Model-Based Covert Timing Channels: Automated Modeling and Evasion

Steven Gianvecchio[1], Haining Wang[1], Duminda Wijesekera[2], and Sushil Jajodia[2]

[1] Department of Computer Science
College of William and Mary, Williamsburg, VA 23187, USA
{srgian,hnw}@cs.wm.edu
[2] Center for Secure Information Systems
George Mason University, Fairfax, VA 22030, USA
{dwijesek,jajodia}@gmu.edu

Abstract. The exploration of advanced covert timing channel design is important to understand and defend against covert timing channels. In this paper, we introduce a new class of covert timing channels, called model-based covert timing channels, which exploit the statistical properties of legitimate network traffic to evade detection in an effective manner. We design and implement an automated framework for building model-based covert timing channels. Our framework consists of four main components: filter, analyzer, encoder, and transmitter. The filter characterizes the features of legitimate network traffic, and the analyzer fits the observed traffic behavior to a model. Then, the encoder and transmitter use the model to generate covert traffic and blend with legitimate network traffic. The framework is lightweight, and the overhead induced by model fitting is negligible. To validate the effectiveness of the proposed framework, we conduct a series of experiments in LAN and WAN environments. The experimental results show that model-based covert timing channels provide a significant increase in detection resistance with only a minor loss in capacity.

Keywords: covert timing channels, traffic modeling, evasion.

1 Introduction

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]. There are two types of covert channels: covert storage channels and covert timing channels. A covert storage channel manipulates the contents of a storage location (e.g., disk, memory, packet headers, etc.) to transfer information. A covert timing channel manipulates the timing or ordering of events (e.g., disk accesses, memory accesses, packet arrivals, etc.) to transfer information. The focus of this paper is on covert timing channels.

The potential damage of a covert timing channel is measured in terms of its capacity. The capacity of covert timing channels has been increasing with the development of high-performance computers and high-speed networks. While covert timing channels studied in the 1970s could transfer only a few bits per second [2], covert timing channels in modern computers can transfer several megabits per second [3]. To defend against covert timing channels, researchers have proposed various methods to detect and disrupt them. The disruption of covert timing channels manipulates traffic to slow or stop covert timing channels [4–8]. The detection of covert timing channels mainly uses statistical tests to differentiate covert traffic from legitimate traffic [9–13]. Such detection methods are somewhat successful, because most existing covert timing channels cause large deviations in the timing behavior from that of normal traffic, making them relatively easy to detect.

In this paper, we introduce model-based covert timing channels, which endeavor to evade detection by modeling and mimicking the statistical properties of legitimate traffic. We design and develop a framework for building model-based covert timing channels, in which hidden information is carried through pseudo-random values generated from a distribution function. We use the inverse distribution function and cumulative distribution function for encoding and decoding. The framework includes four components, filter, analyzer, encoder, and transmitter. The filter profiles the legitimate traffic, and the analyzer fits the legitimate traffic behavior to a model. Then, based on the model, the encoder chooses the appropriate distribution functions from statistical tools and traffic generation libraries to create covert timing channels. The distribution functions and their parameters are determined by automated model fitting. The process of model fitting proves very efficient and the induced overhead is minor. Lastly, the transmitter generates covert traffic and blends with legitimate traffic.

The two primary design goals of covert timing channels are high capacity and detection resistance. To evaluate the effectiveness of the proposed framework, we perform a series of LAN and WAN experiments to measure the capacity and detection resistance of our model-based covert timing channel. We estimate the capacity with a model and then validate the model with real experiments. Our experimental results show that the capacity is close to that of an optimal covert timing channel that transmits in a similar condition. In previous research, it is shown that the shape [9, 10] and regularity [11, 12] of network traffic are important properties in the detection of covert timing channels. We evaluate the detection resistance of the proposed framework using shape and regularity tests. The experimental results show that both tests fail to differentiate the model-based covert traffic from legitimate traffic. Overall, our model-based covert timing channel achieves strong detection resistance and high capacity.

There is an arms race between covert timing channel design and detection. To maintain the lead, researchers need to continue to improve detection methods and investigate new attacks. The goal of our work is to increase the understanding of more advanced covert timing channel design. We anticipate that our demonstration of model-based covert timing channels will ultimately lead to the development of more advanced detection methods.

The remainder of the paper is structured as follows. Section 2 surveys related work. Section 3 provides background information on covert timing channels and describes two base cases in their design. Section 4 details the design and implementation of the proposed framework. Section 5 validates the effectiveness of the model-based covert timing channel through live experiments over the Internet. Finally, we conclude the paper and discuss future directions in Section 6.

2 Related Work

To defend against covert timing channels, researchers have proposed different solutions to detect and disrupt covert traffic. The disruption of covert timing channels adds random delays to traffic, which reduces the capacity of covert timing channels but reduces the network performance as well. The detection of covert timing channels is mainly accomplished using statistical tests to differentiate covert traffic from legitimate traffic. While the focus of earlier work is on the disruption of covert timing channels [4–8], more recent research has begun to investigate the design and detection of covert timing channels [9–12, 14].

Kang et al. [5] designed a device, known as "The Pump," which reduces the capacity of covert timing channels by disrupting the timing of communication. This device increases the number of errors by randomly manipulating the timing values. The basic version of "The Pump" is designed to address covert timing channels within systems. A network version was later designed and developed [6,7]. Giles et al. [8] studied the disruption of covert timing channels from a game theoretic perspective. The authors takes the point of view of both the jammer and the covert timing channel, and discusses the strategies for both optimal jammers and optimal input processes. Fisk et al. [4] investigated the concept of Active Wardens in relation to covert channels. The authors introduced the quantity of Minimal Requisite Fidelity (MRF), which is the minimum fidelity needed to support the communication channel, and proposed a system to identify and eliminate unneeded fidelity in traffic that could be used for covert channels.

Cabuk et al. [11] designed and implemented a simple covert timing channel and showed that the regularity of the covert timing channel can be used in its detection. To disrupt the regularity, the authors tried two approaches. The first is to change the timing intervals, which is still successfully detected. The second is to introduce noise in the form of legitimate traffic. However, the covert timing channel is still sometimes detected, even with 50% of the inter-packet delays being legitimate traffic. This covert timing channel has similar regularity test scores to Fixed-average Packet Rate (FPR) and OPtimal Capacity (OPC) (described in Section 3) but transmits information more slowly.

Berk et al. [9, 10] proposed a scheme for detecting binary and multi-symbol covert timing channels. The detection method measures the distance between the mean and modes, with a large distance indicating a potential covert timing channel. The detection test assumes a normal distribution for the inter-packet delays and, as a result, is not applicable to the covert timing channels we discussed. The authors used the Arimoto-Blahut algorithm [15, 16] in the binary case without considering the cost. In contrast, we use the Arimoto-Blahut algorithm in the multi-symbol case but with a cost constraint, to formulate the optimal input distribution for FPR.

Shah et al. [12] developed a keyboard device, called *JitterBug*, to create a loosely-coupled covert channel capable of leaking information typed on a keyboard over the network. Such a covert timing channel takes advantage of small delays in key-presses to affect the inter-packet delays of a networked application. As a result, the keyboard slowly leaks information to an observer outside of the network. The authors showed that the initial scheme leaves a regular pattern in the inter-packet delays, which can be removed by rotating the position of the window. The JitterBug transmits information much more slowly than our model-based covert timing channel, but does so under tighter constraints on the transmission mechanism.

Borders et al. [17] developed a system, called *Web Tap*, to detect covert tunnels in web traffic based on header fields, inter-request delays, request sizes, capacity usage, request regularity, and request time. Such a system is successful in detecting several spyware and backdoor programs. However, the technique used by our model-based covert timing channel to mimic the inter-request delays and request regularity of traffic, could be used by spyware and backdoor programs to evade the Web Tap.

While some recent research has taken steps to better hide covert timing channels [11, 12], these works focus on removing regularity rather than making the covert timing channel look like legitimate traffic. Moreover, removing regularity is the last step in the covert channel design process, instead of a consideration up front. In contrast, our framework is designed from the ground up to provide high detection resistance. As a result, the proposed model-based covert timing channel is able to provide much stronger detection resistance than most practical implementations of covert timing channel presented in the literature.

There are recent works on using timing channels to watermark traffic [18,19] and on detecting such timing-based watermarks [20]. Wang et al. [18] developed a robust watermarking scheme for tracing encrypted attack traffic through stepping stones. The scheme, through the use of redundancy, can resist arbitrarily large timing perturbations, if there are a sufficient number of packets to watermark. Peng et al. [20] investigated how to detect such watermarks, as well as methods for removing or duplicating the watermarks. Yu et al. [19] developed a sophisticated technique for hiding watermarks by disguising them as pseudonoise. There are some interesting differences between timing-based watermarking and traditional covert timing channels, such as the fact that the defender, not the attacker, uses the timing channel in the watermarking schemes.

3 Background

In this section, we describe basic communication concepts and relate them to covert timing channels. Then, based on these concepts, we formulate two base cases in covert timing channel design. The basic problem of communication, producing a message at one point and reproducing that message at another point, is the same for both overt and covert channels, although covert channels must consider the additional problem of hiding communication.

3.1 Basic Communication Concepts

The capacity of a communication channel is the maximum rate that it can reliably transmit information. The capacity of a covert timing channel is measured in bits per time unit [21]. The capacity in bits per time unit C_t is defined as:

$$C_t = \max_X \frac{I(X;Y)}{E(X)},$$

where X is the transmitted inter-packet delays or input distribution, Y is the received inter-packet delays or output distribution, $I(X;Y)$ is the mutual information between X and Y, and $E(X)$ is the expected time of X.

The mutual information measures how much information is carried across the channel from X to Y. The mutual information $I(X;Y)$ is defined as:

$$I(X;Y) = \begin{cases} \sum\limits_X \sum\limits_Y P(y \mid x) P(x) \log \frac{P(y|x)P(x)}{P(x)P(y)}, \text{(discrete)} \\ \int\limits_X \int\limits_Y P(y \mid x) P(x) \log \frac{P(y|x)P(x)}{P(x)P(y)}\, dx\, dy, \text{(continuous)} \end{cases}$$

The noise, represented by the conditional probability in the above definitions, is defined as:

$$P(y \mid x) = f_{noise}(y, x),$$

where f_{noise} is the noise probability density function, x is the transmitted inter-packet delays, and y is the received inter-packet delays.

The noise distribution f_{noise} is the probability that the transmitted inter-packet delay x results in the received inter-packet delay y. The specific noise distribution for inter-packet delays is detailed in Section 5.2.

3.2 Base Cases in Design

The two main goals of covert timing channel design are high capacity and detection resistance. There are few examples of practical implementations of covert timing channels in the literature, so we begin to explore the design space in terms of both capacity and detection resistance. The focus of our model-based covert timing channel is to achieve high detection resistance. In the following section, we formulate two base cases in covert channel design as comparison to the model-based covert timing channel.

The first case, optimal capacity, transmits as much information as possible, sending hundreds or more packets per second. Such a design might not be able to achieve covert communication, but is useful as a theoretical upper bound. The second case, fixed average packet rate, sends packets at a specific fixed average packet rate, encoding as much information per packet as possible. The fixed average packet rate is mainly determined by the packet rate of legitimate traffic.

Optimal Capacity Channel The first design, OPtimal Capacity (OPC), uses the discrete input distribution that transmits information as fast as possible. The optimal capacity is dependent on the optimal distance between two symbols. The first symbol is (approximately) zero and the second symbol is non-zero, so the use of more symbols (i.e., four or eight) will introduce more non-zero symbols and decrease the symbol rate. The use of smaller distances between the two symbols increases the symbol rate and the error rate. The optimal distance is the point at which the increase in error rate balances the increase in symbol rate.

The code operates based on two functions. The encode function is defined as:

$$F_{encode}(s) = d_s = \begin{cases} 0, & s = 0 \\ d, & s = 1 \end{cases}$$

where s is a symbol, d_s is an inter-packet delay with a hidden symbol s, and d is the optimal distance between the two symbols. The decode function is defined as:

$$F_{decode}(d_s) = s = \begin{cases} 0, & d_s < \frac{1}{2}d \\ 1, & \frac{1}{2}d \leq d_s \end{cases}$$

where d_s is an inter-packet delay with a hidden symbol s.

Channel Capacity: The channel capacity of OPC is dependent on the optimal input distribution and noise. The input distribution is defined as:

$$P(x) = \begin{cases} p, & x = d \\ 1 - p, & x = 0 \\ 0, & \text{otherwise} \end{cases}$$

where p is the probability of the symbol $s = 1$, and $1 - p$ is the probability of the symbol $s = 0$.

Therefore, the capacity of OPC is the maximum of the mutual information with respect to the parameters d and p of the input distribution over the expected time $d \cdot p$:

$$C_t = \max_{d,p} \frac{1}{d \cdot p} \sum_X \sum_Y P(y \mid x) P(x) \text{log} \frac{P(y \mid x) P(x)}{P(x) P(y)}.$$

Fixed-Average Packet Rate Channel The second design, Fixed-average Packet Rate (FPR), uses the input distribution that encodes as much information per packet as possible with a constraint on the average cost of symbols. The cost is measured in terms of the time required for symbol transmission. Therefore, the optimal input distribution is subject to the constraint on the average packet rate, i.e., the cost of symbol transmission.

The optimal input distribution for FPR is computed with the Arimoto-Blahut algorithm generalized for cost constraints [16]. The Arimoto-Blahut algorithm computes the optimal input distribution for capacity in bits per channel usage. The capacity in bits per channel usage C_u is defined as:

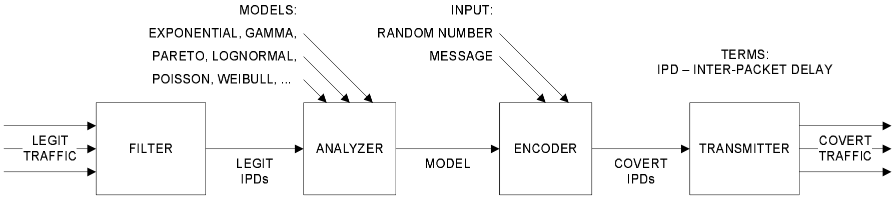
$$C_u = \max_{X} I(X;Y).$$

In general, C_u and C_t do not have the same input distribution X. However, if the input distribution is constrained so that $E(X) = c$ (where c is a constant), then the optimal input distribution X is optimal for both C_u and C_t, and $C_u = C_t \cdot c$. Thus, FPR transmits as much information per packet (channel usage) and per second (time unit) as possible with a fixed average packet rate. We use the Arimoto-Blahut algorithm to compute the optimal input distribution for FPR. The capacity results for FPR, based on the Arimoto-Blahut algorithm, are detailed in Section 5.

4 The Framework

The covert timing channel framework, as shown in Figure 1, is a pipeline that filters and analyzes legitimate traffic then encodes and transmits covert traffic. As the output of the pipeline, the covert traffic mimics the observed legitimate traffic, making it easy to evade detection. The components of the framework include filter, analyzer, encoder, and transmitter, which are detailed in the following paragraphs.

Fig. 1. Framework for building model-based covert timing channels.



The filter monitors the background traffic and singles out the specific type of traffic to be mimicked. The more specific application traffic the filter can identify and profile, the better model we can have for generating covert traffic. For example, FTP is an application protocol based on TCP, but generating a series of inter-packet delays based on a model of all TCP traffic would be a poor model for describing FTP behaviors. Once the specified traffic is filtered, the traffic is further classified into individual flows based on source and destination IP addresses. The filter then calculates the inter-packet delay between subsequent pair of packets from each flow, and forwards the results to the analyzer.

Table 1. The scores for different models for a sample of HTTP inter-packet delays

model	parameters	root mean squared error
Weibull	0.0794, 0.2627	0.0032
Gamma	0.1167, 100.8180	0.0063
Lognormal	-4.3589, 3.5359	0.0063
Pareto	3.6751, 0.0018	0.0150
Poisson	11.7611	0.0226
Exponential	11.7611	0.0294

The analyzer fits the inter-packet delays in sets of 100 packets with the Exponential, Gamma, Pareto, Lognormal, Poisson, and Weibull distributions. The fitting process uses maximum likelihood estimation (MLE) to determine the parameters for each model. The model with the smallest root mean squared error (RMSE), which measures the difference between the model and the estimated distribution, is chosen as the traffic model. The model selection is automated. Other than the set of models provided to the analyzer, there is no human input. The models are scored based on root mean squared errors, as shown in Table 1. The model with the lowest root mean squared error is the closest to the data being modeled. Since most types of network traffic are non-stationary [22], the analyzer supports piecewise modeling of non-stationary processes by adjusting the parameters of the model after each set of 100 covert inter-packet delays. The analyzer refits the current model with new sets of 100 packets to adjust the parameters. The analyzer can take advantage of a larger selection of models to more accurately model different types of application traffic. For example, if we know that the targeted traffic is well-modeled as an Erlang distribution, we will add this distribution to the set of models. For each of the current models, the computational overhead is less than 0.1 milliseconds and the storage overhead for the executable is less than 500 bytes, so the induced resource consumption for supporting additional models is not an issue.

The filter and analyzer can be run either offline or online. In the offline mode, the selection of the model and parameters is based on traffic samples. The offline mode consumes less resources, but the model might not represent the current network traffic behavior well. In the online mode, the selection of the model and parameters is based on live traffic. The online mode consumes more resources and requires that the model and parameters be transmitted to the decoder with the support of a startup protocol, but the model better represents the current network traffic behavior. The startup protocol is a model determined in advance, and is used to transmit the online model (1 byte) and parameters (4-8 bytes) to the decoder.

The encoder generates random covert inter-packet delays that mimic legitimate inter-packet delays. The input to the encoder includes the model, the message, and a sequence of random numbers. Its output is a sequence of covert random inter-packet delays. The message to be sent is separated into symbols.

The symbols map to different random timing values based on a random code that distributes symbols based on the model.

Using a sequence of random numbers r_1, r_2, ..., r_n., we transform the discrete symbols into continuous ones. The continuization function is

$$F_{continuize}(s) = (\frac{s}{\mid S \mid} + r) \bmod 1 = r_s,$$

where S is the set of possible symbols, s is a symbol and r is a Uniform(0,1) random variable. The corresponding discretization function is:

$$F_{discretize}(r_s) = \mid S \mid \cdot ((r_s - r) \bmod 1) = s,$$

where r_s is a Uniform(0,1) random variable with a hidden symbol s.

The encoder and decoder start with the same seed and generate the same sequence of random numbers, r_1, r_2, ..., r_n. To maintain synchronization, the encoder and decoder associate the sequence of symbols with TCP sequence numbers, i.e., s_1 with the first TCP sequence number, s_2 with the second TCP sequence number, and so on. [3] Therefore, both the encoder and decoder have the same values of r through the sequence of symbols. The inverse distribution function F^{-1}_{model} takes a Uniform(0,1) random number as input and generates a random variable from the selected model as output. The sequence of transformed random numbers r_{s1}, r_{s2}, ..., r_{sn} is used with the inverse distribution function to create random covert inter-packet delays d_{s1}, d_{s2}, ..., d_{sn}. The encode function is:

$$F_{encode} = F^{-1}_{model}(r_s) = d_s,$$

where F^{-1}_{model} is the inverse distribution function of the selected model. The decode function is:

$$F_{decode} = F_{model}(d_s) = r_s,$$

where F_{model} is the cumulative distribution function of the selected model, and d_s is a random covert inter-packet delay with a hidden symbol s.

The transmitter sends out packets to produce the random covert inter-packet delays d_{s1}, d_{s2}, ..., d_{sn}. The receiver then decodes and discretizes them to recover the original symbols s_1, s_2, ..., s_n.

4.1 Model-Based Channel Capacity

The model-based channel capacity is also dependent on the input distribution and noise. The input distribution is defined as:

$$P(x) = f_{model}(x)$$

where f_{model} is the probability density function of the selected model.

[3] With this mechanism, repacketization can cause synchronization problems, so other mechanisms such as "bit stuffing" [12] could be useful for synchronization.

Therefore, the capacity of the model-based channel is the mutual information over the expected time $E(X)$:

$$C_t = \frac{1}{E(X)} \int\limits_X \int\limits_Y P(y \mid x)P(x)\log\frac{P(y \mid x)P(x)}{P(x)P(y)}.$$

4.2 Implementation Details

We implement the proposed framework using `C` and `MATLAB` in Unix/Linux environments. The components run as user-space processes, while access to `tcpdump` is required. The filter is written in C and runs `tcpdump` with a user-specified filtering expression to read the stream of packets. The filter processes the traffic stream and computes the inter-packet delays based on the packet timestamps. The analyzer is written in `MATLAB` and utilizes the fitting functions from the statistics toolbox for maximum likelihood estimation.

The encoder is written in `C`, and uses random number generation and random variable models from the Park-Leemis [23] simulation C libraries. The transmitter is also written in `C`, with some inline assembly, and uses the Socket API. The timing mechanism used is the Pentium CPU Time-Stamp Counter, which is accessed by calling the `RDTSC` (Read Time-Stamp Counter) instruction. The `RDTSC` instruction has excellent resolution and low overhead, but must be calibrated to be used as a general purpose timing mechanism. The `usleep` and `nanosleep` functions force a context switch, which delays the packet transmission with small inter-packet delays, so these functions are not used.

5 Experimental Evaluation

In this section, we evaluate the effectiveness of a model-based covert timing channel built from our framework. The OPC and FPR covert timing channels, discussed in Section 3, are used as points of comparison. In particular, we examine the capacity and detection resistance of each covert timing channel.

5.1 Experimental Setup

The defensive perimeter of a network, composed of firewalls and intrusion detection systems, is responsible for protecting the network. Typically, only a few specific application protocols, such as HTTP and SMTP, are commonly allowed to pass through the defensive perimeter. We utilize outgoing HTTP inter-packet delays as the medium to build model-based covert timing channels, due to the wide acceptance of HTTP traffic for crossing the network perimeter. We refer to the model-based HTTP covert timing channel as MB-HTTP.

Testing Scenarios There are three different testing scenarios in our experimental evaluation. The first scenario is in a LAN environment, a medium-size campus network with subnets for administration, departments, and residences. The LAN connection is between two machines, located in different subnets. The connection passes through several switches, the routers inside the campus network, and a firewall device that protects each subnet.

The other two scenarios are in WAN environments. The first WAN connection is between two machines, both are on the east coast of the United States but in different states. One is on a residential cable network and the other is on a medium-size campus network. The second WAN connection is between two machines on the opposite coasts of the United States, one on the east coast and the other on the west coast. Both machines are on campus networks.

Table 2. The network conditions of each test scenario

	LAN	WAN E-E	WAN E-W
distance	0.3 miles	525 miles	2660 miles
RTT	1.766ms	59.647ms	87.236ms
IPDV	2.5822e-05	2.4124e-03	2.1771e-04
hops	3	18	13
IPDV - inter-packet delay variation			

The network conditions for different experiment scenarios are summarized in Table 2. The two-way round-trip time (RTT) is measured using the ping command. We compute the one-way inter-packet delay variation based on the delays between packets leaving the source and arriving at the destination. The inter-packet delay variations of the three connections span three orders of magnitude, from 1×10^{-3} to 1×10^{-5}. The LAN connection has the lowest inter-packet delay variation and the two WAN connections have higher inter-packet delay variation, as expected. The WAN E-E connection is shorter and has smaller RTT time than the WAN E-W connection. However, WAN E-E has higher inter-packet delay variation than WAN E-W, due to more traversed hops. This implies that the inter-packet delays variation is more sensitive to the number of hops than the physical distance and RTT between two machines.

Building MB-HTTP We install the components of the framework on the testing machines. The filter distinguishes the outgoing HTTP traffic from background traffic. The analyzer observes 10 million HTTP inter-packet delays, then fits the HTTP inter-packet delays to the models, as described in Section 4. The fitting functions use maximum likelihood estimation (MLE) to determine the parameters for each model. The model with the best root mean squared error (RMSE), a measure of the difference between the model and the distribution being estimated, is chosen as the traffic model.

For the HTTP inter-packet delays, the analyzer selects the Weibull distribution based on the root mean squared error. Note that HTTP inter-packet delays have been shown to be well approximated by a Weibull distribution [22]. The Weibull probability distribution function is:

$$f(x, \lambda, k) = \frac{k}{\lambda}(\frac{x}{\lambda})^{(k-1)} e^{-(\frac{x}{\lambda})^k}.$$

The parameters, which vary for each set of 100 packets, have a mean scale parameter λ of 0.0371 and a mean shape parameter k of 0.3010. With these parameters, the mean inter-packet delay is 0.3385, approximately 3 packets per second.

Table 3. The mean packets per second and mean inter-packet delay for OPC

	LAN		WAN E-E		WAN E-W	
channel	PPS	IPD	PPS	IPD	PPS	IPD
OPC	12,777.98	7.87e-05	137.48	7.31e-03	1,515.56	6.63e-04
PPS - mean packets per second, IPD - mean inter-packet delay						

Formulating OPC and FPR The average packet rate for FPR is fixed at $\frac{1}{0.3385} = 2.954$ packets per second, based on the average packet rate of HTTP traffic. We use the Arimoto-Blahut algorithm to compute the optimal input distribution, with the average packet rate of 2.954 as the cost constraint. The optimal input distribution balances high cost symbols with low probabilities and low cost symbols with high probabilities, such that the average cost constraint is satisfied. The constraint can be satisfied for infinitely large symbols with infinitely small probabilities, and hence, the optimal input distribution decays exponentially to infinity. The results of the Arimoto-Blahut algorithm, as the number of intervals increases, reduce to an Exponential distribution with an inverse scale parameter of $\lambda = 2.954$. The Exponential probability distribution function is:

$$f(x, \lambda) = \lambda e^{-\lambda x}.$$

We compute the optimal distance between packets for OPC based on the noise distribution. The optimal distance between packets and the average packet rate for OPC is shown in Table 3. For connections with higher inter-packet delay variation, OPC increases the time elapse between packets to make the inter-packet delays easier to distinguish, and, as a result, lowers the average number of packets per second.

5.2 Capacity

The definition of capacity allows us to estimate the capacity of each covert timing channel based on the network conditions of each connection. In previous research [24], the inter-packet delay differences have been shown to be well-modeled by a Laplace distribution. The probability density function of the Laplace distribution is:

$$f(x, \mu, b) = \frac{1}{2b} e^{-\frac{|x-\mu|}{b}}.$$

The setting of the scale parameter b is based on the inter-packet delay variation for each connection. The variation of the Laplace distribution is $\sigma^2 = 2b^2$. Therefore, we set b to:

$$b = \sqrt{\frac{1}{2}\sigma^2},$$

where σ^2 is the inter-packet delay variation for each connection.

Table 4. The theoretical capacity of each covert timing channel

	LAN		WAN E-E		WAN E-W	
channel	CPP	CPS	CPP	CPS	CPP	CPS
MB-HTTP	9.39	27.76	4.12	12.19	6.84	20.21
FPR	12.63	37.32	6.15	18.17	9.59	28.35
OPC	0.50	6395.39	0.50	68.80	0.50	758.54
CPP - capacity per packet, CPS - capacity per second						

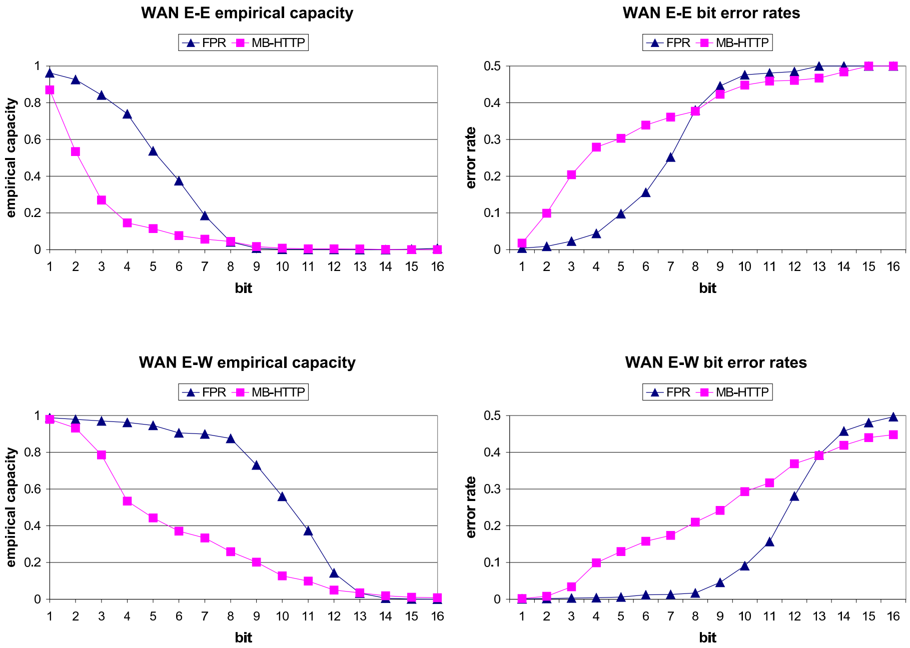
The results, in terms of capacity per packet and capacity per second, are shown in Table 4. While OPC has the highest capacity, it is the least efficient in terms of capacity per packet. Furthermore, with the large number of packets per second, it can be easily detected by most intrusion detection systems.

The capacity of MB-HTTP is 67% to 74% of that of FPR, with larger differences for connections with high inter-packet delay variation than for those with low inter-packet delay variation. The Weibull distribution has a larger proportion of very small values than the Exponential distribution. As a result, MB-HTTP uses more small values than FPR and benefits more from lower inter-packet delay variation.

The theoretical capacity is somewhat optimistic. The model only considers the noise introduced after packets leave the transmitter. With the real covert timing channels, noise is introduced before packets leave the transmitter. The transmitter is sometimes not able to transmit at the appropriate times, due to slow processing, context switches, etc. Thus, the actual distance between packets can increase or decrease from the intended distance as the packets are transmitted.

Empirical Capacity To evaluate the channel capacity in practice, we run covert timing channels on each connection. The channels are configured to transmit 16,000 random bits of information. For FPR and MB-HTTP, the number of

Fig. 2. The empirical capacity and bit error rates for WAN E-E and WAN E-W



bits encoded per packet is set to 16 (i.e., $2^{16} = 65,536$ different values), while OPC transmits a single bit per packet.

During these tests, we measure the bit error rate of each covert timing channel from the most significant bit to the least significant bit of each packet. The most significant bit represents a large part of the inter-packet delay, where the least significant bit represents a small part of the inter-packet delay. While flipping the most significant bit causes a difference in seconds or tenths of seconds, changing the least significant bit means a difference only in milliseconds or microseconds. In other words, the higher the number of bits encoded per packet, the smaller the precision of the lowest order bits. Interestingly, encoding at 16 bits per packet and decoding at 8 bits per packet produces the most significant 8 bits of the 16 bit code.

To determine the transmission rate with error correction, we measure the empirical capacity of each bit as a binary symmetric channel. The binary symmetric channel is a special case where the channel has two symbols of equal probability. The capacity of a binary symmetric channel is:

$$C = I(X;Y) = 1 - (p\,\log p + q\,\log q),$$

where p is the probability of a correct bit and $q = 1 - p$ is the probability of an incorrect bit.

The empirical capacity and bit error rate for each bit, from the most significant to the least significant, are shown in Figure 2. The empirical capacity per bit degrades as the bit error rates increase. The total capacity of the channel is the summation of the capacity for each bit. For MB-HTTP, the bit error rate increases somewhat linearly. For FPR, the bit error rate accelerates gradually, eventually overtaking the bit error rates of MB-HTTP, though at this point the capacity per bit is insignificant.

Table 5. The empirical capacity of each covert timing channel

	LAN		WAN E-E		WAN E-W	
channel	ECPP	ECPS	ECPP	ECPS	ECPP	ECPS
MB-HTTP	6.74	19.93	2.15	6.35	5.18	15.31
FPR	10.95	32.35	4.63	13.67	9.37	27.69
OPC	0.85	10,899.62	0.66	91.28	0.98	1,512.53
ECPP - empirical capacity per packet, ECPS - empirical capacity per second						

The empirical capacity of each covert timing channel is shown in Table 5. The empirical capacity of MB-HTTP is still about 46% to 61% of that of FPR, somewhat lower than the case in the theoretical model. This is because a larger proportion of MB-HTTP traffic has small inter-packet delays than that of FPR, and small inter-packet delays are more sensitive to noise caused by transmission delays (i.e., slow processing, context switches, etc.) than large inter-packet delays, which is not represented in the theoretical model.

5.3 Detection Resistance

The detection resistance, as described in Section 3, is estimated based on the shape and regularity tests. To examine the shape of the distribution, we use the Kolmogorov-Smirnov test [25], which is a non-parametric goodness-of-fit test. To examine the regularity of the traffic, we use the regularity test [11], which studies the variance of the traffic pattern. In this section, we detail these two tests and show the detection resistance of MB-HTTP against both tests.

Shape Tests The two-sample Kolmogorov-Smirnov test determines whether or not two samples come from the same distribution. The Kolmogorov-Smirnov test is distribution free, meaning the test is not dependent on a specific distribution. Thus, it is applicable to a variety of types of traffic with different distributions. The Kolmogorov-Smirnov test statistic measures the maximum distance between two empirical distribution functions.

$$KSTEST = \max \mid S_1(x) - S_2(x) \mid,$$

where S_1 and S_2 are the empirical distribution functions of the two samples.

In our experiments, we test a large set of legitimate inter-packet delays against a sample of either covert or legitimate inter-packet delays. The large set is a training set of 10,000,000 HTTP inter-packet delays. The training set is used to represent the normal behavior of the HTTP protocol.

The test score by comparing the two sets is used to determine if the sample is covert or legitimate. A small score indicates that the behavior is close to normal. However, if the test score is large, i.e., the sample does not fit the normal behavior of the protocol, it indicates a potential covert timing channel.

Table 6. The mean and standard deviation of the Kolmogorov-Smirnov test scores

	LEGIT-HTTP		MB-HTTP		FPR		OPC	
sample size	mean	stdev	mean	stdev	mean	stdev	mean	stdev
100x 2,000	.193	.110	.196	.093	.925	.002	.999	.000
100x 10,000	.141	.103	.157	.087	.925	.001	.999	.000
100x 50,000	.096	.088	.122	.073	.924	.000	.999	.000
100x 250,000	.069	.066	.096	.036	.924	.000	.999	.000

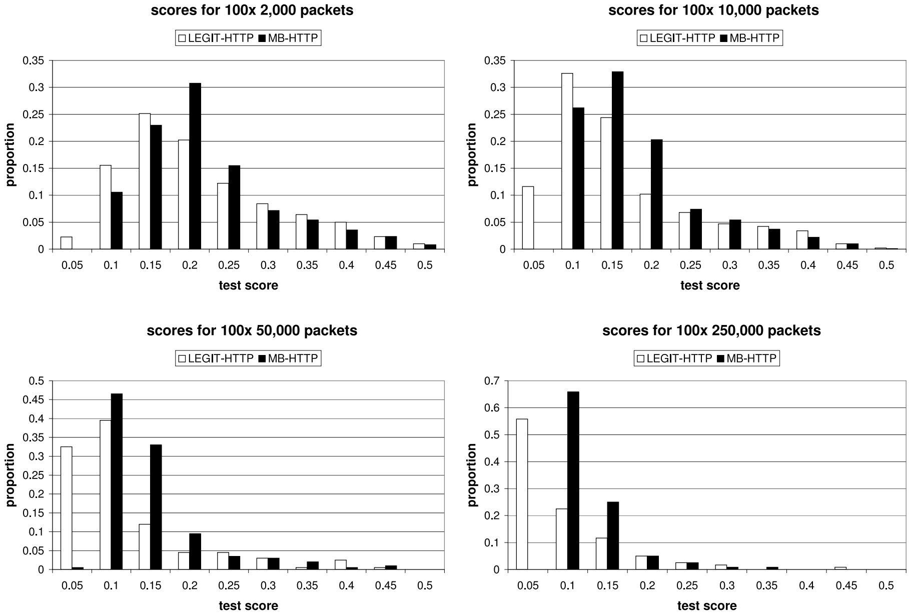
The Kolmogorov-Smirnov test is run 100 times for each of 2,000, 10,000, 50,000, and 250,000 packet samples of legitimate and covert traffic from each covert timing channel. The mean and standard deviation of the test scores are shown in Table 6. For FPR and OPC, the mean scores are over 0.90 and the standard deviations are extremely low, indicating that the test can reliably differentiate both covert timing channels from normal HTTP traffic. By contrast, the mean scores for MB-HTTP samples are very close to those of legitimate samples. The mean scores are for 100 tests, which in total include as many as 25 million (250,000 x 100) inter-packet delays. The distribution of individual test scores is illustrated in Figure 3.

Table 7. The false positive and true positive rates for the Kolmogorov-Smirnov test

		LEGIT-HTTP	MB-HTTP	FPR	OPC
sample size	cutoff	false pos.	true pos.	true pos.	true pos.
100x 2,000	$KSTEST \geq .66$	.01	.01	1.00	1.00
100x 10,000	$KSTEST \geq .65$	.01	.01	1.00	1.00
100x 50,000	$KSTEST \geq .41$	.01	.01	1.00	1.00
100x 250,000	$KSTEST \geq .21$	.01	.02	1.00	1.00

The detection resistance based on the Kolmogorov-Smirnov test is shown in Table 7. The targeted false positive rate is 0.01. To achieve this false positive rate, the cutoff scores—the scores that decide whether samples are legitimate or covert—are set at the 99th percentile of legitimate sample scores. The true positive rates, based on the cutoff scores, are then shown for each covert timing channel. Since the true positive rates in all 100 tests are 1.00,

Fig. 3. The distribution of Kolmogorov-Smirnov test scores



the Kolmogorov-Smirnov test detects FPR and OPC easily. However, the true positive rates for MB-HTTP are approximately the same as the false positive rates. The Kolmogorov-Smirnov test cannot differentiate between MB-HTTP and legitimate samples. Such a result can be explained based on the distribution of individual test scores, which is shown in Figure 3. While the mean scores of MB-HTTP traffic in Table 6 are slightly higher than those of LEGIT-HTTP, the distributions of individual scores overlap so that the false positive rate and true positive rate are approximately equal.

Regularity Tests The regularity test [11] determines whether the variance of the inter-packet delays is relatively constant or not. This test is based on the observation that for most types of network traffic, the variance of the inter-packet delays changes over time. With covert timing channels, the code used to transmit data is a regular process and, as a result, the variance of the inter-packet delays remains relatively constant over time.

In our experiments, we test the regularity of a sample of either covert or legitimate inter-packet delays. The sample is separated into sets of w inter-packet delays. Then, for each set, the standard deviation of the set σ_i is computed. The regularity is the standard deviation of the pairwise differences between each σ_i

and σ_j for all sets $i < j$.

$$regularity = STDEV(\frac{\mid \sigma_i - \sigma_j \mid}{\sigma_i}, i < j, \forall i, j)$$

The regularity test is run 100 times for 2,000 packet samples of legitimate and covert samples from each covert timing channel. The window sizes of $w = 100$ and $w = 250$ are used. The mean regularity scores are shown in Table 8. If the regularity is small, the sample is highly regular, indicating a potential covert timing channel.

Table 8. The mean of the regularity test scores

sample size	LEGIT-HTTP	MB-HTTP	FPR	OPC
100x 2,000 w=100	43.80	38.21	0.34	0.00
100x 2,000 w=250	23.74	22.87	0.26	0.00

The mean regularity scores for OPC are 0.0 for both tests, indicating regular behavior. There are two values, each with 0.5 probability. Therefore, the standard deviation within sets is small $\sigma = 0.5d = 3.317e - 4$, and there is no detectable change in the standard deviation between sets. The mean regularity score for FPR is small as well, showing that the test is able to detect the regular behavior. While the standard deviation of FPR, which is based on the Exponential distribution, is $\sigma = \lambda = 0.3385$, the code is a regular process, so the variance of the inter-packet delays remains relatively constant.

The mean regularity scores for MB-HTTP are close to those of legitimate samples. This is because the parameters are recalibrated after each set of 100 packets, as described in Section 4. The parameters of the distribution determine the mean and standard deviation, so adjusting the parameters changes the variance after each set of 100 inter-packet delays. As a result, like legitimate traffic, the variance of the inter-packet delays appears irregular.

Table 9. The false positive and true positive rates for the regularity test

		LEGIT-HTTP	MB-HTTP	FPR	OPC
sample size	cutoff	false pos.	true pos.	true pos.	true pos.
100x 2,000 w=100	$reg. \leq 6.90$	.01	.00	1.00	1.00
100x 2,000 w=250	$reg. \leq 5.20$	.01	.00	1.00	1.00

The detection resistance based on the regularity test is shown in Table 9. The targeted false positive rate is 0.01. The cutoff scores are set at the 1st percentile of legitimate sample scores, in order to achieve this false positive rate. The true positive rates, based on the cutoff scores, are then shown for each covert timing channels. The regularity test is able to detect FPR and OPC in all 100 tests.

The resulting true positive rates for MB-HTTP are approximately the same as the false positive rate. Basically, the test is no better than random guessing at detecting MB-HTTP.

6 Conclusion

We introduced model-based covert timing channels, which mimic the observed behavior of legitimate network traffic to evade detection. We presented a framework for building such model-based covert timing channels. The framework consists of four components: filter, analyzer, encoder, and transmitter. The filter characterizes the specific features of legitimate traffic that are of interest. The analyzer fits the traffic to several models and selects the model with the best fit. The encoder generates random covert inter-packet delays that, based on the model, mimic the legitimate traffic. The transmitter then manipulates the timing of packets to create the model-based covert timing channel.

Using channel capacity and detection resistance as major metrics, we evaluated the proposed framework in both LAN and WAN environments. Our capacity results suggest that model-based covert timing channels work efficiently even in the coast-to-coast scenario. Our detection resistance results show that, for both shape and regularity tests, covert traffic is sufficiently similar to legitimate traffic that current detection methods cannot differentiate them. In contrast, the Kolmogorov-Smirnov and regularity tests easily detect FPR and OPC.

Our future work will further explore the detection of model-based covert timing channels. There are other non-parametric goodness-of-fit tests, such as the Anderson-Darling and Cramer-Von Mises tests [25], that are less general than the Kolmogorov-Smirnov test but might be more effective in measuring certain types of traffic. We will also further consider the regularity test at different levels of granularity. We believe that a scheme capable of detecting model-based covert timing channels will be effective in detecting other types of covert timing channels as well.

Acknowledgments

We would like to thank Cheng Jin and Lachlan Andrew at CalTech for assisting us in the coast-to-coast experiments. We also thank the anonymous reviewers for their insightful comments. This work was partially supported by NSF grants CNS-0627340 and CNS-0627493.

References

1. Department of Defense, U.S.: Trusted computer system evaluation criteria (1985)
2. Lampson, B.W.: A note on the confinement problem. Communications of the ACM **16**(10) (Oct. 1973)
3. Wang, Z., Lee, R.: Covert and side channels due to processor architecture. In: Proc. of ACSAC 2006. (Dec. 2006)

4. Fisk, G., Fisk, M., Papadopoulos, C., Neil, J.: Eliminating steganography in internet traffic with active wardens. In: Proc. of the 2002 International Workshop on Information Hiding. (Oct. 2002)
5. Kang, M.H., Moskowitz, I.S.: A pump for rapid, reliable, secure communication. In: Proc. of ACM CCS 1993. (Nov. 1993)
6. Kang, M.H., Moskowitz, I.S., Lee, D.C.: A network version of the pump. In: Proc. of the 1995 IEEE Symposium on Security and Privacy. (May 1995)
7. Kang, M.H., Moskowitz, I.S., Chincheck, S.: The pump: A decade of covert fun. In: Proc. of ACSAC 2005. (Dec. 2005)
8. Giles, J., Hajek, B.: An information-theoretic and game-theoretic study of timing channels. IEEE Trans. on Information Theory **48**(9) (Sep. 2002)
9. Berk, V., Giani, A., Cybenko, G.: Covert channel detection using process query systems. In: Proc. of FLOCON 2005. (Sep. 2005)
10. Berk, V., Giani, A., Cybenko, G.: Detection of covert channel encoding in network packet delays. Technical Report TR2005-536, Department of Computer Science, Dartmouth College, Hanover, NH., USA (Aug. 2005)
11. Cabuk, S., Brodley, C., Shields, C.: IP covert timing channels: Design and detection. In: Proc. of ACM CCS 2004. (Oct. 2004)
12. Shah, G., Molina, A., Blaze, M.: Keyboards and covert channels. In: Proc. of the 2006 USENIX Security Symposium. (July–Aug. 2006)
13. Gianvecchio, S., Wang, H.: Detecting covert timing channels: An entropy-based approach. In: Proceedings of the 2007 ACM Conference on Computer and Communications Security. (October 2007)
14. Luo, X., Chan, E.W.W., Chang, R.K.C.: Cloak: A ten-fold way for reliable covert communications. In: Proc. of ESORICS. (Sept. 2007)
15. Arimoto, S.: An algorithm for computing the capacity of arbitrary discrete memoryless channels. IEEE Trans. on Information Theory **18**(1) (Jan. 1972)
16. Blahut, R.E.: Computation of channel capacity and rate-distortion functions. IEEE Trans. on Information Theory **18**(4) (July 1972)
17. Borders, K., Prakash, A.: Web tap: Detecting covert web traffic. In: Proc. of ACM CCS 2004. (Oct. 2004)
18. Wang, X., Reeves, D.S.: Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays. In: Proc. of ACM CCS 2003. (Oct. 2003)
19. Yu, W., Fu, X., Graham, S., Xuan, D., Zhao, W.: Dsss-based flow marking technique for invisible traceback. In: Proc. of the 2007 IEEE Symposium on Security and Privacy, Washington, DC, USA (May 2007)
20. Peng, P., Ning, P., Reeves, D.S.: On the secrecy of timing-based active watermarking trace-back techniques. In: Proc. of the 2006 IEEE Symposium on Security and Privacy. (May 2006)
21. Moskowitz, I.S., Kang, M.H.: Covert channels - here to stay? In: Proc. of the 1994 Annual Conf. on Computer Assurance. (June 1994)
22. Cao, J., Cleveland, W.S., Lin, D., Sun, D.X.: On the nonstationarity of internet traffic. In: Proc. of SIGMETRICS/Performance 2001. (June 2001)
23. Leemis, L., Park, S.K.: Discrete-Event Simulation: A First Course. Prentice-Hall, Upper Saddle River, NJ., USA (2006)
24. Zheng, L., Zhang, L., Xu, D.: Characteristics of network delay and delay jitter and its effect on oice over IP (VoIP). In: Proc. of the 2001 IEEE International Conf. on Communications. (June 2001)
25. Duda, R., Hart, P., Stork, D.: Pattern Classification. Wiley-Interscience, New York, NY., USA (2001)