WADeS: A TOOL FOR DISTRIBUTED DENIAL OF SERVICE ATTACK DETECTION

A Thesis

by

ANU RAMANATHAN

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

August 2002

Major Subject: Computer Engineering

WADeS: A TOOL FOR DISTRIBUTED DENIAL OF SERVICE ATTACK DETECTION

A Thesis

by

ANU RAMANATHAN

Submitted to Texas A&M University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Approved as to style and content by:

A.L. Narasimha Reddy (Co-Chair of Committee)	Marina Vannucci (Co-Chair of Committee)
Pierce Cantrell (Member)	Riccardo Bettati (Member)
Chanan Singh (Head of Department)	

August 2002

Major Subject: Computer Engineering

ABSTRACT

WADeS: A Tool for Distributed Denial of Service Attack Detection. (August 2002)

Anu Ramanathan, B.Tech., Indian Institute of Technology, Madras

Co–Chairs of Advisory Committee: Dr. A.L. Narasimha Reddy
Dr. Marina Vannucci

The increasing popularity of web-based applications has led to several critical services being provided over the Internet. This has made it imperative to monitor the network traffic so as to prevent malicious attackers from depleting the network's resources and denying service to legitimate users. In our research work, we propose WADeS (Wavelet based Attack Detection Signatures), an approach to detect a Distributed Denial of Service Attack using Wavelet methods. We develop a new framework that uses LRU cache filtering to capture the high bandwidth flows followed by computation of wavelet variance on the aggregate miss traffic. The introduction of attack traffic in the network would elicit changes in the wavelet variance. This is combined with thresholding methods to enable attack detection. Sampling techniques can be used to tailor the cost of our detection mechanism. The mechanism we suggest is independent of routing information, thereby making attack detection immune to IP address spoofing. Using simulations and quantitative measures, we find that our mechanism works successfully on several kinds of attacks. We also use statistical methods to validate the results obtained.

To my Parents

ACKNOWLEDGMENTS

I am greatly indebted to my advisor, Dr.Reddy for having given me an opportunity to work with him. I would like to thank him for the invaluable guidance and encouragement which he has provided throughout my research. Weekly meetings to discuss research were highly instrumental in keeping the research focussed. The calm with which he handled obstacles we faced during the course of research really helped a lot in keeping my enthusiasm high. He was always open to new ideas, and I really appreciate the freedom he gave me while working on my research.

I would like to express my sincere appreciation to Dr.Vannucci for her guidance, in the field of Wavelet Analysis and constant encouragement. I would also like to thank her for allowing us to use her matlab codes.

I thank Dr.Cantrell and Dr.Bettati for their interest in this research and invaluable suggestions.

I would like to thank Pravi and Umang for patiently listening to my ideas during various stages of my research.

Last, but not the least, I'd like to thank my parents. But for their absolute confidence in me, I would have not been able to pursue my M.S degree or this research. They were, are and will always be the source of inspiration in all my endeavors.

TABLE OF CONTENTS

CHAPTER Page

LIST OF TABLES

TABLE Page

LIST OF FIGURES

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

A *Denial of Service* (DoS) Attack is defined as an explicit attempt by a malicious user to consume the network's resources, thereby preventing legitimate users from using the services provided by the network. The most common DoS attacks typically involve flooding with a huge volume of traffic, thereby consuming limited resources such as the network's bandwidth, packet buffers at the routers, CPU time and memory cycles of the target server. Some of the common DoS attacks are SYN Flooding, UDP Flooding, DNS based Flooding, ICMP Directed Broadcast, Ping Flood Attack, IP Fragmentation and CGI Attacks. Based on the number of attacking machines deployed to implement the attack, DoS nes deployed to implement the attack, DoS attacks are classified under two broad categories - a single intruder consumes all the available bandwidth by generating a large number of packets operating from a single machine, or the distributed case where multiple attackers coordinate together to produce the same effect from several machines on the network. The latter is referred to as a *Distributed Denial of Service* (DDoS) Attack and owing to its distributed nature, adds a "many to one" [1] dimension to the attacks, thereby making them difficult to detect. It has become highly critical to be able to successfully detect such attacks as quickly as possible.

A. Overview of Denial of Service Attacks

A *Denial of Service* (DoS) Attack may consume network's resources such as bandwidth, packet buffers, CPU cycles and memory of the target server. We now focus

---

The journal model is *IEEE Transactions on Automatic Control.*

our attention on Bandwidth Attacks and discuss the ways in which a packet flood can be launched.

SYN Flood Attack: This attack exploits the working mechanism of the TCP protocol and is one of the most common attacks that is used to consume the resources of the target server. A TCP connection is established using a 3-way handshake. First, a SYN request is sent from the attacking host to the target server using a spoofed source address. The server acknowledges the request with a SYN-ACK, allocates the data structures for the connection and stores the information for connections waiting to be completed in the connection queue. It waits for the sender of the SYN-request to acknowledge the SYN-ACK. Once this is done, the connection is removed from the connection queue and the connection is established. But since the source IP address is a spoofed one, the last stage of the 3-way handshake is never completed. Thus, the information for the connection is stored until the end of a certain time-out interval and is removed. Since each server has memory limitations that restrict the number of connections waiting to be completed, an attacker who sends SYN-requests with forged source IP addresses at a high rate can keep the connection queue filled with such requests, therefore denying legitimate requests to the target server.

UDP Flood Attack: Unlike TCP which is connection-oriented, UDP is a connectionless protocol. This can effectively be used to launch an attack by sending a huge volume of traffic with spoofed source addresses to one of the ports of the target server. The target processes the packets to soon realize that these are not valid requests for services. Thus, in effect the CPU cycles of the target server have been wasted in the process, and by sending a very large volume of such spoofed traffic, it is possible to bring down the server by overloading it with requests. Moreover, since UDP does not have congestion control, it would tend to hog the network bandwidth as its sending rate increases and can force legitimate users who use TCP which has

congestion control and is forced to lower the sending rate, in effect getting denied the fair share of service.

Smurf Attack: The attacker sends ICMP echo packets to broadcast addresses of vulnerable networks. This causes all the hosts on the network to issue ICMP echo replies, which eventually causes a huge volume of traffic in the network and tends to consume network bandwidth.

DNS Zone Transfer based Flooding: A Zone Transfer request involves requesting the Name Server information about the entire zone. It is typically used by a secondary server to update its own information by sending such a request to a primary server. When repeated requests are sent to the name server by an attacker, this ends up consuming a significant portion of the network's bandwidth due to the huge volume of data transferred for each request, which can successfully enable eating up the network's bandwidth.

Ping based Attacks: This is commonly referred to as the "Ping of Death" or "Long ICMP" attack where an attacking host sends a huge volume of ICMP echo requests to the target server. The attacker deliberately uses packets of size larger than 65536 bytes, the maximum packet size that can be handled by the IP protocol. This typically results in fragmentation and consumes both CPU cycles and network's bandwidth.

CGI Attacks: Execution of CGI scipts is very time-consuming on a processor. Thus, by repeatedly requesting a large number of CGI scripts to be executed, it is possible for an attacker to slow down the target server.

Fragmentation Attack: IP restricts the maximum packet size in a network by fragmenting packets that are of size larger than the Maximum Transfer Unit (MTU). By sending a large volume of packets with packet sizes larger than the MTU to a target server, an attacker can consume a large portion of the server's CPU cycles in

fragmenting and reassembling of the packets.

Thus, all the above attacks aim to slow down the network's functioning and in extreme cases could cause a network shutdown. These attacks may be launched in a distributed fashion making it still tougher to identify an attack. It becomes essential to study the behavior of network traffic, so that we can successfully isolate an attacker and prevent damage to the network. In the next section, we discuss the various models used to characterize network traffic behavior and how they can be employed as an effective tool for DoS attack detection.

B. Characteristics of Network Traffic

In a wide range of situations, network traffic is observed to be comprised of traffic bursts which are contributed typically by a few high bandwidth connections, which dominate over the others [2],[3]. Attacks are characterized by a large number of packets over a small duration of time.

Fair Queueing[4] and its variants have been shown to isolate individual flow behavior while providing fair service. While these mechanisms may be useful for flow isolation (and hence containment of attacks), current core routers over the Internet typically handle millions of flows, making it difficult to maintain state information for all of them. Caching mechanisms coupled with queue management as in LRU-RED [2] have been suggested to solve this problem by maintaining state only for those flows which consume network resources heavily. These mechanisms typically employ a limited amount of state and an appropriate state management algorithm to monitor the few dominant flows in the traffic. It is possible to then contain these flows (for which state is maintained) by appropriate control mechanisms that treat the "cached flows" differently from the rest of the aggregate traffic [2],[5],[6],[7],[8],[9].

For example, we could use Fair Queueing with two queues (one for cached traffic and the second one for "stateless" traffic) to limit the cached traffic from taking more than 40 % of the link bandwidth. Hence, these partial state mechanisms can contain attacks staged from fewer sources than that for which state is maintained at the router.

This poses an intriguing question regarding how to detect or curtail a DDoS attack as the number of attacking hosts increases. Moreover, an additional requirement for attack detection is the need for facilitating detection of an attack as quickly as possible to minimize the debilitating effect left by the attacker on the target server or the network.

Traffic models are highly invaluable resources for identifying the characteristics of network traffic. Several network models proposed so far have argued that aggregate network traffic is highly self-similar in nature at all time-scales [10]. This attributes Long Range Dependence (LRD) to the aggregate traffic. But it has been observed that network traffic is highly bursty in nature, thereby not adhering to the Fractional Gaussian (fGn) Model. The burstiness results in non-Gaussian marginal distributions, which has been demonstrated using the Multifractal Wavelet Model [11]. It has been shown that Wavelet-based scaling analysis can be used to characterize Internet traffic [12]. Therefore, the scaling properties of wavelets can be effectively tapped to capture the variations in behavior of network traffic during an attack. These are referred to as *Wavelet Signatures* and are extremely helpful in Denial of Service Attack Detection. In our research, we will attempt to use Wavelet methods to detect a DDoS attack.

C. Related Work

Several mechanisms have been proposed to solve the problem of Denial of Service Attacks. Most of these focus on attempting to traceback the source of an attack or employ mechanisms to prevent attacks. These may be deployed with WADeS, thereby augmenting the reactivess of the attack containing process.

Network Ingress Filtering [13] is one of the mechanisms suggested to prevent attacks that are staged using spoofed source addresses. This involves configuring the routers to drop packets that have illegitimate source IP addresses. This is implemented at the edge routers by the Internet Service Providers (ISP). One of the serious pitfalls of this method is its inability to curtail a flood attack that originates with a spoofed IP address from within the network. Another disadvantage is the complexity involved in demarcation of addresses, especially when there is aggregation of traffic from different networks over transit networks.

ICMP Traceback [14] messages are useful to identify the path taken by packets through the Internet. This requires a router to use a very low probability with which traceback messages are sent along with the traffic. Thereby, with sufficiently large number of messages, it is possible to determine the route taken by the traffic during an attack, thus enabling localization of the attacking host.

Another alternative to overcome the problems associated with ascertaining the validity of IP addresses in Ingress filtering suggested the use of routing information instead of just the source address. IP Traceback [15] proposes a reliable way to perform hop by hop tracing of a packet to the attacking source from where it originated. However, this requires coordinated effort from all the routers in the network along the path from the victim to the attacker, and examination of the packet logs to be able to localize the attacking source.

Deterministic Packet Marking (DPM) [16], another solution to detect DoS attacks, heavily relies on routing information which is inscribed in the packet header by the routers as the packet traverses the network. This approach leads to an increase in the size of the IP packet header, which increases linearly with the number of hops traversed, thereby resulting in variable header sizes which eventually increases the complexity of processing at the routers.

Probabilistic Packet Marking (PPM) for IP Traceback [17] suggested by Kihong Park et al. is a solution that attempts to improve DPM. This works to eliminate IP address spoofing by allowing each router to probabilistically inscribe local path information onto a packet that traverses it, thereby enabling a victim or destination to be able to localize the attacking source despite retaining fixed packet header sizes. This relies on route stability between the attacker and the victim. The increase in distributed nature of an attack makes it more difficult to localize the attacker. Another architecture suggested is Route-Based packet filtering [18]. This seeks to improve on the following weaknesses of the IP Traceback mechanism with PPM: Reactiveness - as traceback is able to relate to attack information only after the DoS attack has adversely affected the network, Scalability with increase in number of attacking hosts, Overhead involved in inscribing routing information in the packet header. This packet filtering mechanism uses the source and destination addresses on a packet header and ascertains the validity of the route taken by the packet using the reachability constraints which are imposed by the network topology. Multipath routing can easily evade this mechanism and design of the efficient filters involves high complexity.

Thomer et al. propose MULTOPS [19], where routers detect bandwidth attacks using a heuristic based on packet sending rates. Under non-attack circumstances, the packet rate in one direction over the Internet is directly proportional to the packet rate

of the traffic going in the opposite direction. Thus, when this is violated, an attack is detected. However, due to memory limitations, efficiency of MULTOPS degrades with randomized IP source addresses. This might result in "collateral damage"[19] as the packet-drop policy might adversely affect the non-attack traffic.

D. Motivation

Most of the methods employed so far work based on routing information and can be classified under two extremes. A large majority of them focus on detecting the source of the attack once it has occurred, while at the other end there are mechanisms that seek to use proactive measures to prevent attacks. We focus on detection of Distributed Denial of Service Attacks on the network's bandwidth and seek to identify an attack as close as possible to the origin of the attack, so as to minimize the damage caused to the network. Our framework, WADeS, employs a simple LRU filtering mechanism coupled with wavelet methods to detect a DDoS attack.

E. Organization

The rest of the thesis is organized as follows. In Chapter 2, we provide a detailed discussion of the design of WADeS. Chapter 3 shows the effectivess of WADeS using several simulations in addition to a discussion of the quantitative measures developed. In Chapter 4, we discuss techniques that can be employed to fine-tune the cost of deploying the method. In Chapter 5, we use statistical methods to validate the heuristics developed in the previous chapter. Chapter 6 concludes with a note on contributions of this work and directions for future work.

CHAPTER II

DESIGN OF WADES

A. Overview

Network Traffic can be classified into long-term and short-term flows. Our earlier work [2],[5],[6],[7],[8],[9] has shown that partial state mechanisms can be employed to effectively separate the long-term and short-term flows. Here, once such a separation is made, a wavelet transform of the traffic comprised of the short-term flows (or the miss traffic) is computed. The wavelet transform of this traffic normally shows no correlation over long-term time-scales. When a DoS attack is staged with many flows, the available state at the router is not enough to filter out all the long-term flows in the traffic. Hence, the miss traffic consists of the short-term flows along with some of the long-term flows. When a wavelet transform is computed over the miss traffic (with a mixture of short-term and long-term flows), we observe a correlation over long-term time-scales. The key idea of our method is to use the observed differences in wavelet signatures to detect a DDoS attack.

In this approach, the amount of (partial) state (or the cache size) determines the minimum number of flows required to stage a DDoS attack, since an attack with fewer flows can be contained through fair queueing mechanisms over cached/non-cached traffic as shown in Fig. 1 where we use a Weighted Fair Queueing Scheduler (WFQ) [20], with w_1 and w_2 as the weights to compute the bandwidth being allocated to the cached and non-cached flows.

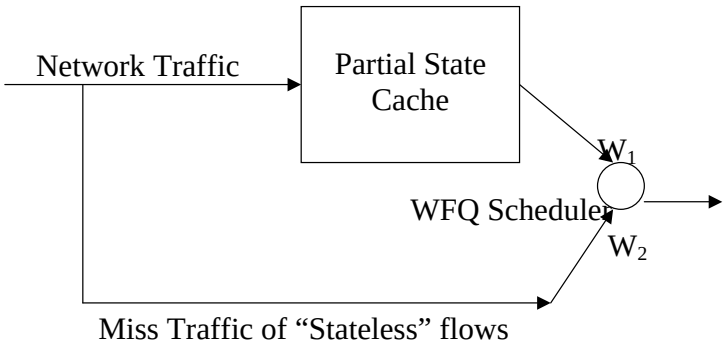

Fig. 1. Scheme for containing DoS attacks from a few flows

B. Wavelet Approach

Wavelets are excellent mathematical tools for analyzing the properties of data series [21],[22]. Wavelets split up data into different frequency components and each component is studied at a certain resolution corresponding to its time-scale. Conventional methods for signal analysis such as Fourier Transforms work in the frequency domain. This is useful when signals do not possess transitory characteristics. But often, in real situations, we encounter signals which are characterized by abrupt changes. In such cases, it becomes essential to be able to relate to the occurrence of an event in time, which makes frequency-domain based analysis unsuitable. Since wavelets work on the time domain, they serve as invaluable tools in analyzing such signals.

In our research, we employ changes in wavelet variance to detect a DDoS attack. We now give a brief introdution to the method used for computing the wavelet variance of a time series. Given a uniformly spaced signal $X_0, X_1, ..., X_{N-1}$, whose d^{th} order backward differences form a stationary process, we can compute the wavelet coefficients $\tilde{W_{j,t}}$ using a Daubechies wavelet filter of width $L \geq 2d$ as shown in Eq. 2.1. Here, $\tilde{h_j}$ are the Daubechies filter coefficients. Wavelet coefficients can be used to

compute the wavelet variance $\nu^2(\tau_j)$[23] at a certain time-scale τ_j, which is shown in Eq. 2.2

$$\tilde{W_{j,t}} \equiv \sum_{l=0}^{L_j-1} \tilde{h_{j,l}} X_{t-lmodN}, t = 0, 1, 2, .., N-1 \tag{2.1}$$

$$\nu^2(\tau_j) \equiv \frac{1}{N} \sum_{t=0}^{N-1} \tilde{W_{j,t}}^2, \text{where } \tau_j \equiv 2^{j-1} \tag{2.2}$$

We now describe in detail out proposed scheme for DDoS attack detection.

C. Proposed Mechanism

The overall functional overview of the mechanism is shown in Fig. 2

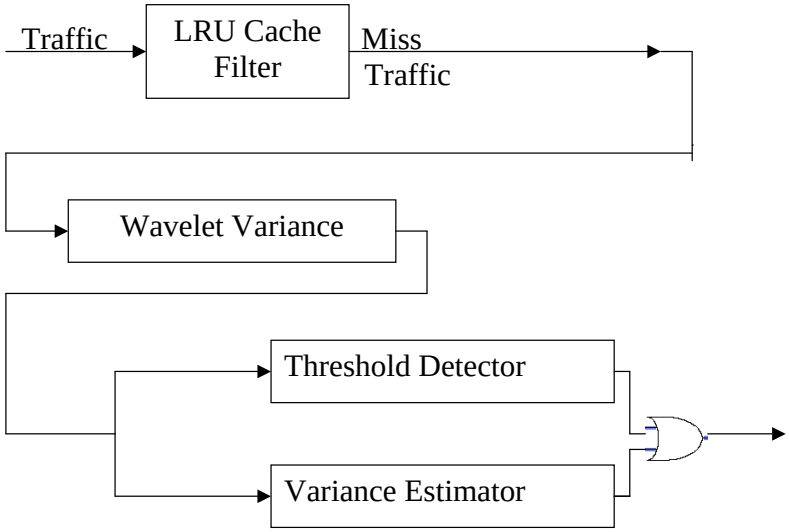

Fig. 2. Schematic: Overview of WADeS

This first stage of the framework comprises of a Least Recently Used (LRU) Cache. This filter is based on a connection-level approach to traffic analysis. A flow is defined by a particular source and destination address pair. To obtain an overall

idea regarding attacking hosts, we ignore the source and destination ports for the purpose of identification of the attack.

Upon arrival of a packet, the cache is checked to see if it already has an entry corresponding to the flow. If the flow is in cache, then the corresponding entry for the flow is upgraded in the cache by moving it up the cache and the packet count for the flow is updated. If the flow is not in cache and if the cache is cold, then a new entry is created for the incoming flow and is placed on top of the cache. If the cache is already warm, then the oldest flow in cache, i.e the bottom-most flow entry in the cache is removed with a predetermined probability, p [2]. If a flow replaces an entry in cache, an entry is created at the top of the cache for the corresponding flow and its packet-count is initialized. Thus, a flow that does not send packets frequently enough, will eventually degrade in position in the cache and keep moving down with the arrival of more incoming flows and eventually will be removed from the cache. This imposes a logical classification on the aggregate traffic that is a result of the filtering operation into two classes - the hit and the miss traffic. This is done by using a threshold, T for the packet-count, and those flows which enter the cache and have a packet-count that crosses the threshold, constitute the hit traffic, and the remaining flows constitute the miss traffic. High bandwidth traffic will show up more often at the router and will therefore end up getting cached and will appear in the hit traffic, while short term flows like HTTP flows will contribute to the miss traffic. Thus, the information in the cache may be used to determine the long-term high bandwidth flows.

If we use a cache of suitable size S, then we can effectively filter the high-bandwidth flows. In the event of a DDoS attack comprised of N new flows which coordinate to launch an attack, where N is comparable to S, then this will result in some of these attack flows entering the cache, thereby tending to displace some of the long-term flows which were already in cache, pushing them from the hit traffic onto

the miss traffic. Thus, observing the changes in the behavior of the miss traffic would provide us useful information regarding the attack.

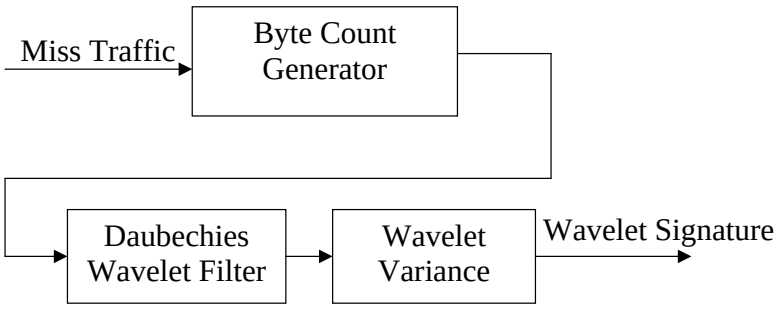

Fig. 3. Wavelet variance computations on aggregate miss traffic

The LRU filtering operation is followed by a wavelet variance computation of the aggregate miss traffic. The introduction of attack traffic would cause changes in the wavelet variance. Therefore, observing changes in the wavelet variance across time-scales, would help us to identify the start of an attack. A block diagram for the Wavelet Variance Computation procedure is given in Fig. 3.

We compute the wavelet variance of the aggregate miss traffic by using uniformly spaced samples of byte counts of the traffic as the time series. Wavelet Signature is being computed on the byte counts of the aggregate traffic at this stage and there is no flow-level analysis. The wavelet variance of a fixed number of samples, referred to as a window is computed and the window slides forward in time. The resulting wavelet variance fluctuations are observed at each time-scale. The introduction of the attack traffic introduces sudden rises and falls in the wavelet variance, which is noticed by sliding the window over time. This is further passed through an Attack Detection filter, which uses quantitative measures to identify an attack.

D. Implementation Issues

We observe that the implementation of the above algorithm plays a vital role in deciding the performance of attack detection. This is primarily because we are targetting at detecting an attack as close to the origin of the attack as possible. Therefore, we need to ensure that the signature processing time does not form a bottleneck. The LRU Cache searching mechanism is implemented using a hash table to determine if a flow is present in cache or not. This makes the LRU mechanism's processing cost as O(1) for each packet. The cost of computation of the wavelet variance is governed by the computational efficiency of the method that is used to calculate the wavelet coefficients. Typically discrete wavelet transform computations take $O(N^2)$ time, for a time series consisting of N samples. But this can be improved using Pyramid Algorithms [22] that take O(N) time, where N is the number of samples used in computing the signature. By choosing an appropriate sampling interval, we can ensure several packets arrive at the router within each sample. The total byte count of these packets is issued as one sample in signature computation. Thus, the processing cost for our mechanism is O(1) per packet. The cost of implementation can be lowered still further, which is discussed later. In our experiments, we use a sampling interval of 0.78ms and we compute the wavelet variance of 256 samples at a time in a single window. We reduce this sampling interval still further for implementations with lesser amount of computation and is discussed in detail later. The Daubechies Filter used for computation of wavelet coefficients has $L = 4$. We use Matlab's Wavelet Toolbox for carrying out the wavelet computations. In the next chapter, we present the results obtained from simulations.

CHAPTER III

SIMULATION RESULTS

In our research, we use Internet packet traces obtained from http://pma.nlanr.net website to test the effectiveness of our proposed scheme. These measurements were carried out using a PMA monitor which is connected between NYSERnet's router and Buffalo's router [24] (OC3 Packet over Sonet (PoS)), via a fiber tap. The traces provide information regarding the source, destination IP addresses, the packet size and the time stamps with a resolution of 1 μs. The OC3 link speed is 155 Mbps. First, we provide information about the traffic on which we simulated the attacks. We refer to this traffic as *Background Traffic*.

A. Details of Background Traffic

The details of the constituent flows in the background traffic for one of the traces is summarized in Table I. There are totally 3192 flows. The aggregate sending rate is 74.74 Mbps. The traffic is observed for a duration of 43.757932 s. On an average, each flow sends 128KB during this interval. We observe that most of the flows are short-term in nature and have low sending rates, while the dominant or high bandwidth flows are few in number. We notice that a major portion of the network's bandwidth is consumed by less than 100 flows. The sampling interval chosen is 0.78 ms and a window size of 256 is used for computing the wavelet variance.

When there is no filtering used, the wavelet variance of the traffic is as shown in Fig. 4. The wavelet variance plot with LRU cache filtering in the absence of an attack is shown in Fig. 5.

We observe that the wavelet variance is more or less fluctuating around the mean

Table I. Trace-1: Summary of bytes sent per flow in the background traffic

Data sent (in MB)	No: of flows
0-0.5	3078
0.5-1	21
1-2	41
2-10	45
≥ 10	7

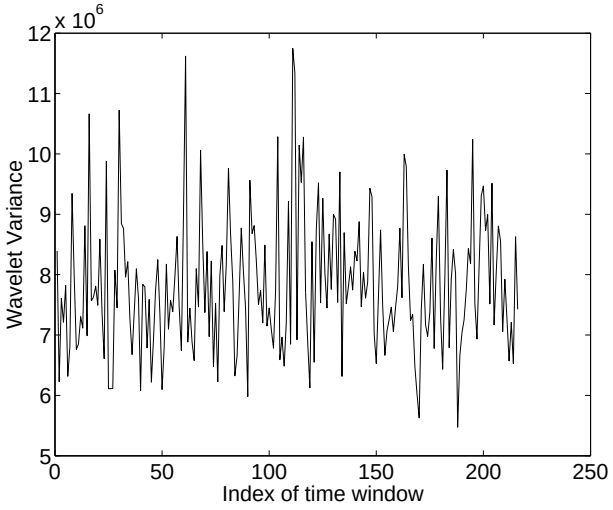

Fig. 4. Trace-1: Wavelet variance of aggregate traffic without filtering, scale-1, cache size-100, (0.78 ms)

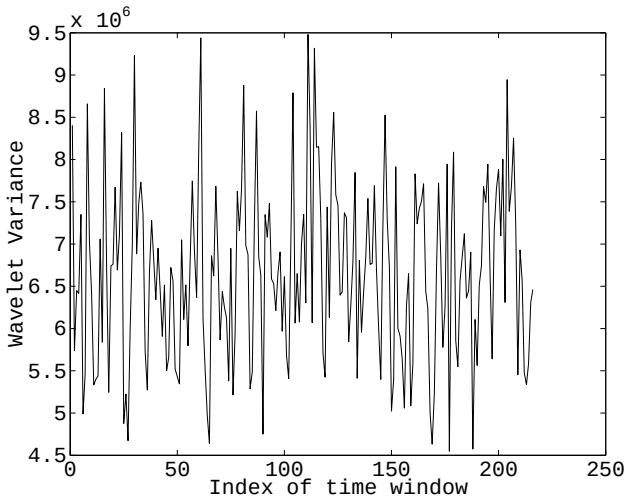

Fig. 5. Trace-1: Wavelet variance of aggregate miss traffic, cache size-100, scale-1 (0.78 ms)

value, without any sharp rises and falls, thereby indicating that the long-term flows have been effectively filtered by the cache, as opposed to Fig. 4 where there are steep fluctuations in the value of wavelet variance from the mean. The cached or long-term flows are responsible for this behavior and this is clearly seen in Fig. 6 which shows the wavelet variance of the cached flows. We find that there are sharp changes in wavelet variance in the aggregate hit traffic as opposed to the aggregate miss traffic, which clearly explains the need for filtering to remove the long-term flows. This is necessary here as we are using the fluctuations in wavelet variance for attack detection, therefore filtering effectively removes the fluctuations in the absence of an attack.

We simulate several kinds of attacks on this background traffic namely - CBR Spread Out Attack, CBR Back-to-Back Attack, VBR Spread Out Attack, VBR Back-to-Back Attack and Randomly Generated Attack, TCP Attack. The first four attacks are launched in 2 different ways - first case where we superimpose attack traffic over

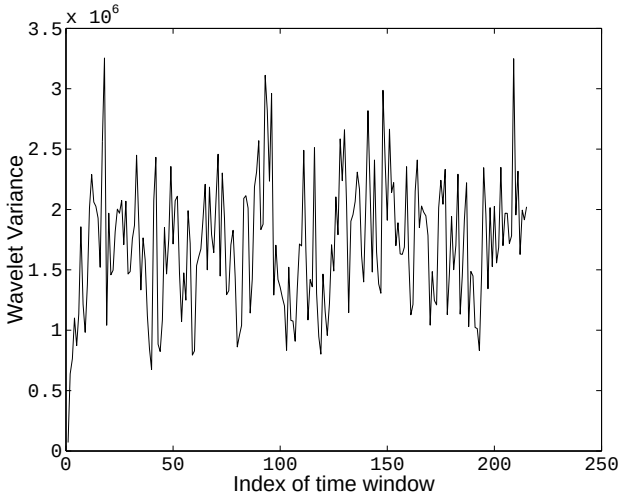

Fig. 6. Trace-1: Wavelet variance of aggregate hit traffic, cache size-100, scale-1 (0.78 ms)

the background traffic, which we refer to as the Superimposed Attack, second one where we replace the background traffic with the attack traffic during the attack duration, which we refer to as the Replacement Attack. We discuss each of these attacks in detail in the sections that follow.

B. CBR Spread Out Attack

UDP traffic unlike TCP does not respond to congestion, thereby making it a potential tool for launching an attack on a network. We use CBR UDP flows as representative attack flows in this scenario. The simulated DDoS attack is comprised of 200 CBR flows which send packets of size 1000 bytes over a duration of 15.34642 seconds when the attack lasts. It commences 24.671992 seconds from the reference point where the trace starts. The sending rate of each of the flows is 0.5213 Mbps and the aggregate attack traffic rate is 104.259 Mbps during the attack interval. Each of these flows

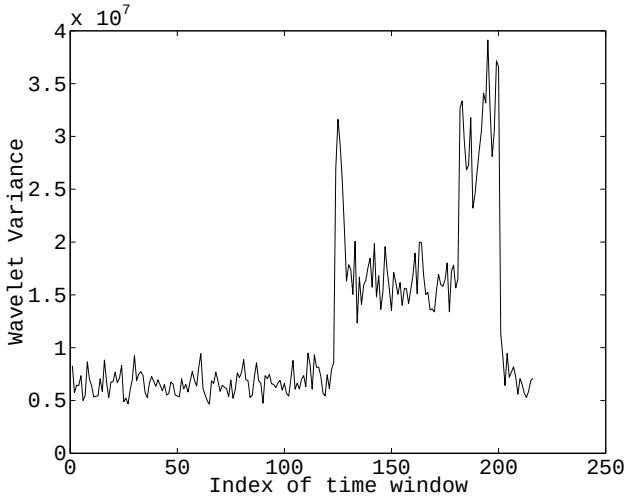

Fig. 7. Trace-1: CBR Spread Out Superimposed Attack - Wavelet variance of miss traffic, cache size-100, scale-1 (0.78 ms)

sends 1000 packets uniformly dispersed over the entire duration of the attack. The introduction of traffic drives the network into congestion.

If this attack is launched as a Superimposed Attack, it is observed that there is a sharp increase in the wavelet variance across the time-scales at the 124^{th} window, corresponding to 24.8 seconds from the start which can easily enable us to detect that an attack is in progress. This is observed in Fig. 7.

Whereas when launched as a Replacement Attack, the same attack is found to exhibit inverse behavior. The wavelet variance plot in Fig. 8 demonstrates a marked drop in the wavelet variance values at the commencement of the attack around the 125^{th} window corresponding to 25 seconds. This is because, the attack pattern is a periodic one and exhibits strong correlation and the wavelet variance drops during the attack, due to the absence of the network's background traffic.

The experiments were repeated for different cache sizes and the hit rates are

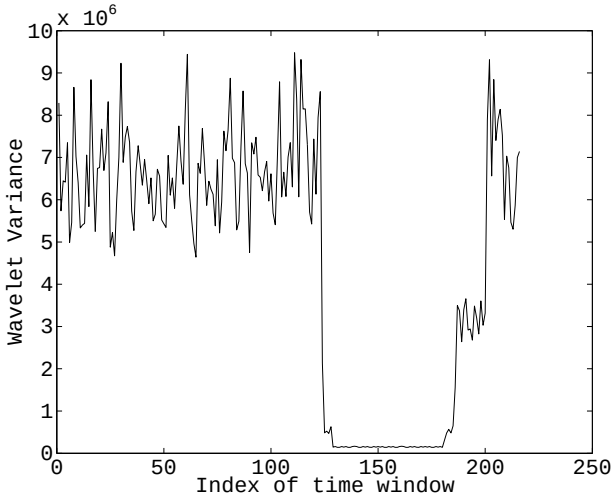

Fig. 8. Trace-1: CBR Spread Out Replacement Attack - Wavelet variance of miss traffic, cache size-100, scale-1 (0.78 ms)

summarized in Table II against the background traffic. It is observed that hit rates are higher for the Replacement Attack, because now the attack flows take over the cache and replace the background traffic flows that resided in it. After the initial misses, it results in successive hits. Whereas in a Superimposed Attack, the cache consists of both the background traffic's long-term flows and the attack flows.

C. CBR Back-to-Back Attack

The next DDoS attack we simulate is launched on the same background traffic. This time, we employ 200 CBR flows, each sending 1000 packets of size 1000 bytes at a rate of 104.25 Mbps. This attack has each of the flows sending packets one after the other in sequence over the duration of 15.34642 seconds during which the attack lasts. Thus each flow sends a burst of packets for a duration of 76 ms. We study the effect of this attack by launching it as a Superimposed Attack and as a Replacement Attack. For

Table II. Trace-1: Comparison of hit rates for superimposed and replacement attacks for CBR Spread Out Attack

S	Background Traffic	Superimposed Attack	Replacement Attack
20	0.0096	0.0042	0.0057
50	0.0910	0.1713	0.1716
100	0.1785	0.1853	0.2531
200	0.3584	0.3879	0.5384

the same reason mentioned earlier, it is observed that the wavelet variance shows a marked difference when the attack starts. Thus, we have been able to observe a clear change in levels of wavelet variance within less than 0.4 seconds from the start of the attack. The wavelet variance plots are shown in Fig. 9, Fig. 10 for the Superimposed and Replacement Attacks respectively.

D. VBR Spread Out Attack

To ensure that the results we obtained so far are not biased by the fact that the attacking flows are sending fixed size packets, we test our mechanism using a DDoS attack that is comprised of 200 flows, each of which send packets over the same attack duration as before, but now with variable packet sizes. The average sending rate of each of the flows is 0.3482 Mbps and the aggregate sending rate is 69.64 Mbps. It is observed that for the Superimposed Attack, there is a marked rise in the variance at the start of the attack, which can be attributed to the sudden change in behavior of the traffic during an attack. When the attack is launched as a Replacement Attack, it is observed that the variance drops at the start of the attack. This can be observed

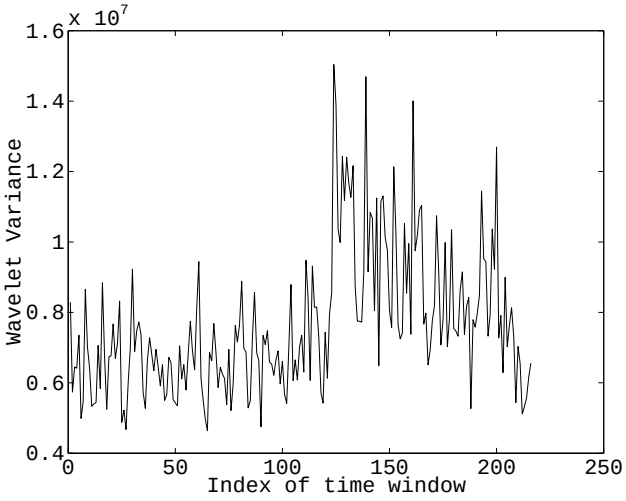

Fig. 9. Trace-1: CBR Back-to-Back Superimposed Attack - Wavelet variance of miss traffic, cache size-100, scale-1 (0.78 ms)

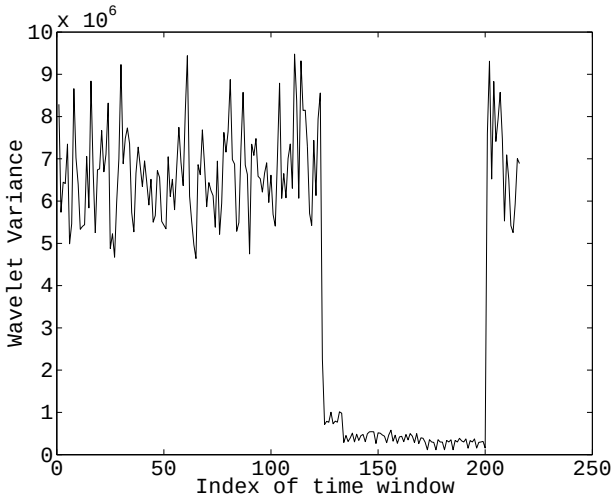

Fig. 10. Trace-1: CBR Back-to-Back Replacement Attack - Wavelet variance of miss traffic, cache size-100, scale-1 (0.78 ms)

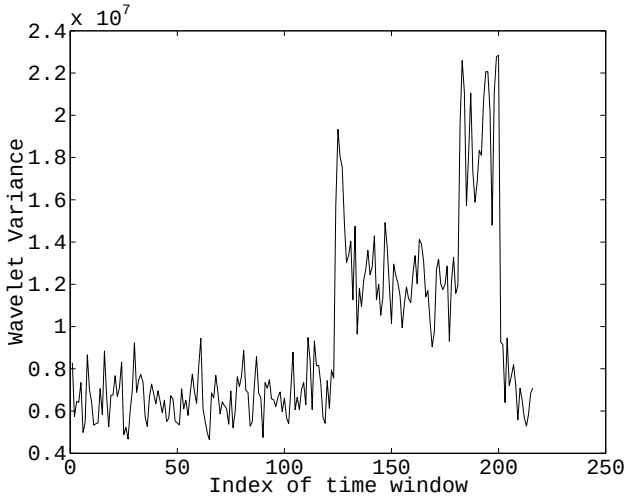

Fig. 11. Trace-1: VBR Spread Out Superimposed Attack - Wavelet variance of miss traffic, cache size-100, scale-1 (0.78 ms)

from Fig. 11 and Fig. 12.

E. VBR Back-to-Back Attack

Here, we use the same VBR traffic generated above, but now the same flows send packets one after the other in succession. The average sending rate of each VBR flow during the attacking period is 69.64Mbps. As in the case of the CBR back-to-back attack, in Fig. 13 and Fig. 14 a steep change in wavelet variance is observed in both cases namely the Superimposed and Replacement Attack indicating the start of an attack.

F. Randomly Generated Attack

The DDoS attacks that we launched so far have not taken into consideration the possibility of IP address spoofing. In order to understand the effect of this, a DDoS

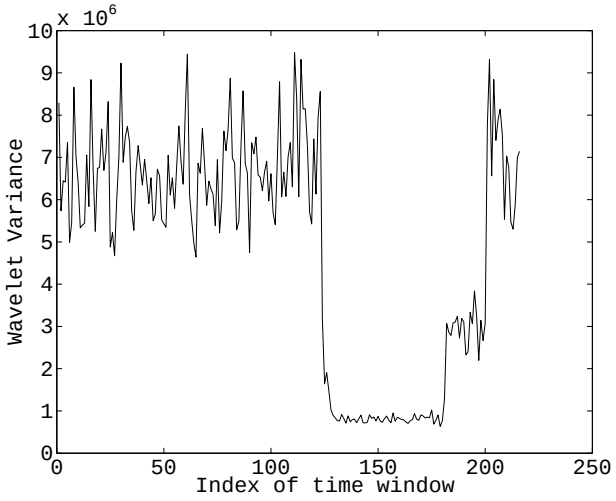

Fig. 12. Trace-1: VBR Spread Out Replacement Attack - Wavelet variance of miss traffic, cache size-100, scale-1 (0.78 ms)

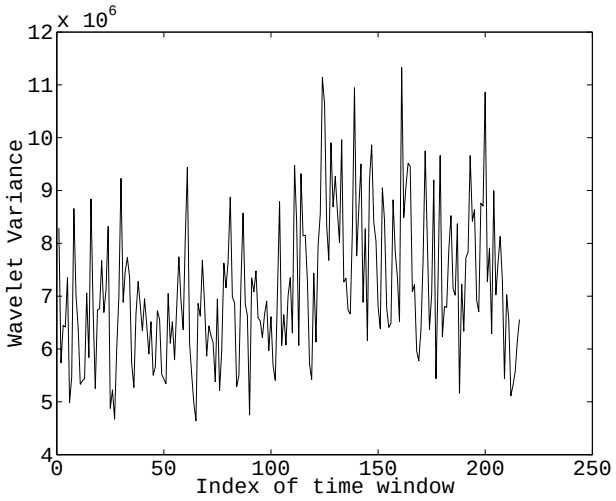

Fig. 13. Trace-1: VBR Back-to-Back Superimposed Attack - Wavelet variance of miss traffic, cache size-100, scale-1 (0.78 ms)

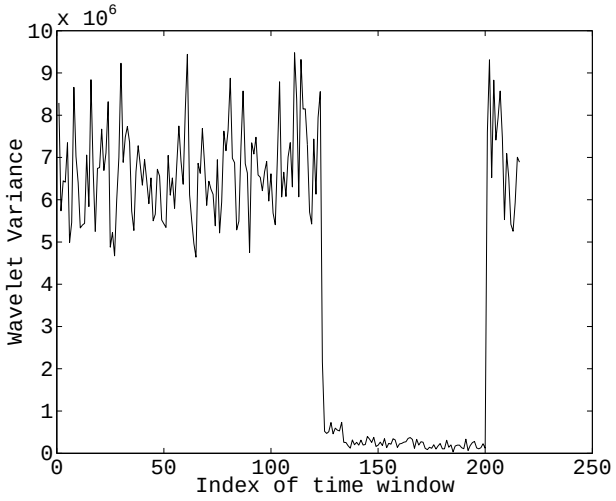

Fig. 14. Trace-1: VBR Back-to-Back Replacement Attack - Wavelet variance of miss traffic, cache size-100, scale-1 (0.78 ms)

attack is created for the same duration as before which comprised of flows with random source and destination IP addresses and random packet sizes. The aggregate sending rate is 82.43 Mbps during the duration of the attack, thereby driving the network to congestion.

The random attack is launched as a Superimposed Attack and it is observed that there is a sharp change in the wavelet variance at 24.8 seconds, close to the start of the attack, which clearly shows that the detection of an attack is immune to IP address spoofing as shown in Fig. 15. Despite the fact that the attack is launched by several short term flows, we can conclude that these flows have pushed some of the long-term flows away from the cache, therefore introducing sharp changes in wavelet variance of the aggregate miss traffic.

The experiments are repeated for different cache sizes and the hit rates are summarized in Table III. Here, we observed a decrease in hit rates when compared to

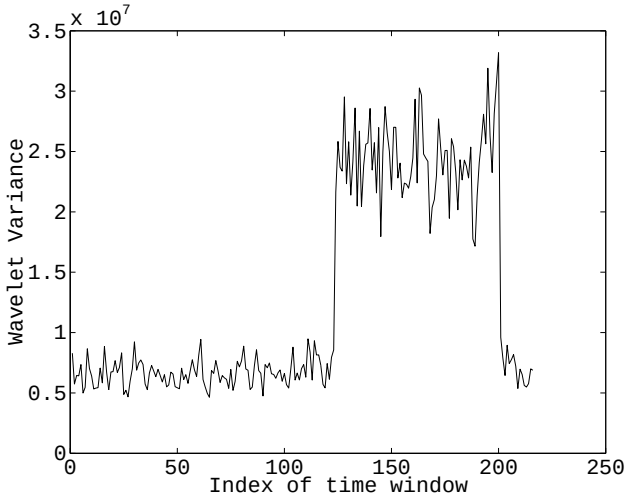

Fig. 15. Trace-1: Randomly Generated Attack - Wavelet variance of miss traffic, cache size-100, scale-1 (0.78 ms)

the background traffic. This can be attributed to the fact that we had used random attacking hosts here, so as in earlier cases, the attack flows enter the cache, but does not produce a sequence of hits as the attack has no periodicity. In fact, owing to the increase in the number of flows, more number of long-term flows that were present in cache tend to get pushed out to the aggregate miss traffic, thereby making attack detection more pronounced than other cases.

G. TCP Attack

So far, we have studied the effect of attacks that have been staged using flows that do not demonstrate congestion control. In order to understand the effect of congestion control mechanisms on the filtering technique we have used, we stage a distributed attack using 200 TCP flows. These flows have an aggregate sending rate of 86.1765 Mbps and the attack lasts for 15.34642 seconds. The network used for the TCP attack

Table III. Trace-1: Summary of hit rates for Random Attack

Cache Size	Hit Rate
20	0.0052
50	0.0546
100	0.0899
200	0.1354

is shown in Fig. 16.

We find that unlike the UDP attacks, here we observe a gradual rise in the values of wavelet variance from the 104^{th} to the 109^{th} window. This can be related to the slow-start phase, where the attack gradually builds up. Fig. 17 shows the gradual rise in wavelet variance until congestion is encountered, when the sending rate is lowered, thus resulting in a drop in wavelet variance.

H. Quantitative Measures for Attack Detection

In the previous sections, we noted that the wavelet variance exhibits sudden changes when an attack starts. Based on the observations, we propose quantitative measures that would enable successful detection of a DDoS attack.

1. Simple Threshold Mechanism

A straightforward approach to attack detection using the wavelet variance computed at the various scales would be to use a function of the mean and standard deviation of the wavelet variance at each scale as a threshold. When the wavelet variance crosses

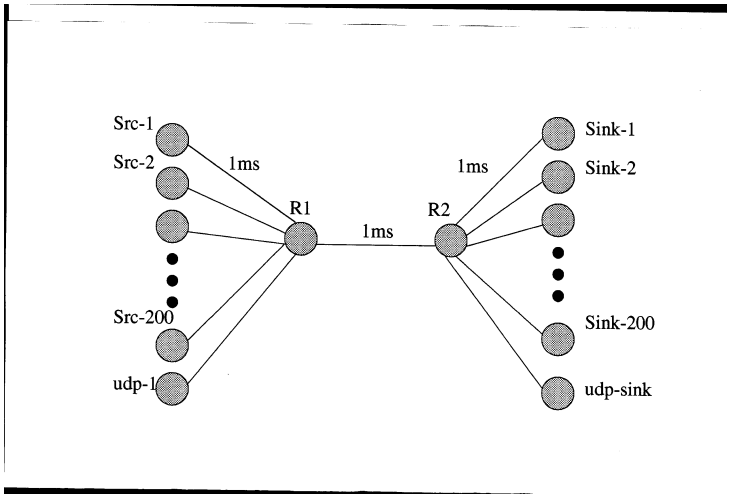

Fig. 16. Network topology for TCP Attack

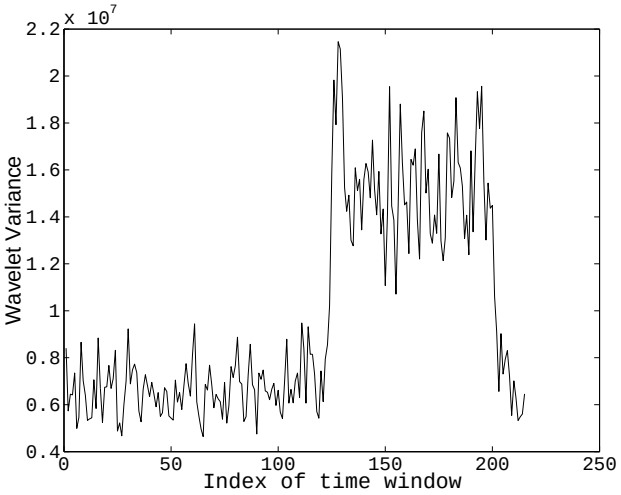

Fig. 17. Trace-1: TCP Attack - Wavelet variance of miss traffic, cache size-100, scale-1 (0.78 ms)

the threshold, then the attack detector signals an attack in progress. We used a threshold of $\overline{x}+2\sigma$ where $\overline{x}$ is the mean of the wavelet variance over an interval and σ denotes the corresponding standard deviation. While this method is effective most of the time, the measure is ineffective when the standard deviation σ could be much larger than the mean $\overline{x}$. When traffic is very light, σ tends to be larger than $\overline{x}$. Since this method relies on averages over long time-scales, the reactiveness of the method is also impacted.

2. Absolute Threshold using Ratios

In order to overcome the pitfalls of the previous method, we suggest an absolute threshold mechanism using wavelet variance ratios. At each window k, we compute the ratios V_k/V_{k-1} and V_{k-1}/V_k at each time-scale, where V_k and V_{k-1} represent the wavelet variance at that time-scale in the k^{th} and $k-1^{th}$ windows respectively. We evaluate the condition in Eq. 3.1. If this is satisfied, the detector signals an attack.

$$max(V_k/V_{k-1}, V_{k-1}/V_k) \geq \theta \tag{3.1}$$

This metric shows a marked change at the start and end of the attack. This experiment is repeated on different traces with different kinds of DDoS attacks and it is observed that an absolute threshold, θ of 2.0 seems to work quite well for the cutoff. The cutoff 2.0 is a heuristic and we are in the process of using statistical methods to determine a threshold parameter more formally. There is a steep change in wavelet variance at the start of the attack, while fluctuations otherwise are quite marginal, therefore enabling successful detection. We find that Eq. 3.1 for $\theta = 2.0$ enables successful detection of the attack for all the attacks we have staged so far over all the time-scales except the CBR Back-to-Back Superimposed, VBR Back-to-Back Superimposed and TCP Attacks. A closer look at the wavelet variance plots Fig. 9 on p. 22

and Fig. 13 on p. 24 shows that there is a not a steep rise in the wavelet variance values at the start of the attack as observed in all other cases, but there is a notable rise when we observe for a longer duration than between immediate wavelet variance values. In the case of the TCP Attack, we observe the same behavior. We now suggest a mechanism that can enable successful detection of changes of this nature using wavelet variance estimation, which we discuss below.

3. Wavelet Variance Estimation

The above mechanism works successfully if the distributed attack starts with a large number of flows that flood the network over a short span of time. If an attack is launched such that it builds up gradually, then the ratio of the successive averages of wavelet variances might not detect an attack, as the attack builds up marginally in each interval, thereby evading the above mechanism. In order to overcome this, we make use of a wavelet variance estimation, using exponential averages to estimate the wavelet variance at the current instant using past values of the wavelet variance. The estimate is obtained using Eq. 3.2.

$$Vest_k = \alpha.Vest_{k-1} + (1-\alpha).V_k \tag{3.2}$$

where α is a parameter that estimates the wavelet variance at window k, $Vest_k$ using the estimate in window k-1, $Vest_{k-1}$ and the value of the current wavelet variance, V_k. Here, the first term keeps track of the history of the fluctuations in wavelet variance, while the second term gives a weight to the current value. Since we aim at capturing a gradual build up in wavelet variance, we give lower weightage to sudden fluctuations associated with V_k and make use of α=0.8. Thus, as time progresses, a gradual rise in the estimate is an indication of an attack in progress. We now study the CBR Back-to-Back Superimposed Attack and the VBR Back-to-Back Superimposed

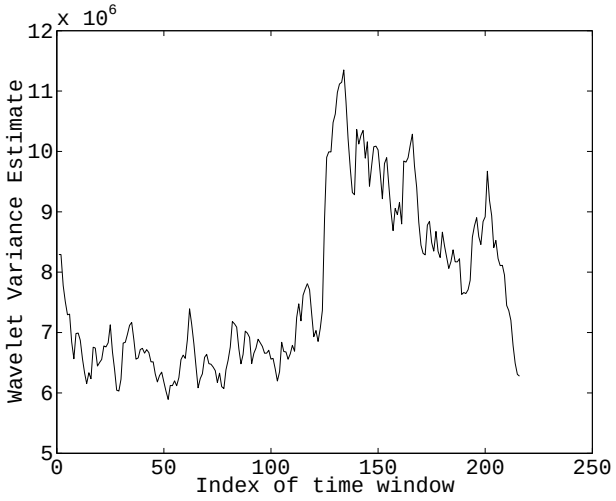

Fig. 18. Trace-1: CBR Back-to-Back Superimposed Attack: Wavelet variance estimate, cache size-100, scale-1 (0.78 ms)

Attack using wavelet variance estimation.

It is observed from Fig. 18 and Fig. 19 that there is a sharp rise in wavelet variance estimate at the start of the attack. The same wavelet variance estimation technique can be used for the TCP Attack too. Therefore, the absolute threshold mechanism coupled with the variance estimation can serve as an effective metric for attack detection.

I. Results of Trace-2

We now present the results obtained by staging these attacks on another trace obtained from the Buffalo site. A summary of the details of the background traffic for this trace is provided in Table IV.

There are totally 3341 flows and approximately 65 flows are dominant. A cache size of 100 has been used for presenting the results obtained for the different kinds of

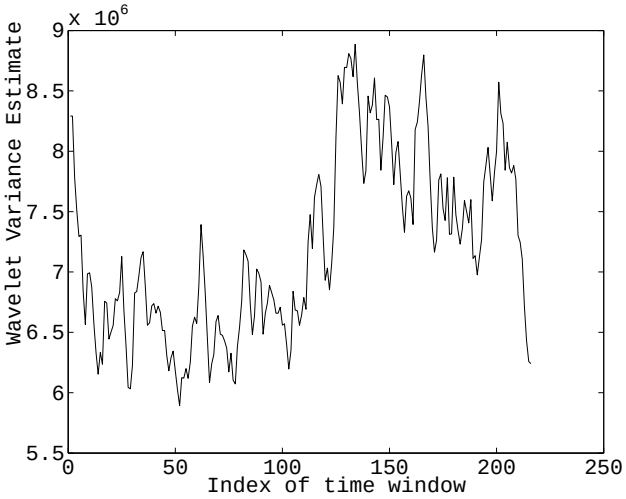

Fig. 19. Trace-1: VBR Back-to-Back Superimposed Attack: Wavelet variance estimate, cache size-100, scale-1 (0.78 ms)

Table IV. Trace-2: Summary of bytes sent per flow in the background traffic

Data sent (in MB)	No: of flows
0-0.5	3217
0.5-1	59
1-2	32
2-10	25
≥ 10	8

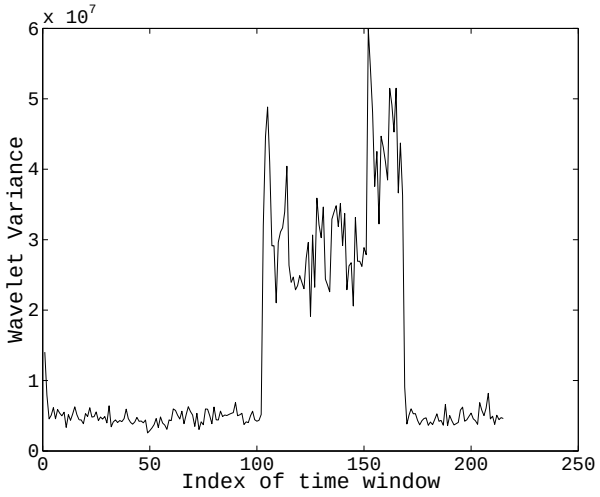

Fig. 20. Trace-2: CBR Spread Out Superimposed Attack - Wavelet variance of miss traffic, cache size-100, scale-1 (1.63 ms)

attacks. The aggregate sending rate of the background traffic is 39.88 Mbps.

A sampling interval of 1.63 ms was used with 100 % duty cycle and the attacks commence at 42.83 seconds from the start of the traffic and lasts for a duration of 27.5 seconds. 200 flows are used to launch the attacks. For the CBR Spread Out Superimposed Attack, we observe in Fig. 20 that there is a rise in wavelet variance at the 103^{rd} at 42.97 seconds indicating the start of the attack. This is observed across all the 6 time-scales. The same CBR Spread Out Attack when launched as a replacement attack is found to demonstrate a drop in wavelet variance across all 6 time-scales as observed in Fig. 21.

For the CBR Back-to-Back Superimposed Attack, as seen in Trace-1, we observe in Fig. 22, that in Trace-2, the rise is seen at the start of the attack corresponding to the 103^{rd} window indicating the start of the attack. We find that the wavelet variance satisfies the absolute-cutoff condition unlike Trace-1. This can be attributed to the fact that the average inter-arrival times is higher in Trace-2 as opposed to Trace-1

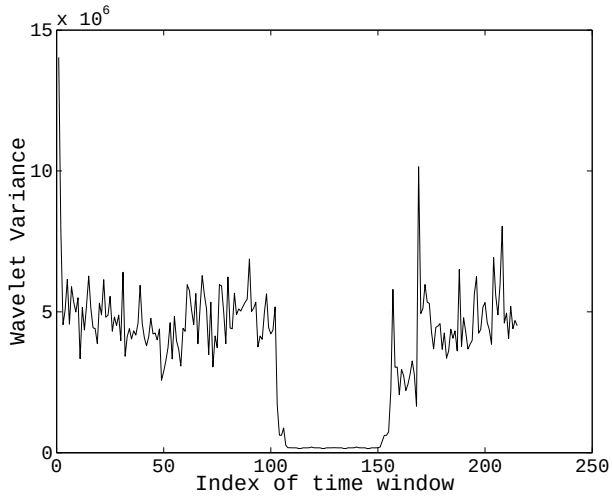

Fig. 21. Trace-2: CBR Spread Out Replacement Attack - Wavelet variance of miss traffic, cache size-100, scale-1 (1.63 ms)

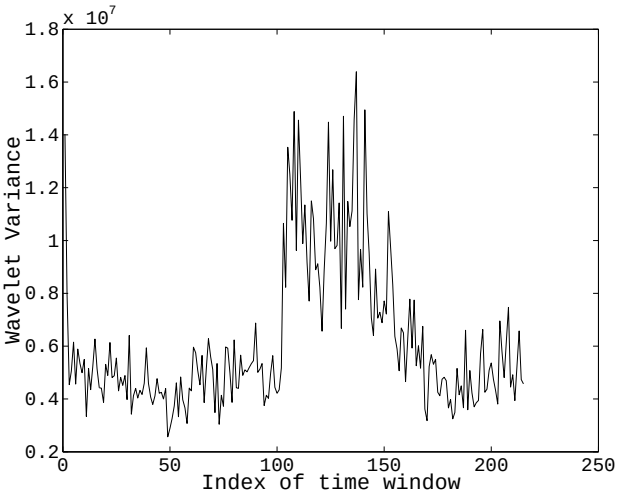

Fig. 22. Trace-2: CBR Back-to-Back Superimposed Attack - Wavelet variance of miss traffic, cache size-100, scale-1 (1.63 ms)

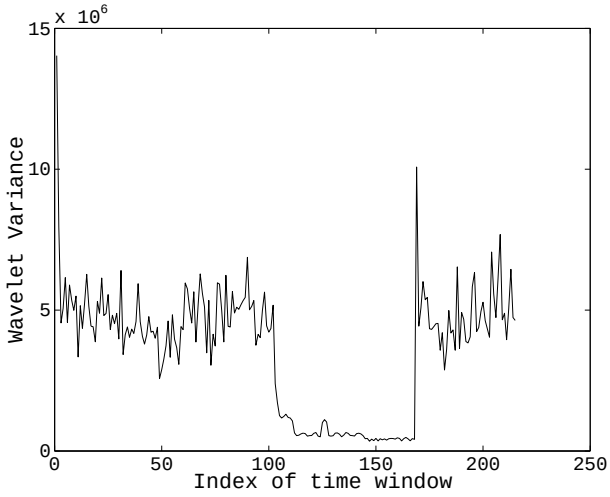

Fig. 23. Trace-2: CBR Back-to-Back Replacement Attack - Wavelet variance of miss traffic, cache size-100, scale-1 (1.63 ms)

causing higher changes in wavelet variance over small durations.

As in Trace-1, Trace-2 indicates a drop in wavelet variance in the CBR Back-to-Back Replacement Attack over all the time-scales, as seen in Fig. 23.

For the VBR Spread Out Attack, both in the case of Replacement and Superimposed Attack, it is observed from Fig. 24 and Fig. 25 that there are steep changes in wavelet variance at the start of the attack.

Unlike Trace-1, we find that in Trace-2 for the VBR Back-to-Back Superimposed Attack, the values of wavelet variance over the 6 time-scales satisfy the absolute threshold condition. This can be attributed to the difference in average interarrival times between Trace-1 and Trace-2. The plots for the Superimposed and Replacement Attacks are shown in Fig. 26 and Fig. 27.

As in Trace-1, Trace-2 also successfully detects attacks despite IP address spoofing which is shown from the plot of wavelet variance for the Random Attack in Fig. 28.

The TCP Attack is staged in the same way as in Trace-1 using 200 TCP flows.

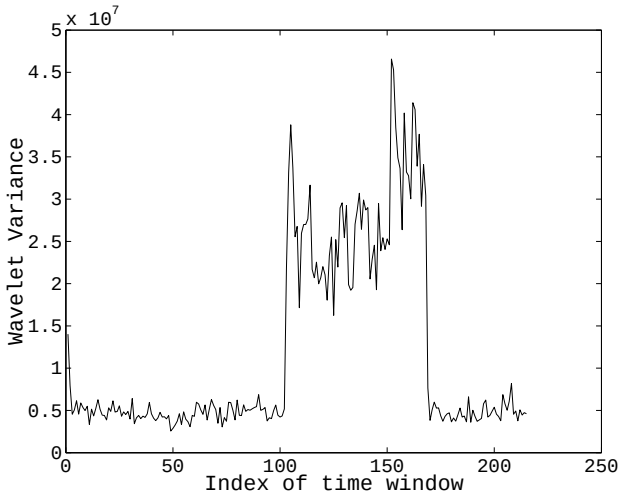

Fig. 24. Trace-2: VBR Spread Out Superimposed Attack - Wavelet variance of miss traffic, cache size-100, scale-1 (1.63 ms)

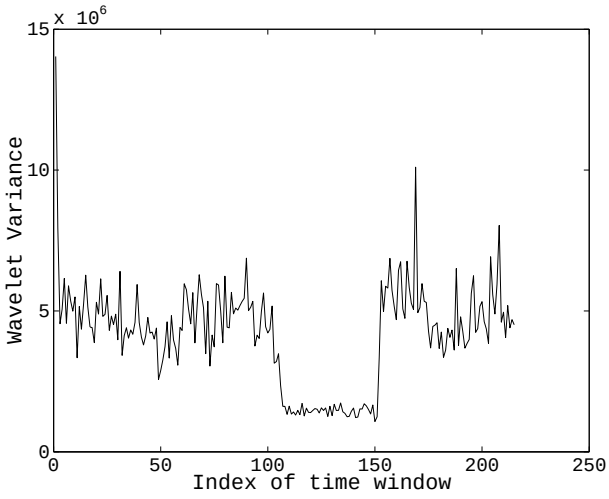

Fig. 25. Trace-2: VBR Spread Out Replacement Attack - Wavelet variance of miss traffic, cache size-100, scale-1 (1.63 ms)

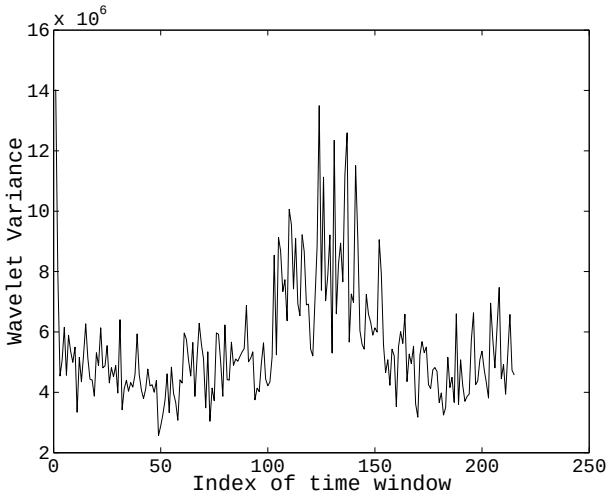

Fig. 26. Trace-2: VBR Back-to-Back Superimposed Attack - Wavelet variance of miss traffic, cache size-100, scale-1 (1.63 ms)

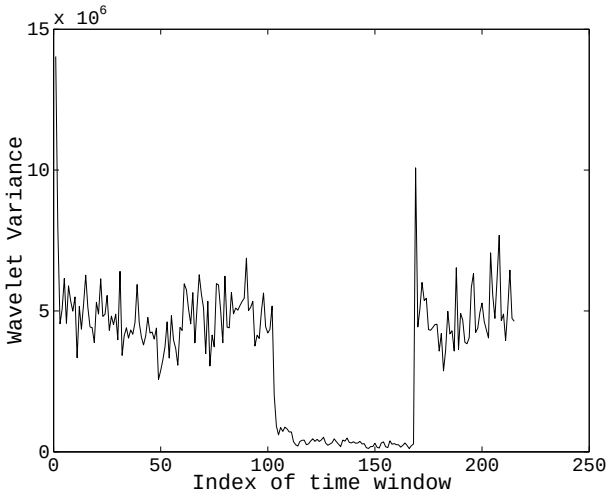

Fig. 27. Trace-2: VBR Back-to-Back Replacement Attack - Wavelet variance of miss traffic, cache size-100, scale-1 (1.63 ms)

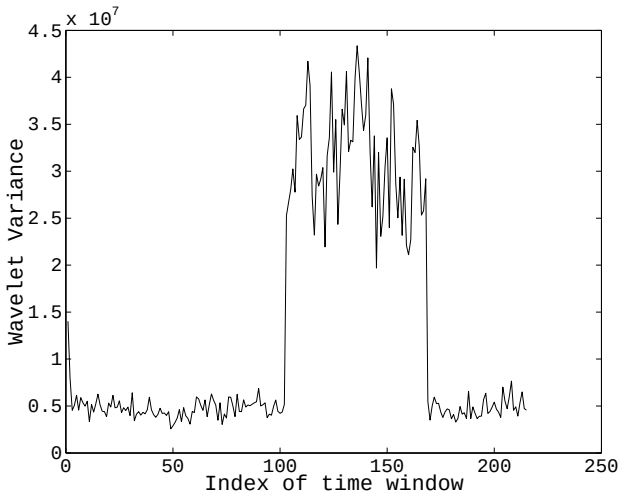

Fig. 28. Trace-2: Random Attack - Wavelet variance of miss traffic, cache size-100, scale-1 (1.63 ms)

In this case, unlike Trace-1, we find that the Absolute Threshold Mechanism works. Here, the aggregate sending rate of the background traffic is lower, therefore TCP attack is more predominant as more number of packets are sent by the TCP flows than before, therefore a sharper rise in wavelet variance as shown in Fig. 29.

Thus, from the results obtained we can conclude that we can observe the traffic over a network and use this as a deciding factor for the cutoff for detection. This can be done using statistical methods and this is discussed in depth in Chapter 5.

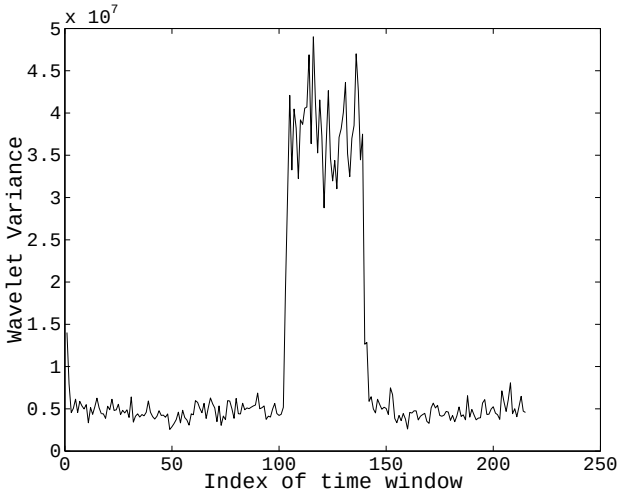

Fig. 29. Trace-2: TCP Attack - Wavelet variance of miss traffic, cache size-100, scale-1 (1.63 ms)

CHAPTER IV

DISCUSSION

The effectiveness of the DDoS attack detection mechanism proposed here depends on the choice of parameters that control the LRU Cache filter and the sampling used for computation of the wavelet variance. We now discuss the impact of varying each of these parameters on the end results.

A. Effect of Cache Size

The Cache size, S is a measure of the number of high-bandwidth flows whose state information can be maintained by us at a time. A very large value of cache size would result in capturing some of the short term flows also in cache, while a very small size of cache, would result in displacing some of the long-term high bandwidth flows from the cache onto the miss traffic even in the absence of an attack, thereby resulting in false-alarms.

If we want to maintain state for N dominant flows with a very high probability, the cache needs to provide state for $S > N$ flows since flows are randomly replaced in the cache. Our current analysis seems to suggest that $N \leq S \leq 2N$ seems to suffice for a wide range of traffic distributions. The cache size has to be chosen such that our wavelet mechanism does not trigger "false-alarms" with normal traffic. When an attack is staged with fewer than $S-N$ flows, where N is the number of long-term flows in the normal traffic and S is the chosen cache size, it is possible that the attack flows do not change the distribution or wavelet signatures of the miss traffic. As pointed out earlier, even though the wavelet signatures will not detect such attacks (with few flows), they can be contained by a policy-based regulator of the cached traffic shown in Fig. 1 on p. 10. WADeS is designed to detect DDoS attacks with greater than 'S'

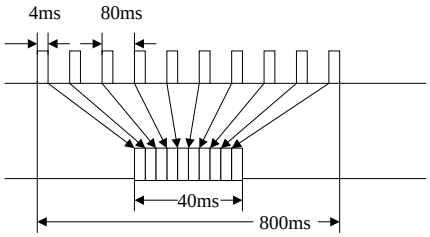

Fig. 30. Distributed Sampling

sources.

B. Effect of Sampling Style

The use of sampling for the aggregate miss traffic prior to computation of wavelet variance results in significant decrease in computation required for the detection of an attack. But it is observed that deciding on how sampling can be done is highly effective in attack detection. We use a byte-sampling interval of 156μs. A 40ms sample of data in every 800ms is used for our computations. If the 40ms sample is taken at the start of the 800ms interval, then it is possible to launch an attack after the first 40ms at regular intervals and continuously evade our detection mechanism. In order to reduce the possibility of not sampling during an attack, we used a distributed sampling technique with clustering. Fig. 30 provides a clearer picture of this.

We take 4ms samples in every 80ms and cluster 10 such samples to give a 40ms sample on which the same techniques of wavelet variance computation and attack detection are used. The smaller the time between 2 sampling intervals, the lower the probability of an attacker being able to evade detection. To improve the performance, we also add a random offset within each 80ms interval to decide the starting point of

the 4ms interval where samples are to be obtained. This will prevent any attacker from generating a predetermined pattern of attack traffic to evade detection. Thus, with 5% the effort, we can detect the attack with reasonable accuracy. The more distributed the sampling, the more effective it is in decreasing the probability of missing an attack due to not sampling a portion of the traffic. In addition to this, when we have smaller sampling intervals, we require noise suppression. The smaller sampling interval is more efficient in capturing changes, therefore even slight fluctuations in wavelet variance is reflected here in a far more magnified form than using a larger sampling interval that we used earlier. So, we now use the average over a few samples of wavelet variance (in our case, we use 5) to get the values V_k and V_{k-1} and use the same criterion in Eq. 3.1 to declare the start of an attack.

Table V, VI, VII, VIII, IX, X, XI, XII, XIII compare the efficiency of attack detection for the conventional and distributed sampling techniques for the various types of attacks we have discussed so far. We observe that these sampling techniques are almost as effective as the case where we looked at every packet that traverses the network. We redefine our attack detection criterion for absolute threshold using ratios to accomodate for the decrease in computation due to sampling of the miss traffic. If more than 75% of the time-scales satisfy Eq. 3.1, then the detector signals an attack. We find that with lower computational cost, we are able to achieve the same detection efficiency. The sucess ratio is defined as in Eq. 4.1

$$\text{Success Ratio} = \frac{\text{No: of time-scales satisfying Eq.3.1}}{\text{No: of time-scales per window}} \tag{4.1}$$

We find that distributed sampling with random offset improves performance with the same amount of work being done by conventional sampling.

Table V. Summary of success percentage ratio using Absolute Threshold-CBR Spread Out Superimposed Attack

Cache Size	Conventional Sampling	Distributed Sampling
20	1.0	1.0
50	1.0	1.0
100	1.0	1.0
200	1.0	1.0

Table VI. Summary of success percentage ratio using Absolute Threshold-CBR Spread Out Replacement Attack

Cache Size	Conventional Sampling	Distributed Sampling
20	1.0	1.0
50	1.0	1.0
100	1.0	1.0
200	1.0	1.0

Table VII. Summary of success percentage ratio using Absolute Threshold-CBR Back to Back Superimposed Attack

Cache Size	Conventional Sampling	Distributed Sampling
20	1.0	1.0
50	0.5	1.0
100	1.0	1.0
200	0.833	1.0

Table VIII. Summary of success percentage ratio using Absolute Threshold-CBR Back to Back Replacement Attack

Cache Size	Conventional Sampling	Distributed Sampling
20	0.8333	0.8333
50	0.8333	0.8333
100	0.8333	0.8333
200	0.8333	0.8333

Table IX. Summary of success percentage ratio using Absolute Threshold-VBR Spread Out Superimposed Attack

Cache Size	Conventional Sampling	Distributed Sampling
20	1.0	1.0
50	1.0	1.0
100	1.0	1.0
200	1.0	1.0

Table X. Summary of success percentage ratio using Absolute Threshold-VBR Spread Out Replacement Attack

Cache Size	Conventional Sampling	Distributed Sampling
20	1.0	1.0
50	1.0	1.0
100	1.0	1.0
200	1.0	1.0

Table XI. Summary of success percentage ratio using Absolute Threshold-VBR Back to Back Superimposed Attack

Cache Size	Conventional Sampling	Distributed Sampling
20	0.1667	0.1667
50	0.1667	0.3333
100	0.1667	0.3333
200	0.1667	0.3333

Table XII. Summary of success percentage ratio using Absolute Threshold-VBR Back to Back Replacement Attack

Cache Size	Conventional Sampling	Distributed Sampling
20	1.0	1.0
50	1.0	1.0
100	1.0	1.0
200	1.0	1.0

Table XIII. Summary of success percentage ratio using Absolute Threshold-Random Attack with and without distributed sampling

Cache Size	Conventional Sampling	Distributed Sampling
20	1.0	1.0
50	1.0	1.0
100	1.0	1.0
200	1.0	1.0

C. Effect of Duty Cycle

We define duty cycle to be the ratio of the duration when a window of samples is taken to the duration between consecutive samples. In the simulations mentioned in the previous set of results, we use a duty cycle of 0.05. We observe that even when we sampled at a rate of 1 in 20, DDoS attack detection is highly successful. Sampling could however lead to simple strategies for evading attack detection. If the attack happens in such a way that it always evades the sampling for small duty cycles, then an increase in duty cycle is the only alternative to capturing the attack. This is shown using a simulated attack which consisted of random flows launched on the background traffic. This attack sends bursts of 50-200 packets over very short durations with ON period of 50μs-200μs and with an average OFF period of 76ms. It is observed that a duty cycle of 0.05 is unable to detect an attack. On increasing the duty cycle, it is found that at duty cycles of 0.1, 0.2 the attack is still undetected. But at 0.4 and higher duty cycles, the attack is successfully detected over all six scales. As the duty cycle is increased still further, all the attack bursts can be successfully located by a change in the wavelet variance. This can be observed from Fig. 31 and Fig. 32. Even when attack is not detected with a duty cycle of 0.4 or lower, it is observed that the attack consumes resources only for only 1.5% of the time. Thus, we have achieved the goal of containing the attacks by making it harder to stage attacks that consume 100% of the resources. This clearly shows the tradeoff between the processing cost and attack detection success.

D. Effect of DDoS Attack with few flows

The mechanism that we have used so far relies heavily on changes that are caused by introduction of attack traffic to the aggregate miss traffic, which is captured by the

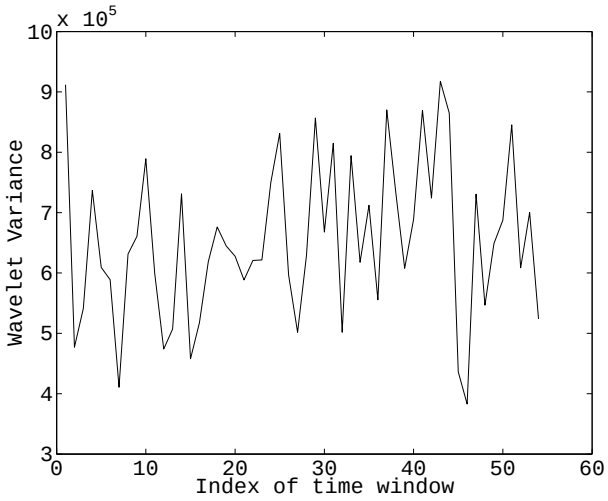

Fig. 31. Effect of Duty Cycle: Wavelet variance plots at scale-1 (0.156 ms) cache size-50, duty cycle-0.05

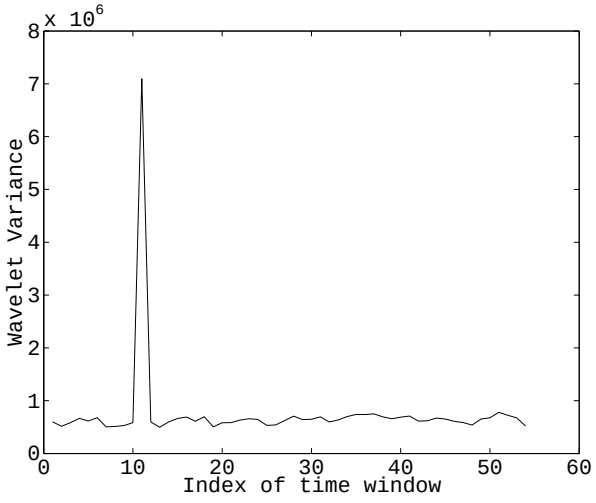

Fig. 32. Effect of Duty Cycle: Wavelet variance plots at Scale-1 (0.156 ms) cache size-50, duty cycle-0.4

Table XIV. Summary of attack detection efficiency for Absolute Threshold method for an attack with few flows

Cache Size	DC 0.05	DC 0.1	DC 0.2	DC 0.4
20	0.1667	0.1667	0.1667	0.1667
50	0.1667	0.1667	0.1667	0.1667
100	0.1667	0.1667	0.1667	0.1667
200	0.1667	0.1667	0.1667	0.1667

wavelet variance computed. But, if a DDoS attack is comprised of very few flows, then they do not cause great changes to the contents of the cache as they occupy a very small portion of the cache, so there is no displacing of high-bandwidth traffic onto the miss traffic, since the attack flows once they start sending at a sufficiently high rate, then they end up becoming a part of the hit traffic in a very short time, therefore not causing any changes on the miss traffic. This is the reason why we are not able to detect a DDoS attack which comprises of few flows using our approach. This can be observed from the results in Table XIV which we obtain for an attack comprising of 4 flows. The network traffic without attack has an aggregate sending rate of 129.486 Kbps, while the sending rate of each of the attacking flows is 62.433 Kbps. We note that at higher duty cycles (DC) the attack is not detected. However, these DoS attacks can be contained with an accompanying mechanism as in Fig. 1 on p. 10.

CHAPTER V

STATISTICAL VALIDATION

So far, we developed heuristics to help us identify a Denial of Service Attack. We use statistical hypothesis testing procedures to test the effectiveness of the methods we discussed in the previous chapter. In the following section, we provide a background on the statistical measures employed.

A. Background on Hypothesis Testing

The *Null Hypothesis* (H_o) is defined as a hypothesis about a parameter. It is usually the reverse of the parameter's expected behavior. Hypothesis testing is used to verify the validity of the Null Hypothesis in the light of experimental data. Depending on this, we may accept or reject the Null Hypothesis. We can use experimental data to reject the Null Hypothesis, in which case we accept the *Alternative Hypothesis*, which is usually defined to be the parameter's expected behavior. The significance level is a criterion used for Hypothesis testing and can be used to reject the Null Hypothesis. We first obtain the difference between the experimental results and the results from Null Hypothesis. Then, the probability that the Null hypothesis is rejected is computed and is compared to a certain significance level. This is computed using Eq. 5.1.

$$\text{Probability of Rejection} = \frac{\text{No: of times } H_o \text{ is violated}}{\text{No: of trials}} \tag{5.1}$$

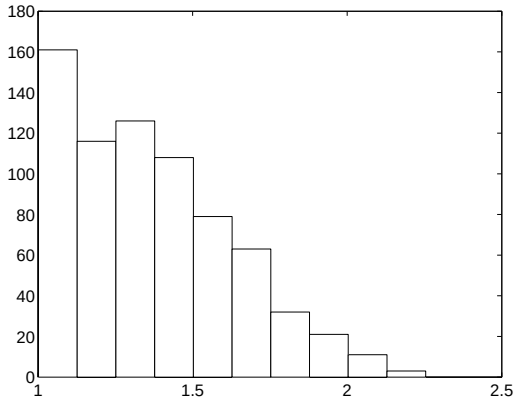

Fig. 33. No attack: scale-1

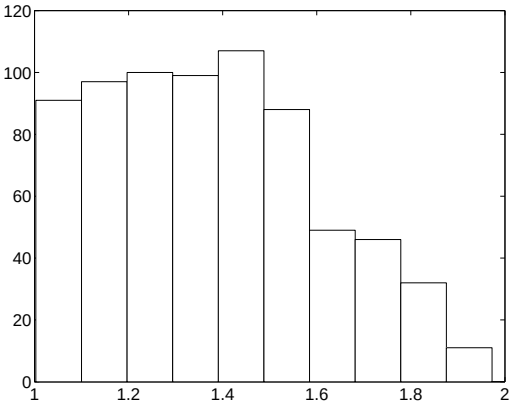

Fig. 34. No attack: scale-2

B. Hypothesis Testing for Absolute Threshold Method

In our experiments, we define the Null Hypothesis, H_o and Alternative Hypothesis as follows:

H_o : No Attack

H_a : There is an Attack

We generate data for our experiments by permuting the traffic corresponding to the portion where the cutoff is computed. We use 6! permutations of the data and compute the cutoffs in each case and obtain the distributions of the cutoff at the different time-scales. This is done for the traffic with and without the attacks.

From Figs. 33-38, we find that at higher time-scales, there are more number of ratios that are greater that 2, thereby can cause false alarms. The results of the computed probabilities for the Null Hypothesis is shown in Table XV. We first compute the probabilities for the fixed cutoff of 2.0 over all six time-scales. The next column indicates the cutoffs chosen to obtain significance levels of lower than 2.5%. We find from our experiments that the second choice of cutoffs successfully helps attack detection. By increasing the cutoffs at higher time-scales, we lower the

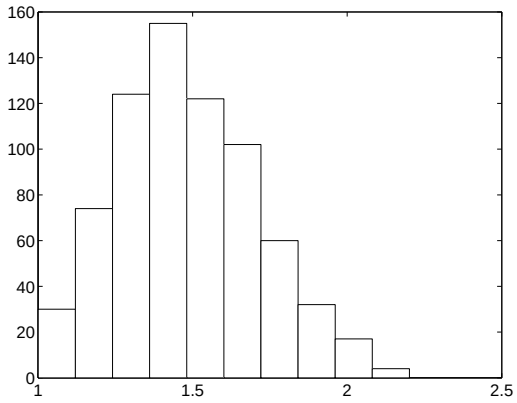

Fig. 35. No attack: scale-3

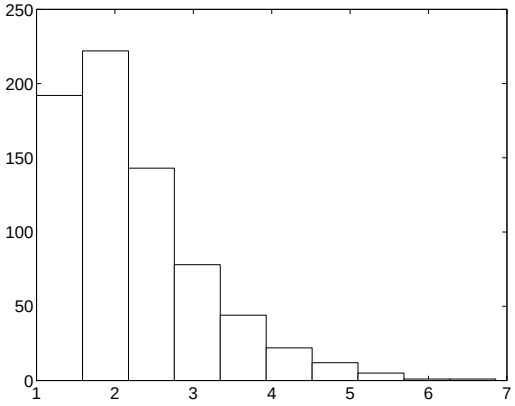

Fig. 36. No attack: scale-4

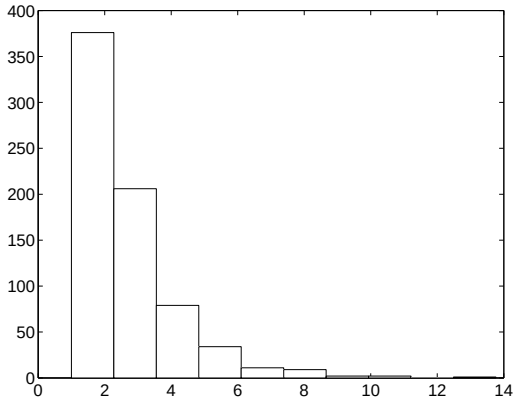

Fig. 37. No attack: scale-5

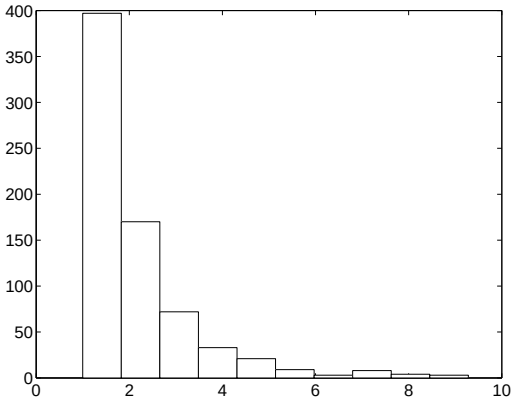

Fig. 38. No attack: scale-6

Table XV. Summary of computed probabilities for the Null Hypothesis for fixed and variable cutoffs

Scale	p for H_o-fixed cutoff 2.0	p for H_o-variable cutoff
1	0.0194	0.0194 (2.0)
2	0.0000	0.0000 (2.0)
3	0.0222	0.0222 (2.0)
4	0.5097	0.0111 (5.0)
5	0.5597	0.0138 (8.0)
6	0.3722	0.0083 (8.0)

probability of false alarms which is higher when we had a fixed cutoff as seen from the computed probabilities in the first column.

We find that using variable cutoffs for attack detection is more effective in reducing the number of false alarms. But depending on how the cutoff is chosen, there is a trade off between the false alarm percentage and the efficiency of attack detection as seen in Table XVI.

We also obtain the distributions for the different time-scales using permutation methods when an attack is in progress and we find that over all the time-scales, the values of the cutoffs are higher than the threshold values chosen for detection, thus enabling statistical verification of the cutoffs which we have chosen earlier. These distributions can be seen in Figs. 39-44.

Table XVI. Summary of cutoffs for different significance levels for Null Hypothesis

Scale	Cutoff (p=0.025)	Cutoff (p=0.05)	Cutoff (p=0.1)
1	1.997	1.984	1.968
2	1.981	1.975	1.963
3	1.998	1.986	1.973
4	4.534	4.0	3.508
5	6.567	5.460	4.470
6	6.000	4.700	3.855

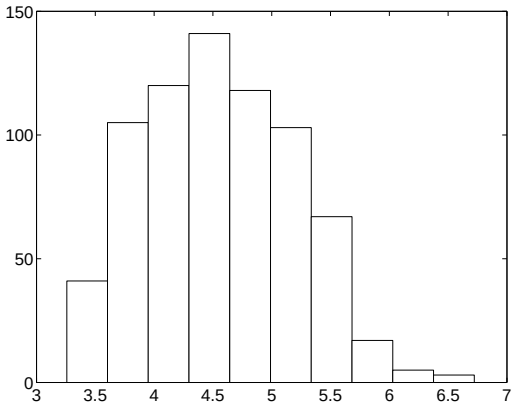

Fig. 39. With attack: scale-1

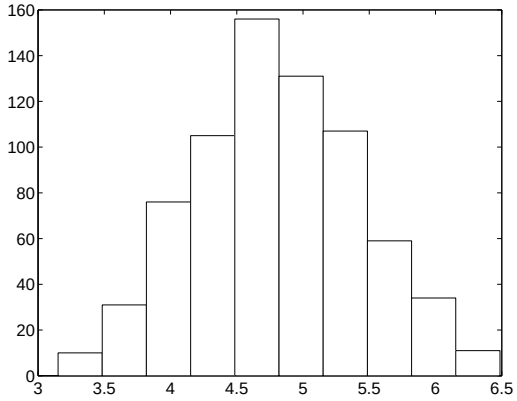

Fig. 40. With attack: scale-2

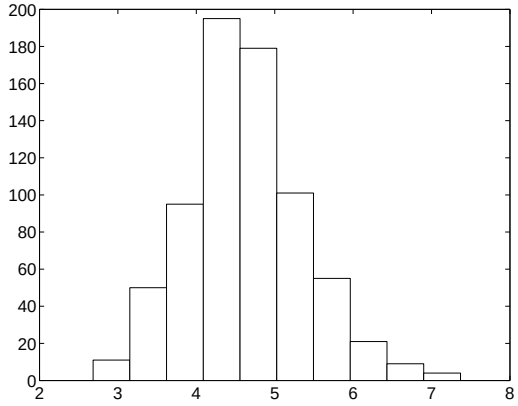

Fig. 41. With attack: scale-3

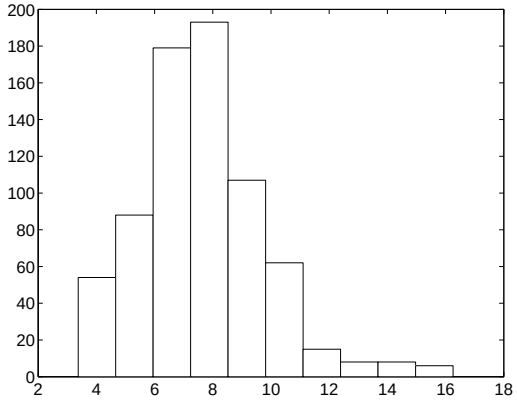

Fig. 42. With attack: scale-4

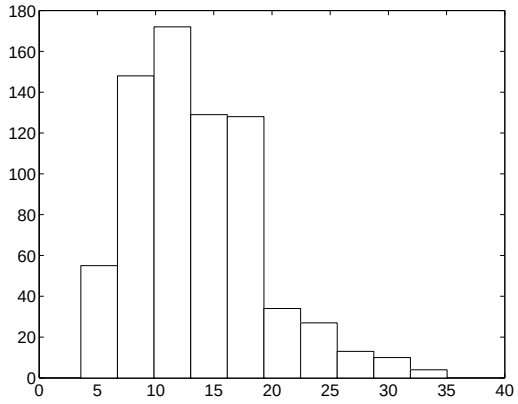

Fig. 43. With attack: scale-5

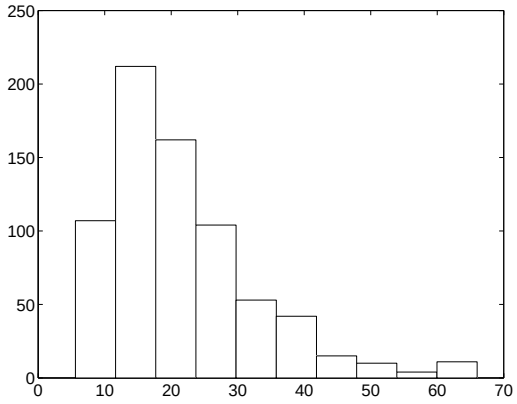

Fig. 44. With attack: scale-6

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

This research work proposes WADeS, a wavelet signature based tool for DDOS attack detection. WADeS employs a filter of network traffic to remove a number of dominant flows before computing the wavelet variance estimates. The size of the employed filter determined the minimum number of flows required to stage a DDOS attack and hence detection by WADeS. WADeS is shown to be highly effective in detecting a number of simulated DDOS attacks. It is shown that WADeS detects an attack fairly quickly from the onset of an attack. WADeS is shown to incur O(1) cost per packet arrival. It is shown that appropriate choice of sampling intervals and lower duty cycles can be used to lower the cost of WADeS further. Since WADeS depends only on the aggregate byte counts of the filtered traffic, it is not sensitive to IP address spoofing.

WADeS employs a number of heuiristic measures for attack detection thresholds. These thresholds are derived from trace-driven studies of network traffic. We plan on estimating these thresholds through more formal statistical measures in the future. For example, Permutation methods can be used to derive a value for the threshold for the absolute threshold attack detector for a certain significance level to enable automatic detection of an attack. Similarly, trend estimation using Bootstrap techniques can be used for the prediction of an attack using the Wavelet Variance Estimation detector.

While our results here reported no false alarms, we plan to evaluate WADeS over a larger number of traces and attacks to study and minimize the rate of false alarms.

WADeS is an effective mechanism for early attack detection, but is not a complete solution to preventing DDoS attacks, as it needs to work in coordination with other mechanisms to identify and contain the source of an attack. We plan to generalize

the approach of WADeS along multiple dimensions (protocol based, destination based along with flow-based filtering) to facilitate this analysis.

REFERENCES

[1] B.Martin, "Have script, will destroy(lessons in DoS)," Feb 2000, http://www.attrition.org.

[2] Smitha and A.L.Narasimha Reddy, "LRU-RED: An active queue management scheme to identify high bandwidth flows at a congested router," in *Proc. of IEEE Globecomm.*, San Antonio, TX, Nov 2001, vol. 4, pp. 2311–2315.

[3] Shriram Sarvotham, Rudolf Riedi, and Richard Baranuik, "Connection-level analysis and modeling of network traffic," in *Proc. of the ACM SIGCOMM Internet Measurement Workshop*, San Francisco, CA, Nov 2001, pp. 99–103.

[4] A.Demers, S.Keshav, and S.Shenker, "Analysis and simulation of a fair-queueing algorithm," *Journal of Internetworking Research and Experience*, vol. 1, no. 1, pp. 3–26, Sept 1990.

[5] Ratul Mahajan and Sally Floyd, "Controlling high bandwidth flows at the congested router," in *9th International Conference of Networking Protocols*, Riverside, CA, Nov 2001, pp. 192–201.

[6] Deying Tong and A.L.Narasimha Reddy, "QoS enhancement with partial state," in *International Workshop on QoS*, London, U.K, June 1999.

[7] Phani Achanta and A.L.Narasimha Reddy, "LRU-FQ: A fair-queueing mechanism to control high bandwidth flows," Master's thesis, Texas A&M University, Under preparation.

[8] Cristian Estan and George Varghese, "New directions in traffic measurement and accounting," in *Proc. of ACM SIGCOMM*, Jan 2002, vol. 32, p. 75.

[9] Rong Pan, Lee Breslau, Balaji Prabhakar, and Scott Shenker, "Approximate fairness through differentiated dropping," in *Proc. of ACM SIGCOMM*, Jan 2002, vol. 32, p. 72.

[10] W.Leland, M.Taqqu, W.Willinger, and D.Wilson, "On the self-similar nature of ethernet traffic(extended version)," *IEEE/ACM Trans. Networking*, vol. 2, no. 1, pp. 1–15, 1994.

[11] R.H.Riedi, M.S.Crouse, V.Ribiero, and R.G.Baranuik, "A multifractal wavelet model with application to TCP network traffic," *IEEE Trans. Inform. Theory*, 1999.

[12] Anja Feldmann, Anna C.Gilbert, Polly Huang, and Walter Willinger, "Dynamics of IP Traffic: A study of the role of variability and the impact of control," in *Proc. of the ACM/SIGCOMM'99*, Boston,MA, Aug 1999, pp. 301–303.

[13] P.Ferguson and D.Senie, "Network ingress filtering:Defeating denial of service attacks which employ IP address spoofing," Internet Draft, January 1998, http://www.landfield.com/rfcs/rfc2267.html.

[14] Steve Bellovin, Marcus Leech, and Tom Taylor, "ICMP traceback messages," Internet Draft, October 2001, http://www.ietf.org/internet-drafts/draft-ietf-itrace-01.txt.

[15] Cisco Systems Inc., *Characterizing and tracing packet floods using Cisco routers*, http://www.cisco.com/warp/public/707/22.html.

[16] H.Burch and B.Cheswick, "Tracing anonymous packets to their approximate source," in *14th Systems Administration Conference*. Usenix LISA, Dec 2000, pp. 319–327.

[17] Kihong Park and Heejo Lee, "On the effectiveness of probabilistic packet marking for IP traceback under denial of service attack," in *Proc. of IEEE INFOCOM'01*, Anchorage, Alaska, 2001, pp. 338–347.

[18] Kihong Park and Heejo Lee, "On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets," in *Proc. ACM SIGCOMM '01*, San Diego, CA, Aug 2001, pp. 15–26.

[19] Thomer M. Gil, "MULTOPS: a data-structure for bandwidth attack detection," M.S. thesis, Vrije Universiteit, Amsterdam, Netherland, Aug 2000.

[20] A.Parekh and R.Gallagher, "A generalized processor sharing approach to flow control- the Single Node Case," in *Proc. of INFOCOMM*, May 1992, IEEE, pp. 915–924.

[21] P.Huang, A.Feldmann, and W.Willinger, "A non-intrusive, wavelet-based approach to detecting network performance problems," in *Proc. of Internet Measurement Workshop*, San Francisco, CA, Nov 2001, pp. 213–227.

[22] Donald B.Percival and Andrew T.Walden, *Wavelet Methods for Time Series Analysis*, chapter 4, Cambridge, Cambridge University Press, 2000.

[23] Donald B.Percival and Andrew T.Walden, *Wavelet Methods for Time Series Analysis*, chapter 8, Cambridge, Cambridge University Press, 2000.

[24] "National Laboratory for Applied Network Research (NLANR)- BUF site information," http://pma.nlanr.net/PMA/Sites/BUF.html, accessed on 01/05/2002.

VITA

Anu Ramanathan was born on November 2,1978 in Trivandrum, India. She received her B.Tech degree in Electrical Engineering from Indian Institute of Technology, Madras in July 2000. She joined the Department of Electrical Engineering at Texas A&M University to pursue her M.S degree in Computer Engineering in the fall of 2000. She worked with Dr.A.L.Narasimha Reddy, and her main interests are Network Security and Internet Traffic Analysis. Her address is Department of Electrical Engineering, Zachry Engineering Center, Texas A&M University, College Station, Texas 77843-3128.

The typist for this thesis was Anu Ramanathan.