Raising Flags: Detecting Covert Storage Channels Using Relative Entropy

Josephine Chow
University of Maryland, College Park
College Park, Maryland, US
kchow8@umd.edu

Xiangyang Li
Johns Hopkins University Information Security Institute
Baltimore, Maryland, US
xyli@jhu.edu

Xenia Mountrouidou
College of Charleston
Charleston, South Carolina, US
mountrouidoux@cofc.edu

***Abstract*— This paper focuses on one type of Covert Storage Channel (CSC) that uses the 6-bit TCP flag header in TCP/IP network packets to transmit secret messages between accomplices. We use relative entropy to characterize the irregularity of network flows in comparison to normal traffic. A normal profile is created by the frequency distribution of TCP flags in regular traffic packets. In detection, the TCP flag frequency distribution of network traffic is computed for each unique IP pair. In order to evaluate the accuracy and efficiency of the proposed method, this study uses real regular traffic data sets as well as CSC messages using coding schemes under assumptions of both clear text, composed by a list of keywords common in Unix systems, and encrypted text. Moreover, smart accomplices may use only those TCP flags that are ever appearing in normal traffic. Then, in detection, the relative entropy can reveal the dissimilarity of a different frequency distribution from this normal profile. We have also used different data processing methods in detection: one method summarizes all the packets for a pair of IP addresses into one flow and the other uses a sliding moving window over such a flow to generate multiple frames of packets. The experimentation results, displayed by Receiver Operating Characteristic (ROC) curves, have shown that the method is promising to differentiate normal and CSC traffic packet streams. Furthermore the delay of raising an alert is analyzed for CSC messages to show its efficiency.**

Keywords—covert storage channel; relative entropy; anomaly detection

I. INTRODUCTION

Over the past few decades, concern about information security has increased dramatically as private and federal businesses discover more frequent traces of data breaches and unauthorized information acquisition. Advances in hacking have made these breaches easier to hide and harder to detect. Attackers can transmit messages containing valuable information undetected through an interconnected network like Internet, using covert communication channels.

Covert channel communication uses an otherwise legitimate medium of storage or time to transmit messages secretly. Covert channels pose a considerable threat and often facilitate various cyber-attacks in different stages, such as the secret command control of a botnet or exfiltration of stolen personal data. In general, they are categorized as Covert Storage Channel (CSC) and Covert Timing Channel (CTC) [1]. A CSC writes to a storage location by a sender process and then a receiver process reads the message according to a pre-defined coding scheme of information [2]. In order to transmit information secretly, a CTC schematically modulates the temporal characteristic of a process by a sender, which is then measured by the receiver.

Unlike most other exploits, covert channels often go unnoticed by the access control mechanisms of most operating systems. They can use a regular medium, such as unused protocol bits in a networking protocol, in a way that does not explicitly violate security policies. While both covert channels constitute substantial threats, CSCs are more common because they are relatively easier to implement [3]. However, a significant challenge to CSC analysis is there is no real data available.

This study took into consideration different information encoding schemes and strategies that can be used for a CSC. In detection, we employed two data processing methods, with one based on individual network traffic flows between computers and the other using a moving window to process such flows in snapshots. We compared the detection performance of our method for plaintext and encrypted CSC messages, as well as considered the different TCP flags used in encoding. Interestingly this study has discovered a dilemma for the covert communication accomplices: although encryption will make an intercepted message difficult to read, it will increase the chance of the message to be detected.

II. RELATED LITERATURE

Communication through covert channels can be traced back to steganography. Steganography is the practice of concealing secret messages or information within a non-secret message so that only the intended receiver understands the secret message. In the digital world, steganography means hiding information in digital files or other digital structures. With advances in technology and tools that have been developed to detect steganography within raw texts, attackers have moved from transmitting messages through texts to other mediums.

This section will review related efforts in detecting covert channel traffic in general. Covert channels can leak

NSF awards 1525485 and 1700254 supported this work.

information to unauthorized individuals or introduce unauthorized changes to a system. It is important to develop efficient detection mechanisms that overcome their threat. However, covert channels are hard to detect and often go unnoticed. Not enough research has been conducted, especially on CSC's. The high complexity and variation in legitimate network traffic makes detection of an embedded CSC even more challenging.

A. Covert Timing Channel (CTC)

A CTC manipulates the time of a process to transmit information, in either active or passive manners [4]. A passive CTC manipulates the timing or ordering of existing network traffic events or packets. On the other hand, an active CTC generates synthetic traffic to transmit information covertly. In [4], the authors tried to identify the existence of irregular traffic based on the observation that the generation of CTC will alter the entropy of the timing of original traffic. Similarly to our study, the authors used entropy-based tests in CTC irregularity detection. In another study [5], CTC detectability is analyzed using statistical methods, including relative entropy and other dissimilarity measures.

Since all processes on a computer follow a time clock, every process is vulnerable to become a covert timing channel. Implementation and testing of a CTC using network packet inter-arrival timing for detectability and robustness is described in [6]. However, a disadvantage of using a CTC is that the medium variation and environment noise make it hard to be practical. Moreover, such communications also need to overcome changes introduced by firewalls and routers over a computer network. In [7], the authors analyzed CTC encoding schemes with statistical testing of its robustness, and described how they could adapt the design by repeating redundant bits to reduce error rate and adding randomness to defeat detection.

B. Covert Storage Channel (CSC)

Covert storage channels are used for secret communication by manipulating an otherwise legitimate medium of storage [8]. Thus, the sender writes to this shared medium, while the receiver process reads it and decodes the message. Since a CSC is simply using a medium that one process writes to and another reads, any kind of digital storage including emails, websites, instant messaging, TCP/IP networking, and shared file systems can be deployed in CSC communication.

Here we focus on TCP/IP network packets that can be prime CSC media. Protocols such as ICMP, TCP, and IGMP are ideal carriers for covert channels. Sorting and rearranging packets often help to set up covert channels [9]. In this study, specifically TCP flag headers are used to encode messages. Fig. 1 shows an example, only for illustration, of a CSC message. If the accomplices use valid flags to communicate, it would be very challenging to detect it without paying close attention. These TCP flags are necessary elements to support a range of network protocols, and so the accomplices' packets may look like the rest of network traffic [10].

An advantage of using network packets for CSC is that malicious activities are hard to notice. To implement a storage channel successfully, attackers target computer networks with heavy traffic in a hope of hiding a covert channel signal in the regular traffic [3]. A computer network that hosts a high volume of regular network traffic may hide covert messages in such "noise." It is arduous to detect illicit use of covert channels in such noisy channels. However, if the network is too noisy, it might reduce the bandwidth that the covert channel can utilize, decreasing its capacity.

TCP Flags	Meaning
101010	J
101000	H
110101	U
101001	I
110011	S
101001	I

Fig. 1. An illustration of a covert storage channel using the 6-bit TCP flag of network packets

Some computer scientists have been studying ways to analyze CTCs using similarity index, however inadequate efforts have been aimed at developing an effective mechanism to detect the existence of CSCs [2]. In a previous study, Mahalanobis distance was used to detect the irregularity in TCP flag frequency distributions [10]. This paper presents a study that uses the Kullback-Leibler divergence, also known as relative entropy (RE) [11], to compare the difference of TCP flag frequency distributions. Moreover, we consider different data processing techniques and assumptions about the CSC coding scheme.

III. Approach

In our study, first we build a normal traffic profile as the baseline of regular network traffic, represented by the frequency distribution of TCP flags. In testing, we calculate the same distribution as observation, and compare it to the baseline. A higher RE value in this comparison shows that the observation deviates more from the normal traffic, an indicator of greater chance of the existence of a CSC.

A. Irregularity Measure

The Relative Entropy (RE), is a metric of difference between two sets of probability distributions, P and Q [11], shown below for discrete variables as Equation (1):

$$D_{KL}(P||Q) = \sum_i P(i) log \frac{P(i)}{Q(i)} \quad (1)$$

where Q is a theoretical model distribution, e.g., TCP flag frequencies expected for normal network traffic, and P is a real observation, e.g., TCP flag frequencies of the testing network traffic.

P generally represents the probability function of a "real" process, while Q represents a theoretical model as the expectation of the normal traffic, i.e., a normal profile. In our study, P is what we observe in the traffic in testing, using the frequency distribution of TCP flags. Without knowing its true nature, this distribution could be from legitimate network traffic or CSC traffic. On the other hand, Q represents the model of regular traffic, i.e., the probability distribution of TCP flags extracted from normal traffic traces used for training. Using relative entropy, we are able to measure the randomness of data set P against a regular traffic profile. The larger the RE

value, the less likely the observation P is from the normal traffic distribution Q.

Fig. 2 and Fig. 3 show insights on how it is likely to capture irregularities in TCP flags. Fig. 2 plots the frequency distributions of the TCP flags for two regular network traffic datasets from the CAIDA database [10]. Their similarity is obvious.

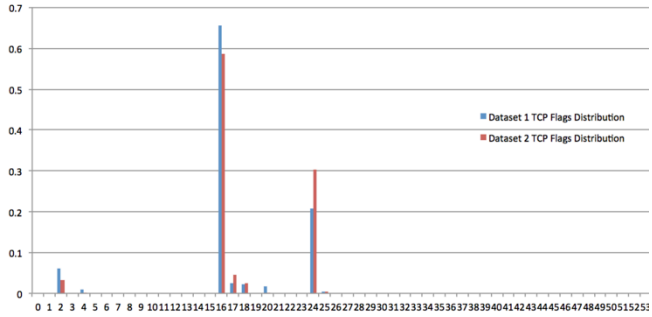

Fig. 2. TCP flag frequency distributions for two regular traffic datasets, where x: TCP flag and y: frequency [3].

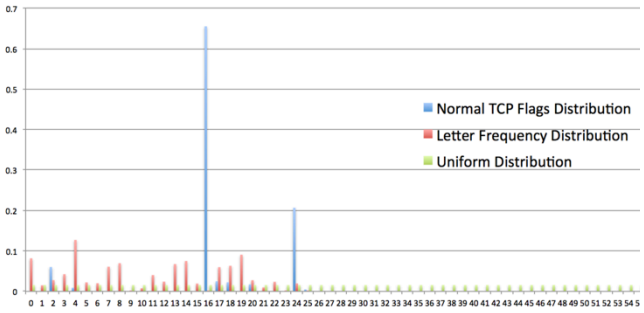

Fig. 3. TCP flag distribution of a normal traffic dataset versus other distributions, where x: TCP flag and y: frequency [3].

Fig. 3 shows two additional distributions of the English letter frequency distribution and uniform distribution. It is clear that the TCP flag distribution from normal traffic is different from them.

The RE between either of these two distributions and the expected normal traffic profile, as Q, should return a large value because of their obvious difference. The larger this value, the more likely this frequency distribution as observed comes from an irregular TCP flag generation process, likely a CSC considered in this study. In testing, we can set different thresholds on such RE values to raise alarms.

B. Normal Training Data

To experiment with our approach, we used real regular network traffic from the MAWI dataset [12]. We divide this regular traffic into a training dataset of 5 million packets, which is used to generate the regular traffic profile, or in another work, the normal model Q of TCP flag frequencies, as a baseline for comparison, and a testing dataset of another 5 million packets to provide normal traffic samples. This normal profile of TCP flag frequencies is shown in Fig. 4.

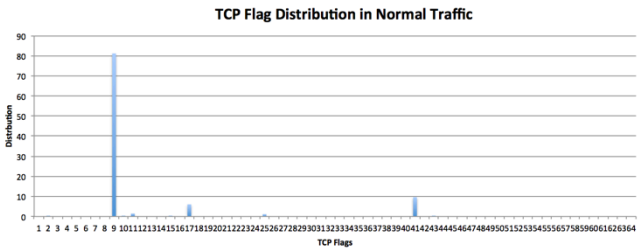

Fig. 4. TCP flag frequency distribution as the normal profile, which is summarized of a training dataset.

C. CSC Traffic Generation

CSC traffic is needed in testing. However, one challenge that we have faced is that no real data are available for this type of CSC traffic. In this study, we generate synthetic CSC traffic for testing. Each CSC message has 140 characters, the length of a tweet. We differentiate our experiments to cover a range of several coding schemes, in terms of clear or encrypted text, as well as the use of TCP flags.

1) Clear Text Versus Encrypted Text

First, we consider the message content. The first scenario uses a common keyword list compiled of Unix systems. We convert every letter of a message to its ASCII code and then use the 6-bit TCP flags to transmit all the 140 letters. Since the ASCII coding uses 8 bits per letter, we need more than one packet to transmit one letter.

In the second scenario, we generate the TCP flags in CSC packets according to uniform distribution, which simulates the effect of encryption on a message. Our hypothesis is that an accomplice may like to secure these messages by scrambling them.

2) Use of TCP Flags

Next, we consider how to use TCP flags to encode the ASCII code after converting letters in a message. In the first scenario we deploy all possible 64 TCP flags in coding the 8-bit code of a letter, resulting in 187 packets per CSC message. However, there is only a small subset of the 64 possible TCP flags that is ever used in all network protocols. Practically it would be easy to flag such a CSC message if invalid TCP flags are seen.

In the second scenario, a smart accomplice uses only the TCP flags ever used in the normal traffic. There are only 9 different TCP flags in the MAWI dataset in this study. In order to transmit a letter, we first convert the 8-bit ASCII value to its ternary code in 6 digits. Then every 2 digits will be mapped to one of the 9 TCP flags. For example, a ternary value of "21" has the decimal value of 7, represented by the 7th TCP flag. In this coding, each CSC message of 140 letters results in 420 packets.

Thus, considering the combinations of message content and the use of TCP flags, we generated four different datasets.

D. Data Processing and Calculation

To calculate RE we will need the observation distribution P of the testing data. We process the testing data, both the normal dataset and CSC messages, with two different methods and

calculate the RE value with regard to the normal traffic distribution profile.

1) Per-Flow Analysis

In this so-called "per-flow calculation," we simply aggregate all the packets that belong to one IP pair, called one flow. Then we summarize the TCP flag frequencies of each flow and calculate the RE value per flow.

First, we separate and group network packets together for each unique IP pair, as the sender and the receiver of a network traffic flow. We then compute the TCP flag frequency distribution P for each flow. So the RE value is calculated of two distributions of P and the Q. Finally, a flow is flagged if this RE value is higher than the chosen control threshold.

This calculation is straightforward but it needs all the packets of a flow available for detection. So, this processing has to be done offline and that introduces delay in detection.

2) Packet Window Analysis

The second method is to generate relative entropy values with a moving window over each flow, in order to reduce the delay in detection. This can more quickly generate observations while the network traffic is still active for an IP pair. This may be used for either offline or online calculations and does not require all the packets that belong to the same flow to be collected.

We call this approach "packet window calculation." The number of packets in one packet window frame, the window size, is a control variable that we can vary. Moreover, a new window frame is created when we slide this moving window frame in time, to replace a few oldest packets at the beginning in the current frame by new packets coming in. The amount of old packets to be replaced each time is another control variable.

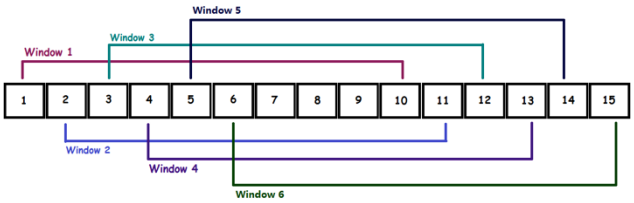

Fig. 5. A standard sliding moving window implementation.

Similarly to the "per flow" method, we first separate and group network packets according to sender and receiver IP addresses into individual flows. Then we arrange these network packets in their chronological order. Shown in Fig. 5, the moving window starts from the first packet of a flow and the number of packets in the current frame, according to the given window size, is examined to generate an observation P. Then we can slide the window to create the next frame, by removing the first segment of packets (e.g., 10% of the current frame) and bringing in new packets to its end. This process continues until the last packet is brought in and processed.

Flows for different IP pairs vary a lot in the normal testing dataset. A set of observations, P's, are generated for each flow. Some could include thousands of packets, generating lots of more window frames than others. In detection, we would need to flag a flow by setting a rule on the set of RE values calculated of every P and the profile Q.

IV. Experimenation

We obtain real normal traffic from the MAWI dataset [12]. We calculate the TCP flag frequencies of five million network packets each in training and testing respectively. Four sets of CSC messages are generated according to keywords or uniform distributions as well as using all 64 TCP flags or the 9 TCP flags seen in the normal traffic datasets.

We discarded traffic flows that contain fewer than 100 packets in order to be able to compare the results of using three window sizes of 10, 50, and 100 packets. This also helps to remove outlier flows that are too short from skewing the experiment. This results in 1707 flows of 4,389,829 packets in total.

Each of the four CSC datasets has 40 CSC messages. Since there is only one IP pair in our CSC traffic, we treated every message one separate CSC instance. In testing, each CSC message is one sample that would be either flagged or not for evaluating the accuracy and efficiency of our approach.

We choose different thresholds by going over the range of RE values generated. Then we can examine the detection performance to pick the threshold for the best tradeoff of detection rate and false alarm rate. For per-flow analysis, if the RE value is greater than the chosen threshold, we flag the entire flow as CSC.

For per packet window analysis, a flow, same for a CSC message, is flagged if any window frame generates a RE value greater than the threshold. Therefore, the per packet window detection method is more sensitive than the per flow method. In this analysis, not only could users analyze the IP pairs raising alarms, they could also identify the first window frame that triggers an alarm. This can tell how quickly and efficiently a CSC message is detected.

V. Results

We measure the accuracy using Receiver Operating Characteristic (ROC) curves to show the detection rate of CSC messages versus the false positive rate when characterizing normal traffic as CSC. Moreover, we analyze the efficiency of our approach by looking at how quickly alerts are triggered for CSC messages.

A. Accuracy

Fig. 6 plots the ROC curves for the per-flow calculation. It can achieve a 100% true positive rate and no false alarms for CSC generation using all the 64 TCP flags. But, the accuracy decreases drastically if the CSC generation uses only the 9 flags seen in normal traffic. Specifically, it gets a very high false alarm rate of 89% for CSC messages to transmit the keywords. It does not fare well for CSC messages following uniform distribution either, with the "best" performance of only 49.5% true positive rate and 5% false positive rate.

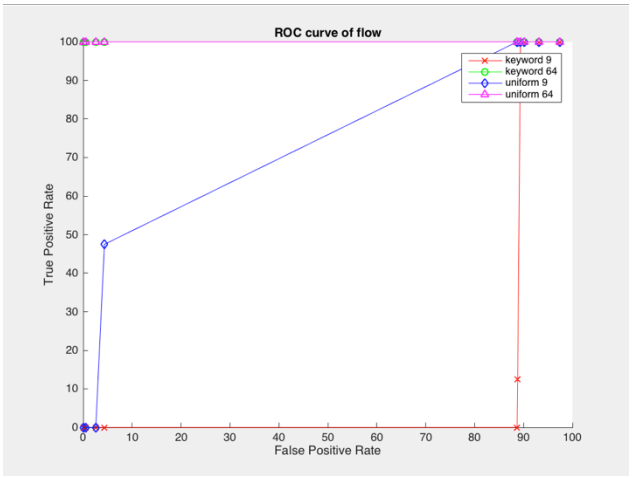

Fig. 6. ROC curves of per-flow analysis.

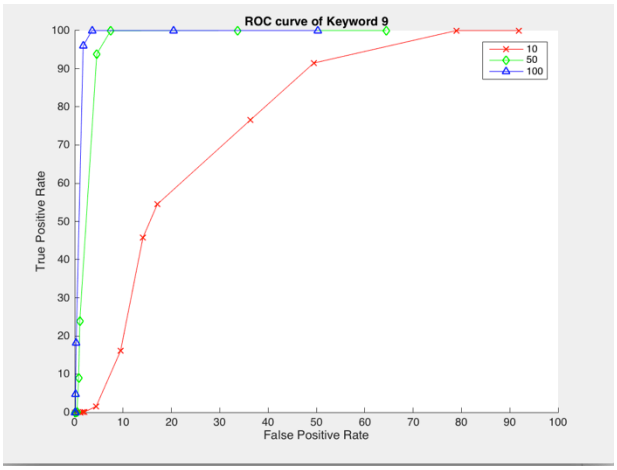

Fig. 7. ROC curves of packet window analysis with different window sizes using keywords and 9 TCP flags.

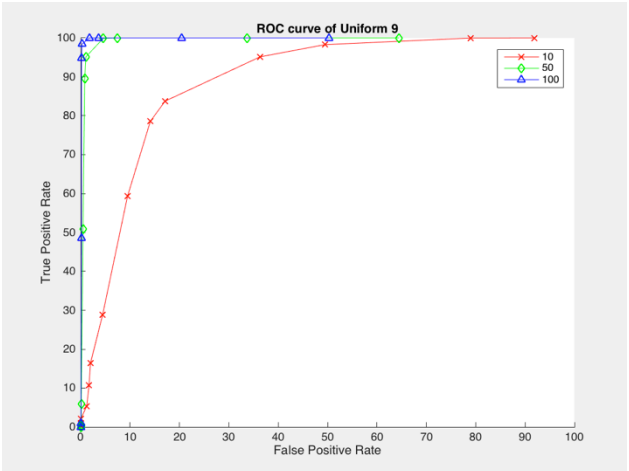

Fig. 8. ROC curves of packet window analysis with different window sizes using uniform distribution and 9 Flags.

In comparison, as shown in Fig. 7, the packet window detection method is much more promising. Using a moving window of 100 packets and replacing 10% of packets in each frame is the most accurate. Using a moving window frame of 10 packets per frame achieves 91% true positive rate. However, this threshold is too sensitive to result in a 50% false positive rate, triggering false alarms often. A moving window size of 50 packets improves the accuracy to close to that of the window size of 100 packets.

Fig. 8 shows the results for uniform distribution in CSC message generation resulted from encryption. We note the overall improvement in detection accuracy. Specifically, a moving window size of 100 packets results in close to 100% true positive rate with close to 0% false positive rate.

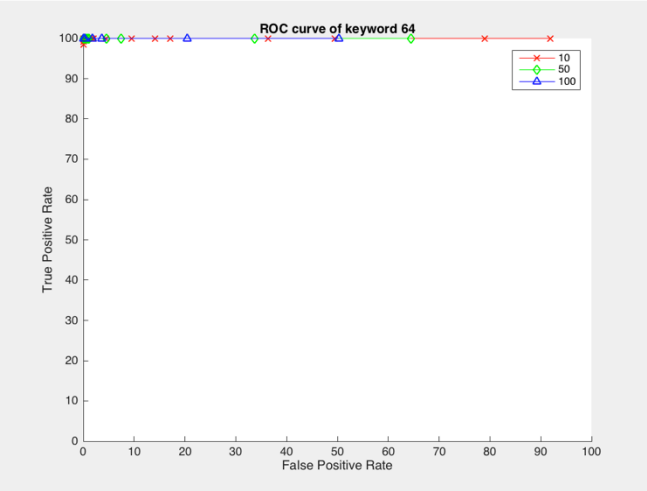

Fig. 9. ROC curves of packet window analysis with different window sizes using keywords and 64 TCP flags.

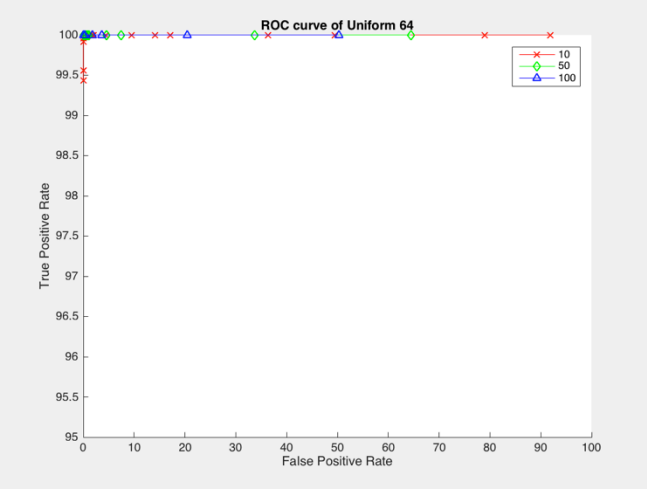

Fig. 10. ROC curves of packet window analysis with different window sizes using uniform distribution and 64 Flags.

At last, we show the performance for CSC generation using all the 64 TCP flags in Fig. 9 and Fig. 10. In contrast, detecting CSC traffic in datasets using all 64 possible TCP flags is a lot easier. We can achieve 100% true positive rates with 0% false positive rate for both scenarios of using the keywords and the uniform distribution.

In conclusion, the per-flow calculation did not perform as well as the packet window calculation. For packet window calculation, the two different methods to generate CSC traffic

using keywords and uniform distribution resulted in different detection performance. A moving window of 50 packets or higher can perfectly distinguish CSC messages if the 64 TCP flags are used from normal traffic. Detection performance worsened for a smarter accomplice using only the 9 common TCP flags seen in normal traffic.

Moreover, detection is easier when uniformly distributed TCP flags were used to emulate message encryption, even with only 9 TCP flags. This presents CSC accomplices with an interesting dilemma – they can use encryption to protect the confidentiality of a message if it is detected, but it does not help with the stealth of communication. In fact, encrypting a secret message increases the chance to reveal the existence of such as CSC.

B. Efficiency

We also characterized the efficiency, i.e., how fast the detection using a moving packet window raises an alarm for a CSC message. For comparison, we first find the best control thresholds that can differentiate normal and CSC traffic. The detection delay for a CSC message detected is given as the number of window frames before the first alarm is triggered using these thresholds.

TABLE I. EFFICIENCY FOR DETECTING CSC MESSAGES USING 9 TCP FLAGS AND KEYWORDS.

Window Size	Threshold	Average	Standard Deviation	Detected	False Positive Rate
50	0.8	1.2479	1.1168	40	4.57%
50	1.1	16.82	17.3233	38	1.05%
100	0.8	1.25	1.1347	40	1.76%
100	1.1	12.56	9.1655	25	0.29%

TABLE II. EFFICIENCY FOR DETECTING CSC MESSAGES USING 9 TCP FLAGS AND UNIFORM DISTRIBUTION.

Window Size	Threshold	Average	Standard Deviation	Detected	False Positive Rate
50	1.1	1	0	40	1.05%
50	1.2	1.1499	1.1499	40	0.82%
100	1.1	1.0249	0.156125	40	0.29%
100	1.2	1.325	0.7870	40	0.18%

Table I and Table II show the efficiency for the chosen window sizes and thresholds that result in the best accuracy performance. They give the average delay, its standard deviation, and the number of messages detected in each setting. They only show the more challenging scenarios of generating CSC messages using 9 TCP flags. We can see clearly that mostly the alarm is raised for the first window frame in every setting. This is very desirable performance.

VI. CONCLUSION AND FUTURE STUDY

In this study, we simulated differential hacker behaviors in coding CSC messages and compared per-flow and packet window detection strategies. Our results have shown that the detection approach with relative entropy and data processing methods we employed are promising. At the same time, the CSC message coding, using TCP flags and encryption, and parameters in analysis such as the packet window size, affect the detection performance.

We have gained interesting insights into the problem of covert channel analysis. It is easier to detect CSC when the TCP flags are uniformly distributed, i.e., when accomplices attempt to encrypt their messages. This, however, presents them a challenge: it is more likely to transmit messages without being detected if they do not encrypt those. But once the messages are detected they are plaintext for people to read. Of course, even more sophisticated accomplices may adopt an encryption scheme that does not result in uniform distribution of these TCP flags. We may consider such scenarios in our ongoing research effort of analyzing more advanced covert communication channels.

This study has a few limitations that we plan to address in the future. This study used only one normal profile. For improvement, we are studying finer granularities by using multiple normal profiles. Moreover, we performed offline analysis of real normal data and simulated CSC traffic. We are working on real time analysis to be implemented in a networking testbed.

ACKNOWLEDGMENT

We like to thank Liangjia Fu from the Johns Hopkins University for his contribution to the CSC traffic data generation.

REFERENCES

[1] R. Trimble, W. Oblitey, S. Ezekiel, and J. Wolfe, Covert Storage Channels: A Brief Overview. Indiana University of Pennsylvania.

[2] S. Cabuk, C. Brodley, and C. Shields, “IP covert channel detection,” ACM Transaction on Information and System Security (TISSEC), 12(4), 2009.

[3] E. Pennington, W. Oblitey, S. Ezekiel, and J. Wolfe, An Overview of Covert Channels. Indiana University of Pennsylvania.

[4] S. Gianvecchio and H. Wang, “Detecting covert timing channels: An entropy-based approach,” ACM Conference on Computer and Communications Security.

[5] H. Okhravi, S. Bak, and S. King, "Design, implementation and evaluation of covert channel attacks," 2010 IEEE International Conference on Technologies for Homeland Security (HST), 2010.

[6] H. Chaturvedi and M. Hemmel, A Novel Timing-Based Network Covert Channel. John Hopkins University, 2015.

[7] A. Houmansadr and N. Borisov, "CoCo: Coding-based covert timing channels for network flows," Lecture Notes in Computer Science Springer, vol. 6958, pp. 314-328

[8] Department of Defense, Trusted Computer System Evaluation Criteria. 1985.

[9] K. Ahsan, Covert Channel Analysis and Data Hiding in TCP/IP. University of Toronto. 2002.

[10] Y. Hu, X. Li, and X. Mountrouidou, “Improving covert storage channel analysis with SDN and experimentation on GENI,” National Cyber Summit’16, Huntsville, Alabama, June 7-9, 2016.

[11] J. Shlens, Notes on Kullback-Leibler Divergence and Likelihood Theory. Systems Neurobiology Laboratory, 2007

[12] MAWI Working Group Traffic Archive. [Online]. Available: http://mawi.wide.ad.jp/mawi.