IP Covert Timing Channels: Design and Detection

Serdar Cabuk
Electrical and Computer Engineering
Purdue University
scabuk@ecn.purdue.edu

Carla E. Brodley
Department of Computer Science
Tufts University
brodley@cs.tufts.edu

Clay Shields
Department of Computer Science
Georgetown University
clay@cs.georgetown.edu

ABSTRACT

A network covert channel is a mechanism that can be used to leak information across a network in violation of a security policy and in a manner that can be difficult to detect. In this paper, we describe our implementation of a covert network timing channel, discuss the subtle issues that arose in its design, and present performance data for the channel. We then use our implementation as the basis for our experiments in its detection. We show that the regularity of a timing channel can be used to differentiate it from other traffic and present two methods of doing so and measures of their efficiency. We also investigate mechanisms that attackers might use to disrupt the regularity of the timing channel, and demonstrate methods of detection that are effective against them.

Categories and Subject Descriptors

C.2.0 [**Computer-Communication Networks**]: General—*Security and Protection*;
D.4.6 [**Security and Protection**]: [Information flow controls];
K.6.5 [**Security and Protection**]: [Unauthorized access]

General Terms

Security

Keywords

Network covert channels, TCP/IP, covert timing channels, detection

1. INTRODUCTION

A *covert channel* is a mechanism that can be used to violate a security policy by allowing information to leak to an unauthorized process [14]. Two types of covert channels exist: storage and timing channels. A *storage channel* "involves the direct or indirect writing of a storage location by one process and the direct or indirect reading of the storage location by another process" [25]. A *timing channel* involves a sender process that "signals information to another by modulating its own use of system resources (e.g., CPU time) in such a way that this manipulation affects the real response time observed by the second process" [25]. This classification can be taken further by identifying *hybrid channels* in which the timing and storage information are used together, and *counting channels* [19] in which the number of events come into play instead of the occurrence of a single event.

Detecting and preventing covert channels is particularly important for multi-level security (MLS) systems in which processes working with classified information may leak information to processes with a lower classification level via the use of shared resources [23]. Indeed, the evaluation criteria for trusted computer systems includes the requirement to analyze covert channels [25] in terms of their bandwidth and to develop policies to monitor and maintain their bandwidth below maximum acceptable levels. In this paper, we focus on the analysis and detection of covert timing channels in the TCP/IP protocol suite. Although some work has been done on timing channel analysis in general, little attention has been paid to channels in IP. Note that the Trusted Computer System Evaluation Criteria (TCSEC [25]) requires storage channel analysis for a class B2 system, and timing channel analysis for higher classes.

In this initial exploration, we first present a design of an IP timing channel and provide the details of its implementation. While simple in concept, there proved to be some non-obvious issues in designing the software. We then look at the detection problem and present a set of methods for detecting IP timing channels based on analysis of traffic flows.

In the following section, we provide background information on covert channels. We present our IP covert timing channel design and implementation in Section 3 and point out some difficulties in implementing synchronous timing channels in asynchronous environments where no global reference clock exists. We present the results of an empirical study evaluating the performance of our channel. In Section 4 we present our proposed detection method and an empirical evaluation of its ability to detect IP timing channels. We conclude with directions for future work in Section 5.

2. NETWORK COVERT CHANNELS

While initial research in covert channels focused on single systems [23, 27, 32, 34], our focus here is on network

CCS'04, October 25-29, 2004, Washington, DC, USA.

covert channels, which have become a pervasive security threat to trusted distributed systems. Network covert channels have been used by attackers to communicate with compromised hosts, particularly in distributed denial of service attacks [18]. Many tools exist for setting up network covert channels using a variety of protocols including TCP, IP, HTTP and ICMP [9, 10, 29, 31].

The data section of packets is the easiest place to convey covert information, due to its large size and because it is relatively unstructured compared to headers. Modifying the packet payload is outside the scope of this paper as it falls in the realm of steganography or watermarking. Our focus in this section is instead on storage channels in the packet headers and on timing channels.

Unused header fields that are either designed for future protocol improvements or in general go unchecked by firewalls and network intrusion detection devices, may convey information in the form of a covert channel [2, 3, 10, 12, 29, 30]. The ID field (for unfragmented packets) in TCP and the option bits in IP have been used for storage channels [12]. A smart attacker can even devise means to use some of the header fields that *do* fall under scrutiny, such as the IP checksum field [1]. An effective way to eliminate most storage channels is through traffic normalizers [11, 17], which modify both incoming and outgoing packets by standardizing fields that are unused or redundant. Unusual traffic patterns may also lead to discovery of storage channels. For example, multiple ping requests within a small time interval may indicate a storage channel in the ICMP protocol such as that used by Loki [9]. In addition, covert storage channels can sometimes be detected by observing variations in unused packet header fields [17].

Less attention has been placed on *network timing channels*. These channels convey information through the arrival patterns of packets, rather than through the contents of the packets themselves. Network timing channels include packet sorting channels [2, 3], in which the order of packet arrival conveys information, and timing channels in which it is the reception or absence of packets within specific time intervals that carries significance. In our research we have focused on the latter type of timing channel.

To understand how these channels work, consider a distributed MLS system which uses the TCP/IP protocol suite to provide the necessary communication between remote users of the system. For the sake of simplicity, we will assume that the two parties have information access levels of HIGH and LOW. We assume that the system is capable of securing all overt communication and further mechanisms such as a *packet sanitizer* are also employed, which remove all sensitive data from the message content when data is transferred from HIGH to LOW security levels. Our research addresses two questions: How can information be leaked using IP from a HIGH node to a LOW node? How can the system detect such leakage?

In terms of a client/server architecture, the covert channel can be set to leak information in either direction: server to client or client to server. In the first case, the server resides on a HIGH node running a form of malware. The client initiates the covert communication by a *connect* request over a known port (e.g., FTP). The trojaned server recognizes the IP address of the client, and begins the covert communication. Note that the server exhibits normal behavior on connection requests from all other clients. In the second

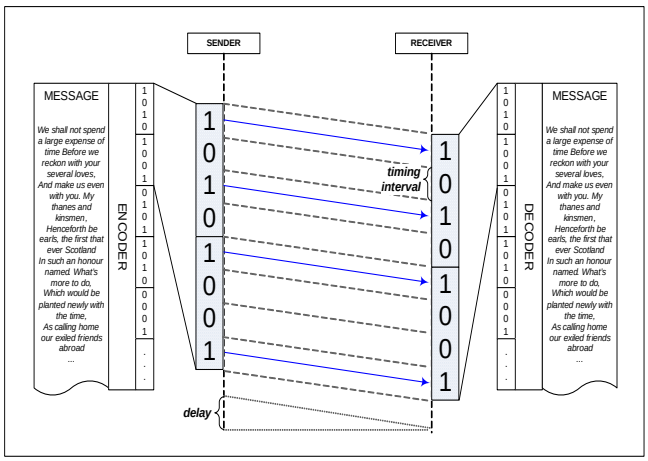

Figure 1: IP covert timing channel. The example text is first encoded with a coding scheme and then bit by bit sent to the receiving end. The message is rebuilt by decoding the bit stream.

case, malware in a client on a HIGH node initiates the connection. In this case, the server's IP address is known to the malware. The server responds and the covert communication is started, this time from client to server. Given our implementation experience (see Section 3), we conjecture that fewer hacker tools use timing channels because of the difficulties in synchronizing such channels and because of their reduced bandwidth as compared to storage channels.

Network implementations of the pump [20] as well as timing jammers [13], which act as intermediaries between networks and modify packet inter-arrival times, are the principal defenses against timing channels. These defenses are aimed at stopping such channels rather than detecting them. An attacker who is aware of the existence of such countermeasures may intentionally decrease the bandwidth of the covert channel, reducing the effect of fluctuations in packet inter-arrival times on message accuracy. This ensures that the introduced timing discrepancies will be small compared to the length of each timing interval. Detection may also be more desirable than stopping covert channels because of the added benefits of locating compromised internal hosts as well as in blacklisting external IP addresses that are found to participate in the covert communication. Consequently, the focus of our research is to detect network timing channels.

3. IP COVERT TIMING CHANNEL IMPLEMENTATION

In a timing channel, the receiver and sender agree a priori on a timing interval and the starting protocol (either a particular time or in response to a network event, such as the first packet sent). During each time interval the sender either transmits a single packet or maintains silence. The receiver monitors each interval to determine whether a packet was received or not. The result is a binary code where a 1 represents the detection of packet in an interval and a 0 represents the absence of a packet (see Figure 1). Note that the raw data that flows across the channel is binary but the actual interpretation of the binary stream is up to the

communicating parties.

Regardless of whether the binary stream is the data itself or whether it represents a message, additional bits are usually included in the transmission for three reasons. Firstly, additional parity bits may be appended to the data to add redundancy for error correction due to transmission errors (e.g., errors arising when a packet is lost). Secondly, additional bits may be added for purposes of maintaining synchronization between sender and receiver. Finally, the data may be encrypted in order to add a further layer of privacy and obfuscation. This third issue is beyond the scope of this paper because our detection schemes are concerned with detecting only the presence of a covert channel, and are not designed to infer the content of such channels.

The message for transmission is subdivided into smaller blocks of binary data, referred to as *frames* in this paper. An example frame consists of data bits, synchronization bits, and error-correcting bits. While all the frames are of equal length, the actual length, as well as the interval between frames, is influenced by parameters of the encoding scheme and the network. This is further examined in Section 3.4, where we look at synchronization issues. Note that although one can employ error-correcting code bits, we have not included this option in our initial implementation.

The IP covert timing channel can be configured to run on any application port. Because the traffic pattern is expected to vary based on the application, choice of the protocol in which to hide the channel can affect detection ability. Indeed, we illustrate this empirically in Section 4.

In our experiments we have assumed a unidirectional communication model for the covert channel. Note that only the covert communication is assumed to be unidirectional; the communication itself is still bidirectional and the TCP/IP packets are ACKed. Assuming a unidirectional channel means that the receiver side cannot communicate with the sender of the covert channel using the covert channel itself. Restricting the channel to be unidirectional increases the difficulty in implementing an error-free channel. In particular, the receiver cannot 1) acknowledge the correct receipt of covert packets, 2) rate limit the sender, or 3) indicate when to resynchronize. We also assume that both the sender and receiver have the software to send/receive the covert channel packets.

3.1 Performance Factors

Several factors impact the performance of a network timing covert channel.

Network conditions: The channel performance is directly affected by the network conditions between the communicating parties. During the peak hours, when a congestion is likely, IP packets are more likely to be delayed, arrive out of order, or be lost during transmission. Additionally, *jitter*, which is the variability in round trip times (RTT) can cause synchronization problems.

Sender/receiver processing capabilities: The sender and receiver processing units may be congested under heavy load (e.g., a web server observing high traffic at peak times). Under these conditions, the processing of the packets may be delayed. The bottleneck could be either the network processing card or any other busy resource that might delay packet processing.

The complexity of the algorithms: The algorithms used in designing the communication channel should be efficient. In our experiments we were able to decrease the timing interval to millisecond precision, hence the socket algorithms should operate faster than this interval to meet the data rate.

The portability of the programming language: The synchronization of the channel depends solely on the correct and consistent functionality of program subroutines and the libraries used. One example is the `nanosleep` subroutine. The operation of these subroutines should be standard in different operating environments (e.g., same precision in both sender and receiver).

All four factors affect the packet synchronization, maximum allowable bandwidth, and may introduce noise into the channel. Some of these factors can be mitigated (e.g., complexity and portability) by efficient design and implementation methodologies. Others, such as varying network conditions, need more intelligent mechanisms to cope with them. We come back to this problem in Section 3.4 where we present different mechanisms for decreasing noise and for resynchronizing the channel.

3.2 Channel Implementation

We implemented our covert channel as a client and server using Berkeley sockets library in C for our communication protocol, and Python version 2.3 to encode/decode the data sent on the channel and as a wrapper that called the C library functions. This software was developed for and ran under RedHat Linux 9.0 kernel version 2.4.20.

The effective operation of the channel depends on synchronization between the sender and receiver. In our implementation, the receiver initially listens on a blocking socket, and is therefore suspended until the initial transmission, called the *start of frame* (SOF), arrives. It then continues executing and checks whether a packet arrives during the covert interval over non-blocking sockets. At the end of the frame, it reverts to a blocking socket until the next SOF. This scheme does not entirely solve the synchronization problem and several other schemes are discussed in Section 3.4.

Given the encoded message, the sender sends a packet in the middle of the timing interval for each 1, and stays silent for each 0. Before sending the data bits of a frame, the sender sends the SOF denoting the beginning of frame.

3.3 Determining the Timing Interval

The bandwidth of the timing channel is determined by the choice of *timing interval*, which is the interval between successive transmissions. The smaller the interval the higher the transmission data rate. The bandwidth of the channel can be made as high as the processing speeds that the receiver and sender allow. There is a tradeoff though, because network jitter, scheduling in the system, and clock skew increase the probability of errors as the timing interval is decreased. In Section 3.5 we explore this tradeoff experimentally.

The time interval of the channel must be known to both the sender and receiver for communication to be successful. It might be established by default, set ahead of time, or the sender could use a storage channel in the initial SOF packet to communicate what the interval will be.

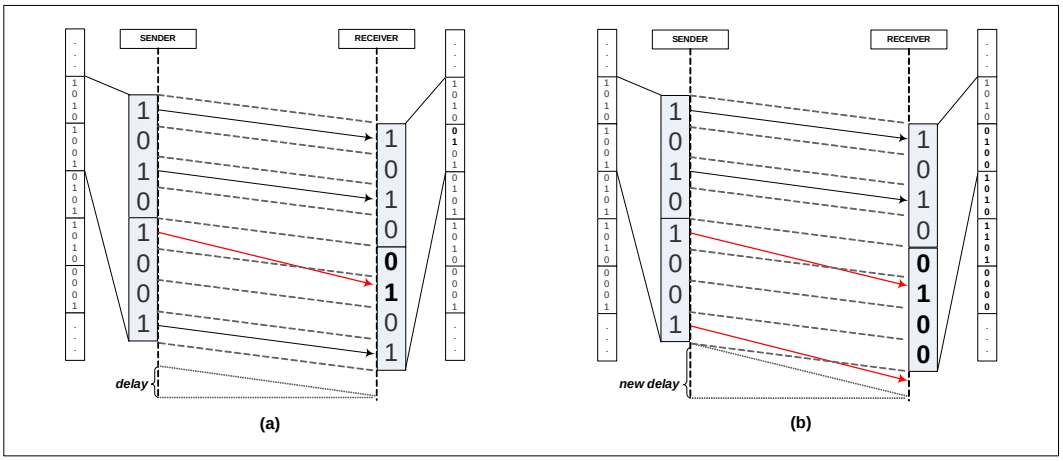

Figure 2: The synchronization problem in the IP covert timing channel. (a) A temporary change in network conditions causing the channel to enter the error state temporarily after the fourth bit. (b) A longer-term change in network conditions causing the channel to enter the error state and stay there.

3.4 Synchronization

In a covert timing channel, all information transmitted is based entirely on the arrival time of packets at the receiver. Because the sender and receiver may operate with different clocks, it becomes a challenge to implement end-to-end synchronization, particularly in a one-way channel. Jitter can cause packets to be recorded as arriving in a time period before or after the intended one, as shown in Figure 2(a).

While some error from jitter can be corrected with error-correcting codes, longer-term changes that occur in the middle of a transmission might cause an entire series of transmission to be shifted (Figure 2(b)). Clearly this problem can be solved by simply making the timing interval much larger than any expected network or processing delays, but this reduces the bandwidth of the channel. In this section, we describe techniques we used to help maintain synchronization.

3.4.1 *Start of frame (SOF):*

As a precaution against low levels of jitter in the network, each packet is sent in the middle of the timing interval. Moreover, upon receipt of every SOF packet, the receiver aligns itself with the newly received SOF by assuming that the SOF arrived exactly in the middle of the timing interval. This aligns the sender and receiver timing windows and in turn helps maintain synchronization.

3.4.2 *Silent intervals:*

We enhance the previous scheme by introducing silent intervals between frames. During a silent interval no packet transfer occurs between sender and receiver. We assume that the parties have previously determined the length of the silent interval. This interval can either be a default value or the covert channel itself can be initially used to send this value before the actual data transfer begins. The sender can enter the silent state any time during the transmission. Note that the sender has no way of knowing whether the receiver received the covert bits correctly or not. Therefore, it is up to the sender to observe the changing network conditions and make the decision when to pause the transmission. As an example, a sudden change in the RTT between the sender and the receiver might be a good signal for entering the silent state. On the other end, the receiver simply waits for the arrival of the SOF packet and takes no action. A simpler option is to enter the silent state periodically to clear the channel. This method increases channel accuracy at the expense of transmission rate. We investigate this tradeoff between channel accuracy and transmission time in Section 3.5.

3.4.3 *Interval adjusting:*

Rather than slow down the transmission by introducing silent periods in which no transfer occurs, the channel can adapt to the changes gradually as the network conditions change. In our *interval adjusting* scheme, the receiver closely monitors the time each packet arrives and compares it to the projected ideal case (the expected arrival time of the next packet) based on the current timing interval. Comparing the two, a *delta* is computed, which is the deviation between the ideal and actual times. The receiver then simply adds this value to its timing interval and adjusts its clock for the next arriving packet. Note that *delta* can be positive or negative, depending on whether the packet arrived early or late. This scheme is most useful when there is an incremental change in the network conditions that persists for longer than the lifetime of a single packet. It can however lead to errors if the change in the network delay is greater than 50% of the timing interval (e.g., adjust to an incorrect timing interval). As a precaution, we restrict the magnitude of each adjustment to be less than 10% of the difference between two consecutive intervals.

3.4.4 *Phase locked loop (PLL):*

A more promising solution for combating errors caused by variable network delays is to make interval adjustment more responsive to changes in delay. A phase-locked loop (PLL) is a popular method in communications used for bit and symbol synchronization. A PLL is a closed-loop feedback control circuit that is designed to track or synchronize an output signal with an input signal in frequency and phase

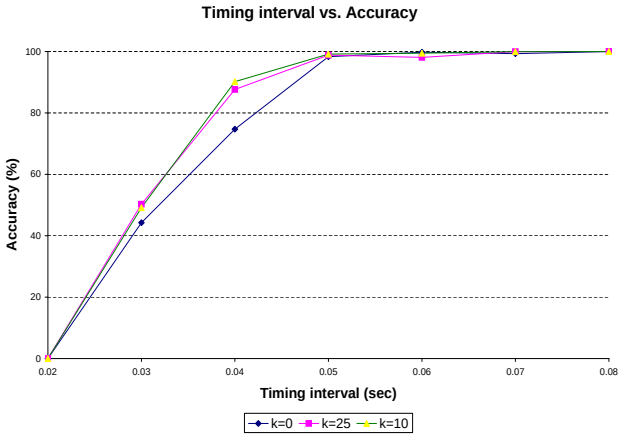

Figure 3: Timing interval versus accuracy with different values of k for the silent interval synchronization scheme.

[5]. The current error in synchronization between the output and input signal at a given instant in time is used to refine the synchronization between the signals at a future instant. Our future work includes investigation of a PLL as a synchronization mechanism.

3.5 An Empirical Evaluation of the IP Timing Channel Performance

In this section we show the performance of our IP timing channel. The communication channel becomes lossy as the timing interval is decreased due to the impact of one or more performance factors described in Section 3.1. We investigate the maximum data rate provided by our IP timing channel by decreasing the timing interval and observing the corresponding accuracy.

The channel accuracy can be measured as the percentage of correctly received bits, characters, or words. Because of potential erasures or shifting of bits, the number of bits, characters, or words may not be identical in the sent and received messages. We therefore measure accuracy based on *edit distance*, which is the minimum distance between two strings (in our case bits or characters) that is needed to transform the first string into the second. We use an efficient ($\Theta(mn)$ where m and n are the lengths of the strings) dynamic programming approach to calculate the edit distance known as the Wagner-Fischer technique [33].

Our covert channel ran between Purdue and Georgetown Universities, and was subject to changing network conditions. During "normal" network conditions, the route between communicating parties was twelve hops with an average RTT of 31.5 msec. In order to assess the accuracy of our covert channel under varying traffic loads, we ran our experiments at different times. Our results show that an IP timing channel is highly dependent on network factors.

3.5.1 *Effect of timing interval size:*

We first investigate the potential data rate of our channel by decreasing the timing interval until the accuracy drops. We mark this point as a threshold that can be thought as a boundary between the lossless and lossy communication and calculate the corresponding channel bit and character rate. In this experiment we used the periodic silent intervals synchronization scheme described in Section 3.4, with k denoting the frequency the synchronization scheme goes into a silent period (e.g., every twenty timing intervals). The character coding is eight bit ASCII with no error correction. Figure 3 shows the trade-off between the timing interval and the channel accuracy. Our channel provides nearly lossless communication for larger intervals at the cost of lower transmission bandwidth.

The experiment results show that the threshold value for the covert interval is around 0.06 seconds, which guarantees nearly 98% character accuracy for all three values of k. The equivalent channel bit rate is 16.666 bits per second (bps). With ASCII encoding and the SOF bit taken into account, we calculate the channel character rate around 1.852 characters per second (cps). As expected, the channel accuracy remains high for larger timing intervals. It also remains slightly higher when the transmissions are periodically paused for resynchronization.

3.5.2 *Effect of network conditions:*

In this experiment, we demonstrate an example of a network congestion and its effects on the performance of the covert channel. We plan to expand on these results in future work with reproducible network conditions using the DETER test bed [8].

We ran our covert channel on a congested network with a highly varying RTT between the sender and receiver with mean RTT at 42.07 msec. The normal RTT values for this channel have a mean RTT at 31.5 msec. Our evaluations show that congestion lowers the accuracy rate. For example, with timing interval set to 0.08, we observe 100% average character accuracy under normal conditions, but the accuracy drops to 82.11% for the congested network. Clearly, the interval must be increased to retain accuracy during periods of high congestion.

4. DETECTING IP COVERT TIMING CHANNELS

In this initial exploration, our focus is on whether we can create mechanisms that can detect covert channels in IP traffic. To this end we have developed and experimented with two different methods. As we explain in Section 4.1, each method tries to detect the fundamental regularity that must exist for a covert timing channel to exist. In Figure 4(a), we show the inter-arrival times of a simple covert timing channel. The y-axis is the inter-arrival time and the x-axis is the packet number. In Figure 4(b), we have sorted the inter-arrival times from smallest to largest. The result is a step function (note that because of varying network load, it is not a perfect step function). From these two figures, we observe that there appear to be approximately 4 or 5 different inter-arrival times. This highly regular behavior is a direct result of the static encoding of the frames in the timing channel. The arrival of packets is separated by 0, 1, 2, 3, 4,... intervals (the number of intervals separating packets is the number of "zeros" between two consecutive "ones" in a codeword). In contrast overt traffic packets can arrive anytime, resulting in an irregular pattern.

We present empirical results that show that for the simple case of a covert channel with a single interval and no noise

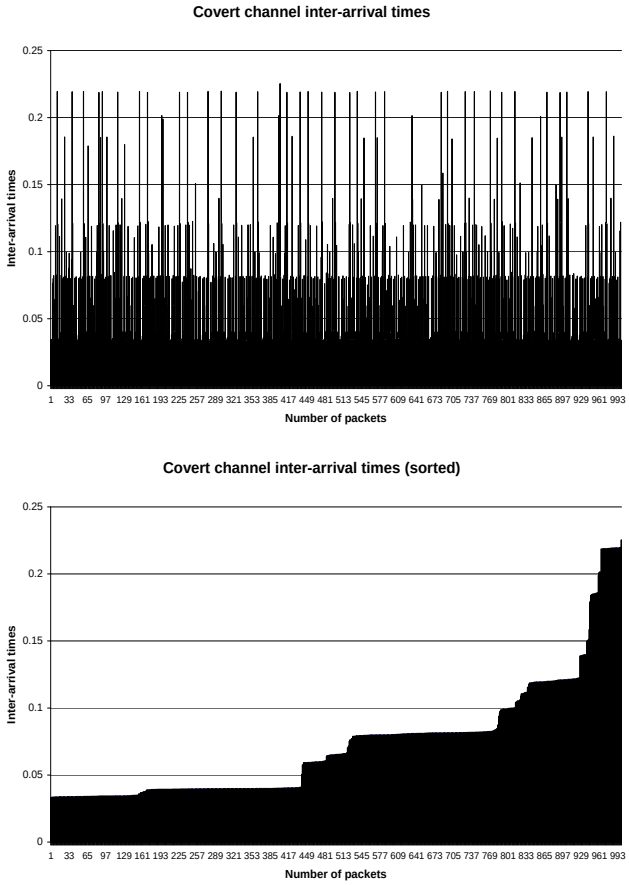

Figure 4: Inter-arrival times for the covert timing channel. (a) Actual values. (b) Sorted values.

that both of the proposed methods are highly effective at detecting covert channels. We then explore how well each method performs when measures are taken to try to hide the covert channel's regularity.

4.1 Methods for Detecting Regularity in Inter-arrival Times

Assume that we have observed n packets (in our experiments we set n to be 2000). Our objective is to develop metrics that capture any pattern of regularity in the traffic that is suggestive of a covert timing channel.

4.1.1 Measure 1: Examining patterns in the variance:

Our first method examines whether the variance in the inter-arrival (IA) remains constant. To this end, we separate the traffic into non-overlapping windows of size w packets. For each window i, we compute the standard deviation σ_i of the IA times. To compute our heuristic measure of regularity, we then calculate the pairwise differences between σ_i and σ_j for each pair of windows $i < j$. Finally to obtain a summary statistic, we compute the standard deviation of the pairwise differences. The following formula summarizes the process:

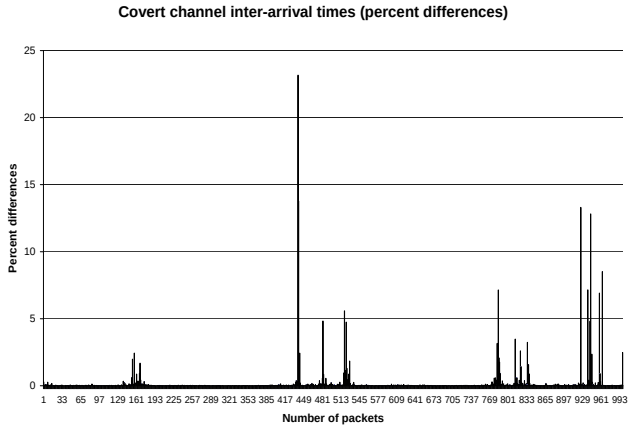

Figure 5: Relative differences of the covert timing channel inter-arrival times.

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

4.1.2 Measure 2: ϵ-Similarity between adjacent inter-arrival times

The second measure is derived from the sorted IA times (see Figure 4(b)). From this sorted list, we compute the relative difference between each pair of consecutive points. For example the relative difference between P_i and P_{i+1} is computed as $|P_i - P_{i+1}|/P_i$. We show these pairwise relative differences plotted in Figure 5. We can then compute a measure of similarity, which we call ϵ-Similarity by computing the percentage of relative differences that are less than ϵ. For covert channels the majority of the pairwise differences in the sorted list of IA times will be very small. It is large only for jumps in the step function (see Figure 4(b)).

4.1.3 A discussion of other approaches:

We also investigated several approaches that were not fruitful, but were more obvious from a statistical point of view.

Indexes of dispersion of a point process have been used as a tool in network characterization [16, 28]. In particular, *index of dispersion for intervals (IDI)* can be used to qualitatively compare the inter-arrival times of a point process with the Poisson process serving as the basis (for which the IDI is unity) [7]. IDI provides a finer measure for defining the *variability* of the process than does a second order moment analysis. In [16], the variability, or the burstiness, of the network traffic is defined as "the changes in the variance of the sum of consecutive inter-arrivals." Although this measure appears promising, it makes a number of assumptions including *stationarity*, which needs to be verified for the correct interpretation of the results. In this initial study, we do not impose such assumptions on the distributions of covert or overt traffic. Our future work includes such analysis of both types of traffic.

Another avenue we examined was statistical non-parametric tests similar to those used in other work [26, 4, 6]. Applications of these tests has mainly concentrated on network traffic characterization and modeling. The goal is often to determine whether two streams come from the same

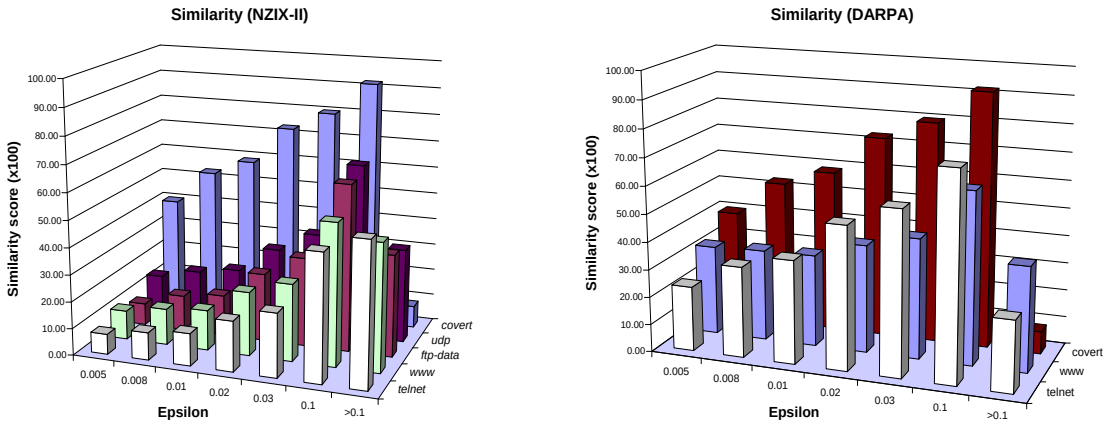

Figure 6: Similarity of different types of traffic (a) Comparison of NZIX-II unhidden and covert traffic. (b) Comparison of DARPA and unhidden covert traffic.

empirical distribution. For example, there are applications of the Kolmogorov-Smirnov test. In our research, we are not seeking to model either the overt or the covert network traffic. Our goal is to define metrics that differentiate covert from overt traffic, therefore, these methods are not directly applicable to the detection of IP timing channels.

4.2 An Empirical Evaluation

The goal of our experiments was to examine the efficacy of our two metrics. To this end we first report experiments with a basic covert channel that employs a single timing interval throughout the communication and does not try to mask itself in any way. Our second set of experiments looks at how our metrics fare when the measures are taken to hide the channel. Our ultimate experimental objective is to measure not only our method's false negative rate for covert channels but also its false positive rate for non-covert traffic. To this end, our third experiment explores how our metrics can be combined to form an automated detection method.

4.2.1 Data sets:

In our experiments, we used both synthetic and real traffic data sets for the sake of completeness. Our synthetic data set is the '99 DARPA data set for Telnet and HTTP traffic [21]. Additionally, we employ the second version of NZIX data sets (NZIX-II) which is a collection of TCP and UDP traces collected by the WAND research group [15]. For the TCP traces, we chose to investigate Telnet, FTP, and HTTP traffic.

For each experiment we report results for traffic flows of 2000 packets. Our goal is not to model or identify a traffic distribution, but to determine whether we can accurately detect a covert channel in a window of 2000 packets. In future work we will investigate what is the minimum length of the window for which our methods are still effective. Note that although the covert channel was run between Purdue and Georgetown Universities, for the non-covert traffic we use the recorded IA times in the datasets. A drawback is that we cannot have the same network conditions (e.g., number of hops, same jitter), but excluding the case of jitter this does not impact our results. None of our measures look at absolute IA values, but rather compute measures of regularity in terms of the relative differences among IA values.

Dataset	Application	w=250	w=100
NZIX-II	WWW	22.14	34.32
NZIX-II	FTPD	7.77	16.46
NZIX-II	TELNET	12.08	18.15
NZIX-II	UDP	16.57	27.18
DARPA	WWW	21.59	62.32
DARPA	TELNET	17.70	52.21
	COVERT-I	2.18	4.63

Table 1: Regularity of NZIX-II, DARPA, and covert traffic with windows of size 250 and 100.

4.2.2 Covert Channel I: A simple timing channel:

Our first experiment examines each metric's ability to detect a covert timing channel that employs a single timing interval (set to be 0.04 sec) for the entire communication. In Table 1 we show the regularity of the variance for two window sizes (100 and 250) within the 2000 packet dataset. Our results are the average of ten different sets of data for each protocol, including the covert channel.

Observe that the variance in the pairwise differences between the variance of each pair of windows is on average less for the covert channel than for the other traffic. However, one FTP and one UDP dataset had similarly low scores. This is to be expected because FTP and UDP send streams of data as fast as possible resulting in a uniform IA. Note that the smaller window size appears to better differentiate the covert channel's regularity from the other protocols. In other words, there is a larger difference between the value (4.63) for the covert channel and the values for the non-covert channels.

In Figure 6 we show the results for the second metric, ϵ-Similarity. The x-axis shows ϵ and the y-axis shows the percentage of all pairs of sorted IA values whose difference is less than ϵ. For a covert channel we would anticipate that the majority of the traffic would have small differences in the sorted IA values. For both the NZIX-II and the DARPA datasets, the graph show the results for Telnet, WWW,

Method	t	ϵ-Similarity Score 0.005	0.008	0.01	0.02	0.03	0.1	>0.1
Sequential	250	34.17	45.17	51.23	67.38	75.29	90.75	9.25
	100	34.12	45.77	52.78	67.53	75.54	90.50	9.50
	50	34.22	46.87	53.68	67.68	75.09	89.89	10.11
	10	34.87	46.37	51.83	67.58	76.19	90.65	9.35
Random	250	36.51	48.02	53.47	68.30	76.20	90.49	9.51
	10	35.21	46.88	52.55	68.29	75.67	90.28	9.72
Original		39.92	52.83	58.58	72.79	79.74	91.85	8.15

Table 2: ϵ-Similarity scores for Covert Channel II. For each window of t packets, the interval is selected to be from the set (0.04, 0.06, 0.08). Results are shown for both selection methods (Sequential and Random) and for the original covert channel that employs a single interval (0.04).

FTP-data, UDP and the covert channel. The reported values are averaged over ten runs. The results show a striking difference between the covert channel and non-covert flows for the NZIX-II data. For example, 40% of the covert traffic has a difference of less than $\epsilon = 0.005$. Whereas for the non-covert channel less than 15% are this similar. What is interesting is that although the trend is similar for the DARPA dataset, there is far more regularity in the DARPA data than in the NZIX-II data. Indeed, studies have shown that because the normal traffic in the DARPA dataset was synthetically generated, it is not entirely representative of real traffic [24, 22]. Although previous studies have not examined the specific inter-arrival times, they have illustrated that 1) many attributes of DARPA network traffic are more predictable than the real traffic, and 2) the synthetic dataset shows different statistical characteristics than real data. Hence we conjecture that the regularity shown in Figure 6 for ϵ-Similarity for the DARPA dataset is a direct consequence of the nature of the synthetic data.

4.2.3 Covert Channel II: Varying the timing interval:

To understand how our metrics work when the sender tries to hide the covert channel, we first experimented with covert channels where the sender alternates between different intervals. The motivation from the sender's viewpoint is to obfuscate the regularity. In our experiment, we chose three different interval values 0.04, 0.06, and 0.08. After t packets, we switch to a new interval. We experimented with two different methods of specifying the new interval: cycling through them sequentially or random selection.

Varying the interval impacts Measure I (regularity) because the variance of the windows are no longer comparable unless t is much smaller than w. In this case, all three intervals would be observed several times in each window of w packets, and therefore the variance for each window would be similar. However, for cases where t approaches or exceeds w this metric *cannot detect covert timing channels* and hence due to space we do not show the actual numbers.

On the other hand, our second metric (ϵ-Similarity) still shows differences in values for the covert versus the non-covert traffic. In Table 2 we show the results for the original single-interval covert channel, and for several choices of t for both methods for selecting a new interval period after t frames. Note that the results are averaged over ten runs for each parameter setting. Looking at each of the seven values of ϵ, we see little difference for either the sequential or random method. These results show that the ϵ-Similarity metric is robust for this method of hiding covert traffic.

4.2.4 Covert Channel III: Injecting noise:

Our third experiment examines how our measures fare when we explicitly introduce irregularity into the covert channel. We inject noise into the channel as follows. For a covert channel operating on a port typically associated with a particular application X, we insert portions of inter-arrival times from a non-covert traffic stream for application X. For example, if the covert channel runs on Port 80, we use WWW traffic. We then break the covert channel into blocks of 100 packets, and randomly replace blocks of the covert traffic with the non-covert traffic of application X until we achieve the desired noise level (e.g., for 10% noise, the IA times for two randomly selected blocks of 100 packets would be replaced in our 2000 packet stream).

This scheme again impacts our first measure because of the random nature of noise injection. Because a window may include components from the noisy traffic, the windows are no longer comparable and our regularity measure fails to discriminate covert from non-covert traffic.

Our second measure, however, fares better. In Table 3 we show the ϵ-Similarity values for the original covert channel (Covert Channel I – shown in the 0% noise row), and for noise levels of 10, 25 and 50%. In addition, we include the values for the non-covert traffic in the bottom three rows of the table. Note that as the noise level increases the covert traffic begins to have ϵ-Similarity values close to the non-covert traffic. However, a drawback from the sender/receiver's viewpoints is that the covert bandwidth decreases linearly with the noise level.

4.3 Automatic Detection of IP Covert Timing Channels

In this section, we present the results of an experiment designed to evaluate our metrics' ability to be used to automatically detect covert timing channels. Both of our methods require that we set a threshold. For ϵ-Similarity, we need to choose a threshold for each value of ϵ [1]. For our regularity metric, values below the threshold are considered to have been generated by covert traffic. To set the parameters, we first ran experiments with ten flows from each protocol type. Note that we experimented only with WWW and FTPd traffic, as in the NZIX-II dataset there is insufficient data for the other protocols to find ten flows of 2000 packets. After we collected the data from the ten *train-*

[1]Note that for values of $\epsilon < 0.1$ observations above our threshold are considered covert traffic and for $\epsilon > 0.1$ values below our threshold are considered covert, because the majority of covert traffic has a similarity ≤ 0.1

Noise Level	Type of Noise	ϵ-Similarity Score 0.005	0.008	0.01	0.02	0.03	0.1	>0.1
0%		39.92	52.83	58.58	72.79	79.74	91.85	8.15
10%	WWW	36.54	47.50	52.67	66.46	73.39	87.46	12.54
10%	FTPD	35.03	46.05	51.30	64.89	71.45	84.94	15.06
10%	TELNET	34.89	45.83	51.14	64.29	70.70	83.17	16.83
25%	WWW	31.88	40.93	44.45	58.96	65.76	83.01	16.99
25%	FTPD	30.69	39.93	44.43	56.88	63.14	78.80	21.20
25%	TELNET	29.06	38.34	42.61	54.12	60.04	73.27	26.73
50%	WWW	31.70	37.31	40.33	53.15	59.52	79.32	20.68
50%	FTPD	26.12	32.21	35.60	46.35	52.39	70.53	29.47
50%	TELNET	24.21	30.31	33.31	42.47	47.72	61.40	38.60
Non-covert Traffic								
WWW		10.81	13.49	14.96	23.76	28.70	52.69	47.31
TELNET		7.54	10.25	12.04	18.69	23.65	46.99	53.01
FTPD		8.20	13.19	15.19	25.36	33.20	62.05	37.95

Table 3: ϵ-Similarity scores with different classes and levels of noise.

WWW	**Threshold**	FP	Cov-I	Cov-II	Cov-III(10%)	Cov-III(25%)	Cov-III(50%)
	$\mu + 2\sigma$	10.0	0.0	0.0	86.6	100.0	100.0
	$\mu + 1.5\sigma$	10.0	0.0	0.0	0.0	53.0	86.6
	$\mu + 1\sigma$	10.0	0.0	0.0	0.0	0.0	86.6
	$> Max$	10.0	0.0	0.0	0.0	20.0	86.6
FTPD	**Threshold**	FP	Cov-I	Cov-II	Cov-III(10%)	Cov-III(25%)	Cov-III(50%)
	$\mu + 2\sigma$	10.0	0.0	66.7	86.6	100.0	100.0
	$\mu + 1.5\sigma$	10.0	0.0	0.0	0.0	80.0	93.3
	$\mu + 1\sigma$	30.0	0.0	0.0	0.0	6.7	93.3
	$> Max$	10.0	0.0	0.0	0.0	33.3	86.6

Table 4: False positive (FP) and false negative (FN) rates for covert channel detection.

ing flows, we compute various statistics to determine our thresholds. Specifically we calculate the mean and standard deviation, and record the minimum and maximum observed values for each of our metrics. We then applied several different thresholds to a new set of independent *testing* data. For each protocol we collected ten new flows (we did not examine them until after parameter selection).

The results for ϵ-Similarity are shown in Table 4. For each protocol, we show the results for several different choices of threshold. Specifically, values smaller than 2, 1.5 or 1 standard deviation(s) away from the mean value observed for the non-covert traffic. We also show values for thresholds set to be greater than the largest observed value for $\epsilon < 0.1$ (and less than the smallest observed value for $\epsilon > 0.1$). To be classified as a covert channel all seven ϵ-Similarity metrics are computed and we take a majority vote.

In the third column of the table we show the results for the non-covert traffic, which gives us the false positive rate (FP). Our results show that the false positive rate for both WWW and FTPd traffic is 10% for most choices of the threshold value.

We also report the false negative rates for several different types of covert channel. Specifically we report the false negative rate for the basic channel (Cov-I), for a channel for which the interval is varied randomly between three choices (Cov-II), and for channels with noise injected (Cov-III). For the original covert timing channel (Cov-I) our false negative rate is 0%. For the interval varying covert channel (Covert II), all values of our thresholds work well except for the strictest test which is $\mu + 2\sigma$, in this case the covert channel begins to resemble the normal FTPd traffic. Finally, for the channels in which noise is injected (Covert III) the higher the level of noise the more difficulty our method has in recognizing that it is a covert channel. However, for both the WWW and FTP traffic at 10% noise, most values of the threshold work well.

We also ran experiments for our measure of regularity. For the simple timing channel without any noise, our method is able to detect it with 100% accuracy with a 0.0% false positive rate. However, as discussed earlier this method fails when attempts are made to obfuscate the channel (Cov-II and Cov-III).

5. CONCLUSIONS AND FUTURE WORK

In this paper we have presented the design and implementation of a network covert timing channel; developed methods of distinguishing the covert traffic generated by our channel from normal traffic; and examined the efficacy of our detection methods in the face of counter-measures attackers seem likely to pursue.

The implementation of the timing channel raised a number of non-obvious issues in its design, particularly in methods of determining timing intervals in the absence of an accurate, shared clock. Our implementation uses a variety of mechanisms to synchronize the data stream, including use of blocking and non-blocking sockets; periodic idle intervals; and dynamic adjustment of the intervals. We then evaluated the performance of the channel to determine the maximum dependable speed of transmission.

We then collected data while our timing channel communicated between two remote locations on the Internet, and using this data, developed two methods to differentiate covert traffic traces from normal traffic traces obtained from widely used research data. The first method measures

the regularity the inter-arrival time of packets in the trace. The second, ϵ-Similarity, measures the similarity of pairs of sorted inter-arrival times.

We then empirically evaluated the performance of these methods in three different scenarios: a simple, unobfuscated timing channel; a channel in which the timing interval varied during transmission; and a timing channel that paused periodically for transmission of noise of a form that would mimic the protocol used for cover. Both detection methods could reliably differentiate the covert traffic in the simple case. In the second case, with varying timing intervals, the ϵ-Similarity measure succeeded in identifying the timing channel after the regularity measure failed. In the third scenario, as the amount of noise and available covert bandwidth increased, the success of our methods decreased.

This work was an initial exploration into the creation and detection of network covert timing channels and there are many avenues for future work. In the short term we will add error-correction and better synchronization techniques to increase the bandwidth of the covert channel. In the longer term we will investigate other detection methods designed to be robust in the face of attempts to hide its regularity.

Acknowledgements

Carla Brodley's research was supported by AFRL grant number F30602-02-2-0217 and by a grant from the National Science Foundation grant number 0335574. The authors would like to thank Miguel Rui Forte for his participation in discussions about this research.

6. REFERENCES

[1] Christopher Abad. IP checksum covert channels and selected hash collision. Technical report, 2001.

[2] Kamran Ahsan. Covert channel analysis and data hiding in TCP/IP. Master's thesis, University of Toronto, 2000.

[3] Kamran Ahsan and Deepa Kundur. Practical data hiding in TCP/IP. In *Proc. Workshop on Multimedia Security at ACM Multimedia*, December 2002.

[4] Hari Balakrishnan, Mark Stemm, Srinivasan Seshan, and Randy H. Katz. Analyzing stability in wide-area network performance. In *Proceedings of the 1997 ACM SIGMETRICS international conference on Measurement and modeling of computer systems*, pages 2–12. ACM Press, 1997.

[5] Ronald E. Best. *Phase-locked loops: Design, simulation and applications*. McGraw-Hill Professional, 5th edition, 2003.

[6] Kimberly C. Claffy, George C. Polyzos, and Hans-Werner Braun. Application of sampling methodologies to network traffic characterization. In *Conference proceedings on Communications architectures, protocols and applications*, pages 194–203. ACM Press, 1993.

[7] D. R. Cox and P. A. W. Lewis. *The statistical analysis of series of events*. Chapman and Hall, 1966.

[8] Cyber Defense Technology Experimental Research (DETER) network. http://www.isi.edu/deter/.

[9] Daemon9. Project Loki. *Phrack*, 49(6), August 1996.

[10] Alex Dyatlov and Simon Castro. Exploitation of data streams authorized by a network access control system for arbitrary data transfers: tunneling and covert channels over the HTTP protocol. June 2003.

[11] Gina Fisk, Mike Fisk, Christos Papadopoulos, and Joshua Neil. Eliminating steganography in Internet traffic with active wardens. In *5th International Workshop on Information Hiding*, volume 2578, pages 18–35, October 2002.

[12] John Giffin, Rachel Greenstadt, Peter Litwack, and Richard Tibbetts. Covert messaging through TCP timestamps. In *Workshop on Privacy Enhancing Technologies*, volume 2482, pages 194–208, April 2002.

[13] James Giles and Bruce Hajek. An information-theoretic and game-theoretic study of timing channels. In *IEEE Transaction on Information Theory*, volume 48, pages 2455–2477, September 2003.

[14] Virgil Gligor. A guide to understanding covert channel analysis of trusted systems. Technical Report NCSC-TG-030, National Computer Security Center, Ft. George G. Meade, Maryland, U.S.A., November 1993.

[15] WAND Research group. NZIX-II trace archive, data available at http://pma.nlanr.net/traces/long/nzix2.html.

[16] Riccardo Gusella. Characterizing the variability of arrival processes with indexes of dispersion. *IEEE Journal on Selected Areas in Communications*, 9(2):203–211, February 1991.

[17] Mark Handley and Vern Paxson. Network intrusion detection: Evasion, traffic normalization, and end-to-end protocol semantics. In *Proceedings of the 10th USENIX Security Symposium*, August 2001.

[18] Paul A. Henry. Covert channels provided hackers the opportunity and the means for the current distributed denial of service attacks. Technical report, 2000.

[19] James W. Gray III. Countermeasures and tradeoffs for a class of covert timing channel. Technical report, 1994.

[20] M. Kang, I. Moskowitz, and D. Lee. A network version of the pump. In *Proceedings of the IEEE Symposium in Security and Privacy*, pages 144–154, May 1995.

[21] Richard Lippmann, Joshua W. Haines, David J. Fried, Jonathan Korba, and Kumar Das. The 1999 DARPA off-line intrusion detection evaluation. *Computer Networks*, 34(4):579–595, 2000.

[22] M Mahoney and P Chan. An analysis of the 1999 DARPA/Lincoln Laboratory evaluation data for network anomaly detection. In *Proceeding of Recent Advances in Intrusion Detection (RAID)-2003*, volume 2820, pages 220–237, September 8-10 2003.

[23] John McHugh. Covert channel analysis. Technical report, December 1995.

[24] John McHugh. Testing intrusion detection systems: A critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Transactions on Information and System Security*, 3(4):262–294, November 2000.

[25] U.S. Department of Defense. Trusted computer system evaluation "The Orange Book". *DoD 5200.28-STD Washington: GPO:1985*, 1985.

[26] Vern Paxson. Empirically derived analytic models of wide-area TCP connections. *IEEE/ACM Trans. Netw.*, 2(4):316–336, 1994.

[27] Phil A. Porras and Richard A. Kemmerer. Covert flow trees: A technique for identifying and analyzing covert storage channels. In *Proceedings of the 1991 IEEE Computer Society Symposium on Research in Security and Privacy*, May 1991.

[28] C. Rosenberg, F. Guillemin, and R. Mazumdar. New approach for traffic characterisation in ATM networks. In *IEE Proceedings - Communications*, volume 142, pages 87–90, April 1995.

[29] C. Rowland. Covert channels in the TCP/IP protocol suite. *First Monday: Peer-reviewed Journal on the Internet*, 2(5), 1997.

[30] Sergio D. Servetto and Martin Vetterli. Communication using phantoms: Covert channels in the Internet. In *IEEE International Symposium on Information Theory*, June 2001.

[31] J. Christian Smith. Covert shells. *SANS Institute Information Security Reading Room*, November 2000.

[32] C.R. Tsai, V.D. Gligor, and C.S. Chandersekaran. A formal method for the identification of covert storage channels in secure XENIX. In *Proceedings of the 1987 IEEE Symposium on Security and Privacy*, April 1987.

[33] Robert A. Wagner and Micheal J. Fischer. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, January 1974.

[34] John C. Wray. An analysis of covert timing channels. In *Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy*, May 1991.