Mimic: An active covert channel that evades regularity-based detection

Kush Kothari\textsuperscript{a,*}, Matthew Wright\textsuperscript{b,*}

\textsuperscript{a} Qualcomm Inc. 5775 Morehouse Drive San Diego, CA 92121, USA
\textsuperscript{b} Department of Computer Science and Engineering, The University of Texas at Arlington, Arlington, TX 76019, USA

Key words: covert channels, intrusion detection, timing analysis

1. Introduction

To counter the threat of leaks of sensitive and mission-critical information, high-security facilities employ multi-level security mechanisms in which information flows are prevented from high-security systems to lower-security systems. For networks, 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 email and file transfer. 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 [1]. 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 par-
particularly difficult to detect. Most CTCs proposed to date send one bit per IPD at most. 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 CTCs.

Defenses against CTCs can be classified as **prevention**-based and **detection**-based. The primary goal of prevention-based defenses is to eliminate the possibility or make it impractical to establish a CTC. The operation 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 may unacceptably disrupt normal operations.

Detection-based approaches instead take advantage of the fact that 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.

A number of works have proposed methods to detect covert channels based on statistical differences in the shape and regularity of the channel. Detecting changes in shape can be done with any of the statistical tests that measure the difference between two distributions, such as the Kolmogorov-Smirnov Test. Cabuk et al. [2] describe a heuristic to detect increased regularity. Gianvecchio et al. [3] propose the use of entropy to measure the shape and regularity of
a network stream and differentiate legitimate traffic from covert traffic. This technique is quite effective and able to detect most existing CTCs.

1.1. Contributions

In this paper, we propose and evaluate Mimic, an active CTC that is able to evade all known detection techniques while maintaining throughput of one bit per IPD. In particular, we designed Mimic to generate traffic with regularity that cannot be easily distinguished from the regularity of legitimate traffic.

We first provide a brief discussion (§2) of the current state of research in network-based covert timing channel design and detection. We then begin our description of Mimic (§3) with a detailed description of the corrected conditional entropy (CCE) detection test proposed by Gianvecchio et al. [3]. Specifically, we focus on the 5-ary tree used in CCE (§3.1), which provides a useful model for understanding the difference between prior active covert channels and legitimate traffic. Understanding this tree thus helps us to design a covert channel that has regularity properties that are much closer to those of legitimate traffic. We note that simply generating a CTC and testing it with CCE to ensure that it is not regular will not work; the resulting sequence of IPDs will typically be highly regular and no channel will be produced. Thus, we focus on first generating a channel with low regularity. Based on this understanding, we describe the design of a Regularity Tree (§3.2) that allows an active covert channel to control the amount of regularity to match legitimate traffic.

We then describe the design of Mimic (§3.4). Mimic employs the Regularity Tree in a multi-component design that applies and extends other recent advances in covert channel design. First, legitimate traffic is collected and passed through a filter component that selects the traffic to Mimic. Then the filtered traffic is given to a modeler component that combines a Shape Modeler and a Regularity Modeler, which mimic the shape and regularity, respectively, of legitimate traffic distributions. Third, the models are used in an encoder component that determines how IPDs should be generated to encode the covert message in line with models of shape and regularity. Finally, a transmitter component sends
the traffic in accordance with the encoded IPDs. Using these four components, Mimic can extract statistical properties of any legitimate traffic and generate covert traffic with the same properties.

To evaluate the effectiveness of Mimic, we ran a set of experiments (§4) for different geographic locations of sender and receiver. Our experimental results (§5) show that Mimic has nearly the same statistical shape and regularity as legitimate traffic and that it is able to evade all known detection techniques.

2. Background

In this section, we briefly describe recent research in the design and detection of network-based covert timing channels (CTCs). We begin with an overview of covert timing channels, which we categorize as either passive or active channels, and then describe some of the methods proposed for detecting these channels. We note that Mimic uses ideas from prior work in covert channel design, and we focus more attention to CTCs from which we draw these ideas.

2.1. CTC Design

We classify CTCs into passive and active CTCs. Passive CTCs rely on existing traffic, in that they do not generate additional traffic to transmit the hidden message. Thus, they are less prone to detection from host-based intrusion detection and other system-level techniques. Passive CTCs do not have to compromise the entire machine if creatively positioned; for example, Shah et al. proposed a CTC sent from a hardware keystroke logger (a JitterBug) that would be placed between the keyboard and the computer [4]. The JitterBug CTC [4] adds very small delays (e.g. less than 20 ms) to each packet to encode one bit per IPD. JitterBug rotates the mapping of the IPD values to symbols according to a pseudo-random function to prevent recurring patterns that could be detected. Walls [5] proposes an extension of the JitterBug CTC called Liquid, in which a portion of the IPDs are used to smooth out the distortion detected by shape detection tests (e.g., Entropy Based Detection [3]).
To use an active CTC, the attacker must have compromised the victim machine. The attacker can send covert messages from this compromised machine by creating his own connection. As the attacker is creating connections without the user’s direct involvement, active CTCs are more prone to detection. However, their capacity is higher than passive CTCs, because the attacker can initiate traffic, and use high-rate flows, without waiting for the user to initiate connections.

Cabuk et al. proposed the IP Simple Covert Channel (IPSCC), an active CTC that can be a storage covert channel or timing covert channel according to the encoding used for the hidden message [6]. As it only uses a two different IPD values to encode the message, IPSCC is easy to detect. Cabuk [7] also proposed the Time Reply CTC (TRCTC). TRCTC replays randomly-selected IPDs from a set \( S \) of legitimate IPDs to send the hidden message. 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, it is therefore highly regular.

Gianvecchio et al. [8] 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 designed with the goal of being able to use any type of legitimate traffic to create a covert channel. Encoding is performed by using the inverse distribution function of the selected model and decoding is performed by using the cumulative distribution function. Gianvecchio et al. show that the shape of the MBCTC traffic is almost same as that of legitimate traffic and that it is undetectable by any known shape test [8]. However, as there is no correlation between consecutive IPDs, MBCTC is highly regular.

Sellke et al. describe a covert channel using properties derived from information theory [9]. However, these properties are based on the assumption that legitimate IPDs are independently and identically distributed (i.i.d.). In other words, the assumption is that legitimate traffic is completely regular. Liu et al. describe the construction of an active covert channel based on spreading codes.
that specifically seeks to undermine the K–S test and the Regularity test (both described below) [10]. It is difficult, however, to compare our approach directly with theirs as they do not test their channel against entropy-based detection, which is known to detect channels that the K–S test and the Regularity test do not [3]. In particular, the Regularity test is based only on consecutive pairs of IPDs, while entropy-based detection uses sequences of up to 50 IPDs at once. Thus, we believe that it is likely that the approach of Liu et al. will be detected by a test aimed at longer sequences such as entropy-based detection.

2.2. CTC Detection

The most commonly used statistical properties for detection are shape and regularity. The shape of a distribution refers, for example, to the shape of a histogram of the IPDs. Regularity of a distribution is based on the recurrence of specific patterns. A pattern might be a sequence, for example, of several long IPDs in a row followed by several short IPDs. Highly regular streams have equal recurrence of all patterns, while legitimate traffic has more irregular behavior like repeated patterns and patterns that are not observed for relatively long periods of time.

In general, passive CTCs are more prone to shape detection than active CTCs as they add delay in legitimate streams, thereby changing the shape of the IPD distribution. Active CTCs can make the encoded stream replicate the shape of legitimate traffic; MBCTC and TRCTC both produce very good shape statistics. However, active CTCs are more prone to regularity detection. Active CTCs generate traffic using a computer program, so it is harder to reproduce the irregular patterns found in legitimate streams. For passive CTCs, if the added delays are small, they will not substantially modify the patterns of legitimate traffic and the stream’s underlying irregularity will be preserved.

We now describe specific tests, starting with different shape tests and then regularity tests.

The Kolmogorov-Smirnov test (K–S test) provides a measure of the distance between the empirical distribution of two samples. The test sample, which may
or may not contain a CTC, is measured against a *training sample* of previously measured legitimate traffic. If the K–S test measure is less than a threshold, this indicates the test sample belongs to the set of legitimate traffic [3]. However, if it is above the threshold, the two samples belong to different distributions, thus suggesting the presence of a CTC in the test sample.

Cabuk et al. propose a shape test called $\epsilon$-Similarity [2]. This test sorts the IPDs in a sample and measures the number of IPDs that are less than a threshold $\epsilon$ greater, relative to the IPD value itself, than the preceding IPD in the sorted list. In a poorly-designed covert channel, many IPDs will be nearly the same as other IPDs.

Gianvecchio et al. [3] propose a detection mechanism based on Shannon entropy. In a sample of IPDs, each IPD can be mapped to one of $M$ symbols. Visually, we can picture each symbol mapping to a bin in a histogram of IPDs. The range of these bins is set based on measurements of the IPDs of a large set of legitimate traffic, such that each bin contains the same number of legitimate IPDs. In other words, each bin should have equal probability for legitimate traffic. To detect the presence of CTC, a sample of IPDs is mapped to the bins. The entropy $H$ is calculated using Shannon entropy. Any sample set whose entropy is less than the legitimate set is considered a CTC. The entropy test uses a large number of bins, e.g. $M = 2^{16} = 65536$ bins, while the size of sample set used for the test is relatively small (e.g., 2000 IPDs). 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), which also accounts for the number of empty bins.

Turning to regularity, we note that there are relatively few regularity-based tests for detecting CTCs. One such test, the Regularity Test, is based on the variance of IPDs [2]. This method is based on the fact that the variance of IPDs changes over time in legitimate traffic, but it remains relatively constant in a CTC. The Regularity Test measures the variance in IPDs over windows of time and takes a low variance as a sign of a CTC. The other test is CCE, proposed by Gianvecchio et al. [3], which is based on conditional entropy (CE). The basic
idea of the CCE method is to use a histogram of IPDs, with equiprobably bins as in the CEN test, but with fewer bins (e.g. \( M = 5 \)). Then the patterns of bin sequences are tracked to observe overlap; if sequences are repeated, the regularity is low. CCE uses a tree to track the bin sequences. The way that the CCE tree works is described in more detail in Section 3.1.

While both the CEN and CCE tests are heuristics, we have found no more effective tests for shape and regularity, respectively, in the literature. There are more well-known ways to compare the shapes of two distributions, such as Kullback-Leibler divergence (which essentially captures the same information as the CEN test). However, regularity as tested in covert channels has not been studied in depth. As such, we found no metrics of regularity beyond the two tests.

3. Mimic

In this section, we first analyze the CCE tree and then we describe the Regularity Tree mechanism used to mimic the regularity of legitimate traffic in an active covert channel. We also describe a Shaper mechanism that we use to ensure that the channel retains good shape characteristics. Finally, we describe the design of Mimic in detail.

3.1. Corrected Conditional Entropy

To understand how we can evade detection of regularity, we now describe the use of a tree structure to keep track of patterns for CCE.

In CCE, we construct a tree of height \( H \). 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 \( H \). Gianvecchio et al. used \( H = 50 \) and \( M = 5 \) bins, meaning that they constructed a 5-ary tree with \( 5^{50} \) leaves [3]. Every node has a count that indicates the number of times that node has been visited. This count also indicates the number of times a given pattern has occurred. Entropy values are calculated for every level of the tree and the minimum value of entropy among all the levels is referred to
Figure 1: Tree structure for legitimate traffic

as the conditional entropy for that set. To overcome the problem with limited datasets, Gianvecchio et al. proposed the use of corrected conditional entropy (CCE), following the work of Porta et al. [11].

Figure 1 shows the CCE tree structure for legitimate traffic. The nodes labeled $N$ 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 $N$ nodes observed in a particular level based on one hundred different samples of size 2000 IPDs.

From Figure 1, we see that legitimate traffic has a non-negligible number of IPD sub-sequences repeated at every length of sequence. This is quite surprising; the likelihood of a repeated sequence assuming two independent sequences of length 50 is $5^{50}$. By the birthday paradox, about $5^{25}$ such sequences are needed to get just a single collision with probability 0.5. However, with 1950 such sequences (2000 IPDs), an average of 135 collisions occur. In legitimate traffic, a long sequence of IPDs from just one bin may occur, helping to lead to such patterns.

In the CCE tree structure, passive CTCs (e.g., JitterBug and Liquid) have almost the same number of $N$ nodes as the legitimate traffic. This is as would be expected, since the patterns are essentially based on legitimate streams. However, the number of $N$ nodes in active CTCs (e.g., TRCTC and MBCTC)
is much less than the number of $N$ nodes in the legitimate traffic. It is observed that for active CTCs, there are no $N$ nodes in the top level of the tree. Fewer $N$ nodes represents the lack of recurrence of patterns in the stream. This follows the expected pattern for random traffic, with an even distribution across possible sequences. A stream that has a high recurrence of the same patterns results in a tree with many $N$ nodes. The higher the number of $N$ nodes, the lower the detection rate in regularity tests (until the regularity is consistently lower than legitimate traffic). From these observations, we believe that CCE provides a good measure of the regularity of the stream, as it captures a deviation from the low regularity of legitimate traffic at almost any reasonable length of IPD sequence.

Before we introduce the approach we use in Mimic, we note that some simple approaches to creating a low-regularity active CTC would not be sufficient. First, one could use a CTC like MBCTC to generate a well-shaped channel and test it for regularity using CCE. Unfortunately, nearly all IPD sequences generated by MBCTC would fail the CCE test, as they are highly regular. Thus, we believe that it is necessary to focus on generating an IPD sequence that has low regularity while keeping enough randomness to allow for shaping. Second, generating a simple low-regularity stream, e.g. by repeating the same IPD, results in a stream is easily detectable by shape tests. So the challenge in constructing a hard-to-detect active covert channel can be summarized as generating a large number of $N$ nodes while maintaining the shape of a legitimate stream.

3.2. Regularity Tree

To control irregularity in CTCs, we propose a Regularity Tree, a mechanism based on the CCE tree structure to mimic the irregularity of legitimate traffic. The main idea of the Regularity Tree is to limit the number of possible patterns used in a fixed period of time. This ensures lower regularity values; varying the number of allowed patterns allows us to control the regularity score. At the same time, the Regularity Tree allows for enough flexibility in choosing IPDs to enable us to fit the shape of legitimate traffic.
To construct the Regularity Tree, we first calculate the average number of $\mathcal{N}$ nodes $\bar{N}_h$, for each level $h$ of the CCE tree for legitimate traffic. Then we make the structure of the Regularity Tree be similar that of the CCE tree, where every node has $M$ children corresponding to the $M$ CCE bins. We build the tree based on the intuition that any $\mathcal{N}$ node $X$ makes it impossible to build longer paths in the tree that include the path from the root to $X$ as a subpath. For example, if we decide that the path $1 \rightarrow 2$ is not allowed, that means $X \rightarrow 1 \rightarrow 2$ and $1 \rightarrow 2 \rightarrow X$ are also not allowed, where $X \in \{1, 2, 3, 4, 5\}$. This allows us to construct a rule set to update lower levels of tree based on constraints created by $\mathcal{N}$ nodes selected in the upper levels.

For the first level, the rule set is empty, so we select $\bar{N}_1$ nodes randomly and mark 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. Now for every level $h$, we will first identify $\mathcal{N}$ nodes in the level based on the rule set, i.e. not allowed because of upper level $\mathcal{N}$ nodes. Then we randomly choose additional nodes in this level so that in total we get $\bar{N}_h$ $\mathcal{N}$ nodes. The paths from the root to these newly marked $\mathcal{N}$ nodes will then be added to rule set as shown in Algorithm 1.

**Algorithm 1 Design of Regularity Tree**

```
Path(K) := [Label(1st level ancestor), . . . , Label(Immediate Parent), Label(K)]
\[\bar{N}] \leftarrow \text{Average no. of } \mathcal{N} \text{ nodes in a legitimate tree.}
\mathcal{R} = \{
\text{for all } i \text{ such that } 1 \leq i \leq H \text{ do}
\text{ for all } r \in \mathcal{R}, \mathcal{N} \in \text{nodes at level } i \text{ do}
\text{ count } \leftarrow 0
\text{ if } r \cap \mathcal{R} = \mathcal{R} \text{ then}
\text{ Discard node } \mathcal{N}
\text{ count } \leftarrow \text{count + 1}
\end{if}
\end{for}
\text{ D } \leftarrow \text{Select at random } \bar{N}[i] - \text{count nodes from level } i.
\text{ } \mathcal{R} = \mathcal{R} \cup \mathcal{R}(n); n \in D
\text{ Mark } n \text{ as } \mathcal{N}; n \in D\n\end{for}
```

To generate the $\bar{N}$ values we take a sample of 10000 IPDs and generate five
CCE trees for 2000 IPDs. The average of these nodes is stored in $\bar{N}$.

3.3. **Shaper**

Similar to the way we use the CCE test to improve the regularity of our channel, we can use the CEN test to ensure that the shape of our channel is as close to that of legitimate traffic as possible. To this end, we propose a mechanism called Shaper to control the shape of the covert channel. If we have equi-probable bins of IPDs, legitimate IPDs are uniformly distributed among these bins [3]. Shaper uses this knowledge to provide a way to control the shape of the covert traffic.

Shaper can be used to decide whether an encoded IPD belongs to the given distribution or not. It keeps a track of the bin frequency of all the previously encoded IPDs. 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 and otherwise it does not. Threshold $t$ for the frequency can either be the same for all bins or can be individually determined for each bin based on the legitimate traffic.

Shaper uses a bin of the IPDs, similar to the shaping used in Liquid [5]. In Liquid, however, a portion of IPDs are modified in such a way that they increase the frequencies of bins that are not used. As Mimic instead is an active CTC, Shaper instead chooses the IPDs in such a way that they are uniformly distributed among the bins.
3.4. Mimic

Based on the Regularity Tree, we propose a new active CTC called *Mimic* that has a channel capacity of 1.0 bit per IPD (or between 0.874 and 0.983 bits per IPD when errors are considered) and, to the best of our knowledge, is undetectable by all known detection tests.

3.4.1. Design of Mimic

While it is relatively easy to evade either shape detection or regularity detection, doing both is challenging. The basic idea of our design is to first choose a desired regularity pattern with a reasonable level of flexibility and then choose packets that meet the bit transmission and shaping requirements from within the bounds of the regularity pattern.

*Mimic*, as shown in figure 2 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 very similar to the analyzer of MBCTC [8]. We begin by taking a sample of IPDs and fit it 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 model with 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 in parallel to the shape modeler. The main objective of the regularity modeler is find the values of $N_h$ at each height $h$ of the CCE tree, which are needed to build the Regularity Tree as described in
Section 3.2.

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, described in Section 2.1. This encoding is based on modular arithmetic, which is neither based on order (as in TRCTC) nor on pseudo random numbers (as in MBCTC). As the encoding is independent of IPD selection, we can encode bit ‘0’ or ‘1’ using any IPD.

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 Regularity Tree using DFS and store all the paths $P_i$ from root to leaf in set $P$. Now a path $P_i$ is randomly chosen from the set $P$ and added to the bin sequence. We encode the message using Algorithm 2.

Algorithm 2 Mimic Encoding

\begin{algorithm}
\begin{algorithmic}
\REPEAT
\WHILE{$IPD_e \notin B_i$}
\STATE Randomly select an $IPD : IPD \in B_i, B_i \in B$
\STATE $IPD_e = IPD +$ Delay added to encode message bit by JitterBug
\ENDWHILE
\STATE Use Shaper to decide whether $IPD_e \in$ current distribution
\UNTIL{$IPD_e \notin$ current distribution}
\end{algorithmic}
\end{algorithm}

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. The Shaper is used to determine if the IPD is a valid value of the current distribution. This process is repeated until an 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 possible IPD values for the first node in the Regularity Tree. In this case, we ignore the
regularity constraint.

We note that, overall, the Shaper has a small role in keeping Mimic from being detected. The Regularity Tree allows enough randomness within each node in the sequence to spread out the IPDs over the bins in the distribution.

Mimic is undetectable by any known detection test as described in Section 5. Mimic is able to generate covert traffic over any legitimate traffic while ensuring 100% channel capacity.

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 covert streams and to perform detection.

4.1. Data Collection

All the IPDs used in our experiments are taken from network traces collected at the University of North Carolina at Chapel Hill (UNC). We extract SSH IPDs from these traces, since SSH is the protocol used by both Liquid and JitterBug. Since covert channels use unidirectional communication, only the traces from client to server are considered and not vice versa. To do this, we first 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. To emulate a real attack scenario, the detector has more data from the network than the attacker and they use different data sets. In other words, the TrainingSet is larger and different from the SampleSet. The size of each set is shown in Table 1.
Table 1: Experimental Set Sizes. Sizes of the TrainingSet and the SampleSet.

<table>
<thead>
<tr>
<th>Type</th>
<th>Number of IPDs</th>
</tr>
</thead>
<tbody>
<tr>
<td>TrainingSet</td>
<td>8,632,235</td>
</tr>
<tr>
<td>SampleSet</td>
<td>2,426,264</td>
</tr>
</tbody>
</table>

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 sends a keystroke directly to the SSH session at the end of each IPD period. We call a sequence of IPDs a 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, 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 [3]. 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 $N_t$ 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 [5].

**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.
Table 2: Parameters for different distribution models for legitimate SSH traffic

<table>
<thead>
<tr>
<th>Model</th>
<th>Parameters</th>
<th>RMSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weibull</td>
<td>0.1279, 0.4401</td>
<td>0.3030</td>
</tr>
<tr>
<td>Gamma</td>
<td>0.2360, 3.7353</td>
<td>0.3170</td>
</tr>
<tr>
<td>Lognormal</td>
<td>-3.1641, 2.3639</td>
<td>0.3510</td>
</tr>
<tr>
<td>Pareto</td>
<td>1.1871, 0.0476</td>
<td>0.3349</td>
</tr>
<tr>
<td>Poisson</td>
<td>0.8815</td>
<td>0.3141</td>
</tr>
<tr>
<td>Exponential</td>
<td>0.8851</td>
<td>0.3144</td>
</tr>
</tbody>
</table>

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 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.

**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. 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 the sender and
receiver. In the *UTA-UMass* experiments, the sender was located at the University of Texas at Arlington (UTA) and the receiver was located at the University of Massachusetts at Amherst (UMass). In the *UTA-VT* experiments, the sender was located at Virginia Tech University (VT) and the receiver was located at UTA.

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 1st percentile was used for the corrected entropy test, while the 99th 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.

5. Experimental Results

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

5.1. Detection Resistance

As described in Section 4, we conducted our experiments over two connections: UTA-UMass and UTA-VT. For each connection, we ran 100 samples of
### Table 3: Cutoff score for CEN and CCE in UTA-UMass Experimental Setup

<table>
<thead>
<tr>
<th>Test</th>
<th>Score</th>
<th>False Positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CEN</td>
<td>$\leq 11.48$</td>
<td>1%</td>
</tr>
<tr>
<td>CCE</td>
<td>$\geq 1.7$</td>
<td>1%</td>
</tr>
<tr>
<td>Remote</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CEN</td>
<td>$\leq 11.84$</td>
<td>1%</td>
</tr>
<tr>
<td>CCE</td>
<td>$\geq 1.85$</td>
<td>1%</td>
</tr>
</tbody>
</table>

### Table 4: Regularity Test Results

<table>
<thead>
<tr>
<th>Location</th>
<th>False Pos.</th>
<th>Ave.</th>
<th>Std. Dev.</th>
<th>Detection</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UTA-VT</td>
<td>1%</td>
<td>19.7</td>
<td>39.54</td>
<td>0%</td>
</tr>
<tr>
<td>UTA-VT</td>
<td>10%</td>
<td>19.7</td>
<td>39.54</td>
<td>0%</td>
</tr>
<tr>
<td>UTA-UMass</td>
<td>1%</td>
<td>31.7</td>
<td>67.0</td>
<td>0%</td>
</tr>
<tr>
<td>UTA-UMass</td>
<td>10%</td>
<td>31.7</td>
<td>67.0</td>
<td>2%</td>
</tr>
<tr>
<td>Remote</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UTA-VT</td>
<td>1%</td>
<td>18.29</td>
<td>35.79</td>
<td>0%</td>
</tr>
<tr>
<td>UTA-VT</td>
<td>10%</td>
<td>18.29</td>
<td>35.79</td>
<td>0%</td>
</tr>
<tr>
<td>UTA-UMass</td>
<td>1%</td>
<td>36.86</td>
<td>75.76</td>
<td>0%</td>
</tr>
<tr>
<td>UTA-UMass</td>
<td>10%</td>
<td>36.86</td>
<td>75.76</td>
<td>4%</td>
</tr>
</tbody>
</table>

### Table 5: Corrected Entropy Test Scores

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Local</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UTA-UMass</td>
<td>18.24</td>
<td>2.19</td>
<td>1%</td>
<td>20.28</td>
<td>0.63</td>
<td>0%</td>
</tr>
<tr>
<td>UTA-VT</td>
<td>19.78</td>
<td>1.59</td>
<td>1%</td>
<td>20.40</td>
<td>0.59</td>
<td>0%</td>
</tr>
<tr>
<td>Remote</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>UTA-UMass</td>
<td>18.25</td>
<td>2.20</td>
<td>1%</td>
<td>20.35</td>
<td>0.61</td>
<td>0%</td>
</tr>
<tr>
<td>UTA-VT</td>
<td>18.93</td>
<td>3.06</td>
<td>1%</td>
<td>20.51</td>
<td>0.62</td>
<td>0%</td>
</tr>
</tbody>
</table>
Table 6: Corrected Conditional Entropy Test Scores

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Local</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>False</td>
<td></td>
<td></td>
<td>legitimate</td>
</tr>
<tr>
<td>UTA-UMass</td>
<td>1.15</td>
<td>0.36</td>
<td>1%</td>
<td>1.04</td>
<td>0.33</td>
<td>0%</td>
</tr>
<tr>
<td>UTA-VT</td>
<td>1.21</td>
<td>0.21</td>
<td>1%</td>
<td>1.17</td>
<td>0.30</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Remote</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>False</td>
<td></td>
<td></td>
<td>legitimate</td>
</tr>
<tr>
<td>UTA-UMass</td>
<td>1.17</td>
<td>0.36</td>
<td>1%</td>
<td>1.05</td>
<td>0.33</td>
<td>0%</td>
</tr>
<tr>
<td>UTA-VT</td>
<td>1.13</td>
<td>0.29</td>
<td>1%</td>
<td>1.06</td>
<td>0.21</td>
<td>0%</td>
</tr>
</tbody>
</table>

Table 7: Results for SampleSet Size

<table>
<thead>
<tr>
<th>SampleSet Size</th>
<th>2000</th>
<th>3000</th>
<th>4000</th>
<th>5000</th>
<th>6000</th>
<th>7000</th>
<th>8000</th>
<th>9000</th>
<th>10000</th>
</tr>
</thead>
<tbody>
<tr>
<td>False Positive</td>
<td>1%</td>
<td>1.5%</td>
<td>2%</td>
<td>2.5%</td>
<td>3%</td>
<td>3.5%</td>
<td>4%</td>
<td>4.5%</td>
<td>5%</td>
</tr>
<tr>
<td>Detection Rate</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>5%</td>
<td>3%</td>
<td>0%</td>
<td>0%</td>
<td>5%</td>
<td>5%</td>
</tr>
</tbody>
</table>

Table 8: Results for varying CCE Bin

<table>
<thead>
<tr>
<th>CCE Bins</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>15</th>
<th>20</th>
</tr>
</thead>
<tbody>
<tr>
<td>False Positive</td>
<td>1%</td>
<td>1%</td>
<td>1%</td>
<td>1%</td>
<td>1%</td>
<td>1%</td>
<td>1%</td>
<td>1%</td>
</tr>
<tr>
<td>Detection Rate</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>5%</td>
<td>3%</td>
<td>0%</td>
<td>0%</td>
<td>5%</td>
</tr>
</tbody>
</table>

Table 9: Results for varying maximum length of sequence

<table>
<thead>
<tr>
<th>Max. Length of Sequence</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
</tr>
</thead>
<tbody>
<tr>
<td>False Positive</td>
<td>1%</td>
<td>1%</td>
<td>1%</td>
<td>1%</td>
<td>1%</td>
<td>1%</td>
</tr>
<tr>
<td>Detection Rate</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
</tbody>
</table>
legitimate and Mimic traffic with 2000 IPDs. We captured the IPDs at local as well as remote locations. The corrected entropy, Regularity, and corrected conditional entropy tests were performed for each sample at both locations. 1% and 10% false positive rates for legitimate traffic are used as cutoff scores. Samples with a corrected entropy value less than or equal to the CEN or Regularity Test cutoff score and corrected conditional entropy greater than or equal to the CCE cutoff score are considered to contain covert channels. Table 3 shows the cutoff scores for local and remote locations in UTA-UMass experimental setup. Table 4 shows that the detection rate for Mimic was at most 4% in all experimental setups for a fixed false positive rate of 10%. With 1% false positive rate, Mimic is able to achieve 100% detection resistance. Table 5 shows the CEN test scores for both legitimate and Mimic traffic. Average Mimic test scores in both the local and remote locations for both connections are higher than the corresponding legitimate test scores (i.e. Mimic IPDs are uniformly distributed among the bins). Mimic is able to evade the CEN test in all the experiments. Table 6 shows the CCE test scores for legitimate and Mimic traffic. Average Mimic test scores in both the local and remote locations in both the 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 experiments. In Table 7, Table 8, and Table 9, we report the detection rates for varying sample size, varying number of CCE bins, and varying maximum sequence length, respectively. They show that Mimic evades detection in CCE for a variety of parameters for the CCE test. This is important, as it shows that our evasion results are not limited to the attacker being able to match the parameters to the CCE test that is running.

5.2. Decoding

We now describe the decoding error rates for the covert messages that we sent in our experiments. In our local experiments (UTA-UTA) with 10,000 IPDs (10000 bits of information transmitted), no packets were dropped, only two bits were flipped when observed at the sending-side router, and seven bits
total were flipped when observed at the receiver, resulting in an error rate of 0.07%. In the remote experiment (UTA-UMass) with 1000 IPDs (1000 bits of information), four packets were dropped as a result of network congestion, four bits were flipped at the sender-side router, and 15 bits total were flipped at the receiver, resulting in an error rate of 1.5%.

Thus, we see that we can decode the covert message with a very small error rate. While some covert channels include a sophisticated error-mitigating encoding technique [10], we note a standard error-correcting code could be used to mitigate these errors with reasonable overhead. For example, a Reed-Solomon (255, 223) code could encode 892 bits in 1021 IPDs for 0.874 bits per IPD and correct errors in up to 16 different bytes out of 223 transmitted [12]. For a local setting, a (255, 251) code should be sufficient, encoding 1004 bits in 1021 IPDs (0.983 bits per IPD) and enabling error correction in two of the 251 bytes sent.

5.3. 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 3 and 4. These results are based 100 samples of each CTC with a sample size of 2000 IPDs. In comparison, we have only shown the results for UTA-UMass experimental setup; the other results are similar.

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 chose a cutoff with 10% false positives to better see the difference between the different CTCs.

Figure 3 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 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 both 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, MBCTC and TRCTC being 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 atleast as irregular as legitimate traffic)and thus, is able to evade CCE detection. Figure 4 shows the average test scores and detection rates for 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 average CEN value for the 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 the 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, as an active channel with a capacity of one bit per IPD, has greater throughput than Liquid, a passive channel with a capacity of 0.5 bits per IPD.

5.3.1. Bin Frequency

As described in Section 3.1, 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 we show the count of each bin for 200,000 IPDs for legitimate, JitterBug, and Mimic. As shown in Figure 5(a), the bin frequency for the legitimate IPDs is roughly uniform. Figure 5(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
Figure 5: Histograms of the training bins used by the corrected entropy test for different sets of traffic: (a) legitimate (b) JitterBug (c) Mimic
entropy test. Figure 5(c) shows the bin frequency for Mimic. As we can see, the IPDs are roughly uniformly distributed among the bins, meaning that Mimic is able to evade the CEN test.

6. Conclusion

In this paper, we proposed Mimic, an active covert timing channel based on a Regularity Tree. Mimic’s design employs the Regularity Tree to control the regularity of the channel without distorting the channel’s shape. We showed through experimentation that Mimic successfully evades all known CTC detection tests under a variety of network conditions. The design of Mimic also provides a way to create a CTC based on any legitimate traffic. The Shaper and the Regularity Tree, by mimicking the statistical properties of legitimate traffic, provide a framework that can be used against future detection tests based on shape and regularity of traffic. This may be done by 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. While we know of one other such CTC, we are the first to test our approach against entropy-based detection, which is known to detect channels that the regularity test does not.

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. It is possible that our basic approach can be extended to other domains in computer science where a need to mimic human behavior is required.

In future work, we would like to design an effective defense against channels like Mimic. We believe that it is interesting to explore the idea of regularity in more detail than we have seen in the literature, as further attributes of the distribution may be fundamentally more difficult to mimic than the sequences used in our regularity tree.
Acknowledgments.

Thanks to Steven Gianvecchio and Haining Wang for providing code for MBCTC. Thanks to Robert Walls for code and traces. Thanks also to Robert Walls and Sumit Ahuja for allowing us to use their servers for remote experiments and to Ryan McKenzie at UNC for network traces. This material is based upon work supported by the National Science Foundation under CAREER Grant No. 0954133.

References


