EVADING EXISTING STEPPING STONE DETECTION METHODS USING BUFFERING

by

MADHU VENKATESHAIAH

Presented to the Faculty of the Graduate School of

The University of Texas at Arlington in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2006

To my Mom Padma and Dad M.R. Venkateshaiah without whose support I wouldn't be where I am today.

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Matthew Wright for giving me a chance to work with him in this research project. My association with him has been an invaluable experience and has given me a new perspective towards research in general and Network Security in particular. I am indebted to him for his patience towards me during my temporary failures and shortcomings and for inspiring me to overcome them. I would also like to thank Dr. Wright for providing financial support for my studies.

I am grateful to Dr. Hao Che and Dr. Donggang Liu for their interest in my research and for taking time to serve on my thesis committee.

I wish to thank all the members of iSEC lab at UTA for their help and support.

I would like to thank all my friends for their support and encouragement.

Most importantly I would like thank my parents who dedicated their whole life for the betterment of my future.

November 22, 2006

ABSTRACT

EVADING EXISTING STEPPING STONE DETECTION METHODS USING BUFFERING

Publication No. _____

Madhu Venkateshaiah, M.S.

The University of Texas at Arlington, 2006

Supervising Professor: Dr. Matthew K. Wright

Network based intrusions have become a serious treat to the users of the internet. To gain anonymity and complicate their their apprehension, attackers launch attacks not from their own systems but from previously compromised systems called *stepping stones*. Attackers do this by establishing a chain of connections using protocols like Telnet or SSH. The stepping stones could be located in different autonomous domains or even in different countries. This makes it very difficult to trace an attack back to its origin. The owner(administrator) of the compromised system is not even aware of the fact that attacks are being launched from his computer. But for an external observer, the attacks seem to originate from the stepping stone(compromised system), which may lead to the prosecution of the owner of the system. So it is important for administrators to detect if a system on their network is being used as a stepping stone.

An effective way to detect stepping stones is by comparing of incoming and outgoing connections in a network to find correlations. Content based correlation is not effective because of the use of encrypted communication protocols like SSH. So we have to turn

to other aspects of connections/traffic for correlation. The use of timing characteristics of traffic for correlation has been explored and many *Passive* 2.3.1 and *Active* 2.3.2 approaches to correlate connections have been proposed. The attacker can attempt to break correlation between traffic streams by delaying and dropping packets and adding dummy packets(chaff) to the streams. Though earlier approaches to stepping stone detection do address these issues to a certain extent, they make certain assumptions about the capabilities of the attacker that does not reflect reality and give rise to a weak attacker model.

For the sake of simplicity, earlier approaches ignore the fact that an attacker can add dummy packets (chaff) to a traffic stream. But in reality, an attacker might have total control over all the stepping stones on the connection chain. This makes it easier for the attacker to install rogue applications on the stepping stone or even modify the operating system. In this scenario, an attacker can modify applications to use cover traffic and introduce delays to make the incoming and outgoing streams look very different thus making correlation very hard. The attacker can also use constant rate cover traffic to break correlation.

We loosen some assumptions made by earlier researchers and assume that an attacker can add cover traffic to traffic streams. We propose a simple buffering technique that could be used by an attacker on a stepping stone to evade or severely degrade detection. In our technique, packets are buffered, selectively dropped and chaff packets are added to generate constant rate traffic. To test our technique we need a correlation scheme that can correlate constant rate streams. Wang, Chen, and Jajodia [9] proposed a watermark based correlation scheme to track VoIP calls on the internet. This scheme is designed to correlate constant rate traffic streams like VoIP. To test the effectiveness of our technique to evade detection, we choose this scheme for our simulations and show that our buffering technique can successfully evade detection.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Stepping stones

Hackers attack computers to compromise security and gain control of systems, compromise privacy or to create a disruption so as to deny or degrade access to legitimate users. It is being increasing recognized that detecting and preventing such attacks is an important requirement of computer systems. With the emergence of computer networks and the Internet, the problem of detection has been compounded. The biggest advantage of the Internet from an attackers perspective is that he can be located anywhere in the world and launch an attack on a system located at any other part of the world. To further compound the problem of detection the attackers can launch attacks indirectly by relaying their attack through a chain of intermediate (previously compromised) systems called *stepping stones*. The attacker does this by constructing a chain of interactive connection using protocols like Telnet or SSH. The commands that the attacker types on his local terminal are relayed through the stepping stones until they reach the victim. This gives the attackers a certain a level of anonymity, and makes it hard for an investigator to traceback the attack to its origin.

1.2 Stepping stone detection

The fact that stepping stones maybe in different administrative domains or countries, further increases the difficulty of tracing the attack to the point of origin. The problem of detecting stepping stones in not only important to an investigator but also

to network/security administrators. Detecting stepping stones has many benefits for the administrators of networks:

- To flag suspicious activity
- To avoid prosecution incase a break-in is detected as having come from their local site
- To detect insider attack that are relayed through external hosts

Since the attacker is sending attack traffic to a stepping stone and the stepping stone is just relaying (forwarding) it to an external host, the stepping stone detection problem boils down to finding an outgoing connection with the same characteristics as an incoming connection or vice-versa. An intuitive approach to solve this problem would be to compare the contents of the incoming and outgoing packets in a network to find packets with the same content. But the development of encrypted communication protocols like *SSH*, have made this approach ineffective. So we need to use other characteristics of the traffic like timing characteristics to detect stepping stones. One of the promising approaches to solve the detection problem is Active Perturbation (see Section [2.3.2]). In this approach, timing based watermarks are embedded into incoming connections and the outgoing connections are searched for these watermarks.

1.3 Contribution

Earlier active perturbation approaches to stepping stone detection(see Section [2.3.2]) make certain assumptions about the what the attacker can do on the stepping stone:

- The attacker does not use cover traffic
- Though the attacker can introduce perturbations by adding delays, the maximum delay that the attacker can add is bounded
- When the attacker adds huge delays on the stepping stone, such delays can be detected as anomalies

These assumptions lead to weak attacker model that does not reflect reality. In reality, the attacker has total control over the stepping stones. So he can change the configuration of the system and install rogue applications. In our work we loosen some of the assumptions made by earlier approaches. Our contribution is a simple technique that an attacker can use on the stepping stone to evade detection. Our technique uses buffering of incoming packets, selective dropping and adding chaff to generate traffic that has very different characteristics than the incoming traffic. We finally show that with very little buffering delay, packet drops and chaff our technique is effective in severely degrading detection.

1.4 Thesis Organization

In Chapter 2, we discuss the background concepts and related work to justify the design decisions explained in our description of the system. In Chapter 3, we present our proposed buffering technique in detail. In Chapter 4, we explain the simulation setup and aspects of the system that we tested. Section 4.5 presents the results of the simulations and their interpretation. Chapter 5 concludes with ideas for future work.

CHAPTER 2

BACKGROUND

In this chapter we formally define a *stepping stone* and present some background on the techniques that have been used to detect stepping stones. We also introduce some attacks on stepping stone detection and some possible defence against them. The terminology used in this document are also introduced in this chapter.

2.1 Stepping Stone

When a person logs into one computer and from there logs into another computer and so on to perhaps a number of computers, we refer to the sequence of logins as a *connection chain* [1]. Any intermediate host on a connection chain is called a *stepping stone*. Any two connections that are part of a connection chain are called a *connection pair*. The connections that appear earlier in the connection chain are said to be *upstream connections* to those that appear later on in the chain. Similarly, the connections that appear later on in the chain are said to be *downstream connections* to those that appear earlier on in the chain. Upstream connections are closer to the attacker. Generally in the Internet connection chains are formed by network attackers by using terminal emulation programs like telnet and SSH. The stepping stones thus formed are called *interactive stepping stones* since the attacker types in commands and waits for a response. Though it is possible for a computer program to create a connection chain, we limit the scope of our research to interactive stepping stones.

2.2 Traceback

Intruders use long chains of connections through stepping stones to attack their targets. The stepping stones through which the attacker relays his/her connections may be located in different countries and in networks operated by different network operators. This obscures the identity of the attackers and gives them a certain level of anonymity. Tracing an intruder to the origin of the attack requires that we follow the connection chain and examine each host on the chain to determine its predecessor in the chain. To further complicate traceback, attackers usually delete logs on stepping stones. Since the stepping stones could be in different countries, it takes a lot of time and effort to get the governments and network operators of the countries to cooperate in the investigation. Many approaches the solve the tracing problem have been proposed and can be broadly classified into categories:

- Host-based
- Network-based

Host-based approaches [2] [3] setup and use some kind of tracing components at each host. The major drawback of these host-based systems is that the system would fail if the system is not used on a particular host or has been modified by an attacker. It also requires that all autonomous systems in the internet use a particular tracing mechanism on all hosts. On the other hand, *network-based* approaches [1] [4] [5] require that some tracing component be setup in the network infrastructure. The advantage of this approach is that it does not require the participation of monitored hosts nor does it place the trust on monitored hosts. All network based tracing mechanisms use correlation of network connections to trace the intruder to the origin of the attack. From now on we only focus on network based approaches.

2.3 Stepping Stone Detection

Stepping stone detection can be defined as a process of observing all incoming and out going connections in a network and determining which ones are a part of a connection chain. This problem is closely related to the problem of *traceback* or tracing intruders through the internet by following the connection chain. In both the above applications, the fundamental underlying problem is to compare and analyze two connections and determine if there is any correlation between them. This problem has been studied by number of researchers and solutions have been proposed. Two main approaches proposed to detect stepping stones are:

- Passive Monitoring
- Active Perturbation

2.3.1 Passive Monitoring

Passive monitoring is an approach where the characteristics of traffic streams are analyzed to construct correlations between them. The interactive stepping stone problem was first formulated and studied by Staniford and Heberlein [1]. They proposed a content based algorithm that creats thumbprints of streams and compares them, looking for good matches. The problem with this approach is that it is content based and assumes that the traffic is un-encrypted. Zhang and Paxson [4] were the first to propose a scheme to correlate traffic across stepping stones even if the traffic is encrypted by the stepping stone. The method is based on correlation of the ends of OFF periods (or equivalently the beginnings of ON periods) of interactive traffic, rather than the connection contents. Since this method only uses the timing information of packets it can be applied to encrypted traffic. Yoda and Etoh [5] proposed a deviation-based approach for correlation of streams. They define the minimum average delay gap between the packet streams of two TCP connections as the deviation. The deviation based approach considers both

the packet timing characteristics and their TCP sequence numbers. It does not require clock synchronization and is able to correlate connections observed at different points of network. Wang, Reeves and Wu [6] address the problem of correlation by a scheme based on inter-packet timing characteristics. While timing-based correlation approaches have the advantage that they are simple and do not disturb normal traffic, they are vulnerable to countermeasures by the attacker. The attacker can perturb the timing characteristics of connections by selectively or randomly delaying packets at the stepping stone [7]. This kind of perturbation adversely affects the effectiveness of timing-based correlation.

2.3.2 Active perturbation

A promising defense against the problem of random timing perturbation by an attacker is active perturbation. In this approach, an incoming connection is perturbed by inducing a packet loss or delay and the outgoing connections are checked to see if the perturbation is echoed in them. Since the attacker does not know what the perturbation is, he/she will not be able to effectively degrade correlation by random perturbation. Wang and Reeves [8] proposed the first active watermark-based correlation method that is designed to be robust against random timing perturbation. In this scheme, a unique delay-based watermark is embedded into a traffic flow by slightly adjusting the timing of selected packets in the flow. The watermarked flow can be uniquely identified and thus correlated with other flows in the connection chain. We provide further details on this correlation scheme in Section 2.3.2.1. Wang, Chen, and Jajodia adapted the above watermarking technique to track VoIP calls on the Internet [9]. We provide further details of VoIP tracking in Section 2.3.3.Since VoIP streams have more stringent real-time constraints than TCP, [8] can not be directly used to correlate VoIP streams.

2.3.2.1 Watermark-based correlation

The objective of watermark-based correlation is to make the correlation of connections robust against random timing perturbations introduced by the attacker. The process of watermarking consists of two complementary processes:

- Embedding the watermark
- Decoding the watermark

A watermark is basically a unique binary string. The process of embedding one bit of the watermark consists of changing some property of a traffic flow such that the change represents watermark bit.

In the technique proposed by Wang and Reeves [8], the packets that are watermarked are randomly chosen and paired to obtain *inter-packet delays* (IPDs). A watermark is embedded by manipulation of these IPDs. The IPD between two packets p_a and p_b is $ipd_{(a,b)} = t_b - t_a$, where p_b is transmitted later than p_a and t_a and t_b are timestamps of p_a and p_b, respectively. IPDs are quantized for robustness. Given a *quantization step* S, the quantization function $q(ipd, S)$ rounds off ipd/S to the nearest integer. To embed one watermark bit w(0 or 1), ipd is slightly increased (by delaying the second packet by a small amount) so that the watermarked IPD, denoted as ipd^W, satisfies the condition $q(ipd^W, S)mod2 = w$. This means ipd^W is even multiples of S when 0 is embedded, and odd multiples of S when 1 is embedded. The watermark decoding function is $d(ipd, S) = q(ipd, S)mod2$. To make the scheme robust against random perturbations, multiple IPDs are used to embed one bit. In their paper the authors make the following assumptions:

1. While the attacker can add extra delay to any or all packets of an outgoing flow of the stepping stone, the maximum delay that he can introduce is bounded
2. The attacker does not know which packets are being watermarked

3. The component that decodes watermarks in traffic flows knows which packets have been watermarked

2.3.3 Tracking VoIP calls

In VoIP tracking, the objective is to determine who called a particular person or whom a person called. The problem boils down to finding correlations between the VoIP flows of the caller and the callee. This is similar to the Traceback and stepping stone detection problem where a set of connections need to be correlated. With the emergence of P2P VoIP applications like Skype [10] that offer end-to-end encryption and could be routed through low latency anonymous networks like TOR [11], content based approaches are not effective. Wang, chen, and Jajodia propose a watermark based technique to track VoIP calls [9]. In this technique the packets that are watermarked are chosen randomly. The chosen packets are delayed by a fixed amount. This delay is called the *watermarking delay*. Since the attacker has no way of knowing the watermarking delay and which packets are delayed, random perturbations by the attacker are not effective. For the sake of completeness of this report, we provide the basic concept of this watermarking scheme below. The reader can refer the original paper [9] for further details of the scheme:
Given any packet flow $P_1, ..., Pn$ with time stamps $t_1, ..., t_n$ respectively ($t_i < t_j$ for $1 \leq i < j \leq n$), $2r$ distinct packets denoted as $P_{z1}, ..., P_{z2r} (1 \leq zk \leq n - d$ for $1 \leq k \leq 2r)$ are independently and randomly selected. Here r denotes the redundancy. For each of these packets, a packet at a distance d is chosen to create $2r$ packet pairs: $< P_{zk}; P_{zk+di} > (d \geq 1, k = 1, ..., 2r)$. The IPD (Inter-Packet Delay) is calculated for each of these 2r pairs $< P_{zk+d}, P_{zk} >$ as:

$$ipd_{zk} = t_{zk+d} - t_{zk}, (k = 1, .., 2r)$$

Since all the packets are chosen independently, the IPDs are independent and identically distributed(iid). These 2r IPDs are randomly divided into 2 distinct groups of equal size.

Let $ipd_{1,k}$ and $ipd_{2,k}$ $(k = 1, ..., r)$ denote the IPDs in group 1 and group 2 respectively.
Let $Y_k = \frac{(ipd_{1,k} - ipd_{2,k})}{2}$ $(k = 1, .., r)$
Let the average of r Y_k's be represented as: $\overline{Y} = \frac{1}{r}\sum_{k=1}^{r} Y_k$.
Here $\overline{Y}$ represents the average of a group of normalized IPD differences and is symmetrically centered around 0. Increasing or decreasing $\overline{Y}$ by an amount $a > 0$ will shift its distribution to the left or right so that $\overline{Y}$ will be more likely be negative or positive. This property is used to embed watermark bits into the traffic stream. To embed a bit 0, $\overline{Y}$ is decreased by a and for a bit 1, $\overline{Y}$ is increased by a.

2.4 Anonymity

Anonymity can be defined as a state of being indistinguishable from other members of a set. In recent times, anonymity has become synonymous to protecting one's online privacy. Though at the first glance the problems of anonymity and stepping stone detection seem different they are closely related. The approaches to provide anonymity can be used to evade stepping stone detection. While stepping stone detection tries to find correlations between traffic streams, anonymity tries to evade correlation. Our work has drawn inspiration in part by some techniques used in anonymity.

One of the goals of Anonymous communication systems is to achieve sender-receiver unlinkability. Anonymous communication systems try to achieve Anonymity by the use of Mixes. A *Mix* is a network node that relays traffic and tries to hide the correspondence between its incoming and outgoing messages. They try to achieve this by using a variety of techniques like transforming the messages cryptographically, batching or delaying messages and injecting "dummy" messages. Communication is routed through a series of mixes to achieve Sender-Receiver unlinkability. A number of attacks on mix based systems have been published [12, 13, 14, 15, 16]. *Traffic analysis* which is one such attacks, deals with the analysis of meta data associated with data - the sender, the receiver, the

time and the length of messages to extract information about the communication contents and information about the identities of the parties of communication.

Danezis [17] presents an attack based on traffic analysis on connection-based mixed networks functioning in continuous mode. The attack tries to break anonymity of such mixes by tracing streams of messages through the network. It uses signal detection techniques to compare a traffic pattern extracted from the stream that is being tracked with all the links in the network. A degree of similarity with the traced input is assigned to each link. This is used to infer information about the end points and intermediate nodes on the communication path.

Levine et al. [18] study the weaknesses in low-latency mix-based systems like Onion Routing [11]. They show that when two are more mixes are controlled by an attacker, the attacker can study the timings of messages moving through the system to find correlations and might be able to determine if the mixes he controls are on the same communication path. They also propose a novel defense against such timing analysis attacks called *defensive dropping*. With defensive dropping the initiator of a communication constructs extra dummy packets such that an intermediate mix(node) in the communication path is instructed to drop the packet. If these defensive drops are placed randomly and frequently in the packet stream, the correlation between the timing information seen by the mixes controlled by the attacker would be reduced.

CHAPTER 3

SYSTEM DESCRIPTION

To show that the attacker can use the techniques used by low latency anonymous systems to reduce or totally break correlation between traffic flows, we propose an algorithm that the attacker can use on the stepping stone to buffer packets and add dummy traffic. As explained in the earlier sections, the attacker has total control over the stepping stone. So he can modify the application or the TCP/IP stack to implement the buffering algorithm. In the next section we explain the overall architecture of our system.

3.1 Overview

As we describe in Chapter 2, for *stepping stone detection* or *traceback* to be successful, traffic flows need to be correlated. The correlation can be passive or active. Both of these approaches rely on the timing characteristics of traffic flows. As described in earlier sections, an attacker can evade detection by randomly perturbing the timing characteristics of the connection. Active perturbation based approaches like watermarking were proposed to overcome this problem [8]. For the sake of simplicity ,earlier watermarking based approaches have ignored the fact that the attacker controls both the end points of communication and also the intermediate nodes on the path. This gives the attacker much more freedom than assumed by previous work in this area. In this study we show that by selectively buffering and adding dummy traffic, it is possible to break the correlation between traffic flows and thus make watermarking based correlation ineffective. Previous works have assumed that attacker perturbations using huge delays can cause high jitter that can be detected as unusual traffic patterns [19]. We propose a way for

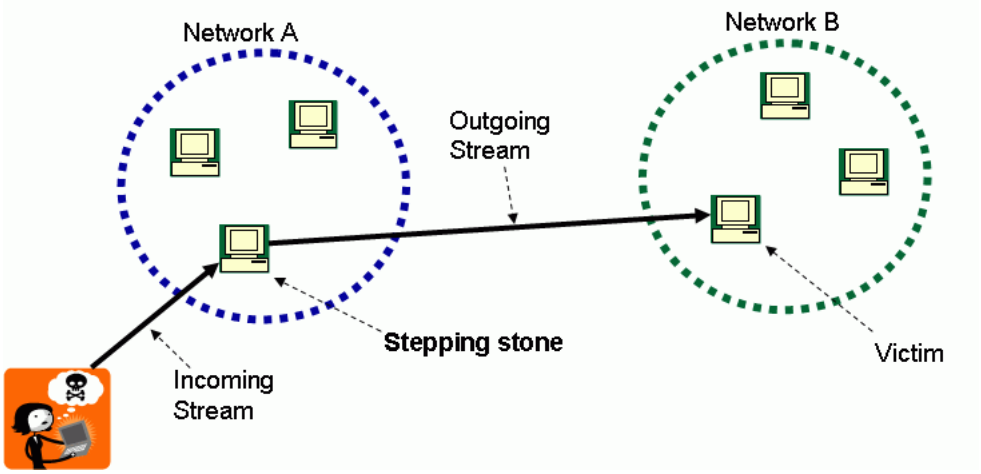

Figure 3.1 Stepping stone

the attacker to perturb traffic without excessive delay, with reduced jitter, and without exhibiting "unusual" traffic patterns.

Consider an attacker who is using a compromised system in a network as a stepping stone to launch an attack on the target system in another network. For simplicity, let us assume that there is only one stepping stone. i.e. the attacker is relaying the attack through only one compromised system as shown in Figure 3.1.

Let us call the person who is trying to detect the stepping stone the *observer*. The *observer's* objective is to detect that an incoming connection is being relayed through the stepping stone to a host outside the network. The observer does this by embedding a timing-based watermark into all the connections coming into the network and tries to detect if any of the outgoing connections contain the watermark that he embedded. The attacker's objective is to evade detection by the observer by distorting the watermark to an extent that it is hard to detect. The attacker can do this by buffering packets and adding dummy packets to generate a constant rate traffic stream. Since all the timing information is lost, the observer will not be able to detect the watermark. We believe that the same techniques can be applied to evading traceback and tracking of VoIP calls.

For the purpose of our study, we make the following assumptions:

- The attacker has complete control on the system that he is using to attack and all the intermediate systems in the connection chain. Thus he can modify the system or use rogue applications on any of these systems
- The observer does not know if there is a stepping stone in his network
- The attacker does not know which packets are watermarked but the observer does
- The attacker does not know what the watermarking delay is

3.2 Connection profiling

The attacker's objective is to remove all timing information from the traffic stream by generating constant rate traffic. To do this he needs to know the delay characteristics of the network. The attacker profiles the network connection between his host and the stepping stone.Before establishing the connection chain to launch his attack, the attacker sends a stream of packets from his system at a specific rate to the stepping stone. On the stepping stone, the attacker records the arrival time of these packets and calculates the inter-packet delays(IPDs) and standard deviation of the IPDs. He does this to collect information about the delay characteristics of the connection. The attacker combines this information with his knowledge of the traffic rate to arrive at the expected arrival time of packets.

Given a packet stream $P_1, ..., P_n$ being sent at a rate r and received on the stepping stone with time stamps $t_1, ..., t_n$ respectively ($t_i < t_j$ for $1 \leq i < j \leq n$), we define the *inter-packet delay* (IPD) between P_{k+1} and P_k as:

$$ipd_k = t_{k+1} - t_k, (k = 1, ..., n-1)$$

Since the attacker already knows the traffic rate r at the source, he knows the mean inter-packet delay $\overline{ipd} = 1/r$. He calculates the IPD standard deviation as:

$$\sigma = \sqrt{\frac{1}{n-1} \sum_{i=1}^{(n-1)} (ipd_i - \overline{ipd})^2}$$

Since the attacker sends packets at a constant rate, he can expect that the IPDs are normally distributed around the mean when the packets arrive at the stepping stone. According to the empirical rule for normal distribution, the attacker can deduce that :

- 68.27% of the packets will arrive within 1 standard deviation of the mean
- 95.45% of the packets will arrive within 2 standard deviations of the mean
- 99.73% of the packets will arrive within 3 standard deviations of the mean

3.3 Evading detection

In this section we describe our technique using which the attacker can evade detection by the observer. Once the attacker has profiled the connection he establishes the connection chain through the stepping stone(s) to the target host and starts the attack. When the attack packets arrive at the gateway, the observer embeds a watermark as explained in Section 2.3.3. Since the attacker does not know which packets have been watermarked, he has to act without this knowledge. The attacker knows at what rate he is sending traffic and he can calculate the standard deviation on inter-packet delay as explained in Section 3.2. To evade detection by generating a constant rate traffic stream, the attacker needs to buffer(delay) packets that arrive early and drop packets when they arrive late. To achieve this, the attacker divides the time-line starting at the arrival of the first packet into time slots. The length of each slot is 1 $\overline{ipd} = 1/r$ which is the mean inter-packet delay (IPD) at the source of the traffic. Each packet is expected to arrive in its respective slot. But the packets may arrive earlier or later than expected (due to the internet and watermarking delays encountered by packets) as illustrated in Figure 3.2. So the attacker needs to have a tolerance margin for each slot to decide if the packet arrived in its respective slot or not. The tolerance is a configurable parameter that the

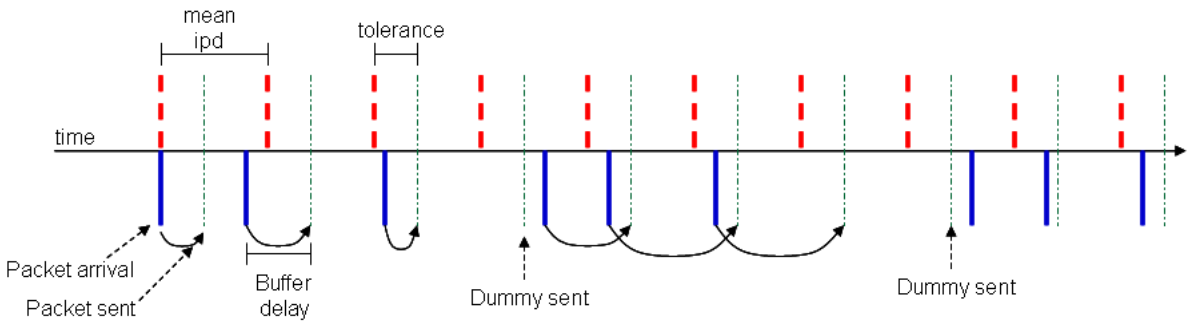

Figure 3.2 Traffic time-line

attacker can adjust based on the standard deviation of the IPDs. Since the IPDs are normally distributed over the mean, using the empirical rule for normal distribution, the attacker knows that around 95% of the packets arrive within two standard deviations of the mean. So the attacker can set the tolerance level to, for example 2σ. This would also be the maximum buffering delay B. If a packet arrives early, the packet is delayed till the end of the slot. If a packet arrives late, i.e. after the end of its slot, a dummy packet is sent in place of the actual packet. When the delayed packet finally arrives, it is buffered(queued) till the end of the next available slot and then sent. Some packets may arrive very late (For ex. after two slots) immediately followed by packets that arrive in time. This could lead to multiple packets being queued in the buffer and have a cascading effect on the buffering delays thus affecting the quality of the connection. The attacker can eliminate the problem of cascading delays by dropping packets that arrive very late. Since the attacker knows that almost all the packets would arrive within 4σ of the mean, at any given point of time, there would be a maximum of 2 packets in the buffer(queue). If there are are more than 2 packets in the buffer, the attacker drops the first packet (which presumably arrived very late) and sends the next packet in the buffer. The entire process can be summarized by the following algorithm:

1. When the first packet arrives, delay the packet by the maximum buffering delay

2. From the time the first packet was sent, trigger an event at intervals of duration $\overline{ipd}$
3. From the second packet onwards, buffer the packets that arrive before an event is triggered
4. When an event is triggered:
    - if there is no packet in the buffer, send a dummy packet
    - if there are more than 2 packets in the buffer, drop the first packet and send the second packet
    - if there are less than or equal to two packets in the buffer, send the first packet in the buffer

3.4 Watermark detection and observer counterattack

Since the buffering algorithm generates a constant rate traffic stream, all timing information of the traffic flow is lost. In effect this totally removes the watermark that was previously embedded. We performed some experiments to show the effect of buffering on watermark detection. The results are presented in Section 4.5. The observer in an effort to make it harder for the attacker to evade detection, can increase the watermarking delays. Increasing the watermarking delay makes the detection scheme more robust. This would also increase the jitter in the stepping stone connection. To work against this, the attacker has to buffer packets for a longer duration and would have to drop more packets, thus degrading the quality of his own connection. It should be noted that increasing the watermarking delay does affect connection quality. Since the observer needs to watermark all incoming connections, an increase in watermarking delay would not only affect the connection being used by the attacker, but all other incoming connection in the network. we performed simulations to determine the the resilience of the buffering technique to increased watermarking delays and the results are presented in Section 4.5.

CHAPTER 4

SIMULATIONS

Earlier, we have made an argument that the attacker can use cover traffic to generate constant rate traffic stream to evade correlation. We wanted to show that our buffering technique can be effective even in this scenario. To demonstrate this, we need a watermarking technique that is capable of correlating constant rate traffic streams. The technique that comes closest to satisfying our requirements was proposed by Wang, Chen, and Jajodia [9] to track VoIP calls. We chose this technique for our simulations. Although, in this work, we only apply our buffering technique to constant rate traffic streams, buffering can also effective for streams of other delay characteristics like interactive traffic generated by a user typing commands at a console. We intend to perform experiments using other watermarking techniques for interactive stepping stones in our future work.

We conducted a number of experiments to test the effectiveness of out technique to evade of watermark detection. We also performed experiments to determine the affect of different watermarking delays on the drop rates and amount of chaff. The system was simulated in Java. Figure 4.1 shows the architecture of our experimental setup. We describe each individual component of the system in the following sections.

4.1 Traffic generator

The traffic generator generates Internet traffic as well as attack traffic based on a delay distribution. The delays simulate Internet packet delays. For our experiments we used a normal distribution with a standard deviation of one fourth of the mean for

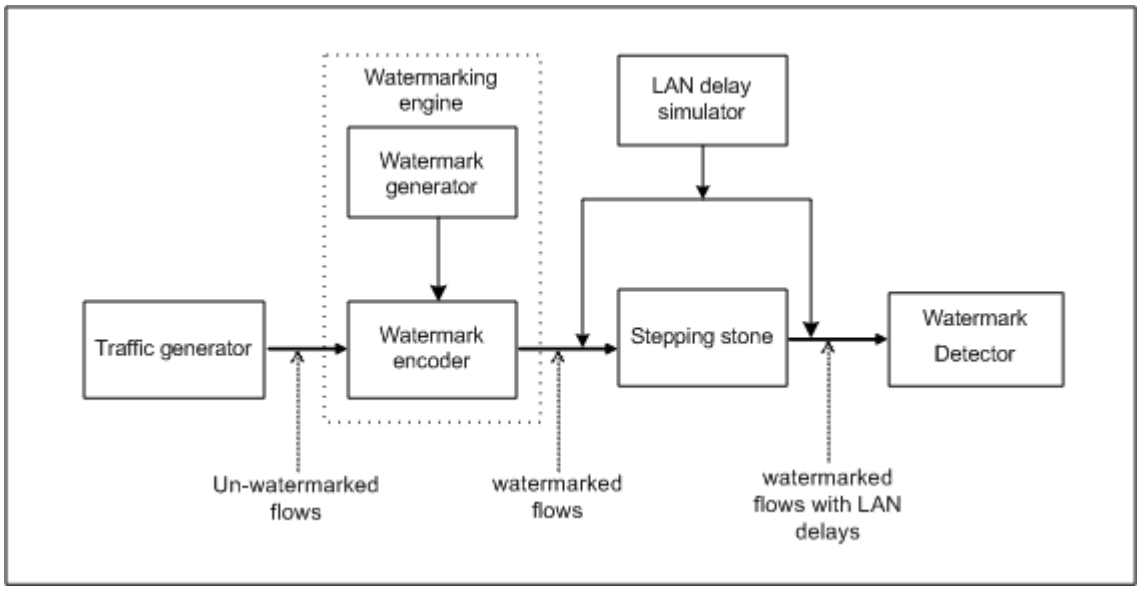

Figure 4.1 Experimental setup

Internet delays. The characteristics of the generated traffic, such as rate, average end-to-end delay and packet loss can be controlled by configurable parameters. We chose parameters that reflect realistic VoIP traffic on the internet. For each experiment we generated 100 traffic flows at rate of 30 pkts/sec for a duration of 900 secs (15 mins).The average end-to-end delay was set to 100ms and the drop rate was set to an average of 1%. This traffic is in the form of traffic logs.

4.2 Watermarking engine

The watermarking engine uses the watermarking technique proposed by Wang, Chen, and Jajodia [9] to alter the inter arrival timing of packets that are generated by the traffic generator.

4.2.1 Watermark generator

The watermark generator generates unique watermarks with a specified hamming distance. The distance is important to ensure low false positives. We chose a hamming distance of 9 based on the best results of [9]. These watermarks are embedded into traffic flows by the watermarking engine.

4.2.2 Parameter selection

The parameters used for the watermarking algorithm were chosen based on the results obtained by Wang, Chen and Jajodia [9]. The parameters corresponding to their best results were chosen. All the watermarks are 24 bits in size. The watermarks are generated such that the minimum hamming distance between any two watermarks is 9. Wang, Chen and Jajodia did some experiments to determine the optimal redundancy factor to be used. They found that a redundancy factor of 25 yields a very low average bit error rate. So for our project we chose to have a redundancy factor of 25. Having a large redundancy makes the watermark robust against network jitters and results in a low bit error rate. The delay introduced by the watermark has to be small in order to make it difficult for the attacker to determine if his flow is watermarked. Wang, Chen and Jajodia found that a watermarking delay of 3ms was sufficient to confuse the attacker and achieve high true positive rate.

4.3 LAN delay simulator

The LAN delay simulator is treated as a black box in to which packets enter and suffer a delay based on a delay model, shown in Figure 4.2, and go out of the network. This basically simulates the network and processing delays that the packets encounter before reaching the stepping stone and after leaving it.

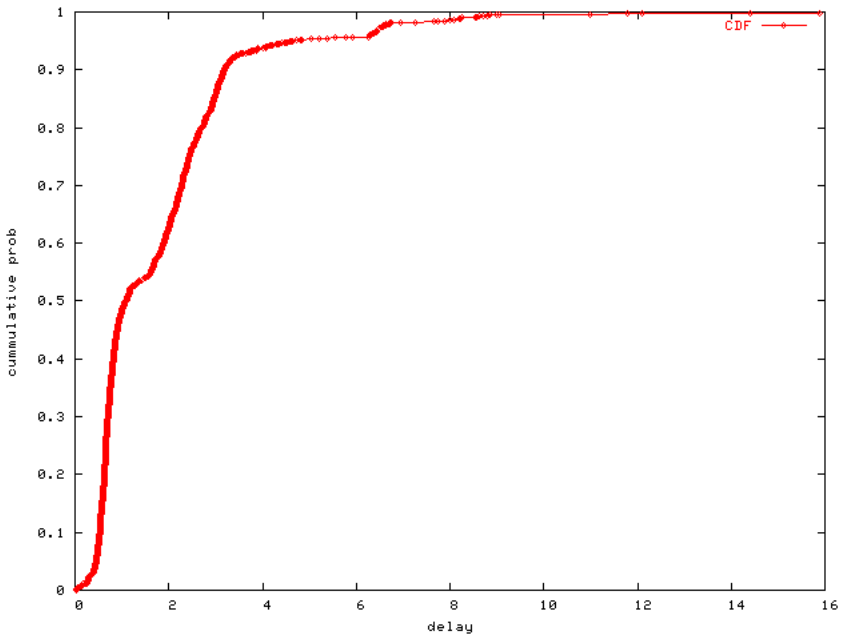

Figure 4.2 LAN delay model

4.3.1 LAN delay model

To obtain a model for delays in a *Local Area Network*(LAN), we performed some experiments on the local area network of the University Of Texas at Arlington. Ping packets were sent to randomly chosen systems in the network and the round trip times(RTTs) were recorded. 10 ping packets were sent to each of the 60 randomly chosen hosts. This experiment was repeated on three different systems on the network. A cumulative distribution function(CDF) was plotted with the RTTs collected as shown in Figure 4.2 and curve fitting was used to obtain a function for the delay model.

4.4 Watermark decoder

The Watermark decoder acts as the egress monitor which checks outgoing traffic flows for watermarks. It is assumed that there is coordination between the watermarking engine and the decoder and the decoder knows which packets are watermarked. The

watermark that is found is compared with all the embedded watermarks and analyzed to determine false positives.

4.5 Results

In this section we present the results of the experiments we performed to test the effectiveness of our proposed technique to evade watermark detection. We also conducted experiments to determine the effect of watermarking delays on the buffering delay, drop rate and the amount of chaff.

4.5.1 Watermark detection

For a detection scheme to be effective, it should not only have a high detection rate but also a low *false positive* rate. A *false positive* can occur when the watermark decoder erroneously finds a watermark in an un-watermarked flow or in a flow that has a different watermark. We performed two sets of experiments to test the effectiveness of our proposed technique to evade detection. For both sets ,we used 100 simulated traffic flows generated as explained in Section 4.1. Our first set of experiments was to show the effectiveness of the watermarking technique when there is no buffering done by the attacker. This is an ideal case when the attacker is not perturbing any characteristics of the traffic flows. To demonstrate the effectiveness of our proposed buffering technique to evade detection, we conducted a second set of experiments where we assume that the attacker is perturbing traffic flows that are being relayed through the stepping stone by using the buffering technique. Figure 4.3 shows our results is the form of *Receiver operating characteristic*(ROC) curves. An ROC curve is a graphical plot of the sensitivity (fraction of true positives) vs. specificity (the fraction of false positives). As can be seen, when buffering is not used, the detection rate is high with a low false positive rate. But when buffering is used, the ROC curve becomes almost linear with a 45 degree gradient

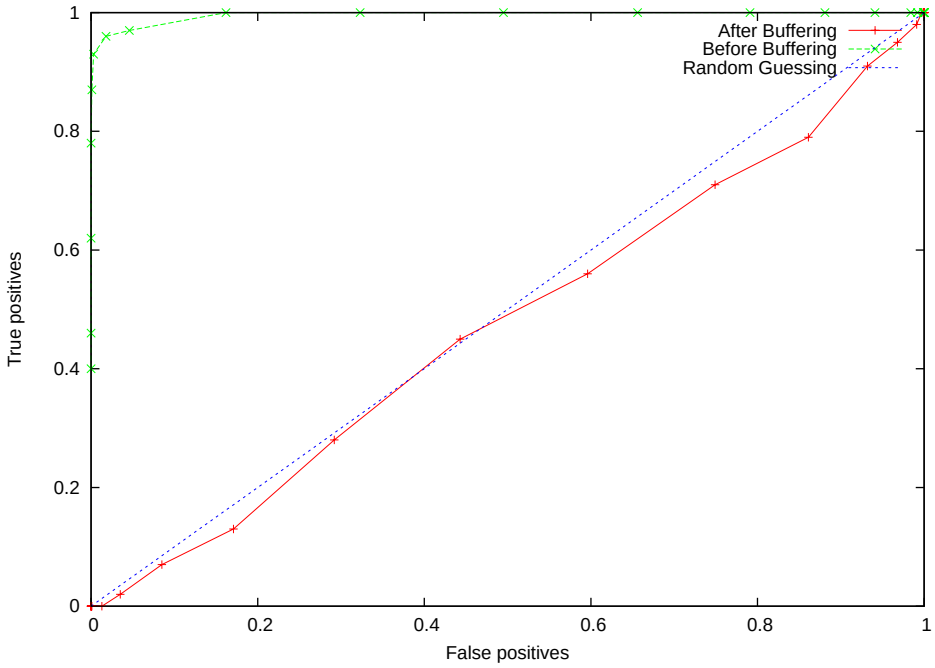

Figure 4.3 Effectiveness of buffering

indicating that the detection rate has reduced drastically and false positive rate has increased. This also means that the detection rate can be increased only at the cost of drastically increasing false positives.

4.5.2 Buffering delay

When the attacker uses buffering to evade detection, the observer may try to counter it by increasing the watermarking delay. If the attacker has to maintain a constant rate traffic stream, he would have to delay packets for a longer duration. We performed a experiments to determine the effect of increased watermarking delays on the buffering delay. Figure 4.4 shows that even when the watermarking delay is varied from 5ms to 50ms, the Average buffering delay only varies by less than 10ms. This shows that with a slight increase in buffering delay an attacker would still be successful in evading detection. The choice of the tolerance margin used by the buffering algorithm could have

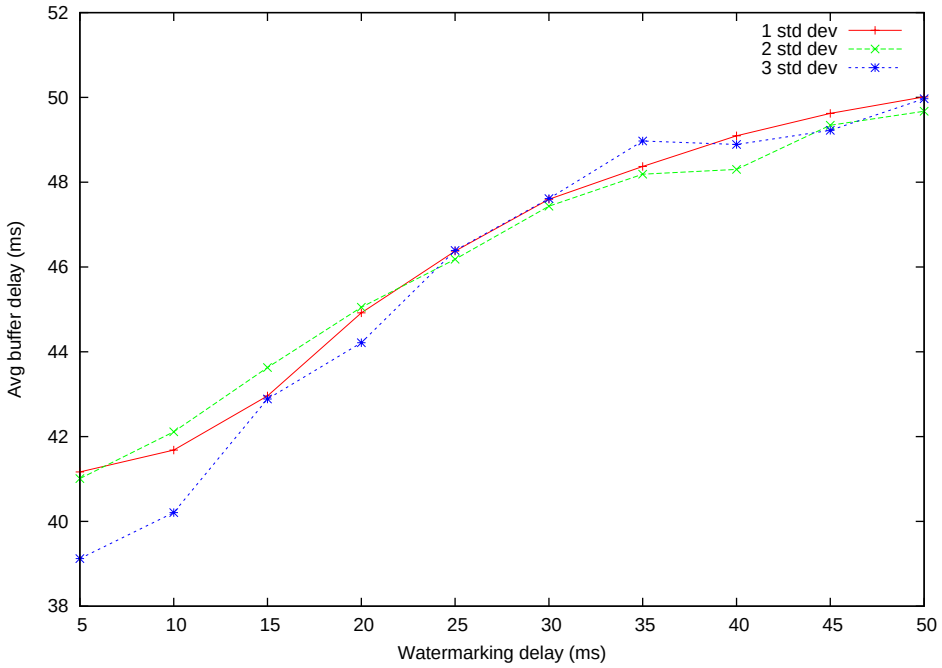

Figure 4.4 Average buffer delay w.r.t watermarking delay

an effect on the buffering delay. To determine this effect, we performed experiments by varying the tolerance margin from 1 standard deviation to 3 standard deviations. As seen in Figure 4.4, increasing the tolerance margin does not affect the buffering delay much. This is counter-intuitive because in principle, as the tolerance margin increases, one would expect the average buffering delay to increase. The average buffer delay remains pretty much the same because as the tolerance margin increases, packets that arrive late are sent in their respective slots rather than being buffered. So the packets that arrive after a late packet do not have to wait for the late packet to be sent. In effect, this reduces the average buffer delay.

4.5.3 Drop rate

Our next set of experiments were to determine the drop rates that we incur for different watermarking delays. We varied the watermarking delays from 5ms to 50ms.

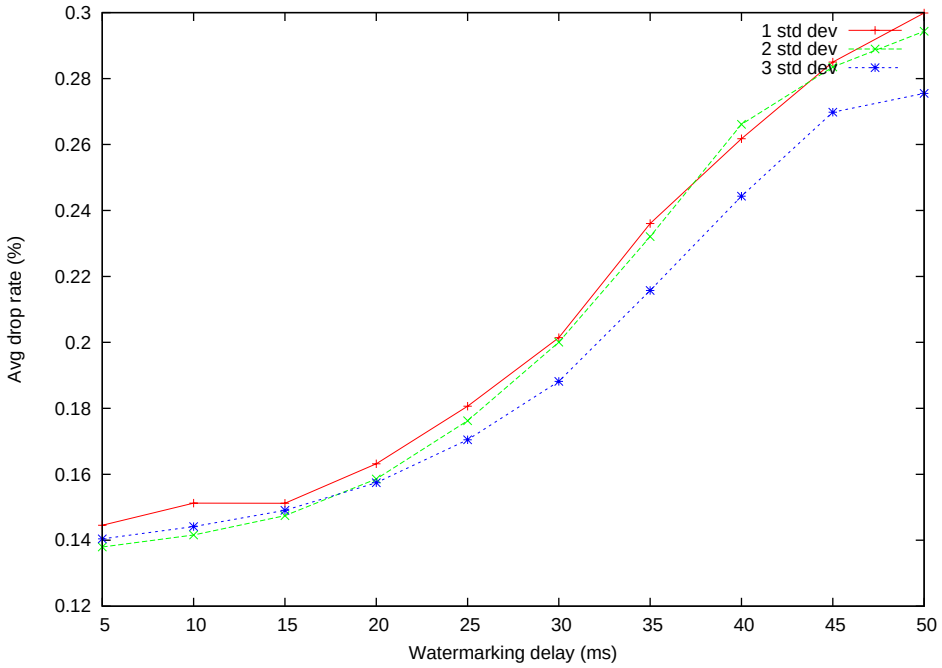

Figure 4.5 Drop rate

Figure 4.5 shows that even for a watermarking delay of 50ms, the average drop rate is very small (0.3%). This supports our argument that the watermark detection can be severely degraded with our buffering technique with a very low drop rate. The choice of the tolerance margin used by the buffering algorithm could have an effect on the drop rate. To determine this effect we performed experiments by varying the tolerance margin from 1 standard deviation to 3 standard deviations. As seen in Figure 4.5, increasing the tolerance margin reduces the drop rate. The reason for this is that as the tolerance margin increases, packets that arrive late are sent in their respective slots rather than being buffered and eventually dropped.

4.5.4 Amount of chaff

Though the amount of chaff used by the attacker would considerably affect the quality of the connection, it would still be desirable to determine the amount of chaff

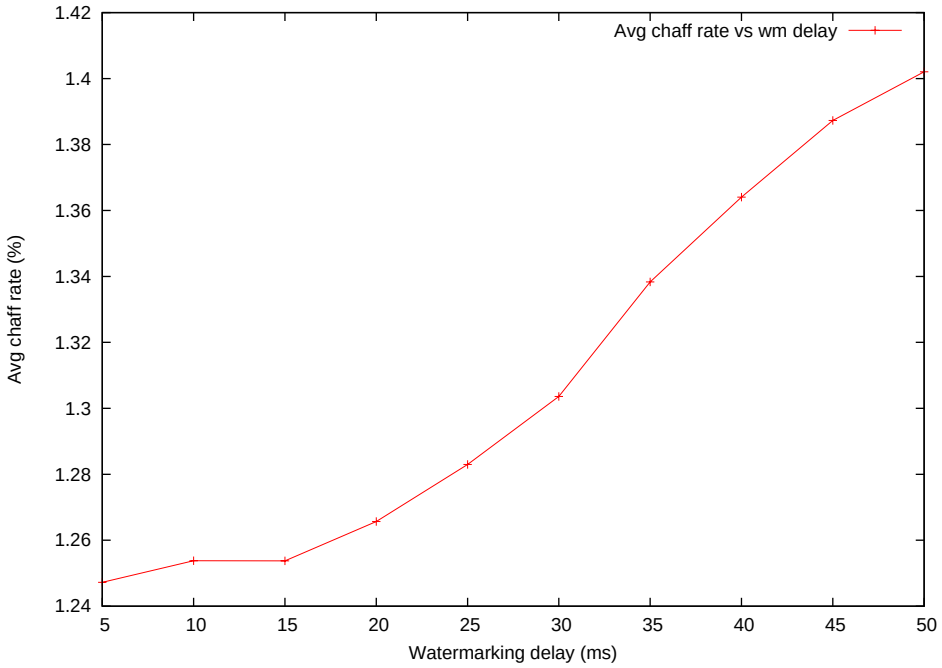

Figure 4.6 Chaff rate

needed by the attacker to evade detection. We performed experiments to determine the amount of chaff needed to counter different watermarking delays. Figure 4.6 shows that the overall chaff rate is very low. Even with a watermarking delay of 50ms, the chaff rate is only 1.4%. We also see that as the watermarking delay is varied from 5ms to 5ms, the chaff rate only varies by about 0.15%. This further corroborates our argument that with buffering and a very low amount of chaff the attacker can degrade watermark detection severely.

CHAPTER 5

CONCLUSION

In this thesis we make arguments against some of the assumptions about the capabilities of an attacker made by earlier approaches to stepping stone detection. We loosen some of these assumptions and assume that an attacker is capable of adding delays and cover traffic to his traffic streams on the stepping stone. We propose a simple buffering technique that when used by an attacker on a stepping stone, is effective is in severely degrading detection. Our technique involves buffering of packets and adding chaff to generate constant rate traffic streams. We perform simulations using a watermark based detection scheme [9] designed to detect correlations between constant rate traffic streams and show that our technique is effective is evading detection even by this correlation scheme.

5.1 Future Work

The criteria used to drop packets in our buffering algorithm can be improved further by dropping packets more intelligently by using packet sequence numbers and keeping track of the number of chaff packets sent. If the stepping stone knows that it has already sent a chaff packet in place of a late packet, it can drop the late packet when it arrives and thus reduce the buffering time of consequent packet. We believe that this modification can reduce the average buffering delay by half. We also want to do an actual implementation of our technique by modifying an SSH server and client to further demonstrate our technique. The application of our buffering technique to the area of anonymity needs to be explored.

REFERENCES

[1] S. Staniford-Chen and L. Heberlein, "Holding Intruders Accountable on the Internet," *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, pp. 39–49, 1995.

[2] H. Jung, H. Kim, Y. Seo, G. Choe, S. Min, C. Kim, and K. Koh, "Caller Identification System in the Internet Environment," *Proceedings of 4th USENIX Security Symposium*, vol. 246, 1993.

[3] S. Snapp, J. Brentano, G. Dias, T. Goan, L. Heberlein, C. Ho, K. Levitt, B. Mukherjee, S. Smaha, T. Grance, *et al.*, "DIDS (Distributed Intrusion Detection System)-Motivation, Architecture, and an Early Prototype," *Proceedings of the 14th National Computer Security Conference*, pp. 167–176, 1991.

[4] Y. Zhang and V. Paxson, "Detecting stepping stones," *Proceedings of the 9th USENIX Security Symposium*, pp. 171–184, 2000. [Online]. Available: citeseer.ist.psu.edu/article/zhang00detecting.html

[5] K. Yoda and H. Etoh, "Finding a Connection Chain for Tracing Intruders," *F. Guppens, Y. Deswarte, D. Gollmann and M. Waidner, editors, 6 thEuropean Symposium on Research in Computer Security–ESORICS 2000 LNCS-1895*, 2000.

[6] X. Wang, D. Reeves, and S. Wu, "Inter-packet delay-based correlation for tracing encrypted connections through stepping stones," 2002. [Online]. Available: citeseer.ist.psu.edu/wang02interpacket.html

[7] D. Donoho, A. Flesia, U. Shankar, V. Paxson, J. Coit, and S. Staniford, "Multiscale stepping-stone detection: detecting pairs of jittered interactive

streams by exploiting maximum tolerable delay," 2002. [Online]. Available: citeseer.ist.psu.edu/donoho02multiscale.html

[8] X. Wang and D. Reeves, "Robust correlation of encrypted attack traffic through stepping stones by manipulation of interpacket delays," *Proceedings of the 10th ACM conference on Computer and communication security*, pp. 20–29, 2003.

[9] X. Wang, S. Chen, and S. Jajodia, "Tracking anonymous peer-to-peer VoIP calls on the internet," *Proceedings of the 12th ACM conference on Computer and communications security*, pp. 81–91, 2005.

[10] "Skype - the global internet telephone company." [Online]. Available: http://www.skype.org

[11] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[12] A. Back, I. Goldberg, and A. Shostack, "Freedom systems 2.1 security issues and analysis," Zero Knowledge Systems, Inc.," White Paper, May 2001.

[13] M. Wright, M. Adler, B. N. Levine, and C. Shields, "An analysis of the degradation of anonymous protocols," in *Proceedings of the Network and Distributed Security Symposium - NDSS '02*. IEEE, February 2002.

[14] P. Syverson, G. Tsudik, M. Reed, and C. Landwehr, "Towards an Analysis of Onion Routing Security," in *Proceedings of Designing Privacy Enhancing Technologies: Workshop on Design Issues in Anonymity and Unobservability*, H. Federrath, Ed. Springer-Verlag, LNCS 2009, July 2000, pp. 96–114.

[15] A. Serjantov, R. Dingledine, and P. Syverson, "From a trickle to a flood: Active attacks on several mix types," in *Proceedings of Information Hiding Workshop (IH 2002)*, F. Petitcolas, Ed. Springer-Verlag, LNCS 2578, October 2002.

[16] O. Berthold, A. Pfitzmann, and R. Standtke, "The disadvantages of free MIX routes and how to overcome them," in *Proceedings of Designing Privacy Enhancing Tech-*

nologies: Workshop on Design Issues in Anonymity and Unobservability, H. Federrath, Ed. Springer-Verlag, LNCS 2009, July 2000, pp. 30–45.

[17] G. Danezis, "The traffic analysis of continuous-time mixes," in *Proceedings of Privacy Enhancing Technologies workshop (PET 2004)*, ser. LNCS, vol. 3424, May 2004.

[18] B. N. Levine, M. K. Reiter, C. Wang, and M. K. Wright, "Timing attacks in low-latency mix-based systems," in *Proceedings of Financial Cryptography (FC '04)*, A. Juels, Ed. Springer-Verlag, LNCS 3110, February 2004.

[19] A. Blum, D. Song, and S. Venkataraman, "Detection of interactive stepping stones: Algorithms and confidence bounds," in *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection(RAID '04)*. Springer-Verlag, LNCS 3224, Jan 2004, pp. 258 – 277. [Online]. Available: citeseer.ist.psu.edu/698115.html

[20] N. Forrester, "Measuring performance and quality of packet voice systems," Motorola Computer Group, Tech. Rep., January 2003.

BIOGRAPHICAL STATEMENT

Madhu Venkateshaiah was born in Thimmanahalli, India, in 1978. He received his B.E. degree from Mysore University, India, in Computer Science and Engineering in 2000, his M.S. degree with a Thesis from The University of Texas at Arlington in 2006 in Computer Science and Engineering. He has been part of the Information Security(iSEC) Lab at UTA since its inception. His research interests include Network Security, Anonymity and Privacy on the Internet.