ABSTRACT

Wang, Xinyuan. Tracing Intruders behind Stepping Stones. (Under the direction of Dr. Douglas S. Reeves.)

Network based intruders seldom attack directly from their own hosts but rather stage their attacks through intermediate "stepping stones" to conceal their identity and origin. To track down and apprehend those perpetrators behind stepping stones, it is critically important to be able to correlate connections through stepping stones.

Tracing intruders behind stepping stones and correlating intrusion connections through stepping stones are challenging due to various readily available evasive countermeasures by intruders:

- Installing and using backdoor relays (i.e. netcat) at intermediate stepping stones to evade logging of normal logins.
- Using different types of connections (i.e. TCP, UDP) at different portions of the connection chain through stepping stones to complicate connection matching.
- Using encrypted connections (with different keys) across stepping stones to defeat any content based comparison.
- Introducing timing perturbation at intermediate stepping stones to counteract timing based correlation of encrypted connections.

In this dissertation, we address these challenges in detail and design solutions to them.

For unencrypted intrusion connections through stepping stones, we design and implement a novel intrusion tracing framework called Sleepy Watermark Tracing (SWT), which applies principles of steganography and active networking. SWT is "sleepy" in that it does not introduce overhead when no intrusion is detected. Yet it is "active" in that when an intrusion is detected, the host under attack will inject a watermark into the backward connection of the intrusion, and wake up and collaborate with intermediate routers along the intrusion path. Our prototype shows that SWT can trace back to the trustworthy security gateway closest to the origin of the intrusion, with only a single packet from the intruder. With its unique active tracing, SWT can even trace when intrusion connections are idle.

Encryption of connections through stepping stones defeats any content based correlation and makes correlation of intrusion connections more difficult. Based on inter-packet timing characteristics, we develop a novel correlation scheme of both encrypted and unencrypted connections. We show that

(after some filtering) inter-packet delays (IPDs) of both encrypted and unencrypted, interactive connections are preserved across many router hops and stepping stones. The effectiveness of IPD based correlation requires that timing characteristics be distinctive enough to identify connections. We have found that normal interactive connections such as telnet, SSH and rlogin are almost always distinctive enough to provide correct correlation across stepping stones

The timing perturbation at intermediate stepping stones of packet flows poses additional challenge in correlating encrypted connections through stepping stones. The timing perturbation could either make unrelated flows have similar timing characteristics or make related flows exhibit different timing characteristics, which would either increase the false positive rate or decrease the true positive rate of timing-based correlation. To address this new challenge, we develop a novel watermark based correlation scheme that is designed to be specifically robust against such kinds of timing perturbation. The idea is to actively embed a unique watermark into the flow by slightly adjusting the timing of selected packets of the flow. If the embedded watermark is unique enough and robust enough against the timing perturbation by the adversary, the watermarked flow could be uniquely identified and thus effectively correlated. By utilizing redundancy techniques, we develop a robust watermark correlation framework that reveals a rather surprising result on the inherent limits of independent and identically distributed (*iid*) random timing perturbations over sufficiently long flows. We also identify the tradeoffs between the defining characteristics of the timing perturbation and the achievable correlation effectiveness. Our experiments show that our watermark based correlation performs significantly better than existing passive timing based correlation in the face of random timing perturbation.

In this research, we learn some general lessons about tracing and correlating intrusion connections through stepping stones. Specifically, we demonstrate the significant advantages of active correlation approach over passive correlation approaches in the presence of active countermeasures. We also demonstrate that information hiding and redundancy techniques can be used to build highly effective intrusion tracing and correlation frameworks.

Tracing Intruders behind Stepping Stones

by

Xinyuan Wang

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Computer Science

Raleigh

2004

APPROVED BY:

Dr. Gregory T. Byrd	Dr. George N. Rouskas
Dr. Peng Ning	Dr. Douglas S. Reeves Chair of Advisory Committee

To those who love me

My parents Qingfeng Wang, Maolan Jiang

My wife Ning Li

My daughter Serena Wang

BIOGRAPHY

Xinyuan Wang was born in Beijing. P.R. China. He received B.S. in Computer Science from Peking University in 1987, M.S. in Computer Science from Chinese Academy of Space Technology in 1990 respectively.

ACKNOWLEDGEMENTS

First I want to thank my advisor, Dr. Douglas S. Reeves. Without his advice and guidance in the past four years, this dissertation would never come into existence. Getting a PhD is never an easy task, especially when I have a demanding full-time job at the same time. There was a moment that I was lost and I wanted to quit. Thanks to God, I was able to find Dr. Reeves and ask him to be my advisor. This has turned out to be the turning point of my PhD research.

I have benefited greatly from Dr. Reeves' insight and the hard questions he asked. It is those hard questions that guide me to look deeper into the problem and the solution I investigated, and it is those hard questions that lead me to conduct award-winning research on a seemly unsolvable problem.

I would like to thank Dr. George N. Rouskas, Dr. Gregory T. Byrd and Dr. Peng Ning for serving my advisory committee. I appreciate Dr. Byrd's encouragement and his word-by-word proofreading on my dissertation. I always enjoy the discussion with Dr. Ning about the tracing work, which has inspired a new solution to the tracing problem.

I would like to thank Dr. Felix Wu of The University of California at Davis for leading me into the exciting field of network security research while he was faculty of N.C. State University.

I would also like to thank my parents for their love and support and for giving me the best possible education they could.

Finally, my utmost thanks go to my wife for her love, support and encouragement over the years.

Contents

LIST OF FIGURES IX

LIST OF TABLES XI

CHAPTER 1 INTRODUCTION 1

1.1 MOTIVATION 1
1.2 CHALLENGES IN TRACING INTRUDERS IN THE INTERNET — THE BIG PICTURE .. 2
1.3 CHALLENGES IN TRACING THROUGH STEPPING STONES 6
1.3.1 Host Login Information Disguise, Deletion and Forgery 7
1.3.2 Connection Content Transformation 8
1.3.3 Timing Perturbation 8
1.3.4 Introducing Loops to the Intrusion Connection Chain 9
1.3.5 Flow Level Traffic Padding 9
1.3.6 Packets Drop and Retransmission 9
1.3.7 Flow Repacketization 10
1.3.8 Packet Reorder 10
1.3.9 Flow Split and Merge 10
1.3.10 Mixing Multiple Flows 11
1.4 CONTRIBUTIONS 11
1.5 ORGANIZATION 13

CHAPTER 2 STEPPING STONE TRACING PROBLEM AND RELATED WORKS 15

2.1 THE OVERALL STEPPING STONE TRACING PROBLEM MODEL 15
2.1.1 Correlation Problem Solution Model 17
2.2 EVALUATION CRITERIA AND CLASSIFICATION OF TRACING AND CORRELATION APPROACHES 18
2.2.1 Evaluation Criteria of Tracing and Correlation Approaches 19
2.2.2 Classification of Tracing and Correlation Approaches 21
2.3 TRACING AND CORRELATING UNENCRYPTED CONNECTIONS 23
2.3.1 Distributed Intrusion Detection System (DIDS) 23
2.3.2 Caller Identification System (CIS) 23
2.3.3 Thumbprinting 24
2.3.4 Intrusion Detection and Isolation Protocol (IDIP) 24
2.3.5 Session Token Protocol (STOP) 25
2.4 TRACING AND CORRELATING ENCRYPTED CONNECTIONS 26
2.4.1 ON/OFF Based 26
2.4.2 Deviation Based 26
2.5 TRACING AND CORRELATING ENCRYPTED CONNECTIONS WITH TIMING PERTURBATION 27
2.5.1 Wavelet Based 27
2.6 SUMMARY 28

CHAPTER 3 SLEEPY WATERMARK TRACING 30
3.1 INTRODUCTION 30
3.2 MINIMIZING CORRELATION FALSE POSITIVE RATE 32
3.2.1 Making the Network Content More Unique by Watermarking 33
3.2.2 Making the Embedded Watermark Invisible by Steganography 35
3.3 SWT OVERVIEW 36
3.3.1 SWT Objective 36
3.3.2 Basic SWT Concepts 37
3.3.3 Basic SWT Assumptions 38
3.4 SWT ARCHITECTURE AND PROTOCOL 39
3.4.1 Sleepy Intrusion Response 41
3.4.2 Watermark Correlation 43
3.4.3 Active Tracing 44
3.5 SWT ANALYSIS 48
3.5.1 Robustness and Security 48
3.5.2 Intrusiveness and Privacy 49
3.6 EXPERIMENTS 50
3.7 SUMMARY 53

CHAPTER 4 IPD-BASED CORRELATION OF ENCRYPTED CONNECTIONS..... 55
4.1 INTRODUCTION 55
4.2 IPD BASED CORRELATION MODEL 56
4.2.1 Basic IPD Correlation Concepts and Definitions 57
4.2.2 Heuristics in Finding Correlation Points 59
4.2.3 Correlation Point Function Assessment Criteria 60
4.3 CORRELATION POINT FUNCTION 61
4.3.1 Mini/Max Sum Ratio (MMS) 61
4.3.2 Statistical Correlation (STAT) 62
4.3.3 Normalized Dot Product 1 (NDP1) 62
4.3.4 Normalized Dot Product 2 (NDP2) 63
4.4 CORRELATION VALUE FUNCTION 64
4.5 EXPERIMENT 65
4.5.1 Correlation Point Experiment 65
4.5.2 Aggregated Flow Correlation Experiment 69
4.5.3 Correlation Performance 74
4.6 SUMMARY 74

CHAPTER 5 ROBUST CORRELATION OF ENCRYPTED CONNECTIONS THROUGH WATERMARKING 77
5.1 INTRODUCTION 77
5.2 OVERVIEW OF WATERMARK-BASED CORRELATION 79
5.2.1 Watermarking Model and Concept 80
5.3 EMBEDDING ONE WATERMARK BIT INTO ONE IPD 81
5.3.1 Basic Watermark Bit Embedding and Decoding 81
5.3.2 Maximum Tolerable Perturbation, Tolerable Perturbation Range and Vulnerable Perturbation Range 83
5.4 PROBABILISTICALLY ROBUST WATERMARKING OVER IPD 85

5.4.1 Embedding Watermark Bit over the Average of Multiple IPDs *85*
5.4.2 Embedding Multiple Watermark Bits.. *87*
5.4.3 Attacker's Impact Over the Average of Multiple IPDs.................................. *88*
5.5 ANALYSIS ON THE DISTRIBUTION OF WATERMARK BIT ROBUSTNESS 90
5.6 WATERMARK DETECTION, AND ANALYSIS OF DETECTION AND COLLISION RATES 94
5.6.1 Limitations *97*
5.7 EXPERIMENT 97
5.7.1 Correlation True Positive Experiment.. *98*
5.7.2 Correlation False Positive Experiment *101*
5.7.3 Tradeoffs in Watermark Detection Rate *102*
5.7.4 Impact of Non-Independent Timing Perturbation over Watermark-Based Correlation.. *105*
5.7.5 Embedding and Detecting Multiple Watermarks at the Same Time............ *106*
5.8 SUMMARY.. 108

CHAPTER 6 THE SERIALIZATION IN TRACING INTRUSION CONNECTIONS THROUGH STEPPING STONES.. 110
6.1 INTRODUCTION 110
6.1.1 Ordinals of Basic Set Theory.. *111*
6.1.2 Tracing Problem Formulation... *112*
6.2 THE LOOP FALLACY IN DETERMINISTIC SERIALIZATION OF CORRELATED CONNECTIONS.. 113
6.3 DETERMINISTIC SERIALIZATION OF CORRELATED CONNECTIONS................. 115
6.3.1 Point Connectivity and Serialization Based on Point Adjacency................ *116*
6.3.2 Edge Connectivity and Serialization Based on Edge Adjacency................. *118*
6.3.3 Serialization Based on Adjacent Connection Pairs..................................... *121*
6.3.4 Finding Adjacent Correlated Connection Pairs.. *125*
6.4 SUMMARY.. 127

CHAPTER 7 CONCLUSIONS AND FUTURE WORK.. 128
7.1 SUMMARY.. 128
7.1.1 Tracing and Correlation of Unencrypted Connections across Stepping Stones *129*
7.1.2 Correlation of Encrypted Connections across Stepping Stones *130*
7.1.3 Robust Correlation of Encrypted Connections against Active Timing Perturbation.. *130*
7.1.4 Deterministic Serialization in Tracing Intrusion Connections through Stepping Stones... *131*
7.1.5 General Principles and Lessons *132*
7.2 FUTURE DIRECTIONS.. 134
7.2.1 Robust Correlation against Sophisticated Countermeasures....................... *134*
7.2.2 Tracing and Correlation of Non-Interactive Traffic.................................... *135*
7.2.3 The Balance between Privacy and Traceability *135*
7.2.4 Automatic Intrusion Response *136*
7.2.5 Tracing of Other Forms of Network-Based Attacks *136*
7.2.6 Survivable System and Network.. *137*

BIBLIOGRAPHY 138

List of Figures

Figure 1. 1 Packet Transformation when Forwarded through Intermediate Host 3
Figure 1. 2 The Overall Tracing Problem Model 4

Figure 2.1 Venn Diagram of Detection Problem.. 19

Figure 3.1 Random String Collision Rate with TCP/UDP Payload 34
Figure 3.2 Guardian Gateway Set.. 38
Figure 3.3 Intrusion Chain Tracing Model 38
Figure 3.4 SWT Architecture... 39
Figure 3.5 Active Intrusion Tracing Table 41
Figure 3.6 SIR Processing.. 41
Figure 3.7 Guardian Gateway Correlation... 43
Figure 3.8 Gateway Correlation Table... 45
Figure 3.9 Active Tracing Architecture.. 46
Figure 3.10 Active Tracing Protocol Operations... 47
Figure 3.11 Latency of SWT Guardian Gateway 51
Figure 3.12 Throughput of SWT Guardian Gateway (Kpps) 52
Figure 3.13 Throughput of SWT Guardian Gateway (KB/Sec) 53

Figure 4.1 CPF over IPD Correlation Windows $W_{j,s}$(X) and $W_{j+k,}$....................................... 58
Figure 4.2 Bounded Area of Correlation Points 60
Figure 4.3 Correlation Experiment on telnet and ssh 66
Figure 4.4 Correlation Point between Two Correlated Flows Detected by *MMS* 67
Figure 4.5 Correlation Points Detected by MMS, Stat, NDP1 and NDP2 with 69
Figure 4.6 True Positive and False Positive of Correlation between 16 and 15..................... 71
Figure 4.7 True Positive Rate of Correlation between 279 and 279, 400.............................. 72
Figure 4.8 False Positive of Correlation between 16 and 279 Uncorrelated Flows 73
Figure 4.9 False Positive of Correlation between 144 and 135 Uncorrelated Flows 73

Figure 5.1 Quantization of the scalar value *x* 81
Figure 5.2 Mapping between Unwatermarked *ipd* and Watermark ipd^w after Embedding
Watermark Bit *w* 82
Figure 5.3 Embedding/Decoding Watermark Bit over the Average of Multiple (*m*) IPDs.... 86
Figure 5.4 Embedding *l*-bit watermark into *l* sequences of IPDs 87
Figure 5.5 Probability Distribution of the Impact of Random Delays over the Average of
Multiple (*m*) IPDs 92
Figure 5.6 Distribution of Expected Watermark... 96
Figure 5.7 Comparison of 288 Selected IPDs before and... 99
Figure 5.8 Time Adjustment on Selected Packets by 100
Figure 5.9 Correlation True Positive Rates under Random Timing Perturbation 100
Figure 5.10 Correlation False Positive (Collision) Rate vs Hamming Distance Threshold *h*
.... 101

Figure 5.11 Correlation False Positive (Collision) Rate vs 102
Figure 5.12 Watermark Detection Rate vs Redundancy Number *m*.................................... 103
Figure 5.13 Watermark Detection Rate vs Hamming Distance Threshold *h* 104
Figure 5.14 Watermark Detection Rate vs Watermark Bit Number *l*.................................. 104
Figure 5.15 Correlation True Positive Rates under Self-Similar Timing Perturbation 105

Figure 6.1 Loopless Linear Connection Chain 114
Figure 6.2 The Loop Fallacy in Serializing Correlated Connections 114
Figure 6.3 Tracing Dilemma with Limited Observing Area.. 114
Figure 6.4 Point Connectivity $<_{PC}$ and Edge Connectivity $<_{EC}$ 121
Figure 6.5 Edge Serialization Based on Edge Connectivity $<_{EC}$ 121
Figure 6.6 *m* Incoming Connections Adjacent to *n* Outgoing Connections 122
Figure 6.7 Edge Serialization Based on Adjacent Connection Pair PE-Adj........................ 125

List of Tables

Table 2.1: Classification of Existing Tracing and Correlation Approaches 22

Table 4.1: Traces of Flows Used in Correlation Experiments 70
Table 4.2: Throughput (Millions per Second) of Correlation Point Calculation with Correlation Window Size 15 74

Table 5.1 Watermark Bit Robustness Estimation and Simulation with 93
Table 5.2 Watermark Detection Rates of Multiple Watermarks Embedded at the Same Time 107

Chapter 1

Introduction

1.1 Motivation

Network based attacks have become an increasingly serious threat to the critical information infrastructure we depend on daily. According to CERT/CC [22], the number of computer security incidents reported has been doubling every year since 1997. For example, there were 2,134 and 3,724 incidents reported to CERT in 1997 and 1998 respectively. The number of reported incidents has increased to 52,658 in 2001 and 82,094 in 2002. In 2003, there are 137,529 incidents reported.

Why is this the case? We believe there are two major factors that contribute to the growing trend of network based attacks:

- *The widespread proliferation of global Internet access*. At the beginning of 1990, there were about 1.1 million Internet users worldwide, about 86% of whom were in the US. By end of 2002, there were about 544 million Internet users, 70% of whom were outside the US. Now in many countries, people can easily access the Internet from an ISP, a public library, a café, a hotel, an airport and even a flying airplane. While the widespread global Internet access brings productivity and convenience in our daily life, it also enables the perpetrators to attack Internet hosts from virtually anywhere in the world.
- *The lack of effective source tracing and identification in the current Internet*. Unlike the telephone systems, the Internet was never designed for tracking and tracing users' behavior. Most existing network security mechanisms such as firewalls [38, 55], IPSEC [41] and IDS [34, 50, 57] are focused on intrusion prevention and detection. Until recently, intrusion tracing and response have been an afterthought and are generally limited to logging, notification and disconnection at local hosts. What is missing from existing network security mechanisms is an effective way to identify network based intruders and hold them accountable for their intrusions.

With widespread global Internet access, network based attackers have more possible points to launch attacks from and more targets to attack. Without effective intrusion source tracing and identification, those network based intruders have all the potential gains with virtually no risk of being caught. On the other hand, an effective and accurate attack tracing capability helps to eliminate network based attack from its root by identifying and catching those perpetrators responsible for the attack. From the attacker's point of view, if the risk of being caught and the consequent penalty are high enough compared with the potential gain of network based attack, he or she would be reluctant to attack again. Thus even an imperfect attack tracing capability could help to repel potential future attacks. However, because of the current Internet architecture, it is much easier for network based attackers to conceal their origin than for defenders to trace and identify their origin. Consequently, there is a pressing need to develop a capability for identifying the intruders responsible for an attack. Network based attack can not be effectively repelled or eliminated until its source is known.

1.2 Challenges in Tracing Intruders in the Internet — the Big Picture

In this dissertation, we refer to network-based attack as the process or event that an *attacker* sends out *attack packets* from an *attacker host*, through the network, to one or more *victim hosts* (or *target hosts*) and causes damage. The tracing (or traceback) problem of network-based attack is to identify the attack source. Ideally a tracing system, as a solution to the tracing problem, would be able to identify the person who is responsible for the attack. However, the identification or authentication of a user of a machine will not be addressed here as it is a generic problem to information assurance. In this research, the tracing problem is limited to determining the source host of the network-based attacks.

When attack packets reach the attack target, one important clue about the attack source is the source of attack packet. However, the source of attack packet is usually not the attack source as most attackers take advantage of one or more anonymity techniques to hide their origin. The tracing problem is to determine the attack source disguised by various anonymity techniques.

$$\begin{tikzcd} \boxed{H_{i-1}} \arrow[r, "P^i"'] & \boxed{H_i} \arrow[r, "P^{i+1}"'] & \boxed{H_{i+1}} \end{tikzcd}$$

P^i (src: H_{i-1}, dst: H_i, content, t)
P^{i+1} (src: H_i, dst: H_{i+1}, T(content),

Figure 1.1 Packet Transformation when Forwarded through Intermediate Host

Another technique to disguise the attack source is by "staging" the attack packets through some intermediate hosts. For example, the attack traffic may pass through a number of intermediate hosts, H_1, …H_n, before attacking the final target. When passing the intermediate hosts, the attack traffic will be forwarded with transformation. Figure 1.1 depicts such transformation at intermediate host. The reception of packet P^i at time t causes H_i to generate and send out a new packet P^{i+1} to H_{i+1} at time $t+\delta$. When packet P^i reaches H_i from H_{i-1}, it has source address H_{i-1}, destination address H_i. Now packet P^{i+1} has source address H_i and destination address H_{i+1}. Furthermore, the packet content of P^{i+1} could be transformed from that of P^i. For example, P^i from H_{i-1} to H_i could be a telnet packet and P^{i+1} from H_i to H_{i+1} could be a SSH packet. One invariant relation between packets P^i and P^{i+1} of the same attack traffic is the causality relation: the arrival of packet P^i at H_i at one time somehow causes H_i to generate and send out packet P^{i+1} to a new destination at a later time. However, the causality between packet P^i and packet P^{i+1} becomes difficult to recognize when the delay δ between the two packets is large.

Depending on the degree of transformation and connection characteristics, staging hosts along the attack path could be classified into three types: 1) *stepping-stone*, 2) *zombie*, and 3) *reflector*. A stepping-stone [71] is a compromised host that acts as a bidirectional-conduit for the attack traffic. The stepping-stone supports real-time bidirectional communication and it usually introduces very small delay δ. While the content of the packets could be changed drastically (by encryption for example), the essence of the packet content remains the same across the stepping-stone. Because stepping-stone supports bi-directional, interactive communications while conceals the source of the communication, it is frequently used in intrusion type of attacks. Classic penetration attack [71] usually comes through multiple stepping-stones.

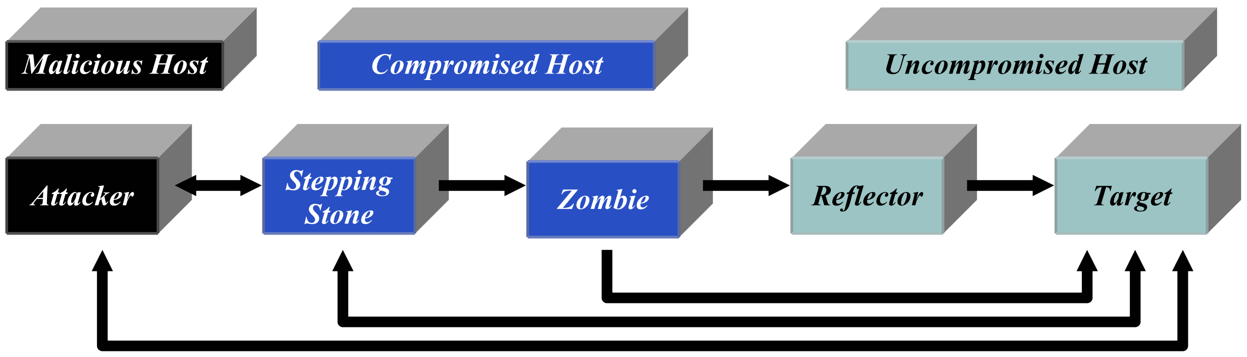


Figure 1.2 The Overall Tracing Problem Model

A zombie [83] host is a compromised intermediate host that is used as an attack launching point when triggered by the attacker. The trigger of the attack traffic from the zombie could be some special packet sent by the attacker or a Trojan or logic bomb previously planted by the attacker into the zombie host. The zombie is unidirectional and the attack triggering traffic and the triggered attack traffic are usually fundamentally different. For example, a single ping packet from the attacker to the zombie could trigger an enormous amount of essentially different attack traffic sent from zombie further to the target. The attacker could also plant into the zombie some logical bomb timed to execute minutes, hours or even days later. All these make it very difficult to identify the trigger of the attack traffic. This kind of zombie hosts has been widely used in distributed denial-of-service (DDoS) attacks.

Unlike stepping-stone and zombie host, a reflector is an uncompromised host that has been tricked to take part in the attack in an innocent manner that is consistent with its normal operation. For example, the attacker could send some host with ICMP request packet devised with attack target's address as its source address. Upon reception of such ICMP request, the host would think it comes from the attack target, and it would send the ICMP reply to the attack target. In such a way, the uncompromised host acts as the reflector of the attack traffic. When the attacker tricks many such reflectors into sending ICMP reply packets to one target, that target could be flooded. "Smurf" [19] is a well-known DDoS attack that utilizes an ICMP reflector to flood the target.

Figure 1.2 depicts a model of existing anonymity gaining techniques that a network-based attacker could use to conceal its real source. The double arrowed line represents bidirectional connection and the single arrowed line denotes one-way communication. Generally, zombie and reflector can only be used in unidirectional attack (i.e. denial of service attack [17,18,19,20]). Stepping-stone, however, could be used in both bi-directional, interactive break-in type of intrusions and unidirectional flooding type of attacks.

Analysis of the overall tracing problem model shows that there are two distinct sub-problems that are fundamental to the general tracing problem of network-based attacks:

1) To identify the real source of packet with spoofed source address.
2) To identify the causality of traffic into and out of the same intermediate host (or equivalently to identify which incoming flow, if there is any, causes a particular outgoing flow).

For the first sub-problem, IP traceback techniques [28,31,56,60,66,67] have been developed to trace packets with spoofed source address. However, IP traceback could not identify the causality of traffic through intermediate host thus it could not trace through intermediate hosts. For example, when attack traffic originating from host A is forwarded by host B toward host C, the victim at host C can use IP traceback technique to find the attack traffic comes from host B (even if the attack traffic from B to C has a spoofed source address). But IP traceback techniques could not determine that the attack traffic from B to C actually originated from host A. In order to trace through the intermediate host B, we need techniques that could identify which, if any, incoming flow into host B causes the attack flow from B to C.

The stepping-stone is a special case of staging intermediate host, and it is attractive to the attacker due to the following reasons. First, a stepping stone supports real-time bidirectional communication, thus it conserves all the attack capability from the attack source. Second, it just needs normal user’s access to convert any networked host into a stepping stone. Third, the stepping stone makes the intrusion source tracing significantly harder. All these make the using of stepping stones one of the most widely used and effective attack source concealment techniques [90].

Since the stepping stone acts merely as a bidirectional conduit, it will send out the packets it has received after some transformation has been performed. Therefore the identification of the causality of connections through a stepping-stone could be achieved by matching the incoming connections with outgoing connections of the stepping-stone. We refer this matching as the *correlation* of connections. One major challenge in the correlation is that the connections may be transformed in content or other flow characteristics. Another challenge comes from the fact that the stepping stone may be unknown before any correlation has been identified. Determining which connections need to be compared becomes non-trivial when there are a large number of concurrent connections in the network. Further more if the attack passes some hosts outside the observing area of the tracing system, correlation is needed among those connections separated by unknown number of intermediate stepping-stones. This requires the capability to correlate any two connections observed at different points in the network. Efficient scoping of the connections to be matched is needed to make real-time tracing through stepping-stones practical.

In this research, we focus on addressing how to trace the interactive intrusion connections through stepping-stones at real-time.

1.3 Challenges in Tracing through Stepping Stones

Tracing attack connections through stepping stones is challenging, because there are a number of techniques available for the attackers to obscure the correlation of intrusion connections across the stepping stones. They include:

1) Host Login Information Disguise, Deletion and Forgery
2) Connection Content Transformation (i.e. encryption, compression)
3) Timing Perturbation
4) Introducing Loops to the Intrusion Connection Chain.
5) Traffic Padding (i.e. adding bogus packet, packet padding)
6) Packet Drop and Retransmission
7) Flow Repacketization
8) Packet Reorder
9) Flow Split and Merge

10) Mixing Multiple Flows (i.e. tunneling)

These evasive techniques represent different classes of countermeasures. They can be used either separately or jointly to make the correlation of intrusion connections through stepping stones much more difficult. Existing intrusion tracing and correlation schemes have not yet been able to addressed all these countermeasures and thus are not effective when all the countermeasures are used by intruders.

In this dissertation, we analyze these countermeasures in detail and develop solutions to the first four classes, using information hiding techniques and active approach. The rest of this section summarizes the problems caused by these countermeasures.

1.3.1 Host Login Information Disguise, Deletion and Forgery

Disguise, deletion and forgery of local host login information aim to defeat any host based tracing and correlation approach. While the attackers do not need root access of a host to make it a stepping stone, attackers usually use some previously compromised hosts as stepping stones. In this case, the attacker could have total control of the stepping stone, and could easily modify, remove and even forge any connection login information. If one stepping stone is providing misleading login and host activity information, the whole host-based tracing systems would be fooled. In addition, the attacker could install some backdoor[1] at the stepping stone to bypass the normal login process. Furthermore, the attacker could install some back door relay at the stepping stone that forwards the incoming connection to some outgoing connection of different type. For example, netcat at one host could relay an incoming TCP connection to a predefined outgoing UDP connection to some other host.

Earliest work on tracing intruders behind stepping stones were based on tracking users' login activities at different hosts [40,65], and they are vulnerable to above host login information disguise. To overcome this shortcoming, tracing and correlation based on connection content have been developed [69,82].

[1] An undocumented way of gaining access to a program, online service or an entire computer system.

1.3.2 Connection Content Transformation

One fundamental limitation of any network content based correlation and tracing approach is that it requires that the payload content of the intrusion connections remain the same across stepping stones. Therefore, network content based correlation and tracing approaches are vulnerable to connection transforms that changes the connection payload.

The connection content transforms an attacker could use include payload encryption, payload compression and payload padding. In particular, payload encryption is easily achievable through widely available SSH [54,87] and IPSEC[41,37]. The encryption of connection content defeats any content based correlation and tracing.

To address this countermeasure, inter-packet timing based correlation has been proposed. While the timing based correlation is currently the most promising and capable approach in correlating encrypted connections through stepping stones, it is also subject to a number of countermeasures that are specifically designed to obscure the inter-packet timing.

1.3.3 Timing Perturbation

To disguise encrypted connections from being correlated by timing based approaches, the attacker could actively perturb the timing characteristics of a connection by selectively or randomly introducing extra delays at some stepping stone when forwarding packets. The timing perturbation could either make unrelated flows have similar timing characteristics or make related flows exhibit different timing characteristics, which would increase the false positive rate and decrease the true positive rate of timing based correlation respectively.

In the extreme (but unrealistic) case, adding extra delays at the stepping stones could make all flows exhibit very similar timing characteristics. For example, a stepping stone could, in theory, buffer all packets of each flow, and later flush out all packets in burst). However, due to the real-time constraint of interactive connections, there is an upper bound on the delays that any stepping stone could introduce when forwarding packets. The upper bound of the delays the adversary could add imposes some inherent limit of the adverse impact by the timing perturbation.

1.3.4 Introducing Loops to the Intrusion Connection Chain

Another way to confuse the correlation of connections across stepping stones is to have the intrusion connections passing some stepping stones more than once. This would make some stepping stones have more than one incoming connection and one outgoing connection correlated. The multiple correlated incoming and outgoing connections at a stepping stone would make the serialization of those correlated connections nondeterministic. In case some intrusion connections are out of the observing scope of the tracing system, it could be difficult for the stepping stone with multiple correlated incoming and outgoing connections to determine the right direction from which the intrusion really comes from. Therefore, even a perfect correlation solution, which gives us all and only those correlated connections, is not adequate to construct the complete and accurate intrusion path when there is loop in it.

1.3.5 Flow Level Traffic Padding

To disguise encrypted connections from being correlated by timing based approaches, the attacker could, at some stepping stones, introduce bogus packets[2] into the flow he wants to disguise. The injected bogus packets would change the inter-packet timing characteristics of the disguised flow. For example, assume flow f_{AB} from host A to B is correlated to flow f_{BC} from host B to host C, then the number of packets per unit time and the inter-packet timing characteristics of flow f_{AB} would be very similar to those of flow f_{BC}. If the attacker adds some bogus packets to flow f_{BC} at host B, the original flow f_{BC} would be turned into some other flow f'_{BC} with substantially different inter-packet timing characteristics, which would obscure the correlation of flow f_{AB} and f_{BC} based on inter-packet timing characteristics.

Obviously adding bogus packets wastes useful network bandwidth. This would in a way limit how many bogus packets could be injected by the attackers.

1.3.6 Packets Drop and Retransmission

Another way to disguise encrypted connections from being correlated by timing based approaches is to selectively or randomly drop some packets at some stepping stones. By

[2] The packets that will be ignored or removed at the receiver end without affecting the semantics of original flow

reducing the number of packets in the flow, the inter-packet timing characteristics are changed. For a connection based on reliable transport protocols such as TCP, packet dropped could trigger packet retransmission, which in turn further obscures the inter-packet timing characteristics of the flow.

Packet drop and retransmission also waste useful network bandwidth.

1.3.7 Flow Repacketization

Flow repacketization refers to either splitting one bigger packet into multiple smaller packets or merging several adjacent packets into one bigger packet in the same flow. Each case would change the original inter-packet timing characteristics.

Flow repacketization generally does not waste useful network bandwidth except increased IP header overhead. Packet splitting could be used as long as the packet has a payload bigger than one byte. Merging adjacent packets is only feasible for flows with short inter-packet timing due to the real-time constraint.

1.3.8 Packet Reorder

To disguise encrypted connections from being correlated by timing based approaches, the attacker could, at some stepping stone, buffer some packets of a flow and later send out buffered packets with different order and inter-packet timing.

Packet reorder generally does not waste useful network bandwidth. Similar to the packet merging case of flow repacketization, packet reordering is only feasible for flows with short inter-packet timing. The exact number of packets can be buffered before flushing is determined by the maximum tolerable delays and the inter-packet timing of the original flow.

1.3.9 Flow Split and Merge

Another way to disguise encrypted connections from being correlated by timing based approaches is to split the flow at a stepping stone into multiple flows and later merge the multiple flows at the next stepping stone. The flow split and merge can be done by simply changing the IP header of the packets after which the packets would belong to some different flows. By splitting one flow of packets into multiple flows of packets, the inter-packet timing characteristics of original flow is lost.

Flow split and merge does not waste useful network bandwidth as no new packet is introduced and there is no change on the payload part of the packets.

1.3.10 Mixing Multiple Flows

To disguise encrypted connections from being correlated by timing based approaches, the attacker could mix and encapsulate packets from multiple flows at a stepping stone to form a new flow. The encapsulated packets could be encrypted so that the outside observer has no way to determine which encapsulated packet is from which flow originally. An example of such a packet mixing and encapsulation is IPSEC tunnel mode. Similar techniques have been proposed to build anonymous communication [3, 58, 73].

1.4 Contributions

In this dissertation, we address the tracing and correlation problem of intrusion connections through stepping stones under various settings and we analyze the theoretical limits of correlation only approaches. We design, implement and evaluate solutions to the first three classes of countermeasures and fill the gap between the perfect correlation solution and perfect tracing solution.

For unencrypted intrusion connections through stepping stones, we design and implement a novel intrusion tracing framework called Sleepy Watermark Tracing (SWT), which applies principles of steganography and active networking. SWT is "sleepy" in that it does not introduce overhead when no intrusion is detected. Yet it is "active" in that when an intrusion is detected, the host under attack will inject a watermark into the backward connection of the intrusion, and wake up and collaborate with intermediate routers along the intrusion path. Our prototype shows that SWT can trace back to the trustworthy security gateway closest to the origin of the intrusion, with only a single packet from the intruder. With its unique active tracing, SWT can even trace when intrusion connections are idle.

Encryption of connections through stepping stones defeats any content based correlation and makes correlation of intrusion connections more difficult. Based on inter-packet timing characteristics, we develop a novel correlation scheme of both encrypted and unencrypted connections. We show that (after some filtering) inter-packet delays (IPDs) of both encrypted and unencrypted, interactive connections are preserved across many router hops and stepping stones. The effectiveness of IPD based correlation requires that timing characteristics be distinctive enough to identify connections. We have found that normal interactive connections such as telnet, SSH and rlogin are almost always distinctive enough to provide correct correlation across stepping stones.

The timing perturbation at intermediate stepping stones of packet flows poses additional challenge in correlating encrypted connections through stepping stones. The timing perturbation could either make unrelated flows have similar timing characteristics or make related flows exhibit different timing characteristics, which would either increase the false positive rate or decrease the true positive rate of timing-based correlation. To address this new challenge, we develop a novel watermark based correlation scheme that is designed to be specifically robust against such kinds of timing perturbation. The idea is to actively embed a unique watermark into the flow by slightly adjusting the timing of selected packets of the flow. If the embedded watermark is unique enough and robust enough against the timing perturbation by the adversary, the watermarked flow could be uniquely identified and thus effectively correlated. By utilizing redundancy techniques, we develop a robust watermark correlation framework that reveals a rather surprising result on the inherent limits of independent and identically distributed (*iid*) random timing perturbations over sufficiently long flows. We also identify the tradeoffs between the defining characteristics of the timing perturbation and the achievable correlation effectiveness. Our experiments show that our watermark based correlation performs significantly better than existing passive timing based correlation in the face of random timing perturbation.

We use set theoretic approach to analyze the theoretical limits of the correlation-only approach and demonstrate the gap between the perfect correlation-only approach and the perfect solution to the tracing problem of stepping stones. In particular, we identify the serialization problem and the loop fallacy in tracing connections through stepping stones. We formally demonstrate that even with perfect correlation solution, which gives us all and only those connections that belong to the same connection chain, it is still not adequate to serialize the correlated connections in order to construct the complete intrusion path deterministically. We further show that the complete set of correlated connections, even with loops, could be serialized deterministically without synchronized clock. We present an efficient intrusion path construction method based on adjacent correlated connection pairs.

In this research, we learn some general lessons about tracing and correlating intrusion connections through stepping stones. Specifically, we demonstrate the significant advantages of active correlation approach over passive correlation approaches in the presence of active countermeasures. We also demonstrate that information hiding and redundancy techniques can be used to build highly effective intrusion tracing and correlation frameworks.

1.5 Organization

The remainder of this dissertation is organized as follows. In Chapter 2, we first formulate the tracing problem and correlation problem of intrusion connections through stepping stones. We then review related work in tracing and correlating connections through stepping stones. In particular, we assess previous work in the presence of the various countermeasures.

In Chapter 3, we describe Sleepy Watermark Tracing (SWT) in detail. After presenting the SWT architecture, we analyze SWT and discuss our prototype implementation. We then present our experimental results about SWT.

In Chapter 4, we address the challenges posed by encryption of intrusion connections through Inter-Packet Delay (IPD) based correlation. We first present our correlation solution model, we then present four different correlation point functions. We evaluate the correlation effectiveness of our correlation metrics in a number of experimental settings.

In Chapter 5, we present IPD-based watermark correlation that is designed specifically to be robust against 1) host login information corruption; 2) network content transformation; and 3) random timing perturbation by adversary. We first describe the basic embedding of a single watermark bit into the inter-packet timing domain, then present a probabilistically robust watermark bit embedding scheme over multiple IPDs. After analyzing the watermark bit robustness and tradeoffs, we further analyze the overall watermark detection and collision probability and formally establish the robustness strength of our watermarking scheme. We empirically evaluate the correlation effectiveness and validate the tradeoffs of our watermark based correlation.

In Chapter 6, we analyze the theoretical limits of correlation-only approach in solving the problem of tracing intrusion connections through stepping stones. We identify the gap between the perfect correlation solution and the perfect tracing solution, and we show what it takes to bridge the gap.

In Chapter 7, we summarize our work and contributions. We conclude this dissertation with a discussion of the general lessons learned about the robust and effective correlation and tracing of intrusion connections through stepping stones and outline some directions for future research.

Chapter 2

Stepping Stone Tracing Problem and Related Works

In this chapter, we formally formulate the stepping stone tracing problem, and we review and evaluate previous related works.

2.1 The Overall Stepping Stone Tracing Problem Model

In this dissertation, we refer to one network communication between two hosts in the Internet as one *connection*. The host that originates the connection is called the source host, and the host that terminates the connection is called the destination host. The connections that originate from one host are called the outgoing connections of the host, and the connections that terminate at one host are called the incoming connections of the host. One host may have multiple outgoing connections and multiple incoming connections at the same time. In the Internet environment, a connection consists of flows of packets between the source host and the destination hosts, and it could be either unidirectional or bidirectional. Each packet in a connection has at least one header and optional payload, and each packet has a departure time from the source host and arrival time at the destination host. Generally, a connection can also be called flow, which can be uniquely identified by tuple

<Source IP, Source Port, Destination IP, Destination Port, Protocol>

Given a series of computer hosts H_1, H_2, … H_{n+1} (n>1), when a person (or a program) sequentially connects from H_i into H_{i+1} (i=1,2, … n), we refer to the sequence of connections $<c_1, c_2, \dots c_n>$, where $c_i=<H_i, H_{i+1}>$ (i=1, …n), as a *connection chain* on $<H_1, H_2, \dots H_{n+1}>$, and intermediate hosts H_2, … H_n as *stepping stones*. Here connection c_i terminated at host H_{i+1} causes another connection c_{i+1} outgoing from host H_{i+1} and host H_{i+1} essentially forwards the traffic of c_i to c_{i+1}. All c_i's are always distinct, but not all H_i's are always distinct. In case some host appears more than once in sequence $< H_1, H_2, \dots H_{n+1}>$, there exists loop in the connection chain $<c_1, c_2, \dots c_n>$.

The *tracing problem* of a connection chain through stepping stones is, given c_n of some unknown connection chain $<c_1, c_2, \ldots c_n>$ ($n>1$), to identify $<c_1, c_2, \ldots c_n>$.

Any particular connection chain $<c_1, c_2, \ldots c_n>$ is a sequence of connections. We refer those connections within the same connection chain as *correlated* to each other and the corresponding set $\{c_1, c_2, \ldots c_n\}$ as the *set of correlated connections* or *correlation set*. This can be formally modeled by a binary relation on the overall connection set. Let $\hat{C}$ represent all connections being examined, we define binary relation *CORR* on the overall connection set $\hat{C}$ such that

$$\forall c, c' \in \hat{C}\ (c\ \mathit{CORR}\ c'\ \mathit{iff}\ (c \in \{c_1, c_2, \ldots c_n\} \Rightarrow c' \in \{c_1, c_2, \ldots c_n\})) \qquad (2.1)$$

It is obvious that *CORR* is specific to correlation set and it is 1) self-reflexive; 2) symmetric and 3) transitive. Therefore binary relation *CORR* is an equivalence relation on $\hat{C}$ and it partitions the overall set of connections into a particular set of correlated connections and else.

Because connection chain $<c_1, c_2, \ldots c_n>$ is a sequence of connections, each c_i has a unique order number $Ord(c_i)$ associated with it. The overall ordering information of $<c_1, c_2, \ldots c_n>$ can be formally modeled by the binary relation $\angle$ on $\{c_1, c_2, \ldots c_n\}$ such that

$$\forall c, c' \in \{c_1, c_2, \ldots c_n\}\ (c \angle c'\ \mathit{iff}\ Ord(c) < Ord(c')) \qquad (2.2)$$

It is obvious that $\angle$ well orders set $\{c_1, c_2, \ldots c_n\}$ and it uniquely determines $<c_1, c_2, \ldots c_n>$ from $\{c_1, c_2, \ldots c_n\}$.

For any particular connection chain $<c_1, c_2, \ldots c_n>$, there exists unique binary relation *CORR* and $\angle$, which in turn uniquely determines $<c_1, c_2, \ldots c_n>$. Therefore, the overall tracing problem of stepping stone can be divided into the following sub-problems:

1) *Correlation Problem*:

Given c_n of some unknown connection chain $<c_1, c_2, \ldots c_n>$, identify set $\{c_1, c_2, \ldots c_n\}$; Or equivalently, given any two connections c and c', determine if c *CORR* c'.

2) *Serialization Problem*:

   Given a set of correlated connections $C=\{c_1, c_2, \ldots c_n\}$ and some information about the relative order of correlated connections, serialize $\{c_1, c_2, \ldots c_n\}$ into a sequence $<c_1', c_2', \ldots c_n'>$ ($c_i' \in C$, i=1, … n) such that $c_i' \angle c_{i+1}'$ (i=1, … n-1); Or equivalently, given any two connections c and c', determine if c $\angle$ c' or c' $\angle$ c.

The result of solution for the correlation problem is an unordered set of connections and the result of solution for the serialization problem is a sequence of connections. It is easy to see that the solution of the serialization problem is based upon the result of the correlation problem solution. Therefore, the correlation problem has to be solved first in order to solve the overall tracing problem of stepping stones.

2.1.1 Correlation Problem Solution Model

Given binary relation *CORR*, we can define corresponding *correlation function CF*: $\hat{C} \times \hat{C} \rightarrow \{0, 1\}$ such that:

$$CF(c,c') = \begin{cases} 1 & c \ CORR \ c' \\ 0 & otherwise \end{cases} \tag{2.3}$$

Therefore the correlation problem can be equivalently expressed as: given any two connections c and c', find correlation function *CF* such that *CF*(c, c')=1 *iff* c *CORR* c'.

In practice, connection correlation analysis is based on the characteristics of the connections, which may include connection content, header information (such as packet size) and inter-packet timing. The connection characteristics can be modeled by a *metric function* of the connection

$$M: \hat{C} \times P \rightarrow Z \tag{2.4}$$

where $\hat{C}$ is the set of connections to be correlated, P is some domain of parameters and Z is the connection metric domain.

Based on connection metric, a *correlation value function* (*CVF*) can be defined as

$$CVF: Z \times Z \times \delta \rightarrow \{0, 1\} \tag{2.5}$$

where δ is some threshold[3]. The result of *CVF* is either 0 or 1 indicating whether the two connections are detected to be correlated based on their corresponding correlation metric and threshold δ.

In other word, connections *c* and *c'* are considered correlated *iff*

$$CVF(M(c, p), M(c', p), \delta) = 1 \tag{2.6}$$

Therefore the correlation problem is now translated to: *find or construct M, p, CVF and δ such that*

$$\forall c_i \in \langle c_1, c_2, \ldots c_n \rangle\ \forall c\ [CVF(M(c_i, p), \mathrm{M}(c, p), \delta) = 1\ \textit{iff}\ c \in \{c_1, c_2, \ldots c_n\}] \tag{2.7}$$

In finding M, p, CVF and δ, the key is to identify those unique characteristics of connections that are invariant across routers and stepping-stones. If those identified invariant characteristics of connections are unique enough to exclude other uncorrelated connections, reliable correlation of connections could be constructed from these metrics. In case the original connections do not have unique enough correlation metric, the original connections could be actively but slightly adjusted to make their correlation metric more unique so that the connections could be correlated more effectively.

2.2 Evaluation Criteria and Classification of Tracing and Correlation

[3] Depending the metric function M and parameter P, threshold δ can be real value, integer or other appropriate form.

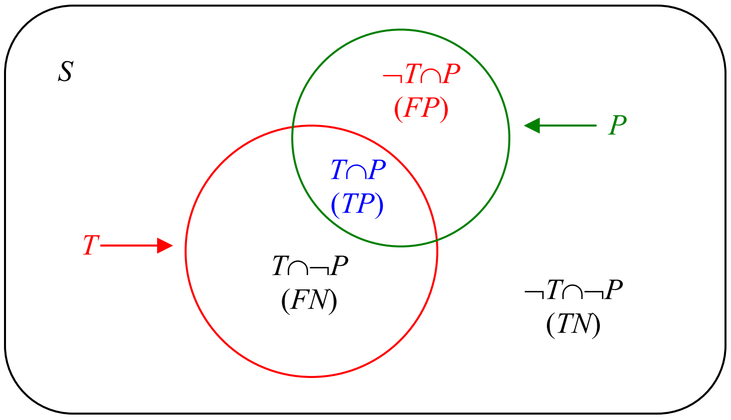


Figure 2.1 Venn Diagram of Detection Problem

Approaches

2.2.1 Evaluation Criteria of Tracing and Correlation Approaches

According to the overall tracing problem model, both correlation and serialization problems can be modeled as detection problem – determining if certain property exists or not. Therefore, the stepping stone tracing and correlation approaches can be evaluated by the generic detection problem assessment criteria.

Assume p is the property to be detected. Let S denote the whole space in which property p could be true, let T denote the space within S in which property p is true, and let P denote the space within S that the detector reports positive of property p. Therefore $T \subseteq S$, $T + \neg T = S$, $P \subseteq S$ and $P + \neg P = S$. Figure 2.1 illustrates the relations between S, T and P. A perfect detector would generate P exact the same as T, and a less than perfect detector would generate P that either misses part of T or mistakenly includes part of $\neg T$.

Set of $T \cap P$ is considered *true positive* (TP), set of $\neg T \cap P$ is considered *false positive* (FP), set of $T \cap \neg P$ is considered *false negative* (FN) and set of $\neg T \cap \neg P$ is considered *true negative* (TN). A TP indicates the case that a positive detection is actually correct in that the property p exists. A FP indicates the case that a positive detection is actually wrong in that the property p does not exist.

The conditional probability $Pr(P|T) = Pr(T \cap P)/Pr(T)$ is considered as *true positive rate*

(TPR), which represents the probability of positive detection assuming the property p exists. The TPR quantitatively expresses the "completeness" of the detector. In Figure 2.1, the TPR is represented by the ratio between area of $T \cap P$ and area of T.

The conditional probability $Pr(P|\neg T) = Pr(\neg T \cap P)/Pr(\neg T)$ is considered as *false positive rate* (FPR), which represents the probability of positive detection assuming the property p does not exist. The FPR quantitatively express the "soundness" of the detector. In Figure 2.1, the FPR is represented by the ratio between area of $\neg T \cap P$ and area of $\neg T$.

The conditional probability $Pr(\neg P|T) = Pr(T \cap \neg P)/Pr(T)$ is considered as *false negative rate* (FNR), which represents the probability of negative detection assuming the property p exists. In Figure 2.1, the FNR is represented by the ratio between area of $T \cap \neg P$ and area of T.

The conditional probability $Pr(\neg P|\neg T) = Pr(\neg T \cap \neg P)/Pr(\neg T)$ is considered as *true negative rate* (FNR), which represents the probability of negative detection assuming the property p does not exist. In Figure 2.1, the TNR is represented by the ratio between area of $\neg T \cap \neg P$ and area of $\neg T$.

It is easy to see that TPR+FNR=1 and FPR+TNR=1. Therefore we only need to consider correlation true positive rate and correlation false positive rate when assess a correlation solution.

The solution of tracing problem can be characterized by three criteria: *usability*, *effectiveness,* and *robustness*. In particular, usability is a measure of applicability of the tracing and correlation system, which usually includes 1) assumptions of the attacks; 2) required information for tracing; 3) limitation of the tracing system; and 4) overhead introduced. For example, some tracing approach needs sustained steady packet streams to be useful, while some other tracing approaches may need only a few packets to be effective. Ideally, a tracing solution should have as few as possible assumptions, require as few as possible information for tracing, have as few as possible limitations, and have as few as possible overhead to be effective. Effectiveness is the measure of usefulness and correctness of tracing result under various conditions. It can be quantitatively expressed by true positive rate (TPR) and false positive rate (FPR). Usually the correlation true positive rate and false positive rate are conflicting in that higher true positive rate usually causes higher false positive rate and lower false positive rate usually causes lower true positive rate. Each correlation solution has an inherent tradeoff between the true positive rate and the false positive rate. Ideally, a tracing solution should have as high as possible true positive rate and as low as possible false positive at the same time. The limit of achievable true positive rate and false positive rate at the same time in a way measures the inherent difficulty of the tracing problem. Robustness refers to the capability to withstand active countermeasures by the adversary to further disguise its source and identity from being identified. It could be quantitatively expressed by the countermeasure's negative impact over the tracing and correlation effectiveness. Ideally, the adverse impact of adversary's countermeasures over the true positive and false positive rates of a robust tracing solution should be minimal.

2.2.2 Classification of Tracing and Correlation Approaches

Based on source of tracing or correlation information, the stepping stones tracing and correlation approaches can be divided into two categories: *host-based* and *network-based*. Host-based approaches rely on information collected at the hosts that are used for stepping stones. Such information includes user login activity, new arrival of connections and new initiation of connections to other hosts. Network-based approaches use some characteristics of network connections and exploit the property of network connections: the essence or semantics of the application level content of connections is invariant[4] across stepping stones.

Based on how the traffic is traced, tracing approaches can further be classified into either *active* or *passive*. In particular, passive approaches passively monitor and compare all the traffic all the time and they do not scope the traffic to be traced. On the other hand, active approaches may actively but slightly change some characteristics of selected packets in order to make the packet flow easier to identify and correlate. Furthermore, active approaches could dynamically control when, where, what and how the traffic is to be correlated through

[4] The form of application level content of connections could be transformed by (i.e. encryption, compression).

customized packet processing, and they only trace “interested” traffic when needed. Table 2.1 provides a classification of existing tracing and correlation approaches as well as our tracing and correlation approaches[5].

	Passive	**Active**
Host-based	DIDS CIS STOP	
Network-based	Thumbprinting ON/OFF-based Deviation-based **IPD-based** Wavelet-based	IDIP **SWT** **IPD-Watermarking**

Table 2.1: Classification of Existing Tracing and Correlation Approaches

The fundamental problem of host-based tracing approach is that it requires the participation of each stepping stones and it places its trust upon those monitored stepping stones themselves. In specific, the host-based tracing approach depends on the correlation of connections at every stepping stone in the intrusion connection chain. If one stepping stone provides misleading correlation information, the whole tracing system is fooled. On the other hand, network-based approach does not require participation of stepping stones, nor does it place its trust on the monitored stepping stones. It only requires the information about the connections to and from stepping stones.

The earliest work on connection correlation was host-based which tracks the users’ login activity at different hosts. Later work has been network-based that exploits different characteristics of network connections. The earlier network-based approaches relied on comparing the packet contents of the connections to be correlated, and recent network-based approaches have focused on the timing characteristics of connections, in order to correlate encrypted connections (i.e. traffic encrypted using IPSEC [41] or SSH [54,87]).

[5] printed in bold font

2.3 Tracing and Correlating Unencrypted Connections

The tracing and correlating of unencrypted connections through stepping stones have been studied since the earliest works on the tracing problem of stepping stones. Notable works include:

- Distributed Intrusion Detection System (DIDS)
- Caller Identification System (CIS)
- Thumbprinting
- Intrusion Detection and Isolation Protocol (IDIP)
- Session TOken Protocol (STOP)

In the rest of this section, we review and evaluate these works.

2.3.1 Distributed Intrusion Detection System (DIDS)

Distributed Intrusion Detection System (DIDS) [65] developed at UC Davis was designed to address the intrusion detection problem in LAN environment. To the best of our knowledge, it is the first work that tracks users' login activity across network. DIDS uses a host-based distributed architecture to keep track of all the users in the LAN through so-called NID (Network-user Identification) and account for all activities to network-wide intrusion detection systems. Each monitored host in the DIDS domain collects audit trails and sends audit abstracts to a centralized DIDS director for analysis. Besides the inherent limitations of host-based tracing approach, DIDS is limited to tracking users' login activity across the LAN through normal login within the DIDS domain. Furthermore, because of its centralized monitoring of network activities, it seems not feasible in large-scale network such as the Internet.

2.3.2 Caller Identification System (CIS)

The Caller Identification System (CIS) [40] is another host-based tracing mechanism. It eliminates centralized control by utilizing a distributed model. In CIS, each host assumes each remote user who tries to login has a connection chain, and each host keeps record about its view of the login chain for each logged in user. When the user from host H_{n-1} attempts to login into the host H_n, H_n asks H_{n-1} about its view of the login chain of that user, which should be H_1, H_2, … H_{n-1} ideally. After getting the login chain information from H_{n-1}, H_n queries each host in the login chain (ideally H_1, H_2, … H_{n-1}) about their views of the login chain for the user who tries to login into H_n. Only when the login chain information from all queried hosts matches, will the login be granted at host H_n. Besides the inherent limitations of host-based approach, CIS introduces excessive overheard to the normal login process by requesting and reviewing information from every hosts along the login chain. In addition, CIS requires the capability to correlate incoming connection and outgoing connection at each stepping stone, which was not available when CIS was proposed. Later work of STOP [11] recognized this and tried to provide a way to determine the correlation of incoming and outgoing connections at a host.

2.3.3 Thumbprinting

Thumbprinting [69] is the first published network-based correlation technique. It utilizes a small quantity of information (called thumbprint) to summarize a certain section of a connection. The thumbprint is built, through principle component analysis technique in statistics, upon the frequencies that each character occurs within a period of time. Ideally it can distinguish a connection from unrelated connections and correlate a connection with those related connections in the same connection chain. Because it correlates based on connection content, thumbprinting works even when all stepping stones are compromised and under attacker's total control, and it can be useful when only part of the Internet implements thumbprinting. However, thumbprinting depends on clock synchronization to match the thumbprints of corresponding intervals of connections, and it is vulnerable to packet retransmission variation. One area that thumbprinting has not addressed is how to determine which connections are to be thumbprinted and how to determine which thumbprint should be matched with which thumbprint in order to find correlated connections.

2.3.4 Intrusion Detection and Isolation Protocol (IDIP)

IDIP (Intrusion Identification and Isolation Protocol) [61,62] is a proposal by Boeing's Dynamic Cooperating Boundary Controllers Program that uses an active approach to trace the incoming path and source of intrusion. In the proposal, special gateways called boundary controllers collaboratively locate and block the intruder by exchanging intrusion detection information, namely, attack descriptions. If the distributed boundary controllers are able to detect the ongoing attack described by the attack description, IDIP could identify the ongoing attack path by querying appropriate boundary controllers. While it does not require any boundary controller to record any connections for correlation, its intrusion tracing is closely coupled with intrusion detection. The effectiveness of IDIP depends on the effectiveness of intrusion identification through the attack description at each boundary controller. Therefore IDIP requires each boundary controller to have the same intrusion detection capability as the IDS at the intrusion target host. It is questionable whether the intermediate boundary controller is able to identify an intrusion at real-time based on a hard-coded attack description.

2.3.5 Session Token Protocol (STOP)

Session Token Protocol (STOP) [11] aims to find the mapping between the incoming and outgoing connections at a host. It is based upon the Identification Protocol (ident) defined in RFC1413, which enables the server side of a TCP connection to ask the client side about the process and corresponding UID that initiated the TCP connection. STOP extends the Identification Protocol by allowing the host to save application-level data about the process and user that opened the connection, and to send request to other host recursively. By saving and examining information about processes that originate the outgoing connection or terminate the incoming connection, STOP tries to map an incoming connection to a host with an outgoing connection from the host. However, there are few fundamental flaws in the design of STOP that severely limits its usability. First, STOP is a host-based protocol whose functionality depends on the correct information collected at the stepping stone itself. As the attacker is usually assumed to have total control over the stepping stone, he can easily kill or replace the STOP daemon running at the stepping stone, which would completely defeat the STOP system. Second, even if the STOP daemon is not touched by the attacker, it is still not guaranteed to provide the mapping between the incoming and outgoing connections at a host. This is because one process in a host could open multiple sockets and handles multiple incoming connections concurrently. Therefore, even if the STOP could find the process P and its corresponding UID that opened an outgoing connection C_o, it may not be able to determine which one of the multiple incoming connections handled by process P should be mapped to the outgoing connection C_o. In other word, STOP only works when 1) every stepping stone runs STOP daemon; and 2) no STOP daemon at each stepping stone is stopped or replaced by the attacker; and 3) each process that relays connections at each stepping stone opens only one incoming connection and one outgoing connections. In summary, it is questionable how useful the STOP would be in real-work situations.

2.4 Tracing and Correlating Encrypted Connections

As the connection encryption tools (such as IPSEC and SSH) have been widely deployed, network based attackers can easily encrypt their attack connections when passing stepping stones. To address this new challenge, recent research work on the stepping stones tracing problem has been focused on how trace and correlate encrypted connections through stepping stones.

2.4.1 ON/OFF Based

The ON/OFF based correlation [90] by Zhang and Paxson is the first network-based correlation scheme that utilizes the inter-packet timing characteristics to correlate interactive connections across stepping-stones. Depending on whether there is any traffic for a (adjustable) period of time, the duration of a flow can be divided into either ON of OFF periods. The correlation of two flows is based on mapping the ends of OFF periods (or equivalently the beginnings of ON periods). Because it correlates based on inter-packet timing characteristics rather than packet content, ON/OFF based correlation is able to correlate both encrypted and unencrypted connections, and it is robust against packet payload padding. However, ON/OFF based correlation requires that the packets of connections have precise, synchronized timestamps in order to be able to correlate them. This makes ON/OFF based correlation limited to detecting the correlation between only those connections that can be monitor at the same one point. And it is difficult or impractical for ON/OFF based correlation to correlate measurements taken at different points in the network.

2.4.2 Deviation Based

The deviation-based approach [88] by Yoda and Etoh is another network-based correlation scheme. It defines the minimum average delay gap between the packet streams of two TCP connections as deviation. The deviation based approach considers both the packet timing characteristics and the TCP sequence numbers. It does not require clock synchronization and is able to correlate connections observed at different points of network. However, it can only correlate TCP connections that have one-to-one correspondences in their TCP sequence numbers, and thus is not able to correlate connections where padding is added to the packet payload (e.g. when certain types of encryptions are used).

The deviation based approach has been evaluated against several large network traces, and it has been shown that it is rare to have low deviation between random uncorrelated flows. This suggests that deviation based approach has low false positive rates. However, the published paper by Yoda and Etoh does not have evaluation on the correlation true positive rates.

2.5 Tracing and Correlating Encrypted Connections with Timing Perturbation

The timing-based approach is the most capable and promising current method for correlating encrypted connections. However, existing timing-based correlation approaches, are vulnerable to countermeasures by the attacker. In particular, the attacker can perturb the timing characteristics of a connection by selectively or randomly introducing extra delays when forwarding packets at the stepping stone. This kind of timing perturbation will adversely affect the effectiveness of any timing-based correlation. The timing perturbation could either make unrelated flows have similar timing characteristics, or make related flows exhibit different timing characteristics. Either case could cause a timing-based correlation method to fail.

2.5.1 Wavelet Based

Donoho et al [29] have recently investigated the theoretical limits on the attacker's ability to disguise his traffic through timing perturbation and packet padding (i.e., injection of bogus packets). By using a multiscale analysis technique, they are able to separate the long term behavior of the connection from the short term behavior of the connection, and they show that correlation from the long term behavior (of sufficiently long flows) is still possible despite timing perturbation by the attacker. However, they do not present any tradeoffs between the magnitude of the timing perturbation, the desired correlation effectiveness, and the number of packets needed. Another important issue that is not addressed by [29] is the correlation false positive rate. While the coarse scale analysis for long term behavior may filter out packet jitter introduced by the attacker, it could also filter out the inherent uniqueness and details of the flow timing. Therefore coarse scale analysis tends to increase the correlation false positive rate while increasing the correlation true positive rate of timing perturbed connections. Nevertheless, Donoho et al ‘s work [29] represents a significant first step toward a better understanding of the inherent limitations of timing perturbation by the attacker on timing-based correlation. The important theoretical result is that correlation is still achievable for sufficiently long flows despite certain type of timing perturbations. What left open are the question whether correlation is achievable for arbitrarily distributed (rather than Pareto distribution conserving) random timing perturbation, and an analysis of the achievable tradeoff of the false positive and true positive rates.

2.6 Summary

The earliest work on the correlation of connections through stepping stones had focused on unencrypted connection and had been based on tracking users' login activities at different hosts [40,65,11]. Later work on correlation of unencrypted connections relied on comparing the packet contents of the connections to be correlated [69,82].

To address the challenges introduced by the encryption of packets (i.e. traffic encrypted using IPSEC[41] or SSH [54,87]), recent works [80,88,90] have focused on utilizing the packet timing characteristics to correlate encrypted connections. As a result, timing based correlation approaches are vulnerable to the active timing perturbation by adversary.

To address the new challenges introduced by the active timing perturbation of encrypted connection, Donoho et al [29] has used multi-scale analysis techniques to investigate the theoretical limits of active timing perturbation by attacker. They show that it is still possible to correlate the timing perturbed encrypted connections as long as the flow has enough packets.

We observe that previous approaches for tracing and correlating intrusion connections through stepping stones have substantial limitations and leave a number of fundamental questions open. In the remainder of this dissertation, we address the limitations of existing approaches under various settings and we describe our solutions for

1) Tracing unencrypted connections through stepping stones
2) Correlation of encrypted connections through stepping stones
3) Robust correlation of timing perturbed encrypted connections through stepping stones

Chapter 3

Sleepy Watermark Tracing

This chapter addresses the real-time correlation and tracing of unencrypted connections through stepping stones. Here we assume:

- The attackers have total control over the stepping stones and they can freely disguise, delete or forge the host login information at each stepping stone.
- The attackers use only unencrypted connections[6].

We describe the design and implementation of the *Sleepy Watermark Tracing* protocol, which significantly improves the capability and effectiveness over previous approaches for tracing unencrypted connections through stepping stones.

3.1 Introduction

In chapter 1, we have described that one way for the attackers to conceal their origin and identity is to disguise, delete or forge the host login information at stepping stones. Because the host based tracing approaches (i.e. DIDS[65], CIS[40]) rely on the host login information at each stepping stone, they could easily be defeated by disguise, deletion or forgery of host login information at one stepping stone.

To address the issue of host login information disguise, deletion and forgery in tracing and correlating unencrypted connections through stepping stones, network based approaches have been developed. The network based correlation and tracing is based on some invariant properties of the network connections across stepping stones. For example, the essence of application level content of the connection chain is invariant across the stepping stone, although the form of the application level content could be changed (i.e. by encryption). In particular, the packet content of unencrypted connections (such as telnet, rlogin) remains the

[6] We will address the case of encrypted connections in the following chapters

same across stepping stones. Therefore, the packet content can be used to correlate unencrypted connections through stepping stones.

Notably, thumbprinting [69] is the first network content based correlation approach proposed. It utilizes a small quantity of information called thumbprint to summarize a certain section of unencrypted connection. The thumbprint is defined, through principle component analysis technique in statistics, upon the frequencies that each character occurs within a period of time. Because the packet content of unencrypted connections remains invariant across stepping stones, the thumbprint defined over the packet content of a certain section of connection could be used to correlate unencrypted connection. If the frequencies of each character's occurrence in a certain section of the connection are unique enough, the thumbprint defined over that section of connection can be used to uniquely identify the whole connection and to distinguish that connection from unrelated connections.

While thumbprint is robust against local host login information disguise, deletion and forgery, and it could be useful even when only part of the Internet implements it, it has several drawbacks and limitations. First, thumbprinting requires the thumbprints calculated over the same time interval in order to be able to correlate correctly, which makes it dependent on the clock synchronization over the network. Second, because thumbprinting is defined over all packets transmitted during a certain time interval, it is vulnerable to local packet retransmission variation. Third, thumbprinting is unable to correlate when the connections are idle. Lastly, because the uniqueness of thumbprint depends on the uniqueness of packet content, the false positive rate of thumbprinting depends on the uniqueness of packet content. When different flows have similar packet content (i.e. flows connected to different hosts with same shell prompts), thumbprinting of these flows tends to have high false positive rate.

In this chapter, we address the limitations of thumbprinting in tracing intrusion connections through stepping stones. In particular, we strive to make network content based correlation

- Have as low as possible correlation false positive rate, even if the correlated flow has very similar packet content with uncorrelated flows.
- Have as high as possible correlation true positive rate with as few as possible packets.

- Be able to correlate and trace even when there is no traffic from the intruder.
- Be robust against packet retransmission variation.
- Be effective without clock synchronization.

By applying principles of active networking and steganography, we developed a novel intrusion response framework: Sleepy Watermark Tracing (SWT). SWT is “sleepy” in that it does not introduce any overhead when there is no intrusion detected. Yet it is “active” in that when there is intrusion detected, it will trigger and coordinate network-wide tracing at real-time. SWT exploits these observations: 1) interactive intrusions connections through stepping stones are bi-directional and symmetric at the granularity of connection; 2) application level content of unencrypted connections is invariant across stepping stones. By “injecting” carefully designed watermarks into the payload of backward-response traffic of the intrusion connection chain, SWT is able to trace the intrusion connections through stepping stones within a single keystroke by the intruder. Through its unique active tracing, SWT can trace through the connection chain even when the intrusion connections are idle. All these represent substantial improvements over existing capabilities for tracing unencrypted interactive connections through stepping stones.

In the following sections, we first discuss how SWT minimizes the correlation false positive rate. We then present the SWT architecture and protocols and show how SWT achieves single packet tracing and correlation. After presenting the experimental results, we summarize our findings.

3.2 Minimizing Correlation False Positive Rate

The false positive rate (FPR) of network content based correlation is inherently determined by the uniqueness of the network content. In general, the more unique the network content, the lower the correlation false positive rate; the less unique the network content, the higher the correlation false positive rate. In case the packet payload is not unique enough, network content based correlation needs more packets to achieve a low correlation false positive rate.

Our goal is to achieve as low as possible correlation false positive rate with as few as possible packets, even if the original network content is not unique enough.

3.2.1 Making the Network Content More Unique by Watermarking

Unlike thumbprinting, which passively correlates connections based on their original network content, SWT uses an active approach to make the network content more unique by watermarking the packet content.

Traditional watermark refers to the translucent mark or design that is pressed into fine paper during the paper making process, and the printed watermark is visible when the paper is held up to the light. *Digital watermark*, according to Webopedia [83], is a pattern of bits inserted into a digital image, audio or video file that identifies the file's copyright information. Unlike printed watermark, which is intended to be somewhat visible, digital watermark is designed to be invisible or inaudible. *Digital watermarking*, according to Cox, *et al* [24], is the practice of imperceptibly altering a work to embed a message about that work, where the work can be image, audio, video or any other media.

For the purpose of uniquely identifying unencrypted connections, SWT uses a randomly generated string as watermark. By injecting the unique watermark into the payload content of selected packets of unencrypted connection, the watermarked connection becomes more unique and thus becomes easier to be identified and correlated.

It is desirable to have as low as possible collision probability for the randomly generated watermark. For $n > 1$ sites, assume each site independently generates an equi-probable random integer number between 1 and m, where $m >> n$; let $P(m, n)$ be the probability such that those n random numbers are different from each other. Then we have:

$$P(m,n) = \prod_{i=1}^{n-1} \frac{m-i}{m} = \prod_{i=1}^{n-1} \left(1 - \frac{i}{m}\right)$$

When $m > n^2$, we have:

$$\prod_{i=1}^{n-1} \left(1 - \frac{i}{m}\right) > 1 - \frac{n^2}{2m}$$

Therefore, given $n = 2^{32}$, having $m \geq 2^{73}$ will make $P(m, n) > 0.999$. That means having 73 random bits in watermarks is sufficient to cover the whole IPv4 address space such that the

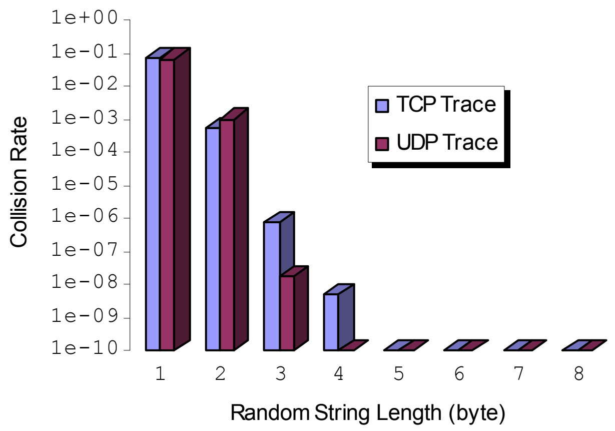


Figure 3.1 Random String Collision Rate with TCP/UDP Payload

probability of collision of generated watermarks by all possible hosts with distinct IP address is less than 0.1%.

Our research indicates that a randomly generated string of length of 4 or more bytes is distinct enough such that it would almost never be part of TCP or UDP payload of any particular IP packet. Figure 3.1 shows the average collision rate of 1000 randomly generated string of different lengths for two representative traces: a trace of 999,987 TCP packets and a trace of 999,986 UDP packets. Both traces were collected from an active 100 Mbps LAN at Computer Science Department of N.C. State University. A collision refers to the case such that the random string is a substring of the payload of an IP packet in the trace. Our experiments showed that each additional byte of the random string would make the collision rate at least 2 orders of magnitude lower. A 4-byte random string has average collision rate of 5×10^{-9} with TCP payload. A 3-byte random string has average collision rate of 1.7×10^{-8} with UDP payload.

Therefore, a few bytes of random string injected into the payload of one packet are able to make the entire unencrypted connections sufficiently unique for effective correlation. In order to be used for correlation, the embedded watermark must be able to traverse multiple stepping stones and remain invariant across stepping stones. This requires the watermark be injected into the application layer of connections; therefore, the watermark on packet content is application specific. For example, to watermark telnet connections, the watermark has to be injected into the telnet payload to make it invariant across stepping stones.

To embed the random watermark into the application payload, SWT requires support from the application server itself. In particular, *watermark-enabled* applications are those network server applications (such as telnetd, rlogind) that have been modified to be able to "inject" requested watermark string into their response traffic upon request.

3.2.2 Making the Embedded Watermark Invisible by Steganography

While embedding watermark could make unencrypted connections much more unique, it is desirable to make the embedded watermark invisible to normal end users of the network applications at the same time. By utilizing steganography techniques, SWT is able to make the embedded watermark invisible to normal users of network applications such as telnet, and rlogin.

Steganography is the art of concealed communication [83]. The term of steganography derives from Greek words *stegano*, which means "covered", and *graphia*, which means "writing". Unlike cryptography, whose goal is to make the message content a secret, steganography aims to conceal the very existence of the message to be communicated.

For text based network applications such as telnet and rlogin, hiding watermark is, in many ways, similar to hiding data in text [6], which is much more difficult than hiding data in pictures or sounds. The open space method is one of the major methods of hiding data in text files through manipulating white space. In particular, inserting spaces at the end of each line of text file will not be noticed by readers. But for network applications such as telnet and rlogin, simply inserting spaces will change the cursor position, and it is likely to be noticed by end users. Fortunately, the text being transferred to network applications is not necessarily the same as that being displayed. For example, the string

"**`See me`***`abc\b\b\b \b`*"

transferred to telnet or rlogin will be displayed as the string

"**`See me`**"

We define a *virtual null string* of a network application as a string that appears null to end users of the network application. A virtual null string consists of two parts, namely, the *random part* and the *covering part*. The random part consists of the random string that is used for identifying the watermarked flow. The covering part consists of string of special characters that is used for hiding the random part from being viewed by the normal telnet/rlogin end users. For example, "`abc\b\b\b \b`" is a virtual null string of telnet and rlogin whose random part is "`abc`" and covering part is "`\b\b\b \b`". For most text based network applications (such as telnet, rlogin), given any particular random string, we can find a covering part such that the concatenation of the random string and covering part is a virtual null string. Therefore by constructing virtual null strings, we can make watermarks invisible to end users of telnet or rlogin.

3.3 SWT Overview

In previous section, we have shown how to minimize the correlation false positive rate by watermarking the packet content. In this section, we give an overview of the SWT architecture and its protocol.

3.3.1 SWT Objective

SWT architecture and protocol have been designed to achieve the following objectives:

1. to provide true real-time, one packet tracing capability and support post-attack tracing.
2. to have the capability to trace even when the intrusion connection is idle.

In order to achieve above objectives, SWT has followed three primary design principles:

1. Use hidden information to watermark backward traffic of the intrusion connections, and use watermarks to guide network-wide correlation of connections.
2. Use active and yet sleepy tracing protocol to dynamically control when, where, what and how to trace network-wide.
3. Integrate both application layer and network layer into a collaborative tracing infrastructure.

Design principle 1 exploits the observation that interactive intrusion connections through stepping stones are bi-directional and symmetric at the granularity of connections. By injecting carefully selected watermarks into the payload of the backward traffic, the watermarked backward traffic could be uniquely identified and thus effectively correlated. Because the payload content of unencrypted connection is invariant across stepping stones, SWT is able to trace intrusion connections across all the stepping stones with a single watermarked packet. Unlike packet marking [31,56,60,67], SWT uses hidden information to watermark the packet; therefore it is more robust against mark spoofing attack [56] than probabilistic packet marking.

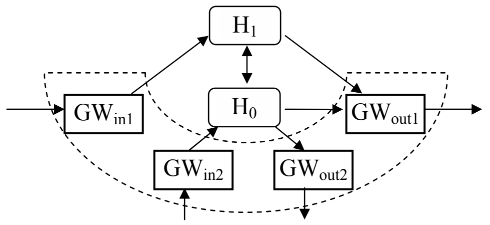
SWT tracing protocol is "sleepy" in that it does not introduce overhead when no intrusion is detected. Yet it is "active" in that when an intrusion is detected, the target will inject a watermark into the backward connection of the intrusion and “wakes up” intermediate routers along the intrusion path. After configurable period of time, the “awakened” intermediate routers that participate watermark tracing will “fall asleep” if no further wakeup message is received.

The integration of application layer and network layer enables SWT’s generic network layer tracing to be synchronized with application-specific watermark injection. This enables SWT to correlate only watermarked traffic at request.

3.3.2 Basic SWT Concepts

In order to keep track of network-based intrusions to hosts, it is desirable to monitor hosts through the nearest router or gateways. We term the monitoring router or gateway as *Guardian Gateway*. We define the *Incoming Guardian Gateway* of host *H* as the nearest router that forwards incoming traffic to *H* and the *Outgoing Guardian Gateway* of host *H* as the nearest router that forwards outgoing traffic from *H*. It is possible that one host has more than one incoming or outgoing guardian gateway. We define the union of incoming and outgoing guardian gateways of a host as its *Guardian Gateway Set* (e.g., $\{GW_{in1}$, GW_{in2}, GW_{out1}, $GW_{out2}\}$ in Figure 3.2). For a host *H*, while the traffic between *H* and its directly-connected neighbor hosts does not pass through any gateways, the traffic between *H* and any non-directly-connected hosts must pass through its guardian gateway set.

We further define a *leap* as one connection between hosts or stepping stones within a



H1
H0
GWin1
GWout1
GWin2
GWout2

Hi: Host
GWin: Incoming Guardian Gateway
GWout: Outgoing Guardian Gateway

Figure 3.2 Guardian Gateway Set

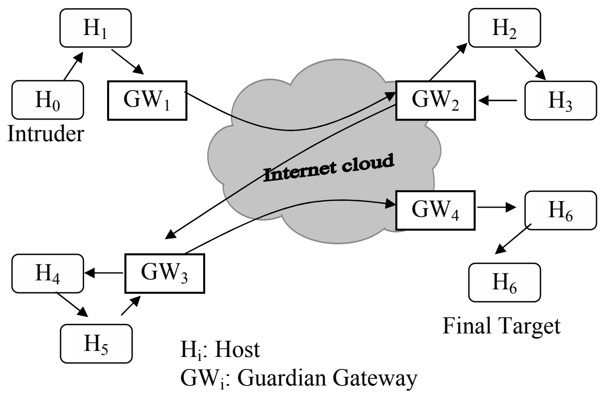


Figure 3.3 Intrusion Chain Tracing Model

connection chain (e.g., <H_i , H_{i+1}> in Figure 3.3). One leap may consist of multiple hops (or links in the physical network) and the two guardian gateways of the two end hosts. A leap can be specified by a 5-tuple consisting of

<protocol number, source ip address, source port number, destination ip address, destination port number>

Now the tracing problem of chained intrusion is defined as discovering and sequencing the guardian gateways of those hosts in the intrusion path, or (equivalently) as finding the leaps along the intrusion path.

3.3.3 Basic SWT Assumptions

We have identified the following assumptions that motivate and constrain our design:

- Intrusions are interactive and bidirectional,
- Routers are trustworthy and hosts are not trustworthy,
- Each host has a single SWT guardian gateway, and
- There is no link-to-link encryption.

The first two assumptions represent our assessments of the nature of the intrusions. Here we refer to intrusions as those attacks aiming to gain unauthorized access, rather than denial of service attacks. A study of CERT security incidents [35] indicates that almost all security

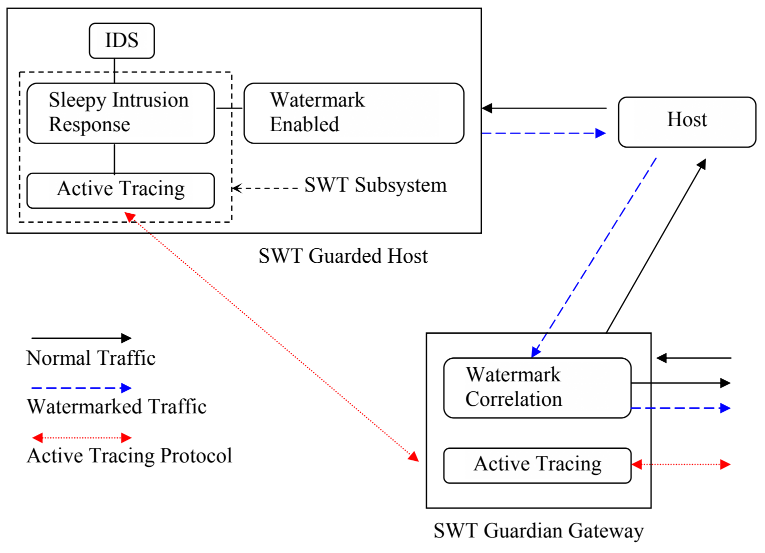


Figure 3.4 SWT Architecture

incidents, especially unauthorized access incidents, happened at computer hosts rather than routers or gateways. Therefore we believe our assumption to trust routers will cover most intrusion cases. In case there are indeed compromised routers involved in intrusion, the compromised router will be effectively indistinguishable from an attacker. The compromised router needs to be addressed first, before the tracing of the intrusion can go any further. In this case SWT can still trace to the farthest trustworthy guardian gateway.

The assumption of each host having a single SWT guardian gateway is only for simplifying the presentation of the SWT architecture. In case some host has multiple SWT guardian gateways, the guardian gateway set will be used in SWT tracing. In case not all stepping stones have SWT guardian gateways, SWT is able to trace to the furthest stepping stone that has SWT guardian gateway.

The final assumption represents the inherent limitation of any tracing based on network content. We will address the correlation of encrypted connections in subsequent chapters.

3.4 SWT Architecture and Protocol

As shown in Figure 3.4 the Sleepy Watermark Tracing Architecture consists of two complementing parties, namely, the *SWT guarded host* and the *SWT guardian gateway*. The

SWT guarded host is the host that supports SWT and thus is protected by SWT. The SWT guardian gateway is the router or gateway that supports SWT. In our trust model, each SWT guarded host has a unique SWT guardian gateway, and it maintains a pointer to its SWT guardian gateway. Each SWT guardian gateway may guard one or more SWT guarded hosts and it maintains the list of its SWT guarded hosts.

IDS and watermark-enabled applications at a SWT guarded host are SWT supporting components. In particular, IDS refers to the application level interface to any Intrusion Detection System, which is the ultimate initiator of SWT tracing. It interacts with SWT subsystem within SWT guarded host and triggers active watermark tracing once it detects an intrusion. Watermark enabled applications are those network service applications (such as telnetd, rlogind) that have been modified to inject arbitrary watermarks upon request.

The core of Sleepy Watermark Tracing Architecture consists of three interacting components: Sleepy Intrusion Response (SIR), Watermark Correlation (WMC) and Active Tracing (AT). In particular, Sleepy Intrusion Response accepts tracing requests from IDS, coordinates active tracing and keeps track of tracing information of intrusions. Watermark Correlation correlates incoming and outgoing connections based on provided watermarks. Active Tracing coordinates different parties in the network to collaboratively trace the incoming path and source of intrusions.

These three components work tightly together across SWT hosts and SWT guardian gateways. In specific, SIR and AT form the SWT subsystem within a SWT guarded host. Upon request from IDS, SIR coordinates appropriate WM-enabled application and the AT module to initiate active tracing from the SWT guarded host to SWT guardian gateways. At the SWT guardian gateway, the AT module receives tracing requests and asks the WMC module to correlate incoming and outgoing flows based on the watermark contained in the tracing request. The WMC module in turn provides AT module information about the next stepping stone after successfully correlating an outgoing flow with an incoming flow. Once the SWT guardian gateway finds the next stepping stone, AT will send trace information to the original host that initiated the whole tracing and notify the SWT guardian gateway of the

⋮			
timeout	watermark	C_0	trace information
⋮			

Figure 3.5 Active Intrusion Tracing Table

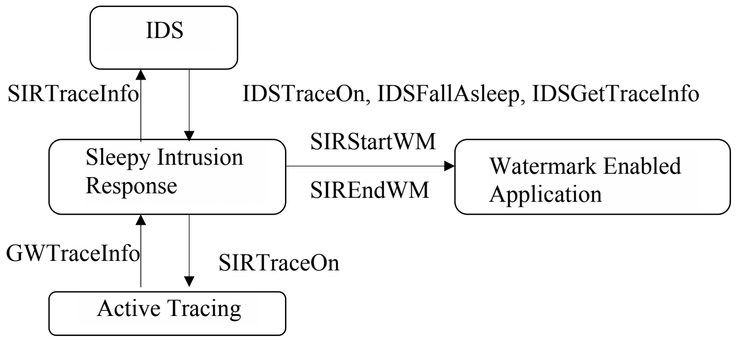

IDS
SIRTraceInfo
IDSTraceOn, IDSFallAsleep, IDSGetTraceInfo
Sleepy Intrusion Response
SIRStartWM
SIREndWM
Watermark Enabled Application
GWTraceInfo
SIRTraceOn
Active Tracing

Figure 3.6 SIR Processing

next stepping stone to start watermark tracing.

3.4.1 Sleepy Intrusion Response

SIR controls and coordinates overall SWT intrusion tracing. It is in a SWT guarded host and it interacts with IDS and WM-enabled applications in the same host. To achieve high efficiency, SIR introduces “sleepiness” into SWT. By default, the SWT system is inactive and in sleep mode. When IDS detects an intrusion, it triggers SWT tracing by notifying SIR with appropriate connection information. Upon request from IDS, SIR first registers the intrusion connection as active for a configurable period of time, if it is not active already. Then SIR triggers active tracing through SWT guardian gateways by sending out tracing request. Finally SIR notifies the WM-enabled application that terminates the intrusion connection to start injecting the requested watermark. SIR also keeps track of tracing information of intrusions returned by the SWT guardian gateway, and upon request from IDS, SIR will provide tracing information on any specific active intrusion. If there is no trace information returned from the SWT guardian gateways and no further tracing request from IDS for an active intrusion connection within a configurable period of time, the intrusion response for the intrusion connection will become inactive (“fall asleep”).

The core of SIR is managed through Active Intrusion Tracing Table (AITT) as shown in

Figure 3.5. Each entry in AITT stores information about an intrusion connection. In particular, the timeout field contains the remaining time for an intrusion connection to be active; the watermark field contains the watermark used for tracing an intrusion connection chain; C0 is the terminating intrusion connection detected by IDS; and tracing information contains the information about stepping stones discovered and reported by the "awakened" SWT Guardian Gateways.

The basic processing of SIR is shown in Figure 3.6. SIR interacts with IDS, watermark-enabled application, and AT through processing 4 messages:

- IDSTraceOn message from IDS, which requests SIR to start active tracing on an intrusion connection. SIR first checks if the to be traced connection is already in AITT. If so, it refreshes the timer on that connection; otherwise, SIR registers the to-be-traced connection and generates a watermark for it. Then SIR requests AT to start tracing with the generated watermark before asking the watermark-enabled application to start injecting a corresponding watermark. By doing so, it makes the watermark correlation module at SWT guardian gateway ready to correlate watermarked connections before the first watermarked packet is sent

- IDSFallAsleep message from IDS, which informs SIR that active tracing of a particular connection is no longer needed. SIR notifies the corresponding watermark-enabled application through the message SIREndWM to stop injecting a watermark.

- IDSGetTraceInfo from IDS, which asks SIR about trace information collected on a particular intrusion connection chain. SIR first checks if the intrusion connection is in AITT; if so, SIR sends the corresponding trace information back to IDS.

- GWTraceInfo from AT, which reports SIR with trace information on a particular intrusion connection chain. SIR first makes sure that the trace information is what it wanted by matching the watermark in the message with an existing watermark in AITT. If a match is found, SIR registers the trace information in AITT.

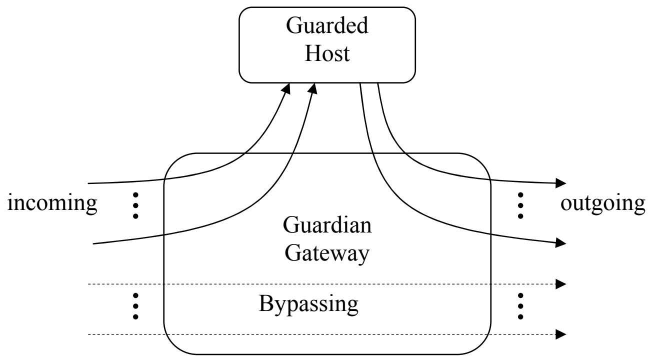


Figure 3.7 Guardian Gateway Correlation

As shown, the sleepiness of SIR is designed to be on the basis of per-intrusion connection. Active tracing is triggered only for detected active intrusion connections and no network-wide action will be taken on any non-active intrusion from SIR. While the IDSFallAsleep message from IDS will explicitly let SIR fall asleep on the corresponding intrusion connection, the timer will also put SIR to sleep for those intrusions that have been idle for a configurable period of time. Thus SIR introduces no tracing overhead if there is no active intrusion detected.

3.4.2 Watermark Correlation

In order to trace the intrusion connections through stepping stones, a mechanism is needed to correlate the incoming connections with outgoing connections of stepping stones. According to the SWT tracing model, the hosts along the intrusion connection chain are not trustworthy; therefore, SWT is designed to correlate at SWT guardian gateways.

The through traffic of a SWT guardian gateway can be divided into two classes: *guarded* and *bypassing* (Figure 3.7). We define *guarded traffic* of a SWT guardian gateway as that traffic that either terminates at or originates from one of the SWT guardian gateway's guarded hosts, and *bypassing traffic* as all other traffic. It is obvious that the SWT guardian gateway needs to scan only the guarded traffic for possible correlation.

One challenge of correlation at the SWT guardian gateway is that there may be multiple incoming or outgoing connections that terminate at or originate from one SWT guardian gateway concurrently. For a SWT guardian gateway with *m* incoming and *n* outgoing connections, there are $m \times n$ combinations of possible matches after those $m+n$ connections have been scanned. In specific, after *m* incoming connections have been scanned, each of the *n* outgoing connections scanned has *m* possible matches for correlation. Therefore exhaustive matching at multiple SWT guardian gateways would be computationally expensive.

SWT utilizes watermark-enabled application to watermark the backward traffic from the attack target to the attack source. The watermark-enabled application processes two messages from SIR: WM-Start and WM-End, where WM-Start requests watermark-enabled application to start injecting the requested watermark for specified times, and WM-End requests the watermark-enabled application to stop injecting the watermark.

With an identifying watermark injected to backward traffic of the intrusion connection chain, correlation at intermediate SWT guardian gateways is simplified to scanning incoming and outgoing connections and matching those with the same registered watermark. In particular, both incoming and outgoing connections are scanned for any registered watermark. If there is no registered watermark at SWT guardian gateway, no incoming and outgoing connections are scanned for correlation. In this sense, the watermark correlation in SWT is “guided”, compared to previous passive correlation.

The following observations can be made about watermark correlation:

- The accuracy of correlation is purely based on the uniqueness of the watermark. The low watermark collision probability ensures low false positive rate of correlation from even a single watermarked packet.
- While the watermark is application specific, watermark correlation is generic. The computation overhead for watermark correlation is linear to the number of stepping stones.

3.4.3 Active Tracing

The AT protocol coordinates SWT guardian gateways in the network to collaboratively trace the incoming path and source of detected intrusion. At each SWT guardian gateway, the AT

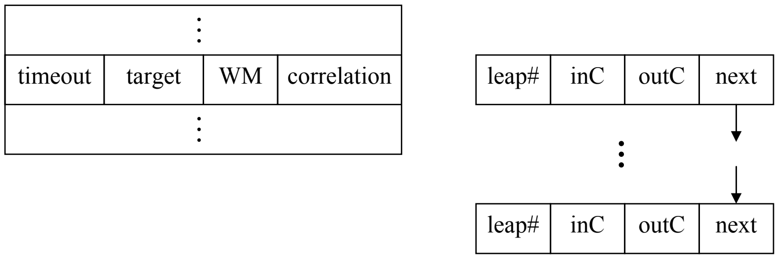


Figure 3.8 Gateway Correlation Table

module works closely with the watermark correlation module. By default, the SWT guardian gateway is in the sleep mode and there is no watermark to scan for. Upon request from a SWT guarded host or guardian gateway, AT will register the corresponding watermark and the incoming connection information as active for a configurable period of time. Once the watermark correlation module finds a match between an incoming connection and an outgoing connection, AT will wake up the next SWT guardian gateway along the outgoing connection and send the correlation information back to the original SWT guarded host. Therefore the overall trace information consists of connection correlation information from various SWT guardian gateways. To facilitate sequencing various correlation information from SWT guardian gateways at a SWT guarded host, AT introduces the *leap number* to the connections. When a SWT guarded host sends out tracing request to its guardian gateway, it initializes the leap number to 1. Each SWT guardian gateway that receives the tracing request will keep the leap number of the incoming connection. When sending the tracing request to the next SWT guardian gateway, it will set the outgoing leap number as the incoming leap number plus one. When sending correlation information back to the original SWT guarded host, the SWT guardian gateway includes the leap number of the incoming connection as the sequence number of correlation.

The core data structure of AT at a SWT guardian gateway is the Gateway Correlation Table (GWCT), as shown in Figure 3.8. Each entry in GWCT contains correlation information for a watermark. In particular, the timeout field contains the remaining time for the corresponding watermark to be active; the target field refers to the IP address of the original SWT guarded host that initiated active tracing; the WM field contains the watermark that AT will scan for; and the correlation field contains a pointer to the correlation information that has been found so far. Because an intrusion connection may pass through a SWT guardian gateway multiple

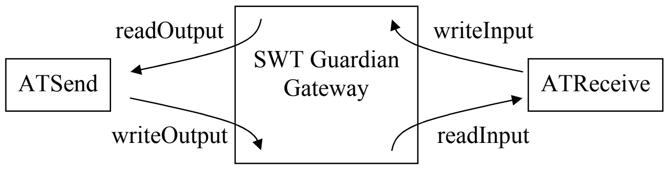


Figure 3.9 Active Tracing Architecture

times, the correlation information is organized as a linked list. Specifically, leap# contains the leap number of the incoming connection; inC contains the incoming leap and outC contains the outgoing connection information.

The structure of the AT protocol is shown in Figure 3.9. AT consists of two communicating halves, namely, the *ATSend* and *ATReceive*. Both halves work at the IP level at the SWT guardian gateway and intercept IP input and output packets. The intercept interface is abstracted into four functions: *readInput()*, *writeInput()*, *readOutput()* and *writeOutput()*.

The AT protocol introduces two SWT messages: *GWTraceOn* and *GWTraceInfo*. Both messages include: (1) the IP address of the original SWT guarded host that initiated the trace, denoted by *msg.target*; (2) the watermark to be scanned, denoted by *msg.WM*; (3) the leap number of the incoming connection, denoted by *msg.leap#*; (4) and the incoming connection information, denoted by *msg.inC*. GWTraceInfo message also contains corresponding outgoing connection information correlated with the incoming connection, denoted by *msg.outC*. Both SWT messages have a SWT header, denoted by msg.SWTHeader, and each message has its own SWT header type, denoted by SWTHeader.type. The two SWT messages are transported through UDP with specific port numbers.

ATSend() in Figure 3.10 shows the AT protocol processing at the IP output side. After intercepting an IP packet *p*, AT first checks if the packet originates from one of its guarded hosts and is not a SWT message; if so, AT scans the packet for registered watermark. That is AT tries to correlate only those packets of outgoing connections. If an outgoing connection is found having a registered watermark, AT first sends the tracing request to the SWT guardian gateway along the outgoing connection and then sends trace information back to the original host that initiated watermark tracing. After that, AT sends out the watermarked packet

```
Function ATReceive()
{ p=readInput();
  if (p.dest is not in my guarded host list or
      p is not SWT message)
    writeInput(p);
  else
  { if (p.SWTHeader.type==GWTraceOn)
    { (target,WM,leap#,inC)=getGWTraceOnMsg(p);
      if (there exist an item in GWCT such that
          item.target==target && item.WM==WM)
      { refresh item.timeout;
        if ((leap#,inC,*) does not in
            item->correlation)
         add(leap#,inC,NULL) to item->correlation;
      }
      else
      { add new item=(timeout,target,WM) into GWCT
        add (leap#,inC,NULL) to item->correlation;
      }
    }
  }
}

Function ATSend()
{ p=readOutput();
  if (p.src is in my guarded host list and
      p is not SWT message)
  { if (Match(p) is found with item in GWCT and
        (leap#,inC,outC) in item->correlation
    { construct GWTraceOn =
         (item.target,item.WM,leap#+1,outC);
      send GWTraceOn to outC.dest;
      construct GWTraceInfo =
         (item.target,item.WM,leap#,inC,outC);
      send GWTraceInfo to item.target
    }
  };
  writeOutput(p);
}
```

Figure 3.10 Active Tracing Protocol Operations

through writeOutput(*p*).

The following observations can be made about AT:

- AT Protocol does not require the SWT guardian gateway to have any knowledge of other SWT guardian gateways. Trace request is sent to the next stepping stone (or the intrusion source) and the SWT guardian gateway receives it by intercepting SWT messages that are destined to one of its guarded hosts.
- There is no explicit FallAsleep message in AT protocol; the SWT guardian gateway falls asleep only through time out on those active tracing items.
- Because AT sends out trace request to the next guardian gateway before sending a corresponding watermark packet to the next stepping stone, the next guardian gateway is

able to get ready before the watermarked packet arrives. This makes SWT to be able to trace back to the farthest trustworthy SWT guardian gateway to the origin of intrusion with only one watermarked packet.

- Because AT intercepts at the IP level and utilizes a watermark to correlate, AT is a generic tracing facility in that it can be used to trace connection chains of various IP based protocols.

3.5 SWT Analysis

By watermarking selected packets and processing them accordingly, SWT provides many potential advantages over existing intrusion tracing approaches. 1) SWT separates intrusion tracing from intrusion detection, and it does not require any node other than the intrusion target to have the intrusion detection capability. 2) Unlike thumbprinting, ON/OFF-based and deviation-based approaches, SWT does not need to record all the concurrent incoming and outgoing connections at any node, and it does not need to match each of the incoming connections with each of the outgoing connections for correlation at any node. 3) SWT requires no clock synchronization and is robust against retransmission variation. 4) SWT traces only when needed. 5) So far the most compelling advantage of SWT is its correlation accuracy and efficiency. By using watermarks, SWT can trace the intrusion connection chain to its origin within a single keystroke of the intruder. With its unique active tracing, SWT can trace the intrusion connection chain back to its origin even when the intruder is inactive. 6) We have found that SWT can be implemented efficiently. It does not introduce any noticeable overhead to routers, and it only requires a few network server applications at the intrusion target host to be modified to inject watermarks.

3.5.1 Robustness and Security

Because SWT puts its trust on SWT guardian gateway rather than hosts, it is robust against compromised hosts. One possible attack on SWT is sending false alarms to SWT guardian gateways. While this may introduce a bogus GWCT entry at SWT guardian gateway, it only affects the receiving SWT guardian gateway, as the further wakeup of other SWT guardian gateways happens only when a correlation is found. The impact of this kind of denial-of-service attack can also be limited as we utilize a least frequently used algorithm to replace those faked items of GWCT with valid tracing requests when they arrive. Some heuristics can also be developed to make SWT more robust against false alarms by differentiating those false alarm messages that have been received multiple times without any correlation found.

Another potential attack against SWT tracing is to detect and filter the embedded watermark from watermarked packets. When watermark uses virtual null string, it could be detected through the backspace characters in the covering part. To make the watermark detection and filtering more difficult, we could put the random part and covering part of the watermark into two consecutive packets so that the first packet only has the random part without backspace. To make the embedded watermark more robust against detection and filtering, we could even omit the covering part and make the watermark a pure random string. This will make the embedded watermark a pure random string and it is difficult to be distinguished from the packet's original payload. A potential side effect of this is that the intruder could be alarmed as the content displayed is different from the original one. This is essentially a tradeoff between the tracing robustness and stealthiness.

In considering security, SWT does not change the correct operation of routers. The SWT gateway only scans guarded traffic and generates tracing information upon request. While bringing the benefits of active networks, this limited programmability does not introduce new security concerns that some active networks may have.

3.5.2 Intrusiveness and Privacy

Unlike most other tracing mechanisms, SWT is intrusive in the sense that the watermark-enabled application actively injects a watermark into the backward traffic of the intrusion connection, which would slightly increase the size of some packets. For interactive applications such as telnet, rlogin, watermark can be made invisible to normal end users by careful selection. For applications such as ftp, rcp, injecting watermark will break the integrity of data that the intruder receives. Because watermark injection only happens when there is an intrusion detected, only the intruder's network application will receive watermarked response. Therefore only the intruder's intrusive network application could potentially be broken by watermarks. We believe this is a reasonable price to pay for the highly accurate, real-time, single packet tracing capability. By controlling the number of watermarks injected by watermark-enabled application, we can further keep the intrusiveness to a minimum and make SWT harder to detect by intruders and their confederates.

As with any other traceback systems, SWT is in direct conflict of interest with the use of anonymizer and its new tracing capability may introduce potential privacy concerns. However, SWT does not disclose any original content to any third party and its tracing is based on the watermark injected by the content sender. While it is already a common practice for the digital content providers to inject cookies or watermarks into the original digital content for profile tracking or copyright protection, it is relatively new to use watermarks to identify and trace computer communications.

3.6 Experiments

As a proof of concept, we have implemented a SWT prototype on FreeBSD 4.3. The prototype includes a SWT guarded host, SWT guardian gateways and a watermark-enabled application all running on the FreeBSD platforms.

For efficiency reasons, SIR and AT at SWT guarded host are combined together into one daemon process. IDS is abstracted into a user process which interacts with SIR through IPC. Strictly speaking, the watermark-enabled application is not part of SWT, but is a supporting component. We have modified telnetd on FreeBSD to support arbitrary watermark injecting. We have used a 72- bit watermark that is invisible to telnet users.

The SWT guardian gateway implementation utilizes *ipfw* and *divert socket* mechanisms from FreeBSD so that all the SWT gateway processing is at the user level. Watermark correlation and AT are implemented into a process that intercepts IP packets through divert socket. We have used UDP port 1999 at SWT guarded host and UDP port 2000 at SWT guardian gateways.

We have performed two functional experiments on tracing a telnet connection chain: A $\Rightarrow$ B

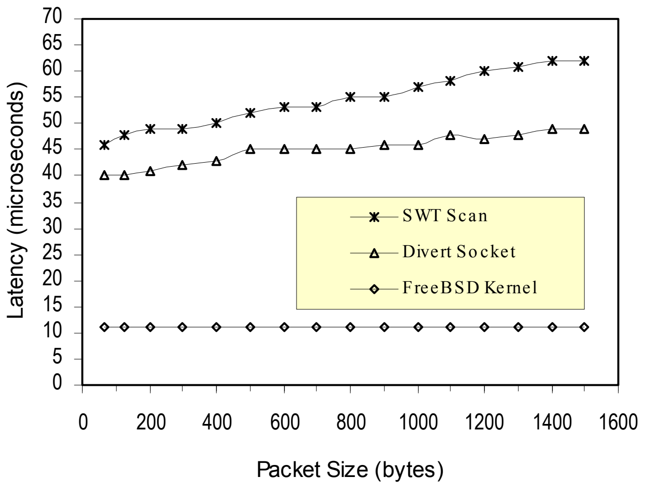


Figure 3.11 Latency of SWT Guardian Gateway

⇒ C ⇒ D, where A is the source of intrusion and D is the final intrusion target. The first is to trace the intrusion source while the intruder is active. Our SWT prototype demonstrates the capability of real-time tracing of a single watermarked packet: SIR at host D gets all the trace information pointing to intrusion source A within one keystroke from intruder at A. The second experiment is to trace the intrusion source while the intruder is inactive or silent. By actively sending back a watermark from watermark-enabled telnetd, our SWT prototype also gets all the trace information pointing to the intrusion source A. As we have expected, for each watermarked packet, SWT triggers one GWTraceOn message travel from D ⇒ C ⇒ B ⇒ A, and two GWTraceInfo messages from C and B respectively.

To quantify the overheads incurred due to SWT itself, we have measured latency and throughput of SWT gateways with four different configurations:

(1) FreeBSD kernel IP forwarding without SWT;
(2) divert socket IP forwarding without SWT;
(3) SWT configured to scan traffic.

The latency and throughput measurements were performed on a three node testbed configured in a straight line topology. The gateway at the intermediate node was a Pentium III 750MHz PC with 512MB RAM, 512KB cache, and two Intel EtherExpress Pro 100 fast

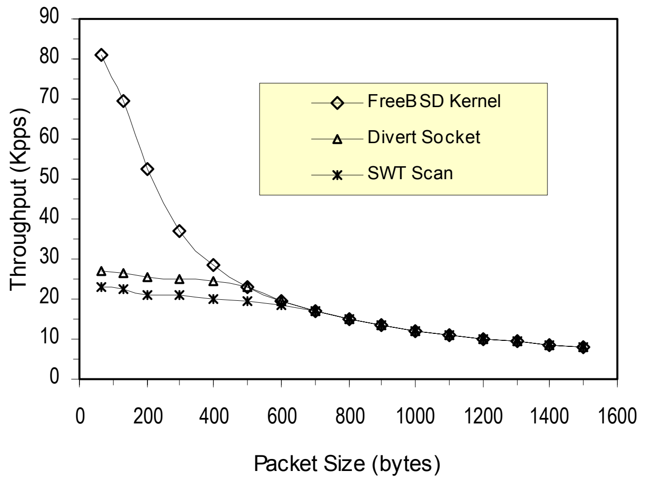


Figure 3.12 Throughput of SWT Guardian Gateway (Kpps)

Ethernet adapters, running FreeBSD 4.3.

Figure 3.11 shows that the latency of FreeBSD kernel IP forwarding is about 11 μs, independent of packet sizes. The latency of divert socket IP forwarding ranges from 40 μs to 49 μs depending on the size of IP packets. The 29 μs to 38 μs overhead for divert socket forwarding over kernel forwarding includes: (1) overhead for two context switches for data reading and writing; (2) overhead for data copy in and out of user space; (3) overhead for dispatching system calls. Compared with divert socket IP forwarding, SWT scanning takes 6 to 13 μs more time to forward IP packets of various sizes. This indicates that the SWT gateway latency overhead due to SWT itself is about 6 to 13 μs.

Figure 3.12 presents throughput in term of packets per second. As shown in the figure, the throughput gap is bigger between FreeBSD kernel and divert socket or SWT scan for smaller packets. In particular, for 64-byte packets, FreeBSD kernel can forward 81,100 packets per second, while divert socket and SWT scan can forward 27,200 and 22,200 packets per second respectively. This suggests that the overheads for context switches and dispatching system calls become dominant factor of overall throughout limit for small packets. Throughputs of FreeBSD kernel and divert socket converge after packet size reaches 500 bytes. At 700-byte packet size, SWT scan reaches the same throughput of FreeBSD kernel.

Figure 3.13 depicts throughput in terms of bytes per second. It shows that SWT scan is able

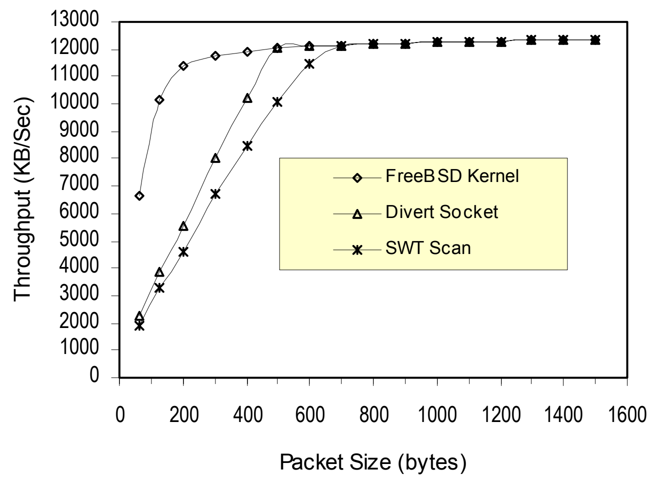


Figure 3.13 Throughput of SWT Guardian Gateway (KB/Sec)

to saturate the fast Ethernet with packets of size of 700 bytes.

3.7 Summary

Tracing intrusion connections through stepping-stones at real-time is a challenging problem, and tracing an idle intrusion connection chain has been viewed as impossible due to the lack of traffic from the intruder. We have presented SWT as an active network-based intrusion response framework to address the problem of real-time tracing unencrypted, interactive intrusion connections through stepping-stones. We believe SWT's key contribution is to demonstrate: 1) watermark and information hiding techniques could be used to build highly effective and efficient IP traceback system; 2) active approach can make the traceback system more effective and efficient; 3) single packet correlation of unencrypted connections is feasible; 4) tracing idle unencrypted connections is feasible.

The integration of watermarking technique and active approach makes SWT's tracing "guided" compared with previous tracing approaches for unencrypted connections. SWT is a complete, practical system and it only requires some of the edge routers to participate tracing. Our prototype shows that SWT is able to trace back to the trustworthy SWT guardian gateway that is closest to the source of intrusion chain, within single keystroke of the intruder. By actively injecting watermark back to the intrusion connection, it is able to trace even when the intruder is silent. Our experience also shows that SWT can be implemented efficiently. Our prototype shows that the SWT’s own impact on a gateway’s processing delay is only about 6~13 μs, and SWT gateway on Pentium III 750MHz PC is able to saturate the fast Ethernet with 700-byte packets.

Chapter 4

IPD-Based Correlation of Encrypted Connections

In this chapter, we address the new challenge, introduced by the encryption of connections by attackers, in correlating and tracing of connections through stepping stones. Here we assume

- The attackers have total control over the stepping stones and they can freely disguise, delete or forge the host login information at each stepping stone.
- The attackers could encrypt any and all connections with any encryption scheme and different keys.
- There is no active timing perturbation from attackers.
- The original packet order is kept.
- There is no bogus packets added or packets dropped.

Our goal is to investigate how encrypted as well as unencrypted interactive connections could be effectively correlated. We propose a novel correlation scheme based on inter-packet timing characteristics of both encrypted and unencrypted connections. Our experimental results demonstrate that both encrypted and unencrypted interactive connections could be effectively correlated based on inter-packet timing characteristics.

4.1 Introduction

Network-based correlation approaches are robust against disguise, deletion and forgery of host login information at stepping stones. In particular, network content based approaches (i.e. Thumbprinting, SWT) correlate flows based on their payload content, which require that the payload content remains invariant across routers and stepping stones. With the widespread deployment and availability of IP connection encryption utilities (i.e. SSH, IPSEC), the attackers can easily defeat network content based correlation by encrypting some or all connections across the stepping stones. This new level of countermeasure against correlation calls for new approaches in correlating connections through stepping stones.

In principle, the correlation of connections is based on inherent characteristics of connections. To correlate potentially encrypted connections, the key is to identify a correlation metric from the connection characteristics that is: 1) invariant across routers and stepping stones; 2) not affected by encryption and decryption; 3) unique to each connection chain across stepping stones; 4) common for all the connections that are in the same connections chain. Potential candidates for the correlation metric of a flow of packets include header information, packet size, inter-packet timing etc.

In this chapter, we investigate the correlation of both encrypted and unencrypted connections based on their inter-packet timing characteristics and we present an original correlation method based on inter-packet delays or IPDs. In particular, we seek answers to the following specific questions:

1. How do inter-packet delays (IPD) change or persist across routers, stepping stones?
2. Whether and how do the packet encryption and decryption affect the inter-packet delays?
3. How distinctive are inter-packet delays in identifying various flows?
4. How well do inter-packet delays differentiate unrelated flows?

To answer these questions, we design and evaluate four correlation metrics defined over inter-packet delays. Our experimental results show that (after some filtering) inter-packet delays (IPDs) of both encrypted and unencrypted, interactive connections are preserved across many router hops and stepping stones. Both encrypted and unencrypted interactive connections can be effectively correlated based on IPDs. The effectiveness of this method for correlation purposes also requires that the timing characteristics be distinctive enough to identify connections. We have found that normal interactive connections such as telnet, SSH and rlogin are almost always distinctive enough to provide correct correlation across stepping stones. The number of packets needed to correctly correlate two connections is also an important metric, and is shown to be quite modest for this method.

4.2 IPD Based Correlation Model

The overall IPD correlation of two connections is a two-step process. First, the two connections to be correlated are processed to generate a number of *correlation points* between the two connections. Second, these generated correlation points are evaluated to obtain the *overall correlation value* of the two connections. The rationale behind this two-step process is to support the true real-time correlation, which is the capability to correlate "live" traffic when they come and go. This means that the approach must be able to correlate connections before their ends are reached. Therefore, the correlation metric for true real-time correlation cannot be defined over the entire duration of a connection; we choose instead to compute it over a window of packets in the connection. A correlation point generated from IPDs within the window reflects some local similarity between the two flows; the overall correlation value obtained from all the correlation points will indicate the overall similarity of the two flows.

4.2.1 Basic IPD Correlation Concepts and Definitions

Given a bi-directional connection, we can split it into two unidirectional flows. We define our correlation metric over the unidirectional flow of connections. Given a unidirectional flow of $n > 1$ packets, we use t_i to represent the timestamp of the ith packet observed at some point of the network. We assume all the t_i's of a flow are measured at the same observation point with the same clock. We define the ith *adjacent inter-packet delay (IPD)* as

$$d_i = t_{i+1} - t_i \tag{4.1}$$

Therefore, for any flow consisting of $n > 1$ packets, we can have the adjacent IPD vector $<d_1, \ldots, d_{n-1}>$.

Ideally, the adjacent IPD vector would uniquely identify each flow and we could construct our correlation metric from the adjacent IPD vectors. To support real-time correlation based on the adjacent IPD vector, we define the *IPD correlation window* $W_{j,s}$ on $<d_1, \ldots, d_n>$ as

$$w_{j,s}(< d_1,\ldots,d_n >) =< d_j,\ldots,d_{j+s-1} > \tag{4.2}$$

where $1 \leq j \leq n\text{-}s+1$ represents the starting point of the window, and $1 \leq s \leq n\text{-}j+1$ is the size of the window.

Given any two flows X and Y, whose adjacent IPD vectors are $<x_1, \ldots x_m>$ and $<y_1, \ldots y_n>$ respectively, we define a *Correlation Point Function* CPF over IPD correlation windows of

Flow X: $x_1, \ldots, \boxed{x_j, \ldots, x_{j+s-1},} \ldots x_m$

Flow Y: $y_1, \ldots, \boxed{y_{j+k}, \ldots, y_{j+k+s-1}} \ldots y_n$

Figure 4.1 CPF over IPD Correlation Windows $W_{j,s}$(X) and $W_{j+k,}$

X: $W_{j,s}$ (X) and of Y: $W_{j+k,s}$ (Y) as

$$CPF(X,Y,j,k,s) = \phi(W_{j,s}(X), W_{j+k,s}(Y)) \tag{4.3}$$

where ϕ is a function of two vectors: $R^s \times R^s \rightarrow [0, 1]$, $1 \le j \le \min(m\text{-}s\text{+}1, n\text{-}k\text{-}s\text{+}1)$ is the start of the IPD correlation window, $-j+1 \le k \le n\text{-}j\text{-}s+1$ is the offset between the two IPD correlation windows, and $1 \le s \le \min(m\text{-}j\text{+}1, n\text{-}j\text{-}k\text{+}1)$ is the size of the two IPD correlation windows. As shown in Figure 4.1, *CPF*(*X*, *Y*, *j*, *k*, *s*) quantitatively expresses the correlation between $W_{j,s}(<x_1, \ldots x_m>)$ and $W_{j+k,s}(<y_1, \ldots y_n>)$. The higher value of *CPF*(*X*, *Y*, *j*, *k*, *s*), the better correlation between $W_{j,s}(<x_1, \ldots x_m>)$ and $W_{j+k,s}(<y_1, \ldots y_n>)$.

Because the value of CPF(X, Y, *j*, *k*, *s*) changes as *j* and *k* change, we can think of *CPF*(*X*, *Y*, *j*, *k*, *s*) as a function of *j* and *k*. Given any particular value of *j*, *CPF*(*X*, *Y*, *j*, *k*, *s*) may have a different value for each different value of *k*. For any particular value of *j*, we are interested in the maximum value of *CPF*(*X*, *Y*, *j*, *k*, *s*), which represents the best correlation between $W_{j,s}(<x_1, \ldots x_m>)$ and $W_{j+k,s}(<y_1, \ldots y_n>)$ for all possible values of offset *k*.

For different values of *j*, the maximum value of *CPF*(*X*, *Y*, *j*, *k*, *s*) could be any where from 0 to 1. We are interested in those maximum values of (*X*, *Y*, *j*, *k*, *s*) that are no less than a certain threshold.

We define (*j*, *j*+*k*) as a *correlation point* if

$$\max_{-j+1 \le k \le n-j-s+1} CPF(X,Y,j,k,s) \ge \delta_{cp} \tag{4.4}$$

where δ_{cp} is the *correlation point threshold* with value between 0 and 1. The δ_{cp} here is for detecting correlation point and is different from δ in inequality (2.6). A correlation point (*j*, *j*+*k*) represents the local correlation between one flow starting from packet number *j* and another flow starting from packet number *j*+*k*.

We further define *k* for this correlation point (*j*, *j*+*k*) as the *correlation-offset* of *CPF*(*X*, *Y*, *j*, *k*, *s*) and the correlation point.

Given flow X, Y, correlation window size *s* and threshold δ_{cp}, by applying formula (4.4), we can obtain a series of correlation points: (j_1, j_1+k_1), (j_2, j_2+k_2), …, (j_n, j_n+k_n), where $n \geq 0$. Here the value of correlation offset k_i may be different. Assuming one packet of flow X corresponds to one packet of flow Y[7], if flow X and Y are really part of the same connection chain, the IPDs of flow X should have a one-to-one correspondence with the IPDs of flow Y. In this case, all the correlation points should have same correlation offset *k'*. This can be formally represented with CPF as

$$\exists k' \forall j_i [CPF(X,Y,j_i,k',s) = \max_{-j_i+1 \leq k_i \leq n-j_i-s+1} CPF(X,Y,j_i,k_i,s)] \tag{4.5}$$

That is there exists an offset *k'* such that for all possible *j*, *CPF*(*X*, *Y*, *j*, *k'*, *s*) is the maximum for all possible *k* (after is *j* determined). Or equivalently, all $k_i = k'$. In this case, all the correlation points (j_i, j_i+k_i) will fall on a line defined by linear function $y=x+k'$.

After obtaining $n>0$ correlation points: (j_1, j_1+k_1), (j_2, j_2+k_2), …, (j_n, j_n+k_n), we represent those *n* correlation points with two *n*-dimensional vectors $C_x = < j_1, \ldots j_n >$ and $C_y = < j_1+k_1, \ldots j_n+k_n >$. The Correlation Value Function $CVF(C_x, C_y)$ gives the overall correlation value between the two flows based on all the correlation points obtained through formula (4.4). We use the *overall correlation threshold* δ to determine if two flows are correlated. If $CVF(C_x, C_y) \geq \delta$, we declare flow X and Y are correlated.

4.2.2 Heuristics in Finding Correlation Points

Given flow X of *m* packets and flow Y of *n* packets, we could find one or more correlation points (j_i, j_i+k_i) through formula (4.4). Because (j_i, j_i+k_i) is a 2-dimensional variable, a brute-force method to find all correlation points would have $O(m \times n)$ computation complexity.

[7] We have found this is true for most packets in correlated flows.

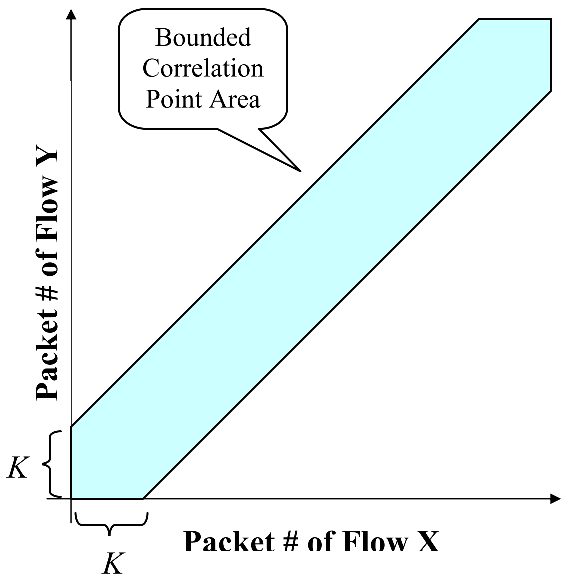


Figure 4.2 Bounded Area of Correlation Points

One heuristics to reduce the computation complexity of finding correlation points is to set an upper bound on correlation-offset k. The idea is that if two flows are really correlated, the corresponding IPD should start at approximately the same time, where the time difference is determined by the network transmission delay (such as processing delay, queuing delay etc. at intermediate routers and hosts). Given that the flows are interactive, the overall network transmission delay could be bounded. Given that the network is loosely synchronized in time, the clock skew could also be bounded. Therefore, if flow X=<$P_{x,1}$,…, $P_{x,m}$> and flow Y=<$P_{y,1}$,…, $P_{y,n}$> are really correlated and packet $P_{x,i}$ corresponds $P_{y,j}$, we have

$$| t_{x,i} - t_{y,j} | < \Delta_t \tag{4.6}$$

where $t_{x,i}$ and $t_{x,i}$ are the timestamp of packets $P_{x,i}$ and $P_{y,j}$ respectively, Δ_t>0 is a time window determined by network delay and clock skew.

Let K denotes the upper bound of correlation-offset k, given Δ_t, we can obtain the value of K with constraints $| t_{x,i} - t_{x,i+K} | < \Delta_t$ and $| t_{x,i} - t_{x,i-K} | < \Delta_t$ within O(K) computations. Having k bounded, the correlation point (j, j+k) is also bounded as shown in Figure 4.2.

4.2.3 Correlation Point Function Assessment Criteria

A critical issue in this method is the choice of the function ϕ for computing the correlation point function CPF. Criteria by which the CFP may be evaluated include:

- Uniqueness of perfect correlation: for any flow *X* and *Y*, only when $W_{j,s}(X)=W_{j+k,s}(Y)$ would have $CPF(X, Y, j, k, s) =1$. This is an expression of the tightness of the correlation point function.
- Correlation Point (CP) *true positive (hit)*: if a correlation point (*j*, *j*+*k*) is detected between two correlated flows by equation (4.4) and packet number *j* in one flow really corresponds to packet number *j*+*k* in the other flow, then the correlation point (*j*, *j*+*k*) is a true positive (or hit). The true positive rate is the number of hits divided by the number of corresponding packet pairs between two correlated flows minus correlation window size *s*[8].
- Correlation Point (CP) *false positive (miss)*: if a correlation point (*j*, *j*+*k*) is detected between two flows by equation (4.4) and packet number *j* in one flow does not correspond to packet number *j*+*k* in the other flow, then (*j*, *j*+*k*) is a CP false positive (or miss).

Ideally, we would expect a perfect correlation point function would 1) have unique perfect correlation; 2) have a 100% CP true positive rate; and 3) have 0 CP misses or false positives.

4.3 Correlation Point Function

We now propose four correlation point functions, each of which enjoys certain advantages or applicability, as discussed below.

4.3.1 Mini/Max Sum Ratio (MMS)

One simple metric to quantitatively express the "similarity" between two vectors is the ratio between the summation of the minimum elements and the summation of the maximum elements.

$$CPF(X,Y,j,k,s)_{MMS} = \frac{\sum_{i=j}^{j+s+1} \min(x_i, y_{i+k})}{\sum_{i=j}^{j+s+1} \max(x_i, y_{i+k})} \tag{4.7}$$

[8] This is because starting from the last *s* packets in a flow, there are not enough (*s*+1) subsequent packets needed for correlation window of size *s*, and we should only count all possible correlated packet pairs that can be the start point of correlation window of size *s*.

The range of CPF(X, Y, j, k, s)$_{\text{MMS}}$ is [0, 1]. Only when $x_i=y_{i+k}$ for $i=j, \ldots, j+k-1$, will CPF(X, Y, j, k, s) $_{\text{MMS}}$ have the value 1. Therefore, CPF(X, Y, j, k, s) $_{\text{MMS}}$ has unique perfect correlation.

4.3.2 Statistical Correlation (STAT)

Based on the concept of the coefficient of correlation from statistics [28], we can define

$$CPF\ (X, Y, j, k, s)_{Stat} = \begin{cases} \rho(X, Y, j, k, s) & , \rho(X, Y, j, k, s) \geq 0 \\ 0 & , \rho(X, Y, j, k, s) < 0 \end{cases} \tag{4.8}$$

where $\rho(X,Y,j,k,s) = \dfrac{\sum_{i=j}^{j+s+1}(x_i - E(X)) \times (y_{i+k} - E(Y))}{\sqrt{\left[\sum_{i=j}^{j+s+1}(x_i - E(X))^2\right] \times \left[\sum_{i=j}^{j+s+1}(y_{i+k} - E(Y))^2\right]}}$

The range of CPF(X, Y, j, k, s)$_{\text{Stat}}$ is also [0, 1]. Unlike CPF(X, Y, j, k, s)$_{\text{MMS}}$, for a given $W_{j,s}$(X), there are more than one value of $W_{j+k,s}$(Y) for which CPF(X, Y, j, k, s)$_{\text{Stat}}$ has the value 1. For example, for a particular $W_{j,s}$(X), any linear transform of $W_{j,s}$(X): $W_{j+k,s}$(Y) w=$a\times W_{j,s}$(X) $+b$ will result in CPF(X, Y, j, k, s)$_{\text{Stat}}$ being equal to 1 (a>0). Therefore CPF(X, Y, j, k, s)$_{\text{Stat}}$ does not have unique perfect correlation, and is more likely to result in false positives.

4.3.3 Normalized Dot Product 1 (NDP1)

In digital signal processing, linear correlation (or matched filtering) of two discrete signals will reach a maximum at the point where the signals have the most similarity. It is well known that linear correlation is optimal in detecting the similarity between a discrete signal and the corresponding signal distorted by additive, white Gaussian noise. However the range of linear correlation is not necessarily between 0 and 1.

If the discrete signals are replaced by two vectors, the corresponding operation to linear correlation of signals is the *inner-product* or *dot-product* of two vectors in n-dimensional space. From linear algebra, the inner-product (or dot-product) of two n-dimensional vectors is equal to the cosine of the angle between the two vectors, multiplied by the lengths of the two vectors. That is:

$$W(X) \bullet W(Y) = \cos(\theta) \times |W(X)| \times |W(Y)| \tag{4.9}$$

or

$$\cos(\theta) = \frac{W(X) \bullet W(Y)}{|W(X)| \times |W(Y)|} \tag{4.10}$$

where θ is the angle between vector W(X) and W(Y), and |W(X)| and |W(Y)| are the lengths of vector W(X) and W(Y) respectively.[9]

cos(θ) in (4.10) can be used as a correlation point function. The range of cos(θ) is [-1, 1] and it provides a measure of the similarity of two vectors. Given any vector $W_{j,s}(X)$, cos(θ) will be 1 for any vector $W_{j+k,s}(Y) = a \times W_{j,s}(X) + b$.

To make the correlation point function to exhibit unique perfect correlation, we can define it as follows:

$$\begin{aligned}
&CPF(X,Y,j,k,s)_{NDP1} \\
&= \frac{\min(|W(X)|,|W(Y)|)}{\max(|W(X)|,|W(Y)|)} \times \cos(\theta) \\
&= \frac{\min(|W(X)|,|W(Y)|)}{\max(|W(X)|,|W(Y)|)} \times \frac{\sum_{i=j}^{j+s-1} x_i \times y_{i+k}}{|W(X)| \times |W(Y)|} \\
&= \frac{\sum_{i=j}^{j+s-1} x_i \times y_{i+k}}{[\max(|W(X)|,|W(Y)|)]^2} \\
&= \frac{\sum_{i=j}^{j+s-1} x_i \times y_{i+k}}{\max(\sum_{i=j}^{j+s-1} x_i^{\,2}, \sum_{i=j}^{j+s-1} y_{i+k}^{\,2})}
\end{aligned} \tag{4.11}$$

Because x_i and y_i are non negative, the range of CPF(X, Y, j, k, s)$_{\mathrm{NDP1}}$ is [0, 1]. It is obvious to see that when $W_{j,s}(X) \neq W_{j+k,s}(Y)$, $\min(|W_{j,s}(X)|, |W_{j+k,s}(Y)|) < \max(|W_{j,s}(X)|, |W_{j+k,s}(Y)|)$ and CPF(X, Y, j, k, s)$_{\mathrm{NDP1}}$<1. That is, CPF(X, Y, j, k, s)$_{\mathrm{NDP1}}$ will be 1 only when $W_{j,s}(X) = W_{j+k,s}(Y)$. Therefore, CPF($X$, Y, j, k, s)$_{\mathrm{NDP1}}$ has unique perfect correlation.

4.3.4 Normalized Dot Product 2 (NDP2)

Another way to normalize the dot-product of two vectors is

[9] We have dropped the subscripts of W(X) and W(Y) for clarity purposes in this section.

$$CPF(X,Y,j,k,s)_{NDP2} = \frac{\sum_{i=j}^{j+s-1} x_i \times y_{i+k}}{\sum_{i=j}^{j+s-1} \left[\max(x_i, y_{i+k})\right]^2} \tag{4.12}$$

Because x_i and y_i are non negative, the range of CPF(X, Y, j, k, s)$_{\text{NDP2}}$ is [0, 1]. It is obvious that CPF(X, Y, j, k, s)$_{\text{NDP2}}$ equals to 1 only when $W_{j,s}(X) = W_{j+k,s}(Y)$.

Among these four proposed correlation point functions, Mini/MaxSum Ratio (MMS) is likely to be the most sensitive to local details of the IPD vectors to be correlated. This is because it does not average any differences, and it accumulates all the IPD differences. As a result, MMS may potentially have a lower CP true positive rate due to its emphasis on local details. While the STAT CPF is much more robust to noise, we expect it to have substantially more CP false positives. The normalized dot product functions (NDP1 and NDP2) are likely to be in between MMS and STAT in terms of sensitivity to local detail and robustness to noise.

4.4 Correlation Value Function

Given flows X, Y, correlation window size s and threshold δ, by applying formula (4.4), we can potentially obtain a set of correlation points: (j_1, j_1+k_1), (j_2, j_2+k_2), …, (j_n, j_n+k_n). We represent this sequence of correlation points through two n-dimensional vectors $C_x=<j_1, \ldots j_n>$ and $C_y=<j_1+k_1, \ldots j_n+k_n>$.

We define the overall *Correlation Value Function* CVF of flows X and Y from this sequence of correlation points, as follows:

$$CVF(C_x, C_y) = \begin{cases} 0 & n = 0 \\ 0 & n > 1 \wedge \rho(C_x, C_y) < 0 \\ \rho(C_x, C_y) & n > 1 \\ 1 & n = 1 \end{cases} \tag{4.13}$$

where $\rho(C_x, C_y) = \dfrac{\sum_{i=1}^{n} (j_i - E(C_x)) \times (j_i + k_i - E(C_y))}{\sqrt{\left[\sum_{i=1}^{n} (j_i - E(C_x))^2\right] \times \left[\sum_{i=1}^{n} (j_i + k_i - E(C_y))^2\right]}}$

CVF(C_x,C_y) has value range of [0, 1], and it quantitatively expresses the overall correlation between flows X and Y based on the correlation points detected through formula (4.4). When flow X and Y have no correlation point detected, $CVF(C_x,C_y)$ is defined to be 0. When flow X and Y has only one correlation point detected, $CVF(C_x,C_y)=1$. When there are more than one correlation point detected and all the correlation points have same correlation offset (i.e., $k_1=k_2=\ldots k_n$), $CVF(C_x,C_y) = 1$. When there are more than one correlation point detected and not all the correlation points have same correlation offset (i.e., $k_i \neq k_j$), $CVF(C_x,C_y) < 1$.

In summary, $CVF(C_x,C_y)$ not only considers the number of correlation points detected (which corresponds to CP true positives), but also evaluates how well those detected correlation points are linearly aligned (which corresponds to CP false positives). $CVF(C_x,C_y)$ considers the two flows are better correlated with fewer, but better linearly aligned (which means fewer *CP* false positives), detected correlation points than more, but poorly linearly aligned (which means more *CP* false positives), detected correlation points.

4.5 Experiment

The goal of the experiments is to answer the following questions about IPD based correlation:

1 Are inter-packet delays preserved through routers and stepping stone, and to what extent?

2 Are inter-packet delays preserved across encryption/decryption and various network applications (such as telnet/rlogin/SSH)?

3 How effective is the IPD-based correlation metric in determining whether two connections belong to the same chain?

4 How well does the IPD-based correlation metric differentiate a connection from connections that are not part of the same chain?

4.5.1 Correlation Point Experiment

To answer the first two questions, we have conducted the following experiment. We first telnet'ed from a Windows 2000 PC behind a cable modem connected to an ISP in North Carolina to a Sun workstation via VPN. From the workstation, we used SSH to login to another workstation at N. C. State University. We then telnet'ed to a PC running Linux at UC Davis, and from there we SSH'ed back to a PC running FreeBSD at NC State. As shown in Figure 4.3, the connection chain has a total of 3 stepping-stones and 59 hops and covers a

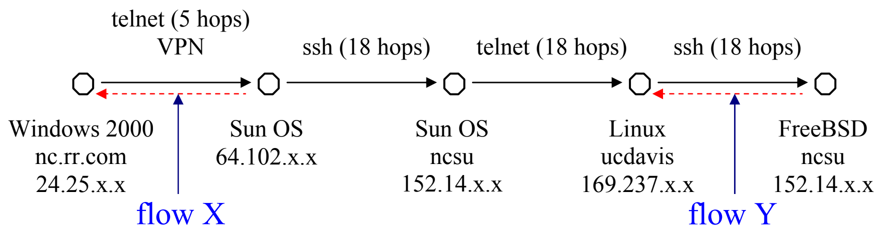


Figure 4.3 Correlation Experiment on telnet and ssh

distance on the order of 10,000 km. The connection chain consists of links of different speeds -including residential Internet access, typical campus LAN and public Internet backbone. We have captured the packet traces at the Windows 2000 node and the FreeBSD node; both traces have a timestamp resolution of 1 microsecond. We label the telnet return path[10] flow from the Sun workstation to the Windows 2000 PC as flow X, and the SSH backwards flow from the FreeBSD PC to the Linux PC as flow Y. Therefore, flow X consists of telnet packets and flow Y consists of SSH packets.

Before calculating the IPD vectors, we have filtered out the following sources of errors from the packet flow:

- Duplicated packets
- Retransmitted packets
- ACK only packets

We then calculated correlation points (j, $j+k$) by applying (4.4) using each of the four correlation point functions, with different correlation window sizes s and correlation point thresholds δ_{cp}.

Figure 4.4 shows the correlation points between flow X and Y obtained by the MMS CPF with different correlation window sizes s and thresholds δ_{cp}. In these plots, a point at position (j, $j+k$) indicates that inequality (4.4) was true for that value of j, k, s and δ_{cp}. True positives are points located along the major diagonal. False positives are points located off the major diagonal.

[10] The “return path” is the echoed traffic generated on the destination host and sent to the origination host.

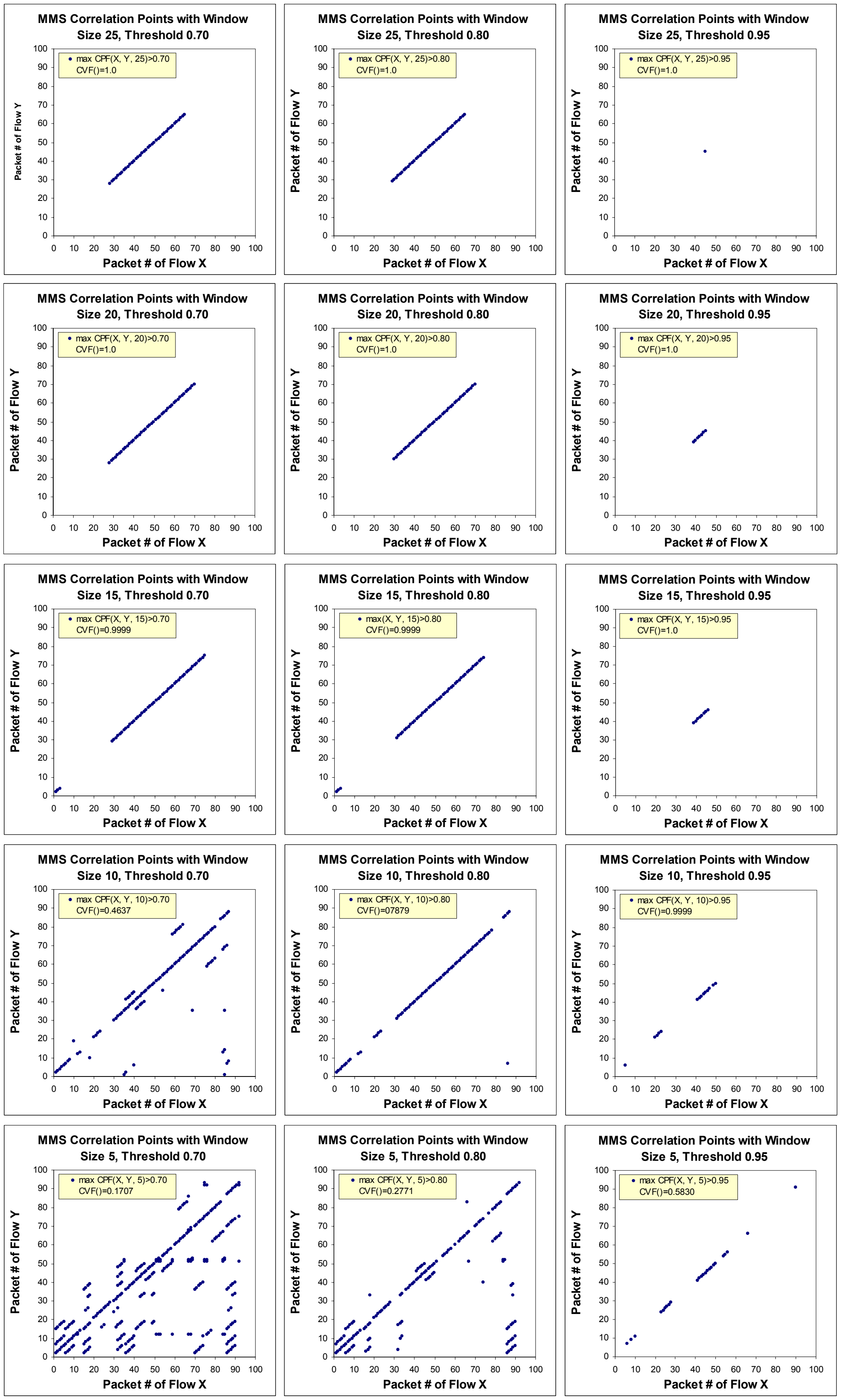


Figure 4.4 Correlation Point between Two Correlated Flows Detected by *MMS* with Different Correlation Window Sizes and Thresholds

With correlation window size of 5 and δ_{cp} threshold of 0.70, there are a large number of falsely detected correlation points (CP false positives) in addition to the true correlation points (CP true positives). The overall CVF value (equation (4.13)) for this case is 0.1707. With larger correlation window size, or a higher threshold δ_{cp}, MMS results in fewer CP false positives and has a higher CVF value, as would be expected. At correlation window size 15, and a threshold δ_{cp} of 0.70, MMS detects most of the true correlation points between flow X and Y, finds no false positives, and has an overall CVF value of 0.9999. When the threshold δ_{cp} is increased to 0.95 with the same correlation window size of 15, MMS detects substantially fewer correlation points between flow X and Y, with no CP false positives, and has an overall CVF value of 1.0. This suggests that with correlation window size 15, the threshold δ_{cp} of 0.95 is probably too high for MMS. The correlation points missed by correlation windows size 15 and threshold δ_{cp} of 0.95 are actually due to correlation-offset shifts. The correlation-offset between our sample flows X and Y has shifted 3 times between the start and finish. This indicates that a telnet packet may trigger more than one SSH packet, or vice versa. Fortunately, such correlation-offset shifts are infrequent between correlated flows. Generally, a larger correlation window size is very effective in filtering out CP false positives, and a higher threshold δ_{cp} tends to filter out both CP true positive and CP false positive. An excessively large correlation window size with a high threshold δ_{cp} tends to have a low CP true positive rate, due to both correlation-offset shifts and IPD variations introduced by the network.

Figure 4.5 compares the detected correlation points between flow X and Y by different CPFs: MMS, STAT, NDP1 and NDP2, with identical correlation window sizes of 10 and threshold δ_{cp} of 0.80. As expected, the statistical CPF results in substantially more CP false positives than the other three CPFs. While NDP2 has slightly fewer CP false positives than NDP1, they both have somewhat more CP false positives than MMS. Generally, MMS is very sensitive to localized details of IPDs and is able to accurately correlate the flows using a smaller correlation window (i.e. 5). NDP1 and NDP2 are less effective with a small correlation window, but they are able to correlate effectively with a moderate window size (15 or 20). The statistical CPF appears to fail to consider enough localized details to correlate accurately.

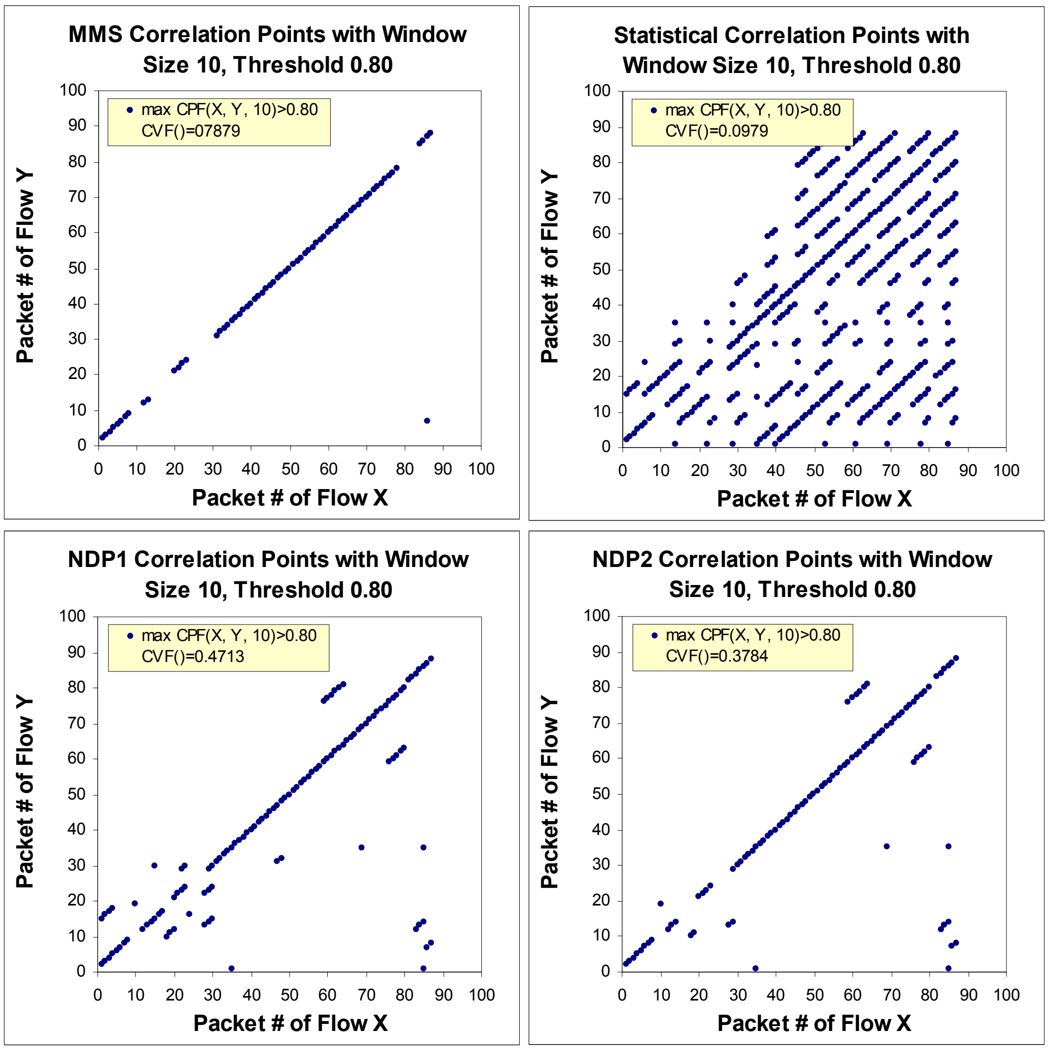


Figure 4.5 Correlation Points Detected by MMS, Stat, NDP1 and NDP2 with Same Window Size and Threshold

4.5.2 Aggregated Flow Correlation Experiment

To evaluate more generally the performance of the different correlation point functions, we have used five sets of flows (Table 4.1). FS1 and FS2 were collected at two ends of connection chains similar to the scenario shown in Figure 4.2. FS1 and FS2 contain 16 SSH flows and 15 Telnet flows, respectively; for each flow in FS2, there is one flow in FS1 which was in the same connection chain. FS3 and FS4 are derived from 5 million packet headers and 12 million packet headers of the Auckland-IV traces of NLANR [52]. FS5 is derived from over 49 million packet headers of the Bell Lab-I traces of NLANR [52].

Flow Set	Date	Flow Type	Flow #	Packet #
FS1	03/02/02	SSH	16	12372
FS2	03/02/02	Telnet	15	5111
FS3	02/20/01	Telnet/SSH	144	34344
FS4	02/26/01	Telnet/SSH	135	38196
FS5	05/xx/02	SSH	400	364158

Table 4.1: Traces of Flows Used in Correlation Experiments

We have conducted four sets of aggregated correlation experiments. Based on results of correlation point experiments shown in Figure 4.4 and Figure 4.5, we choose to set the overall *CVF* correlation threshold δ to be 0.6. For all of these experiments, two flows were regarded as being correlated if the *CVF* of their correlation points (equation (4.13)) was greater than δ=0.6.The first set of experiments tests how well different correlation metrics detect correlation between sets FS1 and FS2. Figure 4.6 shows both the number of true positives (out of a total of 15) and the number of false positives (out of 15*15=225) of flow correlation detection with different correlation window sizes and correlation point thresholds δ_{cp}.

With a δ_{cp} threshold of 0.70, MMS reaches its true positive peak of 93% at a correlation window size of 20, and NDP2 reaches its true positive peak of 80% with a correlation window size of 20 or 25. However NDP2 has a significantly higher number of false positives at the window size corresponding to its peak true positive rate than does than MMS. Both STAT and NDP1 have very low (<7%) true positive rates with all correlation window size. This indicates that STAT and NDP1 are ineffective with a low δ_{cp} threshold.

For all δ_{cp} threshold values, MMS attains its peak true positive rate with 0 false positives. NDP1 and NDP2 show a similar success rate, with a somewhat higher failure (false positive) rate. STAT is generally not successful at correlating the flows in the same chain. The best results are obtained for the highest δ_{cp} threshold setting. MMS is able to achieve 100% true positive rate with 0 false positives with correlation window size 15, δ_{cp} threshold 0.90 and window size 10, δ_{cp} threshold 0.95. NDP2 is also able to have 100% true positive rate with 0 false positive at correlation window size 15, δ_{cp} threshold 0.95. NDP1's overall true positive peak is 93% with 7% false positive at correlation window size 20, δ_{cp} threshold 0.90.

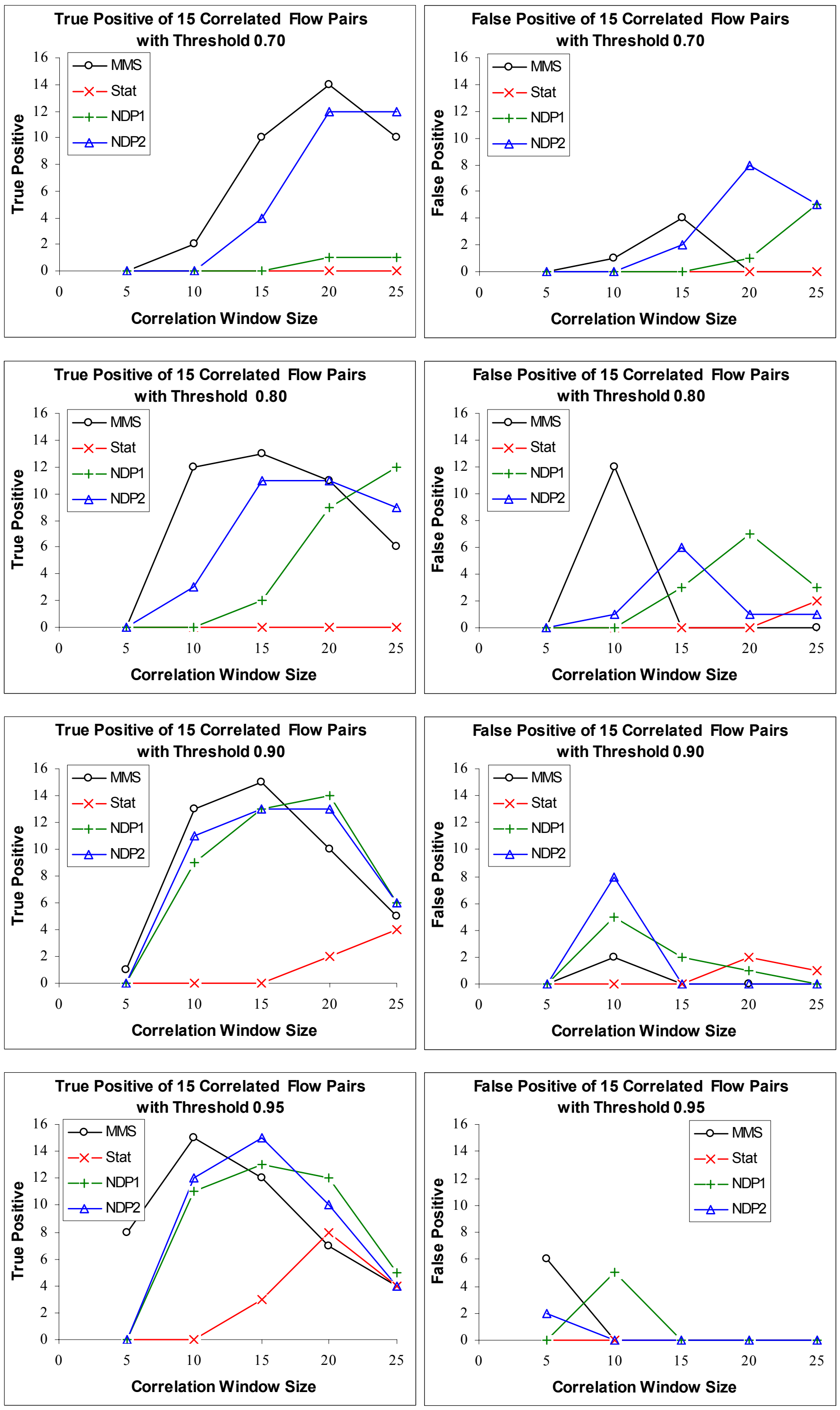


Figure 4.6 True Positive and False Positive of Correlation between 16 and 15 Correlated Flows

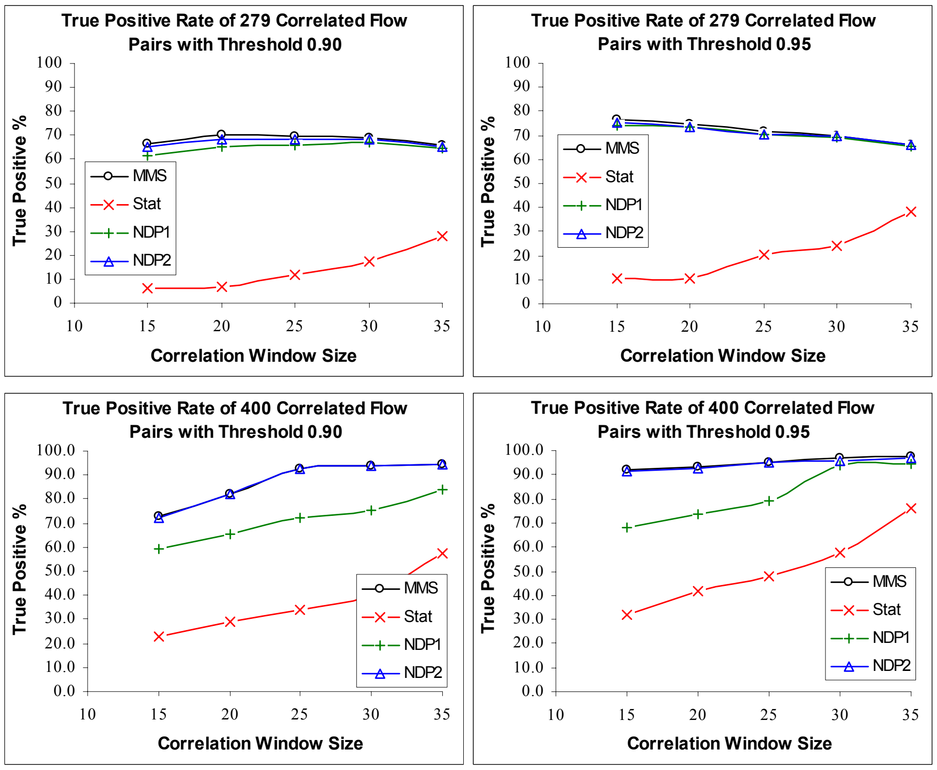


Figure 4.7 True Positive Rate of Correlation between 279 and 279, 400 and 400 Correlated Flows

The second set of experiments shows the correlation detection effectiveness by different correlation metrics. We use combined flow set of FS3 and FS4 (279 flows) and flow set FS5 (400 flows) to correlate themselves respectively. Figure 4.7 shows the true positive rate of different correlation metrics with different correlation window sizes and δ_{cp} thresholds. Again the STAT correlation point function consistently performs poorly. MMS and NDP2 almost have identical correlation detection rates across all the correlation window size and δ_{cp} threshold combinations in both data sets, where NDP1 has little lower detection rate. For flow set FS5, the detection rates of both MMS and NDP2 reach 92% and higher with correlation window size 25 or bigger. At correlation window size 35, MMS's and NDP2's detection rate achieve over 97%. For the combined flow set FS3 and FS4, at a correlation window size of 15, for δ_{cp} threshold 0.95, MMS, NDP1and NDP2 all have the highest correlation detection rate of 76.7%. This lower detection rate is due to the nature of the flows in FS3 and FS4. We have found a number of SSH flows in FS3 and FS4 show very similar periodicity, with constant very short IPDs. We suspect they are bulk data transfers within the

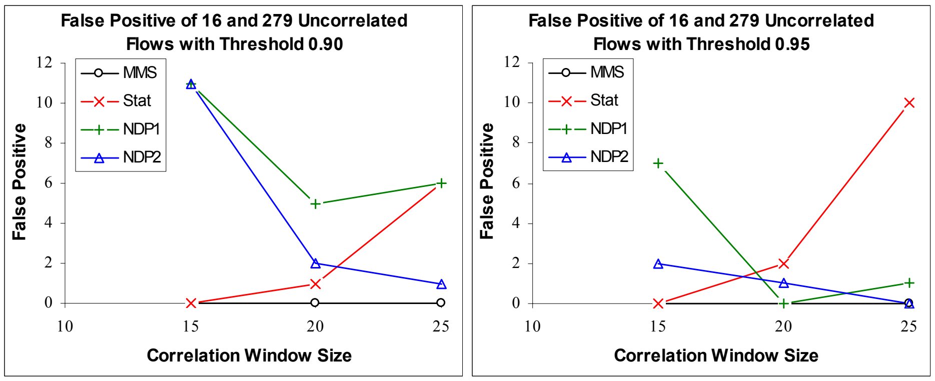


Figure 4.8 False Positive of Correlation between 16 and 279 Uncorrelated Flows

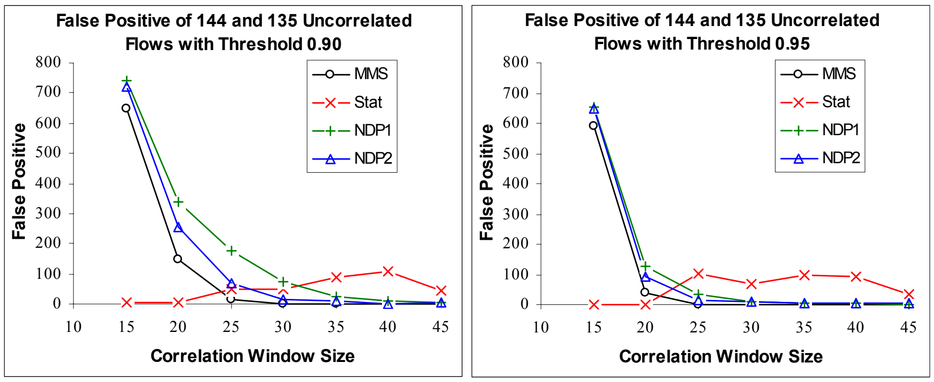


Figure 4.9 False Positive of Correlation between 144 and 135 Uncorrelated Flows

SSH connections. This shows a potential limitation of IPD-based correlation.

The third set of experiments is intended to evaluate the ability of the different correlation point functions to successfully discriminate flows that are not part of the same chain. Figure 4.8 shows the number of false positives (out of 16*279=4464) when correlating FS1 and the combined flow set of FS3 and FS4. Because no flow from FS1 correlates with any flow from FS3 and FS4, any detected correlation by the correlation metric is a false positive. MMS consistently has 0 false positives; and NDP1 and NDP2 false positives decrease as the correlation window size increases. The STAT correlation point function reports an increasing number of false positive with larger correlation sizes.

The fourth set of experiments similarly investigates the false positive rate, this time between sets FS3 and FS4. Figure 4.9 shows the results. The number of false positives (out of 144*135 = 19440) for MMS, NDP1 and NDP2 decreases dramatically when the correlation window size increases; that of MMS decreases faster than NDP1 and NDP2. Again, the statistical correlation metric has a consistently higher false positive rate with increasing correlation window size. For the MMS method, a window size of 20 0r 25 packets is sufficient to reduce the false positive rate to a fraction of a percent.

In summary, we have found that MMS is very effective in both detecting interactive, correlated flows and differentiating uncorrelated flows with even relatively small correlation window sizes (10, 15). NDP1 and NDP2 are not as sensitive as MMS with small correlation windows; however, they both perform well with larger correlation windows. We have confirmed that the statistical correlation metric is not effective in detecting correlation and differentiating uncorrelated flows.

4.5.3 Correlation Performance

	40k	379k	937k	2309k	5704k	54210k
MMS	2.00	1.65	1.74	1.75	1.43	1.35
STAT	0.90	0.76	0.69	1.14	1.22	1.83
NDP1	3.99	3.16	2.23	3.25	2.29	3.24
NDP2	1.33	1.31	1.16	1.37	1.17	1.13

Table 4.2: Throughput (Millions per Second) of Correlation Point Calculation with Correlation Window Size 15

We have measured the number of calculations of correlation points per second achieved by our unoptimized code. Table 4.2 shows the average number of millions of correlation point calculation per second of various correlation point functions under different loads in term of total number of correlation points calculated. Despite dynamic overheads of disk operation, the overall throughput remains largely constant at various loads.

4.6 Summary

The encryption of connections makes the correlation and tracing of intrusion connections through stepping stones much harder. We have addressed the problem of tracing encrypted (as well as unencrypted) connections based on the inter-packet delays of the connections. We proposed and investigated four correlation point functions. Our correlation metric does not require clock synchronization, and allows correlation of measurements taken at widely scattered points. Our method also requires only small packet sequences (on the order of a few dozen packets) for correlation. We have found that after some filtering, IPDs (Inter-Packet Delay) of both encrypted and unencrypted, interactive connections are largely preserved across many hops stepping-stones. We have demonstrated that both encrypted and unencrypted, interactive connections can be effectively correlated and differentiated based on IPD characteristics.

Our experiments also indicate that correlation detection is significantly dependent on the uniqueness of flows. We have found that normal interactive connections such as telnet, SSH and rlogin are almost always unique enough to be differentiated from connections not in the same chain. Bulk data transfer with SSH connection introduces an additional challenge in correlation detection, and its impact on correlation differentiation could be offset by larger correlation windows and higher correlation point thresholds. However, IPD-based correlation is generally not effective in correlating flows of massive data transfer (i.e. ftp) which exhibit similar IPD patterns to each other.

One countermeasure against IPD-based correlation is to set some telnet connection into line mode while keeping other connection in character-at-a-time mode. When telnet client is in line mode, it buffers user's input till the end-of-line is entered, then it tries to send out the whole line input at a time rather than a character at a time. When some part of connection chain uses telnet character-at-a-time mode and some other part of connection chain uses line mode, the boundary and number of packets will be changed. However, the server side shell could always request for character-at-a-time mode [72], and the telnet client always changes to character-at-a-time mode when application at server side requires it.

Another countermeasure the intruders could use to thwart IPD-based correlation is to deliberately perturb the inter-packet timing of some connection in a connection chain by introducing additional delays at some stepping stone. Such timing perturbation could either decrease the correlation true positive rate, or increase the correlation false positives rate. There are relatively simple means of accomplishing such traffic shaping, although they may require kernel-level manipulations. The amount of delay that can be added by the intruder is limited by the maximum delay that is tolerable for interactive traffic.

We will address the active timing perturbation by intruders in next Chapter.

Chapter 5

Robust Correlation of Encrypted Connections through Watermarking

Adding extra delays to packets of encrypted connections will adversely affect any correlation schemes based on inter-packet timing characteristics in that the timing perturbation could either increase the correlation false positive rate or decrease the correlation true positive rate.

In this chapter, we investigate on how to correlate timing perturbed encrypted connections through stepping. Here we assume

- The attackers have total control over the stepping stones and they can freely disguise, delete or forge the host login information at each stepping stone
- The attackers could encrypt any and all connections at any and all stepping stones with any encryption scheme and different keys
- The attacker could add random, but bounded delays at any or all stepping stones when forwarding any or all packets
- The original packets order is kept
- There is no bogus traffic added or packets dropped

Our goal is to understand the adverse impact as well as the inherent limitation of active timing perturbation by adversary over timing based correlation schemes, develop a practical correlation scheme that is robust against random timing perturbation by adversary, and to answer fundamental questions concerning the maximum effectiveness of such techniques and the tradeoffs involved in implementing them.

5.1 Introduction

The timing-based approach is the most capable and promising current method for correlating encrypted connections. However, previous timing-based approaches are vulnerable to packet timing perturbations introduced by the attacker at stepping stones. In particular, the attacker can perturb the timing characteristics of a connection by selectively or randomly introducing extra delays when forwarding packets at the stepping stone. This kind of timing perturbation will adversely affect the effectiveness of any timing-based correlation. The timing perturbation could either make unrelated flows have similar timing characteristics, or make related flows exhibit different timing characteristics. Either case could cause a timing-based correlation method to fail.

Donoho et al [29] have recently investigated the theoretical limits on the attacker's ability to disguise his traffic through timing perturbation and packet padding (i.e., injection of bogus packets). By using a multiscale analysis technique, they are able to separate the long term behavior of the connection from the short term behavior of the connection, and they show that correlation from the long term behavior (of sufficiently long flows) is still possible despite timing perturbation by the attacker. However, they do not present any tradeoffs between the magnitude of the timing perturbation, the desired correlation effectiveness, and the number of packets needed. Another important issue that is not addressed by [29] is the correlation false positive rate. While the coarse scale analysis for long term behavior may filter out the packet timing jitter introduced by the attacker, it could also filter out the inherent uniqueness and details of the flow timing. Therefore coarse scale analysis tends to increase the correlation false positive rate while increasing the correlation true positive rate of timing perturbed connections. Nevertheless, Donoho et al 's work [29] represents a important first step toward a better understanding of the inherent limitations of timing perturbation by the attacker on timing-based correlation. The important theoretical result is that correlation is still achievable for sufficiently long flows despite certain type of timing perturbations. What left open are the question whether correlation is achievable in the presence of arbitrarily distributed (rather than Pareto distribution conserving) random timing perturbation, and an analysis of the achievable tradeoff of the false positive and true positive rates.

In the following sections we show, with our watermark-based correlation, that for sufficient long flows, it is indeed possible to achieve both high true positive rate and low false positive rate at the same time in correlation against arbitrarily distributed *iid* random timing perturbations of arbitrary distribution.

We describe a novel watermark-based connection correlation method that is designed to be robust against random timing perturbations by the attacker. The idea is to actively embed some unique watermark into the flow by slightly adjusting the timing of selected packets in the flow. If the embedded watermark is unique enough and robust enough against the timing perturbation by the attacker, the watermarked flow can be uniquely identified, and thus effectively correlated. It is very desirable to minimize the adjustment of inter-packet timing, so that the watermark embedding is difficult to detect without prior knowledge of how it was created. By utilizing a redundant watermark, we have developed a robust correlation framework for which the following property can be proved: our watermark-based correlation scheme can achieve, with arbitrarily small average adjustment of inter-packet timing, a detection (true positive) rate arbitrarily close to 100%, and a watermark collision (false positive) rate arbitrarily close to 0 at the same time, against an arbitrarily large (but bounded) independent and identically distributed (*iid*) random timing perturbation of arbitrary distribution, as long as there are enough packets in the flow to be watermarked.

5.2 Overview of Watermark-Based Correlation

The objective of watermark-based correlation is to make the correlation of encrypted connections robust against random timing perturbations introduced by the attacker. Unlike existing timing-based correlation schemes, our watermark-based correlation is “active” in that it embeds a unique watermark into encrypted flows by slightly adjusting the timing of selected packets. The unique embedded watermark gives us an advantage over passive timing based correlation in resisting timing perturbation.

We assume the following about the random timing perturbation:

1) While the attacker can add extra delay to any or all packets of an outgoing flow of the stepping stone, the maximum delay he/she can introduce is bounded.
2) The random timing perturbation on each packet is independent and identically distributed (*iid*).
3) All packets in the original flow are kept in their original order, i.e., no padding packet is

added and no packet is dropped by the attacker.

4) While the watermarking scheme may be known to the attacker, the parameters of the watermarking are not known by the attacker.

5.2.1 Watermarking Model and Concept

Generally, digital watermarking[24] involves the selection of a watermark carrier domain and the design of two complementary processes: embedding and decoding. The watermark embedding process embeds the watermark bits into the carrier signal by a slight modification of some property of the watermark carrier, and the watermark decoder process detects and extracts any watermark bits (or equivalently determines the existence of a given watermark) from the carrier signal. To correlate encrypted connections, we propose to use inter-packet timing as the watermark carrier domain.

Given a bidirectional connection, we can split it into two unidirectional flows and process each independently. For a unidirectional flow of $n>1$ packets, we use t_i and t'_i to represent the arrival and departure times, respectively, of the ith packet P_i of a flow incoming to and outgoing from some stepping stone.

Assume without loss of generality that the normal processing and queuing delay added by the stepping stone is a constant $c>0$, and that the attacker introduces extra delay d_i to packet P_i at the stepping stone; then we have $t'_i = t_i + c + d_i$.

We define the *arrival inter-packet delay* (AIPD) between P_i and P_j as

$$ipd_{i,j} = t_j - t_i \tag{5.1}$$

and the *departure inter-packet delay* (DIPD) between P_i and P_j as

$$ipd'_{i,j} = t'_j - t'_i \tag{5.2}$$

We will use IPD to denote either AIPD or DIPD when it is clear in the context. We further define the *impact* or *perturbation* on $ipd_{i,j}$ by the attacker as the difference between $ipd'_{i,j}$ and $ipd_{i,j}$: $ipd'_{i,j} - ipd_{i,j} = d_j - d_i$.

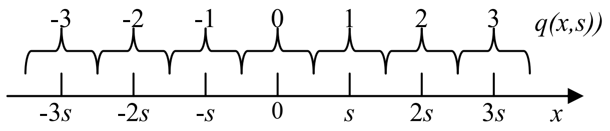


Figure 5.1 Quantization of the scalar value *x*

Assume $D>0$ is the maximum delay that the attacker can add to P_i ($i=1,\ldots,n$), then the impact or perturbation on $ipd_{i,j}$ is $d_j-d_i \in [-D, D]$. Accordingly range $[-D, D]$ is called the *perturbation range* of the attacker.

To make our method robust against timing attacks, we choose to embed the watermark only over selected IPDs. The selection of IPDs consists of randomly choosing the set of packets and random pairing of those chosen packets to get IPDs. The random IPD selection is unknown to the attacker; it should be difficult for the attacker to detect the existence of, extract, or corrupt the embedded watermark, without knowing the IPD selection function and other watermark embedding parameters.

5.3 Embedding One Watermark Bit into One IPD

5.3.1 Basic Watermark Bit Embedding and Decoding

As an IPD is conceptually a continuous value, we will first quantize the IPD before embedding the watermark bit. Given any IPD $ipd>0$, we define the *quantization of ipd* with uniform quantization step size $s>0$ as the function

$$q(ipd, s) = round(ipd / s) \qquad (5.3)$$

where round(x) is the function that rounds off real number x to its nearest integer (i.e., round(x) = i for any $x \in (i - ½, i + ½]$).

Figure 5.1 illustrates the quantization of scalar x. It is easy to see that $q(k \times s, s) = q(k \times s+y, s)$ for any integer k and any $y \in (-s/2, s/2]$.

Let ipd denote the original IPD before watermark bit w is embedded, and ipd^w denote the IPD after watermark bit w is embedded. To embed a binary bit w into an IPD, we slightly adjust that IPD such that the quantization of the adjusted IPD will have w as the remainder when the

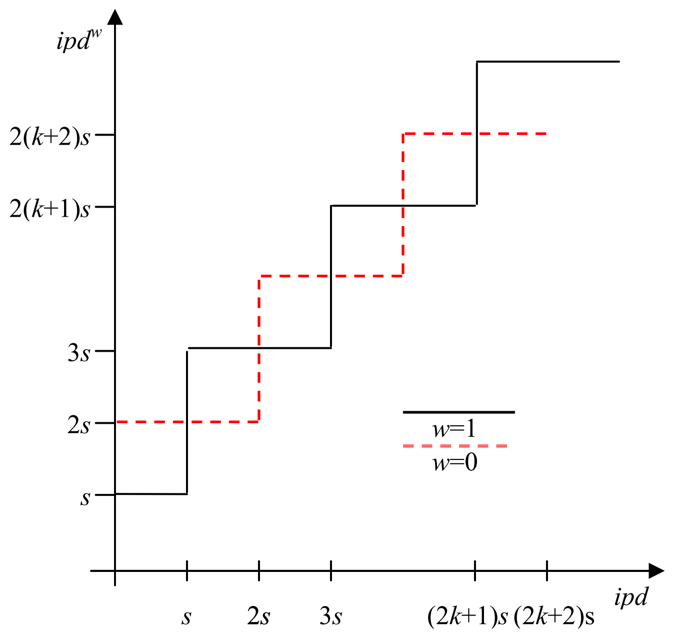


Figure 5.2 Mapping between Unwatermarked *ipd* and Watermark ipd^w after Embedding Watermark Bit *w*

modulus 2 is taken[11].

Given any *ipd*>0, *s*>0 and binary bit *w*, the watermark bit embedding is defined as function

$$e(ipd, w, s) = ceiling((ipd + w \times s) / 2s) \times 2s - w \times s \tag{5.4}$$

where *ceiling*(x) is the function that returns the least integer greater than or equal to x.

The embedding of one watermark bit *w* into scalar *ipd* is done through increasing *ipd* to the least $(2k+w)s$, so that the quantization of resulting ipd^w will have *w* as the remainder when modulus 2 is taken. Figure 5.2 illustrates the embedding of watermark bit *w* by mapping ranges of unwatermarked *ipd* to the corresponding watermarked ipd^w, which is a step function.

The watermark bit decoding function is defined as

$$d(ipd^w, s) = q(ipd^w, s) \bmod 2 \tag{5.5}$$

The correctness of watermark embedding and decoding is guaranteed by the following

[11] We will show later that it is optimal to use modulus 2 and embed the watermark in binary form.

theorems.

THEOREM 5.1. *For any $ipd>0$, $s>0$ and binary bit w, $d(e(ipd, w, s), s) = w$.*

Proof:

Given any $ipd>0$, we can find unique $a\geq 0$ and $0\leq b<2s$ such that $ipd+w\times s=2a\times s+b$. Then we have $ceiling((ipd+w\times s)/2s)=ceiling((2a\times s+b)/2s)=a+1$, and $e(ipd, w, s)=2(a+1)s-w\times s$. Therefore

$d(e(ipd, w, s), s)$

$= q(e(ipd, w, s), s) \bmod 2$

$= q(2(a+1)s-w\times s, s) \bmod 2$

$= round(2(a+1)-w) \bmod 2$

$= [2(a+1)-w] \bmod 2$

$= [2a+2-w] \bmod 2$

$= w$

THEOREM 5.2. *For any $ipd>0$, $s>0$ and binary bit w,* $0 \leq e(ipd, w, s) - ipd < 2s$.

Proof:

By definition of function $ceiling(x)$, we have $x\leq ceiling(x) <x+1$. Then we have

$$(ipd+w\times s)/2s \leq ceiling((ipd+w\times s)/2s) < (ipd+w\times s)/2s)+1$$

Therefore

$$ipd \leq ceiling((ipd+w\times s)/2s)\times 2s-w\times s < ipd+2s$$

That is

$$ipd \leq e(ipd, w, s) < ipd+2s$$

Therefore $0 \leq e(ipd, w, s) - ipd< 2s$.

5.3.2 Maximum Tolerable Perturbation, Tolerable Perturbation Range and Vulnerable Perturbation Range

Given any $ipd>0$, $s>0$, we define the *maximum tolerable perturbation* $\Delta_{\max}$ of $d(ipd, s)$ as the upper bound of the perturbation over ipd such that

$$\forall x>0\ (x<\Delta_{\max} \Rightarrow d(ipd\pm x, s) = d(ipd, s))$$

and either

$$(d(ipd+\Delta_{\max}, s) \neq d(ipd, s)$$

or

$$d(ipd-\Delta_{\max}, s) \neq d(ipd, s))$$

That is, any perturbation smaller than $\Delta_{\max}$ on *ipd* will not change $d(ipd, s)$, while a perturbation of $\Delta_{\max}$ or greater on *ipd* may change $d(ipd, s)$.

We define the *tolerable perturbation range* as the subset of the perturbation range $[-D, D]$ within which any perturbation on *ipd* is guaranteed not to change $d(ipd, s)$, and the *vulnerable perturbation range* as the perturbation range outside the tolerable perturbation range.

Given any $ipd>0$, $s>0$ and binary watermark bit *w*, by definition of quantization *q* in (5.3) and watermark decoding function *d* in (5.5), it is easy to see that when $x \in (-s/2, s/2]$

$$d(e(ipd, w, s)+x, s) = d(e(ipd, w, s), s)$$

and

$$d(e(ipd, w, s)-s/2, s) \neq d(e(ipd, w, s), s).$$

This indicates that the maximum tolerable perturbation, the tolerable perturbation range and the vulnerable perturbation range of $d(e(ipd, w, s), s)$ are $s/2$, $(-s/2, s/2]$ and $(-\mathrm{D}, -s/2] \cup (s/2, \mathrm{D})$, respectively.

In summary, if the perturbation of an IPD is within the tolerable perturbation range $(-s/2, s/2]$, the embedded watermark bit is guaranteed not to be changed by the timing attack. If the perturbation of the IPD is outside this range, the embedded watermark bit may be altered by the attacker. Therefore the larger the value of *s* (equivalently, the larger the tolerable perturbation range), the more robust the embedded watermark bit will be. However, a larger value of *s* may disturb the timing of the watermarked flow more, as the watermark bit embedding itself may add up to $2s$ delay to selected packets.

It is desirable to have a watermark embedding scheme that 1) disturbs the timing of watermarked flows as little as possible, so that the watermark embedding is less noticeable; and 2) ensures the embedded watermark bit is robust, with high probability, against timing perturbations that are outside the tolerable perturbation range (-*s*/2, *s*/2].

In the following section, we address the case when the maximum delay *D*>0 added by the attacker is bigger than the maximum tolerable perturbation *s*/2. By utilizing redundancy techniques, we develop a framework that could make the embedded watermark bit robust, with arbitrarily high probability, against arbitrarily large (and yet bounded) *iid* random timing perturbation by the attacker, as long as the flow to be watermarked contains enough packets.

5.4 Probabilistically Robust Watermarking Over IPD

5.4.1 Embedding Watermark Bit over the Average of Multiple IPDs

We have shown that watermark bit embedded with quantization step size *s* over any *ipd*>0 has maximum tolerable perturbation *s*/2, and the embedded watermark bit is vulnerable to any delay added by the attacker that is greater than *s*/2.

To make the embedded watermark bit probabilistically robust against larger random delays than s/2, the key is to contain and minimize the impact of the random delays on the watermark-bearing IPDs so that the impact of the random delays will fall, with high probability, within the tolerable perturbation range (-*s*/2, *s*/2].

We exploit the assumptions that: a) the attacker does not know the exact IPD(s) where the watermark bit(s) will be embedded; and, b) the random delays added by the attacker are independent and identically distributed (*iid*).

We apply the following strategies to contain and minimize the impact of random delays over the watermark-bearing IPDs:

1) Distributing watermark-bearing IPDs over a longer duration of the flow
2) Embedding a watermark bit in the average of multiple IPDs
3) Offsetting the impact of random delays with each other

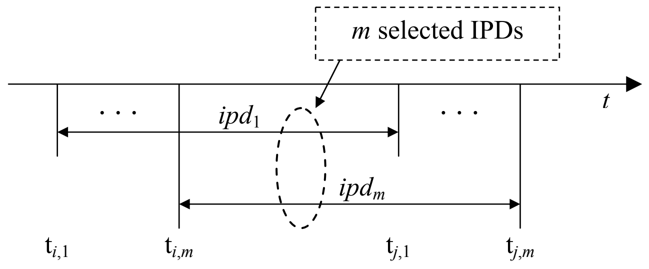


Figure 5.3 Embedding/Decoding Watermark Bit over the Average of Multiple (*m*) IPDs

The rationale behind these strategies is as follows. While the attacker may add a large delay to a single IPD, it is impossible for the attacker to add large delays to all IPDs. In fact, random delays tend to increase some IPDs and decrease others. Therefore the impact on the average of multiple IPDs is more likely to be within the tolerable perturbation range $(-s/2, s/2]$, even when the perturbation range $[-D, D]$ is much larger than $(-s/2, s/2]$.

Instead of embedding a watermark bit in one IPD, we propose to use $m \geq 1$ IPDs. The watermark bit is embedded in the average of the *m* IPDs (as shown in Figure 5.3). Since one bit is embedded in *m* IPDs, we call *m* the *redundancy number*.

Let $<P_{i,k}, P_{j,k}>$ be the *k*-th pair (out of $m \geq 1$ pairs) of the packets selected to embed the watermark bit, whose timestamps are $t_{i,k}$ and $t_{j,k}$ respectively. Then we have *m* IPDs: $ipd_k = t_{j,k} - t_{i,k}$ ($k=1, \ldots, m$). We represent the average of these *m* IPDs as

$$ipd_{avg} = \frac{1}{m} \sum_{k=1}^{m} ipd_k \tag{5.6}$$

Given a desired $ipd_{avg}>0$, and the values for *s* and *w*, we can embed *w* into ipd_{avg} by applying the embedding function defined in (5.4) to ipd_{avg}. Specifically, the timing of the packets $P_{j,k}$ $t_{j,k}$ ($k=1\ldots m$) is modified so that ipd_{avg} is adjusted by $e(ipd_{avg}, w, s) - ipd_{avg}$. If all ipd_k are known before any packet needs to be delayed, we know what ipd_{avg} is, thus we can easily adjust each ipd_k by delaying each packet $P_{j,k}$ for $e(ipd_{avg}, w, s) - ipd_{avg}$. However, during real-time communication, we see one packet at a time, and we may not be able to buffer any packet before we need to forward the newly received packet $P_{j,k}$. In this case, ipd_{avg} is

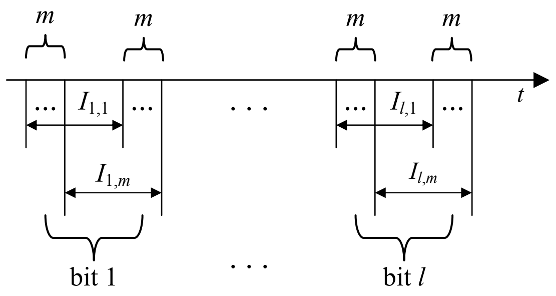


Figure 5.4 Embedding *l*-bit watermark into *l* sequences of IPDs

unknown before the last selected packet is seen by the watermark embedder. This makes the adjustment of $P_{j,k}$ challenging as the watermark embedder doesn't know exactly how much delay needs to be added. Fortunately, the watermark embedder can always adjust the last selected packet $P_{j,m}$ to make the final adjustment on ipd_{avg} correct, even if the timing adjustment on all previous selected packets are not even. The challenge in real-time watermark embedding is how to delay each packet $P_{j,k}$ evenly at real-time to achieve the desired overall adjustment on ipd_{avg}. This is an area of future work.

To decode the watermark bit, we first collect the *m* IPDs (denoted as ipd_k^w, k=1…*m*) from the same *m* pairs of chosen packets and compute the average ipd_{avg}^w of $ipd_1^w \dots ipd_m^w$. Then we can apply the decoding function defined in (5.5) to ipd_{avg}^w to decode the watermark bit.

5.4.2 Embedding Multiple Watermark Bits

We have described how to use $m \geq 1$ IPDs to embed one watermark bit with the desired robustness. Embedding this bit requires 2*m* packets selected (to form *m* packet pairs), and *m* packets delayed appropriately.

An *l*-bit watermark can be embedded simply by applying the above method *l* times, to *l* sequences of *m* packet pairs. This is illustrated in Figure 5.4. It is possible to reduce the number of packets selected from 2*lm* to $(l+1)\times m$ by making the second packet of the *k*th (k=1,…*m*) packet pair chosen for embedding bit *i* the same as the first packet of the *k*th packet pair chosen for embedding bit *i*+1.

The following information about watermark embedding is shared between the watermark embedder and the decoder, which is assumed to be unknown to the attacker.

1) The random selection of the $(l+1)\times m$ packets and random pairing of those $(l+1)\times m$ packets for embedding and decoding the watermark.
2) The redundancy number m.
3) The number of watermark bits l.
4) The quantization step size s.

Without knowing the timing of both watermarked and unwatermarked flows[12], it is difficult for the adversary to find out which packets have been adjusted in timing by watermarking. Without knowing which packets have been adjusted by watermarking, it is difficult for the adversary to find out watermarking parameters from watermarked (or unwatermarked) flow only.

If the adversary has access to the timing of both watermarked and unwatermarked flows, depending on the characteristics of the natural (i.e. processing, queuing, propagation) delay on each packet, the adversary could potentially identify those packets whose timing have been adjusted by watermarking. It is an open problem to establish the computation complexity for identifying watermarking parameters given the characteristics of natural timing perturbation and the timing of both unwatermarked and watermarked flows.

5.4.3 Attacker's Impact Over the Average of Multiple IPDs

Let $d_{i,k}$ and $d_{j,k}$ be the random variables that denote the random delays added by the attacker to packets $P_{i,k}$ and $P_{j,k}$ respectively for k=1,…,m. By assumption, $d_{i,k}$ and $d_{j,k}$ (k=1,…,m) are independent and identically distributed. Therefore $d_{i,1},\ldots,d_{i,m}$ and $d_{j,1},\ldots,d_{j,m}$ form two random samples from the distribution of random delays added by the attacker.

Let $X_k=d_{j,k}-d_{i,k}$ be the random variable that denotes the impact of these random delays on ipd_k

[12] If the flow is watermarked from its source, then the adversary does not see the timing of the original unwatermarked flow.

and $\overline{X_m}$ be the random variable that denotes the overall impact of random delay on *ipd*$_{avg}$. From (5.6) we have

$$\overline{X_m} = \frac{1}{m}\sum_{k=1}^{m}(d_{j,k} - d_{i,k}) = \frac{1}{m}\sum_{k=1}^{m} X_k \tag{5.7}$$

Therefore the impact of the random delay by the attacker over *ipd*$_{avg}$ equals the sample mean of $X_1 \ldots X_m$.

By the result of the previous section, we know that as long as the overall impact of the attacker on *ipd*$_{avg}$ is within the tolerable perturbation range $(-s/2, s/2]$, the watermark bit embedded in *ipd*$_{avg}$ is guaranteed to be unchanged by the timing perturbation by the attacker.

We define the probability that the impact of the timing perturbation by the attacker is within the tolerable perturbation range $(-s/2, s/2]$ as the *watermark bit robustness p*, which can be expressed as $p = \Pr(\ |\overline{X_m}| < s/2\ )$.

Similarly we define the probability that the impact of the timing perturbation by the attacker is out of the tolerable perturbation range $(-s/2, s/2]$ as the *watermark bit vulnerability*, which can be quantitatively expressed as $\Pr(\ |\overline{X_m}| \geq s/2\ )$.

Let σ^2 be the variance of the random delay added by the attacker. Because the maximum delay that may be added by the attacker is assumed to be bounded, σ^2 is finite.

From the properties of the mean and variance of random variables, we have $E(X_k) = E(d_{j,k}) - E(d_{i,k}) = 0$ and $Var(X_k) = Var(d_{j,k}) + Var(d_{i,k}) = 2\sigma^2$. We further have $E(\overline{X_m}) = 0$ and $Var(\overline{X_m}) = 2\sigma^2/m$. This indicates that the probability distribution of $\overline{X_m}$ is more concentrated around its mean than X_k.

According to the Chebyshev inequality in statistics [28], for any random variable X with finite variance $Var(X)$ and for any $t>0$, $\Pr(|X-E(X)|\geq t)\leq Var(X)/t^2$. This means that the probability that a random variable deviates from its mean by more than t is bounded by $Var(X)/t^2$.

By applying the Chebyshev inequality to $\overline{X_m}$ with $t=s/2$, we have

$$\Pr(|\overline{X_m}|\geq s/2)\leq 8\sigma^2/ms^2 \tag{5.8}$$

This means that the probability that the overall impact of *iid* random delays on ipd_{avg} is outside the tolerable perturbation range $(-s/2, s/2]$ is bounded. In addition, that probability can be reduced to be arbitrarily close to 0 by simply increasing m, the number of redundant IPDs averaged for embedding the watermark.

Inequality (5.8) is a powerful result. Regardless of the mean or the variance of the *iid* random delays added by the attacker, or of the maximum quantization delay allowed for watermark embedding, the robustness of the embedding can be made arbitrarily close to 1 by increasing the number of redundant IPDs averaged.

5.5 Analysis on the Distribution of Watermark Bit Robustness

In the previous section, we established an upper bound for watermark bit vulnerability $\Pr(|\overline{X_m}|\geq s/2)$ through the Chebyshev inequality. Now we apply the well-known Central Limit Theorem in statistics[28] to get an accurate approximation to the distribution of the robustness of embedded watermark bit .

Central Limit Theorem. *If the random variables X_1, ..., X_n form a random sample of size n from a given distribution X with mean μ and finite variance σ^2,then for any fixed number x*

$$\lim_{n\to\infty}\Pr[\frac{\sqrt{n}(\overline{X_n}-\mu)}{\sigma}\leq x]=\Phi(x) \tag{5.9}$$

where $\Phi(x)=\int_{-\infty}^{x}\frac{1}{\sqrt{2\pi}}e^{-\frac{u^2}{2}}du$.

The theorem indicates that whenever a random sample of size *n* is taken from any distribution with mean μ and finite variance σ^2, the sample mean $\overline{X_n}$ will be approximately normally distributed with mean μ and variance σ^2/n, or equivalently the distribution of random variable $\sqrt{n}\,(\overline{X_n}-\mu)/\sigma$ will be approximately a standard normal distribution.

Let σ^2 denote the variance of the distribution of the random delays added by the attacker, then we have $\mathrm{Var}(d_{i,k})=\mathrm{Var}(d_{j,k})=\sigma^2$. Applying the Central Limit Theorem to random sample $X_1=d_{j,1}-d_{i,1}$, …, $X_m=d_{j,m}-d_{i,m}$, where $\mathrm{Var}(X_k)=\mathrm{Var}(d_{i,k})-\mathrm{Var}(d_{j,k})=2\sigma^2$ and $E(X_k)=E(d_{j,k})-E(d_{i,k})=0$, we have

$$\begin{aligned}&\Pr[\frac{\sqrt{m}(\overline{X_m}-E(X_i))}{\sqrt{Var(X_i)}}<x]\\&=\Pr[\frac{\sqrt{m}\overline{X_m}}{\sqrt{2}\sigma}<x]\\&\approx\Phi(x)\end{aligned}\tag{5.10}$$

Since $\Phi(x)$ is symmetric, for $x\geq 0$, we have

$$\Pr[|\frac{\sqrt{m}\overline{X_m}}{\sqrt{2}\sigma}|<x]\approx 2\Phi(x)-1\tag{5.11}$$

Therefore,

$$\begin{aligned}p&=\Pr[|\overline{X_m}|<\frac{s}{2}]\\&=\Pr[|\frac{\sqrt{m}\overline{X_m}}{\sqrt{2}\sigma}|<\frac{s\sqrt{m}}{2\sqrt{2}\sigma}]\\&\approx 2\Phi(\frac{s\sqrt{m}}{2\sqrt{2}\sigma})-1\end{aligned}\tag{5.12}$$

This means that the distribution of the watermark bit robustness is approximately normally distributed with zero mean and variance $2\sigma^2/m$.

For random delay $d_{j,k}$ of range $[0, D]$ ($D>0$), we call random variable $d_{j,k}/D$ the normalized random delay whose range is normalized to [0, 1]. Let σ_u^2 denote the variance of the normalized random delay $d_{j,k}/D$, then $\sigma_u^2=\mathrm{Var}(d_{j,k}/D)=\mathrm{Var}(d_{j,k})/D^2=(\sigma/D)^2$. That is $\sigma=D\sigma_u$.

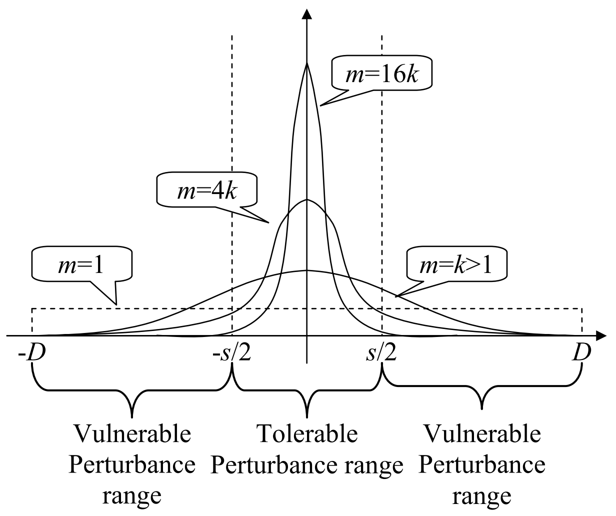


Figure 5.5 Probability Distribution of the Impact of Random Delays over the Average of Multiple (*m*) IPDs

Replace σ in equation (5.12) with $D\sigma_u$, we have

$$\begin{aligned} p &= \Pr[|\overline{X_m}| < \frac{s}{2}] \\ &\approx 2\Phi(\frac{\sqrt{m}}{2\sqrt{2}\sigma_u} \times \frac{s}{D}) - 1 \end{aligned} \tag{5.13}$$

This indicates that given any bounded random delay of any particular distribution, the watermark bit robustness is determined by the square root of redundancy number $\sqrt{m}$ and the ratio between the quantization step and the maximum delay s/D.

Equation (5.12) and (5.13) confirm the result of equation (5.8). Figure 5.5 illustrates how the distribution of the impact of random timing perturbation by the attacker can be "squeezed" into the tolerable perturbation range by increasing the number of redundant IPDs averaged.

Equation (5.12) and (5.13) also give us an accurate estimate of the watermark bit robustness. For example, assume the maximum delay by the attacker is normalized to be 1 time unit, the random delays added by the attacker are uniformly distributed over [0, 1] (whose variance σ^2 is 1/12), s=0.4 units and m=12, then $\Pr[|\overline{X_{12}} < 0.2] \approx 2\Phi(1.2 \times \sqrt{2}) - 1 \approx 91\%$. We can expect the impact of the random delays on the average of those 12 IPDs, with about 91% probability, fall within the range [-0.2, 0.2]. Table 5.1 shows the estimation and simulation results of watermark bit robustness with uniformly distributed random delays over [0, 1], s=0.4 and various sample values for m. It demonstrates that the Central Limit Theorem can give us a precise estimate with a sample size as small as m=7.

m	**7**	**8**	**9**	**10**	**11**	**12**
Estimated Robustness (%)	80.46	83.32	85.84	87.86	89.58	91.02
Simulated Robustness (%)	80.27	83.27	85.68	87.79	89.54	91.02

Table 5.1 Watermark Bit Robustness Estimation and Simulation with Uniformly Distributed Random Delay over [0, 1], and *s*=0.4

From equation (5.12) and (5.13), we can also see that it is easier to achieve the same robustness by increasing s than by increasing m. For example, if s were reduced by a factor of 2, m would have to be increased by a factor of 4 to maintain the same robustness level.

So far we have been using binary form and modulo 2 to embed watermark digits into IPDs, we now consider using non-binary form and modulo b (b>2) to embed watermark digits.

Assume the maximum allowed averaged timing adjustment is C>0, the maximum quantization steps for embedding watermark digits in binary form and non-binary form (with based b>2) are s_2 and s_b respectively. Then we have $2\times s_2=b\times s_b=C$. That is $s_b=2s_2/b$.

Assume the entropy (number of binary bits) of the watermark information needs to be embedded is l_2. Let l_b denote the minimum number of non-binary digits (of base b>2) needed to represent l_2 binary digits. We have $l_b = \log_b 2 \times l_2$.

Let m_2 and m_b denote the redundancy numbers needed to achieve certain watermark bit robustness in binary form and non-binary form of base b respectively. From equation (5.12) or (5.13), we have $s_2\sqrt{m_2} = s_b\sqrt{m_b}$. Replace s_b with $2s_2/b$, we get $m_b = b^2\times m_2/4$.

Therefore, the number of IPDs needed to embed l_b digits of base b is

$$l_b \times m_b = (\log_b 2 \times l_2) \times (\frac{b^2 \times m_2}{4}) = \frac{b^2}{4\log_2 b} \times l_2 m_2$$

It is easy to see that $\frac{b^2}{4\log_2 b} > 1$ for b>2. That means it takes more IPDs to embed same amount of information in non-binary form than in binary form.

In summary, given any particular maximum timing adjustment allowed for embedding watermark of certain amount of information, it is optimal to embed watermark in binary form in terms of minimum number of packets that need to be adjusted.

5.6 Watermark detection, and Analysis of Detection and Collision Rates

Watermark detection refers to the process of determining if a given watermark is embedded in the IPDs of a specific connection or flow.

Let the information shared between the watermark embedder and decoder be represented as <*S*, *m*, *l*, *s*, *wm*>, where *S* is the selection function that returns $(l+1)\times m$ packets and the pairing for $l\times m$ IPDs, $m\geq 1$ is the number of redundant pairs of packets in which to embed one watermark bit, $l>0$ is the length of the watermark in bits, $s>0$ is the quantization step size, and *wm* is the *l*-bit watermark to be detected. Let *f* denote the flow to be examined and wm_f denote the decoded *l* bits from flow *f*.

The watermark detector works as follows:

1) Decode the *l*-bit wm_f from flow *f*.
2) Compare the decoded wm_f with *wm*.
3) Report that watermark *wm* is detected in flow *f* if the Hamming distance between wm_f and *wm*, represented as H(wm_f, *wm*), is less than or equal to *h*, where *h* is a threshold parameter determined by the user, and $0\leq h<l$.

The rationale behind using the Hamming distance rather than requiring an exact match to detect the presence of *wm* is to increase the robustness of the watermark detector against countermeasures by the attacker. Given any quantization step size *s*, there is always a slight chance that the embedded watermark bit is corrupted by countermeasures by the attacker no matter how many redundant pairs of packets are used. Let $0<p<1$ be the probability that each embedded watermark bit will survive the timing perturbation by the attacker. Then the probability that all l bits survive the timing perturbation by the attacker will be p^l. When l is reasonably large, p^l will tend to be small unless p is very close to 1.

By using the Hamming distance h to detect watermark wm_f, the expected watermark detection rate will be

$$\sum_{i=0}^{h}\binom{l}{i}p^{l-i}(1-p)^i \qquad (5.14)$$

For example, for the values p=0.9102, l=24, h=5, the expected watermark detection rate with exact bit match would be p^l =10.45%. For the same values of p, l, and h, the expected watermark detection rate using a Hamming distance h=5 would be 98.29%.

It is possible for the watermark detector to mistakenly report a watermark for a flow in which no watermark has been embedded. It is termed a *collision* between *wm* and *f* if $H(wm_f, wm)\leq h$ for an unwatermarked flow *f*. That is, a collision indicates an unwatermarked flow happens to exhibit the chosen watermark naturally.

Assuming the l-bit wm_f extracted from random flow f is uniformly distributed, then the expected watermark collision probability between any particular watermark *wm* and a random flow *f* will be

$$\sum_{i=0}^{h}\binom{l}{i}(\frac{1}{2})^l \qquad (5.15)$$

Figure 5.6 shows the derived probability distribution of the expected watermark detection and collision rates with l=24 and p=0.9102. Given any watermark bit number l>1 and any

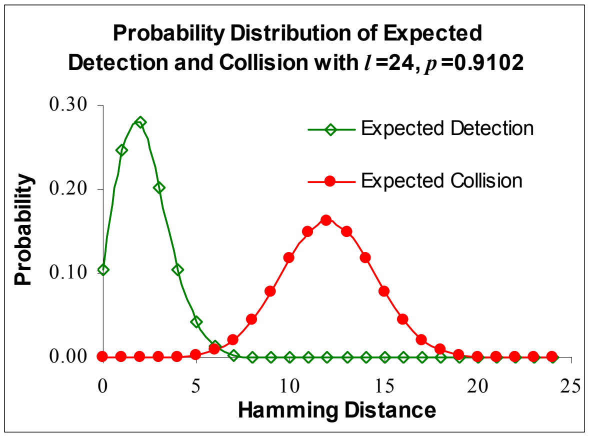


Figure 5.6 Distribution of Expected Watermark Detection and Collision

watermark bit robustness $0<p<1$, the larger the Hamming distance threshold h is, the higher the expected detection rate will be. However, a larger Hamming distance threshold tends to increase the collision (false positive) rate of the watermark detection at the same time. An optimal Hamming distance threshold would be the one that gives a high expected detection rate, while keeping the false positive rate low.

Given any quantization step size $s>0$, any desired watermark collision probability $P_c>0$, and any desired watermark detection rate $0<P_d<1$, we can determine the appropriate Hamming distance threshold $0<h<l$. Assuming that h is chosen such that $h < l/2$ (we can always make this true by increasing l to $2h+1$), then we have

$$\sum_{i=0}^{h}\binom{l}{i}(\frac{1}{2})^l \le \sum_{i=0}^{h}\binom{l}{h}(\frac{1}{2})^l \le (h+1)\frac{l^h}{2^l} \tag{5.16}$$

Because $\lim_{l\to\infty}\frac{l^h}{2^l}=0$, we can always make the expected watermark collision probability $\sum_{i=0}^{h}\binom{l}{i}(\frac{1}{2})^l < P_c$ by having sufficiently large watermark bit number l. Since $\sum_{i=0}^{h}\binom{l}{i}p^{l-i}(1-p)^i \ge p^l$, we can always make the expected detection rate $\sum_{i=0}^{h}\binom{l}{i}p^{l-i}(1-p)^i > P_d$ by having $0<p<1$ sufficiently close to 1. From equation (5.8), this can be accomplished by increasing the redundancy number m.

Therefore, in theory, our watermark based correlation scheme can, with arbitrarily small averaged adjustment of inter-packet timing (for embedding watermark), achieve arbitrarily close to 100% watermark detection rate and arbitrarily close to 0% watermark collision probability at the same time against arbitrarily large (and yet bounded) independent and identically distributed (iid) random timing perturbation of arbitrary distribution, as long as there are enough packets in the flow to be watermarked.

5.6.1 Limitations

In theory, our watermark correlation is effective and robust against random delays that are independent and identically distributed (*iid*) over the set of watermarked packets. For random delays that are independent but have different distributions over the set of watermarked packets, the maximum tolerable perturbation $s/2$ may have to be greater than a specific non-zero value to achieve an arbitrarily high watermark detection rate and arbitrarily low watermark collision rate at the same time. This is due to the fact that the random variable $X_k = d_{j,k}-d_{i,k}$ may have a non-zero mean if $d_{j,k}$ and $d_{i,k}$ are of different distributions. In addition, our watermark correlation method may be not as robust against non-independent random delays. An extreme case would be when the attacker knows exactly which packets have been delayed and by how much, making it much easier to corrupt the embedded watermark bits.

5.7 Experiment

The goal of the experiments is to answer the following questions about watermark-based correlation (as well as existing timing-based correlation) in the face of random timing perturbation by the attacker:

1) How vulnerable are existing (passive) timing-based correlation schemes to random timing perturbations?
2) How robust is watermark-based correlation against random timing perturbations?
3) How effective is watermark-based correlation in correlating the encrypted flows that are perturbed in timing?
4) What is the collision (false positive) rate of watermark-based correlation?
5) How well do the models of watermark bit robustness, watermark detection rate and

watermark collision rate predict the measured values?

We have used three flow sets, labeled FS1, FS1-Int and FS2 in our experiments. FS1 is derived from over 49 million packet headers of the Bell Labs-1 Traces of NLANR [52]. It contains 121 SSH flows that have at least 600 packets and that are at least 300 seconds long. FS2 contains 1000 telnet flows generated from an empirically-derived distribution [26] of telnet packet inter-arrival times, using the tcplib [25] tool.

Because SSH flows may contain non-interactive traffic such those of bulk data transfer and X Window activities, it would be desirable to filter out those non-interactive traffic as much as possible from the SSH flows to get more accurate evaluation on watermark-based correlation of interactive flows. We examine the adjacent IPD of those SSH flows in FS1, and filter out those flows whose adjacent IPDs are too short to be generated by human being. Since it is very unlikely for human being to type more than 14 keystrokes per second, we choose 70ms as the threshold to tell whether the adjacent IPD is generated by human typing or not. We have found 33 out 121 flows in FS1 satisfy the following filtering conditions:

- Have >40% adjacent IPDs shorter than 70ms
- Have >10% 10-consecutive adjacent IPDs all of which are shorter than 70ms

By filtering out those 33 flows from FS1, we got a new flow set FS1-Int.

In principle, the packets selected for watermark embedding should be random in that every packet in the flow should appear to the adversary equally probable to be used by the encoding. In our experiments, the packets selected for watermark embedding are uniformly distributed within a packet range in the flow that enables us to make the adjacent selected packets at least 3 packets apart[13].

5.7.1 Correlation True Positive Experiment

To answer the first three questions, we have conducted the following experiment. First, we used a *passive* timing-based correlation method called IPD-Based Correlation [80] to

[13] If there are enough packets in the flow, it is desirable to spread selected packets further in the flow.

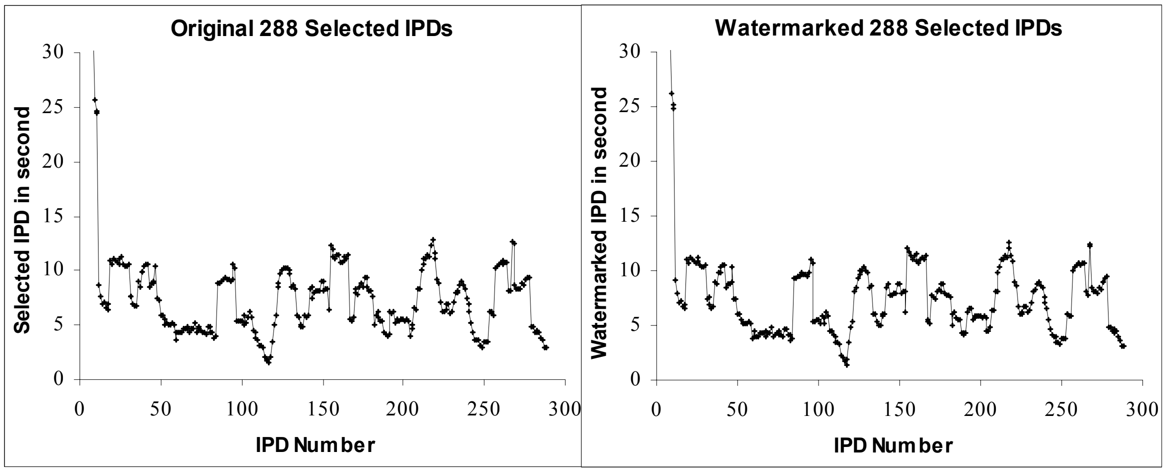


Figure 5.7 Comparison of 288 Selected IPDs before and after Watermark Embedding

correlate each flow in FS1 with the same flow, after the inter-packet delays of the flow have been randomly perturbed. If the flow and the perturbed flow are reported correlated, it is considered a *true positive* (TP) of the correlation in the presence of timing perturbation. Second, we embedded a random 24-bit watermark into each flow of FS1, FS1-Int and FS2, with redundancy number *m*=12, and quantization step size *s*=400ms for each watermark bit. To embed a 24-bit watermark, we evenly selected 300 packets with at least 3 packets apart from each other in the flows. The embedding of 24-bit watermark requires delaying 288 out of the 300 selected packets. Figure 5.7 shows the effect of the watermark embedding, and Figure 5.8 shows the distribution of actual delay over 288 selected packets. They illustrate that the embedding is far from being obvious. Third, we randomly perturbed the packet timing of the watermarked flows of FS1, FS1-Int and FS2. It is considered a *true positive* of watermark-based correlation if the embedded watermark can be detected from the timing perturbed watermarked flows, with a Hamming distance threshold *h*=5. Finally, we calculated the expected detection rate from equations (5.12) and (5.14) under various maximum delays of the random timing perturbation.

Each data point in Figure 5.9 shows the average of 100 separate experiments measuring the true positive rates of IPD-based Correlation on FS1 and watermark-based correlation on FS1, FS1-Int and FS2. The results clearly indicate that IPD-based correlation is vulnerable to even moderate random timing perturbation. Without timing perturbation, IPD-based correlation is

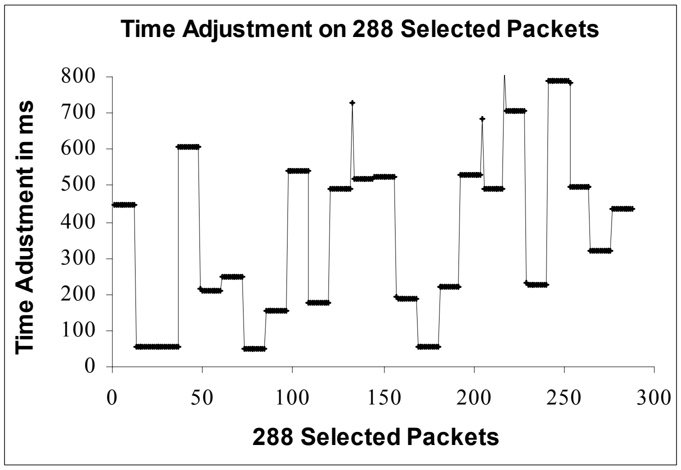


Figure 5.8 Time Adjustment on Selected Packets by Watermark Embedding

able to successfully correlate 93.4% of the SSH flows of FS1. However, with a maximum 100ms random timing perturbation, the true positive rate of IPD-based correlation drops to 45.5%, and for a 200ms maximum delay, the rate drops to 21.5%.

In contrast, the watermark-based correlation of the flows in FS1, FS1-Int and FS2 is able to achieve virtually a 100% true positive rate, with up to a maximum 600ms random timing perturbation. With a maximum 1000ms timing perturbation, the true positive rates of watermark-based correlation for FS1, FS1-Int and FS2 are 84.2%, 89.85% and 97.32%, respectively. It can be seen that the measured watermark-based correlation true positive rates

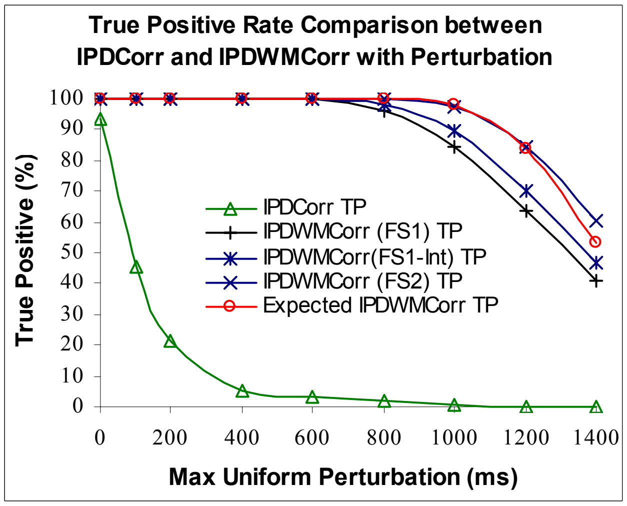


Figure 5.9 Correlation True Positive Rates under Random Timing Perturbation

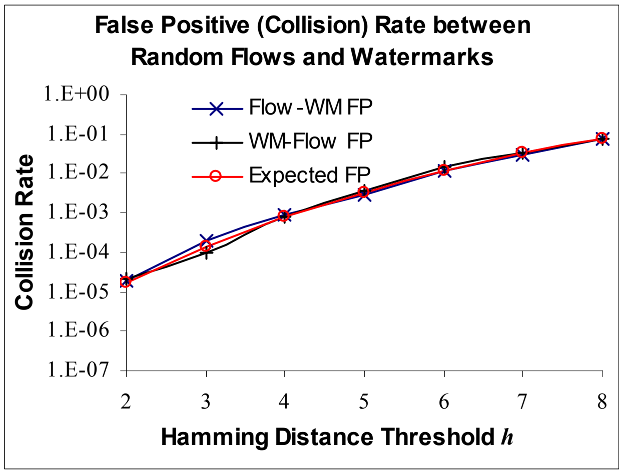


Figure 5.10 Correlation False Positive (Collision) Rate vs Hamming Distance Threshold *h*

are well approximated by the estimated values, based on the watermark detection rate model (equation (5.14)). In particular, the true positive rate measurements of FS2 are almost identical to the estimated values at all perturbation levels.

5.7.2 Correlation False Positive Experiment

As explained above, there is a non-zero probability that an un-watermarked flow will happen to exhibit the randomly chosen watermark. This case is considered a correlation collision, or false positive. According to our correlation collision model (5.15), the collision rate is determined by the number of watermark bits *l* and the Hamming distance threshold *h*.

We therefore experimentally investigated the following, with 24-bit watermarks, for varying values of the Hamming distance threshold *h*:

1) Collision rates between a given flow and 10,000~1,000,000 randomly generated 24-bit watermarks.
2) Collision rates between a given 24-bit watermark and 10,000~1,000,000 randomly generated (using tcplib) telnet flows.

We also experimentally investigated the following, with Hamming distance threshold *h*=3, for varying values of the number of watermark bits *l*:

1) Collision rates between a given flow and 100,000 randomly generated watermarks of

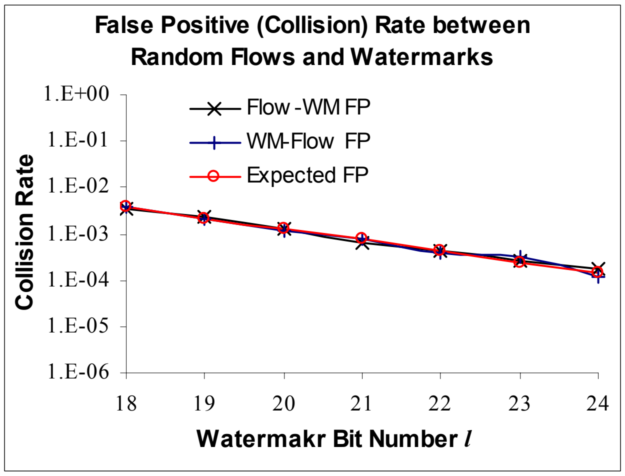


Figure 5.11 Correlation False Positive (Collision) Rate vs Watermark Bit Number *l*

various lengths.

2) Collision rates between a given watermark of various lengths and 10,000~1,000,000 randomly generated (using tcplib) telnet flows.

Figure 5.10 shows the correlation false positive (collision) rates with various Hamming Distance threshold *h*. Figure 5.11 shows the correlation false positive (collision) rates with various watermark bit number *l*. Each data point in Figure 5.10 and figure 5.11 is the average of measured values of 100 separate experiments.

The measured collision rates and expected values are surprisingly close. This validates our assumption that the watermark bits decoded from random un-watermarked flows are uniformly distributed, regardless of the value of the Hamming distance threshold *h*.

5.7.3 Tradeoffs in Watermark Detection Rate

Equation (5.12) gives us the quantitative tradeoff between the expected watermark bit robustness and redundancy number *m*. With a given watermark bit robustness *p*, equation (5.14) gives us the tradeoff between expected watermark detection rate and Hamming distance threshold *h* and number of watermark bit *l*.

To verify the validity and accuracy of our models of watermark bit robustness and watermark

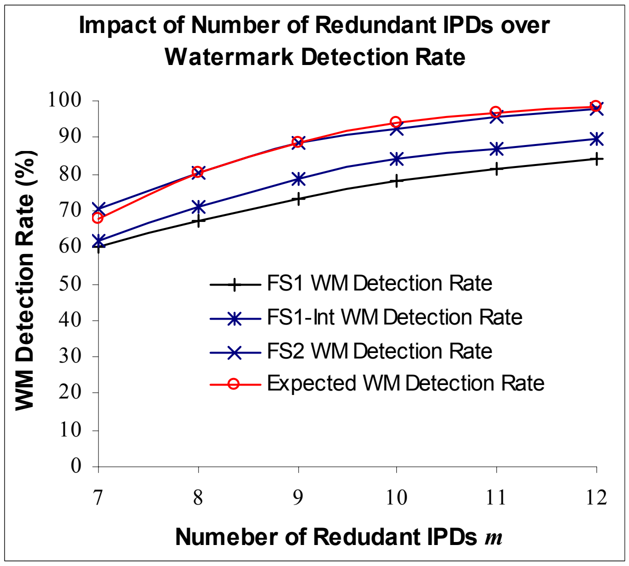


Figure 5.12 Watermark Detection Rate vs Redundancy Number *m*

detection rate, we embedded a random 24-bit watermark into each flow in FS1, FS1-Int and FS2, for different redundancy numbers *m*=7,8,9,10,11,12, different Hamming distance thresholds *h*=2,3,4,5,6,7,8 and different numbers of watermark bit *l*=18,19,20,21,22,23,24. The quantization step *s* was set to 400ms for each watermark bit. Then we perturbed the watermarked flows with 1000ms maximum random delays. Finally, we measured the watermark detection rate of the perturbed, watermarked flows.

Figure 5.12 shows, with *h*=5, *l*=24 and the various values of the redundancy number *m*, the average of 100 experiments for the measured watermark detection rates of FS1 and FS1-Int, the average of 10 experiments for the measured watermark detection rates of FS2, and the expected detection rate derived from equations (5.12) and (5.14). The detection rates of FS2 are very close to the expected values, while the detection rates of FS1 are similar to but lower than the expected values. The detection rates of FS1-Int are in between that of FS1 and FS2.

Figure 5.13 shows, with *l*=24, *m*=12 and the various values of Hamming distance threshold *h*, the average of 100 experiments for the measured watermark detection rates of FS1 and FS1-Int, the average of 10 experiments for the measured watermark detection rates of FS2, and the expected detection rate derived from equations (5.12) and (5.14). The detection rates of FS2 are very close to the expected values, while the detection rates of FS1 are similar to but

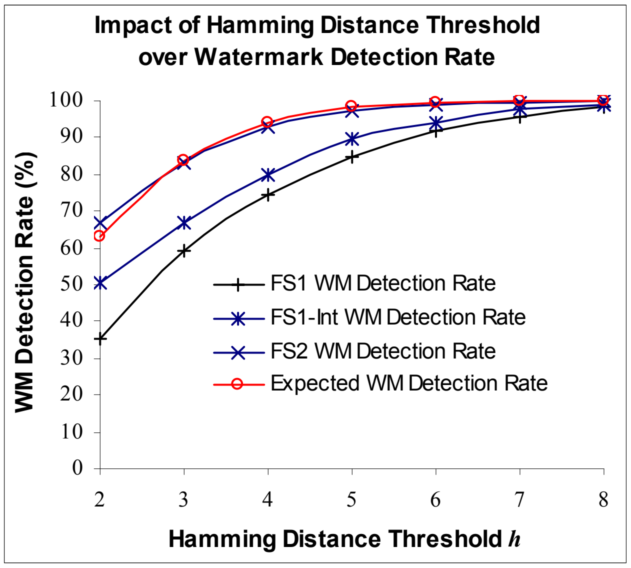


Figure 5.13 Watermark Detection Rate vs Hamming Distance Threshold *h*

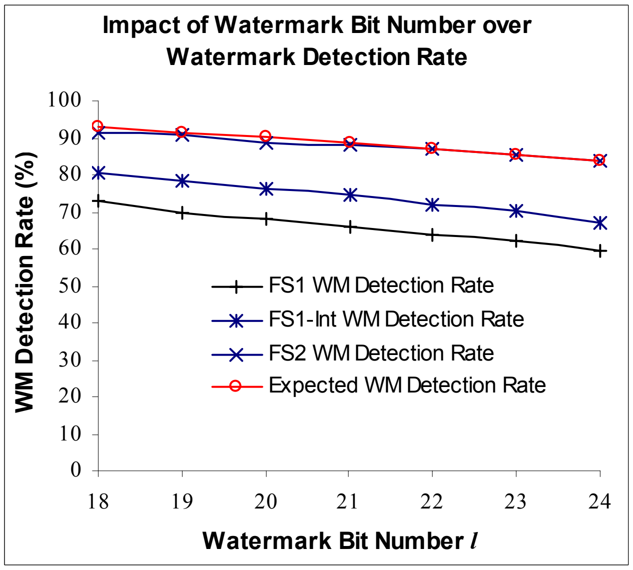


Figure 5.14 Watermark Detection Rate vs Watermark Bit Number *l*

lower than the expected values. The detection rates of FS1-Int are in between that of FS1 and FS2.

Figure 5.14 shows, with h=3, m=12 and various values of the number of watermark bits l, the average of 100 experiments for the measured watermark detection rates of FS1 and FS1-Int, the average of 10 experiments for the measured watermark detection rates of FS2, and the

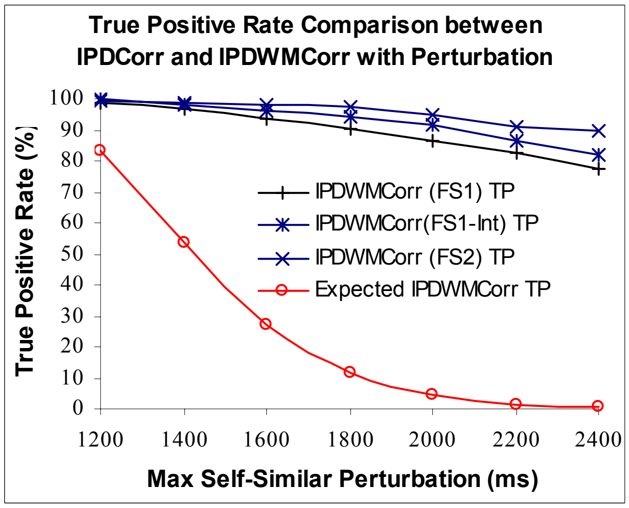


Figure 5.15 Correlation True Positive Rates under Self-Similar Timing Perturbation

expected detection rate derived from equations (5.12) and (5.14). The detection rates of FS2 are very close to the expected values, while the detection rates of FS1 are similar to but lower than the expected values. The detection rates of FS1-Int are in between that of FS1 and FS2.

These results validate our models of watermark bit robustness and watermark detection rate.

5.7.4 Impact of Non-Independent Timing Perturbation over Watermark-Based Correlation

It is one of our assumptions that the random delays added by adversary are independent and identically distributed (*iid*), and we have validated our analysis model empirically with *iid* random timing perturbations. Now we investigate empirically how our watermark-based correlation works in the presence of some non-independent timing perturbations.

5.7.4.1 Self Similar Timing Perturbation

We investigated how our watermark-based correlation works when the inter-packet timing is perturbed by the self-similar (where part resembles whole) type of random delays added by the adversary.

We have used Gleen Kramer's implementation [42] of Taqqu *et al*'s [74] self-similar synthetic traffic generating method to generate bounded self-similar delays for perturbing the inter-packet delays. In particular, we have used 128 aggregating sources of ON/OFF periods to generate self-similar delays in unit of millisecond.

Interestingly, the bounded self-similar traffic appears to have significantly less impact than the same upper bounded uniformly distributed random delays over the watermark correlation true positive rate.

Figure 5.15 shows the measured watermark correlation true positive rates under various bounded self-similar timing perturbation, and expected watermark correlation true positive rates of uniformly distributed random delays, with h=5; l=24; m=12; s=400ms. It clearly shows that self-similar perturbation yields much higher watermark correlation true positive rates than expected true positive rates of uniformly distributed random delay perturbation under same delay upper bounds (1200ms ~ 2400ms).

We have observed that 1) bounded self-similar delays have substantial portions whose values are close to each other; 2) most bounded self-similar delays are quite close to the mean.

As self-similar delays with same upper bound appears more clustered close to the mean than uniformly distributed random delays, self-similar delays must have smaller variance σ^2 than that of uniformly distributed random delays of the same upper bound. From equation (5.12), it is easy to see that the smaller the σ^2 is, the higher the watermark bit robustness is (or equivalently the higher the watermark detection rate is).

In summary, the experiments show that bounded *self-similar perturbation* consistently has less negative impact over watermark correlation true positive rate than the *uniformly distributed random delays* of same upper bound.

5.7.5 Embedding and Detecting Multiple Watermarks at the Same Time

Theoretically, multiple watermarks could be embedded and detected without interfering each other if there is no overlap among the ranges of selected packets for embedding different watermarks. However, if the range of watermark embedding packets of one watermark has overlap with the range of watermark embedding packets of some other watermark, the later embedded watermark could potentially corrupt some bits of the earlier embedded watermark.

To study the potentials and the limitations of multiple watermark embedding and decoding, we first choose 4 different 24-bit watermarks, and then select 4 sets of 300 packets for embedding the 4 watermarks from ranges of [1, 599], [2, 990], [1, 898] and [300, 1197] respectively. With redundancy number m=12, quantization step s=400ms, we repeat the following for 1000 times:

- Randomly generate a flow of 1200 packets (through tcplib)
- Embed watermark #1 into the original flow
- Detect watermark #1 from the flow embedded with watermark #1
- Embed watermark #2 into the flow embedded with watermark #1
- Detect watermark #1 from the flow embedded with watermark #1, #2
- Detect watermark #2 from the flow embedded with watermark #1, #2
- Embed watermark #3 into the flow embedded with watermark #1, #2
- Detect watermark #1 from the flow embedded with watermark #1, #2, #3
- Detect watermark #2 from the flow embedded with watermark #1, #2, #3
- Detect watermark #3 from the flow embedded with watermark #1, #2, #3
- Embed watermark #4 into the flow embedded with watermark #1, #2, #3
- Detect watermark #1 from the flow embedded with watermark #1, #2, #3, #4
- Detect watermark #2 from the flow embedded with watermark #1, #2, #3, #4
- Detect watermark #3 from the flow embedded with watermark #1, #2, #3, #4
- Detect watermark #4 from the flow embedded with watermark #1, #2, #3, #4

	D(WM1)	D(WM2)	D(WM3)	D(WM4)
E(WM1)	100%			
E(WM2)	99.10%	100%		
E(WM3)	42.80%	56.50%	100%	
E(WM4)	28.20%	48.60%	99.10%	100%

Table 5.2 Watermark Detection Rates of Multiple Watermarks Embedded at the Same Time

Table 5.2 shows the detection rates of the 4 watermarks under various conditions. The number at cell (E(WM$_i$), D(WM$_j$)) ($i \geq j$) shows the detection rate of watermark WM$_j$ after WM$_1$, … WM$_i$ have been embedded.

From table 5.2 we can see that even through the range of selected packets of watermark #1 is almost completely within the range of watermark #2, embedded watermark #1 still has a detection rate over 99% after watermark #2 is embedded. Watermark #3, 2/3 of whose range of selected packets is within the range of watermark #4, also has a detection rate over 99% after watermark #4 is embedded. However, the embedding of watermark #3 seems to have more negative impact over watermark #2 embedded. This can be explained by the fact that the range of watermark #2 and the range of watermark #3 are almost identical.

This preliminary result suggests that a later embedded watermark could potentially corrupt a previously embedded watermark – depending on the overlap between the ranges of selected packets for embedding previous watermark and current watermark. When the overlap is small, the previously embedded watermark has good chance to survive the embedding of another watermark. If the overlap between the ranges of selected packet for embedding different watermarks is significant, the later embedded watermark is likely to degrade the detection rate of previously embedded watermark.

5.8 Summary

We presented an active timing-based approach to deal with random timing perturbation. By embedding a watermark into the packet timing, with sufficient redundancy we can correlate in a way that is probabilistically robust against random timing perturbations. Our experiments show that watermark-based correlation is substantially more effective than passive, timing-based correlation in the presence of random timing perturbations.

For independent and identically distributed (*iid*) random delays added by the attacker, our model reveals a rather surprising theoretical result on the limits of watermark-based correlation: *the proposed watermark based correlation scheme can, with arbitrarily small average adjustment of inter-packet timing, achieve arbitrarily close to 100% watermark detection (true positive) rate and arbitrarily close to 0% collision (false positive) probability* *at the same time against arbitrarily large (but bounded) independent and identically distributed (iid) random timing perturbations of arbitrary distribution, as long as there are enough packets in the flow to be watermarked.*

We also developed models of the tradeoff between the watermark detection (or true positive) rate and watermark collision (or false positive) rate. Our experimental results validate the accuracy of these tradeoff models. Thus our tradeoff models are of practical value in optimizing the overall effectiveness of watermark-based correlation in real world situations.

Although our framework assumes the random delays are independent and identically distributed (*iid*), we have found experimentally that certain non-*iid* random delays actually have less adverse impact over the watermark-based correlation than uniformly distributed *iid* random timing perturbation. In particular, we could always select our packets with no obvious repetitive pattern so that the repetitive timing perturbation would have minimum impact. For self-similar type of random timing perturbation, we have found that it almost always has significantly less impact than the uniformly distributed *iid* timing perturbation.

We have also found that embedding and detecting multiple watermarks at the same time are possible. In particular, when there is no substantial overlap among the ranges of selected packets for embedding watermarks, multiple watermarks could be embedded without much interference. This confirms our speculation that embedding another watermark after one watermark is embedded is logically the same as the random timing perturbation on the first embedded watermark, and it is the collision of the ranges of selected packets that determines the impact over the embedded watermarks.

So far we have focused on how to correlate connections through stepping stones under various conditions. However, even a perfect correlation solution has a theoretical limit and there is a gap between the perfect tracing solution and perfect correlation solution. We will address this gap in next chapter.

Chapter 6

The Serialization in Tracing Intrusion Connections through Stepping Stones

This chapter analyzes the theoretical limits of correlation based solutions to the problem of tracing of intrusion connections through stepping stones.

Our goal is to understand the limit on the theoretically achievable effectiveness of the perfect correlation solution, the gap between the perfect stepping stone correlation solution and the perfect stepping stone tracing solution, as well as what it takes to fill the gap.

6.1 Introduction

As we have shown in the overall tracing problem model in Chapter 2, the overall problem of tracing intrusion connections through stepping stones can be divided into two sub-problems: the correlation problem and serialization problem.

Accordingly, a complete solution to the problem of tracing connections through stepping stones consists of two complementary parts. First, the set of correlated connections that belongs to the same intrusion connection chain has to be identified; second, those correlated connections need to be serialized in order to construct the accurate and complete intrusion connection chain.

However, almost all existing approaches to the tracing problem of intrusion connections through stepping stones have focused on identifying the set of correlated connections that belong to the same connection chain and have overlooked the serialization of those correlated connections. While the correlation of encrypted attack traffic is till a challenging task due to various active countermeasures used by adversary, there is a limit on the theoretically achievable effectiveness of even the perfect correlation solution.

In this paper, we use set theoretic approach to analyze the theoretical limits of the correlation-only approach and demonstrate the gap between the perfect correlation solution and the perfect tracing solution of stepping stones problem. In particular, we identify the serialization problem and the loop fallacy in tracing connections through stepping stone. We formally demonstrate that even with perfect correlation solution, which gives us all and only those connections in the connection chain, it is still not adequate to serialize the complete intrusion connection chain deterministically. This is due to the lack of order information from the set of correlated connections. We show that without deterministic connection serialization, the effectiveness of existing correlation-only approaches for tracing intrusion connections through stepping stones could be seriously affected by one simple practice of the attacker: introducing loops by passing some stepping stone more than once. We further demonstrate that correlated connections, even with loops, could be serialized deterministically without synchronized clock. We present an efficient serialization method based on adjacent correlated connection pairs from each stepping stone.

In the rest of this section, we first review the basic concepts of set theory we use, we then use set theoretic approach to formulate and analyze the overall tracing problem.

6.1.1 Ordinals of Basic Set Theory

For binary relation R on set S, we use *Field*(R) to denote the set of elements of each ordered pair in R. That is $\text{Field}(R)=\{x: \langle x,y\rangle \in R \vee \langle y,x\rangle \in R\}$.

Binary relation R is called

Reflexive:	if $\forall x\in\text{Field}(R)\ [x\ R\ x]$
Irreflexive:	if $\forall x\in\text{Field}(R)\ [\neg(x\ R\ x)]$
Symmetric:	if $\forall\ x, y\in\text{Field}(R)\ [x\ R\ y \Leftrightarrow y\ R\ x]$
Anti-symmetric:	if $\forall\ x, y\in\text{Field}(R)\ [x\ R\ y \wedge y\ R\ x \Rightarrow \text{x=y}]$
Transitive:	if $\forall\ x, y, z\in\text{Field}(R)\ [(x\ R\ y \wedge y\ R\ z) \Rightarrow x\ R\ z]$
Linear (connected):	if $\forall\ x, y\in\text{Field}(R)\ [x\ R\ y \vee y\ R\ x]$

Binary relation R on S is a *partial-order* if it is 1) reflexive and 2) anti-symmetric and 3) transitive. Partial order R on S is a *total-order* if it is linear (connected).

Given partial order R on S and $A \subseteq S$, if there exists $a \in A$ such that $\forall x \in A$ $[a\ R\ x]$, we say a is the *R-least* (or *R-minimal*) in A. A total order R on S is a *well-order* on S if every non-empty subset of S has a R-minimal.

6.1.2 Tracing Problem Formulation

For any particular connection chain $<c_1, c_2, \ldots c_n>$, there exists unique (equivalence) binary relation *CORR* that uniquely determines set $\{c_1, c_2, \ldots c_n\}$ and unique (well-order) binary relation $\angle$ that uniquely determines set $<c_1, c_2, \ldots c_n>$. Together binary relations *CORR* and $\angle$ uniquely determine connection chain $<c_1, c_2, \ldots c_n>$. Therefore, the overall tracing problem of connection chain can be divided into the following sub-problems:

1) *Correlation Problem*:

   Given c_1 of some unknown connection chain $<c_1, c_2, \ldots c_n>$, identify set $\{c_1, c_2, \ldots c_n\}$; Or equivalently, given any two connections c and c', determine if c *CORR* c'.

2) *Serialization Problem*:

   Given unordered set of correlated connections $C=\{c_1, c_2, \ldots c_n\}$, serialize $\{c_1, c_2, \ldots c_n\}$ into an ordered set $<c_1', c_2', \ldots c_n'>$ ($c_i' \in C$, $i=1, \ldots n$) such that $c_i' \angle c_{i+1}'$ ($i=1, \ldots n-1$); Or equivalently, given any two connections c and c', determine if c $\angle$ c' or c' $\angle$ c.

Two observations can be made about the overall tracing problem:

1) The result of the serialization problem is based upon the result of the correlation problem
2) The perfect result of the overall tracing problem consists of the perfect result of the correlation problem and the perfect result of the serialization problem based upon the perfect correlation result.

Observation 1) shows the inter-dependency between the correlation problem and the serialization problem, and it explains why existing works on the overall tracing problem have focused on the correlation problem. Observation 2) reveals that while the solution to the correlation problem is the very foundation of the solution to the overall tracing problem, it is not adequate to construct the complete solution to the overall tracing problem. What's missing from the correlation-only approach is the serialization of the correlation result.

In the remainder of this Chapter, we identify, analyze this gap and we present an efficient solution to the serialization problem.

6.2 The Loop Fallacy in Deterministic Serialization of Correlated Connections

Ideally the complete solution to the problem of tracing intrusion connections through stepping stones would give the exact order of the intrusion connections that pass the stepping stones in addition to identifying those correlated connections that belong to the same connections chain. In case the tracing system has limited tracing area (or scope) and there are intrusion connections and stepping stones outside the tracing area, the tracing system should identify the intrusion's first entry point to the tracing area and point out the right direction from which the intrusion comes in. As the stepping stones used by intruders could easily be thousands miles apart and under different jurisdictions, it is critically important to be able accurately identify intrusion's first entry point and point out the right direction from which the intrusion comes in.

Unfortunately, even with perfect correlation solution, which gives all and only those correlated connections within the observing scope that belong to the same connection chain, it is still not adequate to deterministically construct the complete intrusion path. In case there are intrusion connections or stepping stones outside the tracing system, a perfect correlation solution with limited observing area is not sufficient to find the intrusion's first entry point or right direction from which intrusion comes in.

In case the intrusion connection chain passes each stepping stone only once each stepping stone has only one incoming and outgoing connection, and there is only one way to serialize those correlated connections to construct the intrusion path as shown in Figure 6.1.

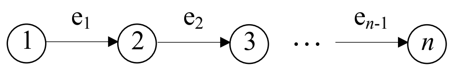


Figure 6.1 Loopless Linear Connection Chain

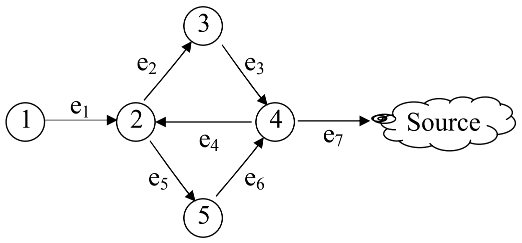


Figure 6.2 The Loop Fallacy in Serializing Correlated Connections

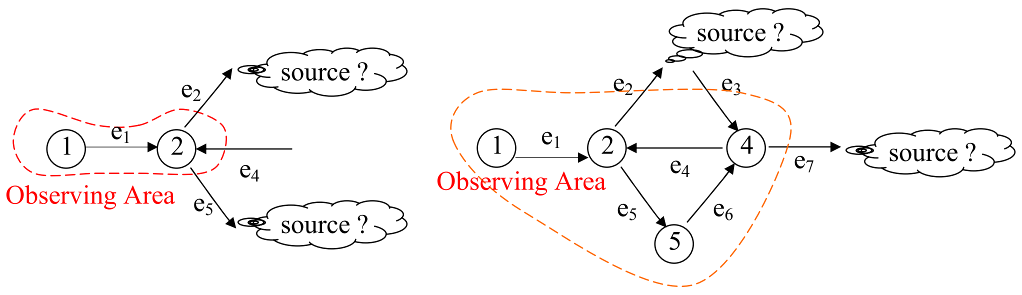


Figure 6.3 Tracing Dilemma with Limited Observing Area

However, when the intrusion connection chain passed some stepping stones more than once, there exists loop or cycle in the intrusion connection chain, and there are more than one ways to serialize those correlated connections. Figure 6.2 shows an example of intrusion connection chain with multiple stepping stones, where node 1 is the intrusion target and e_1, e_2, e_3, e_4, e_5, e_6, e_7 are the backward connections from the intrusion target toward the source of the intrusion. A perfect correlation solution would report that e_1, e_2, e_3, e_4, e_5, e_6, e_7 are correlated and belong to the same intrusion connection chain. Given the knowledge that node 1 is the intrusion target, we know that the intrusion to node 1 comes from node 2 as there is only one correlated connection e_1 from node 1 to node 2. However, node 2 has two outgoing connections e_2 and e_5 that are part of same connection chain, and there are multiple ways to serialize those correlated connections. Furthermore, when some stepping stones are outside the observing area of the tracing system, loops in the intrusion connection chain could introduce dilemma in determining the intrusion's first entry point and the right direction from which the intrusion comes in. Figure 6.3 shows two such examples. When node 3,4,5 are outside the observing area of the tracing system, node 2 sees two correlated outgoing connections e_2 and e_5. Without additional information, there is no way for node 2 to determine which connection points to the right direction to the intrusion source. When node 3 is out of the observing area, there are multiple ways to serializing the correlated connections, which point to different directions to the intrusion source. For example, both serialization $<e_1, e_2, \ldots e_3, e_4, e_5, e_6, e_7>$ and $<e_1, e_5, e_6, e_7, \ldots e_3, e_4, e_2>$ are possible, which imply e_7 and e_2 respectively as the connection pointing to the intrusion source. In this case, a perfect correlation solution is able to detect all the correlated connections, but it is not even able to tell whether node 2 or node 4 is the intrusion's first entry point into the tracing system.

These examples indicate that correlation only approach is a partial solution to the problem of tracing intrusion connections through stepping stones. What is missing from the correlation only solution is the serialization of those correlated connections. It is this phenomenon – that people in general do not take the potential loops or cycles of intrusion connection chain into account when intuitively solving the tracing problem with correlation only approaches – that is named "the loop fallacy" in tracing intrusion connections through stepping stones.

6.3 Deterministic Serialization of Correlated Connections

We have shown in previous section, the set of correlated connections itself is not adequate to serialize those correlated connections deterministically. In order to deterministically serialize correlated connections, some additional information on the correlated connection is needed.

One possible way to serialize correlated connection is use globally synchronized time-stamp to determine the relative order of correlated connections. However, collecting precisely synchronized timestamp on all connections across the internet is difficult due to the following reasons: 1) not all the hosts on the internet have precise clock synchronization; 2) dynamic network delay (which may cause out-of-order delivery) complicates distributed timestamping; 3) distributed clock synchronization is also subject to malicious attacks.

Another way to serialize correlated connections is based on adjacency or causal relationship of those correlated connections. Compared with timestamp based approach, adjacency based approach does not require any global clock synchronization at all and is robust against network delay jitters.

In this section, we focus on solving the problem of deterministic serialization of correlated connections without global clock synchronization. We use set theoretic approach to formally establish that while the set of correlated connection itself is not adequate to serialize those correlated connections, the set of adjacent correlated connection pairs of each stepping stone is sufficient to serialize those correlated connection deterministically even if there is loops with the connection chain.

Given a set of correlated connections C, it can be thought as a set of edges of a directed graph DG such that $DG=<V, E>$, $V=\{x: \exists <x, y> \in C \vee \exists <y, x> \in C\}$ and $E=C$. We assume that there is no *self-loop edge* in DG, that is $\forall <u, v> \in E\ [u \neq v]$. Therefore, the serialization of elements of C can be represented by the ordering of elements of either V or E.

We use $u \rightarrow v$ to represent that there is directed path from u to v. and we define DG to be *one-way connected* if: $\forall u, v \in V\ [\exists\ u \rightarrow v \vee \exists v \rightarrow u]$, and DG to be *edge one-way connected* if: $\forall <u_1, v_1>, <u_2, v_2> \in E\ [v_1 \rightarrow u_2 \vee v_2 \rightarrow u_1]$.

One necessary condition for the serialization of correlated connections to be correct is that the ordering of the correlated connections maintains the one-way connectivity of the edges and end-points of correlated connections.

6.3.1 Point Connectivity and Serialization Based on Point Adjacency

We first consider serialization of correlated connections based on point adjacency property of those correlated connections.

We define **Point-Adjacency** (**P-Adj**) on V as the binary relation $\{<u, v>: <u, v> \in \text{E}\}$. It is easy to see that P-Adj is irreflexive and it models the adjacency relation among the elements of V.

We define **Point Connectivity** (**PC**) as the binary relation on V, such that

1) $\forall \langle u, v\rangle \in E\ [u \text{ PC } v]$
2) $\forall u, v, w \in V\ [\ (u \text{ PC } v \wedge v \text{ PC } w) \Rightarrow u \text{ PC } w]$

Therefore binary relation PC is the *transitive-closure* of P-Adj. Here we use $<_{PC}$ to represent PC. If there exists some $v \in V$, such that $\forall\ u \in V\ [u \neq v \Rightarrow v <_{PC} u]$, we define such an element v as **PC-minimal** on V.

From the definitions, it is easy to see that given a DG, there is only one P-Adj and $<_{PC}$ defined on V.

Here binary relation $<_{PC}$ formally models the directed connectivity among the vertices in V and $u <_{PC} v$ *iff* there exists a path from u to v.

THEOREM 6.1: the necessary and sufficient conditions for $<_{PC}$ to be well-order on V are:

1) DG=<V, E> is one-way connected
2) DG has no directed cycles

PROOF:

Sufficiency:

Given that DG has no directed cycles, $<_{PC}$ is anti-symmetric: $\forall u, v \in V\ [u <_{PC} v \Rightarrow \neg\ (v <_{PC} u)]$. Because DG is one-way connected, $<_{PC}$ is transitive. Therefore $<_{PC}$ is a partial order on V.

Given DG is one-way connected, $\forall u, v \in V\ (u \neq v)$, there exist a directed path either $u \rightarrow v$ or $v \rightarrow u$. We have either $u <_{PC} v$ or $v <_{PC} u$. Therefore, $<_{PC}$ is a total-order on V.

Assume $<_{PC}$ is not a well-order on V, then there exist a non-empty set of vertices $V' \subseteq V$ such that V' does not have PC-minimal. That is $\forall v \in V'$, $\exists u \in V'$ such that $u <_{PC} v$. We list elements of V', starting from $\forall v_1 \in V'$, and adding $v_{i+1} \in V'$ to the left of $v_i \in V'$ if $v_{i+1} <_{PC}$

v_i and $v_{i+1} \notin \{v_i, v_{i-1} \dots v_1\}$ as following:

$$\boldsymbol{v_n \dots v_{i+1}\, v_i \dots v_2\, v_1}$$

Because V' is finite, the above list is also finite. Assume the left-most element of above list is v_n, we have v_i ($1 \le i < n$) such that $v_i <_{\mathrm{PC}} v_n$, therefore $<v_i, v_n, \dots v_i>$ forms a directed cycle in G. This contradicts condition 2). Therefore $<_{\mathrm{PC}}$ well-orders V.

Necessity:

1) Because $<_{\mathrm{PC}}$ is well-order on V, it is total-order on V. $\forall u, v \in$ V ($u \neq v$), we have either $u <_{\mathrm{PC}} v$ or $v <_{\mathrm{PC}} u$. Then there exist a path either $u \to v$ or $v \to u$. Therefore DG is one-way connected.
2) Assume DG has directed cycle of $n>1$ vertices: $v_n \dots v_2\, v_1$, consider non-empty subset of V $\{v_n \dots v_2\, v_1\}$, there is no PC-minimal in that set. This contradicts the prerequisite that $<_{\mathrm{PC}}$ well-orders V. Therefore DG has no directed cycle.

Because an intrusion connection chain may pass a particular stepping stone more than once, which introduces directed cycles in the connection chain, the serialization of end points of correlated connections based on point adjacency is not deterministic.

6.3.2 Edge Connectivity and Serialization Based on Edge Adjacency

We now consider serialization of correlated connections based on edge adjacency relation among those correlated connections. For any connection c between two hosts, we use *Start*(c) to denote the origination host of c and we use *End*(c) to denote the termination host of c.

We define **Edge-Adjacency** (**E-Adj**) on E as the binary relation: $\{<<u, v>, <v, w>>: <u, v>, <v, w> \in \mathrm{E}\}$. It is easy to see that E-Adj is irreflexive and it models the adjacency relation among the elements of E.

We define **Edge Connectivity** (**EC**) as the binary relation on E, such that

1) $\forall e_i, e_j \in \mathrm{E}\ [\mathit{End}(e_i) = \mathit{Start}(e_j) \Rightarrow e_i\ \mathrm{EC}\ e_j]$
2) $\forall\ e_i, e_j, e_k \in \mathrm{E}\ [(e_i\ \mathrm{EC}\ e_j\ \wedge e_j\ \mathrm{EC}\ e_k) \Rightarrow e_i\ \mathrm{EC}\ e_k\ ]$

Therefore binary relation EC is the *transitive-closure* of E-Adj. Here we use $<_{EC}$ to represent EC. If there exists some e $\in$ E, such that $\forall$ $e_i \in$ E [$e_i \neq$ e $\Rightarrow$ $e_i <_{EC}$ e], we define e_i as **EC-minimal** on E.

From the definitions, it is easy to see that given a DG, there is only one E-Adj and $<_{EC}$ defined on E.

Binary relation $<_{EC}$ also models the directed connectivity among vertices of V and $<u_1, v_1> <_{EC} <u_2, v_2>$ *iff* there exists a path from v_1 to u_2.

THEOREM 6.2: the necessary and sufficient conditions for $<_{EC}$ to be well-order on E are:

1) DG=<V, E> is one-way connected
2) DG has no directed cycles
3) DG has no out-branch: $\forall v \in$V (v has at most single successor)

PROOF:

Sufficiency:

Given $\forall <u_1, v_1>, <u_2, v_2> \in$E and $<u_1, v_1> \neq <u_2, v_2>$, we have $u_1 \neq u_2$ because of 3).

Assume $v_1 = v_2$. Consider $u_1, u_2 \in$V, because of 1), there exists path: $u_1 \rightarrow u_2$. Because of 3) we have $v_1 \rightarrow u_2$, that is $v_2 \rightarrow u_2$. Then we have a cycle $<v_2, u_2, v_2>$, and it contradicts condition 2). Therefore $v_1 \neq v_2$.

Assume $v_1 \rightarrow u_2$, because of condition 2), there is no path from u_2 to v_1 (otherwise we have a loop). Because of condition 3), no path from u_2 to v_1 means no path from v_2 to u_1. That is $\forall <u_1, v_1>, <u_2, v_2> \in$E [$<u_1, v_1> <_{EC} <u_2, v_2> \Rightarrow \neg$ ($<u_2, v_2> <_{EC} <u_1, v_1>$)]. Therefore, $<_{EC}$ is a partial order on E.

Assume there is neither path from v_1 to u_2 nor path from v_2 to u_1. Because of 1), we have $u_2 \rightarrow v_1$ and $u_1 \rightarrow v_2$. Because of 3), we have $v_2 \rightarrow v_1$ and $v_1 \rightarrow v_2$. That forms a cycle, which contradicts condition 2). Therefore there is either $v_1 \rightarrow u_2$ or $v_2 \rightarrow u_1$. That is equivalent to either $<u_1, v_1> <_{EC} <u_2, v_2>$ or $<u_2, v_2> <_{EC} <u_1, v_1>$. Therefore $<_{EC}$ is a total-order on E.

Assume $<_{EC}$ is not well-order on E, then there exist a non-empty set E'⊆E such that there is no EC-minimal on E'. That is $\forall <u_1, v_1> \in$ E', $\exists <u_2, v_2> \in$ E' such that $<u_2, v_2> <_{EC} <u_1, v_1>$. We list elements E', starting from $\forall <u_1, v_1> \in$ E', and adding $<u_{i+1}, v_{i+1}> \in$ E' to the left of $<u_i, v_i> \in$ E' if $<u_{i+1}, v_{i+1}> <_{EC} <u_i, v_i>$ and $<u_{i+1}, v_{i+1}> \notin \{<u_i, v_i>, <u_{i-1}, v_{i-1}> \ldots <u_1, v_1>\}$ as following:

$$\boldsymbol{<u_n, v_n> \ldots <u_{i+1}, v_{i+1}> <u_i, v_i> \ldots <u_2, v_2> <u_1, v_1>}$$

Because E' is finite, the above list is also finite. Assume the left-most element of above list is $<u_n, v_n>$, we have $<u_i, v_i>$ $(1 \leq i < n)$ such that $<u_i, v_i> <_{EC} <u_n, v_n>$, therefore $<<u_i, v_i>, <u_n, v_n>, \ldots <u_i, v_i>>$ forms a directed cycle in DG. This contradicts condition 2). Therefore $<_{EC}$ well-orders E.

Necessity:

1) $\forall u, v \in$ V $(u \neq v)$, there exist $e_1, e_2 \in$ E $(e_1 \neq e_2)$ such that u is endpoint of e_1 and v is endpoint of e_2. Without losing generality, we assume e_1=<u, x> and e_2=<v, y>. Because EC well-orders E, it total-orders E. Therefore $e_1 <_{EC} e_2$ or $e_2 <_{EC} e_1$. There exists path either from u to v or from v to u in G.
2) Assume G has directed cycle of $n > 1$ edges: $<v_1, v_2>, <v_2, v_3>, \ldots <v_n, v_1>$, consider non-empty subset of E $\{<v_1, v_2>, <v_2, v_3>, \ldots <v_n, v_1>\}$, there is no PDEC-minimal in that set. This contradicts with the prerequisite that $<_{EC}$ well-orders E. Therefore DG has no directed cycles.
3) Assume DG has out-branch: $\exists <u, x>, <u, y> \in$ E $(x \neq y)$. Because $<_{EC}$ well-orders E, we have either $<u, x> <_{EC} <u, y>$ or $<u, y> <_{EC} <u, x>$. Without losing generality, we assume $<u, x> <_{EC} <u, y>$, then there exist path $x \rightarrow u$. $\{x, \ldots u, x\}$ forms a cycle, which contradicts the necessary condition 2) just proved. Therefore DG has no out-branch: $\forall v \in$ V (v has single successor).

Please be noted that given DG=<V,E>, in order for $<_{EC}$ to well-orders E, DG must have no out-branch, which is not required for $<_{PC}$ to well-order V. Figure 6.4 shows such an example,

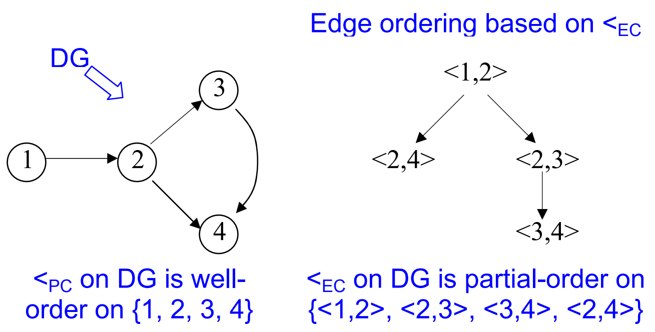

Edge ordering based on $<_{EC}$
DG
<1,2>
<2,4>
<2,3>
<3,4>
1
2
3
4
$<_{PC}$ on DG is well-order on {1, 2, 3, 4}
$<_{EC}$ on DG is partial-order on {<1,2>, <2,3>, <3,4>, <2,4>}

Figure 6.4 Point Connectivity $<_{PC}$ and Edge Connectivity $<_{EC}$

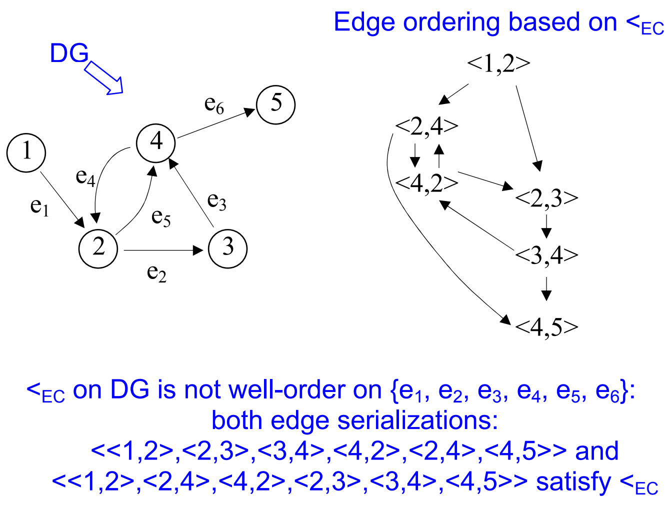

Edge ordering based on $<_{EC}$
DG
<1,2>
<2,4>
<4,2>
<2,3>
<3,4>
<4,5>
e_1
e_2
e_3
e_4
e_5
e_6
1
2
3
4
5
$<_{EC}$ on DG is not well-order on {e_1, e_2, e_3, e_4, e_5, e_6}:
both edge serializations:
<<1,2>,<2,3>,<3,4>,<4,2>,<2,4>,<4,5>> and
<<1,2>,<2,4>,<4,2>,<2,3>,<3,4>,<4,5>> satisfy $<_{EC}$

Figure 6.5 Edge Serialization Based on Edge Connectivity $<_{EC}$

where $<_{PC}$ well-orders {1,2,3,4} and $<_{EC}$ is not even a total-order on E as <2,3> and <2,4> have no relative order.

Because no directed cycles is a necessary condition for $<_{EC}$ to be well-order on E, the serialization of correlated connections based on edge adjacency is not deterministic either. Figure 6.5 shows an example of serialization of connections based on edge adjacency, both edge serializations: <<1,2>,<2,3>,<3,4>,<4,2>,<2,4>,<4,5>> and <<1,2>,<2,4>,<4,2>,<2,3>,<3,4>,<4,5>> satisfy $<_{EC}$.

6.3.3 Serialization Based on Adjacent Connection Pairs

We have demonstrated that the ordering of correlated connections based on either point or edge adjacency is not always deterministic and unique. When the intrusion connection chain

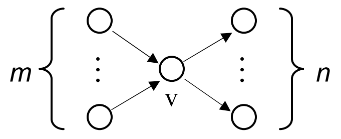


Figure 6.6 *m* Incoming Connections Adjacent to *n* Outgoing Connections

has loops or cycles, there are multiple ways to serialize those correlated connections while keeping the connectivity. This dilemma is due to the fact that there could be more than two connections adjacent to each other through one vertex and the set of correlated connections gives no clue about how to pair match those incoming connections with outgoing connection.

As shown in Figure 6.6, a stepping stone may have multiple incoming connections and outgoing connections correlated. To serialize multiple correlated incoming and outgoing connections deterministically, we need information about how the incoming connections and outgoing connections to and from a stepping stone are pair matched. This is modeled by the concept of adjacent connection pair.

Given a connection chain $<c_1, c_2, \dots c_n>$ on host list $<H_1, H_2, \dots H_{n+1}>$, where connection c_i is from H_i to H_{i+1}, we define $< c_i, c_{i+1}>$ ($i=1,2, \dots n-1$) as the *adjacent connection pair* on host H_{i+1}. Please note that all H_i's ($1 \le i \le n+1$) are not necessarily distinct, but all c_i's ($1 \le i \le n$) are always distinct. Even if both c_i and c_j ($1 \le i,j \le n$ and $i \ne j$) could start from the same host $H_i=H_j$ to the same host $H_{i+1}=H_{j+1}$, connections c_i and c_j are still different based on their setup time. Therefore, the adjacent connection pair carries the relative order information about two adjacent connections on a particular vertex, and $< c_i, c_{i+1}>$ means connection c_i *happens right before* connection c_{i+1}. We use **PE-Adj** to represent the set of adjacent connection pairs. By definition, PE-Adj is anti-symmetric and irreflexive.

Given a set of adjacent connection pairs PE-Adj, we can construct the set of connection

$$E_{\text{PE-Adj}}=\{e: \exists<e, e_i>\in \text{PE-Adj} \lor \exists<e_j, e>\in \text{PE-Adj}\}$$

and the set of vertices

$$V_{\text{PE-Adj}}=\{ v: \exists< e_i, e_j>\in \text{PE-Adj } [v=Start(e_i) \lor v=End(e_i) \lor v=End(e_j)] \}$$

and the directed graph $DG=<V_{\text{PE-Adj}}, E_{\text{PE-Adj}}>$. Therefore PE-Adj is binary relation on $E_{\text{PE-Adj}}$

and PE-Adj $\subseteq$ E-Adj on $E_{\text{PE-Adj}}$.

We define binary relation **Paired Edge Connectivity** (**PEC**) on $E_{\text{PE-Adj}}$, such that

1) $\forall \langle e_i, e_j \rangle \in$ PE-Adj [e_i PEC e_j]
2) $\forall$ $e_i, e_j, e_k \in E_{\text{PE-Adj}}$ [(e_i PEC e_j $\wedge$ e_j PEC e_k) $\Rightarrow$ e_i PEC e_k]

By definition of PEC, e_i PEC e_j means e_i happens before e_j. Therefore binary relation PEC is anti-symmetric. Because of PEC is also transitive, PEC is a partial order. Here we use $<_{\text{PEC}}$ to represent PEC. If there exists $\langle u_1, v_1 \rangle \in$ E, such that $\forall$ $\langle u_2, v_2 \rangle \in$ E [$\langle u_1, v_1 \rangle \neq \langle u_2, v_2 \rangle \Rightarrow$ $\langle u_1, v_1 \rangle <_{\text{PEC}} \langle u_2, v_2 \rangle$], we define $\langle u_1, v_1 \rangle$ as **PEC-minimal** on E.

To utilize result of Theorem 6.1, we transform the directed graph DG into another directed graph. In specific, element of PE-Adj $\langle e_i, e_j \rangle$, can also be thought as a directed edge whose endpoints (tail and head) are e_i and e_j. By mapping edges in DG into vertices and mapping element of PE-Adj, $\langle e_i, e_j \rangle$ into edges, another directed graph can be deterministically constructed.

We define the *paired line graph* of DG, written as PL(DG), as the directed graph whose vertices are the edges of DG, and whose edges are $\langle e_i, e_j \rangle \in$PE-Adj. That is, the edges in DG correspond to vertices in PL(DG), and adjacent connection pairs in DG corresponds to edges in PL(DG).

Therefore V(PL(DG)) $\equiv$ E(DG) $\equiv$ $E_{\text{PE-Adj}}$, PE-Adj on DG corresponds to P-Adj on PL(DG) and $<_{\text{PEC}}$ on DG corresponds to $<_{\text{PC}}$ on PL(DG).

We further define *reachable set* of a particular edge $e \in E_{\text{PE-Adj}}$ as $RS_{\text{PE-Adj}}(e)=\{e_i\text{: } e <_{\text{PEC}} e_i\}$. PE-Adj is *edge one-way connected iff* $\forall e_i, e_j \in E_{\text{PE-Adj}}$ ($e_i <_{\text{PEC}} e_i \vee e_i <_{\text{PEC}} e_i$). PE-Adj is *loop less* iff $\forall e \in E_{\text{PE-Adj}}$ [$e \notin RS_{\text{PE-Adj}}(e)$].

We say PE-Adj is loop less if any connection within the set of adjacent connection pair will not reach itself through the adjacent connection pairs. Because $<_{\text{PEC}}$ is known anti-symmetric, PE-Adj is loopless.

THEOREM 6.3: If PE-Adj is edge one-way connected and loop less, $<_{PEC}$ well-orders E_{PE-Adj}.

PROOF:

Because PE-Adj is edge one-way connected, $<_{PEC}$ total-orders E_{PE-Adj}.

Assume DG=< V_{PE-Adj}, E_{PE-Adj}>, consider the paired line graph of DG: PL(DG)=<V, E>, where V=E_{PE-Adj} and E=PE-Adj. $<_{PEC}$ total-orders E_{PE-Adj} corresponds to $<_{PC}$ on V total-orders V. Therefore PL(DG) is one-way connected.

Because PE-Adj is loop less, $\forall v \in V$, it won't reach v again in PL(DG). That is PL(DG) has no directed cycle.

Apply theorem 6.1, $<_{PC}$ well-orders V on PL(DG), which corresponds to $<_{PEC}$ well-orders E_{PE-Adj}.

For any $e_i \neq e_i \in E_{PE-Adj}$, e_i and e_i are part of some connection chain[14]. Assume without loosing generality that e_i happens before e_i, and the segment between e_i and e_j in the connection chain is: e_i e_{i+1} … e_{i+k} e_i. If PE-Adj contains all the adjacent connection pairs of the connection chain, < e_{i+j}, e_{i+j+1} > $\in$PE-Adj ($0 \leq j \leq k$-1) and < e_{i+k}, e_j > $\in$PE-Adj, that is $e_i <_{PEC} e_j$. Similarly, if e_j happens before e_i, $e_j <_{PEC} e_i$. Therefore, if PE-Adj contains all the adjacent connection pairs from every stepping stone along the connection chain, PE-Adj is edge one-way connected.

Therefore, the complete and accurate intrusion connection chain can be constructed deterministically from the complete set of adjacent correlated connection pairs, even if there are loops within the connection chain. Figure 6.7 illustrate an example of the deterministic serialization of correlated connections from the set of adjacent correlated connection pairs. In

[14] Since we assume we have a perfect correlation solution that gives us all and only those connections in a connection chain.

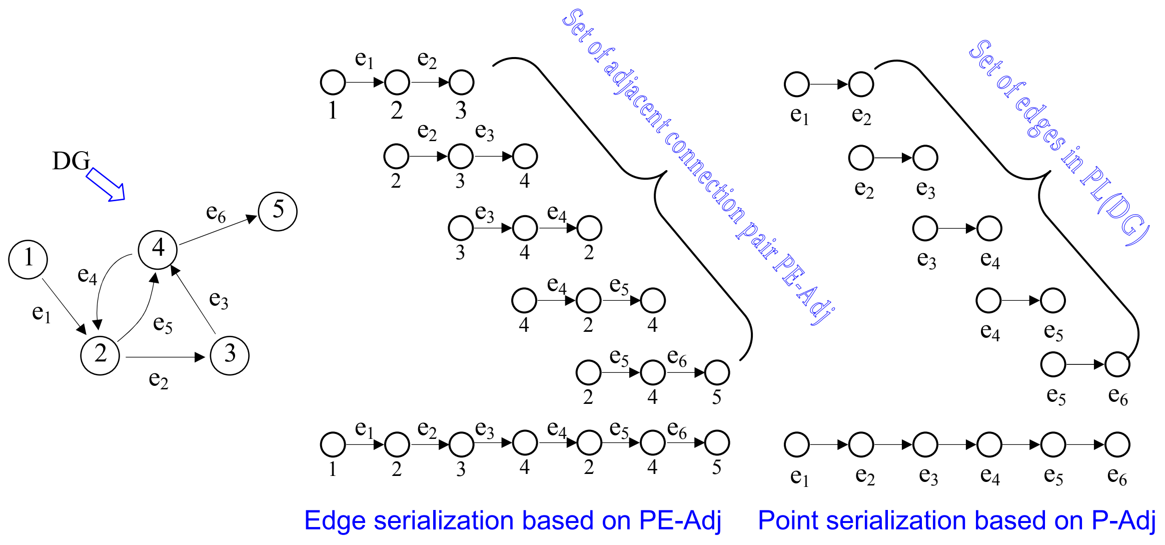

DG
1
2
3
4
5
e1
e2
e3
e4
e5
e6
Set of adjacent connection pair PE-Adj
Set of edges in PL(DG)
Edge serialization based on PE-Adj
Point serialization based on P-Adj

Figure 6.7 Edge Serialization Based on Adjacent Connection Pair PE-Adj

specific, the left graph in Figure 6.7 shows the complete correlated connection chain across all stepping stones; the middle graph shows complete set of adjacent connection pairs PE-Adj and the edge serialization based on PE-Adj; the right graph shows the corresponding point serialization on PL(DG) based on P-Adj.

6.3.4 Finding Adjacent Correlated Connection Pairs

We have established that the complete set of correlated connections can be serialized deterministically based on the complete set of adjacent correlated connections pairs. Now we consider how to find the adjacent correlated connection pairs

We say that the set of adjacent correlated connection pairs is with regard to (*wrt*) connection *c* if *c* is correlated with all connections that form the adjacent correlated connection pairs. The complete set of adjacent correlated connection pairs (with regard to connection *c*) is the union of all subset collected at each stepping stone.

The subset of adjacent correlated connections pairs at each stepping stone can be constructed based on the connection initial arrival or departure time by the following algorithm:

1) For each new incoming (or outgoing) connection I_i (or O_i) that is not self-loop, record I_i

(or O_i) into queue Q: x_1, x_2, …x_{i-1}, where x_j ($1 \le j \le i-1$)could be either incoming or outgoing connection.

2) Using correlation approach to find those, if any, connections that are correlated with c, from all the connections recorded in Q.
3) Extract those correlated connections, in sequence, from Q into correlation queue Q_c.
4) Assume Q_c has c_1, c_2, …c_m, if c_1 is incoming connection, the subset of correlated connection pairs is $\{< c_1, c_2 >, < c_3, c_4 >, \ldots < c_{2\times\lfloor m/2\rfloor-1}, c_{2\times\lfloor m/2\rfloor} >\}$; if c_1 is outgoing connection, the subset of adjacent correlated connection pairs is $\{< c_2, c_3 >, < c_4, c_5 >, \ldots < c_{2\times\lfloor(m-1)/2\rfloor}, c_{2\times\lfloor(m-1)/2\rfloor+1} >\}$.

The correctness of the algorithm is guaranteed by the following property of $Q_c = c_1, c_2, \ldots c_m$: if c_i is incoming connection, then c_{i+1} is outgoing connection; if c_i is outgoing connection, then c_{i+1} is incoming connection.

Therefore, in order to construct the set of adjacent correlated connection pairs, we just need to record the start of all the incoming and outgoing correlated connection at each stepping stone in sequence, from which we can construct the subset of adjacent correlated connection pairs of that stepping stone. Then we can construct the whole set of adjacent correlated connection pairs by union of all the subsets collected at each stepping stone regarding to the same correlation.

For example, assume the sequence of the backward traffic from the attack target to the attack source showed in Figure 6.7 is $<e_1, e_2, e_3, e_4, e_5, e_6>$. By the applying the first three steps of the algorithm described above, node 2 will have its $Q_c = e_1, e_2, e_4, e_5$, node 3 will have its $Q_c = e_2, e_3$, and node 4 will have its $Q_c = e_3, e_4, e_5, e_6$. After step 4, node 2 will have set of correlated connection pairs: $\{<e_1, e_2>, <e_4, e_5>\}$, node 3 will have set correlated connection pairs: $\{<e_2, e_3>\}$, and node 4 will have set correlated connection pairs: $\{<e_3, e_4>, <e_5, e_6>\}$. Therefore, the complete set of the adjacent correlated connection pairs is $\{<e_1, e_2>, <e_2, e_3>, <e_3, e_4>, <e_4, e_5>, <e_5, e_6>\}$.

6.4 Summary

In this paper, we used set theoretic approach to investigate the gap between the perfect stepping stone correlation solution and perfect stepping stone tracing solution. We first identified the largely overlooked serialization problem and the loop fallacy in tracing intrusion connections though stepping stones. Existing approaches to the tracing problem of stepping stones have focused on correlation only and have left the serialization of correlated connections as an afterthought. We demonstrated that even the perfect correlation solution, which gives all and only those correlated connections, is not sufficient to construct the complete intrusion path deterministically, when there is loop in the intrusion connection chain. We further showed that the complete intrusion path can be constructed deterministically from the set of correlated connection pairs, no matter whether there is any loop in the connection chain or not. We presented an efficient algorithm to construct the set of correlated connection pairs and effective method to serialize correlated connections without global clock synchronization.

The solution of serialization is based upon the correlation result, and the correlation of connections through stepping stones is still a challenging and ongoing research task. Our serialization solution helps to increase the effectiveness of existing correlation result. We view our results as complementary to existing correlation approaches in solving the overall problem of tracing intrusion connections through stepping stones.

Chapter 7

Conclusions and Future Work

We conclude this dissertation with a summary of our research contributions and directions for future work.

7.1 Summary

This dissertation addresses the tracing and correlation problem of intrusion connections through stepping stones under various settings. We first identify a number of countermeasures the adversary could use to disguise his/her intrusion connections from being traced across stepping stones, and identify the theoretical limits of correlation only approach in tracing intrusion connections through stepping stones. We analyze the problems posed by the first 4 countermeasures and we design, implement and evaluate a number of solutions to the correlation and tracing problems under these countermeasure settings.

Our major research contributions are:

- We are the first to demonstrate that single packet tracing of unencrypted intrusion connections across stepping stones is feasible
- We are the first to demonstrate that tracing and correlating idle intrusion connections is feasible
- We have developed an efficient and effective method to correlate encrypted connections observed from different point of network
- We are the first to reveal that it is indeed possible to correlate sufficiently long, encrypted connections that are perturbed by arbitrary large *iid* random timing perturbation of arbitrary distribution.
- We are the first to reveal that it is possible to achieve arbitrarily close to 100% true positive rate and arbitrarily close to 0 false positive at the same time in correlating sufficiently long encrypted flows, even under arbitrarily large *iid* random timing perturbation of arbitrary distribution.

- We have identified accurate quantitative tradeoff between the achievable correlation effectiveness and the defining characteristics of the *iid* random timing perturbation. The achievable tradeoff is a more fundamental measure of the effectiveness of correlation solutions and the inherent difficulty of the correlation problem. It is not only of significant practical importance in optimizing the overall correlation effectiveness under various settings but also useful for evaluating and comparing other solutions as well as providing insight for designing new solutions.
- We have identified the gap between the perfect tracing solution and correlation solution of stepping stone problem and we have shown what it takes to fill the gap. Our serialization solution complements all existing correlation solutions and helps to increase their effectiveness.
- We have demonstrated that active approach, information hiding and redundant techniques can be used to build highly effective and robust intrusion connection correlation scheme.

7.1.1 Tracing and Correlation of Unencrypted Connections across Stepping Stones

Our first contribution is a highly effective and efficient correlation and tracing solution for tracing unencrypted intrusion connections across stepping stones, which is robust against the following countermeasures: 1) host login information disguise, deletion and forgery; 2) timing perturbation; 3) flow level traffic padding; 4) packet drop and retransmission; and 5) packet reorder. By applying principles of steganography and active networking, we develop a content based tracing and correlation framework: Sleepy Watermark Tracing (SWT). SWT exploits two properties of the connections across stepping stones: 1) the essence of application level content is invariant across stepping stones; 2) interactive intrusion connections across stepping stones are bidirectional and symmetric at the granularity of connection. Unlike other tracing and correlation approaches [40,65,69], SWT is "sleepy" in that it does not introduce overhead when no intrusion is detected, yet it is "active" in that when an intrusion is detected, the intrusion target "injects" carefully designed watermark into the backward response traffic of the intrusion connection. SWT traces and correlates intrusion connections based on the injected watermarks in their application content rather than host login activities. All previous tracing and correlation approaches require that the intrusion connection not be idle and need multiple packets to correlate, but SWT is able to trace through the intrusion connection chain across all stepping stones within a single keystroke of the intruder. With its unique active tracing, SWT is able to trace through all the stepping stones even when the intrusion connection is idle.

7.1.2 Correlation of Encrypted Connections across Stepping Stones

Our second contribution is an effective Inter-Packet Delay based correlation approach for tracing encrypted connections across stepping stones, which is robust against the following countermeasures: 1) host login information disguise, deletion and forgery; and 2) connection content transformation. In theory, connection transforms such as payload encryption, compression and padding should not change the inter-packet timing characteristics. We have found that after some filtering, IPDs (Inter-Packet Delay) of both encrypted and unencrypted, interactive connections are largely preserved across many router hops and stepping-stones. We proposed and investigated four correlation point functions. Compared with the method of Zhang and Paxson [90], our correlation metric does not require clock synchronization, and allows correlation of measurements taken at widely scattered points. Our method also requires only small packet sequences (on the order of a few dozen packets) for correlation purposes. We have demonstrated that both encrypted and unencrypted interactive connections can be effectively correlated and differentiated based on IPD characteristics.

Our experiments also indicate that correlation detection is significantly dependent on the uniqueness of flows. We have found that normal interactive connections such as telnet, SSH and rlogin are almost always unique enough to be differentiated from connections not in the same chain.

7.1.3 Robust Correlation of Encrypted Connections against Active Timing Perturbation

Our third contribution is the development of robust correlation of encrypted connections though stepping stones against active timing perturbation by theadversary, as well as the identification of the inherent limitation of the negative impact of *iid* random timing perturbation over timing based correlation.

To counteract the adverse impact of active timing perturbation over encrypted connections through stepping stones, we develop inter-packet delay based watermark correlation that is designed to be robust against random timing perturbation by adversary. The key idea is to actively embed some unique watermark into the flow by slightly adjusting the timing of selected packets in the flow and to use redundancy techniques to make the embedded watermark robust. If the embedded watermark is unique enough and robust enough against the timing perturbation by adversary, the watermarked flow could be uniquely identified and thus effectively correlated. By utilizing redundancy techniques, we develop a robust watermark correlation framework that reveals a rather surprising result on the inherent limits of independent and identically distributed (*iid*) random timing perturbations over sufficiently long flows. First, we demonstrate that a previously-proposed passive, timing-based correlation scheme [80] is vulnerable to random timing perturbation. Second, we demonstrate that our watermark-based correlation scheme is much more robust than existing passive timing-based correlation against random timing perturbations. Our experimental results show that the new method consistently has a higher detection (true positive) rate, no matter whether there is random timing perturbation or not. Third, we have developed a robust correlation framework for which the following property can be proved: our watermark-based correlation scheme can achieve a detection (true positive) rate arbitrarily close to 100%, and a watermark collision (false positive) rate arbitrarily close to 0 at the same time, for an arbitrarily large (but bounded) independent and identically distributed (*iid*) random timing perturbation of arbitrary distribution, with arbitrarily small adjustment of inter-packet timing, as long as there are enough packets in the flow to be watermarked. Lastly, we identify accurate models of the tradeoffs between the desired watermark correlation true positive rate (and false positive rate) and the watermark embedding parameters, as well as the defining characteristics of the random timing perturbation. The quantitative expression of the tradeoffs is of significant practical importance in optimizing the overall correlation effectiveness under a range of conditions.

7.1.4 Deterministic Serialization in Tracing Intrusion Connections through Stepping Stones

Our fourth contribution is the identification of the largely overlooked limitation of correlation-only approaches in tracing connections across stepping stones and the solution to bridge the gap between the perfect correlation solution and the perfect tracing solution.

We use a set theoretic approach to analyze the theoretical limits of the correlation-only approach and demonstrate the gap between the perfect correlation-only approach and the perfect solution to the tracing problem of stepping stones. In particular, we identify the serialization problem and the loop fallacy in tracing connections through stepping stones. We formally demonstrate that even with a perfect correlation solution, which gives us all and only those connections that belong to the same connection chain, it is still not adequate to serialize the correlated connections in order to construct the complete intrusion path deterministically. We further show that correlated connections, even with loops, could be serialized deterministically without synchronized clock. We present an efficient intrusion path construction method based on adjacent correlated connection pairs.

7.1.5 General Principles and Lessons

In addition to solving specific problems that arise from various countermeasures, we also derive a set of general principles and lessons to the effective and robust correlation and tracing of intrusion connections. These include:

- **Active approach:** Traditional tracing and correlation approaches are passive in that they passively monitor and examine the intrusion connections. One fundamental limitation of such passive approaches is that they can correlate or trace only when there is traffic in the intrusion connection. Therefore, passive correlation and tracing approaches are vulnerable to deliberate silence by intruders when correlating and tracing intrusion connections. In our work, we apply an active approach in designing our tracing and correlation scheme, such that the intrusion target could actively generate the backward traffic and inject watermark into it. As a result, SWT is able to trace and correlate even when there is no traffic from the intruder. The same active technique could also be applied in IPD-based correlation and IPD-based watermark correlation to make them able to correlate when the intrusion connections are idle.

  The active timing perturbation by adversary imposes great challenge to all timing based correlation approaches, and all passive timing based correlation schemes are vulnerable to active timing perturbation. While the adversary's active timing perturbation could make the timing based correlation more difficult, we have found that we could actually make the timing based correlation easier and more robust by actively embedding unique watermark into the inter-packet timing. The slightly but active adjustment of timing of selected packets gives us significant advantage over passive timing based correlation approaches in the presence of active timing perturbation by adversary. Our analysis and experiments show that our active IPD watermarking correlation is much more effective and robust in correlating (both encrypted and unencrypted) connections than passive IPD-based correlation when the connections are randomly perturbed in timing.

- **Information hiding:** A second general design principle we have used is information hiding. Information hiding contributes to both the effectiveness and robustness aspects of our correlation and tracing approaches. Because of the embedded watermark, the watermarked flow can be made unique in some way and thus easier to correlate and trace. In particular, it is the embedded unique watermark in SWT that enables the capability of one-packet correlation and tracing of unencrypted connections. For encrypted connections, the unique IPD-based watermark makes IPD-based watermark correlation have a higher correlation detection rate than passive IPD-based correlation, even when there is no timing perturbation.

  Because the watermark is hidden information, it is difficult for the adversary to detect the existence of, remove, corrupt or forge any watermark without knowing the specific watermark embedding parameters. Our experiments show that IPD-based watermark correlation is robust to random, repetitive and self-similar timing perturbations.

- **Redundancy technique:** This is one of the key techniques that make IPD-based watermark correlation robust. By utilizing redundancy technique, we develop a robust watermark correlation framework such that our embedded watermark bit can be made with robustness arbitrarily close to 100%, no matter how big the maximum random delay could be, no matter what the distribution of the random timing perturbation is, as long as the random timing perturbation is independent and identically distributed (*iid*) and there are enough packets in the flow to watermark. In addition, the overall watermark correlation could achieve, with arbitrarily small averaged adjustment of inter-packet timing, arbitrarily close to 100% correlation detection (true positive) rate and arbitrarily close to 0% watermark collision (false positive) probability at the same time, as long as the watermark bit has enough redundancy.

7.2 Future Directions

There are several interesting directions for future work based on the work described in this dissertation. Some of these are extensions of our work, and some others are motivated by the more general problems of network security in face of network based attacks.

7.2.1 Robust Correlation against Sophisticated Countermeasures

In this dissertation, we addressed the problem of tracing and correlation of intrusion connections across stepping stones, and investigated problems posed by the following countermeasures by adversary in concealing their source of attack: 1) Host Login Information Disguise, Deletion and Forgery; 2) Connection Content Transformation (i.e. encryption, compression); 3) Active Timing Perturbation; and 4) Introducing Loops in Intrusion Connection Chain. We have successfully developed effective solutions against these countermeasures through combination of active approach, information hiding and redundancy techniques. We have not addressed the following advanced countermeasures against the tracing and correlation of attack traffic: 1) Traffic Padding (i.e. adding bogus packet, packet padding); 2) Packet Drop and Retransmission; 3) Flow Repacketization; 4) Packet Reorder; 5) Flow Split and Merge; and 6) Mixing Multiple Flows (i.e. tunneling). These sophisticated countermeasures form significant challenges in tracing and correlating encrypted intrusion connections across stepping stones by drastically changing the inter-packet timing characteristics. While we expect our active approach, information hiding and redundancy techniques could applied in solving these problems, there are other important problems to overcome. For example, some sophisticated countermeasures move or even remove the watermark position, which makes the watermark detection significantly more difficult. We leave robust correlation and tracing against these sophisticated countermeasures as an area for future work.

7.2.2 Tracing and Correlation of Non-Interactive Traffic

So far we have focused on tracing and correlating interactive intrusion connections, and there also appears a need to trace and correlate non-interactive traffic. For example, more and more voice and teleconferencing communication have shifted from circuit-switched network to packet-switched networks. This situation makes current wiretapping on circuit switched network insufficient for the law enforcement purposes and calls for a new capability of tracing and correlating (potentially encrypted) audio and video streams over IP network.

While we believe our techniques developed for tracing and correlating interactive traffic are useful in tracing and correlating non-interactive traffic, tracing and correlating non-interactive traffic have their special needs to meet to make them effective. The audio and video streams transported in IP network are very different from interactive traffic (such as telnet, SSH) in that 1) there are usually stringent real-time constraints; 2) they usually have very short and similar inter packet delays. The similar inter packet delay would make IPD-based correlation ineffective, and the stringent real-time constraint requires better real-time watermark embedding and decoding capability for the IPD watermarking correlation to be effective.

7.2.3 The Balance between Privacy and Traceability

While we have focused on the tracing intrusion connections through stepping stones, we recognize that there are concerns and needs for privacy. In fact, some of the sophisticated countermeasures against tracing and correlation have already been used in proposed anonymity systems (i.e. Onion Routing [53,58,73], Crowds [59], Anonymizer [3], NetCamo [33], Freenet [14]). It is our belief that no tracing system could be effective against all possible anonymity systems, and no anonymity system is able to provide absolute anonymity against all tracing systems. There appears some maximum achievable effectiveness for both tracing system and anonymity systems. An important open question is where the technically achievable balance and tradeoff between privacy and traceability are.

In term of watermarking and information hiding, there also exist open fundamental questions regarding invisibility and detectability:

- Does there exist an undetectable watermark or hidden information?
- Does there exist a universal detection method that could detect all embedded watermarks or hidden information?
- What is the fundamental tradeoff between the invisibility and capacity of hidden information?

Fred Cohen has demonstrated [16] that there is no algorithm that could detect all computer viruses, and David Chess and Steve White [13] have recently pointed out that there exist computer viruses that no algorithm can detect. We suspect similar theoretical results hold for hidden information detection.

7.2.4 Automatic Intrusion Response

We envision the intrusion source tracing as a building block toward an active, network-wide intrusion response infrastructure. Ideally, such an intrusion response infrastructure could automatically determine the sources of detected intrusions and take actions to "push-back" – contain and stop the attacks near their sources so that the adverse impact of network-based attacks could be minimized. Such an active defense framework show great promise in defending both intrusion and denial of service types of attacks, and would greatly improve the security of national critical information infrastructure.

There are still a number of open research problems need to be solved before such an active, network-wide intrusion response infrastructure becomes available. First, what kind of response is appropriate to certain detected attacks? Even with "perfect" IDS and intrusion source tracing, an "overreacted response" could potentially do more harm than the responded attack. Second, how should different (potentially far away) nodes trust each other? What should be the trust model? How much trust should be there? How do we prevent misuse or abuse of automatic intrusion response? Third, how do we protect privacy while supporting active intrusion tracing and response?

7.2.5 Tracing of Other Forms of Network-Based Attacks

In addition to traditional intrusion and denial of service attack, other forms of network-based attacks have become increasing threats to the normal function of networked systems. These include computer worm, virus and email spam.

While there has been active research on detecting these attacks, little attention has been paid so far to the tracing of these attacks. We believe these attacks could be much better contained by applying appropriate active attack response framework once effective tracing capability of these attacks has been developed. Therefore developing effective tracing capability for better containing these attacks is likely to be an area of fertile research, with promise of significant commercial impact.

7.2.6 Survivable System and Network

The known results that there exists no algorithm that can detect all possible computer virus and that there exists some computer virus that no algorithm can detect clearly indicate the fundamental limitation of virus detection. Similar limitation likely exists for detecting other form of network based attacks such as intrusion, worm and email spam.

One way to mitigate the fundamental limitation of attack detection is to develop survivable system and network. The goals of survivable system and network are 1) to be immune from malicious attacks; 2) to provide as much as possible functionality in the event of partial compromise; and 3) to automatically heal the partial compromise.

Bibliography

[1] R. Anderson. On the Limits of Steganography. In *IEEE Journal of Selected Areas in Communications*, Vol 16(4), Pages 474–482, 1998.

[2] R. Anderson. Stretching the Limits of Steganography. In *Proceedings of the 1st Information Hiding Workshop (IH'1996) LNCS-1174*, Pages 39–48, 1996.

[3] Anonymizer. http://www.anonymizer.com.

[4] S. M. Bellovin. ICMP Traceback Messages. Internet Draft: draft-bellovin-itrace-00.txt, March 2000.

[5] S. M. Bellovin. Using the Domain Name System for System Break-Ins. In *Proceedings of 5th USENIX Security Symposium*, 1995.

[6] W. Bender, D. Gruhl, N. Morimoto and A. Lu. Technique for Data Hiding. *IBM Systems Journal*, Vol. 35(3/4), Pages 313–336, 1996.

[7] S. Bhattacharjee, K. L. Calvert and E. W. Zegura. An Architecture for Active Networking. High Performance Networking (HPN'97), Pages 265–279, White Plans, NY, April 1997.

[8] H. Burch and B. Cheswick. Tracing Anonymous Packets to Their Approximate Source. In *Proceedings of 9th USENIX LISA*, 2000.

[9] K. L. Calvert, S. Bhattacharjee and E. Zegura. Directions in Active Networks. *IEEE Communication Magazine*, Vol. 36(10), Pages 72–78, 1998.

[10] R. H. Campbell, Z. Liu, M. D. Mickunas, P. Naldurg and S. Yi. Seraphim: Dynamic Interoperable Security Architecture for Active Networks. In *Proceedings of IEEE OPENARCH'2000*, March 2000.

[11] B. Carrier and C. Shields. A Recursive Session Token Protocol For Use in Computer Forensics and TCP Traceback. In *Proceedings of IEEE INFOCOM'02*, April 2002.

[12] H. Y. Chang, R. Narayan, S.F. Wu, B.M. Vetter, X. Wang M. Brown, J.J. Yuill, C. Sargor, F. Jou, F. Gong. DecIdUouS: Decentralized Source Identification for Network-Based Intrusions, In *Proceedings of 6th IFIP/IEEE International Symposium on Integrated Network Management*, Pages 701–714, 1999.

[13] D. M. Chess and S. R. White. An Undetectable Computer Virus. In *Virus Bulletin Conference 2000*. http://www.research.ibm.com/antivirus/SciPapers/VB2000DC.pdf.

[14] I. Clarke, O. Sandberg, B. Wiley and T.W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *Proceedings of International Workshop on Design Issues in Anonymity and Unobservability, LNCS–2009*, Pages 46–66, 2001.

[15] F. Cohen. Computational Aspects of Computer Viruses. Computers & Security, Vol. 8(4), Pages 325–344, 1989.

[16] F. Cohen. Computer Viruses: Theory and Experiments. *Computers & Security*, Vol. 6(1), Pages 22–35, 1987.

[17] Computer Emergency Response Team. CERT Advisory CA-96.21: CERT Advisory TCP SYN Flooding and IP Spoofing Attacks. http://www.cert.org/advisories/CA-96.21.tcp_syn_flooding.html, January 1998.

[18] Computer Emergency Response Team. CERT Advisory CA-96.26: Denial-of-Service Attack via Pings. http://www.cert.org/advisories/CA- 96.26.ping.html, December 1996.

[19] Computer Emergency Response Team. CERT Advisory CA-98.01: CERT Advisory "smurf" IP Denial-of-Service. http://www.cert.org/advisories/CA-98.01.smurf.html, January 1998.

[20] Computer Emergency Response Team. CERT Advisory CA-2000-01 Denial-of Service Development. http://www.cert.org/advisories/CA-2000-01.html, January 2000.

[21] Computer Emergency Response Team. CERT/CC Overview 2003 www.cert.org/present/cert-overview-trends/module-1.pdf

[22] Computer Emergency Response Team. CERT/CC Statistics 1988-2003 www.cert.org/stats/cert_stats_html

[23] Computer Emergency Response Team. Results of the Distributed-Systems Intruder Tools Workshop. http://www.cert.org/reports/dsit_workshop.pdf, Nov. 1999.

[24] I. J. Cox, M. L. Miller and J. A. Bloom. *Digital Watermarking*. Morgan-Kaufmann Publishers, 2002.

[25] P. B. Danzig and S. Jamin. tcplib: A Library of TCP Internetwork Traffic Characteristics. USC Technical Report, USC-CS-91-495.

[26] P. B. Danzig, S. Jamin R. Cacerest, D. J. Mitzel and E. Estrin. An Empirical Workload Model for Driving Wide-Area TCP/IP Network Simulations. In *Journal of Intenetworking*, Vol. 3(1), Pages 1–26. March 1992.

[27] D. Dean, M. Franklin and A. Stubblefield. An Algebraic Approach to IP Traceback. In *Proceedings of Network and Distributed System Security Symposium (NDSS'2001)*, 2001.

[28] M. H. DeGroot. *Probability and Statistics*. Addison-Wesley Publishing Company 1989.

[29] D. Donoho, A.G. Flesia, U. Shanka, V. Paxson, J. Coit and S. Staniford. Multiscale Stepping Stone Detection: Detecting Pairs of Jittered Interactive Streams by Exploiting Maximum Tolerable Delay. In *Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID'2002) LNCS-2516*, Pages 17–35, October, 2002.

[30] N.G. Duffield and M. Grossglauser. Trajectory Sampling for Direct Traffic Observation. In *Proceedings of the ACM SIGCOMM '2000*, Pages 271–282, September 2000.

[31] M. T. Goodrich. Efficient Packet Marking for Large-Scale IP Traceback. In *Proceedings of 9th ACM Conference on Computer and Communication Security (CCS'2002)*, Pages 117–126, October 2002.

[32] M. B. Greenwald, S. K. Singhal, J. R. Stone and D. R. Cheriton. Design an Academic Firewall: Policy, Practice and Experience with SURF. In *Proceedings of Internet Society Symposium on Network and Distributed System Security (NDSS '1996)*, February 1996.

[33] Y. Guan, X. Fu, D. Xuan, P. Shenoy, R. Bettati, and Wei Zhao. NetCamo: Camouflaging Network Traffic for QoS-Guaranteed Mission Critical Applications, in *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 31(4) 2001

[34] L. T. Heberlein, K. Levitt and B. Mukherjee. Internetwork Security Monitor: An Intrusion-Detection System for Large-Scale Networks. In *Proceedings of 15th National Computer Security Conference*, 1992.

[35] J. D. Howard. An Analysis of Security Incidents on The Internet 1989 - 1995, PhD Thesis, http://www.cert.org/research/JHThesis/Start.html, April 1997.

[36] J. Ioannidis and S. M. Bellovin. Implementing Pushback: Router-Based Defense against DDoS Attacks. In *Proceedings of Internet Society Symposium on Network and Distributed System Security (NDSS '2002)*, 2002.

[37] J. Ioannidis and M Blaze. The Architecture and Implementation of Network-Layer Security under Unix. In *Proceedings of 4th USENIX Security Symposium*, 1993.

[38] S. Ioannidis, A. D. Keromytis, S. M. Bellovin, and J. M. Smith. Implementing a Distributed Firewall. In *Proceedings of 7th ACM Conference on Computer and Communication Security (CCS'2000)*, October 2000.

[39] W. Jansen, P. Mell, T. Karygiannis, D. Marks. Applying Mobile Agents to Intrusion Detection and Response. NIST Interim Report (IR) – 6416, October 1999.

[40] H. Jung, et al. Caller Identification System in the Internet Environment. In *Proceedings of 4th USENIX Security Symposium*, 1993.

[41] S. Kent, R. Atkinson. *Security Architecture for the Internet Protocol*. IETF RFC 2401, September 1998.

[42] G. Kramer. Generator of Self-Similar Network Traffic. http://wwwcsif.cs.ucdavis.edu/~kramer/code/trf_gen2.html.

[43] S. Lee and C. Shields. Tracing the Source of Network Attacks: A Technical, Legal, and Social Problem. In *Proceedings of the Second Annual IEEE Systems, Man, and Cybernetics Information Assurance Workshop*, June 2001.

[44] L. H. Lehman, S. J. Garland, and D. Tennenhouse. Active Reliable Multicast. In *Proceedings of IEEE INFOCOM '1998*, April 1998.

[45] Mark Linehan. Comparison of Network-Level Security Protocols. IBM T.J. Watson Research Center, June 1994.

[46] M. Machover. *Set Theory, Logic and Their Limitations*. Cambridge University Press 1996.

[47] G. Mansfield, K. Ohta, Y. Takei, N. Kato, Y. Nemoto. Towards Trapping Wily Intruders in the Large. *Computer Networks*, Vol. 34(2000), Pages 659–670, 2000.

[48] J. P. Mayberry. *The Foundations of Mathematics in the Theory of Set*. Cambridge University Press 2000.

[49] S. Mittra, T. Y. Woo. A Flow-Based Approach to Datagram Security. In *Proceedings of the ACM SIGCOMM '1997*, September 1997.

[50] B. Mukherjee, L.T. Heberlein, and K.N. Levitt. Network Intrusion Detection, *IEEE Network*, Vol. 8, No. 3, 1994.

[51] S. Murphy et al. Security Architecture for Active Nets. AN Security Working Group, July 15, 1998.

[52] NLANR Trace Archive. http://pma.nlanr.net/Special/.

[53] Onion Routing. http://www.onion-routing.net.

[54] OpenSSH. http://www.openssh.com.

[55] Rolf Oppliger. Internet Security: Firewalls and Beyond. *Communications of the ACM,* Vol. 40(5), Pages 92–102, 1997.

[56] K. Park and H. Lee. On the Effectiveness of Probabilistic Packet Marking for IP Traceback under Denial of Service Attack. In *Proceedings of IEEE Infocom'2001*, Pages 338–347, 2001.

[57] T. H. Ptacek and T. N. Newsham. Insertion, evasion, and denial of service: Eluding network intrusion detection. Technical Report, Secure Networks, Inc., January 1998.

[58] M. G. Reed, P. F. Syverson, D. M. Goldschlag. Anonymous Connections and Onion Routing. In *IEEE Journal on Selected Areas in Communication*, Vol. 16(4), Pages 482–494, 1998

[59] M. K. Reiter and A. D. Rubin. Crowds: Anonymity for Web Transactions. In *ACM Transactions on Information and System Security (TISSEC)*, Vol. 1(1), Pages 66–92, 2000.

[60] S. Savage, D. Wetherall, A. Karlin and T. Anderson. Practical Network Support for IP Traceback. *Proceedings of the ACM SIGCOMM '2000*, Pages 295–306, September 2000.

[61] D. Schnackenberg. Dynamic Cooperating Boundary Controllers. http://www.darpa.mil/ito/Summaries97/ E295_0.html, Boeing Defense and Space Group, March 1998.

[62] D. Schnackenberg, K. Djahandari, and D. Strene. Infrastructure for Intrusion Detection and Response. In *Proceedings of DISCEX*, 2000.

[63] B. Schwartz, A. W. Jackson, W. T. Strayer, W. Zhou, R. R. Rockwell and C. Partridge. Smart Packets for Active Networks. In *Proceedings of IEEE OPENARCH '1999*, April 1999.

[64] V. Shmatikov. Probabilistic Analysis of Anonymity. In *Proceedings of the 15th IEEE Computer Security Foundations Workshop*, Pages 119−128, 2002

[65] S. Snapp, et all. DIDS (Distributed Intrusion Detection System) – Motivation, Architecture and Early Prototype. In *Proceedings of the 14th National Computer Security Conference*, Pages 167−176, 1991.

[66] A. C. Snoeren, C. Partridge, L. A. Sanchez and C. E. Jone, F. Tchakountio, Stephen T. Kent and W. T. Strayer. Hash-Based IP Traceback. In *Proceedings of the ACM SIGCOMM '2001*, Pages 3−14, November 2001.

[67] D. Song and A. Perrig. Advanced and Authenticated Marking Scheme for IP Traceback. In *Proceedings of IEEE INFOCOM'2001*, Pages 878−886, April 2001.

[68] D. Song, D. Wagner and X. Tian. Timing Analysis of Keystrokes and Timing Attacks on SSH. In *Proceedings of 10th USENIX Security Symposium*, 2001.

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

[70] W. R. Stevens. *TCP/IP Illustrated Volume 1 The Protocol*. Addison-Wesley Publishing Company, 1994.

[71] C. Stoll. *The Cuckoo's Egg: Tracking Spy through the Maze of Computer Espionage*. Pocket Books, October 2000.

[72] R. Stone. Centertrack: An IP Overlay Network for Tracking DoS Floods. In *Proceedings of 9th USENIX Security Symposium*, 2000.

[73] P. F. Syverson, D. M. Goldschlag and M. G. Reed. Anonymous Connections and Onion Routing. In *Proceeding of the 1997 IEEE Symposium on Security and Privacy*, Pages 44–54, 1997

[74] M. S. Taqqu, W. Willinger, and R. Sherman. Proof of a Fundamental Result in Self-Similar Traffic Modeling. *ACM Computer Communication Review*, Vol. 27, Pages 5–23, 1997.

[75] D Tennenhouse and D Wetherall, Towards an Active Network Architecture. In *SPIE Proceedings of Conference on Multimedia Computing and Networking1996*, January 1996.

[76] V. C. Van. A Defense against Address Spoofing Using Active Networks. Master's thesis, Department of Electrical Engineering and Computer Science, MIT, May 1997.

[77] X. Wang. The Loop Fallacy and Serialization in Tracing Intrusion Connections through Stepping Stones. In *Proceedings of the 19th ACM Symposium on Applied Computing (SAC'2004)*, March 2004.

[78] X. Wang. Survivability through Active Intrusion Response. In *Proceedings of the 3rd IEEE Information Survivability Workshop (ISW-2000)*, Pages 173−176, October 2000.

[79] X. Wang, D. S. Reeves. Robust Correlation of Encrypted Attack Traffic through Stepping Stones by Manipulation of Interpacket Delays. In *Proceedings of the 10th ACM Conference on Computer and Communications Security (CCS'2003),* Pages 20–29, October 2003.

[80] X. Wang, D. S. Reeves and S.F. Wu. Inter-Packet Delay-Based Correlation for Tracing Encrypted Connections through Stepping Stones. In *D. Gollmann, G. Karjoth and M. Waidner, editors, 7th European Symposium on Research in Computer Security (ESORICS'2002) LNCS-2502,* Pages 244–263, October 2002.

[81] X. Wang, D. S. Reeves and S. F. Wu. Tracing Based Active Intrusion Response. In *Journal of Information Warfare*, Vol. 1(1), Pages 50–61, October 2002.

[82] X. Wang, D. S. Reeves, S. F. Wu and J. Yuill. Sleepy Watermark Tracing: An Active Network-Based Intrusion Response Framework. In *Proceedings of 16th International Conference on Information Security (IFIP/Sec'01)*, Pages 369–384, June, 2001.

[83] Webopedia. http://www.webopedia.com/

[84] D. Wetherall, J. Guttag and D. Tennenhouse. ANTS: A Toolkit for Building and Dynamically Deploying Network Protocols. In *Proceedings of IEEE OPENARCH '1998*, April 1998.

[85] G. White and V. Pooch. Cooperating Security Managers: Intrusion Detection Systems. *Computer & Security*, Vol. 16, No. 5, 1996.

[86] S. F. Wu. Sleepy Network-Layer Authentication Service for IPSEC. In G. Martella E. Bertino, H. Kurth and E. Montolivo, editors, 4th European Symposium on Research in Computer Security (ESORICS'1996) LNCS-1146, Pages 146–159, September 1996.

[87] T. Ylonen, et al. SSH Protocol Architecture. *IETF Internet Draft: draft-ietf-secsh-architecture-12.txt*, July 2001.

[88] K. Yoda and H. Etoh. Finding a Connection Chain for Tracing Intruders. In F. Guppens, Y. Deswarte, D. Gollmann and M. Waidner, editors, 6th European Symposium on Research in Computer Security (ESORICS'2000) LNCS-1895, Pages 191–205, October 2000.

[89] K.H. Yung. Detecting Long Connection Chains of Interactive Terminal Sessions. In *Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID'2002) LNCS-2516*, Pages 1–16, October, 2002.

[90] Y. Zhang and V. Paxson. Detecting Stepping Stones. In *Proceedings of 9th USENIX Security Symposium*, 2000.