ABSTRACT

A network is an architecture with a lot of scope for attacks. The rise in attacks has been growing rapidly. Denial of Service (DoS) attack and Distributed Denial of Service (DDoS) attack are among the common attacks that are causing disturbances for the network and its corresponding resources. DoS and DDoS, both attacks have significant roles. There are many mechanisms that detect and prevent these attacks. Packet marking schemes are the most successful implementation towards preventing DoS attacks. However, there are some issues with the packet marking schemes, as DoS and DDoS attacks are becoming more sophisticated. Packet marking schemes can traceback the source of attack. In this process the key challenge is to reduce the number of packets that are involved in the traceback of these attacks. The number of packets that are originating from the individual sources are not sufficient to traceback the attackers and thus an efficient packet marking scheme is required in this context.

The main aim of this project is to reduce the number of packets involved across the efficient traceback using Hop-based packet marking technique. The total number of packets involved for a successful traceback of the attacker is at most equal to the hop distance from the attacker to the victim which is equal to 25 [Savage, 2001], [David, 2011]. A simulation is created in this project using Network Simulator 2 (NS2) and the evaluated results prove that Hop-based packet marking technique will reduce the number of packets required for traceback and also detect the DoS/DDoS attacks efficiently.
# Table of Contents

ABSTRACT .......................................................................................................................... ii

Table of Contents .................................................................................................................. iii

List of Figures ......................................................................................................................... v

List of Tables .......................................................................................................................... v

1. Introduction ....................................................................................................................... 1
   1.1 Previous work on IP traceback .................................................................................. 2
   1.2 Aim and Objectives ................................................................................................. 4

2. Existing Packet Marking Schemes ................................................................................ 5
   2.1 Node Append............................................................................................................. 5
   2.2 Node Sampling ......................................................................................................... 5
   2.3 Edge Sampling ......................................................................................................... 6
   2.4 Probabilistic Packet Marking (PPM) ....................................................................... 8
   2.5 Deterministic Packet Marking (DPM) ..................................................................... 9
   2.6 Route Based Packet Marking (RBPM) .................................................................. 10

3. Methodology ................................................................................................................... 12
   3.1 Hop - Based Packet Marking algorithm .................................................................. 12

4. Implementation ............................................................................................................... 17
   4.1 Network Simulator (NS-2) .................................................................................... 17
   4.2 Simulation ................................................................................................................. 20
   4.3 Traceback mechanism ............................................................................................. 21
   4.4 Application in the case of attacks ......................................................................... 23

5. Results ............................................................................................................................ 25
   5.1 Simulation 1 ............................................................................................................ 25
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IP Traceback system</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Route Based Packet Marking</td>
<td>11</td>
</tr>
<tr>
<td>3</td>
<td>Functionality of a router implementing packet marking</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>IPv4 packet header</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>Behavior of the fields MHC, MRK and TTL</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>Hop-Based Packet Marking algorithm</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>Flowchart for Hop-Based Packet Marking algorithm</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>Network Simulator</td>
<td>17</td>
</tr>
<tr>
<td>9</td>
<td>NS-2 Trace file format</td>
<td>18</td>
</tr>
<tr>
<td>10</td>
<td>Modified trace file format</td>
<td>18</td>
</tr>
<tr>
<td>11</td>
<td>A glimpse of trace file</td>
<td>19</td>
</tr>
<tr>
<td>12</td>
<td>Simulated network in NS-2</td>
<td>20</td>
</tr>
<tr>
<td>13</td>
<td>Trace file with unique markings</td>
<td>22</td>
</tr>
<tr>
<td>14</td>
<td>IP traceback to the source</td>
<td>23</td>
</tr>
<tr>
<td>15</td>
<td>Simulated network under attack in NS-2</td>
<td>24</td>
</tr>
<tr>
<td>16</td>
<td>A simulated network with 10 hops</td>
<td>26</td>
</tr>
<tr>
<td>17</td>
<td>Threshold value vs Hop distance</td>
<td>27</td>
</tr>
<tr>
<td>18</td>
<td>Marketing probability vs Threshold value</td>
<td>29</td>
</tr>
<tr>
<td>19</td>
<td>Packets marked by intermediate routers</td>
<td>31</td>
</tr>
</tbody>
</table>
Figure 20: Trace route to Google.................................................................36

Figure 21: Trace route to Japan National Tourism Organization..........................37

Figure 22: Trace route to Incredible India..........................................................37

Figure 23: Trace route to France Guide...............................................................38
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 1: Trace data for IP traceback</td>
<td>22</td>
</tr>
<tr>
<td>Table 2: Threshold value vs Hop distance</td>
<td>26</td>
</tr>
<tr>
<td>Table 3: IP traceback vs Threshold value</td>
<td>28</td>
</tr>
<tr>
<td>Table 4: Number of packets marked by the intermediate nodes</td>
<td>30</td>
</tr>
</tbody>
</table>
1. Introduction

The growth in internet technologies is increasing the risk of attacks exponentially. Although there are a lot of implementations towards identifying these attacks, Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks are increasing rapidly across the internet. The name Denial of Service (DoS) itself states that it is a denial of an approved service to a legitimate user. A DDoS attack is a collective job of a group of attackers to perform DoS attack with a common target. The most common method of DoS and DDoS attacks is to flood the target or network with unwanted traffic, causing interruptions to the communication of legitimate users. In addition, the severity of these attacks is also much more complex.

There have been a lot of incidents and reports against these attacks on high end applications like e-Bay, Yahoo, Google and their root DNS servers as well. Initially when the internet was designed, the main intention was to provide a communication service; however, with the increase in internet usage in later years, a lot of web based applications were added and thus the impact of attacks has also increased proportionally [Khan, 2011].

Identifying and tracing the attacker is considered as the major issue. The source is taken under the security polices of the network and block them permanently. IP traceback is a successful implementation in this context, where the victim can easily identify the attacker and also perform a traceback to its source. Thus required measures can be taken to avoid similar attack [Bellovin, 2006].

DDoS attacks are distributed in nature and thus they can send a maximum number of packets from different attackers to a victim. The net sum of packets creates multiple attacks on
the victim. In this context a traceback scheme which requires fewer packets is required for a perfect traceback of the attackers. Many IP traceback schemes were introduced and most of them require hundreds of packets to traceback the attackers. The effectiveness of an IP traceback mechanism mainly depends on how many packets are required against the traceback of attackers and effectively mitigating the DDoS attacks [Rui, 2006].

1.1 Previous work on IP traceback

IP traceback is used to find the path travelled by packets from source to destination. The source can be an attacker or a legitimate user. The concept of IP traceback was initially introduced by Burch and Cheswich [Savage, 2001]. They were successful in identifying the attack path by flooding the links recursively. The main disadvantage with this implementation is that the attack has to be alive during the traceback. This is not possible all the times and there are situations where IP traceback itself will cause a DoS attack [Perrig, 2005].

Figure 1 depicts network architecture, which might be similar to real time architecture. Consider a scenario consisting of client 2 sending a message (packets of data) to client 8, so the data might travel from client 2 to client 8 through various routers.

From Figure 1, the possible paths for the packets to travel from source to destination are


The receiver upon receiving the packets can find the above path(s) taken by the packets using IP traceback methods.
There are different IP traceback techniques which are classified into four different categories, namely Link Testing, Messaging, Logging and Packet Marking. Packet Marking technique stores a node’s identification data in the IP header of a packet. This stored data helps to trace the source of the packet or the path followed by the packet. This technique utilizes the existing fields in an IP header; therefore, there is no requirement of changing the structure of an IP header. An IP header has limited space in order to accommodate the traceback data. The data has to be fragmented to overcome this problem. There are different packet marking techniques proposed, which utilize the IP header field to store the traceback data efficiently and help to find the source of the data or the path the packets have traversed.
1.2 Aim and Objectives

Aim:

The primary aim of this project is to evaluate the importance of IP traceback and implement a packet marking technique to identify the source of DoS and DDoS attacks. In this context a Hop-Based Packet Marking technique is proposed for an efficient IP traceback. The primary objective of this technique is to perform a traceback using less number of packets and thereby also mitigate the DoS and DDoS attacks.

Objectives:

Following are the key objectives for this project

• To critically review different IP traceback algorithms and identify the key limitations among them

• To design a packet marking technique and explain the complete flow of the application

• To create a simulation using Network Simulator 2 (NS2) to demonstrate the IP traceback using the proposed packet marking algorithm

• To evaluate the performance of the algorithm by identifying the threats and find the source of the attacks

• To evaluate the results achieved and verify that IP traceback using Hop-Based packet marking technique is an efficient mechanism in identifying the attackers and also mitigate the DoS and DDoS attacks with less number of packets required
2. Existing Packet Marking Schemes

2.1 Node Append

Node Append [Savage, 2001] is one of the simplest packet marking techniques. In this method the complete route travelled by the packet is recorded. Every node the packet travels through, appends its address to the packet. This information helps the attacker to construct the path to the attacker.

This method of packet marking is robust in nature. The victim requires just one packet to trace the sender of the packet. However, this method has certain limitations. Since each and every node appends its address to the packet, this increases the overhead at every node in the network. The distance from the source to the destination is unknown, as the path of travel for a packet may change during the course due to various network conditions. Hence it is not possible to reserve enough space on the packet to accommodate the travel path of the packet to the destination. The attacker might fill the space with misleading information even if enough space is ensured in the packet.

2.2 Node Sampling

Node Sampling [Savage, 2001] was introduced in order to reduce the router overhead and also the space in a packet to record the address of the every router it passes through. In this method every packet contains enough space to store the address of a single router. In general the size of a router address is 32 bits in case of an IPv4 network. Every node or a router in a network contains the packet marking algorithm, which is based on a probability to mark the packet or not.
The probability is chosen randomly by the node. If the probability of the algorithm is to mark the packet, then the node inserts its 32 bit address into the packet and forwards it to the next node in the network. A packet which is once marked by a node is not marked again by any other node in the network.

Upon receiving enough number of packets, the receiver can construct the path based on the markings on the packets received. In order to construct the path to the sender, the receiver should collect packets marked by all different nodes in the path to the sender. As the travel path of the packet is not predefined, it becomes difficult to find the exact number of nodes in the path reconstruction. Another limitation of this packet marking algorithm is that the probability of marking is either 0 or 1, which means the probability of marking is 0.5. Hence it becomes difficult to assume that the node marks the packet at instances. The sender should also send enough number of packets required to construct the path. The assumption of a single route for the packets to traverse is not a good assumption to implement this method.

2.3 Edge Sampling

In Node Sampling methodology it is seen that only a single router marks the packet with a probability in the path of its travel from source to destination. It is difficult for the victim to determine how far the marking router is. Edge Sampling [Savage, 2001] overcomes this problem. In this packet marking technique, two routers mark a packet forming an edge. A packet participating in a network employing Edge Sampling should be able to accommodate two 32 bits space, i.e., 64 bits to store the routers’ addresses and also an additional 8 bit space to store the distance between the marking routers.
In this technique every router that receives a packet it chooses a random number among 0 and 1. If the selected number is less than the marking probability, then the router inserts its 32 bit address into the first address space and sets the distance field value to 0. If the selected value is greater than the marking probability and the distance field equals to 0, then the router understands that the packet was marked by the previous router and inserts its 32 bit address into the second address space forming an edge between the previous router and the current router. If the router decides not to mark the packet, it increments the distance field value. The distance field provides the distance between the victim and the edge marking in the packet.

In this method, since there are markings of two different routers and the distance between the victim and the edge marking, it helps the victim to analyze the path to the attacker with samples of different edge markings in the path. The victim requires samples of all the edges in the path to trace the attacker. However, in case of multiple attackers it becomes a tedious task to derive paths to all the attackers, as each one’s path is independent. The number of packets required to trace the attackers in this scenario is also large. If a packet is spoofed in between the course of travel to the victim, the distance field is either equal or more than the actual distance.

In this method, the X number of packets required to traceback to the source of the sender can be obtained by

\[ E(X) \leq \frac{\ln (d)}{p(1-p)^{d-1}} \]  \hspace{1cm} (1)

where d is the number of routers the packet has to pass through from the source to the destination and p is the probability of marking the packet at each router. Consider the probability of marking
the packet as $p = 1/10$ and $d=10$, then the receiver requires at least 75 packets to be collected to trace the route to the sender or attacker.

2.4 Probabilistic Packet Marking (PPM)

Probabilistic Packet Marking (PPM) is an extension to Edge Sampling method. There is not enough space available in a packet header to store the address of a router, which is of 32 bits. Instead the address is fragmented and stored into the packet headers. The forwarding mechanism of routers probabilistically marks the packets with fragments of its identification. The marked packets contain only partial information of the path. This reduces the storage overhead in the packets. The receiver has to receive enough number of packets to construct the path, as each packet contains only partial information of the path.

The minimum number of packets required to construct the path can be obtained using

$$E(X) \leq \frac{k \ln(kd)}{p(1-p)^{d-1}}$$  \hspace{1cm} (2)

where,

$k$ is the number of fragments the address of the router is fragmented into

$d$ is the number of hops the packet has taken from the source to the destination

$p$ is the probability of marking the packet

Consider, the address of the router is fragmented into 8 fragments ($k = 8$), $d = 10$ and $p = 1/20$. Thereafter the receiver requires on an average of 1100 packets to construct the path to the source. If the distance increases to 15 hops, then the receiver might require about 1600 packets to
construct the path. It is also possible that subsequent routers overwrite the marked data in the packets, which might lead to loss of information and the packets required may also increase.

2.5 Deterministic Packet Marking (DPM)

Deterministic Packet Marking [Ansari, 2003] is a packet marking technique. In DPM the edge routers which are closer to the source of the packets are used to mark the packets. In this mechanism the edge routers are considered as interfaces. These interfaces can distinguish between incoming and outgoing packets. When a packet is sent onto the network through this interface, it is considered as an incoming packet. Once an incoming packet arrives at an interface, the interface address is stored in the packet. Generally an interface address is of 32 bits length. A single packet cannot carry all the 32 bits of data, which overloads the packet header, so the marking data has to be fragmented. It requires 16 bit of packet header space and 1 bit space to store the flag value. Therefore it requires 17 bits of space in total to store the interface data. The address of the interface is divided into two parts, each consisting of 16 bits, i.e., 0 - 15 bits and 16 - 31 bits. Once a packet arrives at the interface one of these two parts is stored into the packet header. The part of address to be stored is chosen deterministically. The flag bit is set to 0 if the 0 – 15 bits of the address is stored into the packet, else the flag bit is set to 1 if the 16 – 31 bits of the address is stored into the packet. Once these two packets arrive at the destination, the IP address of the interface is retrieved by reassembling the address parts based on the flag bits of each packet.

This mechanism is simple to implement as it requires minimum of 2 packets to send the edge router’s address. It does not involve router overhead or the network overhead.
2.6 Route Based Packet Marking (RBPM)

Route Based Packet Marking (RBPM) [Doss, 2006] is a packet marking technique to trace back to the source address. The primary objective of Route Based Packet Marking is to reduce the packet overheads. In a network implementing Route Based Packet Marking (RBPM), the nodes in the network are to be pre-configured with node identification (ID) which plays a key role in this method. Each node in the network is assigned an ID of smaller length than its actual IP address, based on the size of the network. The size of the network is determined by the total number of nodes in the network. The length of a node ID is ‘N’ bits for a network consisting of X nodes. The value of N is determined by

\[
N \geq \log_2 X > N - 1
\]  

(3)

A network which implements Route Based Packet Marking can help to identify the route a packet has taken from its edge router to its destination router. In this method, each and every node the packet passes through, the ID of the node is appended to the packet. The receiver after receiving the packet can trace the route the packet has taken based on the appended node IDs’. As the length of each node ID is relatively small, appending multiple node IDs to the packet header does not require much space. This reduces the packet overhead.

Every node appends its node ID and also a flag bit to the packet. The value of the flag is set by the node. During the travel of a packet from source to the destination, the edge router appends its node ID and a flag bit which is set to ‘1’. This indicates that it is the first node the packet has travelled in the network. The intermediate nodes in the network append its node ID and the flag bit, which is set to ‘0’. The flag bit ‘0’ indicates that the node which has set the flag bit to ‘0’ is an intermediate node during the packet’s travel to the destination. This helps the
receiver to trace the route the packet has taken from the source to the destination. This can be understood from the Figure 2.

![Figure 2: Route Based Packet Marking](image)

Appending the node IDs to the packet increases the size of the packet. In an IPv4 network implementing Route Based Packet Marking, if the appending of node IDs to the packet header crosses the Maximum Transmission Unit (MTU) of the packet then the packet is fragmented. An IPv6 network does not allow implementing fragmentation at the nodes.

The main advantage of Route Based Packet Marking is that, only a single packet is required by the destination to construct the packet of the packet to its edge router. Compared to any other traditional packet marking techniques this is more advantageous as it requires a single packet to traceback. But marking every single packet at each and every node increases the overhead of the network as well as the nodes. If the packets are marked randomly at certain frequency, then the overhead can be reduced. But marking at certain frequency can be changed by the edge router, which can lead to attacks. Appending of multiple node IDs to the packet, may cross the Maximum Transmission Unit (MTU) of the packet, which leads to packet fragmentation. This adds to the disadvantage of this method.
3. Methodology

Internet is a large network of interconnected networks serving millions of users. On an average most of the packets on the internet take at most 25 hops to reach the destination [Savage, 2001], [David, 2011]. This has been confirmed by few tests performed in Appendix A. Therefore, this value is considered to be the threshold value for the Hop-Based packet marking algorithm which plays a vital role.

The functionality of a router implementing packet marking technique is shown in Figure 3. Every router in the network executes this packet marking algorithm upon receiving the packet and marks the packet or passes it, based on the output of the algorithm. The receiver might require only packets with unique markings of all the received packets to construct the path to the source or attacker. This might be equal to the total number of hops between the source and the destination.

![Figure 3: Functionality of a router implementing packet marking](image)

3.1 Hop-Based Packet Marking algorithm

The framework for Hop-Based packet marking algorithm has been adapted from the paper [Saurabh, 2012]. It uses the Identification field of the packet header in an IPv4 datagram shown in Figure 4 to store the identification data of the routers. The stored information is known as marked data. The length of the Identification field is 16 bits. It is split into two parts to store the marking node’s identification data (MRK) and the Marked Hop Count (MHC). The Marked
Hop Count (MHC) is used to determine the number of hops between the marking router and the destination. Time to Live (TTL) value depicts the life of the packet on a network in terms of hops. It is known that every router decrements the TTL value of the packet before forwarding the packet to the next router.

<table>
<thead>
<tr>
<th>Version</th>
<th>Header Length</th>
<th>Type of Service (TOS)</th>
<th>Total Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Identification</td>
<td>Flags</td>
<td>Fragment Offset</td>
</tr>
<tr>
<td>TTL</td>
<td>Protocol</td>
<td>Header Checksum</td>
<td></td>
</tr>
</tbody>
</table>

**Source Address**

**Destination Address**

Figure 4: IPv4 packet header

Initially the MHC is set equal to the Time to Live (TTL) value of the packet by the source. Marked node (MRK) field is used to store the information of the marking router. In order to understand the behavior of the fields MHC, MRK and TTL, consider a scenario of a source, sending a packet to a receiver which passes through intermediate routers Router 1, Router 2 and Router 3 as shown in Figure 5. Let Router 2 be the router which marks the packet.

![Diagram of packet traversal](image)

**Sender**

- TTL = 32
- MHC = 32
- MRK = -1

**Router 1**

- TTL = 31
- MHC = 32
- MRK = -1

**Router 2**

- TTL = 30
- MHC = 0
- MRK = 2

**Router 3**

- TTL = 29
- MHC = 1
- MRK = 2

**Receiver**

- TTL = 28
- MHC = 2
- MRK = 2

Figure 5: Behavior of the fields MHC, MRK and TTL

- The values of the fields MRK and MHC of a packet do not change until the packet is marked though the value of TTL decrements for every hop
• The Router 2 that marks the packet by inserting its identification data into MRK field, sets the value of MHC to 0 and the value of TTL decrements as usual for every hop.

• After the packet is marked by Router 2, the value of MHC is incremented by 1 unit and the value of TTL is decremented by 1 unit for every hop, while the value of MRK is not altered.

From the Figure 5, the receiver can identify that the router with identification value 2 (MRK = 2) has marked the packet is 2 hops away (MHC = 2). If the packet is not marked by the Router 2 the values of MHC and MRK remain constant all the way. The receiver can identify the hop distance between the sender and the receiver from the difference of MHC and TTL of the unmarked packet.

An intermediate router upon receiving the packet, checks if the packet has been marked or not. If the packet has not been marked then the router selects a random number ‘x’ between 1 and 25 (threshold value). The value 25 is considered maximum for the random number selection as most of the packets on internet reach destination in 25 hops [Savage, 2001], [David, 2011]. The packet ID is a unique ID assigned to the packet by the source. A modulus operation is performed on the packet ID with ‘x’. Hop count of the packet at the current router is obtained, which is the difference between the Marked Hop Count (MHC) and Time to Live (TTL). If the hop count is equal to the result of (the modulus operation) + 1, then the packet is marked. Here the value of modulus operation is incremented by 1 as the hop count can never be equal to zero.

Consider the values TTL = 25, MHC = 32 and packet ID = 66 of a packet received by a router. Let the random number generated be x = 10, so the result of the operation (packet ID % x) + 1 is 7, which is equal to the hop count (MHC – TTL). As the marking criterion is satisfied.
the packet is marked by inserting the router’s address into the MRK field of the packet and sets the Marked Hop Count (MHC) value to 0.

If the hop count is not equal to the result of (the modulus operation) + 1, then the router forwards the packet to the next router. Consider the values TTL = 16, MHC = 32 and packet ID = 163 of a packet received by a router. Let the random number generated be x = 15 and the result of the operation (packet ID % x) + 1 is 14, which is not equal to the value of the hop count (MHC – TTL) = 16. So the packet is not marked and forwarded to the next router in the path.

If the packet has already been marked, then the router increments the Marked Hop Count (MHC) value of the packet by one unit and forwards it to the next router in the path.

1. if packet is not marked
2. select random number 'x' between 1 and 25
3. hop count = MHC - TTL
4. if ((packet ID % x) + 1 == hop count)
5. insert router address into packet header
6. set MHC = 0
7. end if
8. else
9. increment MHC
10. end if

Figure 6: Hop-Based Packet Marking algorithm

The receiver can distinguish between marked packets and unmarked packets from the information available in the MRK field of the packet. The address of the router that has marked the packet and the Marked Hop Count (MHC) can be obtained from the marked packets. The MHC values of the marked packets depict the number of hops between the router that has marked the packet and the receiver. Based on the values of MRK and MHC fields of the packets received path to the source can be constructed.
The Hop-Based packet marking algorithm shown in Figure 6 is represented with a flowchart in Figure 7.

Figure 7: Flowchart for Hop-Based Packet Marking algorithm
4. Implementation

4.1 Network Simulator (NS-2)

This technique was implemented and tested on a simulated network using Network Simulator (version 2) popularly known as NS-2. NS-2 is a variant of a simulator named REAL developed in 1989 [Simulator]. It was developed by Information Sciences Institute (ISI) at the University of Southern California (USC).

NS-2 is an event driven simulator used to study real time networks. It is used in most of the networking researches. It supports simulation of various protocols over wide variety of networks such as wired, wireless, mobile, satellite, etc.

NS-2 consists of two languages namely C++ and Object-oriented Tool Command Language (OTcl). C++ is the backend of the simulator, while OTcl is the front end. The frontend and backend are linked together using TclCL. The user’s program is written in Tcl and it is provided to the simulator using the command ‘ns’ followed by the script name. The simulator reads the file content and simulates a network based on the time driven events provided in the script.

![Figure 8: Network Simulator](image-url)

Figure 8: Network Simulator
NS-2 generates a trace file reflecting all the events of the simulation. The trace file helps the user to understand the various events involved in the simulation. It also supports a network animator called NAM, which provides a graphical representation of the simulation trace file. This system can be seen in Figure 8. The general format of the trace file is shown in Figure 9. In order to show the packet markings the general trace file format is modified, which is shown in Figure 10. The network animation is shown on a user interface which provides user to view the simulation with control buttons such as play, stop, forward, rewind, etc as shown in Figure 12.

<table>
<thead>
<tr>
<th>Type Identifier</th>
<th>Time</th>
<th>Source Node</th>
<th>Destination Node</th>
<th>Packet Name</th>
<th>Packet Size</th>
<th>Flags</th>
<th>Flow ID</th>
<th>Source Address</th>
<th>Destination Address</th>
<th>Sequence Number</th>
<th>Packet Unique ID</th>
</tr>
</thead>
</table>

Figure 9: NS-2 Trace file format

<table>
<thead>
<tr>
<th>Type Identifier</th>
<th>Time</th>
<th>Source Node</th>
<th>Destination Node</th>
<th>Packet Name</th>
<th>Packet Size</th>
<th>Flags</th>
<th>Flow ID</th>
<th>Source Address</th>
<th>Destination Address</th>
<th>Sequence Number</th>
<th>Packet Unique ID</th>
<th>Time to Live</th>
<th>Marked Node</th>
<th>Marked Hop Count</th>
</tr>
</thead>
</table>

Figure 10: Modified trace file format

*Type Identifier:* It shows the type of event for a packet. The identifiers generally used are

- ‘+’ represents packet enqueue event
- ‘-’ represents packet dequeue event
- ‘r’ represents packet received event
- ‘d’ represent packet drop event

*Time:* It denotes the time the tracing event occurred in the simulation

*Source Node and Destination Node:* Denotes the ID of the link where the tracing has occurred

*Packet Name:* It denotes the type of packet

*Packet Size:* Size of the packet

*Flags:* It denotes any flags if specified

*Flow ID:* It shows the flow id of the packets specified the packet header by the source

*Source Address and Destination Address:* The source and destination address of the packet specified in the packet header

*Sequence Number:* It denotes the sequence number of the packet specified in the packet header
Packet Unique ID: It is the unique id of a packet

Time to Live (TTL): It denotes the TTL value of the packet in the current tracing event

Marked Node (MRK): It denotes if a packet has been marked or not. The default value is ‘-1’ which depicts that the packet is not marked

Marked Hop Count (MHC): It denotes the number of hops the packet has taken after it was marked to the current tracing event. The default value is set equal to the original TTL value by the sender.

A glimpse of the trace file data for this simulation can be seen in Figure 11. It is of the format shown in Figure 10.

Figure 11: A glimpse of trace file

Consider line 3 from Figure 11

r 0.5816 18 24 cbr 500 - - - - - - 1 0.0 25.0 10 699 21 MRK : -1 MHC : 32

It means that a packet is received at node address 24 from node address 18 at 0.5816 seconds of the simulation. The packet received is a CBR packet of size 500 bytes without any flags. The packet’s flow id is 1 sent from port 0 of node address 0 and it is received at port 0 of
node address 25. The sequence number of the packet is 10 with packet ID 699. The current TTL value of the packet is 21. It is also seen that packet is not marked as MRK is shown as -1, so the MHC value is equal to the original TTL value of the packet at the source.

The required results from the trace file can be obtained using AWK scripting language [Teerawat]. The results can also be obtained using the Linux command-line utility command “grep”.

NS-2 requires a UNIX environment to be installed. It can be installed on Windows using a UNIX emulator named Cygwin. The current version ns-2.35 was used in this project on an Ubuntu 12.04 operating system.

4.2 Simulation

Figure 12: Simulated network in NS-2
The algorithm is implemented in a simulated wired network shown in Figure 12. The network consists of 29 nodes interconnected to each other. The node 0 is the source of packets and node 25 is the destination in the simulation. This network deploys dynamic routing, resembling to a real time network. Every intermediate node between the source and destination acts as a router. The code for the network in Figure 12 and the simulation is shown in Appendix B.

Every node in the simulated network showed in Figure 12 implements the packet marking algorithm. A node upon receiving a packet executes the piece of code shown in Appendix C, before forwarding the packet to the next node on the path.

The packet marking algorithm is embedded into the simulator, such that all the nodes in the simulated network execute the packet marking technique. The simulation can be started by the user by clicking the play button in the network animator interface shown in Figure 12. Once the simulation starts, the source node 0 starts sending the packets to the destination node 25 based on the time event set in the simulation code. The flow of packets from source to destination is shown in blue color.

4.3 Traceback mechanism

The receiver can trace the path to the source with the help of markings on the packets. Here as the algorithm is implemented into a simulation, all the simulation data is recorded into a trace file. An analysis of the trace file helps to construct the path to the source. Figure 13 shows filtered trace data with unique packet markings for the network shown in Figure 12. It is seen in Figure 13 that each unique packet marking has a unique Marked Hop Count (MHC), which represents the hop distance between the router that has marked the packet and the destination.
Figure 13: Trace file with unique markings

The packet marking data and the Marked Hop Count (MHC) data in Figure 13 is tabulated in Table 1 in increasing order of the Marked Hop Count (MHC).

Table 1: Trace data for IP traceback

<table>
<thead>
<tr>
<th>Packet ID</th>
<th>Time to Live (TTL)</th>
<th>Marking Node (MRK)</th>
<th>Marked Hop Count (MHC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>802</td>
<td>20</td>
<td>24</td>
<td>1</td>
</tr>
<tr>
<td>699</td>
<td>20</td>
<td>18</td>
<td>2</td>
</tr>
<tr>
<td>928</td>
<td>20</td>
<td>12</td>
<td>3</td>
</tr>
<tr>
<td>757</td>
<td>20</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>788</td>
<td>20</td>
<td>13</td>
<td>5</td>
</tr>
<tr>
<td>695</td>
<td>20</td>
<td>9</td>
<td>6</td>
</tr>
<tr>
<td>711</td>
<td>20</td>
<td>10</td>
<td>7</td>
</tr>
<tr>
<td>703</td>
<td>20</td>
<td>23</td>
<td>8</td>
</tr>
<tr>
<td>698</td>
<td>20</td>
<td>7</td>
<td>9</td>
</tr>
<tr>
<td>676</td>
<td>20</td>
<td>4</td>
<td>10</td>
</tr>
<tr>
<td>666</td>
<td>20</td>
<td>2</td>
<td>11</td>
</tr>
<tr>
<td>665</td>
<td>20</td>
<td>-1</td>
<td>32</td>
</tr>
</tbody>
</table>
Table 1 represents data from both marked and unmarked packet data in an organized way. It is observed from the data in Table 1 that the packet with packet ID 665 was not marked by any of the intermediate routers and it shows the original TTL value assigned to the packet by the source, i.e., MRK = -1 and MHC = 32. From the current TTL value 20 and MHC = 32 at the destination node, it is understood that the source is 12 hops away from the destination. Finally substantial information is available to construct the path to the source shown in Figure 14.

![Figure 14: IP traceback to the source](image)

In order to test the efficiency of the Hop-based packet marking algorithm simulations were conducted considering various threshold values and different number of hops between the source and the destination. The results for these simulations are discussed in the section 5.

### 4.4 Application in the case of attacks

The rise of attacks is growing exponentially. IP traceback is a successful technique to find the source of the attacks and determine the path travelled by the packets to perform the attack. This technique can be used in such scenarios to find the source of the attacks and the path taken by the packets from the attacker to the victim.
Consider a simulated network shown in Figure 15 with node 0 as the source and node 25 as the destination. Assume node 19 plays the role of an attacker. At a point of time stated in the simulation script the attacker node 19 performs a Denial of Service (DoS) attack by flooding the network with unwanted packets. The flow of packets sent by the attacker is shown in red color. It can be observed in the simulation that the packets sent from node 0 hardly reach the destination node 25, and most of them are dropped from the network which can be observed in Figure 15. This interrupts the communication between the source node 0 and destination node 25. The node 25 becomes the victim of the DoS attack. In such attack scenarios IP traceback technique can be used to find the source of attacks.
5. Results

The following simulations are conducted to provide the justification for the considered threshold value of 25 in this project. The results prove that the threshold value of 25 generates efficient results using the Hop-based packet marking technique.

The simulations conducted are

**Simulation 1:** To demonstrate the effectiveness of the threshold value 25 and also to find the minimum number of packets required to trace the source for different threshold values with varying number of hop distances between the source and the destination.

**Simulation 2:** To find the average number of packets required for various threshold values on the network shown in Figure 12.

**Simulation 3:** To find the marking probabilities of the intermediate routers between the source and the destination on the network shown in Figure 12, with the threshold value of 25.

5.1 **Simulation 1**

The purpose of this simulation is to find the minimum number of packets required to trace the source of the packets on a linear network consisting of hops 10, 15, 20 and 25 between the source and destination. The simulation was conducted with threshold values 10, 15, 20 and 25. A linear network consisting of 10 hops between the source and destination is shown in Figure 16. The results obtained from this simulation are tabulated in Table 2.
Figure 16: A simulated network with 10 hops

Table 2: Threshold value vs Hop distance

<table>
<thead>
<tr>
<th>Hop Distance</th>
<th>10</th>
<th>15</th>
<th>20</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td>10 hops</td>
<td>25</td>
<td>25</td>
<td>50</td>
<td>60</td>
</tr>
<tr>
<td>15 hops</td>
<td>$\infty$</td>
<td>100</td>
<td>100</td>
<td>125</td>
</tr>
<tr>
<td>20 hops</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>200</td>
<td>200</td>
</tr>
<tr>
<td>25 hops</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>$\infty$</td>
<td>300</td>
</tr>
</tbody>
</table>
From the results tabulated in Table 2, it is observed that the minimum number of packets required for hop distances greater than the threshold value is leading towards infinity. In spite of sending large number of packets, traceback was not possible for the hop distances greater than the threshold value. As discussed earlier that most of the packets on internet reach the destination within 25 hops. The same can be observed in Table 2, where for a threshold value of 25 the minimum number of packets required for hop distances 10, 15, 20, 25 are obtained without any hassles.

The data in Table 2 is represented in the form of a graph in Figure 17. It can be observed from the graph the required number of packets increases as the threshold value increases for the same number of hops. It is also observed that a threshold value satisfies the traceback for the hop distance between the source and destination is less than or equal to the threshold value.

![Figure 17: Threshold value vs Hop distance](image-url)
5.2 Simulation 2

The purpose of this simulation is to find the average number of packets required for a successful traceback to the source for threshold values 10, 15, 20 and 25. This simulation was conducted on the network shown in Figure 12. This simulation also shows the packet marking efficiency of the Hop-based packet marking algorithm. The number of packets sent from source to destination in this simulation is 25, 50, 60, 75 and 100. The results obtained from this simulation are tabulated in Table 3.

Table 3: IP traceback vs Threshold value

<table>
<thead>
<tr>
<th>Threshold Value</th>
<th>10</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>25</td>
<td>50</td>
</tr>
<tr>
<td>Packets Sent</td>
<td>25</td>
<td>50</td>
</tr>
<tr>
<td>Packets Received</td>
<td>25</td>
<td>50</td>
</tr>
<tr>
<td>Marked Packets</td>
<td>17</td>
<td>36</td>
</tr>
<tr>
<td>Unmarked Packets</td>
<td>8</td>
<td>14</td>
</tr>
<tr>
<td>Tracing</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Threshold Value</th>
<th>20</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>25</td>
<td>50</td>
</tr>
<tr>
<td>Packets Sent</td>
<td>25</td>
<td>50</td>
</tr>
<tr>
<td>Packets Received</td>
<td>25</td>
<td>50</td>
</tr>
<tr>
<td>Marked Packets</td>
<td>13</td>
<td>37</td>
</tr>
<tr>
<td>Unmarked Packets</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>Tracing</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

From the results in Table 3, it is seen that traceback is possible for different set of packets with threshold values of 15, 20 and 25 using the Hop-based packet marking algorithm. It can also be observed from the results in Table 3 that on an average about 65-75% of the packets received at the destination are marked. This proves the marking efficiency of the algorithm. The traceback
is possible for an average minimum number of packets 60 for threshold values 20 and 25. This also supports the consideration of the threshold value 25 for the Hop-Based packet marking algorithm, resulting in positive results.

The tabulated results in Table 3 are represented in the form of a graph in Figure 18. As seen in the graph in Figure 18 it is observed that on an average about 65-75% of the packet flow sent is marked for the threshold values 10, 15, 20 and 25. This rate of marking is seen constant for different set of packets 25, 50, 60, 75 and 100.

![Figure 18: Marking probability vs Threshold value](image)

5.3 Simulation 3

The purpose of this simulation is to find the marking probabilities of the intermediate nodes between the source and the destination. This simulation is carried out on the network shown in Figure 12 with the optimum threshold value 25 obtained from Simulation 1 and
Simulation 2. In this simulation about 250 packets are sent from the source to the destination. The number of packets marked by each of the intermediate nodes is tabulated in Table 4.

The results data in Table 4 indicate that about 67% of the packets are marked. It can also be observed that the edge routers of the source are marking high probability of packets compared to the other routers in the path. This even helps the victim to find the source easily in the case of attacks, as the results are narrowing down directly to the edge routers of the source.

Table 4: Number of packets marked by the intermediate nodes

<table>
<thead>
<tr>
<th>Node ID</th>
<th>Number of packets marked by the node</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node 2</td>
<td>40</td>
</tr>
<tr>
<td>Node 4</td>
<td>32</td>
</tr>
<tr>
<td>Node 7</td>
<td>19</td>
</tr>
<tr>
<td>Node 23</td>
<td>20</td>
</tr>
<tr>
<td>Node 10</td>
<td>13</td>
</tr>
<tr>
<td>Node 9</td>
<td>9</td>
</tr>
<tr>
<td>Node 13</td>
<td>7</td>
</tr>
<tr>
<td>Node 14</td>
<td>8</td>
</tr>
<tr>
<td>Node 12</td>
<td>9</td>
</tr>
<tr>
<td>Node 18</td>
<td>6</td>
</tr>
<tr>
<td>Node 24</td>
<td>5</td>
</tr>
</tbody>
</table>

The tabulated results in Table 4 are represented in the form of a graph in Figure 19. It is also observed from the graph that the edge routers mark large amount of packets, which helps to track the source easily. This also supports the intention of the algorithm to not overwrite the markings on the packets.
Figure 19: Packets marked by intermediate routers
6. Conclusion and Future work

The rise of Denial of Service (DoS) and Distributed Denial of Service (DDoS) attacks on the internet are growing rapidly causing troubles to the network and also decreasing the network performance. IP traceback is a method to traceback to the source of the packets. Packet marking schemes are the most successful implementation towards preventing DoS attacks by tracing to the source of attacks. Most of the existing packet marking techniques requires huge number of packets to traceback the source. They also have drawbacks such as router overhead, packet header overload, network overhead, etc.

The aim of this project is to reduce the number of packets involved across the efficient traceback using Hop-based packet marking technique. The algorithm was implemented on a simulated wired network in NS-2. This algorithm has shown results with efficient marking of packets. It marks on an average of about 65-75% of the packet flow. The algorithm has also shown that the edge routers mark large number of packets than other routers on the path, thus helping to trace the source easily. The results obtained show that this algorithm requires less number of packets to traceback to the source of attacks, compared to the existing techniques. Though the receiver receives duplicate marking data of the intermediate routers, it requires only unique marking data per router to perform successful traceback. This indicates that the receiver requires packets marked with unique data equal to the number of intermediate routers between the source and the destination.

The algorithm can be enhanced in terms of security constraints, such that an attacker cannot modify any marked information on the packet. Storing the 32-bit address of a router may sometimes overload the packet header, which can be enhanced using special encoding...
techniques. This algorithm was tested on a wired network, which can be extended to wireless networks, mobile networks, etc. An IPv4 packet was taken into consideration in this simulation. The algorithm should be extended to IPv6 packet, as today IPv6 is widely in use. If an intermediate router can identify the flow of packets from source to destination and mark only a single packet from the flow, then the total number of packets required by the receiver will be reduced as well as the overall work load of the network reduces, improving network performance. Another key point to be considered for the future work is, if the attackers play the role of intermediate routers, what number of packets would be required by the receiver to traceback successfully.
7. References


Available: http://repository.cmu.edu/cgi/viewcontent.cgi?article=1019&context=ece

Last accessed: March 25, 2012


Last accessed: March 25, 2012


Appendix A

On an average most of the packets take atmost 25 hops on a network to reach the destination. A test was conducted to verify this value using ‘tracert’ command on windows command prompt. The result of this test can be seen in Figure 20, Figure 21, Figure 22 and Figure 23. In this test the number of hops taken for a packet to travel to various web servers across the world is obtained.

It took 11 hops for a packet to reach Google’s server located in Dallas, shown in Figure 20.

![Figure 20: Trace route to Google](image-url)
It took 18 hops for a packet to reach Japan National Tourism Organization’s server which is located in Japan, shown in Figure 21.

![Figure 21: Trace route to Japan National Tourism Organization](image1)

It took 22 hops for a packet to reach Incredible India’s server located in India, shown in Figure 22.

![Figure 22: Trace route to Incredible India](image2)
It took 24 hops for a packet to France Guide’s server located in France, shown in Figure 23.

Figure 23: Trace route to France Guide

Based on the results from Figure 20, Figure 21, Figure 22 and Figure 23 it is deduced that on an average a packet takes about 20 hops to reach the destination. It takes at most 25 hops in certain cases.
Appendix B

# Create a simulator object
set ns [new Simulator]

# Dynamic routing
$ns rtproto DV

# Open the NAM trace file
set nf [open MyProgram.nam w]
$ns namtrace-all $nf

# Record all the activity in trace file
set tracefd [open MyProgram.tr w]
$ns trace-all $tracefd

# Define a 'finish' procedure
proc finish {} {
    global ns nf tracefd
    $ns flush-trace
    # Close the NAM trace file
    close $nf
    close $tracefd
    # Execute NAM on the trace file
    exec nam MyProgram.nam &
    exit 0
}

# Create nodes
for {set i 0} {$i < 29} {incr i} {
    set n($i) [$ns node]
}

# Color nodes
$n(0) color purple
$n(25) color purple
$n(19) color red

# Label nodes
$n(0) label "Source"
$n(25) label "Destination"
$n(19) label "Attacker"

# Create links between the nodes
$ns duplex-link $n(0) $n(2) 10Mb 10ms DropTail
$ns duplex-link $n(1) $n(2) 10Mb 10ms DropTail
$ns duplex-link $n(2) $n(3) 10Mb 10ms DropTail
$ns duplex-link $n(2) $n(4) 10Mb 10ms DropTail
$ns duplex-link $n(2) $n(5) 10Mb 10ms DropTail
$ns duplex-link $n(5) $n(4) 10Mb 10ms DropTail
$ns duplex-link $n(5) $n(16) 10Mb 10ms DropTail
$ns duplex-link $n(5) $n(17) 10Mb 10ms DropTail
$ns duplex-link $n(4) $n(7) 10Mb 10ms DropTail
$ns duplex-link $n(21) $n(6) 10Mb 10ms DropTail
$ns duplex-link $n(7) $n(8) 10Mb 10ms DropTail
$ns duplex-link $n(7) $n(23) 10Mb 10ms DropTail
$ns duplex-link $n(8) $n(23) 10Mb 10ms DropTail
$ns duplex-link $n(23) $n(10) 10Mb 10ms DropTail
$ns duplex-link $n(10) $n(11) 10Mb 10ms DropTail
$ns duplex-link $n(11) $n(28) 10Mb 10ms DropTail
$ns duplex-link $n(11) $n(20) 10Mb 10ms DropTail
$ns duplex-link $n(28) $n(20) 10Mb 10ms DropTail
$ns duplex-link $n(12) $n(21) 10Mb 10ms DropTail
$ns duplex-link $n(21) $n(22) 10Mb 10ms DropTail
$ns duplex-link $n(11) $n(9) 10Mb 10ms DropTail
$ns duplex-link $n(9) $n(13) 10Mb 10ms DropTail
$ns duplex-link $n(13) $n(14) 10Mb 10ms DropTail
$ns duplex-link $n(14) $n(15) 10Mb 10ms DropTail
$ns duplex-link $n(14) $n(12) 10Mb 10ms DropTail
$ns duplex-link $n(15) $n(12) 10Mb 10ms DropTail
$ns duplex-link $n(21) $n(19) 10Mb 10ms DropTail
$ns duplex-link $n(12) $n(18) 1Mb 10ms DropTail
$ns duplex-link $n(18) $n(24) 1Mb 10ms DropTail
$ns duplex-link $n(24) $n(27) 10Mb 10ms DropTail
$ns duplex-link $n(9) $n(10) 10Mb 10ms DropTail
$ns duplex-link $n(24) $n(25) 10Mb 10ms DropTail
$ns duplex-link $n(24) $n(26) 10Mb 10ms DropTail

# Monitor the queue for link (n12-n18). (for NAM)
$ns duplex-link-op $n(12) $n(18) queuePos 0.5

# Define different colors for data flows (for NAM)
$ns color 1 Blue
$ns color 2 Red

# Setup a UDP connection and attach it to node n(0)
set udp0 [new Agent/UDP]
$ns attach-agent $n(0) $udp0
$udp0 set class 1

# Create a CBR traffic source and attach it to udp0
set cbr0 [new Application/Traffic/CRB]
$cbr0 set packetSize 500
$cbr0 set rate 0.25Mb
$cbr0 attach-agent $udp0

set null0 [new Agent/Null]
$ns attach-agent $n(25) $null0

$ns connect $udp0 $null0
# Setup a UDP connection and attach it to node n(19)
set udp1 [new Agent/UDP]
$ns attach-agent $n(19) $udp1
$udp1 set class_ 2

# Create a CBR traffic source and attach it to udp1
set cbr1 [new Application/Traffic/CBR]
$cbr1 set packetSize_ 500
$cbr1 set rate_ 5Mb
$cbr1 attach-agent $udp1

set null1 [new Agent/Null]
$ns attach-agent $n(24) $null1

$ns connect $udp1 $null1

# Schedule events for the UDP agents
$ns at 0.3 "$cbr0 start"
$ns at 2.5 "$cbr1 start"
$ns at 4.3 "$cbr0 stop"
$ns at 4.5 "$cbr1 stop"
$ns at 5.0 "finish"

$ns rtmodel-at 1.0 down $n(7) $n(23)
$ns rtmodel-at 2.5 up $n(7) $n(23)

# Run the simulation
$ns run
Appendix C

```c
hdr_cmnp *th = hdr_cmnp::access(p);
hdr_ip *iph = hdr_ip::access(p);

if(seed == 0)
{
    srand(time(NULL));
    seed++;
}
else seed++;

if(type_ == 'r')
{
    if(iph->mrknode() == -1)
    {
        rnd_num = rand()%25+1;
        if((th->uid()%rnd_num) + 1 == (iph->mrkttl() - iph->ttl()))
        {
            iph->mrknode_ = dst_;
            iph->mrkttl_ = 0;
        }
    }
    else
    {
        iph->mrkttl_ = iph->mrkttl_ + 1;
    }
}
```