MIMIC: AN ACTIVE COVERT CHANNEL THAT EVADES REGULARITY-BASED DETECTION

by

KUSH KOTHARI

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT ARLINGTON

May 2010

To my parents without whom I would not be where I am today and my brother Luv for his unconditional support.

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Matthew Wright, his perpetual energy and enthusiasm in research have motivated me to reach my goals. Without his guidance and persistent help this thesis would not have been possible. I am grateful to Dr. Gautam Das and Dr. Nikola Stojanovic for their invaluable advice and interest in my research and for taking time to serve on my thesis committee.

I want to express my special thanks to: Titus Abraham and Pranav Krishnamoorthy. Our whiteboard duels made complex problems really simple and I attribute a lot of my success to them. I would like to thank Robert Walls, he was always there to guide me like an elder brother whenever I was struggling with concepts and implementation. I would like to thank Arnab Biswas for helpful discussions and for taking the time to critically evaluate this manuscript. I want to thank all the members of the Information Security Lab for providing an excellent and inspiring working atmosphere.

Finally, I would like to express my deep gratitude to my brother who has encouraged and inspired me, I am also grateful to my parents for their encouragement and unconditional support.

April 16, 2010

ABSTRACT

MIMIC: AN ACTIVE COVERT CHANNEL THAT EVADES
REGULARITY-BASED DETECTION

Kush Kothari, M.S.

The University of Texas at Arlington, 2010

Supervising Professor: Matthew Wright

A covert timing channel is a hidden communication channel based on network timing that an attacker can use to sneak secrets out of a secure system. Active covert channels, in which the attacker uses a program to automatically generate innocuous traffic to use as a medium for embedding the covert channel, are especially problematic, as they allow the attacker to output large amounts of secret data. A promising technique for detecting covert timing channels focuses on using entropy-based tests. This technique can reliably detect known covert timing channels by using a combination of entropy (EN) and conditional entropy (CE) to detect anomalies in shape and regularity, respectively. The CE test is particularly effective at detecting regularity in active covert channels.

In this work, we show that these detection techniques can be defeated by an active covert channel that generates traffic in a purposefully irregular manner. In particular, we propose Mimic, an active covert channel that mimics both the shape and regularity of legitimate traffic to disguise its presence. Mimic includes two modules, a shape modeler and a regularity modeler, for learning about the statistical properties

of real traffic and generating traffic with the same properties. The main novelty of Mimic stems from its ability to smooth out the shape of the distribution while maintaining the regularity patterns of legitimate traffic. To measure the effectiveness of our mechanism, we ran experiments for both detection and throughput over a LAN and over the Internet. Our results show that Mimic is undetectable by any known detection technique at without loss of throughput.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

During the past few decades, traditional human-intensive tasks and services have been transformed into their digital equivalents. The growth and convergence of technology, computers, and networks have enabled organizations and businesses to develop information-centric processes. With the growing reliance on information and information technology, maintaining the confidentiality of sensitive data stored in computers is paramount to the success and failure of projects [17].

To counter the threat of leak of sensitive and mission-critical information, secure facilities such as NASA and the CIA implement cyber security mechanisms. This includes the monitoring of all incoming and outgoing traffic, high-grade encryption for all data communication, intrusion detection systems, and rigid enforcement of workstation policies. These measures often make it impossible to leak information by traditional means such as electronic mail or file transfer etc.

A covert channel is a communication channel that can be exploited by a process to transfer information in a manner that violates a system's security policy [4]. An adversary with vested interests can utilize covert channels to leak out information from a secure facility. Ample research is required in understanding and exploring the working of covert channels to develop suitable defenses against them.

Covert timing channels (CTC) are a class of advanced covert channels that use the timing difference between two consecutive packets — inter-packet delays (IPDs) — to encode messages. IPDs are a natural characteristic of network traffic, and as

a result, messages encoded with IPDs make CTCs particularly difficult to detect. Based on the source of the network connection that is used CTCs can be classified as

- *Passive*: CTCs that use an existing connection established by the user to transfer covert data. Their capacity to transfer is limited by the throughput of the base connection; and
- *Active*: CTCs that spawn a separate connection to transfer covert data. They are capable of achieving significantly higher throughput as compared to passive CTC.

Defenses against CTC can be classified as *Prevention*-based and *Detection*-based. The primary goal of prevention-based defenses is to eliminate the possibility of CTCs or make it impractical to establish a CTC. The working of CTCs depends on timing information; therefore, prevention-based defenses erode these properties of a channel by distorting the timing of traffic streams. However, this adversely affects legitimate user traffic and is infeasible for use in most scenarios.

Existing CTC designs introduce anomalies in the statistical properties of network traffic. Typically, passive CTCs add delays in the IPDs, causing the resulting distribution of IPDs be different from that of legitimate traffic. The shape of the IPD distribution of most passive CTCs can be used to detect them. In case of active CTCs, IPDs used to generate covert channel are chosen randomly and do not follow recurrence of patterns observable in legitimate traffic. The regularity of the resulting IPD distribution can be used to detect these covert channels.

Gianvecchio et al. [6] propose the use of entropy to measure the shape and regularity of a network stream and differentiate legitimate traffic from covert traffic. This defense is quite effective and able to detect most existing CTCs. However, a skilled attacker can use more advanced techniques to evade this detection.

1.1 Contributions

In this thesis, we propose *Mimic*, an advanced covert timing channel. Mimic is an active CTC that is able to evade all the known detection techniques at without loss of throughput. First, we analyze how CCE works and we use this understanding to design a mechanism to evade CCE detection. In particular, we have designed a *Regularity Tree* module that provides a way to control the regularity in a generated covert channel and a Shaper module to control the shape of the covert IPD distribution.

We describe the design of a *modeler* component that combines a Shape modeler and a Regularity modeler. This enables us to mimic the shape and regularity of the legitimate distribution. Using a combination of four components — a filter, a modeler, an encoder, and a transmitter, Mimic can extract the statistical properties of any legitimate traffic and generate covert traffic matching those properties. To evaluate the effectiveness of Mimic, we executed several experiments for different geographic locations of sender and receiver. Our experimental results show that Mimic is able to evade all known detection techniques.

The rest of this report is organized as follows. In Chapter 2, we describe existing covert channels and defenses against them. In Chapter 3, we analyze the detailed operations of CCE in detecting currently known CTCs. We further suggest a method to evade regularity-based detection by turing the CCE approach into a method to generate traffic with less regularity. This approach can be adapted for use by any covert channel. Furthermore, we describe how to build the Regularity Tree module and apply it in a new CTC called *i*TRCTC. Finally, we discuss the shortcomings of *i*TRCTC and propose a design framework for Mimic. In Chapter 4, we describe our experimental setup and how we created datasets for our experiments. In Chapter 5, we evaluate the detection resistance of Mimic and compare Mimic with other existing

CTCs. Finally, Chapter 6 concludes the thesis and discusses possible direction for future work.

CHAPTER 2

BACKGROUND

In this chapter, we discuss different type of covert timing channels and different prevention and detection mechanism for covert channels. Then we describe exiting covert timing channels.

2.1 Covert Channel

A covert channel is a communication channel that can be exploited by a process to transfer information in a manner that violates a system's security policy [7].

There are two types of covert channels: covert *storage channels* and covert *timing channels*. In a covert storage channel, a *sending process* alters a particular data item (e.g., stored on hard disk, in memory, in packet headers, etc.) and a *receiving process* detects and interprets the value of the altered data to receive information covertly. In a covert timing channel, the sending process modulates the amount of time or order of events (e.g., disk accesses, memory accesses, packet arrivals, etc.) required for the receiving process to perform a task or detect a change in an attribute, and the receiving process interprets this delay or lack of delay as information [12].

A covert channel can be exploited in a stand-alone system or in a networked environment. In a stand-alone system the hidden message is transferred between entities on a single machine, whereas in a networked environment, information is transferred across the network. Network-based covert channels are different from encrypted channels, though they both provide secret communication. Network covert

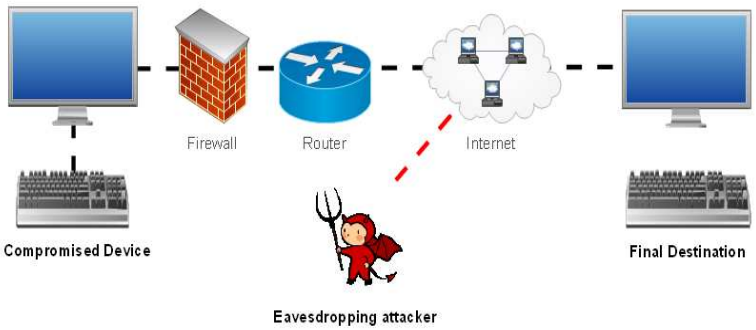


Figure 2.1. Passive CTC.

channels provide a suitable mechanism for remote attackers to steal confidential information without triggering network firewalls and intrusion detection systems [23].

This thesis focuses on covert timing channels used in a network environment. From this point forward, any reference to a covert timing channel (CTC), unless otherwise stated, will refer to timing channels that operate in a network environment.

2.1.1 Passive vs Active CTCs

There are two types of covert timing channels: passive covert timing channels (e.g., JitterBug [14], Liquid [19]) and active covert timing channels(e.g. TRCTC [1], MBCTC [7]).

Passive CTCs rely on existing traffic, in that they do not generate additional traffic to transmit the hidden message. As passive CTCs do not create any additional network connections, they are less prone to detection. However, their capacity to transfer the hidden message is limited to the bandwidth of the legitimate traffic. For example, if a legitimate stream sends 10 packets/second, passive CTCs at most can send 10 bits/second. Passive CTCs do not have to compromise the entire machine if creatively positioned (e.g., keystroke logger in JitterBug) as shown in Figure 2.1.

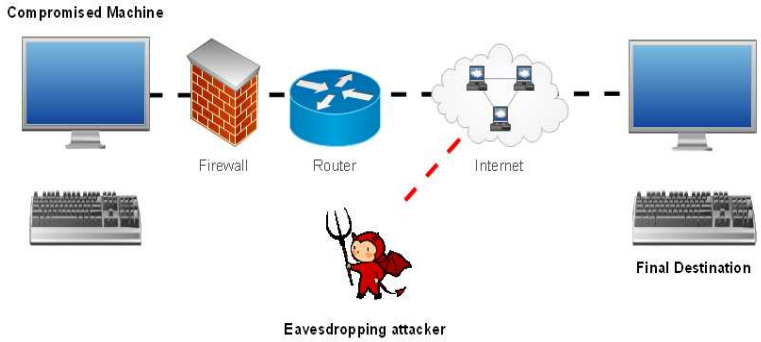


Figure 2.2. Active CTC.

In Active CTCs the attacker must have a compromised machine as shown in Figure 2.2. The attacker can send information from this compromised machine by creating his own connection. As the attacker has to create his own connection they are more prone to detection. However, their capacity is higher than passive CTCs as the attacker can control the rate at which the packets will be sent. 2.2.

Active CTCs are more potent threat as they can transfer data at much higher rate than passive CTCs (e.g. few megabits per second [22]).

2.1.2 Encoding in CTCs

In most CTCs, encoding of the hidden message is performed either by modifying the inter-packet delays (the time difference between the two consecutive packets) or by changing the order of inter-packet delays (IPDs). As encoding is performed only by modify the IPDs, it is effective regardless of the other characteristics of the packet (e.g., payload, size, type, and encryption). The sender in a CTC is a node or a computer that encodes the hidden message in legitimate traffic (passive CTC) or generated traffic (active CTC). The receiver in a CTC is defined as a node or a computer that decodes the hidden message. As IPDs are used to encode the message, receiver must not be the real destination, it is sufficient for the receiver to observe the

network stream that contains the hidden message. As covert channels are generally unidirectional communication, synchronization and error correction are more difficult. The receiver must know all the parameters required for decoding beforehand such as what network stream will be used for the CTC, what pseudo random number generator will be used and any encoding parameters.

2.2 Prevention against Covert Timing Channels

In prevention, the main goal is to either eliminate any possibility of CTCs or to make it impractical for CTCs to transfer any hidden message.

Kermmerer proposed an iterative method based on a Shared Resource Matrix that can be applied in all phases of the software life cycle to help identify the resources and entities that can be used to transfer messages from one process to another covertly [12, 13]. This method serves as a design tool as well as a security analysis tool for the entire software life cycle. Hu proposed a collection of techniques based on adding noise to the system clock, that can reduce the CTC capacity [9].

Kang et al. propose Pump, that reduces the CTC capacity in the Multilevel secure system (MLS) [10]. Pump intercepts the communication between two processes and re-sends the message by adding random noise in the acknowledgment stream to perturb the timing information. As simple Pump can accommodate only one sender and one receiver, later Kang et al. designed the Network Pump that can accommodate many senders and receivers as well as prevent the denial of service attack [11].

Fisk et al. propose the concept of Minimal Requisite Fidelity (MRF) in Active Warden [5]. MRF is a measure of signal fidelity required for an acceptable communication channel and prevents covert communication. Active warden is defined for structured carriers with well defined semantics(i.e. objects that have strict format definitions), such as UDP and ICMP. Similar to pump, Active warden intercepts the

communication between two processes and applies a set of rules to prevent transmission of covert information. For example, any unused field in a UDP packet can be used to send a covert message; setting unused fields to 0 prevents using these fields for a covert channel. For unstructured carriers (e.g., image), MRF is limited to human perception.

2.3 Detection of Covert Timing Channels

In detection, anomalies in statistical properties of network traffic are used to differentiate covert traffic from legitimate traffic. The most commonly used statistical properties are shape and regularity. The shape of a distribution refers to the shape of a probability distribution and regularity of a distribution is defined as the recurrence of specific patterns. For example two sample sets (1,0,1,0,1,0) and (1,1,1,0,0,0) have the same shape (e.g., mean, median and standard deviation). However, if we look at the recurrence of patterns the first set repeats a pattern (1,0) three times whereas the second sample repeats pattern (1) and pattern (0) three times each ,but does not repeat any pattern of length two.

If 100% effective, detection is always preferred over prevention as the system can identify the intrusion. These detection techniques are mostly successful, as most of the CTCs cause large deviations from legitimate traffic.

In general, passive CTCs are more prone to shape detection as they add delay in legitimate streams, unlike active CTCs that can make the encoded stream replicate the shape of legitimate traffic. As passive CTCs add delays on legitimate traffic they almost follow the same recurrence of patterns as in legitimate traffic and are not prone to regularity tests. On the other hand, since active CTCs generate traffic, they can easily maintain the shape of the distribution and are less prone to shape

detection test. However, they cannot maintain the recurrence of patterns and are easily detectable by regularity test.

We now describe specific tests, starting with different shape tests and then regularity tests, followed by a detection matrix for different CTCs and detection tests.

2.3.1 Shape Tests

2.3.1.1 Kolmogorov-Smirnov Test

The kolmogorov-Smirnov test (KS test) is the most useful and generic non-parametric test to compare whether two samples belongs to the same distribution. As the KS test is non-parametric, it does not rely on any assumptions about the distribution and therefore can be used for any distribution.

The KS test measures the maximum distance between the empirical distribution of two sets:

$$Distance = Max|F_1(i) - F_2(i)|$$

Gianvecchio et al. [6] use maximum distance to check if two samples (test sample and legitimate sample) belong to same distribution. If maximum distance is less than a threshold, the test sample belongs to a legitimate set. However, if it is beyond the threshold they belong to a different distribution and have a high probability of CTC presence.

2.3.1.2 Entropy based Detection

Entropy is a measure of uncertainty associated with a random variable. Entropy here refers to Shannon Entropy [15]. Shannon entropy (SE) is a measure of the average information content one is missing when one does not know the value of the random

variable. SE will be low if one can predict the outcome with certainty and SE will be high if one can not predict the outcome. For example, in the case of an unfair coin, SE is low, as one can predict the outcome by observing the coin flip frequency but in case of a fair coin, its hard to predict the outcome and therefore SE is high. Shannon entropy is defined as

$$SE = -\sum_{i=1}^{M} P(x_i) log_2 P(x_i). \tag{2.1}$$

Here $P(x_i)$ is the probability of the occurrence of the random variable x_i. SE will be high if the probability of occurrence of all the random variables in equation 2.1 is the same and it will be low if some of the random variables have higher occurrence then others.

Gianvecchio et al. propose a new detection mechanism based on Shannon entropy [6]. In a sample of IPDs, each IPD can be mapped to one of the M symbols where a symbol in the entropy test is defined as the histogram of IPDs. The range of these histograms are set using a large set of legitimate traffic. These histograms are called bins and each bin contains the same number of legitimate IPDs, resulting in each bin having equal probability.

To detect the presence of CTC, a sample of IPDs is mapped to the bins. The entropy H is calculated using the equation 2.1 where the probability $P(x_i)$ of each symbol/bin is defined as the percentage of IPDs in the sample mapped to a particular bin. Any sample set whose entropy is less than the legitimate set is considered as CTC.

Corrected Entropy Test

Entropy test uses $2^{16} = 65536$ bins while the size of sample set used for the test is relatively small (in most cases 2000). Thus, most of the bins are empty. To

solve this limited sample set problem, Gianvecchio et al. propose a new detection mechanism called corrected entropy (CEN). CEN is calculated as follows:

$$CEN = H + perc(X_1) * H, \tag{2.2}$$

where H is the entropy defined in equation 2.1 and $perc(X_1)$ is the percentage of bins containing exactly one IPD from the test sample.

2.3.2 Regularity Tests

2.3.2.1 Conditional Entropy Test

Entropy test as defined in Section 2.3.1.2, only measure the difference in shape of the distribution regardless of the order of IPDs. To overcome this shortcoming, Gianvecchio et al. propose a detection technique based on conditional entropy (CE), to differentiate the regularity of covert channels from legitimate traffic.

Given a sequence of symbols (bin numbers) X, such that X_i is the i^{th} value in the sequence, the conditional entropy of symbol X_i, given $X_1..X_{i-1}$ is

$$H(X_i|X_1..X_{i-1}) = H(X_1..X_i) - H(X_1..X_{i-1}).$$

In conditional entropy, a tree of height h is created. The level of the tree represents the length of the patterns. For example, the children of the root represent patterns of length one and the leaf nodes represent patterns of length h. The height of the tree is fifty and every node has five children representing the five histograms/bins used for CCE, these values are empirically found to be best. Every node has a count which indicates the number of times that node has been visited. This count also indicates the number of times that pattern has occurred.

Given a sample set of consecutive IPDs, create a set B, of corresponding CCE bins. This set B is divided into small windows of size fifty by shifting the window

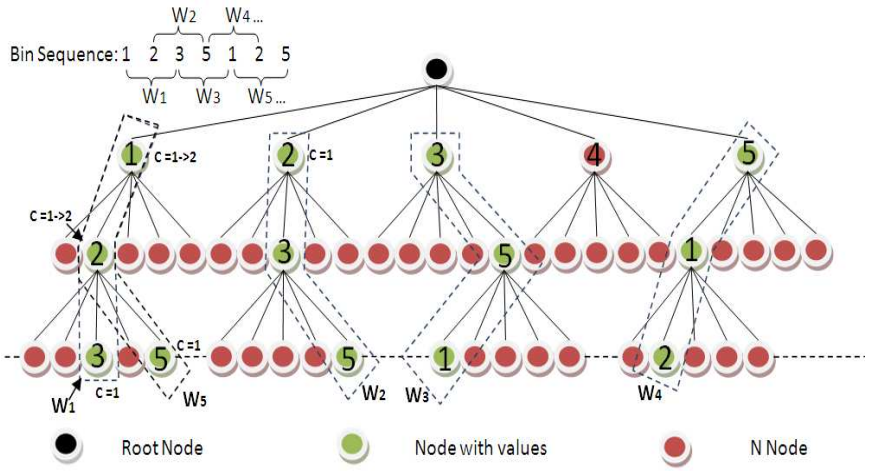


Figure 2.3. Example of CCE mechanism.

by one. For every window, traverse the tree starting with the first bin. If the node already exists in tree (i.e. the pattern exists in previous windows), increase the count by one, otherwise create a new node with count one, indicating the first occurrence of the pattern. After traversing the tree for the sample set, entropy values are calculated for every level of the tree and the minimum value of entropy among all the levels is referred to as the conditional entropy for that set.

For example, let us assume a tree of height three and a bin sequence {1, 2, 3, 5, 1, 2, 5}. Figure 2.3 shows how to create the CCE tree. First, divide the bin sequence in to windows W_i of size three: $\{\{W_1 = \{1, 2, 3\}\}, \{W_2 = \{2, 3, 5\}\}, \{W_3 = \{3, 5, 1\}\}...\}$, and then traverse the tree by updating the count.

Corrected Conditional Entropy Test

To overcome the problem with limited datasets, Gianvecchio et al. proposed the use of corrected conditional entropy (CCE). CCE is calculated as following:

$$CCE(X_i|X_{i-1}) = H(X_i|X_{i-1}) + perc(X_i) * H, \tag{2.3}$$

where $perc(X_i)$ is the percentage of unique subsequences of symbols of length i.

2.3.2.2 Regularity Test

Cabuk et al. propose a detection mechanism based on the variance of IPDs [2]. This method is based on the fact that for legitimate traffic, variance of continuous IPDs changes over time whereas in CTC it remains relatively constant. This method measures the regularity in the network stream and therefore referred as Regularity Test.

To measure the regularity, the network stream is divided into non-overlapping windows of size M IPDs. Then, for each window, standard deviation σ_i is calculated. The regularity is defined as the standard deviation of the pairwise differences between each σ_i and σ_j for all sets i <j.

$$Regularity = STDEV\left(\frac{|\sigma_i - \sigma_j|}{\sigma_i}, \quad i < j, \quad \forall i, j \right)$$

2.4 Examples of Covert Timing Channels

In this section, we describe several covert timing channels, especially active covert timing channels.

2.4.1 JitterBug

Shah et al. propose a passive CTC using a hardware device called JitterBug [14]. This device exploits a network-based timing channel to transmit the hidden message and is designed to transmit passwords and secret information over interactive network applications (e.g., SSH, X-windows). The JitterBug device is a keylogger and resides between the keyboard and the CPU.In interactive network applications, every keystroke generates a packet [16] and by modifying the keystroke timing carefully, JitterBug is able to encode the message. With JitterBug the attacker does not have to compromise the system, as encoding is done by modulating the keystroke timing.

JitterBug uses a timing window w to determine the additional delay required to encode a message bit. The timing window w should be large enough to avoid errors induced by network jitter. A binary sequence will be encoded by delaying the keystroke such that the resulting IPD satisfies the following equation:

$$IPD_i \bmod w = \begin{cases} 0 \quad \pm\frac{w}{4} & \text{for 0 bit} \\ \frac{w}{2} \quad \pm\frac{w}{4} & \text{for 1 bit} \end{cases}$$

Shah et al. determined that this encoding will generate undesired regularity in the IPDs and therefore proposed the use of pseudo-random sequence s_i to rotate the timing window w and remove most of the regularity. Encoding for the binary sequence will be done using the following equation:

$$(IPD_i - s_i) \bmod w = \begin{cases} 0 \quad \pm\frac{w}{4} & \text{for 0 bit} \\ \frac{w}{2} \quad \pm\frac{w}{4} & \text{for 1 bit} \end{cases}$$

2.4.2 Liquid

Walls et al. propose a passive CTC, that uses a portion of the compromised stream to smooth out the distortion detected by shape detection tests (e.g., Entropy Based Detection [6]). Liquid uses a combination of transmitting and shaping IPDs. IPDs used for encoding the hidden message are called transmitting IPDs and IPDs used to maintain the shape of distribution are called shaping IPDs [19].

Liquid uses half of the IPDs to encode the hidden message using JitterBug encoding and the other half of the IPDs to increase the probability of the symbols not used by transmitting IPDs in the entropy test. In the entropy test, each IPD is mapped to one of the M symbols, which are basically histograms of IPDs. Liquid keeps track of the mapped symbols during message encoding and tries to generate other symbols which have not yet been generated or generated very few times. This ensures that the probability of any symbol in the entropy test is almost equal, thus

increasing the entropy value. The shaping IPDs are modified to minimize the following cost function:

$$Penalty_{ipd} = CountOf(BinOf(IPD_{shaping})).$$

Furthermore, Walls et al. provided an estimation framework to determine the trade off between amount of shaping required and the desirable detection resistance. To achieve high detection resistance, the attacker must reduce the throughput of the CTC.

2.4.3 Model Based Covert Timing Channel

Gianvecchio et al. [7] propose a framework to create an active CTC, the Model based covert timing channel (MBCTC), by mimicking the statistical properties of legitimate traffic. MBCTC is composed of four components: a filter, an analyzer, an encoder and a transmitter.

MBCTC is designed with a goal, that it can use any type of legitimate traffic to create a covert channel. The filter and the analyzer of MBCTC provides a way to achieve this goal.

The filter observes all out going traffic and separates a specific type of traffic to be mimicked. The more precise a filter is, the better the statistical model can be formed. For example, a model for SMTP or HTTP is preferred over a generalized model for TCP/IP traffic. The filtered stream will be passed to analyzer which uses the root mean squared error (RMSE) to find the best distribution model (e.g., exponential, gamma or poisson) and the maximum likelihood estimation (MLE) to determine the parameters for each distribution model. The model with minimum RMSE value is chosen to mimic network traffic. This model will be updated automatically after every 100 packets to reflect any changes in traffic characteristics.

Encoding is performed by using the inverse distribution function of the selected model and decoding is performed by using the cumulative distribution function. The shape of the MBCTC traffic is almost same as that of legitimate traffic and undetectable by any known shape test. However, as there is no correlation between consecutive IPDs, MBCTC lacks the recurrence of repeated sequences and therefore, highly regular.

2.4.4 IP Simple Covert Channel

Cabuk et al. proposed the IP Simple Covert Channel (IPSCC) that can be a storage covert channel or timing covert channel according to the encoding used for the hidden message [3].

In storage IPSCC, sender and receiver decide on a time interval t according to which a packet will be set to encode bit '1' or no packet will be sent to encode bit '0'. The receiver can decode the binary sequence by observing the packets in windows of time interval t.

In timing IPSCC, sender and receiver decide on a sequence s_i of timing intervals and corresponding bit (e.g., (IPD_1,"1"),(IPD_2 ,"0")). legitimate IPDs are then modified according to the sequence s_i to encode hidden message.

The timing intervals in both storage and timing IPSCC must be chosen in a way that balances the bit error while maintaining a reasonable channel capacity. If the timing window is small, message bits may be flipped because of the network jitter. If it too large the channel capacity will decrease greatly and eventually make the covert channel impractical. As only a couple of IPDs are used to encode the message, it is easy to detect IPSCC.

Table 2.1. Detection Matrix and % use of throughput

CTC	Shape Test (CEN)	Regularity Test (CCE)	Detectable	% use of throughput
Passive CTC				
JitterBug	Yes	No	**Yes**	100%
Liquid	No	No	**No**	50%
Active CTC				
TRCTC	No	Yes	**Yes**	100%
MBCTC	No	Yes	**Yes**	100%

2.4.5 Time Reply Covert Timing Channel

Cabuk et al. [1] propose the Time Reply CTC (TRCTC). TRCTC replays a set S of legitimate IPDs to send the hidden message. Set S is divided into two equal subsets S_1 and S_2. IPDs are randomly chosen from subset S_1 to encode '0' and from subset S_2 for bit '1'.

As set S is made up of legitimate traffic, the distribution of TRCTC traffic is approximately equal to the distribution of legitimate traffic. However, as there is no correlation between consecutive IPDs, TRCTC lack the recurrences of repeated sequences and therefore, highly regular.

2.4.6 Watermarking and Encrypted Channels

The timing-based watermarking of packet streams has been used to correlate incoming and outgoing streams [20, 21, 8]. Watermarking is different from covert timing channels as their goal is correlate two network stream. Watermarking is used to trace connections in stepping stone attacks [21, 18]. On the other hand, encrypted channels prevent unauthorized parties from reading or extracting useful information from the communication, whereas network covert channels aim to hide the existence of the communication.

Table 2.1 shows performance of different CTCs against shape (corrected entropy) and regularity (corrected conditional entropy) based detection mechanisms. Liquid is the only CTC that is able to evade both the detection mechanisms. However, Liquid is only able to use 50% of throughput.

CHAPTER 3

MIMIC

In this section we describe two new active CTCs, iTRCTC and Mimic, and their design and implementation.

3.1 CCE in details

To explain how CCE works, we explore the detection mechanism of CCE for different CTCs. CCE detection is not effective against all CTCs. In general, CCE can easily detect active CTCs but is not very effective against many passive CTCs.

Figure 3.1 shows the CCE tree structure for legitimate traffic. The $\mathcal{N}$ nodes in this tree are defined as the nodes which do not exist (i.e. the pattern from root to node never occurred). Numbers on the right side of the tree represent the average number of $\mathcal{N}$ nodes on the corresponding level of the tree based on the 100 different samples of size 2000.

Figure 3.2 shows the first few levels of the tree used to detect different channels. Passive CTCs (e.g. JitterBug and Liquid) have almost the same number of $\mathcal{N}$ nodes as legitimate traffic. However, the number of $\mathcal{N}$ nodes in active CTCs (e.g. TRCTC and MBCTC) is much less than the number of $\mathcal{N}$ nodes in legitimate traffic. It is observed that for active CTCs, there are no $\mathcal{N}$ nodes in the top level of the tree. Fewer $\mathcal{N}$ nodes represents a lack of recurrence of patterns in the stream.

Figure 3.3 shows tree for a stream containing frequently-repeated sequences. A stream that has a high recurrence of the same patterns results in a tree with many $\mathcal{N}$ nodes. As CCE is sensitive to the number of $\mathcal{N}$ nodes in the tree, the higher the

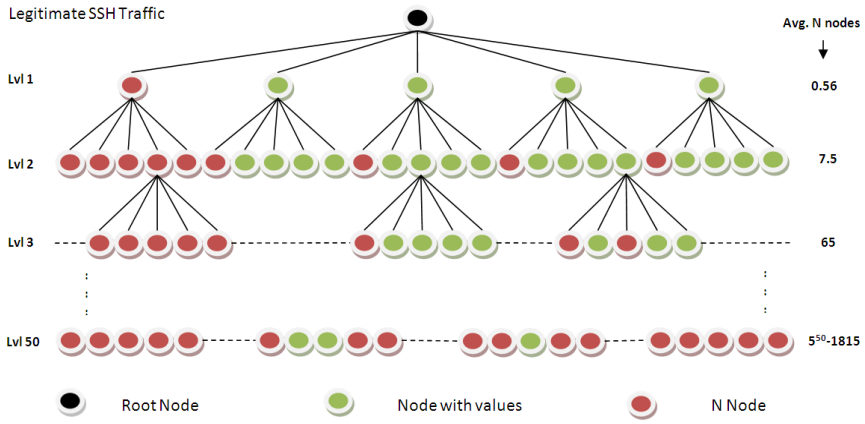


Figure 3.1. Tree structure for legitimate Traffic.

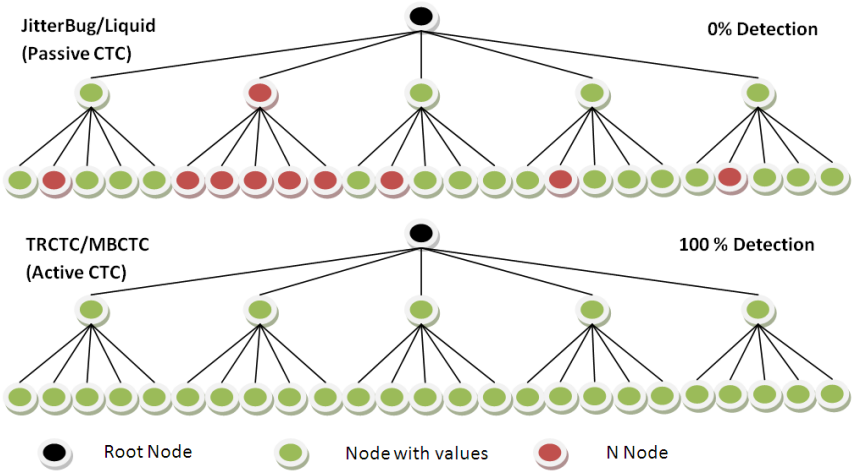


Figure 3.2. Tree structure for Passive and Active CTC.

number of $\mathcal{N}$ nodes, the lower the detection rate. Thus, the detection for the tree shown in Figure 3.3 is 0%. However, this stream is easily detectable by any shape test.

3.1.1 A naive method to evade CCE

A naive approach to evade CCE would be to replay the patterns of the CCE bins. These recurrences of the patterns will decrease the regularity in the CTC stream.

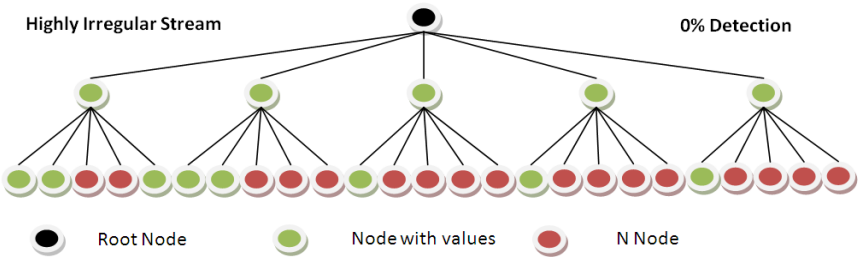


Figure 3.3. Tree structure for Highly Irregular stream.

As described in Section 2.3.2.1, five bins (1, 2, 3, 4 and 5) are used for CCE detection. To select a CCE bin sequence for repetition we can use the following methods

1. Select a CCE bin sequence that has already been sent. For example, if we want to replay a bin sequence of length four and we have sent IPDs using the sequence (1, 1, 1, 2, 3, 4, 2, 3, 5, 4, 5, 2, 3, 4), we can choose any subsequence of length four from the given sequence e.g. (1, 1, 1, 2).
2. Select a CCE bin and make a sequence of desired length by repeating the same bin. For example, in the previous case we can choose (x, x, x, x) where x $\in$ (1, 2, 3, 4, 5).

After choosing the bin sequence, IPDs are replayed randomly from their corresponding bins. These repeated bin sequences will increase irregularity in the CTC stream and evade detection by CCE. However, it will also reduce the channel capacity of the covert channel.

3.2 Regularity Tree

As described in Section 3.1, the CCE detection rate is directly proportional to the number of $\mathcal{N}$ nodes in the CCE tree for a given stream. CCE detection can be evaded by increasing the repeated patterns in the stream (i.e. increasing the number

of $\mathcal{N}$ nodes). However, frequently-repeated patterns may change the shape of the distribution, resulting in the CTC being detected by a shape test.

To control the amount of regularity in CTCs, we propose a *Regularity Tree*, a mechanism based on the CCE tree structure to mimic the irregularity of legitimate traffic. to build the Regularity Tree, we calculate the average number of $\mathcal{N}$ nodes $\bar{N}_h$, for each level h of CCE trees for the legitimate stream. The tree structure is similar to tree structure used for CCE tree, in which every node has five children corresponding to five CCE bins. For creating this tree we used rule set to update the lower level of tree based on the upper level constraints. If a sequence is not allowed in the upper levels of the Regularity Tree, then that sequence should not be allowed in lower levels either. For example, if we decide that the sequence $1 \rightarrow 2$ is not allowed, that means $X \rightarrow 1 \rightarrow 2$ and $1 \rightarrow 2 \rightarrow X$ are also not allowed, where X $\in$ (1, 2, 3, 4, 5).

For the first level, the rule set is empty and we select $\bar{N}_1$ nodes randomly and marked them as $\mathcal{N}$ nodes. Now the nodes we marked as $\mathcal{N}$ nodes are not allowed anywhere in tree and we add them in rule set. For every level, we will first update the level based on the rule set. Specifically, we will mark all the nodes as $\mathcal{N}$ that are not allowed because of upper level. We then, randomly choose additional nodes in this level so that in total we have $\bar{N}_i$ $\mathcal{N}$ nodes. The paths from the root to these newly marked $\mathcal{N}$ nodes will be added to rule set as shown in Algorithm 1.

In some CTCs, where the order of the IPDs is used to encode the message (e.g. TRCTC), Step 1 has to be modified to accommodate the encoding mechanism. In this case, we have to make sure that we will be able to encode any given sequence of a message.

Algorithm 1 Design of Regularity Tree

Path(K) := [Label(1_{st} level ancestor), ... , Label(Immediate Parent), Label(K)]

$\bar{N}[] \leftarrow$ Average no. of $\mathcal{N}$ nodes in a legitimate tree.

$\mathcal{R} = \{\}$

for all i such that $1 \leq i \leq$ height of tree **do**

  for all $r \in \mathcal{R}$, $n \in$ nodes at level i **do**

    $count \leftarrow 0$

    if $r \bigcap Path(n) = r$ **then**

      Discard node n

      $count \leftarrow count + 1$

    end if

  end for

  $D \leftarrow$ Select at random $\bar{N}_i$ - *count* nodes from level i.

  $\mathcal{R} = \mathcal{R} \cup Path(n); n \in D$

  Mark n as $\mathcal{N}; n \in D$

end for

3.3 *i*TRCTC

As most of the active CTCs generate undesired regularity in CTC streams, they are easily detectable by regularity-based tests. We propose *i*TRCTC, a CTC that is undetectable by all currently known tests. The main goal of *i*TRCTC was to defeat the corrected conditional entropy test for regularity in CTCs. We choose TRCTC as a base CTC for its simplicity and because it successfully evades all detection methods except CCE.

As TRCTC encoding is based on the order of IPDs, we have to modify the Regularity Tree. TRCTC creates two subsets S_1 and S_2 to encode '0' and '1' respec-

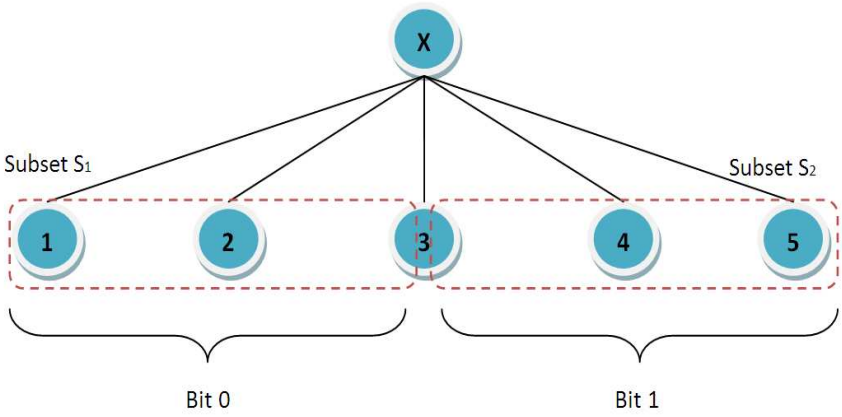


Figure 3.4. Subset S_1 and Subset S_2.

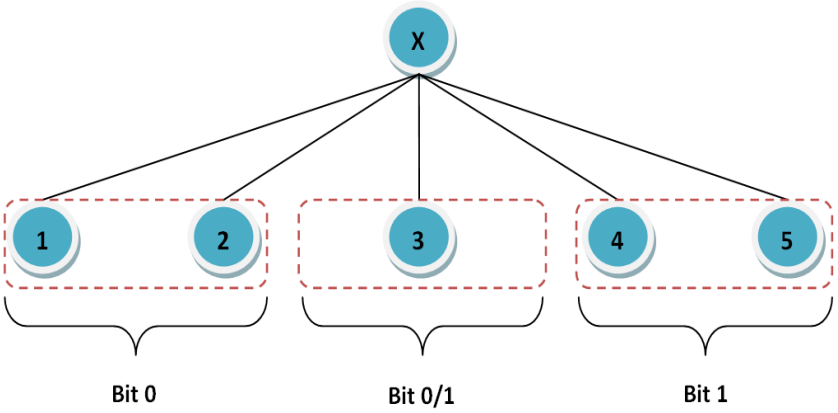


Figure 3.5. Grouping of bins.

tively. S_1 corresponds to CCE bin 1, 2 and the first half of 3 while S_2 corresponds to the second half of bin 3 as well as bin 4 and 5 as shown in Figure 3.4. To be able to encode any given sequence, we must at least have one non-$\mathcal{N}$ node in S_1 as well as in S_2. To achieve this, we create three groups (1, 2), (3), (4, 5), as shown in Figure 3.5. Any two of these groups are suppose to have one node which is not a $\mathcal{N}$ node.

*i*TRCTC replays IPDs based on the Regularity Tree. The Regularity Tree is updated after every 2000 IPDs to reflect any changes in the legitimate stream.

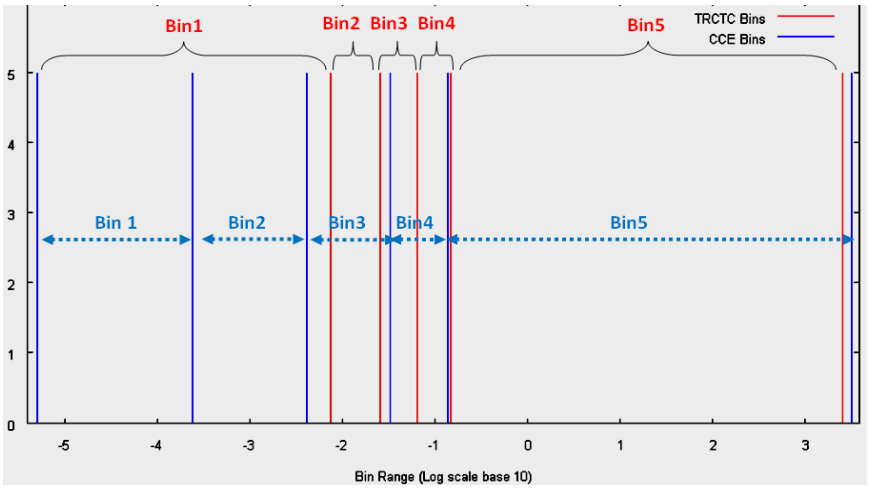


Figure 3.6. Bin Alignment.

Only CCE bin 3 (the middle node in tree) can be used for encoding as both '0 'and '1 '. The resulting stream in highly irregular. However, there is a high probability of changing the shape of the distribution.

One of the problem with this mechanism is the bin range alignment. As the training sets of IPDs used in designing the Regularity Tree and the CCE detection are different, it results in different ranges for the bins as shown in figure 3.6. For example, the bin 3 for the Regularity Tree is partly in the CCE bin 3 and partly in the CCE bin 4. So making Bin 3 as $\mathcal{N}$ node of the Regularity Tree does not make CCE bin 3 as $\mathcal{N}$, it will result in fewer number of $\mathcal{N}$ nodes at the detection side because of the different bin range. To overcome the problem of bin alignment, we have to reduce the channel capacity and use some portion of the compromised IPDs to increase the irregularity as we did in our naive method.

Another problem with this method is the encoding mechanism of TRCTC, as it depends on the way the IPDs are chosen (subset S_1 for '0'and subset S_2 for '1'), the number of $\mathcal{N}$ nodes cannot be increased arbitrarily as either encoding of any sequence is not possible or we will end up using IPDs from bin 3.

3.4 Mimic

*i*TRCTC is the first active CTC that can evade both shape and regularity tests while using 67% of throughput. Liquid and *i*TRCTC both evade all known detection mechanisms. However, *i*TRCTC is a 35% improvement over Liquid in terms of channel capacity.

To make a more robust and higher capacity channel, we propose a new mechanism called *Shaper* to control the shape of the covert channel. Based on Shaper and the Regularity Tree we propose a new active CTC called *Mimic* that transmit data without loss of throughput and, to the best of our knowledge, is undetectable by all known detection tests. Shaper is used to mimic the shape of the legitimate traffic, and Regularity Tree mimics the irregularity of legitimate traffic.

3.4.1 Shaper

If we have equi-probable histograms of IPDs, legitimate IPDs are uniformly distributed among these histograms. Shaper uses this knowledge to provide a way to control the shape of the covert traffic.

Shaper can be used to decide whether or not an encoded IPD belongs to the given distribution. Shaper keeps a track of the frequency of previously encoded IPDs belongs to all the shaping bins. For every new encoded IPD, Shaper checks the frequency of the corresponding bin. If the frequency is less than a threshold t the encoded IPD belongs to the distribution; otherwise it does not. The threshold t for the frequency can either be the same for all bins or can be individually determined for each bin using the legitimate traffic.

Shaper is different from the shaping used in Liquid. However, both are based on the histograms of IPDs. In Liquid, a portion of the IPDs are modified in such a way that they increase the frequencies of bins which are not used, whereas in Shaper

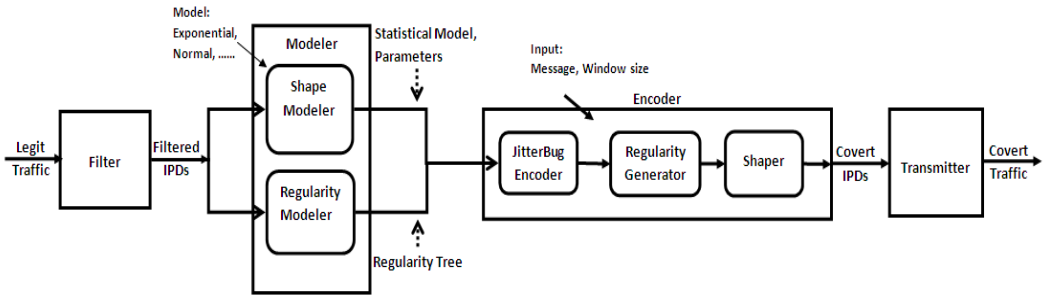


Figure 3.7. A Framework to design Mimic.

we choose the IPDs in such a way that they are uniformly distributed among the bins.

3.4.2 Design of Mimic

Mimic, as shown in figure has a pipeline structure similar to MBCTC, consisting of four components: a filter, a modeler, an encoder, and a transmitter. The filter is used to filter a specific type of network traffic (e.g. SSH, HTTP). Increasing the specificity of the filter improves the ability of the CTC to mimic legitimate traffic. Once the traffic is filtered, we calculate the IPDs for different sender and receiver pairs. These IPDs are then used by the modeler.

The modeler is divided into two components: a shape modeler and a regularity modeler. Our procedure for selecting the distribution is similar to the analyzer of MBCTC [7]. We begin by taking a sample of IPDs and fit them to a distribution — poisson, exponential, pareto, gamma, lognormal, and weibull distributions. Using maximum likelihood estimation (MLE), we determine parameters for distributions. The root mean squared error (RMSE) measures the difference between the model and the estimated distribution. We measure the RMSE values for each distribution model and the model with the least RMSE is chosen as the traffic model. Using this

distribution model and parameters we generate a large training set of IPDs that will be used in Shaper.

The regularity modeler works is parallel to the shape modeler. The main objective of the regularity modeler is to build the Regularity Tree for the filtered IPDs. The Regularity Tree is described in Section 3.2. The average number of $\mathcal{N}$ nodes is determined at every level, and we design the Regularity Tree by selecting these $\bar{N}_h$ nodes randomly for all nodes at each level . Mimic updates the distribution model, parameters for the model and Regularity Tree after every 2000 packets to reflect any changes in the legitimate traffic.

The filter and the modeler combined give us a framework that can model the shape and regularity of any given traffic. Using this framework, we can design an active CTC that can model any given legitimate traffic and use it for covert communication.

Encoding in Mimic is done by JitterBug encoding. This encoding is based on modular arithmetic, which is neither based on the order of IPDs (as in TRCTC) nor on pseudo random numbers (as in MBCTC). JitterBug do use pseudo random number for rotating window but not for encoding. As the encoding is independent of IPD selection, we can encode bit '0' or '1' using any IPD, which plays a major role in Mimic as described later.

To encode a hidden message, a large training set of legitimate traffic is generated using the statistical method provided by the Shape modeler. This training set is used by the Shaper to generate the bins. In parallel, the Regularity Tree generates the sequence of bins B_i. To generate the bin sequence, we traverse the tree using DFS and store all the paths P_i from root to leaf in set P. Then a path P_i is randomly chosen from the set P and added to the bin sequence.

Algorithm 2 Mimic Encoding

repeat

while $IPD_e \notin B_i$ **do**

Randomly select an $IPD : IPD \in B_i$, $B_i \in B$

$IPD_e = IPD+$ Delay added to encode message bit by JitterBug

end while

Use Shaper to decide whether $IPD_e \in$ current distribution

until $IPD_e \notin$ current distribution

We encode the message using Algorithm 2. We select a random IPD from the current bin B_i such that the binary bit encoded by JitterBug also satisfies the regularity of the stream. Shaper is used to determine if the IPD is a valid value of the current distribution. This process is repeated until a IPD is identified that satisfies both the regularity of the stream and belongs to the distribution. Empirically, 10% of the encoded IPDs do not meet this criteria. This is caused by a very small range of set of values in the regularity tree. In this case, we ignore the regularity constraint.

Mimic is undetectable by any known detection test as shown in Chapter 5. Mimic is able to generate covert traffic over any legitimate traffic without loss of throughput.

CHAPTER 4

EXPERIMENTAL SETUP

In this section, we describe the experimental setup used to validate the detection resistance of Mimic. First, we describe our creation of the training sets of IPDs for covert channels and entropy based detection. Then we describe how we use these training sets to design the covet streams and to perform detection.

4.1 Data Collection

All the IPDs used in our experiments are taken from network traces collected at UNC (University of North Carolina) at Chapel Hill and from previously collected traces for Liquid from UNC. We use SSH IPDs since SSH is the protocol used by both Liquid and JitterBug. A Perl script is used to extract the IPDs only for the SSH sections of the traces. Since convert channels use unidirectional communication, only the traces from client to server are considered and not vice versa. First we filter the traces that have the destination port as SSH using `tcpdump`. Next we make pairs of sender and receiver based on the unique combination of client and server. These sequences are concatenated in two disjoint sets. One set is used as the *SampleSet* for training the covert channel and another set is used as the *TrainingSet* to train the entropy test.

In a real attack scenario, it is more realistic to assume that both the *SampleSet* and the *TrainingSet* are not the same and that the *TrainingSet* is larger than the *SampleSet*. The size of each set is shown in Table 4.1.

Table 4.1. Experimental Set Sizes. Sizes of the *TrainingSet* and the *SampleSet*.

Type	Number of IPDs
TrainingSet	8,632,235
SampleSet	2,426,264

4.2 Creating the CTCs Traffic

In a SSH session, every keystroke results in packet. For our experiments, we created a software application in C that simulates the keystroke. The application takes a sequence of IPDs as input and based on these IPDs it will send a keystroke directly to the SSH session. A sequence of IPDs is called test sample. Each test sample is comprised of IPDs taken from the *SampleSet*. IPDs from the *SampleSet* are used as base input to directly compare the difference in performance between legitimate, JitterBug, Liquid, TRCTC, MBCTC, *i*TRCTC, and Mimic. To create a legitimate test sample, we choose sequential IPDs from the *SampleSet* and send them unchanged. For all the covert timing channels, the IPDs are first modified to embed the hidden message. We use multiple sets of the test sample to explore the effect of different scenarios on the detection scores. Each test set was created using 100 samples of 2000 IPDs.

JitterBug Traffic

To create JitterBug traffic, we used the same parameters used by Gianvecchio et al. for their study of entropy based detection [6]. The timing window w was set to 20ms and the length of the rotating sequence s was set to be equal to the sample size, i.e. 2000.

Liquid Traffic

To create Liquid traffic, parameters used for JitterBug were same as described above. We implemented Liquid with sub-millisecond transmit noise with values of N_s and N_t

Table 4.2. Parameters for different distribution models for SSH traffic

Model	Parameters	root mean squared error
Weibull	0.1279, 0.4401	0.3030
Gamma	0.2360, 3.7353	0.3170
Lognormal	-3.1641, 2.3639	0.3510
Pareto	1.1871, 0.0476	0.3349
Poisson	0.8815	0.2841
Exponential	0.8851	0.3144

rotated in the range [1..3] such that $\frac{N_t}{N_s} \approx 1$. Thus, the Liquid cycle would consist of a minimum of 2 or a maximum of 6 IPDs, with the number changing over time. The range for the shaping delay is between [0, w] where w is the window size for JitterBug and the incremental *step* was fixed to 0.1 ms as described in [19].

TRCTC Traffic

To create TRCTC traffic, we divide *SampleSet* in two subsets with range [0,L/2-5,000] and [L/2+5,000,L] where L is the size of *SampleSet*. The middle 10,000 IPDs are left for the threshold to distinguish between the sets, as well as to reduce bit error induced by the network jitter.

MBCTC Traffic

The filter in MBCTC is the same as data collection described in Section 4.1 and therefore not implemented. We assumed that we have filtered IPDs for the SSH application. Table 4.2 shows the parameters for the distribution models and their root mean squared error (RMSE) values. We choose the Weibull distribution with parameters to mimic legitimate SSH traffic. Encoding in MBCTC is performed as described in Section 2.4.3.

Mimic Traffic

To create Mimic traffic, we used the Weibull distribution with the same parameters as used for MBCTC, as described above. Filter mechanism is same as the data collection

and therefore, is not implemented in Mimic. We use the same filtered IPD as used for other CTCs. Based on the distribution parameters, we created a base set of IPDs; these IPDs are later used to encode the hidden message. To create the Regularity Tree, we pass the same distribution parameters (used to create the base set) to the Regularity modeler. In the Regularity modeler, these parameters are used to set the bin range for the Regularity Tree nodes. As the Shape modeler and the Regularity modeler work in parallel, we can either pass the parameter from the Shape modeler to the Regularity modeler or we can create them separately. For encoding, we use the same parameters for JitterBug as described earlier.

4.3 Detection

For evaluating the effectiveness of covert timing channels in realistic settings, we performed multiple experiments with different locations for *sender* and *receiver*. In the simplest experiment, both the sender and the receiver were located at iSec Lab at the University of Texas at Arlington (UTA). We created a receiving server using a dynamic DNS so that the sender and the receiver were four hops apart. This experimental setup is called *UTA-UTA* in Chapter 5. For more complex experiments, the receiver was located at the iSec Lab, UTA and the sender was located at the University of Massachusetts at Amherst (*UTA-UMass*) or at the Virginia Tech University (*UTA-VT*).

IPDs were captured on both sides using `tcpdump`. The IPDs captured at the sender side were only affected by the process of the sending machine itself. This serves as a worst case detection scenario for the covert channel since there is no network jitter to distort the intended IPDs. The second location was chosen to mimic a real-world situation in which the *receiver* may be located at any place across the globe.

We applied the corrected entropy and corrected conditional entropy tests on each captured sample of legitimate, JitterBug, Liquid, MBCTC, and Mimic IPDs. Each test was performed offline, after a complete sample of IPDs had been captured and parsed. The resulting test scores for the legitimate IPDs were used to create a cutoff score for each of the detection tests. To achieve a one percent false positive rate, the 1^{st} percentile was used for the corrected entropy test, while the 99^{th} percentile was used for the corrected conditional entropy. Any sample with a CEN value below the cutoff score would be considered covert. Similarly, any sample with a CCE value above the respective cutoff scores would also be considered covert. For each test, we calculated the detection rate for JitterBug, Liquid, MBCTC, and Mimic. This process was performed separately for each detection location.

CHAPTER 5

EXPERIMENTAL RESULTS

In this section, we describe the results for Mimic and present the comparison between Mimic and different CTCs. We refer to the IPDs used to transmit the message as transmit IPDs and the IPDs used to increase the irregularity are referred as irregular IPDs, unless otherwise specified.

5.1 A naive method to evade CCE

In naive method 1, IPDs are replayed from the previously sent bin sequence, whereas in naive method 2, IPDs are replayed from the same bin. Legitimate streams have many repeated IPDs from the same bin and therefore naive method 2 performs better then naive method 1. Table 5.1 shows the effect of different ratios of transmit to irregular IPDs on the CCE detection and the respective Channel capacity.

5.2 *i*TRCTC

As described in section 3.3, *i*TRCTC is not able to evade CCE detection because of the bin alignment problem as shown in Figure 3.6. If the attacker knows the

Table 5.1. A naive method to evade CCE: Detection and % use of throughput

CTC	Transmit IPDs : irregular IPDs	Detection	% use of throughput
Naive method 1	10 : 20	35%	33.3%
Naive method 1	10 : 30	1%	25%
Naive method 2	10 : 15	0%	40%

Table 5.2. *i*TRCTC: Detection and % use of throughput

CTC	Transmit IPDs : irregular IPDs	Detection	% use of throughput
*i*TRCTC	40 : 10	20%	80%
	30 : 10	13%	75%
	25 : 10	5%	71%
	20 : 10	0%	67%
	10 : 10	0%	50%

bin alignment used for detection, he can evade CCE detection by only creating a Regularity Tree of height three. In a real-world scenario, it is unrealistic to assume that the attacker has this information. For evading CCE detection, we used a portion of compromised IPDs to increase the irregularity. Table 5.2 shows the detection resistance and channel capacity for different ratios of transmit to irregular IPDs.

5.3 Mimic

In this section, we first explore the detection resistance of Mimic against the corrected entropy and corrected conditional entropy tests. We then compare Mimic with JitterBug, Liquid, TRCTC, and MBCTC.

5.3.1 Detection Resistance

As described in Chapter 4, we have outline our experiment for three different setups: UTA-UTA, UTA-VT and UTA-UMass. In each experimental setup, we ran 100 samples of legitimate and Mimic traffic with 2000 IPDs. We captured the IPDs at local as well as remote locations. The corrected entropy and corrected conditional entropy tests were performed for each sample at both locations. A 1% false positive rate for legitimate traffic is used as a cutoff score. Samples with a corrected entropy value less than or equal to the CEN cutoff score and corrected conditional entropy

Table 5.3. Cutoff score for CEN and CCE in UTA-UTA Experimental Setup

Test	Score	False Positive
Local		
CEN	≤ 11.48	1%
CCE	≥ 1.7	1%
Remote		
CEN	≤ 11.84	1%
CCE	≥ 1.85	1%

Table 5.4. Corrected Entropy Test Scores

Location	Average	Standard Deviation	False Positive	Average	Standard Deviation	Detection
Local						
	legitimate			Mimic		
UTA-UTA	18.24	2.19	1%	20.28	0.63	0%
UTA-VT	19.78	1.59	1%	20.40	0.59	0%
UTA-UMass	19.86	1.52	1%	20.42	0.66	0%
Remote						
	legitimate			Mimic		
UTA-UTA	18.25	2.20	1%	20.35	0.61	0%
UTA-VT	18.93	3.06	1%	20.51	0.62	0%
UTA-UMass	20.1	0.92	1%	20.51	0.56	0%

greater than or equal to the CCE cutoff score are considered covert traffic. Table 5.3 shows the cutoff scores for local and remote locations in the UTA-UTA experimental setup.

Table 5.4 shows the CEN test scores for the legitimate and Mimic traffic. Average Mimic test scores in both the local and remote locations in all three experimental setups are higher than the corresponding legitimate test scores (i.e. Mimic IPDs are more uniformly distributed among the histograms). Mimic is able to evade the CEN test in all the experiments.

Table 5.5. Corrected Conditional Entropy Test Scores

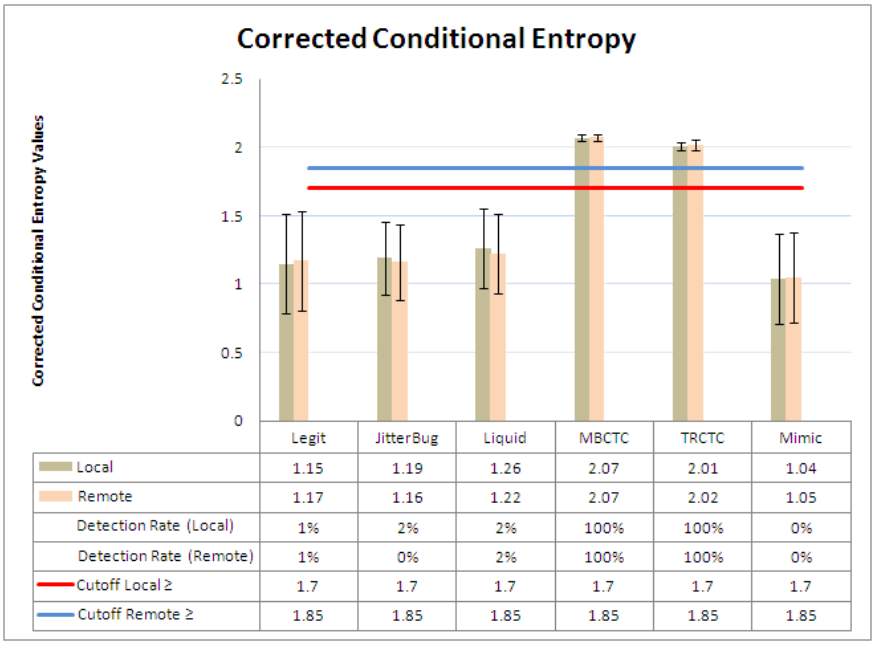
Location	Average	Standard Deviation	False Positive	Average	Standard Deviation	Detection
Local						
	legitimate			Mimic		
UTA-UTA	1.15	0.36	1%	1.04	0.33	0%
UTA-VT	1.21	0.21	1%	1.17	0.30	0%
UTA-UMass	1.10	0.19	1%	1.03	0.21	0%
Remote						
	legitimate			Mimic		
UTA-UTA	1.17	0.36	1%	1.05	0.33	0%
UTA-VT	1.13	0.29	1%	1.06	0.21	0%
UTA-UMass	1.09	0.19	1%	1.04	0.22	0%

Table 5.5 shows the CCE test scores for legitimate and Mimic traffic. Average Mimic test scores in both the local and remote locations in all the three experimental setups are lower than the corresponding legitimate test scores. Thus, Mimic is as irregular as legitimate traffic. Mimic is able to evade CCE test in all the experiments.

5.4 Comparison with other CTCs

In this section, we compare Mimic with JitterBug, Liquid, TRCTC and MBCTC. The test scores and detection rates for these CTC samples are shown in Figures 5.1 and 5.2. These results are based 100 samples of each CTC with a sample size of 2000 IPDs. Here, we have only shown the results for UTA-UTA experimental setup.

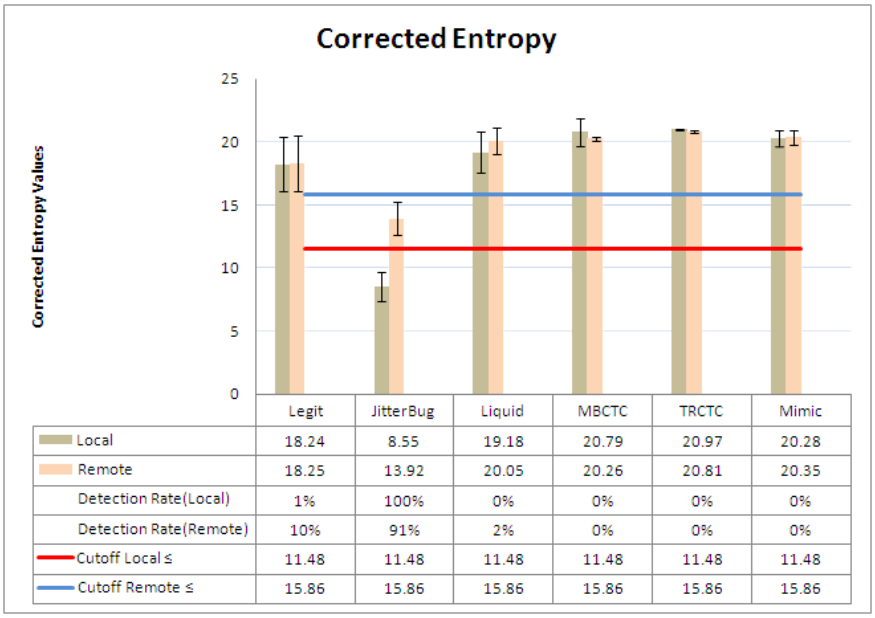
Note that the detection rate of the legitimate samples is the false positive rate, as labeled in the figures and for CEN test at remote location, we have chosen a cutoff of 10% false positives so that we better show the difference between the CTCs. Figure 5.1 shows the average test scores and detection rates for the CCE test. The average CCE values for JitterBug and Liquid are slightly higher than the legitimate



	Legit	JitterBug	Liquid	MBCTC	TRCTC	Mimic
Local	1.15	1.19	1.26	2.07	2.01	1.04
Remote	1.17	1.16	1.22	2.07	2.02	1.05
Detection Rate (Local)	1%	2%	2%	100%	100%	0%
Detection Rate (Remote)	1%	0%	2%	100%	100%	0%
Cutoff Local ≥	1.7	1.7	1.7	1.7	1.7	1.7
Cutoff Remote ≥	1.85	1.85	1.85	1.85	1.85	1.85

Figure 5.1. CCE test scores.

traffic (i.e. they are slightly more regular than the legitimate traffic) but significantly below the detection threshold. This difference is not detectable without a significant increase in false positive rate. At 1% false positive rate, the CCE detection rate for JitterBug and Liquid is less than 2% at both the local and remote locations. MBCTC and TRCTC, on the other hand, have average CCE values that are much higher than the cutoff threshold. The CCE test is able to achieve a 100% detection rates for both MBCTC and TRCTC. As discussed in Section 2, since MBCTC and TRCTC are active covert channels, it is difficult to remove regularity from them. Mimic has an average CCE values less than the legitimate traffic at both the locations (i.e at least as irregular as legitimate traffic) and thus is able to evade CCE detection. Figure 5.2 shows the average test scores and detection rates for the CEN test. The average CEN values for JitterBug are significantly below the detection cutoff values, resulting in detection rates of 100% and 91% for local and remote locations respectively. The



	Legit	JitterBug	Liquid	MBCTC	TRCTC	Mimic
Local	18.24	8.55	19.18	20.79	20.97	20.28
Remote	18.25	13.92	20.05	20.26	20.81	20.35
Detection Rate(Local)	1%	100%	0%	0%	0%	0%
Detection Rate(Remote)	10%	91%	2%	0%	0%	0%
Cutoff Local ≤	11.48	11.48	11.48	11.48	11.48	11.48
Cutoff Remote ≤	15.86	15.86	15.86	15.86	15.86	15.86

Figure 5.2. CEN test scores.

average CEN value for remote location is 5.3 bits higher than the local site. We attribute this increase to network jitter. Liquid, MBCTC, TRCTC, and Mimic all have average CEN values higher than that of legitimate traffic. Liquid has a detection rate of 2%, whereas MBCTC, TRCTC, and Mimic are not detected at all.

Our results show that JitterBug, MBCTC, and TRCTC are detectable using entropy-based detection. Liquid, however, is detected at most 2% of the time. Mimic is able to evade entropy-based detection completely. It is important to note that Mimic uses 100% throughput whereas Liquid is only able to use 50% throughput.

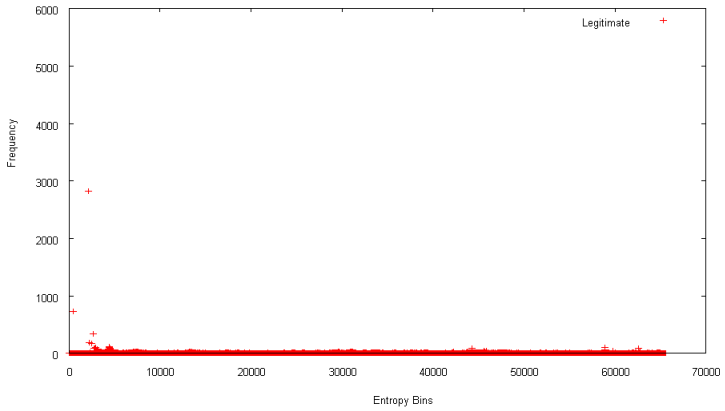
5.4.1 Bin Frequency

As described in Section 2.3.2.1, the corrected entropy test uses histograms of IPDs where each histogram is referred as a symbol. The entropy of a stream is maximized when the probability of each symbol is the same. The bin ranges are constructed in such a way that, for legitimate traffic, each symbol will be equally

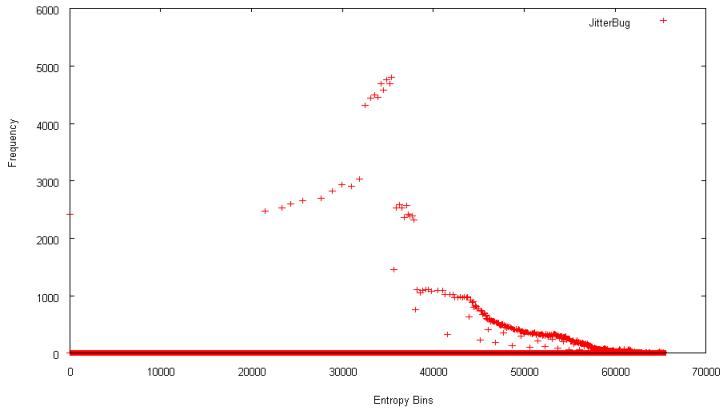
likely, thus maximizing the entropy. In Figure 5.3 we show the count of each bin for 200,000 IPDs for legitimate, JitterBug, and Liquid. In Figure 5.4, we show the count of each bin for 200,000 IPDs for MBCTC, TRCTC, and Mimic.

As shown in Figure 5.3(a), the bin frequency for the legitimate IPDs is roughly uniform. Figure 5.3(b) shows the histogram for JitterBug IPDs. Some of these IPDs are more concentrated around a small set of bins. Thus, JitterBug is easily detected by the corrected entropy test. Figure 5.3(c) shows the bin frequency for Liquid, where few bins have high frequencies, but lower than those seen in JitterBug. These high frequency bins are not detectable by the corrected entropy test as number of these bins are very less for small sample size.

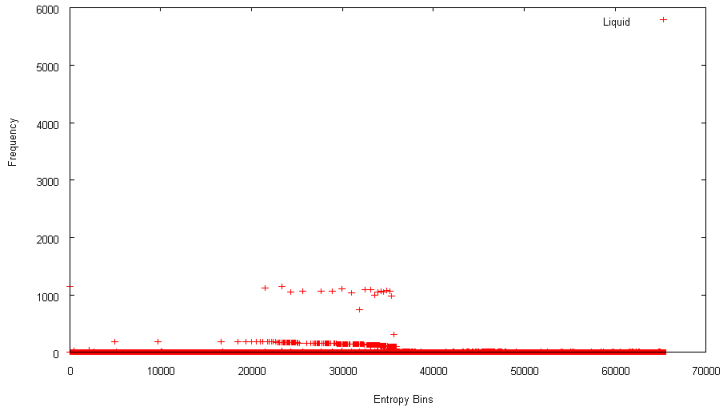
As shown in Figure 5.4, bin frequency for MBCTC, TRCTC, and Mimic are uniformly distributed and therefore not detectable by the corrected entropy test. Figure 5.5 shows the bin frequency for legitimate, MBCTC, and Mimic on a y- axis range $0, 100$ for a better comparison.



(a)

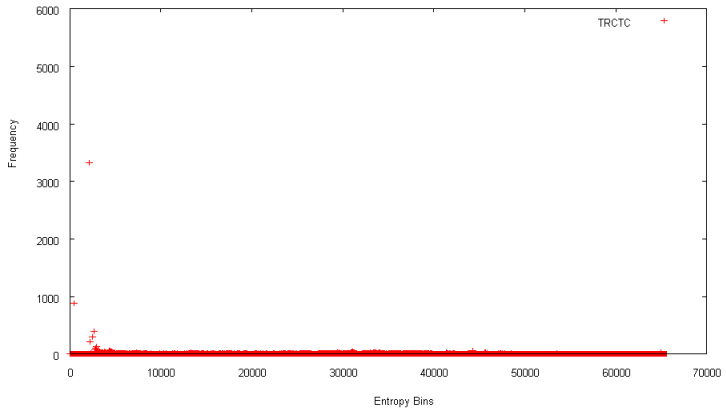


(b)

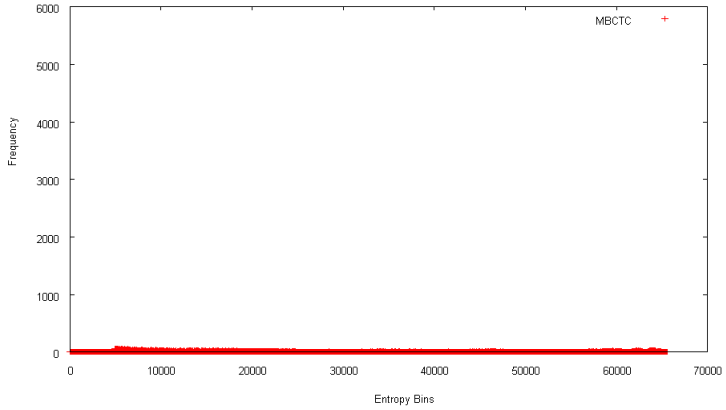


(c)

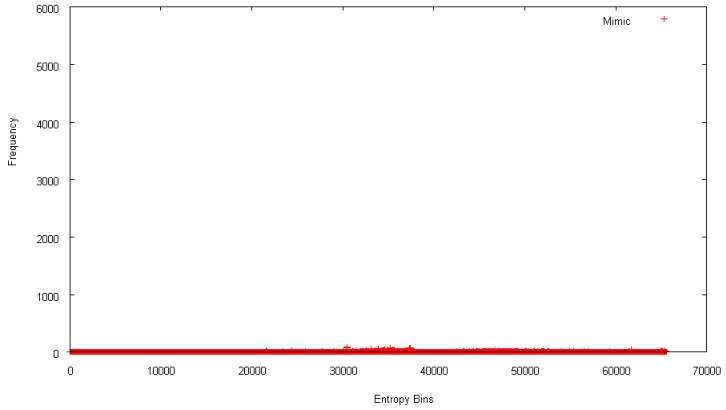
Figure 5.3. CEN Bin Frequency (a) legitimate (b) JitterBug (c) Liquid.



(a)

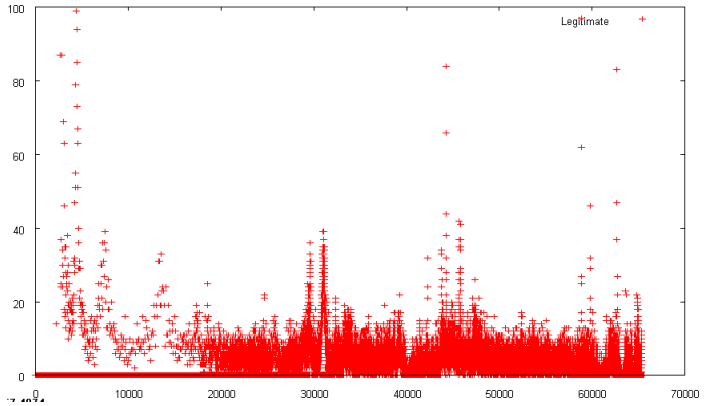


(b)

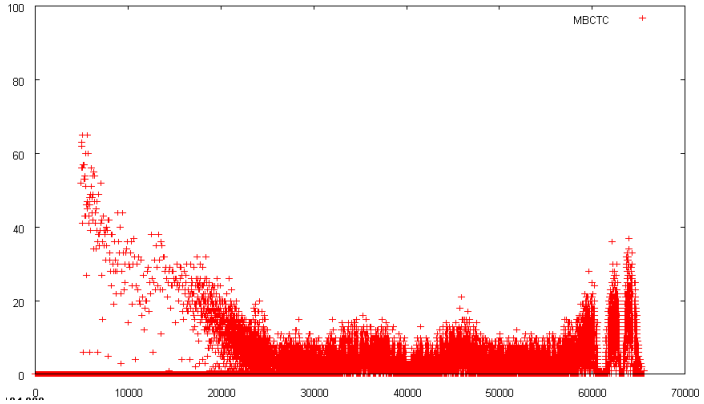


(c)

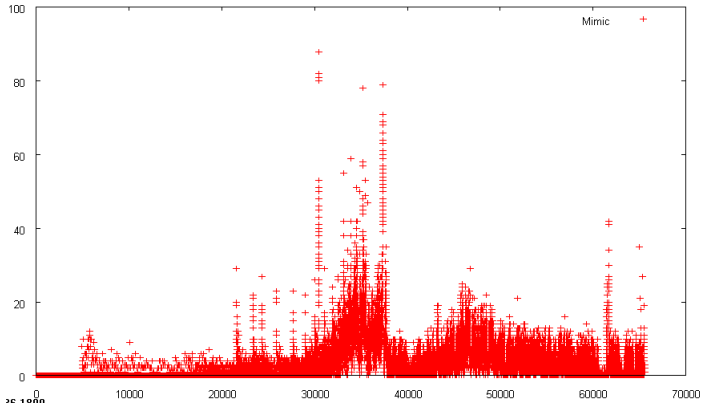
Figure 5.4. CEN Bin Frequency (a) TRCTC (b) MBCTC (c) Mimic.



(a)



(b)



(c)

Figure 5.5. CEN Bin Frequency (a) legitimate (b) MBCTC (c) Mimic.

CHAPTER 6

CONCLUSION

In this thesis, we proposed Mimic, an active covert timing channel based on Shaper and Regularity Tree. Shaper provides a way to control the shape of the distribution whereas Regularity Tree provides a way to control the regularity of the channel. Mimic combines these along with filter and modeler, to create an active CTC that is not only able to evade all known detection tests but also provides a way to create a CTC based on any legitimate traffic. The ability of our CTC to evade detection remains consistent under different network conditions. Shaper and Regularity Tree, by mimicking the statistical properties of legitimate traffic also provide a framework that can be used against future detection tests based on traffic shape and regularity. This may be by done using the existing implementation or a modified version of these two models. The main novelty of Mimic stems from its ability to smooth out the shape of the distribution while maintaining the regularity patterns of legitimate traffic. In recent years, regularity of streams have played a major role in detecting active CTCs. The idea presented by Mimic highlights this flaw in the current CTC and leverages it to design a CTC that mimics the statistical properties of legitimate traffic and thereby evades detection. Our results suggest a pressing need for a detection mechanism that factors this into consideration.

Historically, irregularity of human behavior has been a vital indicator in differentiating computers/machines from humans. The research in this paper indicates that it is possible for a deterministic machine to produce irregularity that erodes this

differentiating factor. Our techniques can be extended to other domains in computer science where a need to mimic human behavior is required.

6.1 Future Work

In future work, we would like to design an effective defense against Mimic. One important area for exploration would be the use of a combination of shape and regularity properties to define a new statistical property to distinguish CTC traffic from legitimate traffic. Furthermore, we may analyze methods for synchronization between the covert sender and receiver.

REFERENCES

[1] Serdar Cabuk. *Network Covert Channels: Design, Analysis, Detection, and Elimination*. PhD thesis, Purdue University, West Lafayette, IN., USA, 2006.

[2] Serdar Cabuk, Carla E. Brodley, and Clay Shields. IP covert timing channels: design and detection. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 178–187, New York, NY, USA, 2004. ACM.

[3] Serdar Cabuk, Carla E. Brodley, and Clay Shields. IP Covert Channel Detection. *ACM Trans. Inf. Syst. Secur.*, 12(4):1–29, 2009.

[4] U.S. Department of Defense. Trusted computer system evaluation criteria, 1985.

[5] Gina Fisk, Mike Fisk, Christos Papadopoulos, and Joshua Neil. Eliminating Steganography in Internet Traffic with Active Wardens. In *IH '02: Revised Papers from the 5th International Workshop on Information Hiding*, pages 18–35, London, UK, 2003. Springer-Verlag.

[6] Steven Gianvecchio and Haining Wang. Detecting covert timing channels: An entropy-based approach. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 307–316, New York, NY, USA, 2007. ACM.

[7] Steven Gianvecchio, Haining Wang, Duminda Wijesekera, and Sushil Jajodia. Model-Based Covert Timing Channels: Automated Modeling and Evasion. In *RAID '08: Proceedings of the 11th international symposium on Recent Advances in Intrusion Detection*, pages 211–230, Berlin, Heidelberg, 2008. Springer-Verlag.

[8] Amir Houmansadr, Negar Kiyavash, and Nikita Borisov. RAINBOW: A Robust and Invisible Non-Blind Watermark for Network Flows. In Giovannia Vigna, editor, *Network and Distributed Systems Security Symposium*. Internet Society, February 2009.

[9] W.-M. Hu. Reducing timing channels with fuzzy time. In *Research in Security and Privacy, 1991. Proceedings., 1991 IEEE Computer Society Symposium on*, pages 8–20, May 1991.

[10] Myong H. Kang and Ira S. Moskowitz. A Pump for rapid, reliable, secure communication. In *CCS '93: Proceedings of the 1st ACM conference on Computer and communications security*, pages 119–129, New York, NY, USA, 1993. ACM.

[11] Myong H. Kang, Ira S. Moskowitz, and Stanley Chincheck. The Pump: A Decade of Covert Fun. In *ACSAC '05: Proceedings of the 21st Annual Computer Security Applications Conference*, Washington, DC, USA, 2005. IEEE Computer Society.

[12] R.A. Kemmerer. A practical approach to identifying storage and timing channels: Twenty years later. In *Computer Security Applications Conference, 2002. Proceedings. 18th Annual*, pages 109–118, 2002.

[13] Richard A. Kemmerer. Shared resource matrix methodology: An approach to identifying storage and timing channels. *ACM Trans. Comput. Syst.*, 1(3):256–277, 1983.

[14] Gaurav Shah, Andres Molina, and Matt Blaze. Keyboards and covert channels. In *USENIX-SS'06: Proceedings of the 15th conference on USENIX Security Symposium*, Berkeley, CA, USA, 2006. USENIX Association.

[15] C. E. Shannon. A mathematical theory of communication. In *Bell System Technical Journal*, volume 27, July and October 1948.

[16] Dawn Xiaodong Song, David Wagner, and Xuqing Tian. Timing analysis of keystrokes and timing attacks on SSH. In *SSYM'01: Proceedings of the 10th*

conference on USENIX Security Symposium, pages 25–25, Berkeley, CA, USA, 2001. USENIX Association.

[17] Jose Torres, Jose Sarriegi, Javier Santos, and Nicolás Serrano. Managing information systems security: Critical success factors and indicators to measure effectiveness. pages 530–545. 2006.

[18] Madhu Venkateshaiah and Matthew Wright. Evading Stepping Stone Detection Under the Cloak of Streaming Media. Master's thesis, University of Texas at Arlington, 2007.

[19] Robert Walls and Matthew Wright. Liquid: A Detection resistance Covert timing channel based on IPD shaping. Master's thesis, University of Texas at Arlington, 2009.

[20] Xinyuan Wang, Shiping Chen, and Sushil Jajodia. Network Flow Watermarking Attack on Low-Latency Anonymous Communication Systems. In *SP '07: Proceedings of the 2007 IEEE Symposium on Security and Privacy*, pages 116–130, Washington, DC, USA, 2007. IEEE Computer Society.

[21] Xinyuan Wang and Douglas S. Reeves. Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 20–29, New York, NY, USA, 2003. ACM.

[22] Zhenghong Wang and Ruby Lee. Covert and side channels due to processor architecture. Proc. of ACSAC, 2006.

[23] Xiaochao Zi, Lihong Yao, Li Pan, and Jianhua Li. Implementing a passive network covert timing channel. In *Elsevier Computer and Security 2010*, January 2010.

BIOGRAPHICAL STATEMENT

Kush Kothari was born in India, in 1984. He received his B.E.(Computer Science And Engineering) degree from Sathyabama University, India in 2007. From December 2006 to July 2007, he was with System and architecture lab, CS dept., UBO, Brest, France as Research Assistant. From November 2007 to August 2008, he was with Cognizant Technology Solutions, India as a Program analyst trainee. He has been a part of iSec - The Information Security Lab at UT Arlington, an information Security Lab from 2008. In 2010, he joined Epic Systems, Madison as a Software Developer. His current research interest includes timing-based attack and in area of covert timing channel.