Detection of Covert Channel Encoding in Network Packet Delays

Department of Computer Science - Dartmouth College
Technical Report TR536, Revision 1
August 2005, revised November 2005

Vincent Berk, Annarita Giani, George Cybenko
Thayer School of Engineering
Dartmouth College
Hanover, NH
{*vberk, agiani, gvc*}*@dartmouth.edu*

Abstract

Covert channels are mechanisms for communicating information in ways that are difficult to detect. Data exfiltration can be an indication that a computer has been compromised by an attacker even when other intrusion detection schemes have failed to detect a successful attack. Covert timing channels use packet inter-arrival times, not header or payload embedded information, to encode covert messages. This paper investigates the channel capacity of Internet-based timing channels and proposes a methodology for detecting covert timing channels based on how close a source comes to achieving that channel capacity. A statistical approach is then used for the special case of binary codes.

1 Introduction

Modern network defense and host intrusion detection technologies are forcing computer hackers to be more creative and subtle in devising and applying attacks. Accordingly, there is growing interest in not only preventing attacks but in detecting compromised machines after the fact. In a situation where an attacker has successfully compromised a protected host, the question of stopping the attack is moot and the problem shifts to collecting evidence that the attack was successful. If the goal of the attack is stealing or otherwise exfiltrating information from the compromised machine or network, some communication mechanism must be used to get the information out. In such a case, the intrusion can, in principle, be detected by identifying a channel that is being used to exfiltrate information.

Assuming the network is heavily guarded with Intrusion Detection Systems, Packet Anomaly Detection Systems, and firewalls, the intruder has limited options for getting the data out. The easiest way is the use of a normal communication protocol, such as FTP (File Transfer Protocol), to exfiltrate the data. This can be detected in log-files and traffic dumps, especially if the communications are with machines that are not part of the normal traffic. Running a similar communication on uncommon, high port numbers would likely trigger Packet Anomaly Detection Systems, because such communications are highly unusual. Encoding data in the unused fields of packet headers is likely to set off most modern intrusion detection systems, as will adding data to the payload section of PING [1] packets. Additionally, transmitting data through a PING payload might trigger Packet Anomaly Detection Systems when the size of the Ping packets is increased and/or irregular. The attacker thus will have to look for more covert ways of moving the data out of the compromised network.

The data exfiltration mechanism under investigation in this paper is the use of inter-packet delay times to encode data. This means that the intruder does not necessarily have to create new traffic - he/she merely modulates the time between packets of regular communications to encode the data. The extent to which existing traffic streams can be used to create a timing channel depends on the location of the recipient. The intruder will need access to the communication stream to measure the packet inter-arrival times and retrieve the data. In this paper we propose a method for investigating inter-packet delays at the border of a protected network. Based on a sequence of delay times, our algorithm decides if the sequence of packets is a potential covert communication channel or not.

We propose two methods for detecting inter-packet timing delay covert channels. The first one is based on the idea that a skillful attacker tries to exfiltrate the data at maximum bandwidth as limited by the channel

This revision differs from the original only in the correction of one reference.

[1] Packet Internet Groper. A utility that forwards data packets to check the quality of a link or verify the connection of a machine to the Internet.

capacity. The probability distribution of interpacket delays that achieves the highest capacity is computed and compared with the real distribution of the input symbols observed in the network. The class of possible detection algorithms based on this method is extensive and we give only one example. The general approach, however, is very promising. The second technique, applicable to the case of a binary code, simply tries to identify two concentrations of inter-packet delay times. These two concentrations should be sufficiently differentiable, when the delays encode a binary bitstream. Extension to the case of a highest even number of input symbol is straightforward.

The remainder of the paper is organized as follows. Section 2 defines the scenario and introduces some background. Section 3 presents an input symbols analysis. Section 4 introduces Information Theory ideas directly applicable to the case of multiple input symbols. Section 5 focuses on binary covert channels. Conclusions and ideas for future research are provided in Section 6.

2 Covert Channels: definition and background

The first definition of covert channel was given in [9]. The paper explores the problem of confining a program during its execution so that it cannot transmit information to any other program except its caller. Channels were classified into three categories, Storage, Legitimate and Covert. Latter are those used for information transmission even though they are neither designed nor intended to transfer information at all.

Ten years later covert channels are defined as those that use entities not normally viewed as data objects, but that can be manipulated maliciously to transfer information from one subject to another [7]. The author examines the connection between covert channels and resource allocation policies. The same author in [8] presents a more general methodology for discovering and dealing with this type of covert communication.

The main characteristic of a covert channel is the aim to hide the fact that a transmission is taking place. Compare this with cryptography where the goal is to transfer data readable only by the receiver. In cryptography there is no intention to camouflage the communication. A covert channels may encrypt the data sent through it, but it mainly seeks to disguise its transmission.

Steganography is the oldest form of covert channel. It is the act of embedding a secret message within a larger message so that others cannot discern the presence or contents of the secret message. In the cyber world, steganography specifically means hiding information in text or multimedia files (like image or video files) in a way that is unnoticeable by an average user. Only special expertise and tools may lead to its detection. Statistical analysis is one way to detect this type of steganography.

Since security analysts first started thinking about covert channel communication, two terms have been introduced, **Covert Storage Channels** and **Timing Covert Channels**. Covert Storage Channels involve the writing to a storage location by one process and the reading of the storage location by another process. The storage resource [such as unused bits in a packet header or the padding fields in a data-gram] is shared between the two subjects. In a way steganography can be seen as a form of storage channel. With Timing Covert Channels the sender transmits data to the receiver by modulating its use of system resources in such a way that the manipulation affects the response time observed by the receiver [6]. Specifically, this can be done by modulating the wait time between packet transmissions (the inter-packet delays). This latter type is the focus of our research.

In our investigation, a covert communication between two machines consists in sending and receiving data bypassing the usual intrusion detection techniques. This subtle mechanism in fact uses only normal traffic, exploiting time delays between transmitted packets to implement a form of Morse-like code. Intuitively, for a two symbol code, this means that a short time delay between two consecutive packets encodes a binary zero, and a long time delay encodes a binary one. More generally, suppose an outside intruder has been able to gain control over a machine X inside our network and wishes to send data to his/her computer A by codifying the information as time delays between packets, [Figure 1].

Note that A does not have to be the destination for the network packets, but merely needs to be on the path so that the packets may be intercepted and their inter-packet delays measured. Intuitively, the fewer hops between machine X and machine A, the more accurate the received signals (delays) will be.

For example we can imagine that the intruder is able to execute instruction **Ping A** from the compromised machine X within the perimeter of our network.

- **Ping A**
- **Ping A**, after Δt_1 seconds
- **Ping A**, after Δt_2 seconds
- **Ping A**, after Δt_3 seconds
- $\cdots$

Machine A receives a sequence of PINGs with time delays between consecutive packets of $\Delta\bar{t}_1$ seconds, $\Delta\bar{t}_2$ seconds, $\Delta\bar{t}_3$ seconds and so on. From an abstract point

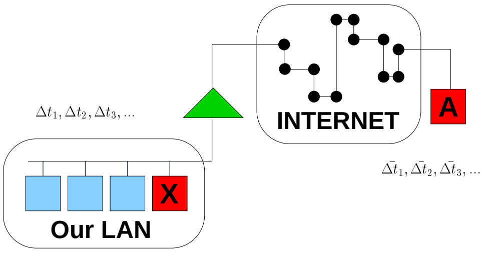
$\Delta t_1, \Delta t_2, \Delta t_3, ...$
INTERNET
A
X
Our LAN
$\Delta\bar{t}_1, \Delta\bar{t}_2, \Delta\bar{t}_3, ...$

Figure 1: An intruder was able to control machine X which is inside our local network and use it to exfiltrate data coded in inter-packet delays. Machine A is the receiver.

of view this technique amounts to sending symbols of a source through a noisy channel, and decoding the message at the destination. We consider in fact that the delays at the arrival computer are not exactly the same as in the departure computer due to noise within the Internet. Any forwarding device (routers, firewalls, switches, repeaters) will incur a small processing delay. This delay will vary in time, thus perturbing the inter-packet times and making the channel noisy.

Assume that all the packets flowing through our network are filtered by an agent (e.g., a firewall) that can record packet inter-delays for each related communication (for example, PINGs to the same destination address, or a stateful TCP session with a web server). From this agent we obtain a list of delays. The question we try to answer is: “Given a chain of consecutive delays Δt_i, is it possible to affirm with a certain probability that there has been malicious intent from within our network?”. In other words, “Are the observed delay times specific/dicernable enough to (most likely) *not* have been generated by chance?”

In the nineties some work was done on techniques to identify, react and quantify covert channel communication. These methods were mostly informal and specific to a particular situation. The effect of noise upon a simple covert timing channel is investigated in [10]. Information Theory is used to quantify the flow of information across the channel. In [1] the Shannon capacity of the single server queue is analyzed, and it is shown that this capacity cannot be increased by feedback.

Since detecting covert communication is very difficult, most researchers resort to investigating methods that minimize the amount of information that can be transmitted using a covert timing channel. They developed different ways of reducing the possible leakage. For instance, there have been studies focused on jammed timing channels [5]. A jammer adds noise to a timing channel by delaying packets. If the rate at which information can be conveyed in the presence of a jamming device is acceptably low, there will be no need to monitor the channel. Many protocols aim at reducing the bandwidth of covert channels. The *Quantized Pump* [14] is a protocol that forces a lower bound on the covert channel bandwidth. In [13] the authors show how trade-offs can be made to reduce the threat of covert channels. They introduce the *Small Message Criterion*, which gives guidelines for what level of covert leaking is tolerable. Their conclusion is that covert channels can never be totally eliminated from high-assurance computing systems.

In [11] the authors analyze timing channels that are discrete, memoryless and noiseless, called *Simple Timing Channels*. They discuss different ways of defining channel capacity and give its bounds when closed forms solutions are intractable or unnecessary. An initial inquiry into the relationship between covert channel capacity and anonymity was developed in [12]. The authors are concerned with how much information a sender to an anonymizer can leak to an eavesdropping outsider. In [3] the authors propose two detection methods that exploit the regularity of a timing channel compared to normal traffic. All detection is done through statistical methods. If the attacker adds noise to modify the statistics, however, this methodology becomes less effective.

In our investigation, we do not consider unskillful adversaries. Instead we suppose that the attacker is highly skilled and has the ability to conduct a thorough study of the network so that either the largest amount of information can be transmitted or stealth can be maximized. This assumption creates a “worst-case” scenario; any channel with a lower than optimal bandwidth or stealth will be easier to detect and disrupt.

Since the general problem is very broad we start with the restrictive assumption that the intruder does not add any artificial noise to prevent an observer from detecting his/her activity. Artificial noise can be added by wildly varying the inter-delays at a predetermined moment. For example, the intruder might send a sequence of packets with randomized inter-delays after successful transmission of every 100 bits. The intruder knows to discard these random inter-delays as malicious noise, but, our algorithms do not. Therefore the algorithms might fail to detect the malicious activity represented by the 100-bit block.

3 Input Symbol Analysis

The decision to use a given number of symbols (or delays) is not always a matter of maximizing transmission bandwidth. Consider, for instance, that bandwidth is maximized when we use two different symbols (i.e, two different delays) to transmit a given data file. Since it often is hard to control existing sessions (it usually requires root privileges), the attacker may be better off sending

HTTP requests to a webserver that he/she controls (alternatively, ping could be used). By measuring the times between the requests, the attacker will be able to decode the transmission.

Although the transmission rate is maximized, this case, is also, by far, the noisiest. For every bit transmitted, an HTTP request is made, something that may stand out when the traffic on the network is analyzed. Conversely, suppose that the attacker prefers stealth over transmission speed. Conceivably a 64 symbol encoding can be chosen, meaning that each HTTP request then carries 6 bits of information. Overall, it is illogical to assume the attacker will pick a given encoding based on the maximization of transmission speed. Especially if we consider that the attacker has chosen a *covert* channel to communicate, it is likely that he/she will try to maximize stealth over transmission bandwidth.

For the above reasons, the communication to modulate preferably would be a continuous datastream, for example, the acknowledgements of a real-time audio stream. An interactive session, such as SSH, will not have a predictable enough release of packets (i.e., only when the user types a command) to allow for transmission of large files. Add to that the fact that the attacker will rarely know beforehand what communications will predictably exist and it is easiest to simply create the traffic, instead of using existing connections, (although at an increased risk.)

In general, an attacker will most likely not have the luxury of choosing the most efficient or stealthiest encoding possible, for it assumes that the attacker can have some form of information exchange with the compromised system. If no such bilateral communication channel exists beforehand, the attacker will never be able to instruct the attacking agent on what encoding is best received (and therefore preferable). For that reason we may safely assume that an attacker will pick an encoding that will yield a *decent* bandwidth on average, while being *sufficiently* stealthy.

Let us now analyze the general case of a channel with multiple input symbols followed by the extreme case of a binary channel. For both cases we propose methods for detecting a covert channel in interpacket delays measured at an arbitrary point in the network. Both techniques are based on capturing related datastreams, meaning either a TCP communication, a PING to the same host address or network, or a series UDP packets with the same destination host and port. The observation sequence that is extracted is the stream of delays between outgoing packets. Thus each time a packet leaves the local network, the time since the last packet went out to the same destination is reported. We call it Δt. Any returning acknowledgements hold no value for an outgoing covert channel.

Since a detector is unaware of the time encoding that the attacker is using, an assumption must be made to attempt differentiation between the Δt's. In our system all the delays are stored and an average is calculated every time a new delay comes in. In the case of a binary encoding, this average delay (basically the sample mean) is used as a cutoff to designate the delays as either zeros (when they are smaller than the mean) or ones (when they are larger than the mean). The estimated bit sequence is re-evaluated for each new delay that comes in, and therefore can change as the mean is adjusted. Generalization to the case of multiple symbols codes is straightforward.

4 Multi-Symbol Channel

A skillful attacker would not choose a random distribution on the delays, however, but would try to pick a distribution that maximizes the Shannon channel capacity. The Shannon capacity, as described in Appendix 8.1. We define *the best coding system* as one that attains the Shannon limit. This is the base of the detection algorithm that we propose. We want to guess, based on the network characteristics, the coding system that the attacker would use, and then analyze the emitted symbols to see if they match such a distribution. If that is the case, we assume that covert communication is taking place. A well known technique used to find an input symbol distribution that maximizes the channel capacity is the Arimoto-Blahut algorithm [4].

We assume that an experienced enemy has gained control over one of our internal machines. He/she starts by conducting an investigation of the network situation around our machine with the intent of discovering the optimal parameters for a covert exfiltration channel. During this first stage the channel will not be utilized in the most efficient way possible. The enemy intends to find the input sequence distribution that can exfiltrate the highest amount of information per time-slice.

A (discrete memoryless) channel can be described by a set of transition probabilities summarized in a transition matrix. Assume that the rows represent the symbols of a discrete input alphabet $X = x_1, \ldots, x_N$ and the columns represent the symbols of a discrete output alphabet $Y = y_1, \ldots, y_N$. The probability of receiving y_i under the condition that x_j was sent is $p(y_i|x_j)$ and this represents the value in cell $[i, j]$ of the matrix. Note that the number of rows does not have to be equal to the number of columns.

The entries of the matrix can be determined experimentally. Let us assume that some previous experiments showed that the delays should be in the range of milliseconds. At the sender side a certain number of packets with

	$\leq$ 1.5	1.5-2.5	2.5-3.5	3.5-4.5	4.5-5.5	5.5-6.5	6.5-7.5	7.5-8.5
1	.05	.02	.91	.02	0	0	0	0
2	0	0	.12	.86	.01	0	.01	0
3	0	0	.01	.02	.96	.01	0	0
4	0	0	0	.01	.04	.94	.01	0
5	0	0	0	0	0	.05	.94	0.01
6	0	0	0	0	0	.08	.9	.02
7	0	0	0	0	0	0	0.07	.93

Table 1: Channel Matrix. The value [i,j] represents the probability of receiving delay j when delay i was sent.

a certain delay are sent. At the receiver's end the delays are measured. This procedure is then repeated and different delays are sent thought the channel, yielding Table 1.

This particular table was derived from experiments that we ran at 1 pm during a work day in the month of March 2005 over a distance of 24 hops. We sent 1000 specific delays and measured what was received. The left-hand column gives the delays that were transmitted (in 100^{ths} of a second), and the subsequent columns give the probabilities that a delay was received in a given time bracket. For instance, the first row concerns packets sent with a 10ms inter-packet delay. 5% of the packets were received with a delay of 1.5 ms or less, 2% with a delay of between 1.5 ms and 2.5 ms, and so on.

Once such a table is completed the Shannon capacity of the represented discrete memoryless channel can be estimated through the Arimoto-Blahut algorithm, (Appendix 8.2). We implemented this algorithm in Java and ran it on the above channel to find its capacity $C = 2.14$ Bit/Symbol attained by the following distribution on the input symbols:

1 ms	2 ms	3 ms	4 ms	5 ms	6 ms	7 ms
.18	.171	.179	.175	.109	.104	.082

The ideal situation would be observing network traffic with this exact distribution. In such a case we can assert a covert channel is in place. Unfortunately there are at least two difficulties with this approach. First, the optimal input delay distribution may not be unique. Second, the matrix is not constant over time (its entries depend on network traffic) and so it has to be periodically updated.

We conducted experiments to characterize these time-varying matrices. We found, for example, that these matrices are banded (nonzero along a constant stripe of diagonals around the main diagonal). This is reasonable since we expect that the majority of received delays are close to the delay that was sent.

We believe that the banded structure can be exploited to determine optimal input probability distributions,(i.e., attaining the Shannon Capacity) which can be matched by a feasible channel coding scheme. These distributions are in fact the true candidates for the enemy. An example by the noisy typewriter that is represented by a quasi-banded stochastic matrix and whose optimal input distributions include both the uniform distribution and the distribution that is zero on $x_2, x_4, \ldots, x_{2k}$ and uniform on $x_1, x_3, \ldots, x_{2k-1}$. The second distribution in this particular case automatically gives a 1-block channel code that has zero error probability in decoding.

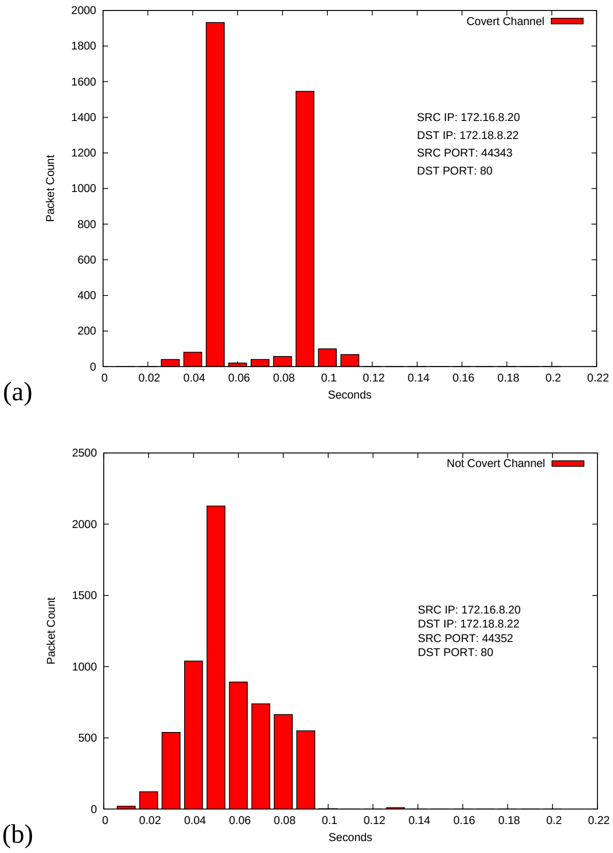

Figure 2: The figures show the number of packets received with a given delay. The horizontal axis shows the inter-arrival time in seconds, and the vertical axis shows the number of packets received. In case (a) the two spikes show that a covert channel communication is in place. Case (b) represents normal communication.

In practice, we monitor outbound communications with the goal of quantifying the distribution of input symbols. Basically, this is done by generating a histogram of the packet inter-arrival times. Individual communications are separated and tracked based on:

- Source IP address
- Destination IP address
- Source port
- Destination port

- Protocol

If no transmissions with the same characteristincs happen in an interval of time ΔT the communication is considered closed. Any time an instance is closed a histogram of the delays is plotted. The current implementation has $\Delta T = 5$ minutes. Examples of such graphs are in Figure 2. So ideally we would have:

1. A network monitor that gives us the actual interpacket delay distribution for each active communication.

2. The Arimoto - Blauth algorithm that calculates the best input delays distribution, hopefully the one used by the malicious intruder.

3. A Similarity Tester that compares the two probability distributions on input symbols, one obtained monitoring the network traffic, and one obtained maximizing the channel capacity.

Unfortunately there are many problems with this approach. To determine the optimal channel input distribution with the Arimoto-Blahut algorithm, we must start with the error matrix. Since the creation of such an error matrix depends on knowledge of the input symbol distribution, it is not obvious that such a matrix can be kept dynamically in real-time. To exactly compute the errors in reception, we must know where the attacker machine is located, which is not the case. We can only calculate the errors at our own border, which we may assume is the closest an attacker may ever be.

Fortunately, the enemy faces the same problem. Since the characteristics of a network change constantly (e.g., delays, congestion, etc.), the attacker will not be able to choose an input distribution that maximizes the channel capacity for all cases. So we can assume that he/she wants to use an input distribution that works well for different possible channels. For example it is intuitive that at different times of the day the error matrix is different, due to varying traffic levels.

It is also intuitive that machines close to each other lead to a less noisy channel. The only advantage that the attacker has with respect to us is that he knows exactly how many hops are between the two machines. For this reason we must assume the attacker is right at our border.

Our approach stores network information, over time, for each active communication. Since the characteristics of the network changes with time of day, congestion, and other parameters, we are essentially dealing with a large collection of channels. We assume that if the number of hops between two machines is the same, they generate the same channel matrix at the same time of day no matter their position in the network.

We could build many different channel matrices, each one representing different times of day and different hop counts. For example, we could monitor the traffic every four hours, and each time analyze the cases of 2, 5, 10, 15, 20, 25, 30 hops. In a multidimensional space, this produces $7 \cdot 6 = 42$ possibilities.

The goal then is to find the best input symbol distribution that combines the properties of these channels in such a way that covert communication is possible under all circumstances. In other words, if we have two channels, C_1 and C_2, we are looking for a definition of channel capacity that applies to both of the channels.

The most intuitive is the following. If C_1 has probability of occurrence p_1 and channel C_2 has probability p_2 ($p_2 = 1 - p_1$), a straightforward way to define the capacity of the composition of the two channels would be

$$C = \max_{p(x)} \left(p_1 \cdot I_{C_1}(x; y) + p_2 \cdot I_{C_2}(x; y) \right)$$

where $I_{C_i}(x; y) = H_{C_i}(x) - H_{C_i}(x|y)$ is the mutual information for channel C_i with $i = 1, 2$ and

$$H(x|y) = \sum_i \sum_j p(x_i, y_j) \log \frac{1}{p(x_i|y_j)}$$

The input distribution to consider is the one that achieves maximum capacity.

Although the above techniques are promising, they do, however, depend on our assumptions that the attacker tries to achieve the maximum capacity.

5 Binary Channel

In these experiments we focus on codes with only two input symbols. Even in the binary case, before saying anything useful about detecting covert channels, we must investigate network properties. We concentrate on the error rates and the channel capacity. For testing of our algorithm we implemented a version of a covert channel with a self-calibrating delay loop in the sender. This means that the sender automatically adjusts sleep times for operating system overhead and load to ensure that the sender's timing is accurate. An arbitrary string of bytes was sent and then verified at the receiver's end. Error rates were measured for zeros being received as ones, and ones being received as zeros.

Experiments were run where the receiver was 4 hops, and 24 hops away. The error rate for 4 hops was mostly dependent on system load rather than network latency differences. Therefore all 4-hop data was discarded.

Over the course of multiple days various predetermined sequences were transmitted and the error rates were recorded. The difference between Δt_0 and Δt_1 was

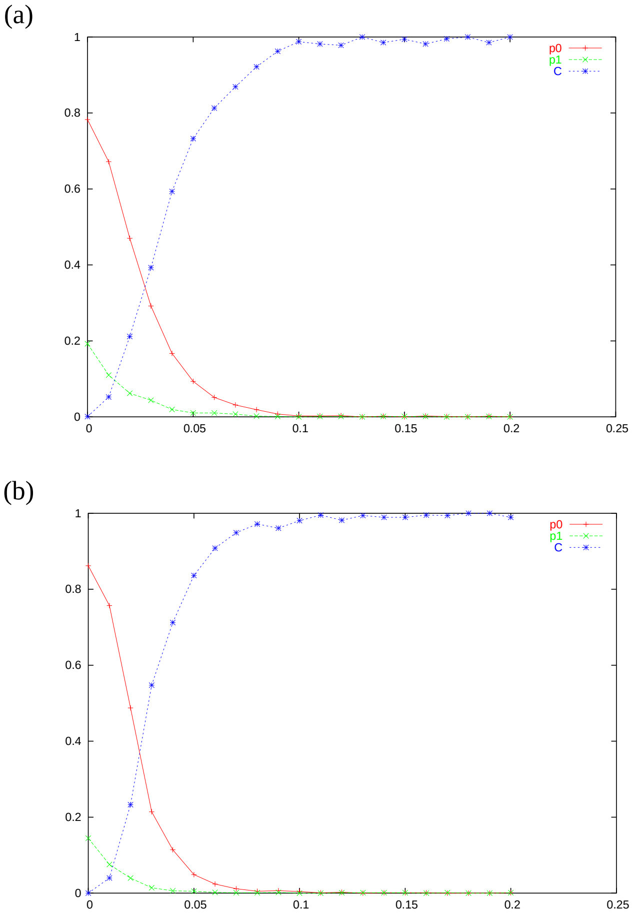

Figure 3: Error rates for zeros being received as ones (p_0), and ones being received as zeros (p_1) and channel capacity (C) for non-symmetric binary channel, where Δt_0 is constant and Δt_1 increases. Horizontal axis shows $\Delta t_1 - \Delta t_0$ in seconds. *(a)* $\Delta t_0 = 0.2$s, $0.21 \leq \Delta t_1 \leq 0.4$s *(b)* $\Delta t_0 = 0.4$s, $0.41 \leq \Delta t_1 \leq 0.6$s

varied from 0.01 second to 0.2 seconds. The graphs in Figure 3 show the error rates p_0 and p_1 that were measured, where p_0 is the fraction of zeros that were received as ones, and p_1 is the fraction of ones that were received as zeros. In the two graphs two different values of Δt_0 are used.

There is no error-free communication over a noisy channel when messages are encoded with zero redundancy. The amount of redundancy that must be added to achieve error-free communication depends on how noisy the channel is, and can be measured with the Shannon Channel Capacity. The channel capacity represents the highest amount of information per symbol (Bit/Symbol) that can be transmitted through the given noisy channel, and for a Binary Symmetric Channel (BSC) is

$$C = 1 - \left[P_e \log \frac{1}{P_e} + (1 - P_e) \log \frac{1}{1 - P_e} \right]$$

where P_e is the error probability [4]. For example, if we observe a train of time delays that forms a "random" (incompressible) binary sequence, we would expect that the number of zeros is about the same as the number of ones. If $C < 1$, however it would be impossible for the intruder to convey any information (with a negligible error probability) transmitting at a rate of 1 Bit/Symbol. In other words, in the presence of noise, the intruder will in general be forced to adopt some form of channel codification with a rate necessarily below C. This redundancy is well known in almost any form of digital communication; popular schemes include parity bits, Cyclic Redundancy Check (CRC) codes, and Error Correcting Codes (ECC). When the channel capacity falls far below 1, however, the amount of redundancy that the intruder is forced to use becomes impractical. Intuitively this is also the solution to preventing this type of covert channel, namely by artificially varying packet delays at the border (be it router or firewall) to force C down so far that successful transmission of data will go too slow, and be too unreliable.

Intuitively our binary channel is not symmetric. Since the inter-packet delays are different for zeros and ones, their respective transmission rates are different as well. This leads to the expectation that the error rates also must be different, and this can be observed from Figure 3 (p_0 and p_1 are different). In other words, it takes less time to transmit 100 zeros than it takes to transmit 100 ones. This means that the rate at which zeros are transmitted is higher than the rate at which ones are transmitted. (The fact that zeros and ones are intermixed in our communication does not matter; the results would have been the same if all the zeros were transmitted first, and then all the ones.) Assuming that the channel does not care whether zeros or ones are transmitted, the error rate is expected to go up as the transition rate goes up. The error rate is higher for those symbols that are associated with shorter delays since those delays are more sensitive to being received erroneously. This explains why the error rate p_0 (zeros being incorrectly received as ones) is much higher than the error rate p_1 (ones being received as zeros). As the difference $\Delta t_1 - \Delta t_0$ increases the error curves both go down, since it becomes easier to distinguish between zeros and ones. Given error rates p_0 and p_1 the channel capacity for a non-symmetric binary channel becomes:

$$C = \log \left(1 + 2^{\frac{H(p_0) - H(p_1)}{p_1 + p_0 - 1}} \right) + \frac{(1 - p_0) H(p_1) - p_1 H(p_0)}{p_1 + p_0 - 1}$$

where

$$H(x) = -x \log x - (1-x)\log(1-x)$$

The blue lines (with $*$) in the graphs in Figure 3 show the channel capacity for a non-symmetric binary channel, based on the measured error probabilities.

Finally we must make some observations about the experiments. First, the time of day and network load can seriously affect the accuracy of the transmission. A congested network forces the sender to adopt larger differences between Δt_0 and Δt_1, thus bringing down transmission speed. Second, the path that the packets traverse is of direct impact on the differences in inter-packet latencies. Not all 24 hops anywhere on the Internet will give the same graph. Larger backbones tend to have faster switching hardware, keeping differences in latencies to a minimum. We found that crossing oceans has the most profound impact on covert channel transmission speeds, often lowering them by a factor of 10. Likewise, rate limiting and quality-of-service queueing may at times completely distort the channel, while at other times allowing flawless transmission. This said, it must be realized that graphs 3(a) and 3(b) teach us more about the expected shape and best-case scenario, than they tell us about what can be expected in general. The observed differences between Δt_0 and Δt_1 (horizontal axis) would certainly incur much higher error rates using same Internet paths.

5.1 Statistical Detection

In the case of a binary input distribution, we developed an approach based on statistical analysis. It can be extended to a higher even number of symbols. In such case, the detection becomes harder.

This method is based on the assumption that (for a binary channel) the inter-packet delays will center around two distinct values, ie. two distinct delays (Figure 2a) while in a normal communication, where the delays are more or less random the Δt's are spread around a single spike, (Figure 2b). We take advantage of this characteristic to make some statistical comparisons between a covert communication channel and normal network traffic.

Looking at Figure 2 the difference between a binary covert channel and regular traffic becomes evident; two spikes versus one large spike. For the covert channel the sample mean μ (average) of the inter-packet delays will be somewhere between the two spikes. The packet-count in the histogram at that point will therefore be very low. Looking at a normal traffic pattern, however, the mean of the inter-packet delays will be in the center of the large spike. The packet-count at the mean will thus be very high, if not the highest. If we divide the packet-count at the mean by the maximum packet-count from the histogram, we get a measure of how likely it is that the communication is a covert channel. In particular the smaller the ratio $\frac{C_\mu}{C_{max}}$, the higher the probability of having a covert channel communication. We can therefore define a probability:

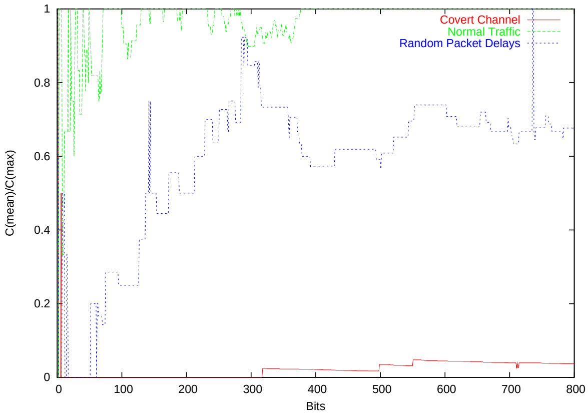

Figure 4: The ratio between the mean packet count and the maximum packet count for normal traffic, fully random delays, and a binary covert channel. The horizontal axis shows the length of the estimated sequence in bits, and the vertical axis shows $\frac{C_\mu}{C_{max}}$.

$$P_{CovChan} = 1 - \frac{C_\mu}{C_{max}}$$

where C_μ is the packet-count at the mean and C_{max} is the maximum packet-count of the histogram. Experiments with three different types of data were conducted, and Figure 4 shows the ratio $\frac{C_\mu}{C_{max}}$ for these experiments.

1. **Normal Data.** Packets with an average delay of 0.2 seconds were transmitted. The inter-arrival times vary, but the spike is at 0.2 seconds. The sample mean μ is therefore represented by a delay very close to 0.2 and the number of packets with exactly that delay, C_μ is very high. The ratio between this number and the histogram maximum C_{max} quickly grows to 1.0, and stays there as more packets are transmitted. Normal traffic is very bursty and inter-packet delays are often dependent on how quickly acknowledgements or responses are returned, which is once again dependent on the distance (and system load) of the two systems communicating. The packet delays therefore center mostly around a single value (0.2 seconds in this case) with occasional outliers (usually on the order or 120 seconds or more).

2. **Random Data.** Packets are sent with a fully random delay. Although this is not realistic for traffic encountered on the network, it does present a good idea of the worst case scenario. Initially, when only a few bits have been sent, the delays scatter across the range, and it is unlikely that the sample mean will have a high count. That explains why until approximately the first 10 bytes have arrived, the ratio $\frac{C_\mu}{C_{max}}$ remains zero. Later on, as more packets arrive, the histogram starts to even out, and the ratio starts to crawl up. As the number of transmitted packets goes up even further, the ratio keeps growing until it eventually hits 1.0 as the packet count goes to infinity.

3. **Covert Channel Communication.** Two delays are used, thus the inter-arrival times concentrate around those two values. The sample mean μ lies approximately in the middle between the two spikes. The count C_μ is low and therefore the ratio $\frac{C_\mu}{C_{max}}$ is approximately zero. As more and more bits are transmitted over the covert channel the spikes increase in size, but the ratio always remains very close to zero.

Our algorithm detects the sequence that most likely represents the covert communication channel by analyzing the value $\frac{C_\mu}{C_{max}}$. The lower that value the higher is the probability of having a malicious communication hidden in inter-packet delays.

Intuitively the relative height of C_μ is directly related to the channel capacity. Consider that, for an approximately equal distribution of input symbols, C_μ forms the cut-off point between zeros and ones; if a received Δt is smaller than the mean a zero is received, otherwise a one is received . Therefore, the more packets that are counted exactly around the mean, the “higher” the confusion. In other words; peak for zeros and the peak for ones overlap. The higher C_μ, the larger the overlap. Confusion, or overlap, means that zeros are incorrectly received as ones, and vice versa. A bigger overlap means higher error rates and a reduced channel capacity. So as the ratio $\frac{C_\mu}{C_{max}}$ grows, the channel capacity drops.

Furthermore, we know that when the peaks are perfectly separated the error will be zero, and therefore the channel capacity will be one. Also $1 - \frac{C_\mu}{C_{max}}$ will be one. Conversely, when the peaks are exactly overlapping $1 - \frac{C_\mu}{C_{max}}$ will be zero, and there will be no distinction between zeros and ones, meaning that the channel capacity will be zero also. The intuitive expectation that $1 - \frac{C_\mu}{C_{max}}$ and the channel capacity are closely related is confirmed by the results shown in Figures 5.

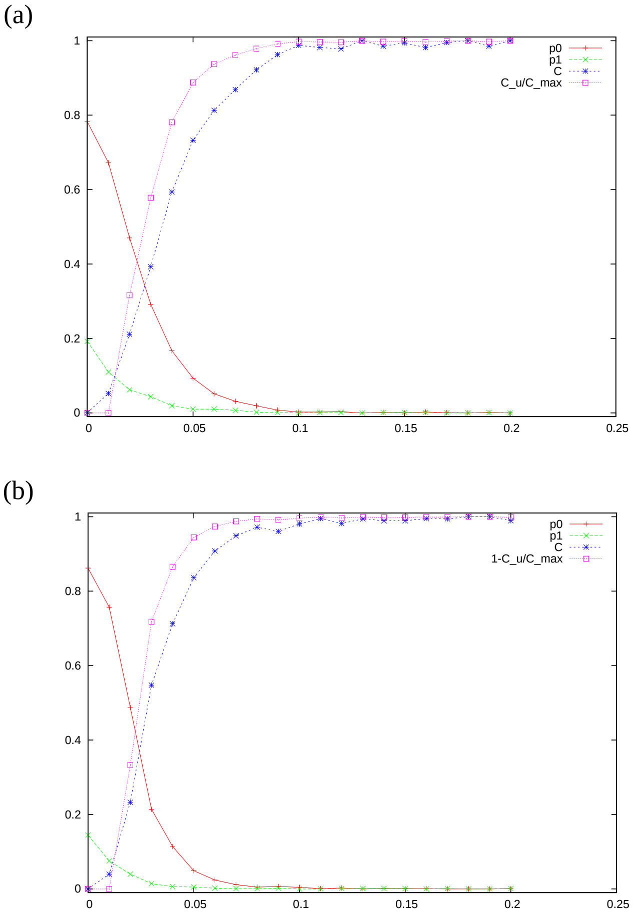

Figure 5: Error rates for zeros being received as ones (p_0); and ones being received as zeros (p_1) and channel capacity (C) for non-symmetric binary channel, where Δt_0 is constant and Δt_1 increases. Horizontal axis shows $\Delta t_1 - \Delta t_0$ in seconds. *(a)* $\Delta t_0 = 0.2$s, $0.21 \leq \Delta t_1 \leq 0.4$s *(b)* $\Delta t_0 = 0.4$s, $0.41 \leq \Delta t_1 \leq 0.6$s

Let us conclude this paragraph with a note on TCP communications. TCP is a stateful protocol that sends keep-alive packets every 4 minutes in case there is no other traffic flowing over the connection. The histogram graphs for TCP connections often reveal bursts (big spikes) around 0.1 or 0.2 seconds, and a small spike around the 4 minutes. Depending on its duration, therefore, a session might inadvertently be classified as a covert channel (the mean lies between the 0.1 and 240 seconds, and there are two peaks). The implementation of our algorithm takes into account that even if this were a covert channel, the delay for one of the symbols would be too long to yield serious bandwidth. Since this case occurs quite frequently, a suspected covert channel where one of the symbols is over 3 minutes in duration is discarded.

6 Conclusion and Future Work

In this paper, we have introduced two methods for detecting timing covert channels. The first one, based on Information Theory concepts, requires a thorough understanding of the network situation and a highly skilled attacker. The second technique, instead, is based on the simple idea that if an even number of input symbols is used, the number of packets at the mean delay is very low compared to the maximum number of packets for any given delay. The statistical analysis of the inter-packet delays does a good job of classifying between regular network traffic and traffic that is communicating through modulation of the inter-packet delays. The algorithm works in a case of code with an even number of delays (symbols) and performs best when the code is binary. Finally, our algorithms seem to perform very well in an experimental test-bed, but have not been tested in heavily utilized networks. The false positive, and false negative rates should be investigated empirically.

Although we investigated a timing channel that uses packet inter-arrival times to encode covert messages, there are other ways of transmitting data undetected. For instance, consider the possibility that the attacker uses two external machines. If he/she receives a packet in machine A a binary 0 is read, and if he/she receives a packet in machine B a binary 1 is read. Systems A and B must communicate to correctly re-order the bitstream. Techniques for detecting and disrupting such a channel are likely to be very different from the techniques we described here. Future work will focus on more generalized detection schemes and broadening our scope to other types of covert channels.

7 Acknowledgments

Supported under ARDA P2INGS Award No. F30602-03-C-0248. Points of view in this document are those of the authors and do not necessarily represent the official position of ARDA.

References

[1] V. Anantharam and S. Verdú. Bits through Queues. *IEEE Trans. on Information Theory*, 42(1):4–18, Jan 1996.

[2] R. E. Blahut. Computation of Channel Capacity and Rate-Distortion Functions. *IEEE Trans. on Information Theory*, IT-18(4):460–473, Jul 1972.

[3] S. Cabuk, C. Brodley, and C. Shields. IP Covert Timing Channels: Design and Detection. *Proc. of the 11th ACM conference on Computer and Communications Security*, 2004.

[4] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley Series in Telecommunications. John Wiley & Sons, New York, NY, USA, 1991.

[5] J. Giles and B. Hajek. An Information-Theoretic and Game-theoretic Study of Timing Channels. *IEEE Trans. on Information Theory*, 48(9):2455–2477, Sept 2002.

[6] D. K. Kamran Ahsan. Practical Data Hiding in TCP/IP. *Proc. Workshop on Multimedia Security at ACM Multimedia*, 2002.

[7] R. A. Kemmerer. Shared Resource Matrix Methodology: An Approach to Identifying Storage and Timing Channel. *ACM Transaction on Computer Systems*, 1(3):256–277, Aug 1983.

[8] R. A. Kemmerer. Shared Resource Matrix Methodology: An Approach to Identifying Storage and Timing Channel : Twenty years later. *Proc. 18th Annual Computer Security Applications Conference (ACSAC)*, pages 109–118, 2002.

[9] B. W. Lampson. A Note on the Confinment Problem. *Proc. of the Communication of the ACM*, 16(10):613–615, Oct 1973.

[10] I. S. Morskowitz and A. R. Miller. The Channel Capacity of a Certain Noisy Timing Channel. *Proc. IEEE Transaction on Information Theory*, 38(4):1339–1344, 1992.

[11] I. S. Morskowitz and A. R. Miller. Simple timing channels. *Proc. IEEE Computer Society Symposium on Research in Security and Privacy*, pages 56–64, 1994.

[12] I. Moskowitz, R. Newman, D. Crepeau, and A. Miller. Covert channels and anonymizing networks. *Proceedings of the Workshop on Privacy in the Electronic Society (WPES 03), Washington, DC, USA.*, October 2003.

[13] I. S. Moskowitz and M. H. Kang. Covert Channels Here to Stay? *Proc. of COMPASS*, pages 235–243, 1994.

[14] N. Ogurtsov, H. Orman, R. Schroeppel, S. O'Malley, and O. Spatscheck. Covert Channel Elimination Protocols. *Technical Reports TR96-14. Department of Computer Science, University of Arizona*, 1996.

8 Appendix

8.1 Shannon Capacity

The channel capacity of a Discrete Memoryless Channel is defined as

$$C = \max_{P_X} I(X;Y) = \max_{P_X} H(X) - H(X|Y)$$

where P_X is a probability distribution on the input symbols and $I(X;Y)$ is the mutual information between X and Y which is a measure of the dependance between the two random variables.

8.2 The Arimoto-Blahut Algorithm

Let us see how the Arimoto-Blahut algorithm works. We rewrite the definition of channel capacity in the following way [4]:

$$C = \max_{r(x)} I(X;Y)$$

$$C = \max_{r(x)} \sum_x \sum_y r(x)p(y|x) \log \frac{r(x)p(y|x)}{r(x)\sum_w r(w)p(y|w)}$$

that written as a double maximization becomes:

$$C = \max_{q(x|y)} \max_{r(x)} \sum_x \sum_y r(x)p(y|x) \log \frac{q(x|y)}{r(x)}.$$

where $q(x|y) = \frac{r(x)p(y|x)}{\sum_w r(w)p(y|w)}$ The optimal input distribution which is then used as the basis for the next iteration is

$$r(x) = \frac{\prod_y q(x|y)^{p(y|x)}}{\sum_w \prod_y q(w|y)^{p(y|x)}}.$$

Let us indicate with R is the number of emitted symbols (number of rows of the error matrix) and T is the number of received symbols (number of columns of the error matrix). We want to find $r(x) = r(x_1), \ldots, r(x_R)$ that maximizes the capacity. Let's fix ϵ. The Arimoto-Blahut algorithm can be summarized in the following three steps.

1. **Initialization.** The initial distribution can be any distribution on the input symbols. We can choose for example:

$$r^{(0)}(0) = 1, r^{(0)}(1) = 0, \ldots, r^{(0)}(R) = 0.$$

2. **Recursion.** For $j = 0, \ldots, T-1$ compute

$$D(y_j) = \sum_{i=0}^{(R-1)} r^{(k-1)}(x_i)p(y_j|x_i)$$

Then for $i = 0, \ldots, R-1$ and for $j = 0, \ldots, T-1$ compute:

$$q(x_i|y_j) = \frac{r^{(k-1)}(x_i)p(y_j|x_i)}{D(y_j)}$$

Finally for $i = 0, \ldots, R-1$ compute:

$$r^{(k)}(x_i) = \frac{\prod_{j=1}^{(T-1)} q(x_i|y_j)^{p(y_j|x_i)}}{\sum_{i=1}^{(R-1)} \prod_{i=1}^{(T-1)} q(x_i|y_j)^{p(y_j|x_i)}}$$

3. **Termination.** A reasonable point where to stop the procedure is when two sequential iterations of the algorithm give a very similar result.We repeat the recursion until:

$$\frac{1}{2} \sum_{i=0}^{(R-1)} |r^{(k-1)}(x_i) - r^{(k)}(x_i)| \leq \epsilon.$$

It is shown [2] that the algorithm converges. But the solution is not unique.