performance of Covert Channel Detection

	TPR	TNR	FPR	FNR	Accuracy	run time
Hashing function	1	0.98	0.015	0	0.99	353 ms
Random Pattern Matching	0.99	0.98	0.013	0.008	0.98	4 ms

VI. CONCLUSION AND FUTURE WORK

A covert channel is a stealth communication mechanism that allows two or many participants to exchange unauthorized data. It represents a real and dangerous threat that affects the confidentiality and availability in an information system. In addition to data leakage, covert channel can be a means to bypass security devices and to control remote machines.

We propose in this paper a promising method to detect ICMP based covert channels. We undergo the received traffic a series of tests that begin with a simple size verification. Besides, we inspect in a sliding window, the ICMP messages rate as well as the replication of ICMP sequence numbers. In case of suspect traffic, we pursue our analysis by conducting a fast random pattern matching or a more accurate hashing verification test. Our experimentation shows that the two detection methods have almost the same accuracy and precision. However random pattern matching is faster.

As a future work, we will investigate the generalization of our method to discover other types of covert channels using any network protocol. Besides, we will conduct time series analysis in order to tackle the problem of temporal covert channel.

REFERENCES

[1] S. Gianvecchio and H. Wang, "An entropy-based approach to detecting covert timing channels," *IEEE Trans. Dependable Sec. Comput.*, vol. 8, no. 6, pp. 785–797, 2011.

[2] B. W. Lampson, "A note on the confinement problem," *Commun. ACM*, vol. 16, no. 10, pp. 613–615, Oct. 1973.

[3] D. N. Muchene, K. Luli, and C. A. Shue, "Reporting insider threats via covert channels," in *2013 IEEE Symposium on Security and Privacy Workshops, San Francisco, CA, USA, May 23-24, 2013*, 2013, pp. 68–71.

[4] H. Okhravi, S. Bak, and S. T. King, "Design, implementation and evaluation of covert channel attacks," in *2010 IEEE International Conference on Technologies for Homeland Security (HST)*, Nov 2010, pp. 481–487.

[5] T. Shon, J. Moon, S. Lee, D. H. Lee, and J. Lim, "Covert channel detection in the ICMP payload using support vector machine," in *Computer and Information Sciences - ISCIS 2003, 18th International Symposium, Antalya, Turkey, November 3-5, 2003, Proceedings*, 2003, pp. 828–835.

[6] P. L. Shrestha, M. Hempel, F. Rezaei, H. Sharif, undefined, undefined, undefined, and undefined, "A support vector machine-based framework for detection of covert timing channels," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 2, pp. 274–283, 2016.

[7] H. Song and X. Li, "Collaborative detection of covert storage channels," in *2016 IEEE Military Communications Conference, MILCOM 2016, Baltimore, MD, USA, November 1-3, 2016*, 2016, pp. 515–520.

[8] K. Stokes, B. Yuan, D. Johnson, and P. Lutz, *ICMP Covert Channel Resiliency*. Dordrecht: Springer Netherlands, 2010, pp. 503–506.

[9] E. Tumoian and M. Anikeev, "Network based detection of passive covert channels in tcp/ip," in *The IEEE Conference on Local Computer Networks 30th Anniversary (LCN'05)*, Nov 2005, pp. 802–809.