ATTACK PROFILING

FOR

DDOS BENCHMARKS

by

Erinc Arikan

A thesis submitted to the Computer and Information Sciences Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Master of Science with a major in Computer Science

Summer 2006

ATTACK PROFILING

FOR

DDOS BENCHMARKS

by

Erinc Arikan

Approved: _____
Jelena Mirkovic, Ph.D.
Professor in charge of thesis on behalf of the Advisory Committee

Approved: _____
B. David Saunders, Ph.D.
Chair of the Department of Computer Science

Approved: _____
Thomas Apple, Ph.D.
Dean of the College of Arts and Sciences

Approved: _____
Daniel Rich, Ph.D.
Provost

ACKNOWLEDGMENTS

This project would not have been possible without the support of many people. Many thanks to my advisor, Jelena Mirkovic, who read my numerous revisions and helped make some sense of the confusion. Also thanks to my committee members, Adarshpal S. Sethi, Ching-Chung Shen who offered guidance. And finally, thanks to my family and friends who endured this long process with me, always offering support and love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Distributed denial-of-service is a serious problem and many defenses have been proposed to handle this threat. A common evaluation platform is needed to comparatively evaluate these solutions. This master's thesis is a part of work on the DDoS benchmarks project, which develops such an evaluation platform. The benchmarks contain three components: a) the set of typical attack scenarios that consist of attack, legitimate traffic and target network resource dimensions, b) the performance metrics that capture the impact of the attack and the defense effectiveness, and c) the testing methodology specification. This thesis describes the work on developing the attack dimension of DDoS benchmarks that summarizes typically seen attacks in today's Internet.

We harvest typical attack information from public packet traces. This approach is challenging due to short length of the traces, the presence of asymmetric traffic in the traces, random anonymization of addresses that hinders understanding of traffic's context and the unknown model of a legitimate user's behavior. An additional challenge lies in our goal to capture sophisticated attacks that are hard to detect, while minimizing false positives. We overcome these challenges through careful trace profiling and multiple-step processing. The attack samples are collected from traces in several steps: (1) we detect and filter one-way legitimate traffic from the traffic identified as likely attack, (2) we detect the attacks using multiple detection criteria, (3) we separate the legitimate from the attack traffic, and finally (4) we create attack samples from attack traffic and summarize attack features in a human-readable format and in a machine-readable format, convenient for application of clustering approaches.

All these steps are automatized via a set of tools, which facilitates easy collection of attack samples from a large number of public traces.

Our tools are tested on a set of synthetic attacks, on labeled traces (known to contain an attack) and on unlabeled traces, and we present the results of these tests in the thesis. In the case of the synthetic attacks, we manage to accurately identify all the attacks, even when they are stealthy or they have a small rate. Our tests on labeled traces detect all the attacks identified and labeled by other researchers, and a few more attacks that existed in the traces, but were not detected by other researchers. The tests on unlabeled 2 weeks long trace accurately identify several types of attacks including SYN floods, ICMP floods, UDP floods, TCP floods and attacks with invalid protocol types. We present detailed statistics on these attacks that indicate that the attackers are shifting from high-volume, easily noticed attacks to low-rate, stealthy attacks to avoid simple detection and defense approaches.

1 INTRODUCTION

1.1 Topic Introduction

The Internet currently connects millions of computers around the worldthat are running on different software and hardware platforms. Every day, our lives become more dependent on the Internet's services that simplify many daily tasks, and every day new users contribute to the Internet's growth. Maintaining correct operation, availability and security of the Internet services is critically important. Just like this high connectivity enables us to develop useful applications, it also provides means to malicious users to recruit and misuse many computers all over the world for various illegal activities.

One type of those malicious activities is *denial of service.* DoS (denial-of-service) attacks do not aim to alter data or gain unauthorized access, but instead they aim to cripple applications, servers and whole networks, disrupting legitimate users' communication [1]. The attacker either exploits some vulnerability in a target host or network, or he misuses many compromised machines to send huge traffic to the target. The denial of service effect is created by the attacker's traffic interfering with a target's operation, which makes it crash, hang, reboot or do useless work [1].

DoS attacks can be launched from either a single source or multiple sources. Multiple-source DoS attacks are called *distributed denial-of-service* (DDoS) attacks. DDoS attacks can sometimes employ up to 100,000 compromised computers to perform a coordinated and widely distributed attack [1]. Automated tools that can be easily used by an amateur to generate those attacks are publicly available with detailed specifications about how to use them.

Both DoS and DDoS are a large threat for online services, but DDoS attacks are more difficult to handle because their traffic can be made highly similar to the legitimate traffic. A disruption of a service has many financial consequences for online business. For instance, if an online bank becomes inaccessible for 2 hours this may mean losing business during the outage but also losing customers, prestige and reliability due to a damaged reputation, over a long time.

1.2 Thesis Motivation

Many approaches have been proposed to handle DoS and DDoS attacks. These approaches address diverse aspects of these complex threats, such as attack prevention, detection or response. Still, there is not a common, comprehensive methodology to evaluate an impact of a DoS attack on a given network, or the performance of a given defense. Such a methodology is needed for the following reasons:

To be able to protect systems from DDoS attacks, we need ways to characterize how dangerous the attack is, to estimate the potential damage/cost from the attack to a specific network (with or without defense).

Given many DDoS defenses, we need a common evaluation setting to evaluate and compare the performance of these defenses. These tests will also indicate a defense's weak features that need improvement.

This thesis is a part of a larger project that develops a common methodology for DDoS defense evaluation. The project consists of: (1) DDoS benchmarks that represent a set of scenarios to be used for defense evaluation, (2) a set of performance metrics that characterize an attack's impact and a defense's

performance, and (3) a detailed specification of evaluation methodology, which provides guidelines on using and interpreting benchmarking results.

The benchmark suite defines all the necessary elements to recreate relevant DDoS attack scenarios in a test bed setting. These relevant scenarios are divided into three categories: (1) typical attacks observed in today's Internet, (2) future attacks that have been proposed by researchers and that are more complex than existing attacks and (3) stress attacks that aim to create a ripple effect in the target network by hitting a critical service for this network's operation (e.g. routing).

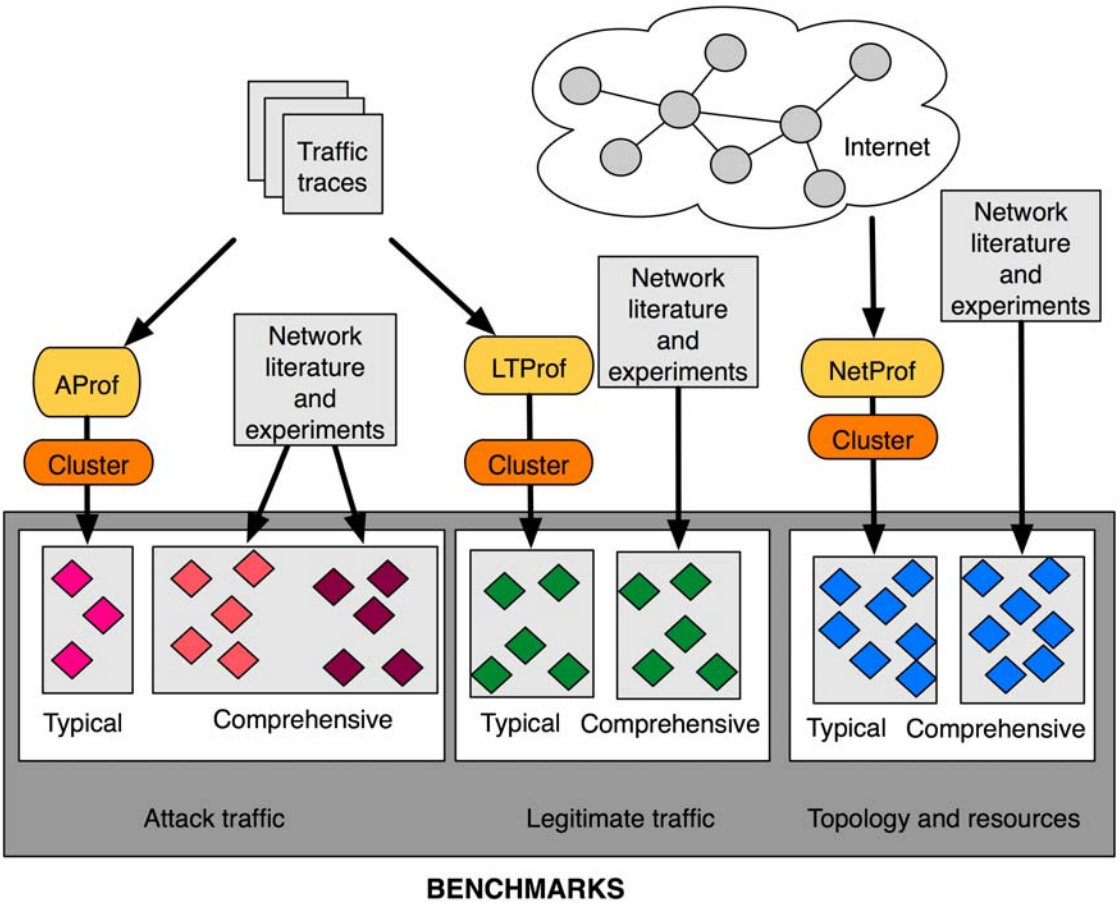

Figure 1 Benchmark components and their generation

DDoS benchmarks must specify all elements of an attack scenario that affect the damage to the target and the effectiveness of a given defense. These elements are: 1) attack traffic, 2) legitimate traffic, and 3) topology and resources. Figure 1 illustrates the benchmarks' components.

1.3 Thesis Statement

This thesis describes the work on the creating a collection of typical attacks, needed for "typical category" of the attack traffic component of DDoS benchmarks. This is accomplished by building a set of automatic tools that harvest this information from the public traffic traces – the AProf toolkit. The tools detect attacks in the trace, separate legitimate traffic going to the target from the attack traffic, and create attack samples that describe important attack features such as strength, type of the attack, number of sources, etc. Challenges addressed in this work are:

- How to collect attack information from traffic traces
- How to avoid false positives in face of asymmetric traffic monitoring, while still detecting sophisticated attacks, and
- How to create meaningful attack samples from the selected attack traffic.

Attack samples can further be clustered to yield representative typical attacks.

1.4 Key Contributions

There are some significant difficulties in creating a benchmark suite that will be able to capture all relevant DDoS attacks and later recreate them in a test bed. Since attackers continuously adjust their tools, relying on a set of attack features linked to a specific tool fails to detect novel attacks. Instead, we have to study attack dynamics and extract some fundamental features about the different types of DDoS

attacks that are invariant of attack tools in the use. The first contribution of this thesis is building of a set of automated tools that enable highly accurate attack detection and selection from a traffic trace.

There is very little information about prevalent attacks in today's Internet. This is mostly because there is no distributed monitoring infrastructure that could observe attacks in different parts of the Internet and correlate this information. Researchers have attempted to deduce Internet attack patterns from responses to spoofed traffic that reach a set of monitors that capture traffic sent to a dark address space (allocated to an organization but not used by a live host) [2]. This provides a valuable insight into attack patterns, but only for attacks that use spoofing. The second contribution of this thesis is that it provides means to deduce prevalent attack information by collecting attack samples from a vast number of publicly available traffic traces. We provide a preliminary step in this direction by applying our attack selection tools to several public traffic traces, and grouping selected attacks into meaningful clusters.

2 APROF TOOLKIT

AProf toolkit harvests attack information from traffic traces that are stored in libpcap format. Attack selection process is performed in the following four steps: (1) legitimate one-way traffic filtering, (2) attack detection, (3) separating the legitimate from the attack traffic and (4) attack feature selection. We describe the statistics we store and each step in attack detection and selection n the following text.

2.1 Data Structures

AProf toolkit harvests attack information from traffic traces that are stored in libpcap format. Attack selection process is performed in the following four steps: (1) legitimate one-way traffic filtering, (2) attack detection, (3) separating the legitimate from the attack traffic and (4) attack feature selection. We describe the statistics we store and each step in attack detection and selection in the following text.

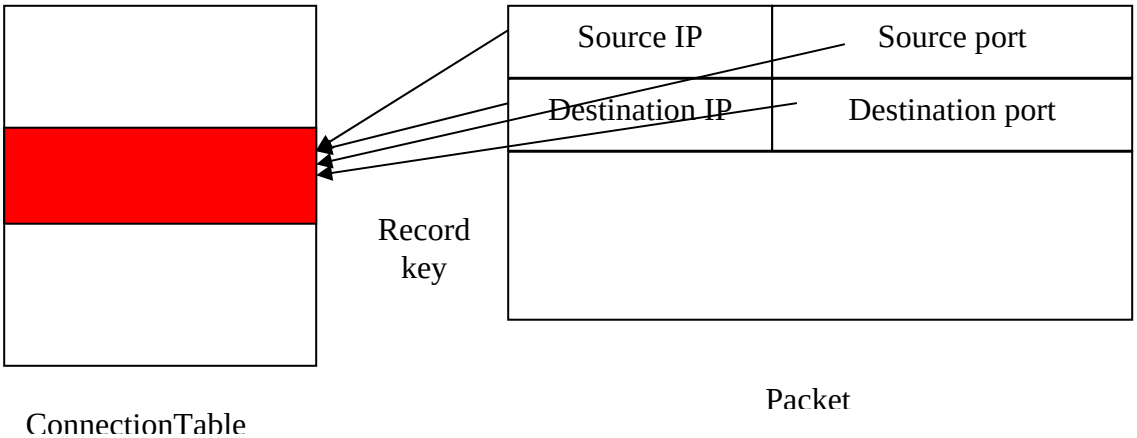

Figure 2 Packet header and the identification of the connection in the table

Each connection record stores the information shown in the Figure 3.

Connection Table Record

{

Sequence number of the first byte of the last packet
Sequence number of the last byte of the last packet
Legitimate flag
One-way flag
Number of packets sent
Number of packets received
Number of retransmissions
Number of fragmented packets
Suspicious points
Timestamp of the last activity
Number of bytes sent

}

Figure 3 Connection Table Record

A table called *DestinationTable* is used to keep information about every destination IP address observed in the trace and is accessed using the destination IP as a key. Figure 4 illustrates the DestinationTable structure.

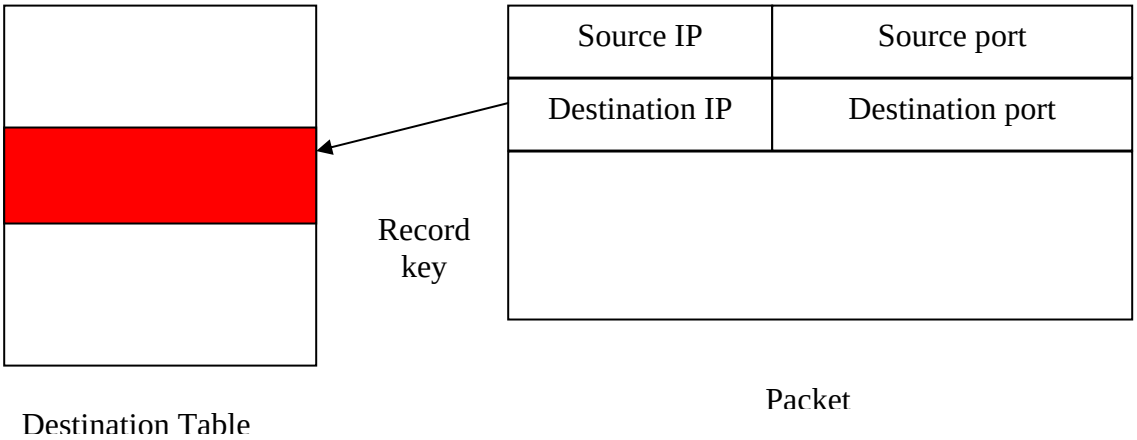

Figure 4 Packet header and the identification of the destination in the table

Each destination table record stores the information shown in Figure 5.

```
Destination Table Record
  {
      Array of attack flags
      Threshold rate for attack stop detection
      Number of SYN's  received
      Number of SYN's  sent
      Number of SYN ACK's  received
      Number of SYN ACK's  sent
      Bytes received for each protocol in the current and past window
      Bytes sent for each protocol in the current and past window
      Packets sent for each protocol in the current and past window
      Packets received for each protocol in the current and past window
      SYN-to-ACK ratio
      Packet-to-reply ratio
      Timestamp of the last activity
  }
```

Figure 5 Destination Table Record

2.2 One-way Traffic Detection

The attack detection criteria we will discuss in Section 2.3 relies mostly on detecting attacks via detection of aggressive flows that should, if they were legitimate, exhibit request/reply dynamics. Examples of flows that usually exhibit request/reply dynamics are TCP flows (data packets flow in one direction with ACKs in the opposite direction), DNS flows (requests to the server invoke its replies) and ICMP_ECHO flows (requests to the destination invoke its replies). Aggressive flows will send a lot of traffic to a single destination, persistently, even though the destination cannot provide sufficient replies to this traffic.

While detection criteria based on presence of aggressive flows work well on traces collected in a test bed setting, we have observed some problems when applying them to real traffic traces. The main source of the problems arises because of the presence of asymmetric traffic in the real trace. The Figure 6 illustrates one

monitoring scenario that will result in asymmetric traffic. There are 3 hosts (A,B,C) and two of them (A and B) have an asymmetric route. The monitor observes only A-to-B direction of traffic. This creates false positives with attack detection criteria that relies on the absence of replies to signal the onset of the attack (described in Sections 2.3.1 and 2.3.2. Our approach to avoid these false positives is to attempt to detect and filter legitimate one-way traffic from the trace, before proceeding with the attack detection.

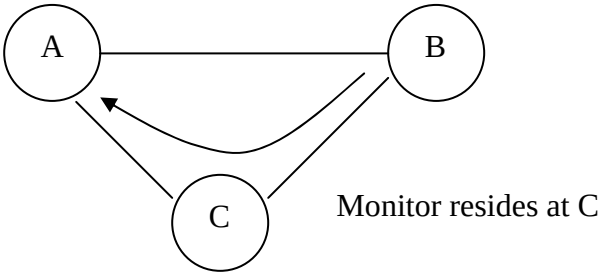

Figure 6 The problem of capturing network traces in presence of asymmetric routes

One-way traffic is detected by keeping track of each TCP connection, and pairing it with a connection in the reverse direction between the same host and port pairs. If such pairing is not possible, the connection is labeled as one-way and additional legitimacy tests are performed to ensure that it appears legitimate. In this case, source and destination IP from this connection will be recorded and all traffic between these two hosts will be regarded as one-way and removed from the trace. Please note that AProf may miss to identify as one-way non-TCP traffic between pairs that do not exchange any TCP traffic during the trace collection, which will necessarily result in some false positives.

As described in Section 2.1 we store traffic information at connection and destination granularity. For one-way traffic detection and filtering we use connection

records along with the additional hash table storing communicating host pairs. We call this table *PairTable* and its structure is shown in Figure 7. The record key is the tuple {source IP address, destination IP address}.

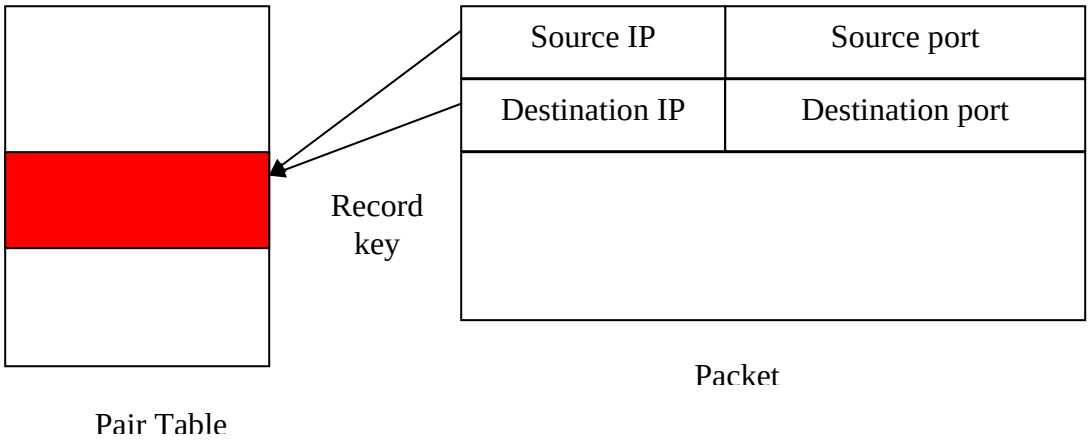

Figure 7 Packet header and the identification of the pair in the table

One-way flag of each TCP connection is initialized as TRUE, and when we encounter a new host pair, it is initialized as one-way pair. One-way flag of the TCP connection is set to FALSE if we observe any traffic in the reverse direction. In this case the associated host pair is marked as two-way.

In a legitimate TCP connection, sequence numbers should be monotonically increasing. TCP connections that contain packets that substantially diverge from this rule are generally an indication of some problem in TCP traffic, e.g., presence of retransmissions or fabricated TCP traffic. These connections may appear one-way but they do not indicate legitimate communication pattern and should not be removed from the trace. To detect anomalous TCP connections, we store the next expected sequence number for each connection. For each packet, we calculate the number of suspicious points as the absolute difference of the packet's sequence

number from the expected value. A connection, which collects more than a given threshold of suspicious points, is considered malicious and its legitimate flag is reset.

Another anomaly of interest in the sequence number space is the case of multiple repetitions of the same packet. Although we could detect this anomaly via suspicious point rule, we seek to minimize the detection delay by introducing a new detection criteria specific to this case. A repeated packet will increment the retransmission counter for its connection. When the number of retransmissions exceeds a threshold the connection is considered malicious and its legitimate flag is reset. Following pseudo code illustrates how a connection's flags are updated upon a packet's arrival.

```
/* Condition: packet P is a TCP packet and the first and the last
   byte's sequence number are recorded from the tcpdump output.
   When a new connection record is allocated, its legitimate and
   one-way flags are set to TRUE. A one-way flag is also set to TRUE
   for each newly allocated pair record. */

Find a connection C that P belongs to, or create it if not found
Find a connection RC that reverse connection belongs to, or create it if not found
Find a pair PR that P belongs to, or create it if not found
Find a reverse pair RPR that reverse source destination address pair belongs to, or
create it if not found
If (RC.oneway = TRUE) then
    RC.oneway = FALSE
    RPR.oneway = FALSE
If (C.legitimate = TRUE) then
    If (C.lastSeqNumber = P.firstSeqNumber or
    C.firstSeqNumber = INIT_SEQ_NUMBER) then
        C.lastSeqNumber = P.lastSeqNumber
        C.firstSeqNumber = P.firstSeqNumber
    else
    /* Retransmitted packets */
    If (C.firstSeqNumber = P.firstSeqNumber and
    C.lastSeqNumber = P.lastSeqNumber ) then
        C.retransmissions++
    else
        C.suspiciousPoints += P.firstSeqNumber – C.lastSeqNumber
    /* Update connection's sequence numbers, make sure that sequence
    numbers are always growing, ignore wraparound */
    If (P.firstSeqNumber > C.lastSeqNumber) then
        C.lastSeqNumber = P.lastSeqNumber
        C.firstSeqNumber = P.firstSeqNumber
    /* Check the legitimacy of the packet and connection */
    If (|C.suspicousPoints| > MAX_SEQ_DIFF) then
        C.legitimate = FALSE
        PR.oneway = FALSE
    If (C.retransmissions > MAX_RETRANSMISSIONS) then
        C.legitimate = FALSE
        PR.oneway = FALSE
```

Figure 8 Update of the one-way connections and pairs

A TCP connection is closed when we encounter a FIN or a RST or when there has been long period of inactivity. The closing of a connection removes its record from a connection table. A pair record is deleted from a pair table after a long period of inactivity or when we encounter the end of the trace. Pairs with one-way flag set are written to *oneway.dat*. The pseudo code for pair record deletion is shown in Figure 9.

```
/* Condition: A pair record P is to be deleted*/

If (P.oneway = TRUE) then
        Write P to the oneway.dat
Delete P from pair table
```

Figure 9 Deleting the pair record

To perform filtering on the original trace, a second pass is needed. We read source and destination IP address pairs from *oneway.dat* and remove all traffic between these pairs. The pseudo code in Figure 10 illustrates how that filtering takes place.

```
/* Condition: oneway.dat is opened. A one-way pair P is read line by line from
oneway.dat */

For each line in oneway.dat do
    Read P.address1
    Read P.address2
    insert P into pair table

/* Condition: P is the packet from the trace, and A is the pair key formed from the
P's source and destination IPs. */

For each P in trace do
    Find A in the pair table
    If (A is NOT_FOUND ) then
        Write P to distilled.trc
```

Figure 10 Filtering the packets

This process effectively filters out known-legitimate traffic from original trace and outputs the remaining traffic into the file *distilled.trc*.

2.3 Attack Detection

In the real world we experience a large variety of attacks; and many of them target different vulnerabilities. For attack detection, it is essential to analyze the dynamics behind the each type of attacks and to design some common set of detection criteria. In the following sections we describe our set of attacks detection criteria that relies, in most part, on detection of aggressive one-way communication patterns, and known abnormal packets. The attack detection criteria are applied to packets, connection and destination records formed by processing the *distilled.trc* file.

A packet belonging to a specific connection or going to a given destination is identified as malicious or legitimate using the detection criteria associated with: (1) this packet's header, (2) this packet's connection and (3) the features of the attack which was detected on the packet's destination. Each malicious packet is stored in an output file called *attack.trc*. In the following we describe in more detail each attack detection criteria we use.

2.3.1 Packet-to-reply ratio for TCP, ICMP and DNS traffic

Many network communications exhibit two-way communication patterns. Aggressive one-way traffic on a protocol or application that is expected to exhibit two-way communication pattern is regarded as a sign of an attack. For example, legitimate TCP traffic should invoke acknowledgments every few packets. If the flow of acknowledgments subsides (e.g., because the destination is under a DoS attack) legitimate traffic will take this as indication of congestion and will reduce its sending

rate. Persistent one-way TCP traffic is thus anomalous and regarded as a sign of an attack.

To detect aggressive one-way traffic we count the number of TCP, ICMP or UDP (DNS) packets sent and received on a connection and calculate their ratio. We smooth this ratio by calculating its weighted average over time to produce a stable measure that does not oscillate with short traffic bursts. If the increase in the ratio is consistent, it will exceed some given threshold in the end and we will detect the attack. Otherwise, it is only one-time fluctuation; exponentially weighted average will smooth this anomaly and will not produce false alarms. A connection whose ratio exceeded the threshold is considered malicious and all its packets are classified as attack. The pseudo code below illustrates the attack detection criterion for TCP traffic.

```
/* Condition: packet P is a TCP packet and source IP, destination IP and length of
the packet are recorded from the tcpdump output
Find a record D in the destination table by using
P's destination IP as a key, or create it if not found
Find a record S in the destination table by using P's source IP as a key,
or create it if not found.
/* Update the data statistics */
If (P.packetType = TCP) then
    D.TCPBytesReceived += length
    D.TCPPacketsReceived++
    S.TCPBytesSent += length
    S.TCPPacketsSent++
    D.TCPBytesReceivedArray[0] += length
    /* Check if source received enough to update TCP ratio */
    If (D.TCPPacketsReceived > MIN_PACKETS) then
        /* Calculate the new TCP ratio by weighted average */
        If (D.TCPPacketsReceived <> 0) then
            D.TCPRto = D.TCPRto*ALPHA + (D.TCPPacketsSent/
            D.TCPPacketsReceived) * (1-ALPHA)
        else
            D.TCPRto = D.TCPRto*ALPHA + (D.TCPPacketsSent / 1) * (1 –
            ALPHA)
        /* Check if destination is under aggressive TCP traffic */
        If (D.TCPRto > TCP_MAX_RTO) then
            attackType = TCP_RTO_TOO_BIG
```

Figure 11 Calculation of TCP ratio and detection of TCP-based attacks

2.3.2 TCP SYN Attacks

A TCP SYN attack exhausts victim's connection buffer by sending too many TCP SYN packets that invoke the victim to allocate a new record in the connection buffer for each SYN. The attacker never completes the three-way handshake, and the victim's state will eventually time out leading to deletion of stale records, but the timeout period is very long. A moderate-rate attack can thus keep

victim's resources effectively bound and deny service to new connections on the attacked port.

We detect TCP SYN attacks by keeping a record of smoothed SYN-to-ACK ratio for each destination of TCP traffic. When this ratio falls below some threshold, provided that there is more than some minimal number of outstanding SYN packets that have not been acknowledged, we raise the attack detection signal. The following pseudo code illustrates this attack detection criterion and the TCP connection state diagram is shown in Figure 13.

```
/* Condition: packet P is a TCP packet and source IP, destination IP and TCP
flags are recorded from the tcpdump output.
Find a destination record D in the destination table by using P's destination IP as a
key, or create it if not found.
Find a source record S in the destination table by using P's source IP as a key, or
create it if not found.
/* Update related SYN packet ratios */
 If (P.syn = TRUE) then
    If (P.ack = FALSE) then
          S.synSent++
          D.synReceived++
    else
          S.synAckSent++
          D.synAckReceived++
    /* Calculate the weighted average for syn Ratio */
    If (D.synAckSent <> 0) then
          D.synRto = D.synSentRatio* ALPHA + (D.synReceived /
          D.synAckSent) * (1-ALPHA)
    else
          D.synRto = D.synSentRatio* ALPHA + D.synReceived * (1-ALPHA)

    If (D.synRto > MAX_SYN_RTO and
    D.synReceived – D.synAckSent > SYN_SYNACK_DIFF) then
          attackType = SYN_FLOOD
```

Figure 12 Calculation of the SYN packet ratios and detection of TCP SYN flood

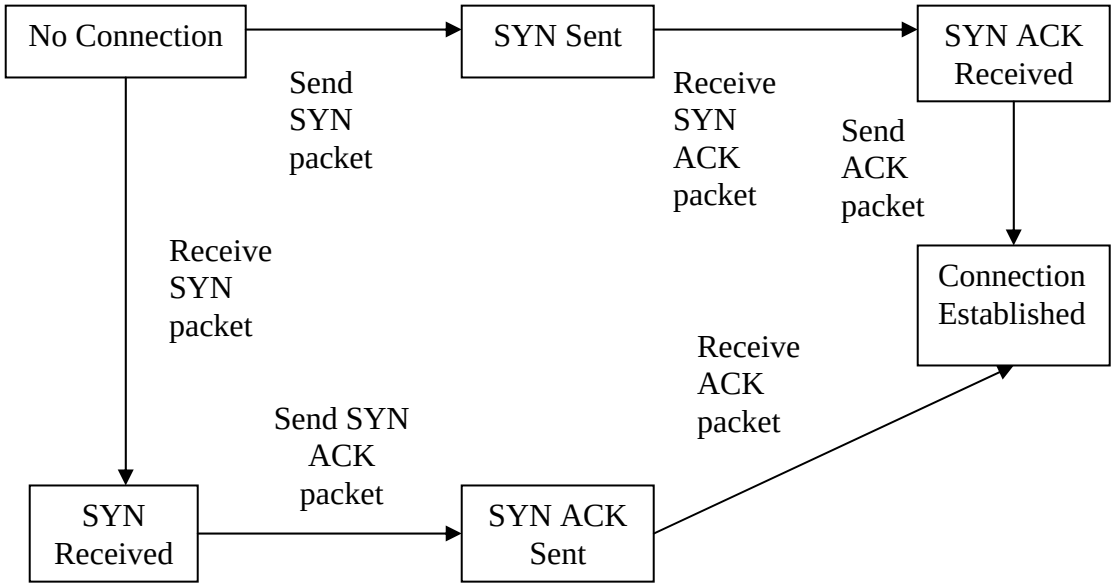

Figure 13 State diagram of a TCP connection

2.3.3 Mismatching Sequence Numbers

The legitimacy test for TCP connections, which we described in Section 2.2 for one-way traffic filtering, can also be used to detect troubled TCP connections that either have a lot of retransmissions or some fabricated TCP traffic. If the detection occurs, all TCP traffic to the destination of the detected attack is considered as part of the attack.

The pseudo code in Figure 14illustrates these two attack detection criteria.

```
/* Condition: packet P is a TCP packet and the first and the last
   byte's sequence number are recorded from the tcpdump output */
Find a connection C that P belongs to, or create it if not found
If (C.legitimate = TRUE) then
    If (C.lastSeqNumber = P.firstSeqNumber or
    C.firstSeqNumber = INIT_SEQ_NUMBER) then
        C.lastSeqNumber = P.lastSeqNumber
        C.firstSeqNumber = P.firstSeqNumber
    else
    /* Retransmitted packets */
    If (C.firstSeqNumber = P.firstSeqNumber and
    C.lastSeqNumber = P.lastSeqNumber ) then
        C.retransmissions++
    else
        C.suspiciousPoints += P.firstSeqNumber – C.lastSeqNumber
    /* Update connection's sequence numbers, make sure that sequence
    numbers are always growing, ignore wraparound */
    If (P.firstSeqNumber > C.lastSeqNumber) then
        C.lastSeqNumber = P.lastSeqNumber
        C.firstSeqNumber = P.firstSeqNumber
    /* Check the legitimacy of the packet and connection */
    If (|C.suspicousPoints| > MAX_SEQ_DIFF) then
        C.legitimate = FALSE
    If (C.retransmissions > MAX_RETRANSMISSIONS) then
        C.legitimate = FALSE
    If (C.legitimate = FALSE) then
        attackType = TOO_MANY_MISMATCHES
```

Figure 14 Calculation of the suspicious points and detection of traffic with too many mismatches

2.3.4 High Fragmentation Rates

Some DoS attacks send high volume of fragmented packets, either because they target a specific vulnerability in the victim's packet reassembly procedure or because many defense approaches cannot examine fragmented packets.

In the Internet the ratio of fragmented packets in all traffic is usually below 0.25% [5], [6]. We keep track of fragmented packet ratio for each connection and each destination, and we detect an attack if this ratio goes over 0.25% threshold. The pseudo code below illustrates this detection criterion:

```
/* Condition: Fragmentation information, source IP and destination IP are
recorded from the packet P */

Find a connection C that P belongs to, or create it if not found
C.numSent++
/* Check for fragmented packets */
If (P.fragmented ) then
    C.numFragmented++
    fragmentationRate = C.numFragmented / C.numSent
    If (fragmentationRate > MAX_FRAG_RATE and
     C.numSent > MIN_CONNECTION LIMIT) then
          AttackType = TOO_MANY_FRAGS
```

Figure 15 Detection of high fragmentation rate

2.3.5 Detection of ICMP or UDP Bandwidth Exhaustion Attacks

It is difficult to precisely detect attacks that generate high-volume ICMP or UDP traffic that exhausts victim's bandwidth. This is because both ICMP and UDP are used for various applications and a rate that is acceptable for some destination may be too large for another destination. For some ICMP and UDP packets that are expected to invoke reverse traffic (ICMP_ECHO and DNS) we can apply the similar packet-to-reply ratio check as used for TCP traffic, to detect aggressive one-way traffic. The pseudo code below illustrates this detection criterion for ICMP_ECHO traffic:

```
/* Condition: packet P is a ICMP packet and source IP and destination IP and
length of the packet are recorded from the tcpdump output */

Find a record D in the destination table by using P's destination IP as a key,
or create it if not found

/* Update the data statistics */
If (D.ICMPpacketsReceived  > MIN_PACKETS) then
    If (D.ICMPpacketsSent  > 0) then
        D.ICMPRto = D.ICMPRto * ALPHA + (D.ICMPpacketsReceived /
        D.ICMPpacketsSent) * (1-ALPHA)
    else
        D.ICMPRto = D.ICMPRto * ALPHA + D.ICMPpacketsReceived *
        (1 – ALPHA)
    /* Check if source received enough to update TCP ratio */
    If (D.ICMPRto > ICMP_MAX_RTO) then
        attackType = ICMP_FLOOD
```

Figure 16 Detection of ICMP flood

We detect other ICMP and UDP attacks using secondary observations of dynamics of TCP traffic going to the same destination. If we observe a sudden increase of ICMP/UDP traffic to a given destination, coupled with a sudden decrease (or congestion response) in TCP traffic to this same destination, we signal attack detection. To detect a rate change in a given protocol's traffic we keep counts of bytes sent to each destination for each protocol (TCP, UDP, ICMP) over a given time window. The media traffic naturally exhibits patterns that would trigger false attack detection – the control connection is established via TCP and the traffic on it subsides when one-way UDP traffic starts flowing. To minimize false positives we perform an additional check on traffic between UDP source and destination hosts. If there is established TCP connection between the same two hosts that exchange one-way UDP

traffic, this traffic is recognized as legitimate media traffic. The pseudo code below illustrates the detection criteria for UDP flood attacks.

```
/* Condition: P is a UDP packet, source IP, destination IP are recorded from the
tcpdump output, D.udpFlood flag is initially set to FALSE */
Find a record D in the destination table by using P's destination IP as a key,
or create it if not found
If  (D.UDPPacketsReceived > MIN_PACKETS) then
    /* Calculate average rate in current and past windows for UDP and TCP */
    curUDP =  rateInWindow (D.currentUDPWindow)
    oldUDP =  rateInWindow (D.pastUDPWindow)
    curTCP = rateInWindow (D.currentTCPWindow)
    oldTCP = rateInWindow (D.pastTCPWindow)

    /* Calculate UDP ratio */
    If  (D.UDPPacketsSent > 0) then
          UDPRto = D.UDPPacketsReceived /D.UDPPacketsSent
    else
          UDPRto = D.UDPPacketsReceived

    /* Check for decline in TCP rate coupled with increase in UDP */
    /* Check if UDP is sending enough and check if there's a TCP connection
between source and destination */
    If (curUDP > 2 * oldUDP and curTCP < 0.5 * oldTCP and
          findTCP(P.sourceIP, P.destinationIP) = FALSE and
          udprto > UDP_MAX_RTO) then
              attackType = UDP_FLOOD
          /* Start of the attack */
              /* Calculate stop rate for the attack */
              If (D.udpFlood = FALSE) then
                    D.stoprate = curUDP
                    D.udpFlood = TRUE
    /* Check the end of attack*/
    If (curUDP < D.stoprate) then
          D.udpFlood = FALSE
```

Figure 17 Detection of UDP flood

2.3.6 Malformed Headers

Some DoS attacks occur when a malformed header is sent to a vulnerable destination. One such attack is a Land attack [3] where the source IP and port are the same as the destination IP and port. We detect this attack by simply checking the information in the packet's IP header. The pseudo code below illustrates this detection criterion:

```
/* Condition: Source IP and destination IP are recorded from the packet P */

/* Check for same source IP and destination IP */
If (P.sourceIP = P.destinationIP and P.sourceport = P.destport) then
    attackType = MALFORMED_HEADER
```

Figure 18 Detection of packets with malformed headers

2.3.7 Invalid Protocol Numbers

Some DDoS attacks have been observed to use invalid protocol numbers in the IP header. We detect this type of attacks by checking the protocol number in a packet's IP header against a list of known protocol numbers obtained from IANA [4]. Packets with invalid protocol numbers are considered attack packets. The pseudo code below illustrates this detection criterion:

```
/* Condition: Protocol number is recorded from the packet P */
If (P.protocolNo is not in the list from IANA ) then
                attackType = NONEXISTING_PROTO
```

Figure 19 Detection of packets with invalid protocol numbers

2.3.8 TCP No-Flag Attack

TCP no-flag attack exhausts victim's bandwidth buffer by sending large TCP packets without any flag set and without any sequence or acknowledgement numbers. TCP protocol layer rejects these packets but they consume network bandwidth and CPU time. The attack using TCP no-flag packets will eventually be detected via packet-to-reply ratio but we can significantly reduce detection time by custom-tailoring a detection criterion for these attacks. The pseudo code below illustrates this detection criterion:

```
/* Condition: packet P is a TCP packet and source IP, destination IP and TCP
flags are recorded from the tcpdump output */

/* Update related SYN packet ratios */
 If (P.syn = FALSE and P.ack = FALSE and P.rst = FALSE and
     P.psh = FALSE and P.fin = FALSE) then
          attackType = NO_FLAG
```

Figure 20 Detection of TCP packets with no flag set

2.3.9 Presence of IP Spoofing

IP spoofing denotes the act of a packet sender putting another node's IP address into the source IP field of the packets it generates. IP spoofing hinders attack detection and response and its presence indicates malicious traffic going to the destination. Unfortunately, there is no reliable approach to detect IP spoofing in trace traffic, especially because public traces tend to be anonymized so their address information is altered. AProf tool can detect spoofing of reserved addresses in the trace, which may indicate presence of malicious traffic if the trace is not anonymized and if it does not capture any internal traffic that naturally carries reserved IP

addresses. We facilitate activation/deactivation of this attack detection criterion through a command-line flag.

The reserved addresses are designated by the Internet Assigned Numbers Authority (IANA) [4] and should not be present in transit traffic but are allowed in intra-network traffic. These addresses are:

- Addresses from 10.0.0.0 to 10.255.255.255
- Addresses from 172.16.0.0 to 172.31.255.255
- Addresses from 192.168.0.0 to 192.168.255.255

The pseudo code below illustrates the detection criteria for reserved IP spoofing:

```
/* Condition: P denotes the packet, source IP, destination IP and length of the
packet are recorded from the tcpdump output */

/* Check for malicious usage of reserved IP addresses */
If (reserved(P.sourceIP) = TRUE or reserved(P.destinatiionIP) = TRUE)
    AttackType = RES_SPOOFING
```

Figure 21 Detection of reserved address spoofing

2.4 Separating Legitimate and Attack Traffic

Each packet is classified as part of legitimate or attack traffic as soon as it is read from the trace, using the attack detection criteria described in Section 2.3. This classification is done by applying each of the listed attack detection steps and raising alarms. Packets that pass all steps without raising an alarm are considered as legitimate packet. Packets that raise one or more alarms are considered attack packets. To deal with cases when more than one alarm is raised we use precedence rules to decide which alarm carries higher confidence, i.e. lower chance for false positives.

Only a single alarm will eventually be chosen for each destination under attack, using these precedence rules. The underlying assumption is that no destination will be target of more than one attack at a given time. The

Figure 22 illustrates precedence rules used to select an alarm that carries higher confidence. Newly detected alarms are written to a file called *victim.dat*.

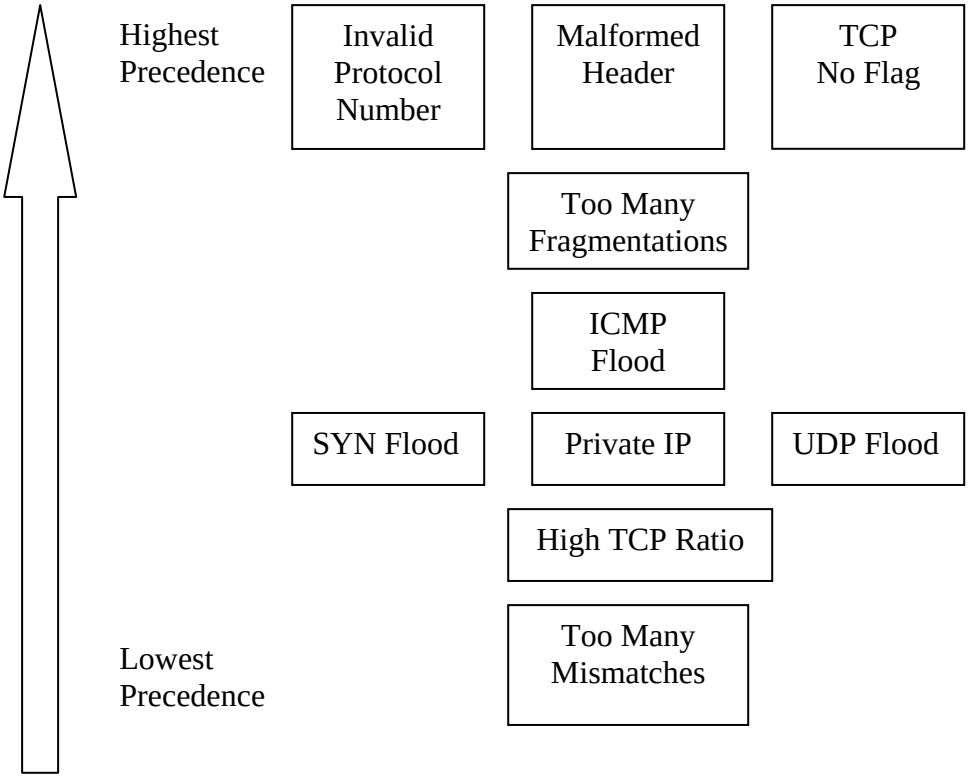

Figure 22 Precedence rules for the alarm selection

Once an alarm is raised for a certain destination, the packets going to this destination will be considered as attack if their type matches the type of the detected attack. For instance, if a SYN flood is detected only SYN packets will be considered

as part of the attack. Attack packets will be written to *attack.trc* and legitimate packets are stored in *legitimate.trc*.

2.5 Attack Feature Selection

Attack features are selected by pairing the packets from the *attack.trc* with the alarms from *victim.trc* and profiling the packets to select the following features:

- Beginning and the duration of the attack
- Attack type
- Type of spoofing
- Packet and byte rate per second
- Number of source IPs if there is no spoofing detected
- Number of source ports if there is no spoofing detected
- Number of destination ports

Attack samples are written into two output files:

- Human readable file *human.dat* with alerts followed by a snippet of 100 attack packets
- Machine readable file *machine.dat* with alerts only.

2.5.1 Detection of IP Spoofing

We detect two categories of spoofing commonly used in attacks: (1) random spoofing chooses addresses from the entire IPv4 address space at random and (2) subnet spoofing chooses random addresses from a given subnet. We check for existence of random spoofing and subnet spoofing while we are generating attack samples and attack statistics. Both spoofing types are detected by observing the distribution of source addresses in the entire IPv4 space. In normal, non-spoofed

traffic, source addresses will be clustered in a few groups in this space, since the allocated IP addresses are clustered in such manner. In randomly-spoofed traffic source addresses will be uniformly distributed over the IPv4 address space, and in case of subnet-spoofed traffic they will be concentrated in one cluster. We approximate the distribution of source IP addresses by keeping count of appearances of octet values in each address, as illustrated in Figure 23. Each octet has 256 possible values, and we keep the count of appearances of each value, for each octet. We signal random spoofing if number of values that have appeared in each octet at least once (we call this *number of hits*) exceeds some expected number given the length of the attack trace. Expected number of hits for random spoofing can be found as follows using the Bernoulli distribution:

$$E[\text{number of hits for each octet in random spoofing}] = 256*(1 - (1-1/256)^N)$$

where N is number of packets in the trace

Detailed explanation:

$$P(\text{Slot being hit by a packet}) = 1/256$$

$$P(\text{Slot not being hit by a packet}) = (1- 1/256)$$

$$P(\text{slot not being hit by N packets}) = (1- 1/256)^N$$

$$P(\text{slot being hit at least once after N packets}) = (1 - (1- 1/256)^N)$$

There are 256 slots for an octet thus:

$$E[\text{number of hits for each octet in random spoofing}] = 256*(1 - (1-1/256)^N)$$

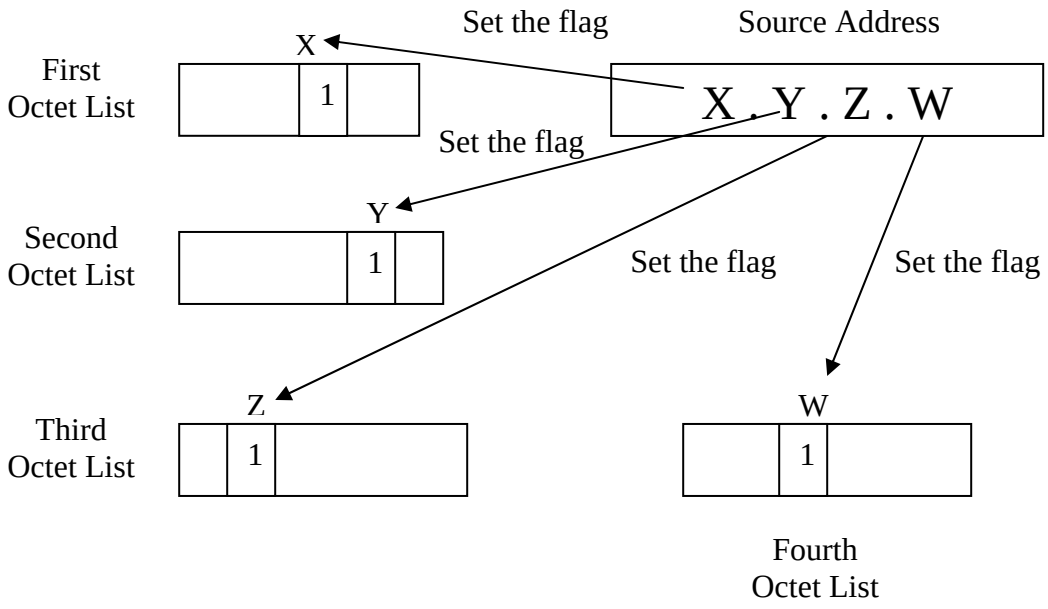

Figure 23 Update of the octet lists

To detect subnet spoofing we use same methodology, but we check if octets 1-3 have a small number of slots hit, while the last octet has all slots hit. This assumes that subnet spoofing is done in /24 space. The pseudo code below illustrates detection of these two spoofing types:

```
/* Condition: P denotes the packet, source IP, destination IP are recorded from the
tcpdump output, all the elements in the arrays are initialized to 0 and all the packets
for a destination are read consecutively */
Separate source IP into its octets
NUM_ADDRESSES = 256
firstOctetElements [firstOctet]++
secondOctetElements [secondOctet]++
thirdOctetElements [thirdOctet]++
fourthOctetElements [fourthOctet]++

/* Calculate the expected number of slots hit by randomly spoofed packets */
phit = pow((1- 1 / NUM_ADDRESSES),number of packets)
Ehit  = NUM_ADDRESSES*(1-pow(1-phit,NUM_ADDRESSES))-1
FirstHit,secondHit,thirdHit,fourthHit = 0

/* Calculate number of hits for each octet */
For ( i = 0 to i = NUM_ADDRESSES-1) do
    If (firstOctetElements[i] > 0) then
        firstHit++
    If (secondOctetElements[i] > 0) then
        secondHit++
    If (thirdOctetElements[i] > 0) then
        thirdHit++
    If (fourthOctetElements[i] > 0) then
        fourthHit++

/* Check for random spoofing */
If (firstHit > Ehit and secondHit > Ehit and
    thirdHit > Ehit and fourthHit > Ehit) then
        spoofingType = RANDOM_SPOOFING
else
    /* Check for subnet spoofing */
    If (firstHit < Ehit and secondHit < Ehit and
    thirdHit < Ehit and fourthHit > Ehit) then
        spoofingType = SUBNET_SPOOFING
```

Figure 24 Detection of random spoofing and subnet spoofing

2.6 Programs and the Flow of the Data

Figure 25 shows the programs in the AProf toolkit and the flow of data. All files with the extension .trc are in libpcap binary format, while files with the extension .dat are plain text files. We provide more details about each component in the following sections.

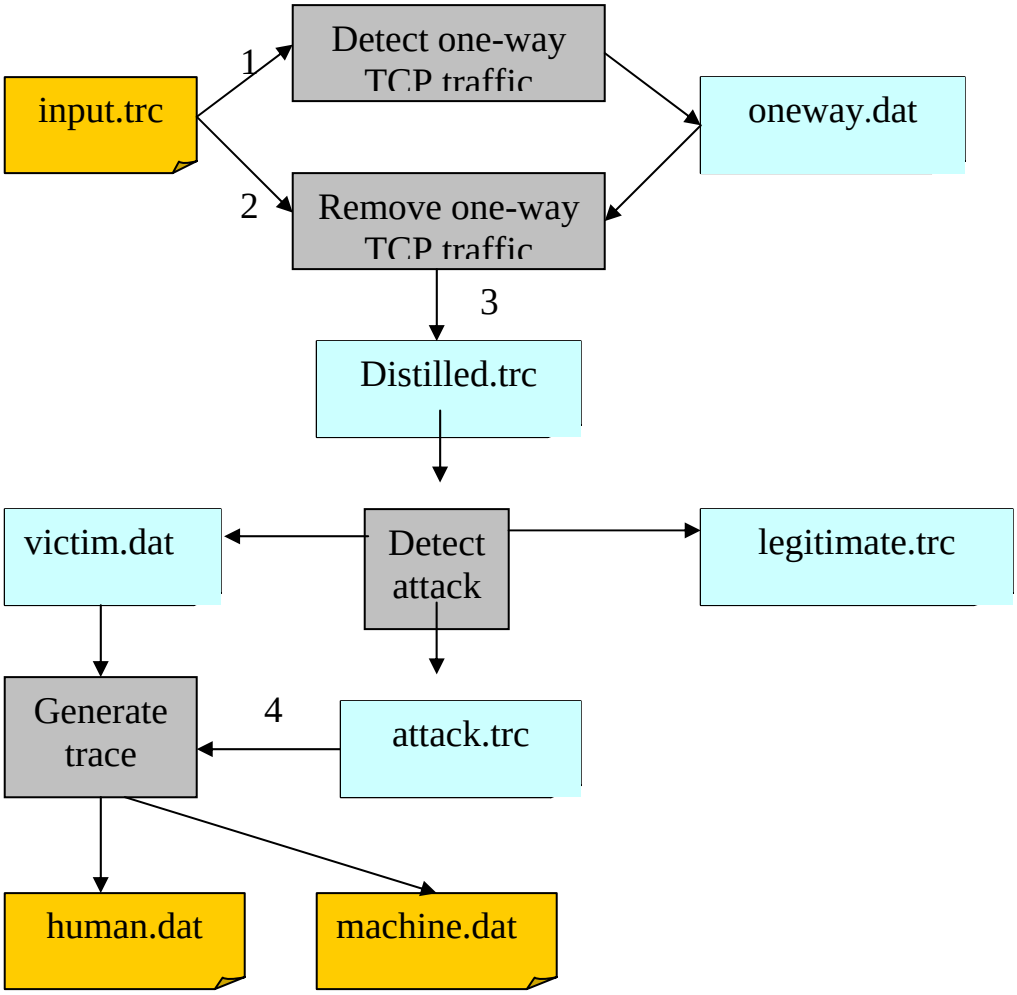

Figure 25 Components of the AProf toolkit and the flow of data

2.6.1 One-way Detector Tool

One-way detector is the tool, which accomplishes the first pass on the traffic trace to detect one-way legitimate connections and writes them to *oneway.dat*. The trace should be in libpcap format. One-way detector uses two hash tables to keep information about connections and pairs as explained in Section 2.1 to detect one-way pairs and these pairs are written to *oneway.dat.*

2.6.2 One-way Remover Tool

One-way remover tool uses the original trace and *oneway.dat*, and removes the traffic between pairs in *oneway.dat* generating the refined traffic trace *distilled.trc*.

2.6.3 Detector Tool

Detector tool uses the attack detection criteria that are explained in Section 2.3 to identify the attacks and victims and to separate the legitimate and the attack traffic. Attack type and victim information are written to *victim.dat*, attack packets are written to *attack.trc* and the legitimate traffic is stored in *legitimate.trc*.

2.6.4 Sample Generator Tool

Sample generator tool associates attack alerts from *victim.dat* with the relevant packets in *attack.trc* and creates one-alert-per-attack file with attack attributes. It generates a human-readable output *human.dat* with alerts and a snippet of 100 attack packets, which allows for quick human inspection to detect false positives. It also generates machine-readable output *machine.dat*, which only contains alert information and is more suitable for further processing, such as alert clustering to find prevalent attacks. Figure 26 shows one snippet from the file *human.dat*.

```
1025390099.800953  proto TCP packet 52.230.58.120:1534 > 52.230.211.105:53 seq 3393850781 ack 0 len
44
1025390099.801354  proto TCP packet 52.230.58.120:1534 > 52.230.211.105:53 seq 3393850781 ack 0 len
44
1025390099.801940  proto TCP packet 52.230.58.120:1534 > 52.230.211.105:53 seq 3393850781 ack 0 len
44
1025390099.802522  proto TCP packet 52.230.58.120:1534 > 52.230.211.105:53 seq 3393850781 ack 0 len
44
1025390099.803104  proto TCP packet 52.230.58.120:1534 > 52.230.211.105:53 seq 3393850781 ack 0 len
44
1025390099.803691  proto TCP packet 52.230.58.120:1534 > 52.230.211.105:53 seq 3393850781 ack 0 len
44
1025390099.804104  proto TCP packet 52.230.58.120:1534 > 52.230.211.105:53 seq 3393850781 ack 0 len
44
1025390099.804677  proto TCP packet 52.230.58.120:1534 > 52.230.211.105:53 seq 3393850781 ack 0 len
44
1025390099.805273  proto TCP packet 52.230.58.120:1534 > 52.230.211.105:53 seq 3393850781 ack 0 len
44
1025390099.805845  proto TCP packet 52.230.58.120:1534 > 52.230.211.105:53 seq 3393850781 ack 0 len
44
1025390066.805695 attack on 52.230.211.105 type SYN flood duration 33.000150 pps 6.969665
Bps 345.452976 packets in trace 230 sources 1 spoofing NO_SPOOFING source ports 1 dst ports 1
/* End of the attack */
```

Figure 26 Snippet from human.dat

3 EVALUATION RESULTS

We apply the AProf toolkit to three sets of traffic traces, with different evaluation goals:

(1) We generate a variety of synthetic attacks in Emulab test bed, record their traces and apply AProf to these traces. These tests are performed to evaluate the accuracy of AProf in detecting attacks and separating the legitimate from the attack traffic.

(2) We apply AProf tool to Los Nettos traces that contain attacks detected by other researchers. These tests are performed to evaluate AProf performance on real traffic traces, and to validate its results against the results obtained by other researchers on attack detection.

(3) We apply AProf to NLANR Auckland traces [7] with the goal to gather more attack samples and gain an insight about the prevalent DoS attacks in the real world.

3.1 Tests with synthetic attacks in Emulab

We initially tested Aprof toolkit with traces that are generated using Emulab [8]. Emulab is a shared testbed hosted by University of Utah, which provides a wide range of experimental environments for live traffic tests, simulation and virtualization. Its ns-based tools and web-based GUI facilitate easy remote configuration of experiments, including the setup of various topologies with diverse machine and link features. Users get a sudoer access to each machine, and the experiment's traffic can be completely contained within the experimental topology, and prevented from reaching the outside world, using Emulab-provided firewalls. There is a one to one mapping between the real machines and addresses and the

experiment nodes are completely isolated from the outside world, except for the ssh access to drive the experiment.

Traffic traces are collected using two different topologies. Experiments that simulate typical attack cases that target bottleneck bandwidth and that fabricate packet information use the simple topology in Figure 27**.** Experiments with attack cases that target the CPU instead of bandwidth require more attackers and will use following topology in Figure 28.

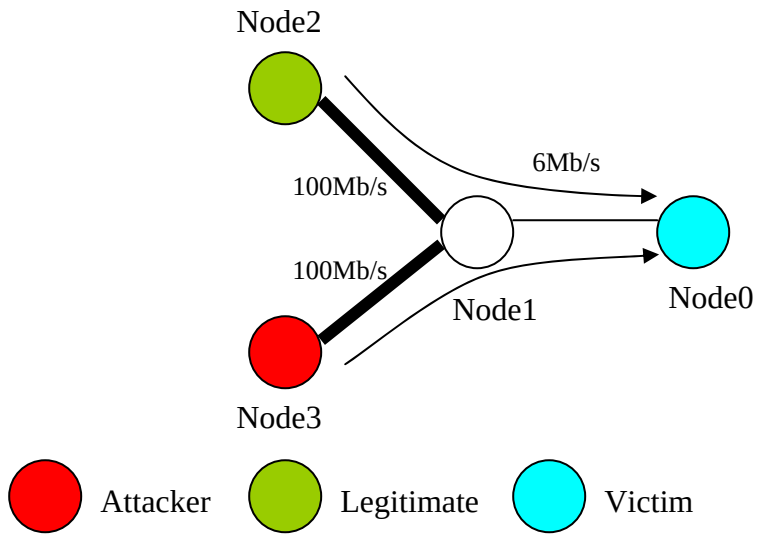

Figure 27 The simple topology used to generate traffic traces for bandwidth exhaustion and fabricated packet attacks.

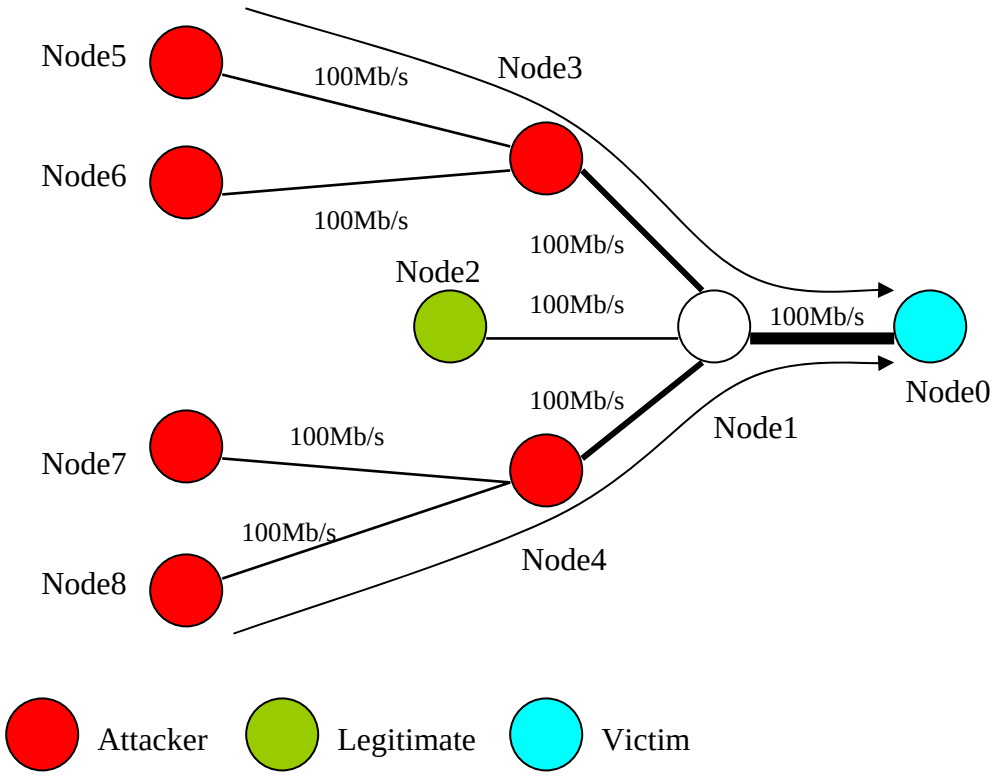

Figure 28 The topology that is used to generate traces for CPU-extensive attacks

In both topologies, node2 plays the role of a legitimate machine, and communicates with the attack's target – node0. The legitimate traffic consists of 10 parallel TCP connections that transfer a 4 MB file via scp simultaneously and continuously.

In bandwidth-exhaustive and fabricated experiments we capture the traffic on the link between node1 and node0. This link is a bottleneck link for the topology in Figure 27**,** because of the difference between incoming bandwidth to node1 and the outgoing bandwidth from node1. The attack that comes from node3 causes losses and delays when it interacts with the legitimate traffic. In CPU-extensive attacks,

traffic is again captured on the link between node1 and node0 but this link is not a bottleneck link.

To generate the attack traffic, we use a flooding tool called dos. This is a custom-made tool, which can generate TCP SYN flood, UDP flood, ICMP flood at any rate with desired spoofing type, using a custom packet length and duration. With a small modification in the source code, it is possible to send packets with non-existing protocol numbers, TCP packets with no flag set and fragmented packets. In the experiments, we generate the following types of attacks:

- TCP traffic with random sequence numbers (targeting the detection criterion explained in section 2.3.3)
- TCP flood (targeting the detection criterion in section 2.3.1)
- TCP SYN flood (targeting the detection criterion in section 2.3.2)
- ICMP flood (targeting the detection criterion in section 2.3.5)
- Spoofing (targeting the detection criterion in section 2.3.9)
- Invalid protocol number (targeting the detection criterion in section 2.3.7)

Other attack types have features similar to the attacks that we tested, so we can infer what the detection performance would be for these attacks from the test results.

3.1.1 TCP Traffic with Random/Constant Sequence Numbers

This synthetic test uses the topology given in the Figure 27**.** Attacker, node3 sends TCP traffic with constant sequence numbers with 2M bytes per second to the victim, which is node0. Traffic is captured on the link between node3 and node1. All the attack traffic consists of TCP packets that have the same sequence number throughout the trace and are sent from the same source IP and port to same destination

and destination port. This attack tests the legitimate traffic detection criterion from Section 2.3.3. Results are shown in Table 1.

Time attack started	1151900617.991576
Time detection started	1151900618.127747
Detection delay in seconds	0.136171
Detection delay in packets	100

Table 1 Detection delay for const sequence numbered TCP traffic of rate 2M bytes/sec

Victim IP	Attack Type	Duration	Packet Rate	Bytes Rate	Number of Packets
191.168.1.2	Seq. num mismatch	60.035	741.437	741436.683	44512

Table 2 Attack attributes for TCP flood with constant sequence numbers and rate 2M bytes/sec

Victim IP	Spoofing	Number of Sources	Number of source ports	Number of destination ports
191.168.1.2	No	1	1	1

Table 3 Attack attributes for TCP flood with constant sequence numbers and rate 2M bytes/sec (cont)

Number of false positives	Number of false negatives
0	100

Table 4 False positives and negatives for TCP flood with constant sequence numbers and rate 2M bytes/sec

The detection delay is very small, indicating that AProf toolkit is detecting the attack in a fast manner. Table 2 and Table 3 show the attributes of a generated attack sample and Table 4 shows us false positives and negatives (in packets). We see that the estimated attack rates are lower than real ones. This is because the bottleneck link

limits the rate of attack that reaches the victim, and this is reflected in the trace that is collected on that link.

3.1.2 TCP flood

This test case also uses the simple topology given in the Figure 27. Node3 sends TCP flood with the rate of 2M packets per second from random port numbers to node0's random ports, with randomized sequence numbers This simulation tests the detection criterion in section 2.3.1.

Time attack started	1151907605.398011
Time detection started	1151907605.406132
Detection delay in seconds	0.008121
Detection delay in packets	6

Table 5 Detection delay for TCP flood of rate 2M bytes/sec

Victim IP	Attack Type	Duration	Packet Rate	Bytes Rate	Number of Packets
191.168.1.2	TCP-ratio	60.169435	738.381538	738381.538435	44428

Table 6 Attack attributes for TCP flood of rate 2M bytes/sec

Victim IP	Spoofing	Number of Sources	Number of source ports	Number of destination ports
191.168.1.2	No	1	32256	32296

Table 7 Attack attributes for TCP flood of rate 2M bytes/sec (cont'd)

Number of false positives	Number of false negatives
0	186

Table 8 False positives and negatives for TCP flood of rate 2M bytes/sec

Table 5 shows detection delay for the TCP flood. Table 6 and Table 7 show the attack characteristics. Again the attack rate is estimated smaller than the real rate because of

the bottleneck link drops, and the port numbers indicate that TCP flood is going between multiple ports. Table 8 shows the information about the false positives and negatives (in packets), which is very low.

3.1.3 TCP SYN Flood

For the TCP SYN flood we use the topology from Figure 28. The reason for that is TCP SYN floods target the victim node's CPU power. Enforcing a bandwidth limit reduces the number of packets that are received by the victim and lessens the attack's power. Therefore to be able to measure the attack effect, we use a topology with multiple attackers (so they can generate sufficient packets-per-second rate) and without a bottleneck link. There are 6 attacker nodes and a legitimate node that are trying to communicate with the victim at the same time. 20 seconds after the legitimate traffic started, attacker nodes start sending attack traffic that lasts 60 seconds. All 6 attackers flood the victim at the maximum possible rate allowed by their operating system. Following tables illustrate the results that are given by AProf by running the toolkit on the collected traces.

Time attack started	1152019536.199468
Time detection started	1152019536.199475
Detection delay in seconds	0.000007
Detection delay in packets	4

Table 9 Detection delay for TCP SYN flood

Victim IP	Attack Type	Duration	Packet Rate	Bytes Rate	Number of Packets
191.168.1.2	SYN flood	42.492	587.127	23485.075	24948
191.168.1.2	TCP-ratio	143.112	352.508	14109.251	50448
191.168.1.2	SYN flood	200.071	722.589	28903.541	144569

Table 10 SYN flood attributes

Victim IP	Spoofing	Number of Sources	Number of Source Ports	Number of Destination Ports
191.168.1.2	No	2	20708	1
191.168.1.2	No	7	35095	1
191.168.1.2	No	6	58353	1

Table 11 SYN flood attributes (cont'd)

Number of false positives	Number of false negatives
64	20590

Table 12 False positives and negatives for SYN flood

Table 9 shows the detection delay for the SYN flood. This shows that when the SYN flood starts, SYN flood alarm signal raised almost simultaneously because of the inability of the victim to respond to the attack. Examining the trace, we notices that as the time passes, replies to some SYN packets start to come in chunks, which leads to decrease in the SYN-to-SYNACK ratio and may turn the alarm off. Because of this some attack packets will be classified as legitimate leading to false negatives as illustrated in Table 12. Table 10 and Table 11 show the attacks found by the AProf. Three attack instances are detected for the same attack. This is a combined effect of a slowdown the attack packets experience at the software routers, who can only process them at a limited rate, and the inability of the tcpdump tool to capture high packet rates accurately.

3.1.4 ICMP Flood

ICMP flood experiments use the topology in Figure 28, because they also target the victim's CPU power instead of bottleneck bandwidth. In the experiments each attacker sends ICMP echo messages, trying to overwhelm the victim by sending

at a high rate. Just like in TCP SYN flood experiments, there are 6 attackers that flood the victim at 10M bytes/sec each. ICMP packets are 1KB long.

Time attack started	1152033938.090070
Time detection started	1152033938.090213
Detection delay in seconds	0.000143
Detection delay in packets	4

Table 13 Detection delay for ICMP flood w/ rate 10M bytes/sec

Victim IP	Attack Type	Duration	Packet Rate	Bytes Rate	Number of Packets
191.168.1.2	ICMP flood	60.867	1700.72	1700720.143	103517

Table 14 ICMP flood attributes with rate 10M bytes/sec

Victim IP	Spoofing	Number of Sources	Number of Source Ports	Number of Destination Ports
191.168.1.2	No	6	0	0

Table 15 ICMP flood attributes with rate 10M bytes/sec(cont'd)

Number of false positives	Number of false negatives
1082	61008

Table 16 False positives and negatives for ICMP flood attributes with rate 10M bytes/sec

Table 13 shows the detection delay for ICMP attacks. We see that the detection occurs almost immediately, because the attack is strong. Since we reset the collected statistics periodically, some attack traffic may be marked as legitimate for a brief time following the reset interval. Table 14 and Table 15 show the attributes of the identified attack. The packet and byte rates are smaller than actually generated, again because of the inability of the tcpdump tool to capture high packet rates accurately. The Table 16 shows the number of false positives and false negatives in packets.

3.1.5 Spoofing Detection

Even though spoofing is not an attack type, we test whether we can properly detect the type of spoofing used in synthetic attacks, to evaluate the correctness of our sample generation tool. For these tests, we generated UDP flood attacks on the topology from Figure 27. These attacks target the bottleneck link. We used the random and the subnet spoofing, with different attack rates. In all cases AProf correctly identified the type of spoofing. Table 17 and Table 18 show the correct identification in one test case.

Victim IP	Spoofing	Number of Sources	Number of Source Ports	Number of Destination Ports
191.168.1.2	Random	35689	27523	27528

Table 17 UDP flood attributes with rate 2M bytes/sec and random spoofing

Victim IP	Spoofing	Number of Sources	Number of Source Ports	Number of Destination Ports
191.168.1.2	Subnet	253	26403	26392

Table 18 UDP flood attributes with rate 2M bytes/sec and subnet spoofing

3.1.6 Invalid Protocol Number

For this test, we use the topology from Figure 27. The attacker (node3) sends packets to victim (node0) with protocol number 155 with rate of 600K bytes per second. The detection of this and other invalid packet attacks (such as TCP no-flag and malformed header) is very simple as the first invalid packet will raise an alarm

Time attack started	1151913776.598986
Time detection started	1151913776.598986
Detection delay in seconds	0
Detection delay in packets	0

Table 19 Detection delay for invalid protocol attack of rate 600K bytes/sec

Victim IP	Attack Type	Duration	Packet Rate	Bytes Rate	Number of Packets
191.168.1.2	Non-existing protocol	60.0003	598.047	598046.789	35883

Table 20 Attack attributes for invalid protocol attack of rate 600K bytes/sec

Victim IP	Spoofing	Number of Sources	Number of source ports	Number of destination ports
177.84.54.80	No	1	0	0

Table 21 Attack attributes for invalid protocol attack of rate 600K bytes/sec (cont'd)

Number of false positives	Number of false negatives
0	0

Table 22 False positives and negatives for invalid protocol attack of rate 600K bytes/sec

Table 19 shows that there is no delay in detecting these attacks, and Table 20 and Table 21 show that all attack attributes are estimated correctly. This high accuracy is in part due to lower attack rate so we were able to capture the entire attack in the trace. Table 22 shows that there are no false positives or negatives. This is to be expected as each packet can be properly evaluated for validity.

3.2 Los Nettos Traces

After identifying the attacks in the test bed, we needed to test our AProf toolkit with real traces. To perform this task, we used Los Nettos traces. Los Nettos is

a regional ISP in Los Angeles relaying traffic for several academic and commercial institutions. Traces were captured by ISI researchers: Alefiya Hussain, John Heidemann, and Christos Papadopoulos, from June 2002 to November 2003 [15]. We were able to obtain access to only 3 of these traces, each containing a single identified attack and possibly more non-identified attacks. The goal of our evaluation was to verify that AProf can detect labeled attacks and potentially discover new attacks, not detected by the ISI researchers. The traces we used for this evaluation were anonymized using prefix-preserved anonymization. Below we provide more details about each trace:

(1) Trace 4 contains a reflector attack that sends echo reply packets to a victim at anonymized IP address 87.134.184.48.

(2) Trace 18 contains a TCP no-flag attack, to IP address 89.1.89.241.

(3) Trace 29 contains IP-proto 255 attack, to IP address 88.89.192.119.

We next present our detection results and comment on them in detail.

3.2.1 Attacks Detected In Trace 4

This trace is previously known to contain a reflector attack that sends ICMP ECHO reply packets to the victim with IP address 87.134.184.48. This attack was detected by the AProf toolkit. Additionally, several other attacks were detected and several false positives were generated.

Victim IP	Attack Type	Duration	Packet Rate	Byte Rate	Number of Packets
80.80.251.63	SYN flood	408.69	0.61	28.81	250
40.72.194.149	ICMP flood	150.81	152.66	156326.43	23023
87.231.216.115	ICMP flood	441.04	310.90	12888.61	137120
87.134.184.48	ICMP flood	252.02	4136.41	198737.53	1042469

Table 23 Attacks detected in Trace 4 and their attributes

Victim IP	Spoofing	Number of Sources	Number of Source Ports	Number of Destination Ports
80.80.251.63	No	5	250	4
40.72.194.149	No	37	0	0
87.231.216.115	No	12	0	0
87.134.184.48	No	143	0	0

Table 24 Attacks detected in Trace 4 and their attributes(cont'd)

Table 23 and Table 24 show the alerts for which we have high confidence that they represent DoS attacks. We show some relevant attributes for each alert. The attack that was originally identified by ISI researchers is shaded.

We manually examined traces each attack alert and established that the above-detected attacks are indeed DoS attacks, in spite of some of them having a low packet rate. The low packet rate can be captured in the attack trace if the trace is collected near a single attack participant. The SYN flood attack we detected has a low packet rate but it exhibits persistent SYN traffic from the same sources to the attack's target over a long time. Since SYN floods can be successful with low packet rates we were confident that this alert was indeed identifying attack and not legitimate behavior. ICMP floods were sending traffic to the victim at very high rates. The highest-rate attack was detected by ISI researchers.

AProf toolkit also raised some alerts that we flagged as possible false positives. We classified the reasons for these false positives into 3 high-level classes:

(1) Alerts that may either indicate a DoS attack or one-way traffic we could not remove from the trace, such as one-way DNS traffic without any TCP traffic to a given destination

(2) Alerts that indicate persistent ICMP traffic to a non-responsive destination but send at a low rate so they may be generated by some kind of network monitoring

(3) Alerts that are triggered by one-way UDP traffic generated by gaming and peer-to-peer applications

Following tables shows some examples of those kinds of alerts and their attributes. Table 25 and Table 26 show examples of one-way DNS traffic and persistent ICMP traffic at a moderate packet rate.

Victim IP	Attack Type	Duration	Packet Rate	Bytes Rate	Number of Packets
87.231.30.56	UDP Flood	0.08	3409.90	208110.03	258
87.231.30.56	UDP Flood	1.08	393.35	24767.29	426
87.231.30.56	UDP Flood	1.15	546.97	34900.27	630
87.231.142.6	UDP Flood	6.82	154.88	10414.56	1057
87.231.30.56	ICMP flood	440.89	46.09	6646.57	20320

Table 25 Alerts and attributes in Trace 4 for UDP-ICMP mix pattern caused by gaming and peer-to-peer

Victim IP	Spoofing	Number of Sources	Number of Source Ports	Number of Destination Ports
87.231.30.56	No	111	71	1
87.231.30.56	No	182	102	1
87.231.30.56	No	248	138	1
87.231.142.6	No	60	42	1
87.231.30.56	Subnet	1411	0	0

Table 26 Alerts and attributes in Trace 4 for UDP-ICMP mix pattern caused by gaming and peer-to-peer (cont'd)

We have 5 alerts in . Table 25 and Table 26 for 2 unique victim IP addresses, IP address 87.231.30.56 appears 4 times, because our toolkit identifies

attacks for the same victim at different times as different attacks if they are separated by the periods of low attack activity.

In Table 27 and Table 28 we show alerts for low-rate ICMP flows that have a considerably long duration indicating traffic monitoring.

Victim IP	Attack Type	Duration	Packet Rate	Bytes Rate	Number of Packets
57.174.178.52	ICMP Flood	435.28	0.78	50.28	342
53.65.96.180	ICMP Flood	436.09	4.52	1642.69	1971
52.230.221.84	ICMP Flood	6.82	7.28	5036.49	3205

Table 27 Alerts and attributes for low rate ICMP flood pattern

Victim IP	Spoofing	Number of Sources	Number of source ports	Number of destination ports
57.174.178.52	No	51	0	0
53.65.96.180	No	237	0	0
52.230.221.84	No	744	0	0

Table 28 Alerts and attributes for low rate ICMP flood pattern (cont'd)

When the related traces are examined, it can be seen that there is some ICMP traffic with small rate targeting the victim, which is accompanied by some UDP traffic and TCP traffic, ICMP traffic is coming from different networks, but UDP and TCP traffic are coming from same subnets generally. This pattern can be attack but it is also possible that it can be another service since the packet rate is small.

Finally we had quite a few alerts triggered by one-way gaming and peer-to-peer traffic. This traffic is aggressive and uses UDP or TCP protocol, but is not part of DoS attack. We can eliminate these false positives by creating a list of destination ports for such traffic and filtering it out of the trace, similarly to one-way traffic filtering.

3.2.2 Attacks Detected In Trace 18

Row	Victim IP	Attack Type	Duration	Packet Rate	Bytes Rate	Number of Packets
1	215.95.218.157	SYN flood	321.69	0.33	17.51	106
2	170.73.56.195	ICMP flood	30.68	3.91	4401.42	120
3	171.221.4.115	SYN flood	486.97	0.26	11.29	125
4	170.73.56.195	ICMP flood	46.12	2.86	3365.12	132
5	170.73.56.195	ICMP flood	40.8	3.26	3813.63	133
6	170.73.56.195	UDP flood	10.03	15.25	5356.54	153
7	170.73.56.195	ICMP flood	70.86	2.38	2885.6	169
8	170.73.50.241	SYN flood	6.3	28.89	1581.1	182
9	214.100.159.28	ICMP flood	6.46	43.15	2416.35	279
10	177.84.13.139	SYN flood	615.42	0.53	27.18	329
11	177.84.13.139	SYN flood	392.06	0.93	45.56	364
12	216.62.149.160	SYN flood	40.5	9.53	571.80	386
13	170.73.50.241	SYN flood	33	14.48	721.1	478
14	170.73.50.241	SYN flood	33.01	14.53	726.98	480
15	167.12.243.15	SYN flood	120.75	4.37	262.34	528
16	180.53.153.74	ICMP flood	39.04	15.03	841.94	587
17	170.73.157.18	SYN flood	1050.66	1.64	79	1720
18	197.65.179.77	ICMP flood	11.29	1150.85	64447.81	12989
19	173.100.10.142	TCP no-flag	60.99	1100.40	44016.22	67116
20	173.100.10.142	TCP no-flag	60.85	1106.46	44258.64	67328
21	89.1.89.241	TCP no-flag	720.94	1094.96	43798.50	789405

Table 29 Attacks detected in Trace 18 and their attributes

Row	Victim IP	Spoofing	Number of Sources	Number of source ports	Number of destination ports
1	215.95.218.157	No	20	23	1
2	170.73.56.195	No	88	0	0
3	171.221.4.115	No	8	125	1
4	170.73.56.195	No	89	0	0
5	170.73.56.195	No	90	0	0
6	170.73.56.195	No	7	7	3
7	170.73.56.195	No	108	0	0
8	170.73.50.241	No	2	2	1
9	214.100.159.28	No	4	0	0
10	177.84.13.139	No	144	200	1
11	177.84.13.139	No	105	250	1
12	216.62.149.160	No	1	386	1
13	170.73.50.241	No	1	1	1
14	170.73.50.241	No	1	1	1
15	167.12.243.15	No	2	502	1
16	180.53.153.74	No	2	0	0
17	170.73.157.18	No	213	535	1
18	197.65.179.77	No	1	0	0
19	173.100.10.142	Random	67073	50818	41961
20	173.100.10.142	Random	67318	50901	42005
21	89.1.89.241	Random	788826	65535	65530

Table 30 Attacks detected in Trace 18 and their attributes (cont'd)

Only known attack in **Trace 18** was TCP no-flag attack targeted at IP address 89.1.89.241, which is shown as shaded in Table 29 and Table 30 In addition to that, Aprof toolkit has extracted 20 more attacks from the trace, shown in these tables and verified manually to represent credible attack alerts.

First attack detected by AProf toolkit is SYN flood targeted at IP address 215.95.218.157. When we examine the trace for this attack, we see that SYN flood is targeted at port number 25. Since the packet rate is small and the duration of attack is long, it can be thought that the destination is a mail server that is down and the SYN

traffic is actually legitimate traffic trying to access this server. A closer examination indicates that each SYN is followed by a RST reply from the server and the SYN packets are sent from the same sources continuously. This verifies our hypothesis that the alert denotes a real DoS attack event.

Second attack that is identified by the AProf toolkit is ICMP flood, which is targeted at 170.73.56.195. This attack appears 5 times in our table, namely in 2nd, 4th, 5th, 6th rows. It stops and starts periodically, and manual trace analysis confirms that it is indeed ICMP flood attack.

Next attack detected by the toolkit is aimed at 171.221.4.115, and it is a SYN flood with the same properties as the first SYN flood attack on 215.95.218.157.

Next attack is aimed at 170.73.50.241 and it is a SYN flood. In addition, it is identified 3 times throughout the trace, namely it stops and starts again periodically. Attributes of the attack can be found at 8th, 13th, 14th rows. When the related traces are observed, it is the case that SYN flood is targeting port 53 and it is generally followed by UDP traffic. It seems to have a strong attack rate for 30 seconds duration for each of the instances of the attack.

Toolkit also detected ICMP flood sent to 214.100.159.28, shown at 9th row. When we examine the trace for the destination it can be seen that UDP flood targeting the port 53 accompanies the ICMP flood. Because of the attack detection criteria's and precedence values that are explained in Section 2.2, toolkit has to choose one of the attack types, and this is the type with the highest precedence, namely ICMP flood.

Next attack detected by the toolkit is SYN flood to 177.84.13.139. It has been identified two tiymes and attack information can be seen in the 10th and 11th

rows. These SYN flood alerts also show the same characteristics with the first SYN flood explained above. Destination's port 25 is targeted by the attack traffic. Even though the packet rate is not really high, each attack instance continues for a long duration and traffic pattern looks like attack.

The next attack identified is for destination IP address 216.62.149.160, and attack type that is identified by the toolkit is SYN flood, it can be seen at the 12th row. According to the related trace, this SYN flood is targeting port 8080, it has single source and it is not very aggressive. But the pattern is like the previous SYN floods, it looks like attack traffic.

Next attack detected is the SYN flood aimed at IP address 167.12.243.15, and it can be found at 15th row. SYN flood is targeting the port 80, and it follows typical SYN flood pattern that is explained above.

ICMP flood at 16th row that is targeted at 180.53.153.74 is a typical ICMP flood with high rate. Flood has two sources, when the traces are examined high rate ICMP flood pattern is examined.

SYN flood at 17th row that is targeted at 170.73.157.18 lasted for a long time with small packet rate. It targeted at port 53 of destination and generally followed by a UDP flood.

ICMP flood at 18th row that is destined to 197.65.179.77 lasted for 11 seconds with very high packet rate of 1150.85 packets/sec. One interesting note about this attack is that there is a single source and a single destination for this attack, and the source was scanning some other IP addresses for port 80 before the attack start.

The next attack that is detected is targeted at 173.100.10.142. This attack has 2 instances in our tables, so it shows that attack stopped and started over after a

while. Attack type is TCP no-flag attack. Attack lasted for two separate 60-second periods with packet rate around 1100 packets/sec and used random spoofing.

The last attack was the strongest attack and was the only attack that was previously known to be in the trace 18. It is another TCP no-flag attack which lasted 720.94 seconds with a very high rate of 1094.96 packets per second. For this destination, the tables show that 788826 different sources exist in the trace, but since AProf detected random spoofing the source count is not reliable.

Just like in the previous trace we had some false positives, mostly due to presence of gaming and peer-to-peer traffic. We show some of these alerts in the following tables. Table 31 and Table 32 show alerts induced by one-way DNS traffic.

Victim IP	Attack Type	Duration	Packet Rate	Bytes Rate	Number of Packets
214.100.170.230	UDP Flood	2.25	57.73	4876.54	130
214.100.170.230	UDP Flood	2.41	62.36	4496.51	150
214.100.107.67	UDP Flood	1054.77	47.41	7299.88	50010

Table 31 Alerts and attributes for gaming and peer-to-peer UDP traffic pattern

Victim IP	Spoofing	Number of Sources	Number of source ports	Number of destination ports
214.100.170.230	No	50	41	2
214.100.170.230	No	48	42	1
214.100.107.67	Random	2246	0	0

Table 32 Alerts and attributes for gaming and peer-to-peer UDP traffic pattern (cont'd)

In the first two alerts packet rate was pretty high and attack pattern took place for short durations for two times. We labeled this as a suspicious case, which requires further investigation.

The last alert that our toolkit detected for this pattern was another UDP flood targeted at 214.100.107.67, when the trace is examined, the pattern of asymmetric DNS traffic with high rate is observed again. This alert lasted for a long time therefore it is more probable that it is DNS server traffic, and we're seeing only one direction in the communication. Our toolkit detected random spoofing for this alert because of the high number of sources.

Another suspicious pattern that may indicate false positives is low rate SYN flood pattern shown in Table 33 and Table 34.

Victim IP	Attack Type	Duration	Packet Rate	Bytes Rate	Number of Packets
214.100.0.129	SYN flood	955.22	0.11	5.18	103
209.139.215.37	SYN flood	434.95	0.31	15.89	136

Table 33 Alerts and their attributes for low-rate SYN flood pattern

Victim IP	Spoofing	Number of Sources	Number of source ports	Number of destination ports
214.100.0.129	No	1	103	1
209.139.215.37	No	21	36	2

Table 34 Alerts and their attributes for low-rate SYN flood pattern (cont'd)

The first alert indicates either a very slow attack or aggressive application because packets are sent from the single source to a single destination. The second alert has a slightly higher packet rate, slightly shorter duration and multiple sources. Packets are targeting port 25 and 80 with low packet count, so the reason for this pattern might be either attack or a server, which is down.

Table 35 and Table 36 show false positives generated by the gaming and peer-to-peer traffic, mixed with some ICMP traffic used by game users to track their delays to game servers.

Victim IP	Attack Type	Duration	Packet Rate	Bytes Rate	Number of Packets
214.31.207.204	UDP flood	6.17	17.18	914.37	106
214.31.207.204	UDP flood	18.27	9.58	711.09	175
214.31.207.204	UDP flood	40.79	4.73	349.97	193
214.31.207.204	UDP flood	25.05	9.90	752.33	248
170.73.56.195	ICMP flood	229.59	1.26	1556.42	290
214.31.207.204	UDP flood	131.06	5.25	372.64	688
216.62.152.142	Frag. Traffic	954.96	2.87	2589.13	2744
216.62.37.35	Frag. Traffic	926.87	3.08	2882.43	2851
213.46.169.209	UDP flood	127.29	29.44	39104.54	3748
214.176.9.187	Frag. Traffic	1019.57	5.38	4225.4	5488

Table 35 Alerts and their attributes in Trace18 for UDP-ICMP mixes caused by gaming and peer-to-peer applications

Victim IP	Spoofing	Number of Sources	Number of source ports	Number of destination ports
214.31.207.204	No	12	3	2
214.31.207.204	No	16	2	2
214.31.207.204	No	16	2	2
214.31.207.204	No	16	2	2
170.73.56.195	No	107	0	0
214.31.207.204	No	17	2	2
216.62.152.142	No	1	244	115
216.62.37.35	No	1	273	124
213.46.169.209	No	1	35	3
214.176.9.187	No	1	466	217

Table 36 Alerts and their attributes in Trace18 for UDP-ICMP mixes caused by gaming and peer-to-peer applications (cont'd)

Another pattern we extracted from the trace is a slow SYN floods with periodic traffic bursts. We show one example of this in Table 37 and Table 38.

Victim IP	Attack Type	Duration	Packet Rate	Bytes Rate	Number of Packets
177.84.54.80	SYN flood	1000.11	0.2	12.06	201

Table 37 Alerts and their attributes low rate, periodical SYN floods

Victim IP	Spoofing	Number of Sources	Number of source ports	Number of destination ports
177.84.54.80	No	1	201	1

Table 38 Alerts and their attributes low rate, periodical SYN floods (cont'd)

When we examined the packet trace for this alert we found that a single SYN packet is sent and then retransmitted periodically with 5-second sleep periods. Since the packet is sent to a port for a modem pool, we concluded this indicated unsuccessful automatic attempts to connect to a modem server.

3.2.3 Attacks Detected In Trace 29

Table 39 and Table 40 show attacks detected in **Trace 29**, with the attack originally identified by the ISI researchers being shown in shaded cells.

Row	Victim IP	Attack Type	Duration	Packet Rate	Byte Rate	Number of Packets
1	164.31.154.4	ICMP flood	19.39	5.26	436.07	102
2	164.31.154.4	ICMP flood	19.47	5.29	437.13	103
3	164.31.154.4	ICMP flood	19.39	5.36	440.35	104
4	198.178.147.30	ICMP flood	15.94	6.59	381.85	105
5	197.1.119.81	SYN flood	873.15	0.14	6.93	126
6	167.110.108.77	SYN flood	241.02	1.11	53.37	268
7	197.1.237.22	UDP flood	25.38	13.24	943.24	336
8	168.135.164.146	SYN flood	715.65	0.48	23.17	342
9	88.89.133.34	ICMP flood	10.85	39.35	2203.84	427
10	166.59.236.47	SYN flood	934.85	0.51	24.29	473
11	167.110.232.45	SYN flood	943.25	0.99	47.67	930
12	166.59.104.140	ICMP flood	952.5	1.05	1553.65	996
13	198.178.98.152	UDP flood	32.98	33.11	44301.58	1092
14	167.110.70.205	ICMP flood	955.91	5.64	5865.99	5391
15	179.154.11.215	ICMP flood	955.63	6.25	349.72	5968
16	166.49.80.232	UDP flood	619.65	20.15	26500.48	12485
17	176.45.238.43	ICMP flood	14.81	1059.17	59313.73	15688
18	166.59.72.68	UDP flood	68.42	281.83	23618.6	19283
19	197.1.39.132	ICMP flood	956.48	66.76	8578.97	63855
20	88.89.192.119	Invalid Protocol	930.52	24375.29	1950023.18	22681920

Table 39 Attacks detected in Trace 29 and their attributes

Row	Victim IP	Spoofing	Number of Sources	Number of source ports	Number of destination ports
1	164.31.154.4	No	8	0	0
2	164.31.154.4	No	7	0	0
3	164.31.154.4	No	7	0	0
4	198.178.147.30	No	38	0	0
5	197.1.119.81	No	1	42	1
6	167.110.108.77	No	1	94	1
7	197.1.237.22	No	118	84	2
8	168.135.164.146	No	100	102	2
9	88.89.133.34	No	1	0	0
10	166.59.236.47	No	1	160	1
11	167.110.232.45	No	122	294	1
12	166.59.104.140	No	26	0	0
13	198.178.98.152	No	1	24	8
14	167.110.70.205	No	716	0	0
15	179.154.11.215	No	1	0	0
16	166.49.80.232	No	1	38	7
17	176.45.238.43	No	1	0	0
18	166.59.72.68	No	78	91	5
19	197.1.39.132	Reserved	2165	0	0
20	88.89.192.119	No	3	0	0

Table 40 Attacks detected in Trace 29 and their attributes (cont'd)

The first 3 rows show the ICMP flood targeted at 164.31.154.4. When we looked at the trace for those alerts, we saw that the ICMP traffic was accompanied by some TCP traffic coming from port 7 (echo server) of a different source. This TCP traffic looks like replies to the victim.

4th row indicates a moderate rate ICMP flood aimed at 198.178.147.30. When the trace is examined it can be seen that victim is under ICMP flood and UDP flood, since the precedence of the ICMP flood is higher, toolkit identified the alert as ICMP flood.

5th row shows a SYN flood alert for the victim 197.1.119.81. When the alert and the trace are further examined, it is observed that SYN flood has small packet rate, and it is sending one packet to port 8080 from the same source in every 5 seconds periodically.

6th row gives information about SYN flood to destination 167.110.108.77. It is exactly same with the previous attack explained above; only difference is the SYN packets go to the port 80 instead of 8080.

In the 7th row, UDP flood alert is given for the victim 197.1.237.22. When the trace and the alert attributes were examined, they indicated a high rate of DNS traffic that is accompanied by SYN flood going to port 53.

8th row is another SYN flood alert targeted at 168.135.164.146. It is a similar SYN flood with small rate where port 6346 is targeted.

In 9th row, ICMP flood is targeting the IP 88.89.133.34. Further examination of the related trace gives interesting results. It looks like main component of the trace is one-way ICMP flood destined at victim. But after that, victim sends UDP flood to the attacker like punishing the attacker. Also real time streaming protocol and TCP connections are present too. This is a difficult pattern to explain but it looked malicious and aggressive.

10th row in the table shows another SYN flood targeted at 166.59.236.47, where attack pattern is similar to the attacks explained above. Again port 6346 is targeted.

In 11th row, there is a SYN flood alert for victim 167.110.232.45. When further analysis is made via examining the trace, it can be seen that it is a typical SYN flood directed to port 45.

In 12th row, ICMP flood alert for 166.59.104.140 is given. Traffic for the destination includes ICMP and UDP flood, and in addition to that, there are some TCP communications, but none of them have high rates. So the tool classified this alert as ICMP flood by taking the highest precedence among the alarm signals that are raised.

In 13th row, UDP alert for victim 198.178.98.152 is shown. If the trace is examined it can be seen that it is a UDP flood. Traffic is sent from one source at a high packet rate of 33.11 packets per second.

In 14th row ICMP flood alert is given for the victim IP 167.110.70.205. Victim is receiving ICMP packets with high rate from different sources, there might be random spoofing but number of packets is still not enough for the toolkit to detect it.

ICMP flood alert in 15th row of tables is similar to the case in row 14, with the difference that victim IP 179.154.11.215 receives packets from single IP instead of different IP addresses.

In 16th row of tables another UDP flood alert is given, it is exactly same pattern with the pattern explained in row 13. There's a single source that is sending UDP packets with high rate. Our toolkit identified this attack correctly again.

In 17th row another interesting alert instance is seen. Victim IP 176.45.238.43 is under a SYN flood but after a while it sends ICMP flood back to the attacker, as if it wants to take revenge from the attacker. Again it is similar to attack alert at 8th row.

UDP flood targeted at 166.59.72.68 can be found at 18th row. It is another UDP flood with high rate, but in contrast to other ones that are explained above it has more than 1 source and packet rate is much higher.

In the 19th row, ICMP flood alert targeted at 197.1.39.132 can be found. When the trace is further examined, victim is under both ICMP flood and UDP flood which goes to port 53 but because of the precedence property, toolkit identified the attack as ICMP attack.

In 20th row, invalid protocol attack targeted at 88.89.192.119 is shown. That was the only previously known attack from the trace specification. The measurements of the rate and duration match the attack specification provided by the ISI researchers. Besides identifying the previously known attack, the toolkit identified 19 more attack cases.

Like in other traces, we had several false positives. Some of these are shown in Table 41 and Table 42. They indicate a small-rate, long-duration SYN flood pattern but there are no packets in the reverse direction. Thus, we could not say for sure if this was a DoS attack or asymmetric traffic and we could not filter out.

Victim IP	Attack Type	Duration	Packet Rate	Bytes Rate	Number of Packets
167.120.4.220	SYN flood	203.57	1.33	64.43	270
167.120.4.220	SYN flood	642.49	1.48	71.87	953
192.164.21.219	SYN flood	954.8	4.93	239.61	4709

Table 41 Alerts and their attributes for low rate, long duration SYN floods

Victim IP	Spoofing	Number of Sources	Number of source ports	Number of destination ports
167.120.4.220	No	73	104	5
167.120.4.220	Reserved	267	367	5
192.164.21.219	Reserved	1499	1646	13

Table 42 Alerts and their attributes for low rate, long duration SYN floods (cont'd)

3.3 Auckland Traces

To perform further testing of the AProf tool, we wanted to examine public traces that were not labeled by researchers to contain known attacks. We used Auckland traces from NLANR Measurement and Network Analysis Group, which are publicly available at [9]. In the following tests, we used the trace Auckland VIII [7]. This is a two-week, random-anonymized, GPS-synchronized IP header trace captured from University of Auckland campus. It contains traffic between the University of Auckland and the rest of the Internet. Trace of each day is divided into 1-hour-long segments; therefore the entire trace is composed of 336 1-hour-long traces.

AProf toolkit is run on the Auckland VIII, and results are gathered and summarized in the following tables and figures. Due to the sheer volume of the data we could not manually verify each alert, like we did with Los Nettos. However, since majority of the identified attacks are SYN flood attacks and we had low number of false positives in this category on Los Nettos traces, we are confident that the majority of attacks we identified represent a DoS activity.

Type of Attack	Number of Instances	Percentage
Fragmented Attack	389	%3.25
ICMP Flood	152	%1.38
Invalid Protocol Attack	281	%2.54
SYN Attack	9826	%89.04
TCP No Flag Attack	18	%0.16
UDP Flood	368	%3.33

Table 43 Attack types in the traces

In Table 43 we show the distribution of attacks according to attack types. If the attack lasts more than one hour, it will be counted multiple times, since the toolkit is run on one-hour long traces. Since there are only a few long-lasting attacks, we believe this

feature of AProf tool does not bias the attack statistics. We see that TCP SYN attacks are the most prevalent, which is expected because these attacks blend easily with legitimate traffic and inflict a lot of damage since most of legitimate traffic uses TCP protocol.

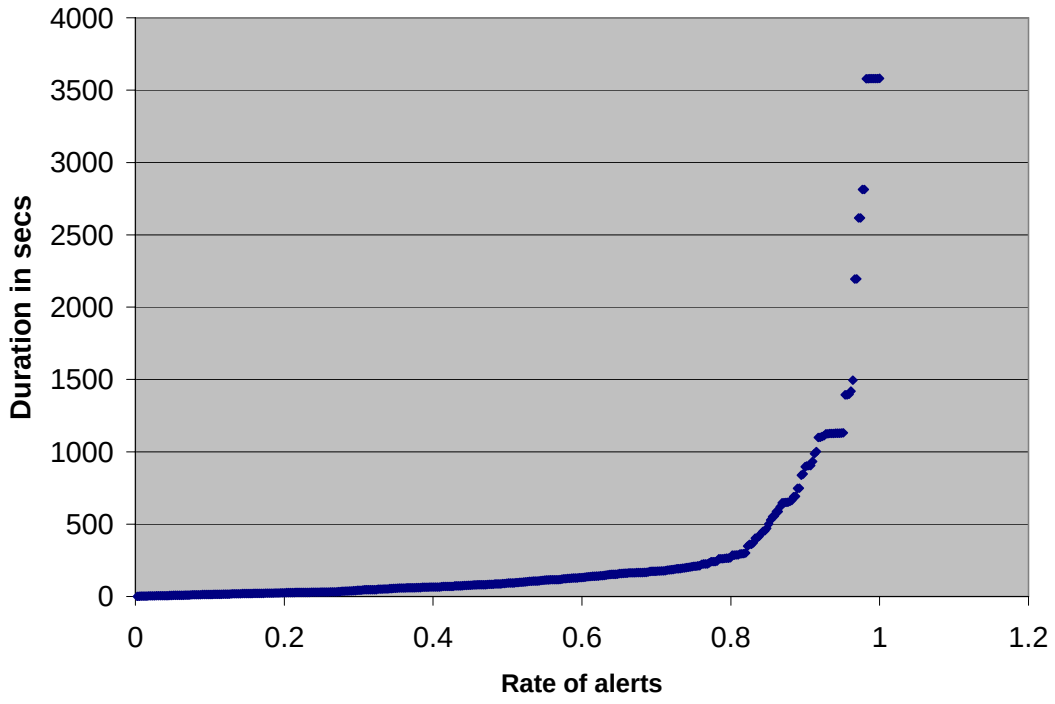

Figure 29 Distribution of duration for fragmented attacks

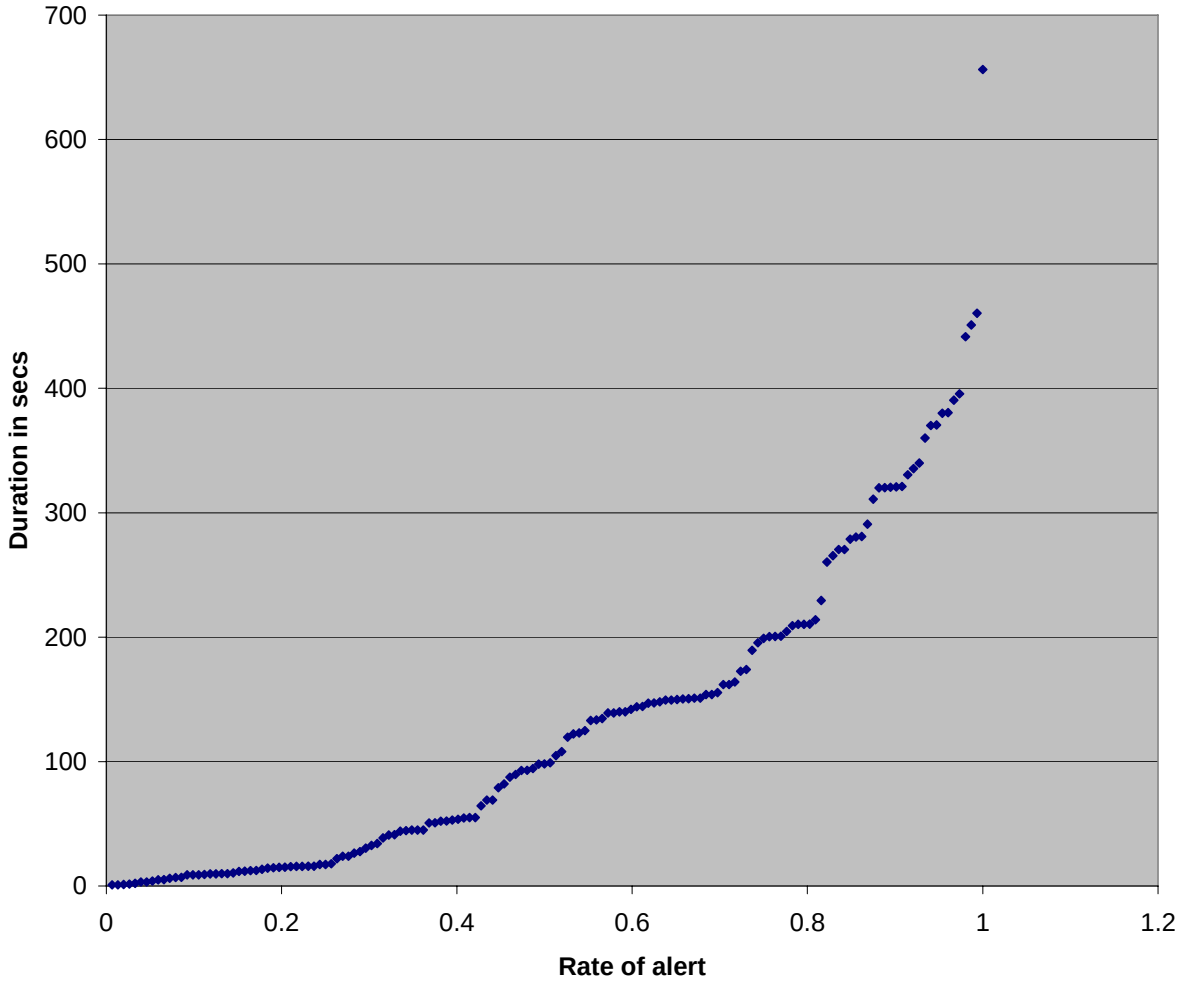

Figure 30 Distribution of duration for ICMP flood

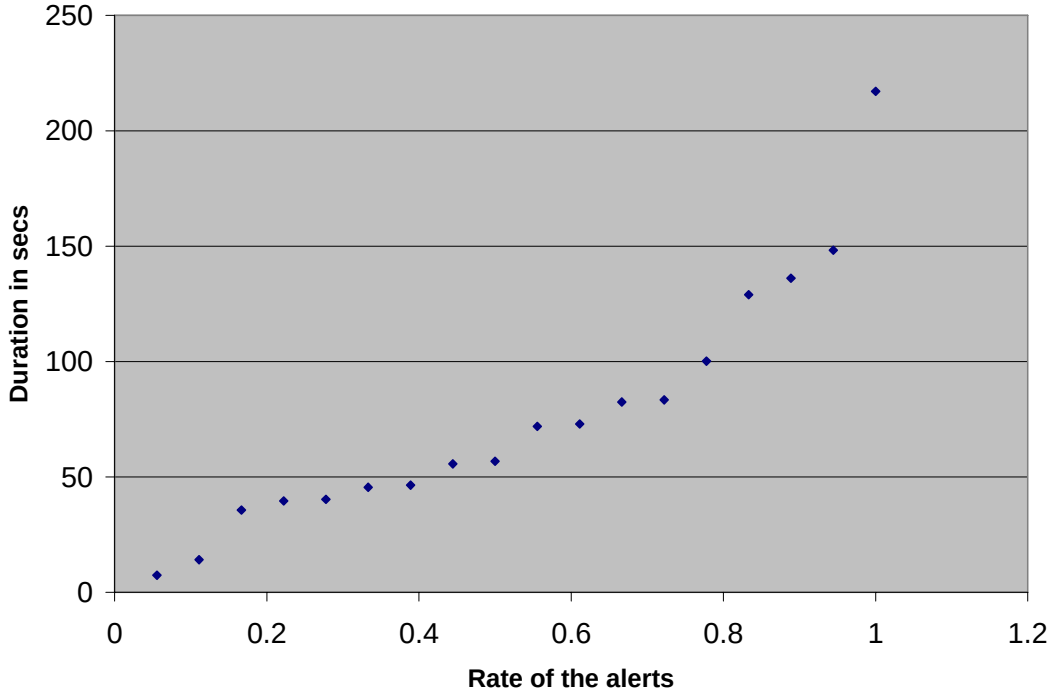

Figure 31 Distribution of duration for no flag TCP attacks

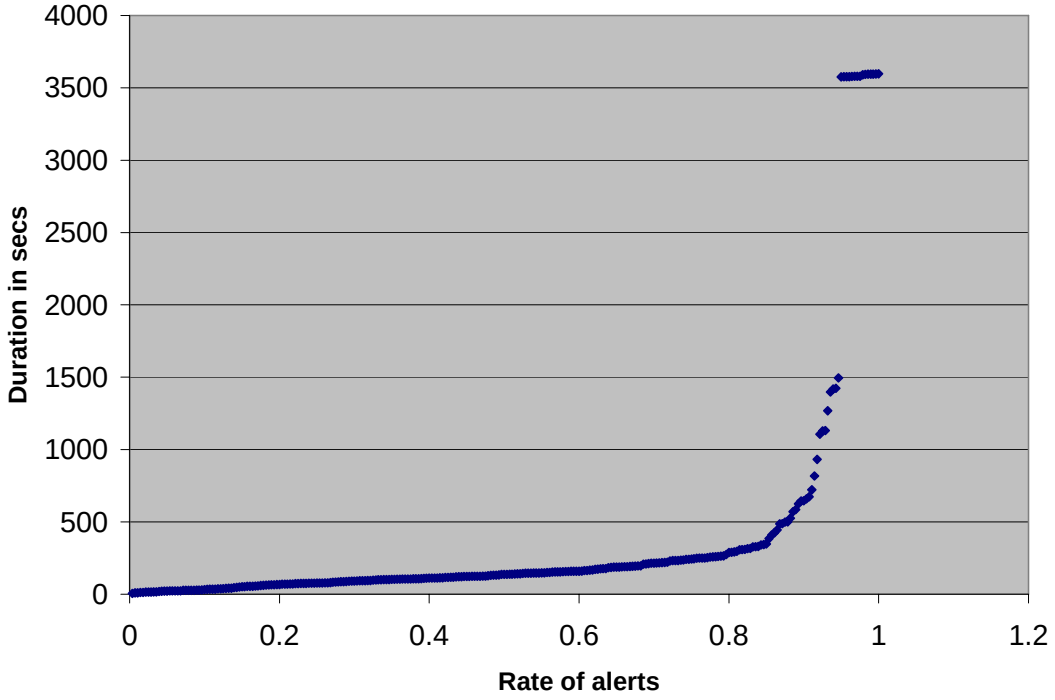

Figure 32 Distribution of duration for invalid protocol attacks

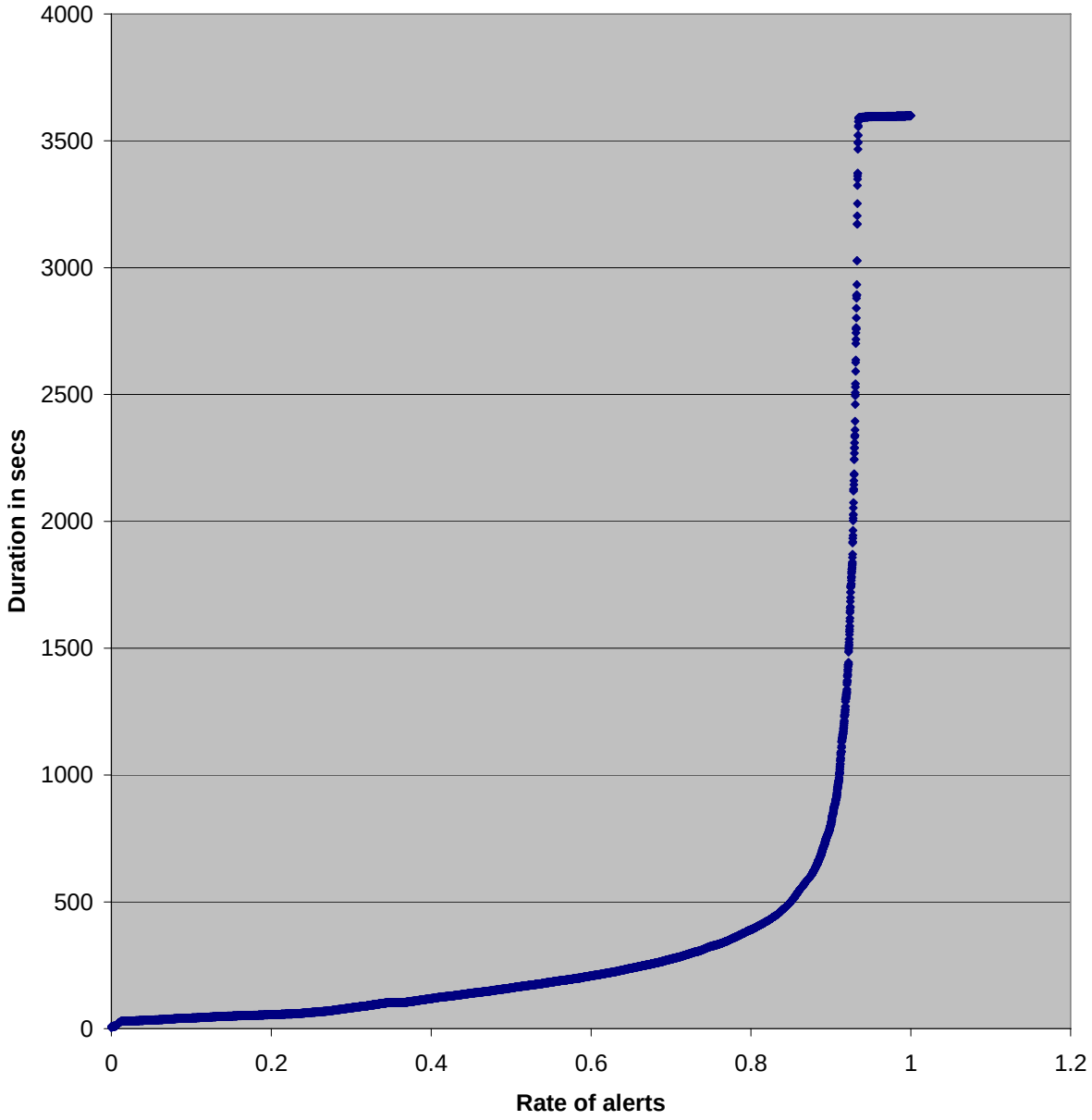

Figure 33 Distribution of duration for SYN floods

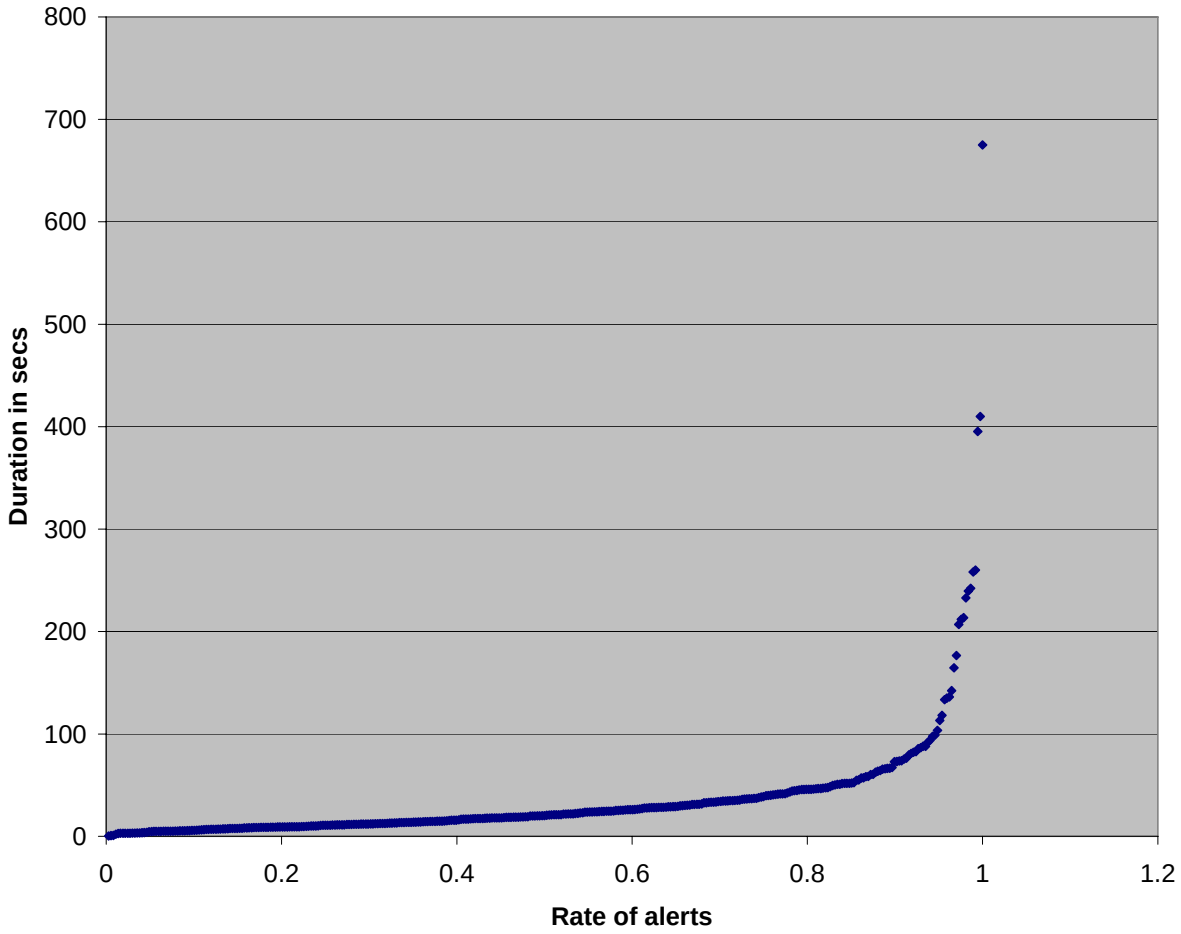

Figure 34 Distribution of duration for UDP floods

Figure 29-Figure 34 show the distribution of duration for different attack types. Figure 29 shows the distribution for the duration of fragmented attacks. According to the results approximately %95 of the attack alerts lasted less than 1500 seconds. The attack alert that lasted longest is 3600 seconds long, which is the maximum amount of time possible for an attack because of the trace length.

Figure 30 shows the distribution for the duration of ICMP floods. The longest ICMP flood lasted for 650 seconds, and most of the alerts lasted less than 500 seconds.

Figure 31 shows the distribution of TCP no-flag attack duration. The longest attack took place for approximately 250 seconds and most of the attacks lasted less than 200 seconds.

Figure 32 shows the distribution of invalid protocol attack duration. Approximately %85 of the invalid protocol attacks lasted less than 500 seconds, and approximately %5 of the alerts lasted 3600 seconds.

Figure 33 shows the distribution for the duration of SYN flood attacks. Approximately %80 of the attacks last less than 500 seconds, but around %10 attacks have the maximum duration, namely 3600 seconds.

Figure 34 shows the distribution of the duration for the UDP flood attacks. Those attacks lasted less than other type of attack alerts , the maximum duration in the traces are around 690 seconds and approximately %95 of the attacks are shorter than 100 seconds. Since we had most of the false positives in Los Nettos traces on UDP flood alerts, we believe many of the alerts for UDP flood attacks in AUCK trace also represent false positives.

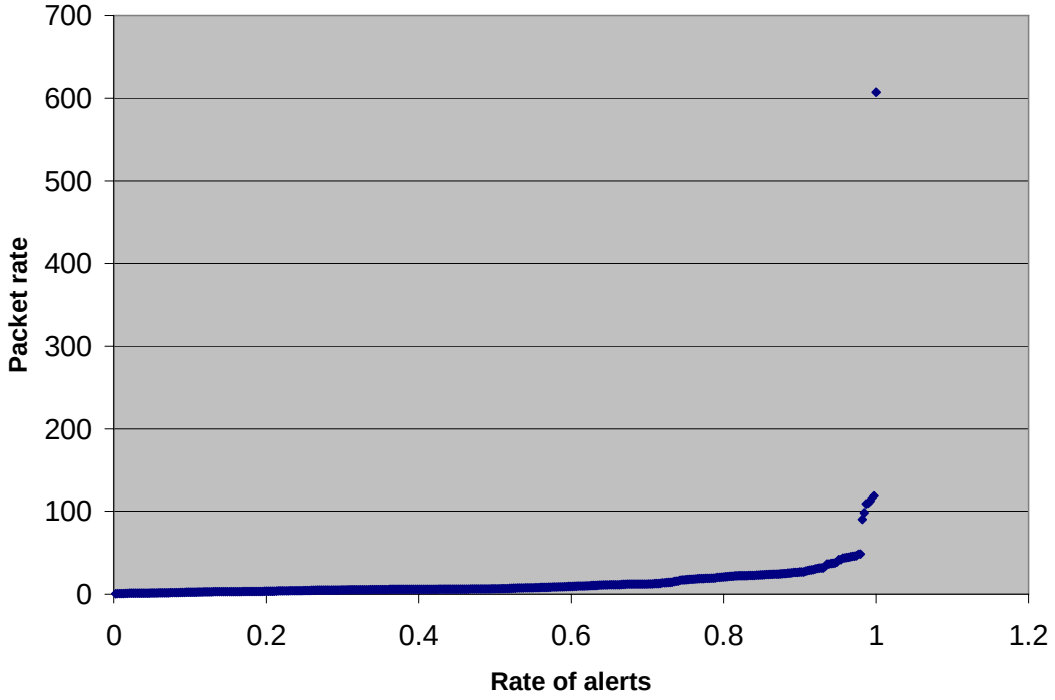

Figure 35 Distribution of packet rate for Fragmented attacks

Next we illustrate the distribution of the packet rate in attacks, in

Figure 35 - Figure 40.

Figure 35 illustrates that approximately 97% of the fragmented attack alerts have less than 50 packets per second, but one alert had nearly 600 packets per second.

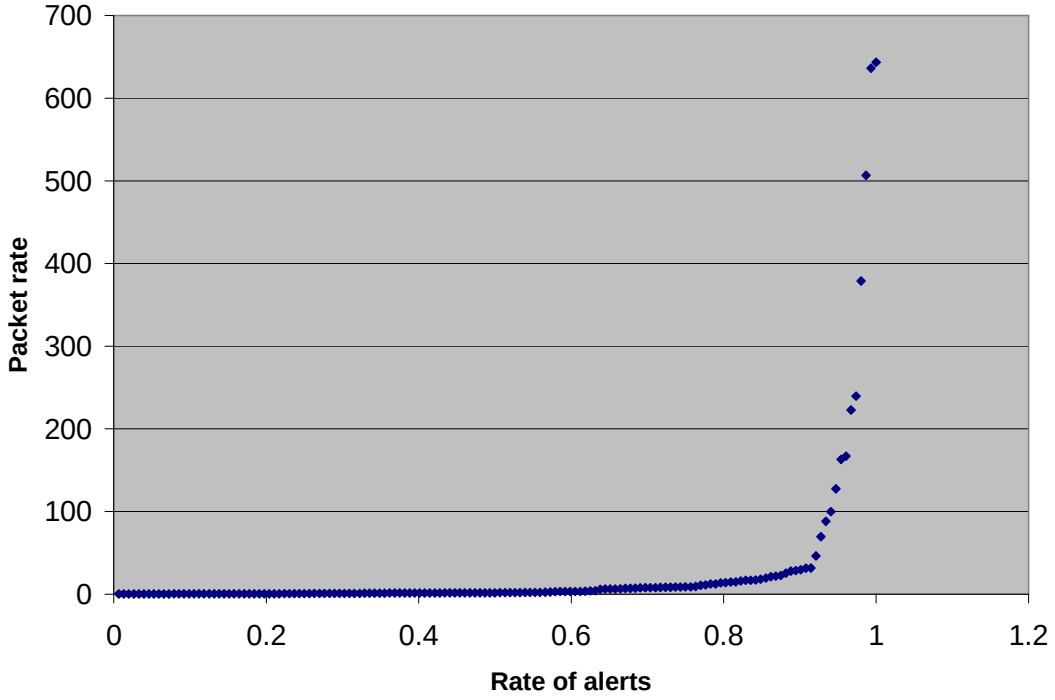

Figure 36 Distribution of packet rate for ICMP flood

Figure 36 shows a similar case to previous figure for ICMP flood. Approximately 90% of the alerts has packet rate smaller 50 packets per second. But this time they show seemingly uniform distribution between 50 packets per second and 700 packets per second. Packet rate has a peak value around 650 packets per second.

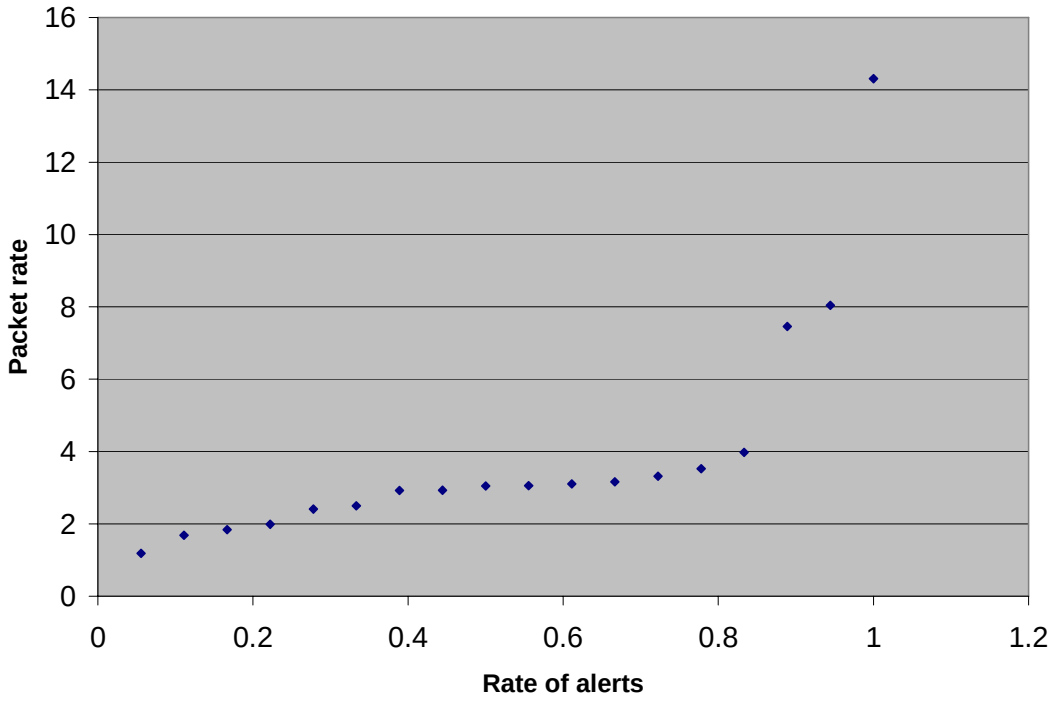

Figure 37 Distribution of packet rate for no flag TCP attacks

As it can be seen from discrete structure of the Figure 37, number of alerts for no-flag TCP attacks is smaller than for other types of attacks. In addition to this, most of attack alerts indicate moderate packet rates with only one alert with a packet rate of 14 packets per second.

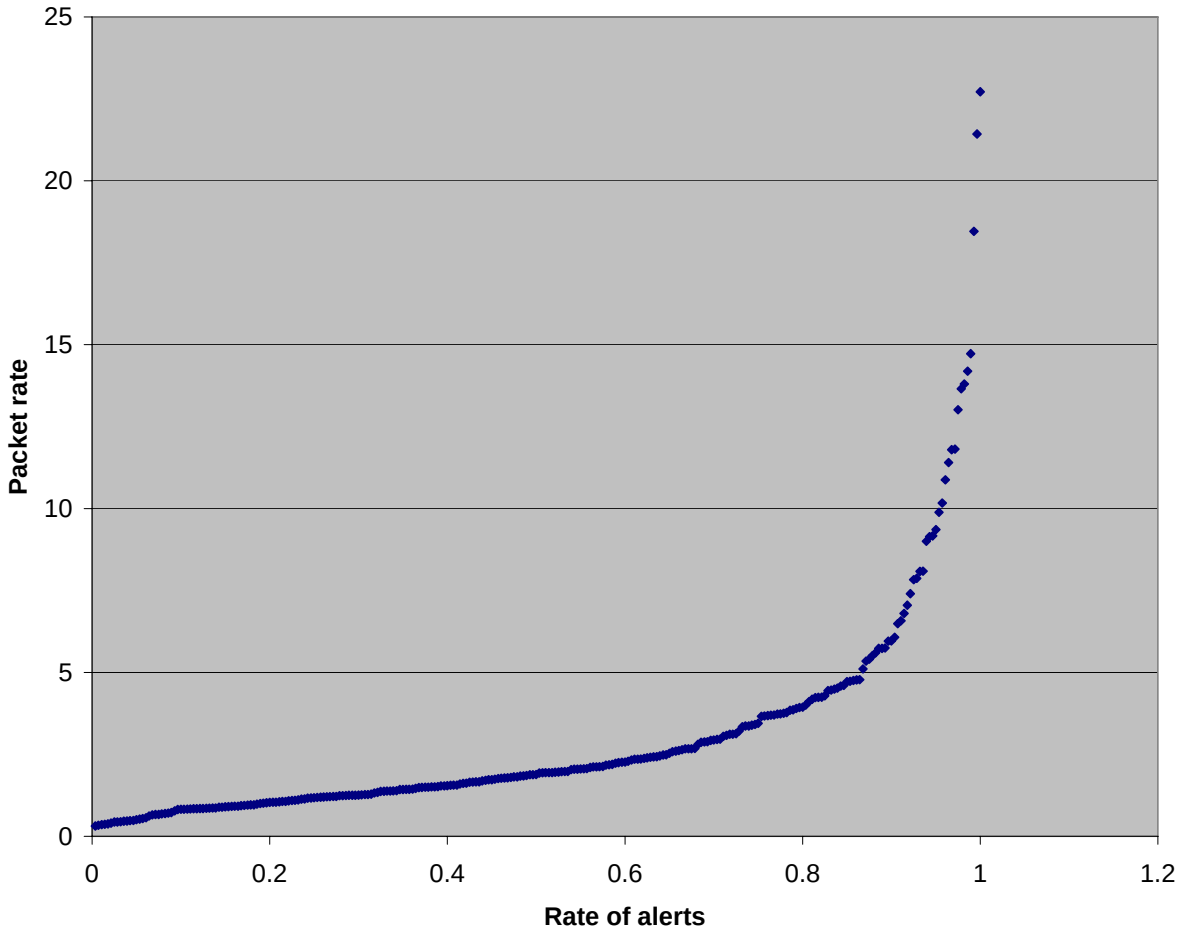

Figure 38 Distribution of packet rate for invalid protocol attacks

Figure 38 shows the distribution of packet rate for invalid protocol attacks. It can be seen that alerts for invalid protocol attacks have smaller packet rate. Approximately 90% of the attacks send less than 5 packets per second and the peak rate for the alerts is around 23 packets per second.

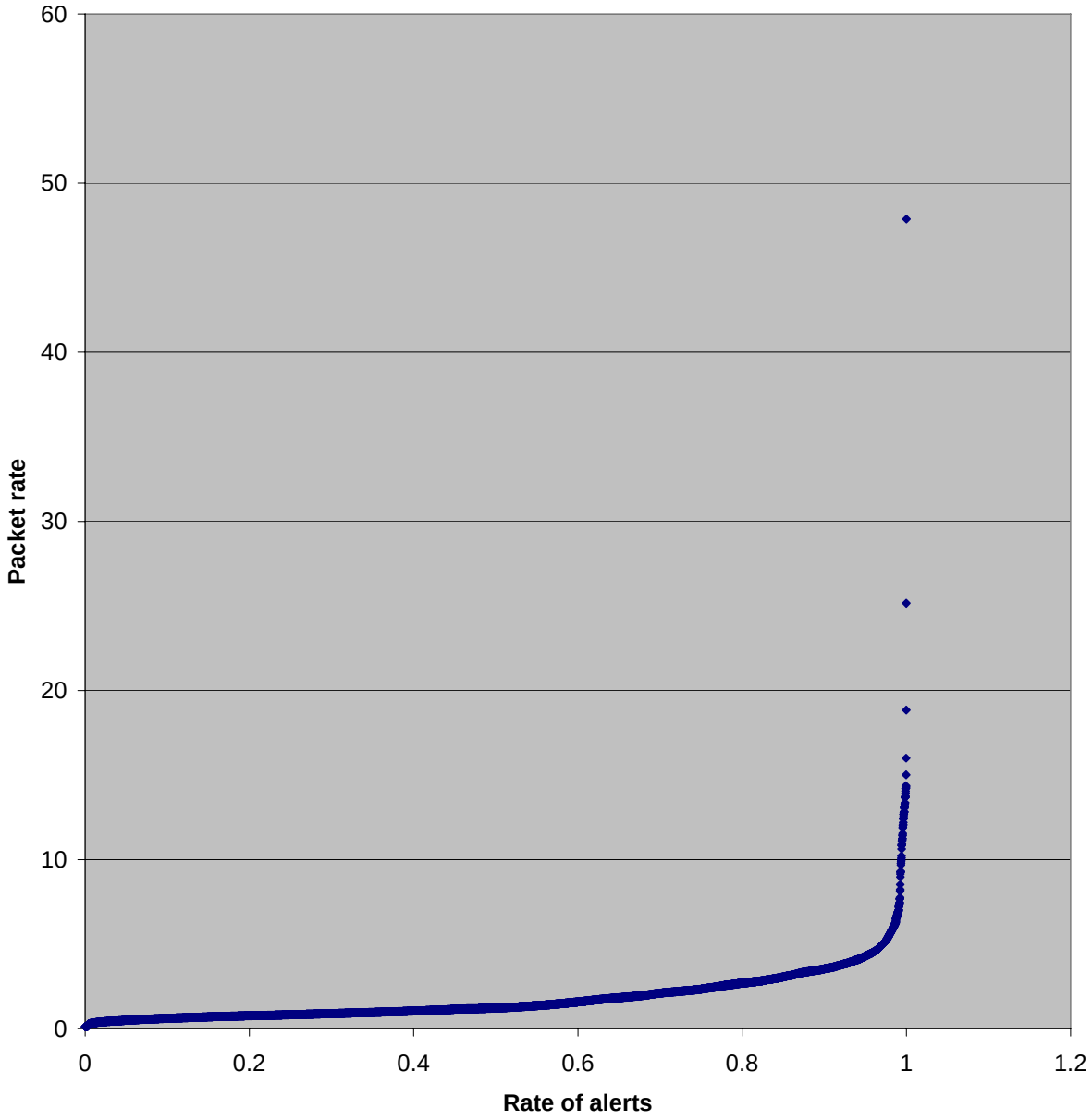

Figure 39 Distribution of packet rate for SYN flood

Figure 39 shows the distribution of packet rates in SYN flood attacks. It can be seen that except 2 alerts, all of the SYN flood alerts have packet rate less than 20 packets/sec, and approximately 97% of the alerts have packet rate smaller than 10 packets/sec. The peak packet rate is around 48 packets/sec. While the small attack

rates may indicate possibility of false positives we examined manually many SYN flood alerts to establish that they exhibit long-lasting attack patterns of SYN-followed-by-RST.

Even though distribution of the duration and packet rate for each attack can give us some idea about the attack, this is not enough. Each alert should be further examined to see if it is a false alarm or not because some legitimate traffic patterns might raise alarm for one of the attack features, as shown in Section 3.2. So further examination of the traffic patterns, ports used and source addresses for each alert are needed.

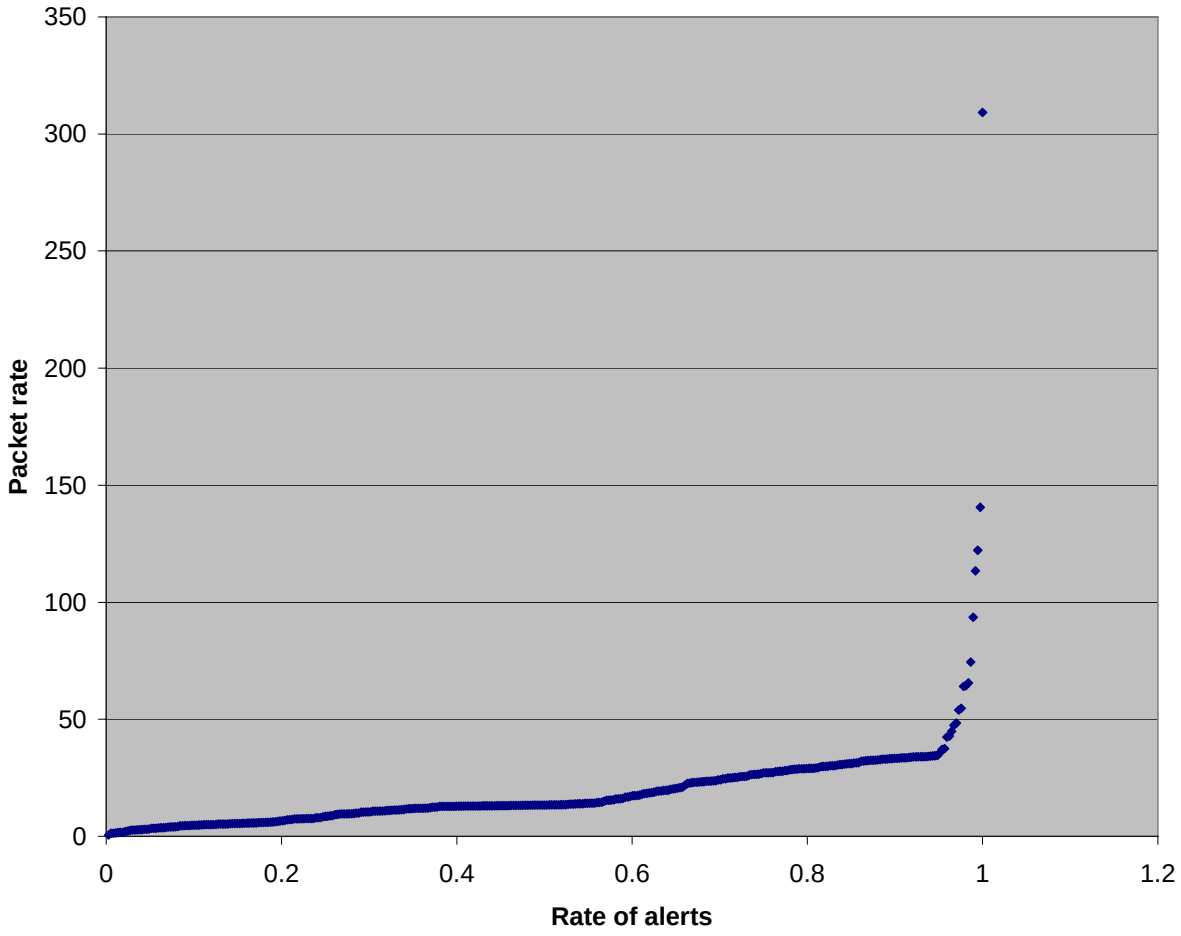

Figure 40 Distribution of packet rate for UDP floods

Figure 40 illustrates distribution of the packet rate for UDP flood. More than 95% of attacks lie below 50 packets/sec limit. Above that limit there are several alerts and only one of them has a packet rate above 300 packets per second.

Attack type	% of no spoofing	% of random spoofing	% of subnet spoofing
Frag. Attack	100	0	0
ICMP flood	100	0	0
Invalid protocol	100	0	0
SYN flood	93.9	5.99	0.11
TCP no flag	100	0	0
UDP flood	100	0	0

Table 44 Types of spoofing for attack types

Table 45 shows the distribution of spoofing techniques across attack categories. The SYN flood attacks were is the only type of attack that used spoofing. There were some significant number of attacks that used random spoofing and a small amount of subnet spoofing.

Num. of alerts	Num. of spoofed alerts	% of spoofed alerts
11033	598	%5.42

Table 45 Percentage of spoofing

Table 45 indicates that among 11033 attack alerts only in 598 had some kind of spoofing detected. So in Auckland VIII trace %5.42 of attack alerts also raised a spoofing alert. This indicates that spoofing is not prevalent among DoS attacks in today's Internet.

Finally, we show the distribution of number of sources participating in the attack for non-spoofed attacks, in Figure 41. Approximately 95% of attacks have less than 1000 sources that were visible in Auckland trace, which is not surprising given that TCP SYN flood attacks can be very successful at a moderate packet rate.

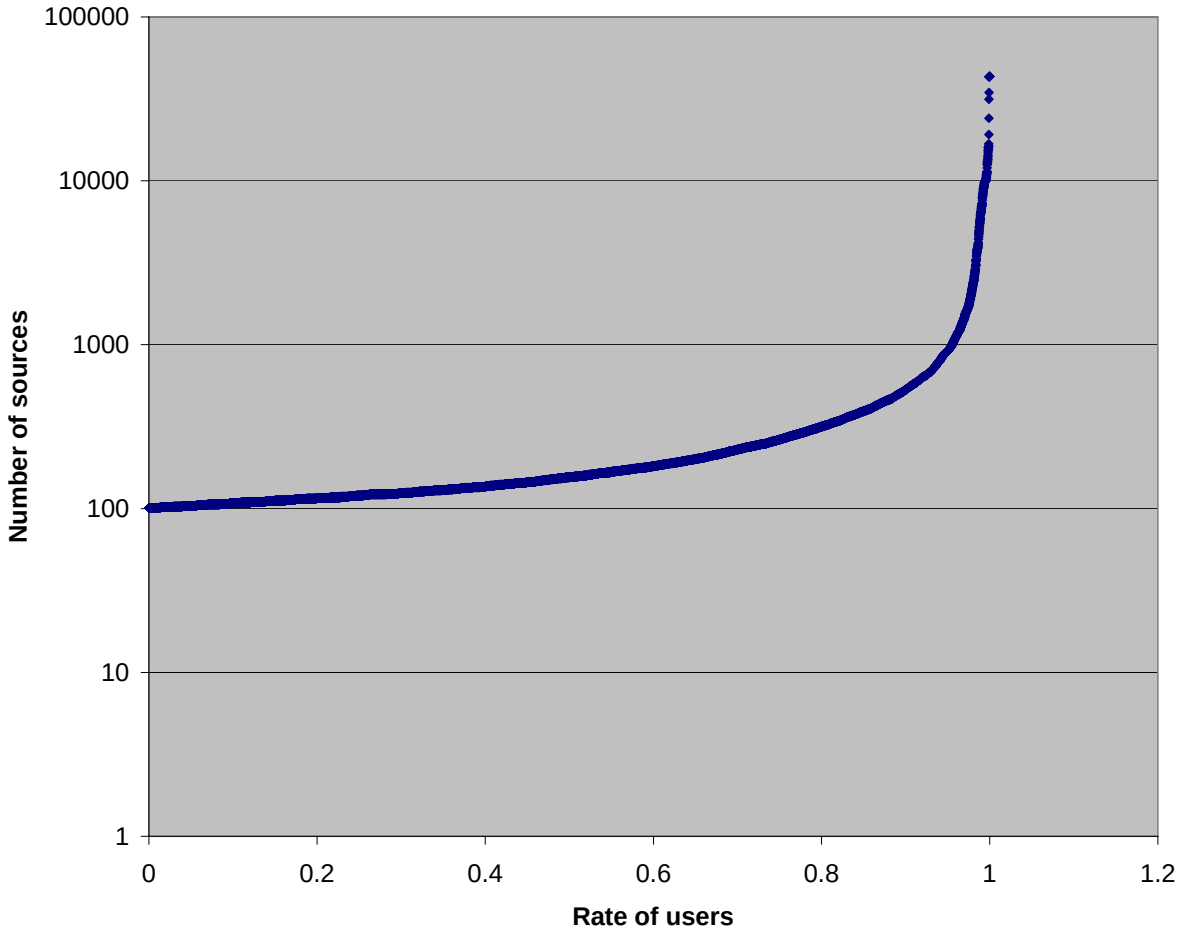

Figure 41 Distribution of number of sources for the alerts without any spoofing

4 RELATED WORK

The value of benchmarks for objective and independent evaluation has long been recognized in many science and engineering fields [10], [11], [12]. Recently, the Internet Research Task Force (IRTF) has chartered the Transport Modeling Research Group (TMRG) to standardize the evaluation of transport protocols by developing a common testing methodology, including a benchmark suite of tests [16].

Although benchmarks importance has been recognized a long time ago, there is less work about denial-of-service benchmarks. The reason for that lies in the difficulty of creating testing scenarios for denial-of-service threats, and characterizing them, since there are many ways to deny the service and many variants of attacks, while impact of the attack depends on network characteristics including traffic and resources.

As it is mentioned above several Internet researchers have attempted to characterize Internet denial-of-service activity [13], [14]. The work that is presented by those researchers used more limited observations and a single trace. Moreover, both of these studies were performed several years ago, and attacks have evolved since then.

Finally, it is clear that there is a lack of studies that characterize denial-of-service attack behavior; our work aims to fill this research space.

5 CONCLUSION

Even though the work in this thesis put substantial amount of effort to design tools for automatic characterization of denial-of-service attacks, there is still a lot to be done. Major challenges are in following directions: (1) Collecting sufficient trace data (2) Running the tools on the traces and extracting attack characteristics (3) Clustering similar attack features into one and identifying test sets that will be included in the benchmark.

Our existing methods have limitations. They depend on the trace analysis to extract information about the attacks. But this approach has a disadvantage; we can only see attacks that are in the trace. In addition to that only limited numbers of real network traces are publicly available. Besides this, traces might not be complete, some part of the attack/legitimate traffic might have been dropped on their way to capturing point.

In spite of limitations explained above, AProf toolkit successfully detected the attacks in public traces and in simulation traces, found more attacks in them than the previous analysis methods, and extracted useful information about the attack behavior. By running the toolkit on public traces, clustering the attack characteristics and combining those results with the characteristics extracted from network literature and experiments, we believe we can characterize the attack behavior and attack part of the benchmarks can be built.

REFERENCES

[1] Jelena Mirkovic, Sven Dietrich, David Dittrich and Peter Reiher. Internet Denial of Service, Attack and Defense Mechanisms. 2005.

[2] David Moore, Geoffrey Voelker, and Stefan Savage. Inferring Internet Denial-of-Service Activity. USENIX Security 2001.

[3] Insecure.org. http://www.insecure.org/sploits/land.ip.DOS.html.

[4] IANA. Special-Use IPv4 Addresses, September 2002. http://www.rfc-editor.org/rfc/rfc3330.txt.

[5] Colleen Shannon, David Moore, and K. C. Claffy. Beyond folklore: observations on fragmented traffic. *IEEE/ACM Transactions on Networking (TON)*, 10(6), 2002.

[6] Ion Stoica and Hui Zhang. Providing Guaranteed Services Without Per Flow Management. In *SIGCOMM'99*, pages 81–94, 1999.

[7] NLANR Measurement and Network Analysis Group. Auckland VIII. http://pma.nlanr.net/Special/auck8.html.

[8] Emulab – Network Emulation Testbed. http://www.emulab.net.

[9] NLANR Measurement and Network Analysis Group. Measurement and Network Analysis Group. http://pma.nlanr.net/Special/index.html.

[10] Project 2061. Benchmarks On-Line. http://www.project2061.org/publications/bsl/online/bolintro.htm

[11] Standard Performance Evaluation Corporation. SPEC Web Page. http://www.spec.org.

[12] Transaction Processing Performance Council. TPC Web Page. http://www.tpc.org/.

[13] D Moore, G Voelker, and S Savage. Inferring Internet Denial-of-Service Activity. Proceedings of the 2001 USENIX Security Symposium, 2001.

[14] Kun chan Lan, Alefiya Hussain, and Debojyoti Dutta. The Effect of Malicious Traffic on the Network. In Passive and Active Measurement Workshop (PAM), April 2003.

[15] Alefiya Hussain, John Heidemann, Christos Papadopoulos. A Framework for Classifying Denial of Service Attacks. June 2003

[16] IRTF TMRG group, The Transport Modeling Research Group's Web Page http://www.icir.org/tmrg.