DETECTION AND FILTERING OF SPOOFED IP PACKETS

M.S. Project submitted by

Mamatha Narri

U00347863

Computer and Information Sciences

SUNY Polytechnic Institute

Supervised by

Dr. Jorge Novillo

May 2024

Master of Science in Computer and Information Sciences

Department of Computer Science

State University of New York Polytechnic Institute

Detection and filtering of spoofed IP packets, a project by Mamatha Narri (U00347863) , is approved and recommended for acceptance as a final project in partial fulfillment of the requirements for the degree of Master of Science in Computer and Information Sciences, SUNY Polytechnic Institute.

Date: 05/03/2024

Dr. Jorge Novillo
Professor, Computer Science

Dr. Saumendra Sengupta
Professor, Computer Science

Dr. Scott Spetka
Professor, Computer Science

STUDENT DECLARATION

I declare that this project is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Mamatha Narri

UOO347863

ABSTRACT

IP spoofing is defined as manipulating or forging the source IP address of a packet. There is no authentication on the source IP address, using this, the attackers can spoof the source IP address and use this to perform several attacks such as denial of service(DOS), distributed denial of service(DDOS), etc. Some methods help in detecting and filtering spoofed IP packets. Some of them are applied at the end host and others are applied at intermediate routers.

This project is based on two of the detection and filtering methods of spoofed IP packets which are Hop Count Filtering(HCF) and Ingress Filtering. HCF is based on the idea that one can change or manipulate the source IP address easily but cannot know the number of intermediate devices i.e., hops between two devices. Even if an attacker manipulates the source IP address it is difficult to know the number of hops between the spoofed source IP address and the receiving destination device, hence the hop count may be different from the attacker and destination device, based on this observation hop count filtering was proposed. Using correct mappings between IP address and their hop count to a host, the host will be able to detect the spoofed IP packets. Based on this hop count filter is applied to the incoming packets. The hop count or number of hops can be inferred from the Time-to-Live (TTL) field value in the IP header. Whenever a packet is received one can know the final TTL value. Based on the final ttl value and initial ttl one can obtain the Hop count. Every end host maintains the IP to the Hop count table. The hop count calculated on receiving the packet is compared against the hop count value stored in the IP to HOP count table. If both are equal then it is legitimate otherwise it is a spoofed IP packet. Ingress filtering validates the source IP address of an incoming packet to a particular network using some filtering rules i.e, for example in the case of ISP and customer relationship only allowing the Source IP address that is provided by ISP but this is not the only way to implement ingress filtering. In this work, we will implement both of the above-mentioned methods i.e., Hop count filtering and Ingress filtering, and know their advantages and disadvantages, and work on the disadvantages of Hop count filtering.

Table of Contents

List of Figures

List of Tables

Chapter 1. INTRODUCTION

This chapter gives a brief introduction to IP protocol, problems in Internet protocol(IP), attacks that occur with the problem of Internet protocol, and methods to defend against the problem in IP.

Internet Protocol(IP) is the central part of the TCP/IP protocol stack [1]. It is used at the network layer. The network layer mainly deals with the transfer of data between devices or hosts that are not directly connected. IP is the process by which data is passed between TCP/IP networks with help from other protocols. Layer 3 protocols provide service to layer 4 protocols in TCP/IP networks. Since IP is a layer 3 protocol it takes the data that has been sent by upper layers either TCP or UDP which are layer 4 protocols, adds an IP header, and sends it out. To be precise IP sends data from one host to another host over an internetwork of connected networks(see figure 1.1).

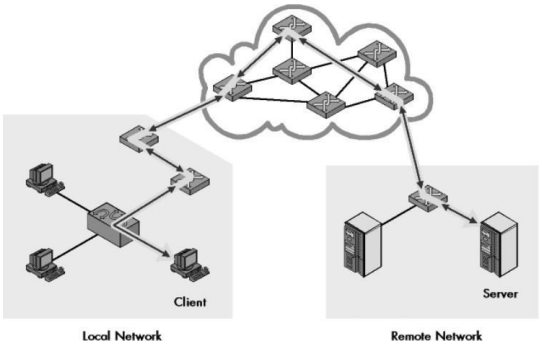

Figure 1.1: Routing between two hosts.

1.1 Main functions of IP

1.1.1 Addressing

This section describes some of the main functions of IP which are Data encapsulation, Indirect delivery, and IP Addressing which are defined in [2]. To deliver the data IP must know where to deliver. For this, every device that is part of the TCP/IP network is assigned a unique identifier. IP uses this identifier known as IP address to deliver the data. An IP address provides unique identification to the interface between a device and a network.

1.1.2 Data Encapsulation

Since IP protocol is layer 3 protocol of TCP/IP it accepts data from the transport layer protocols such as TCP, and UDP. It then places or hides the above-layer data into an IP datagram using a special format before sending the data. It is known as IP header [2]. The main fields of the IP header are: Source address field is the address of the device that sends the datagram. The destination address field is the address of the destination device that one wants to communicate with. A datagram is nothing but an IP header and actual data that one host has sent to another host

1.1.3 Indirect Delivery

To send an IP datagram to a destination that is on the same network, it is done by using direct delivery but in many cases, the final destination is not on the same network, in this scenario the datagram is delivered indirectly. This is done by sending the datagram through intermediate devices known as routers. IP does this with the help of other protocols such as Internet Control Message Protocol(ICMP), routing protocols such as the Open Shortest Path First(OSPF) and Border Gateway Protocol(BGP).

1.2 IP Addressing

The main duty of IP is to deliver data between devices, it can't do this job without knowing where the destinations are present. IP addressing is used not only for host identification but also for the purpose routing of IP datagrams over Internetworks.

1.2.1 IP Address Size

Currently, two well-known versions of IP exist. Internet Protocol version 4 (IPV4)

[3] and Internet Protocol Versio n 6(IPV6) [3]. An IP address is just a 32-bit(IPV4) or 128-bit(IPV6) binary number. since the computer stores the data in 1's and 0's. 32-bit number is divided into 4 octets and 128-bit is divided into groups of 16 bits to make it more manageable. Starting from the leftmost of the IP address each of the eight bits are converted to a decimal number and are separated by a dot(IPV4) for more readability. Since IP address uniquely identifies devices there should be centralized management to make sure that no two devices in a public network have the same IP address. This work of assigning the IP address is done by an organization called Internet Corporation for Assigned Names and Numbers(ICANN).

1.2.2 IP Address Basic Structure

IP addresses are also used to help in the transfer of data from one host to another that are not directly connected. An IP address is structured in such a way that it helps in the transfer of data between hosts and the way they are understood by the routers. Routers are intermediate devices that help in connecting networks. Each IP address is structured in a way that it contains two parts internally (IPV4).

Network Identifier

An identifier is needed to identify a particular network. This can be done by using the IP address. A definite number of bits that start from the leftmost in the binary notation of an IP address is used as an identifier. It is known as network ID [4, 5]

Host Identifier

The remaining portion of the bits that are left after removing network ID bits are known as Host ID [4, 5] bits and are used to identify the device on a particular network.
Network ID helps in the routing of IP datagrams when an IP address is given. Using Network ID routers initially determines whether the destination and host are present on the same local network. If they are not present on the same network then the help of information about networks helps in choosing the path that the packet should take to reach the destination. Host IP helps uniquely identify the interface or device on the network.
The division point that identifies Network ID and Host ID depends on the type of address used. Routers and hosts must know the division point to interpret the IP address.

1.2.3 IP Addressing Categories

Since 32 or 128 bits of IP addresses are divided into network ID and Host ID. The division point is not defined earlier. It is based on the type of addressing used in the network. There are mainly three IP addressing methods: classful, subnetting, and classless.

Classful Addressing:(used in IPV4)

The division point in classful addressing [6] type occurs at the boundaries of the octet. There are three main classes of address, B, and C. In Class A addressing network ID takes left most eight bits and host ID uses the remaining bits which means each network can contain (2 power 24) hosts and left most bits is always 0 which means the network ID range is (0. z.z.z - 127. z.z.z) here z represents host ID bits. Network ID is represented by placing all 0s in the host ID's place. In Class B addressing network ID takes the B addressing network ID takes the leftmost sixteen bits and the rest of the bits are used for host ID and the leftmost two bits are always '10' which means the network ID range is (128. (0-255).z.z-191.(0-255).z.z) here z represents host ID bits. Class C addresses have the leftmost twenty-four bits as network ID and the rest of the eight bits as host ID and the leftmost 3 bits are always '110' which means the network ID range is (192. (0-255). (0-255).z - 223. (0-255). (0-255).z) here z represents host ID bits. So by using this addressing scheme routers can determine network ID and host ID. This scheme is not in use today.

Subnetted Classful Addressing:(used in IPV4)

Subnetted Classful Addressing [7] is mainly used to help in the usage of IPV4 addresses. To divide into networks having a smaller number of hosts this scheme takes some of the host ID bits(Class A, B, or C) and uses them as network ID bits known as Subnet Identifier(subnet ID) and it is also used to reach a particular subnetwork. For example, consider a class B address that uses the leftmost 16 bits for the network ID and the rest of the 16 bits for the host ID. By extracting four bits from the host ID we can further build sixteen subnetworks and each network contains (2 power 12) hosts.

Classless Addressing(classless inter-domain routing):(used in IPV4 and IPV6)

In the classless addressing [8] scheme the division point between network ID and host ID can occur at any arbitrary point not just after eight bits as seen in the classful scheme. The division point is known by putting the number of bits used for network ID, called network prefix after the IP address. For example if given IP address 220.12.12.0/17 in the case of IPV4 that means 17 bits are used for the network ID and the remaining 15 bits are used for the Host ID.

Subnet Mask

In the classful method, the division between Network ID and Host ID is known implicitly whereas in subnetting or classless addressing subnet mask [7] is used to know the number of bits in Network ID and Host ID. A subnet mask is represented by having all 1's in the Network ID and 0's in the Host ID part.

1.3 IP Routing

When the destination device is not in the same network then the delivery of the datagram will go through one or more intermediate devices before finally reaching the destination device. The devices that help in the delivery of datagrams are known as routers. Routers help in connecting networks. Routers are physically connected for routing [7]. The datagram will go from one router to another until it reaches the physical network of the destination device. When sending a datagram outside the network we can only send it to the router we are attached to, the router sees the destination IP address decides what the next hop of the datagram will be, and forwards it. This process is continued until the packet reaches the destination network. This is known as routing [7]. The next hop is nothing but the next device that the datagram should be forwarded. To know the next device to which it has to send it must first know the paths to networks. This can be done by using routing protocols in which each router exchanges information about networks. Each router sends information stating that this destination network can be reached through me. In this way each router exchanges information about routes. The information is maintained in a table known as a Routing table (see Figure 2.1).

1.3.1 Routing Table

The routing table consists of networks that are reachable from it and the next interface it should take to reach a particular network. To get all the other network reachability information routers exchange this information i.e., networks that are reachable from them with the use of routing protocols like OSPF, BGP, etc. After receiving the datagram routers examine the IP address of the destination from the IP header and will decide the next hop that the datagram should take. To do this each router maintains information that gives the mapping between the network IDs and the next hop it should take to reach a destination. This information is maintained in a table called the routing table(see Figure 1.2). Each entry in the routing table gives information about one network. To be precise it says ”If the destination IP address belongs to one of the following networks in the routing table then the next hop should be the following router device”. Each entry also maintains the subnet mask which identifies the network ID bits and host ID bits.

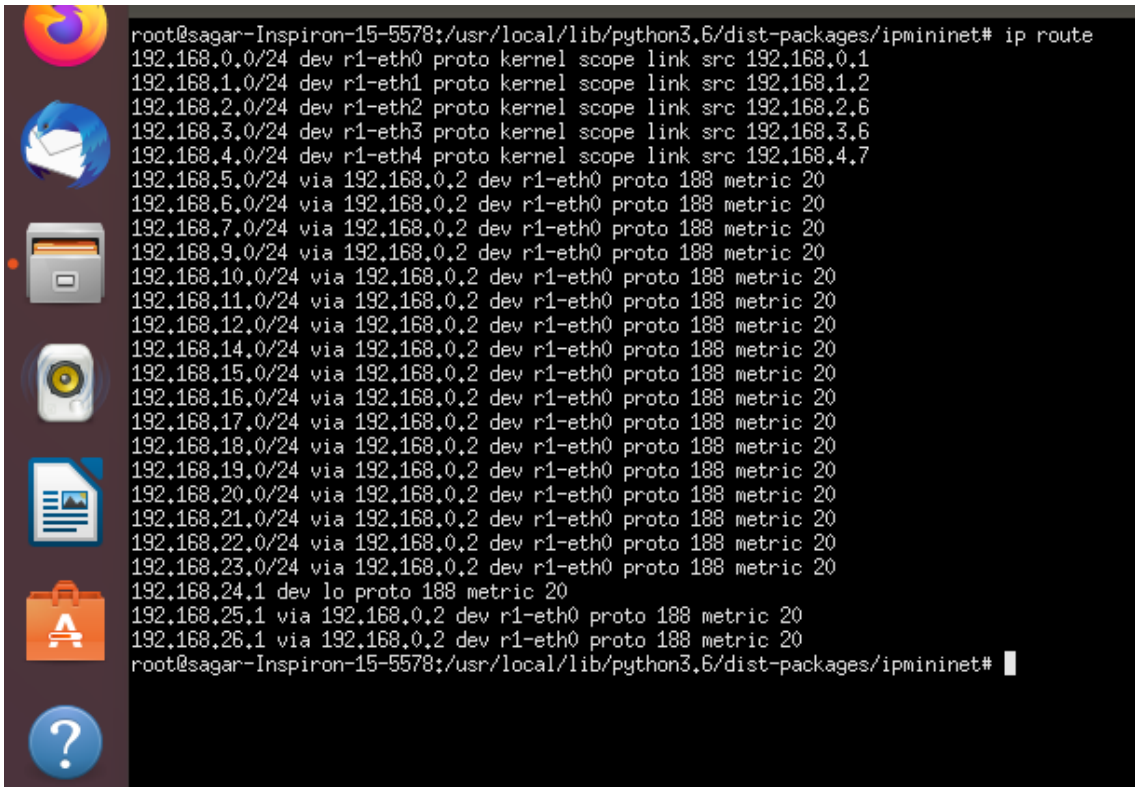

Figure 1.2: Routing Table

To find out the matching network in the routing table the process is firstly the destination IP address is converted into a binary number. A subnet mask of network ID is also converted into a binary number. Operation AND is performed between both of them which gives us the network ID of the destination IP address, if the result matches one of the network IDs in the routing table then the datagram is passed to the corresponding router interface. In the case of CIDR if there is more than one match then the longest prefix match will be selected. Consider for example in the case of IPV4 routing a datagram reaches the router with the destination IP address 180.12.13.14, the router process goes through the routing table to check whether the host belongs to any of the listed network IDs. The routing table consists of network IDs 180.12.13.0/24 and 180.12.0.0/16, AND operation is done between the destination IP address and two subnet masks of the network ID's. In this, both of the results match with the corresponding network IDs, as 180.12.13.0/24 is the longest prefix match so the next hop will be the corresponding interface of the network ID.

1.4 Problem in Internet Protocol

Nowadays everyone has access to the internet. Many services like banking, e-learning, entertainment, etc are based on the Internet service. Therefore any disturbance to internet service is considered as a problem and results in monetary losses to organisations. Unfortunately, the internet is designed based on scalability rather than security. However, there are some design weaknesses in protocols used today that help attackers or hackers to cause disturbance to the internet service.

One such problem in the protocols used on the internet is present in the IP Protocol. The IP header consists of a field called Source IP address which is the address of the host that sends it. Since the routing is based on the destination IP address there is no verification that the given source address has sent the data. Using this weakness attackers can spoof the source IP address and use this to perform several attacks such as denial of service(DOS), distributed denial of service(DDOS), hijacking sessions, etc.

By design, there is no authentication for the source IP address [10] which means there is no guarantee that a host with a source IP address has sent it. Therefore, one can send an IP packet with the wrong IP address rather than using the IP address provided to him by the Internet Service Provider. An attacker can thus use this weakness to remain unknown and launch targeted attacks. Many attacks utilize IP spoofing.

1.5 Some of the Attacks that use IP spoofing

1.5.1 TCP SYN Flood Attack

TCP SYN Flood attack [11] is a denial of service attack that affects the hosts that run the TCP server process. The basic idea is before communicating with the TCP server process one has to go through a way handshake procedure to start communicating. The first step is the client or host who wants to communicate with the TCP server sends a TCP SYN packet to which the server in turn replies with TCP

SYN+ACK packet and waits for the last packet from the client which is the ACK packet to complete the three-way handshake procedure.

Transmission Control Block(TCB) [12] is a data structure that stores all the state information for a connection. After sending the TCP SYN+ACK packet some of the TCB block is initialized which takes some amount of host or server memory. To protect server or host memory from exhaustion by TCP connection requests the number of TCB structures that can be resident at any time is limited by Operating Systems known as backlog. When the limit is reached either the new incoming SYN requests are ignored or incomplete established connections are replaced. Using this what the attacker does is send a lot of connection requests and do not reply to the SYN+ACK packet and the server opens a lot of half-open connections and finally exhausts the backlog. Now when a legitimate host requests a connection client request is turned down thereby causing a denial of service to the client.

To not send the final ACK packet the attacker spoofs the source IP address of the SYN packet and sends it to the server process. The server process sends the reply to the IP address which is spoofed and waits for the reply, the packet doesn't reach the attacker. The spoofed IP address may or may not be active on the internet. If the spoofed IP address is not active the reply doesn't come to the server process. If the spoofed IP address is active on the internet the spoofed IP address replies to the server process with a TCP RST(RESET) packet. After receiving the RST packet the server process releases the TCB and makes room for a new connection.

1.5.2 UDP flood attack

User Datagram Protocol (UDP) is used at layer 4 of the TCP/IP stack which is the transport layer. UDP doesn't require a connection beforehand to send and receive the data from the other server or host that runs a UDP process. So based on this observation, an attacker can send an UDP packet with any amount of data without requiring the connection.

Whenever a UDP packet is sent to a host or server to a particular port number, the server or host checks whether a process is running with that port number if there is

no process with that port number then an Internet Control Message Protocol(ICMP) packet is sent as a reply to the sender that the destination port number is unreachable.
UDP attack [13] tries to overwhelm the server resources and the ability to respond and process. Whenever a UDP packet is the above steps are all followed. The Attacker floods the server or host with a flood of UDP packets which will eventually exhaust the server resources.
An attacker may spoof the source IP address for the following reasons either to hide his own identity or to not get replies from those UDP packets which may saturate his resources.

1.5.3 Memcached attack

Memcached [15] is a memory object caching system used for speeding up web applications by reducing the database load. Generally, memcached servers are not exposed publicly but some of them are available on the public internet. Memcached servers mostly use UDP with port number 112111. Since there is no connection required beforehand for communicating with the UDP Process one can easily spoof the IP address and send a request to one of the memcached servers and the replies are directed toward the spoofed IP address. In this an attacker sends a request to the memcached server with the spoofed IP address, the server then responds to the request with a large amount of data which may overload the spoofed IP address resources. A spoofed IP address host may be unable to process a large amount of data which results in denial of service to legitimate requests. Recently in 2018 a memcached attack
[14] was targeted towards Github [26], an online code management service used by millions of developers. This attack reached 1.3 Tbps, sending packets at a rate of 126.9 million per second. It is considered one of the largest DDOS attacks.

1.6 Methods to defend against IP spoofing

Two well-known methods help in detecting and filtering spoofed IP Unicast packets. Unicast is the process of sending data from one host to another host.

Hop count Filtering [16]: It is based on the idea that since source address manipulation is easy and can do it but the number of intermediate devices it goes through to reach a particular destination i.e, hop count is not in the hands of anyone so if we can store hop count and validate it against incoming packets then one can tell whether the packet is spoofed or legitimate.

Ingress Filtering [17]: This method will filter all incoming packets to a network based on some rules.

1.7 Current State of IP Spoofing

Even though after knowing the weakness in IP protocol for many years and having filtering techniques the attacks that use IP spoofing are still present and frequently reported [18]. According to the spoofer project of CAIDA [18], 25.7% of the IPV4 and 22.7% of the IPV6 Autonomous systems(ASes) did not deploy any countermeasure to filter spoofed IP traffic and 22% of IPV4 blocks, 12.8% of IPV6 blocks in the Internet can be spoofed. So IP spoofing is still a problem to be solved.

1.8 Objective

The objective of this thesis is to work on methods of detection and filtering of Spoofed Internet Protocol(IP) packets and know the advantages and disadvantages of these methods, work on disadvantages of Hop Count Filtering.

1.9 Thesis Organization

This work contains five chapters. In Chapter 1 we discuss Internet Protocol (IP), Internet Protocol (IP) Spoofing, Attacks that use Internet Protocol Spoofing (IP), and the current state of IP spoofing. Chapter 2 describes the literature survey of some of the methods that detect and filter spoofed Internet Protocol(IP) packets. Chapter 3 describes the Implementation of two of the detection and filtering methods of spoofed Internet Protocol(IP) packets and their advantages, and disadvantages, the Proposed algorithm for the extension one of the Hop Count Filtering method, chapter 4 describes the results and analysis and finally, chapter 5 describes conclusion and future work.

CHAPTER 2. LITERATURE SURVEY

This chapter gives a literature survey of some of the methods that help in detecting and filtering spoofed IP packets like Ingress Filtering and Hop Count Filtering.

2.1 Hop Count Filtering

This method is mainly used to filter out spoofed DDOS packets. It is mainly used in services that use TCP as a Transport Layer protocol.

2.1.1 Hop-Count Filtering: Novel Filtering method-2007

Hop count Filtering [16] is based on the idea that changing or manipulating an IP address is easy but one cannot easily modify the number of hops between sender and receiver. Hops are nothing but several routers between sender and receiver. Based on this observation a Hop count filtering method is proposed. The idea is that many of the IP packets that are spoofed will not carry correct hop count values when arrive at the destination. So, if there are correct hop count mappings between one host and another host one can identify whether the packet is spoofed or not. The hop count filtering method maintains an IP address to the Hop count table to verify the hop count of an incoming packet.

Hop Count Table Construction:

IP to Hop-Count table is calculated using traceroute data from source to destination. Hop count is calculated to only those IP address that completes the TCP three-way Handshake. IP addresses are recorded for 1-2 weeks depending on the customers that the service has. Then traceroute is applied to each of the IP addresses to get the hop count value.

Calculation of Hop Count:

We cannot know the Hop count value directly but it can be derived from one of the headers of the IP header which is Time To Live(TTL) [2]. The Time To Live field is used to specify the maximum lifetime of IP packets that can stay on the internet. During transit, each router decreases the time to live value by 1 before sending it to the next hop router. Therefore the number of hops between a sender and receiver can be found out using final ttl value and initial ttl value value. If we know both the values we can get the hop count by subtracting the final ttl value from the initial ttl value. Hop count = (initial time to live value - final time to live value). But the problem is we don't have the information about the initial TTL value it would be easy if all the Operating Systems used the same initial time to live value. But most Operating Systems use 30,32,60,64,128 and 255 [19] as an initial time to live value. There are only a few hosts that have a hop count of more than 30 between them, so by this, we can know the initial TTL value by placing the final TTL value in the list of the above initial TTL values, and sorting is done on that list. The next number which is present after the final ttl value in the sorted list is considered as the initial ttl value. For example, if the final ttl value is 102, the next number to 102 after sorting the initial ttl values list is 128. Hence 128 is the initial TTL value. In the case of 30, 32, and 60, 64 hop count value is calculated using both of the (60 and 64) or (30 and 32) as initial TTL values, and the packet is accepted if one of the hop count values matches.

Inspection Algorithm:

This algorithm starts after building the IP to hop count table. The source IP address and final ttl value are extracted from each IP packet, initial ttl value is found using hop count calculation. The hop count value is obtained by knowing the difference between the final ttl value and the initial ttl value. Using the source address the hop count value is extracted from the IP to the hop count table. If the calculated hop count doesn't match the hop count value in the IP to hop count table then the packet is considered as spoofed (see algorithm 1).

Algorithm 1 Inspection Algorithm

1: for every packet:

2: get the TTL value of the incoming packet which is Tx and IP address P;

3: get initial TTL value T1;

4: calculate hop count H = T1-Tx;

5: get the hop count using index P which is H1;

6: **if** $H == H1$ **then**

7: packet is not spoofed;

8: **else**

9: packet is spoofed;

10: **end if**

States of Hop Count Filtering:

HCF runs in two states: the alert state for detecting packets with spoofed IP addresses and the action state which discards spoofed IP packets. Normally HCF stays in an alert state and only examines the IP packets but will not filter the packets with spoofed IP addresses. Upon detection of a large amount of spoofed IP packets HCF changes into another state which is an action state that examines each packet and discards the packets with spoofed IP address.

This method does not require network support since it can be implemented at the end system. This method is not effective at stopping the bandwidth attacks since the filter is applied only at the end host, the whole network is flooded in case of bandwidth attacks and hence cannot save a network from bandwidth congestion. If an attacker has the same hop count as a spoofed IP address then this method fails. This only works on the client IP address that is already stored in the IP 2 hop count table. Traceroute results from source to destination might be different from destination to source so there is a chance that the hop count value stored might be wrong. Since there might be multiple paths between source and destination so there might be different hop counts between them.

2.1.2 Implemented Hop Count Filtering Method - 2008

These authors implemented the HCF module inside the Linux kernel and they have shown that it can be implemented efficiently using a Hash table to store the IP addresses. [20]

2.1.3 Distributed Hop Count Filtering - 2009

This method [21] tries to solve one of the drawbacks of Hop count Filtering which is it is implemented only at the end host, in case of bandwidth attacks the total network will be flooded by spoofed packets so not to get network flooded this method implements it in the intermediate routers which will help in filtering out packets at the very beginning.

2.1.4 Modified hop count filtering method -2014

Algorithm 2 Modified Inspection Algorithm 1

1: for each packet:

2: get the TTL value of the incoming packet which is Tx and IP address P;

3: get initial TTL value Ty;

4: calculate hop count H = Ty-Tx;

5: get stored hop count list H1 using IP address P as index;

6: **if** *H in H*1 **then**

7: packet is not spoofed;

8: **else**

9: packet is spoofed;

10: **end if**

The improvement that authors made in this work [22] is to consider the possibility of multiple paths between source and destination that there might be changes in hop count values so instead of storing one hop count value they made a list of hop count values for a particular IP address. In this way, all the hop count values can be known if there are multiple paths with varying hop counts (see Algorithm 2).

Storing multiple hop counts for an IP address may also lead to giving attackers many options of hop count values to pass the filter. Since there is more than one value stored there might chance that an attacker can have any one of those hop count values.

2.1.5 A Victim-Based Statistical Filtering - 2017

Algorithm 3 Modified Inspection Algorithm 2

1: Let the Port frequency table be X

2: for every packet:

3: get final ttl value Tx, port number P, and IP address IP;

4: get initial TTL value Ty;

5: calculate hop count Ht = Ty-Tx;

6: the hop count using index IP which is H1;

7: **if** *Ht is not equal to H*1 **then**

8: the packet is spoofed; 9: **else if** *P in X* **then** 10: the packet is spoofed; 11: **else**

12: packet is not spoofed;

13: **end if**

This method [23] is used in the case of TCP-SYN flood attacks [11]. The idea is before an attacker sends a TCP SYN request he has to know the port number of a TCP process to know that attackers use port scanner tools to know the port numbers of a TCP process. Port scanner tools request the host for open port numbers and the host replies with the open port numbers. Finally, the attackers keep on sending SYN requests to these port numbers. So this method observes the usage level of port numbers. A port numbers frequency table is maintained at the host which contains the port numbers with the maximum frequency. A table is maintained which contains the port activity levels which give the frequency except for well-known port numbers. Based on this a small module is added to the Hop count filtering method.

Only the Hop count is validated in hop count filtering even if hop count values are equal this method adds another validation which is every incoming packet's port number is validated against the port frequency table if the incoming packet port number is present in the table then it is termed as spoofed (see Algorithm 3).

The time quantum or observation period for the port frequency analyzed is not clear. They have used a time quantum of 60 seconds.

2.1.6 Enhancement of Hop Count Filtering -2017

Algorithm 4 Modified Inspection Algorithm 3

1: for every packet:

2: get final ttl value Tx, port number P, and IP address IP;

3: get initial TTL value Ty;

4: calculate hop count H = Ty-Tx;

5: get the hop count using index IP which is H1 6:
get the Port number using index IP which is Y 7: **if** *H* == *H*1 *and P* == *Y* **then**

8: packet is not spoofed;

9: **else**

10: packet is spoofed;

11: **end if**

This work [24] tries to solve one of the drawbacks of the Hop count Filtering method which is if both the attacker and legitimate IP address have the same Hop Count then the Hop count Filtering method doesn't work. To overcome this, the author proposes an idea which is to add or store the port number of service that a particular client requested. So during the process of TCP 3-way handshake, the port number for which the client requested is stored with the Hop count. So again if the client's IP address visits or requests something then the port number is validated against the already stored port number. If both the hop count and port number are equal to already stored values then the packet is termed as not spoofed other it is termed as spoofed (see Algorithm 4).

2.2 Ingress Filtering

Since IP routing is based on destination IP address there is no authentication of source IP address. If we can apply some filtering techniques or rules on packets entering into the network then we can limit the usage of spoofed IP addresses. Below are some of the filtering rules at the incoming(ingress) interface of a network. The below methods are applied to Unicast Packets.

2.2.1 Ingress Filtering-May 2000

Ingress filtering [17] is nothing but the filtering of packets at the incoming interface of a network. Filtering rules are applied at the incoming interface of a network. The following is the rule that is made at the incoming interface of a network.

Algorithm 5 Ingress filtering

if the *packet source address is from within the range of the network* **then**

forward it as per the forwarding information base(FIB)

else

discard the packet.

end if

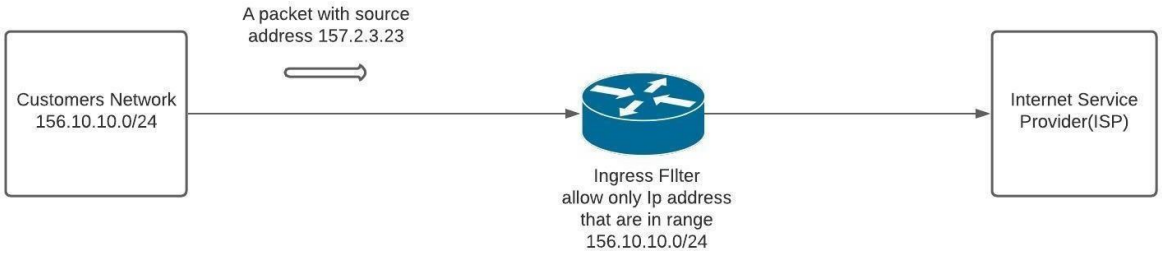

Figure 2.1: Ingress Filtering

for example in the above topology(see figure 2.1), if the attacker resides within 156.10.10.0/24 which is given by the Internet Service Provider. A filter is used on the input interface of the router that provides a connection to the customer network. The filter's rule is if any of the packets arrive with a source address that is not within the range of 156.10.10.0/24 then the packet is discarded other wise packet is forwarded (see algorithm 5). In this way, the filter helps in stopping the packets which have source addresses that are outside the network range.

An attacker or intruder can still spoof the IP address by having a source address that is in the network range. The attacker residing in the same network can still spoof.

2.2.2 Different Ways of Implementing Ingress Filter - 2004

This paper [23] discusses different ways of implementing the Ingress filter. Ingress filter rules are applied at the border routers, peering links, and last mile i.e., ISP to the customer network.

Ingress Access Lists:

An ingress access list is a filter that is applied on the incoming interface of a network which checks the incoming packet's source IP address with a list of acceptable prefixes that are allowed in the network. If any of the source addresses of the packet don't appear in the list that packet is discarded (see algorithm 6).

Algorithm 6 Ingress Access List

for every incoming packet:

if the *source address is present in the access list* **then**

Forward it as per the forwarding information base (FIB)

else

Drop the packet

end if

These lists are maintained manually, so forgetting to update the list if prefixes change might cause the ingress filter to discard legal packets.

Strict Unicast Reverse Path Forwarding:

The process is that whenever a packet is received on the incoming interface of the network it will be checked in the Forwarding Information Base(FIB) table and if the interface on which the packet is received is the same as an interface which it will use to forward the packet when it is received as a destination IP address then it will be forwarded otherwise it is discarded(see algorithm 7).

Algorithm 7 Strict Unicast Reverse Path Forwarding for every incoming packet:

if the *source address is received on the interface which it will use to forward when that IP address appears as a destination* **then**

Forward it as per the forwarding information base (FIB)

else

Drop the packet

end if

This works fine in the network administered by a single organization i.e., an AS since the best path is always chosen to forward a packet. This works only in places where routing is symmetrical, where IP packets outgoing from the network and incoming

Network ID[AS-Path] notation in the above figure is known as BGP route update with network ID as an identifier of the network and [AS-1] is known as AS PATH which says it should pass through these ASes to reach that particular network.

From the other follow the same path. It doesn't work in cases where multihoming is used. Multihoming is the practice of using more than one transit provider. It uses the Border Gateway Protocol(BGP) as a routing protocol to exchange routes since BGP allows ISPs to have policies. BGP policies allow customers to announce some prefixes to one transit provider and some other prefixes to another transit provider. The downfall of the multihoming can lead to the cause of asymmetric routing. For example (see figure 2.2) when a customer AS announces one prefix 150.10.10.0/24 to one Internet service provider(ISP-w) and another prefix 169.12.10.0/24 to another Internet service provider(ISP-x). ISP-w and ISP-x exchange routes that are learned from the customer. Let's consider the border router in ISP-w has interfaces eth0 that connects the customer network and eth1 that connects peer ISP-x. Similarly, ISP-x has borders routers with interfaces eth2 that connect peer ISP-w and eth3 interface that connect customer network.

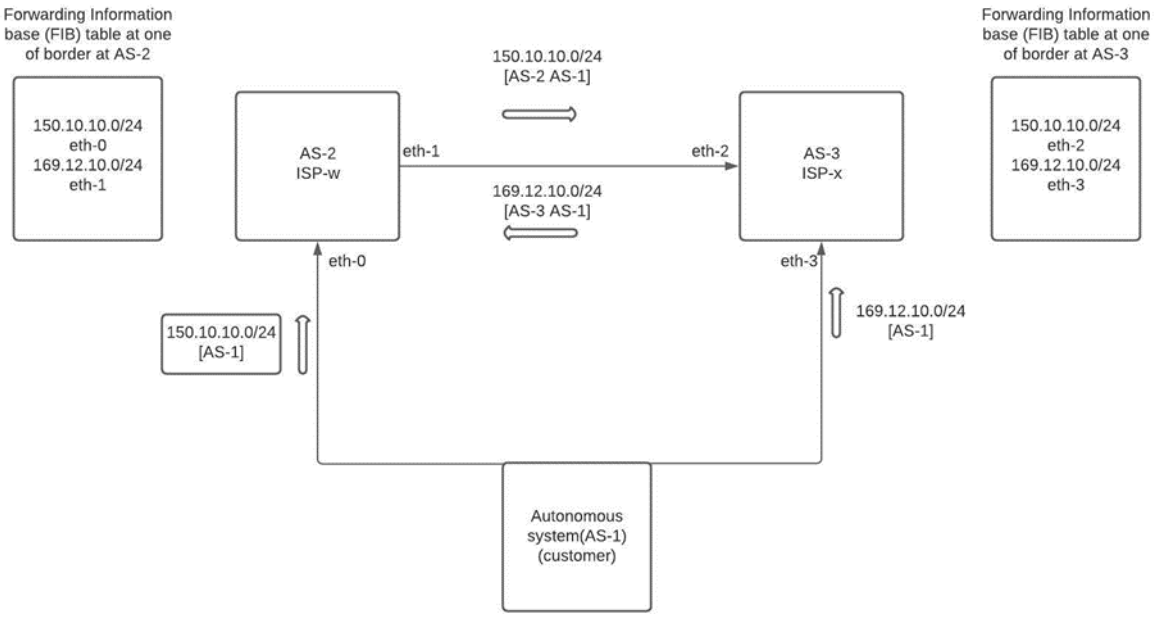

Figure 2.2: Strict Unicast Reverse Path Forwarding

After exchanging routes using BGP the forwarding information base (FIB) table at ISP-w contains 150.10.10.0/24 and uses eth0 for delivering packets with des- a destination address in the range of the above network, for destination IP address within range 169.12.10.0/24 uses eth1 interface to forward. The forwarding information Base (FIB) table at ISP-x contains for destination IP address within the range of 169.12.10.0/24 uses interface eth3 to forward, and for destination IP address within the range of 150.10.10.0/24 uses a n eth2 interface to forward.

Consider if a strict unicast reverse path forwarding filter is applied at ISP-w. Now if the customer routes one of the IP addresses in the range of 169.12.10.0/24 via ISP-w then it will be dropped at ISP-w since when an IP packet with a destination address within the range of 169.12.10.0/24 arrives at the border router of ISP-w it uses interface eth1 to forward but now the packet is arrived at interface eth0 hence it is discarded. Further, if a packet with an IP address in the range 150.10.10.0/24 is received from Customer via ISP-x at ISP-w then these packets are dropped at ISP-w since it uses interface eth0 to forward the packets if the destination IP address is within range 150.10.10.0/24 but now the packet is arrived at interface eth1.

Feasible-path

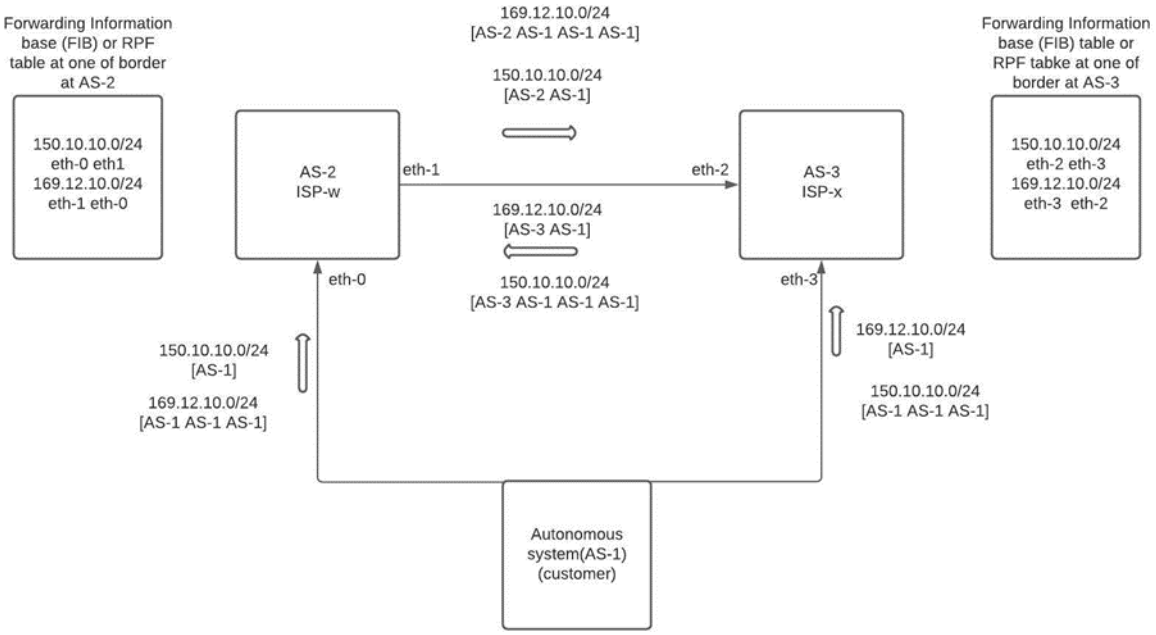

Figure 2.3: Feasible-Path Unicast Reverse Path Forwarding

Unicast Reverse Path Forwarding:

Network ID[AS-Path] notation in the above figure is known as BGP route update with network ID as an identifier of the network and [AS-1] is known as AS PATH which says it should pass through these ASes to reach that particular network.

This method is similar to Strict Unicast Reverse Path Forwarding. When an IP packet is received at the router the source IP address is still looked up in the Forwarding Information Base or RPF table but instead of inserting the best route there all the alternate paths are added and used for validation (see algorithm 8). For example, to reach a particular network there may be different paths and may use different interfaces to reach that particular network from a particular network. So if a packet is received on any of those interfaces then it is forwarded as usual otherwise it is discarded. It is helpful in case of a multihoming scenario where strict reverse path forwarding failed but the following rules should be obeyed:

Announcement of same prefixes should be propagated to all transit providers which applies

the feasible path reverse path forwarding filter.
Announcing more specific prefixes to specific transit providers while announcing less specific prefixes to all transit providers.

Consider for example (see Figure 2.3) a customer AS is connected to two ISP providers. It uses the Border Gateway Protocol(BGP) as a routing protocol to exchange routes since BGP allows ISPs to have policies. BGP policies allow customers to announce some prefixes to one transit provider and some other prefixes to another transit provider.

But in this method either one of the above rules should be followed. In the following example, a customer AS announces the same prefixes 150.10.10.0/24, and 169.12.10.0/24 to both the internet service providers(ISP-x, ISP-w). ISP-w and ISP-x exchange routes that are learned from the customer. Let's consider the border router in ISP-w has interfaces eth0 that connects the customer network and eth1 that connects peer ISP-x. Similarly, ISP-x has borders routers with interfaces eth2 that connect peer ISP-w and eth3 interface that connect customer network.

After exchanging routes using BGP the Reverse Path forwarding table(RPF) table at ISP-w contains network prefixes 150.10.10.0/24, 169.12.10.0/24 with inter-faces eth0 or eth1 that means if a source address in the range 150.10.10.0/24 or 169.12.10.0/24 can come from both of these interfaces. The Reverse Path forwarding(RPF) table at ISP-x contains network prefixes 150.10.10.0/24, 169.12.10.0/24 with interfaces eth2 or eth3 that means if a source address in the range 150.10.10.0/24 or 169.12.10.0/24 can come from both of these interfaces

Algorithm 8 Feasible-path Unicast Reverse Path Forwarding for every incoming packet:

if the *source address is received on the interfaces on which routes to the source IP address are received* **then**

Forward it as per the forwarding information base (FIB)

else

Drop the packet

end if

There is another scenario where the method can fail. It is possible that any of the ISPs from the above (see figure 2.3) may not announce any of one of the routes from the two prefixes to each other in this case if we apply the filter on any of those interfaces and if packets arrive with that prefixes then those packets are discarded. Consider for example ISP-x does not announce the route for the prefix 169.10.12.0/24 due to some policy then if we apply a filter at ISP-x packets arriving with the source address from the network 169.10.12.0/24 those packets are discarded.

Loose Unicast Reverse Path Forwarding:

Loose Reverse Path Forwarding checks for the presence of a route for a particular network even if it is a default route i.e when a packet is received at the incoming interface of a network where a loose reverse path forwarding filter is applied the source IP address is checked in the Forwarding Information Base(FIB) to see whether a route exists for that IP address even if it is a default route (see algorithm 9).

Algorithm 9 Loose Unicast Reverse Path Forwarding for every incoming packet:

if *a route to that source IP address exists even including the default route* **then**

Forward it as per the forwarding information base (FIB)

else

Drop the packet

end if

2.2.3 Enhanced Feasible-path unicast Reverse Path forward- ing - 2020

This method [24] is an improvement to existing feasible reverse path forwarding. It is based on the fact that if an AS(Autonomous System) receives Border Gateway Protocol routes for multiple Prefixes with the same originator AS at multiple interfaces i.e, border routers of an Autonomous system then the incoming packets with source address in any of those prefixes should be accepted on any of those interfaces.

Consider for example(see figure 2.4) the border router of ISP-X has received routes for prefixes l1, l2, and l3 each of them has AS-1 as the origin which is a customer of ISP-X.Border router has received three prefixes l1 from customer link, l2 from peer link and l3 from transit provider of Upstream. Enhance Feasible path reverse path forwarding by adding l1, l2, and l3 in the rpf filter (reverse path forwarding) at the interface where any of the prefixed received.

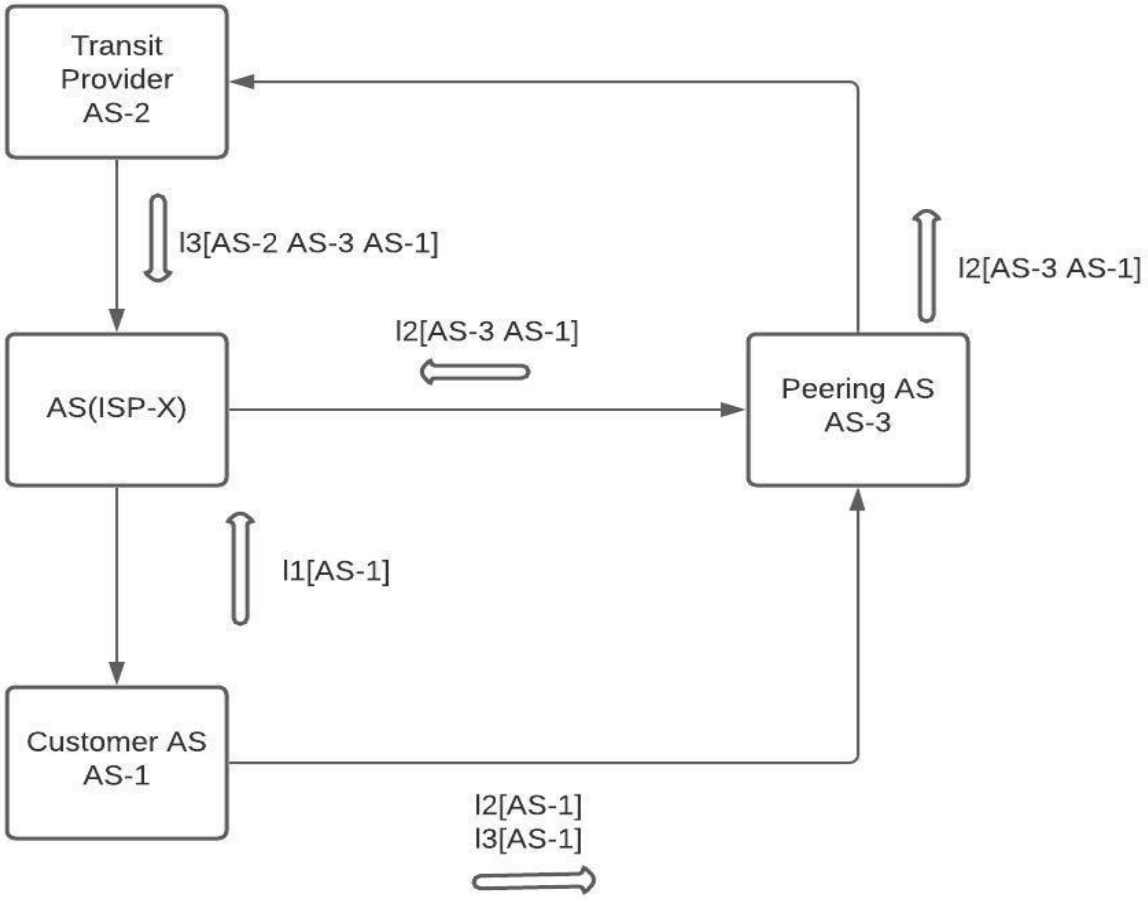

Figure 2.4: A simple topology for illustration of Enhanced Feasible-path unicast Reverse Path Forwarding

Network ID[AS-Path] notation in the above figure is known as BGP route update with network ID as an identifier of the network and [AS-1] is known as AS PATH which says it should pass through these ASes to reach that particular network.

The following algorithm is implemented in transit AS. Transit AS is nothing but a network that provides the transportation to reach the rest of the networks i.e., the Internet.

Algorithm 11 Enhanced feasible-path unicast reverse path forwarding

1: Consider only the routes announced by the customer that are present in the Adjacent routing information base(RIB)-In the table make a list of unique origin ASes. list X = AS1,AS2,...,ASn

2: Now consider all routes in the Adjacent routing information base-in table from all interfaces which are customer, transit, and peer. Make a list of routes that have the same origin i.e. for example AS1. list Y =R1,..Rn

3: Now include the list Y on all the interfaces from which one or more prefixes are received.

4: Repeat the above two steps for all the ASes that are present in list X.

This can be applied to even peer links. It depends on the operator to whether apply Enhanced Feasible Reverse path forwarding or loose reverse path forwarding.

The following rules should be applied to make the method robust:

For stub AS which uses Multihoming:

1. The stub AS should announce at least one of the prefixes it has to each of its transit providers.

For AS that is used for Transit:

1. It should announce at least one of the prefixes it has in its AS to each of its transit providers.
2. It should also announce routes that are learned from its customers i.e., at least one prefix from each origin AS to its transit provider.

CHAPTER 3. IMPLEMENTATION AND METHODOLOGY

We have worked on the two methods Hop count filtering, and ingress filtering which were discussed in Chapter 2.

3.1 Experimental setup

To emulate the network we have used ipmininet [28] which is an extension to Mininet [29] that provides the virtual network environment and also provides some topologies. To do packet manipulation Scapy [30] which is a packet manipulation module written in Python. Implemented both of these methods in python3 on some of the topologies (see Figures 3.1 and 3.2). Fig 3.1 shows a simple OSPF topology, Fig 3.2 shows a simple BGP topology.

3.1.1 Hop Count Filtering

On one of the hosts, the hop count filtering process will be running that uses raw sockets for communication. Raw sockets are used to get raw packets from the client, raw packets are nothing but packets with header and data. Another host will be sending spoofed and unspoofed packets using Scapy to the host that runs the hop count filter process.

Advantages:

1. Does not require any network support. It is implemented at the end host.

Disadvantages:

1. This method only works for IP addresses that are already stored in the ip2Hopcount table, if a new client or host requests for the service then that client is deemed as spoofed and the packet is discarded resulting in a denial of service.
2. According to [25] the hop count between any two hosts observed is more than 30 but the above method takes 30 as the max hop count in calculating the initial hop count. 3. If the attacker and spoofed IP address have the same hop count then this method fails.

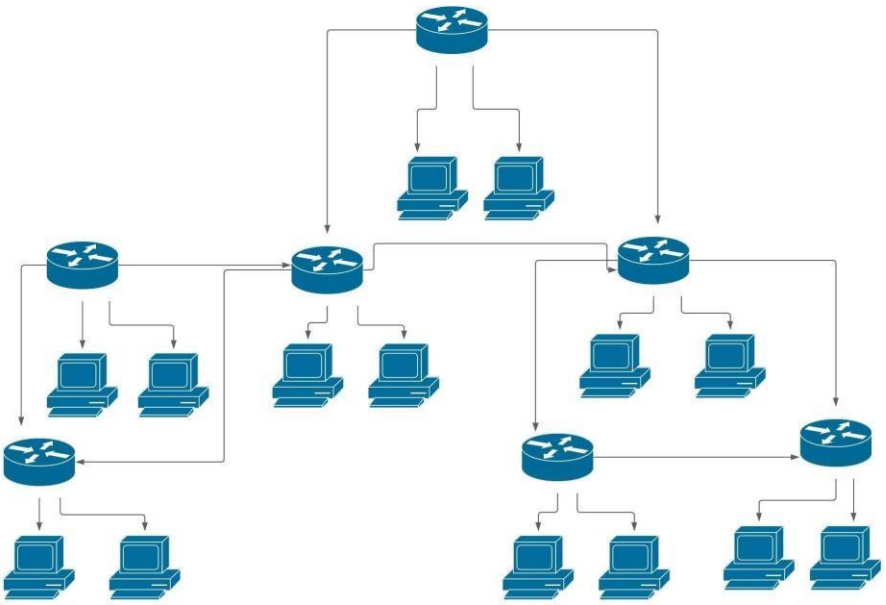

Figure 3.1: Simple OSPF Topology

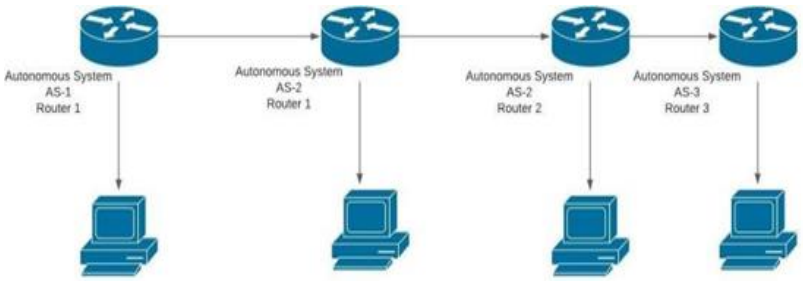

Figure 3.2 Simple BGP Topology

3.1.2 Ingress Filtering

Worked on two ways of implementing Ingress filtering: strict reverse path and loose reverse path forwarding including the default route.
rp filter in Linux is used to implement the above two methods. rp filter takes three values 0, 1, 2. rp filter =0 states that source address verification is off, rp filter=1 source address states that source address verification is on and strict reverse path forwarding method is implemented, rp filter=2 states source address verification is on and loose reverse path forwarding method is implemented.

Advantages:

1. It filters the spoofed packets at the very beginning of the network hence saving the bandwidth of the network.
2. the very beginning of the network hence saving the bandwidth of the network.

Disadvantages:

1. Strict reverse path forwarding suffers from asymmetric routing.
2. Loose path forwarding loses the directionality since it only checks for the presence of route and it is advised to be used on peering links.
3. Ingress filtering should be applied everywhere where i.e. peering links, customer-to-provide links, etc to work effectively.
4. Ingress Filtering cannot filter the spoofed packets if the attacker lies in the same network and spoofs the IP address that belongs to the same network.

3.1.3 Proposed Algorithm:

The hop Count Filtering method only works for IP addresses that are already stored in the ip2 Hop count table, if a new client or host requests for the service then that client is deemed as spoofed and the packet is discarded resulting in a denial of service.

Based on the above disadvantage we have tried a small modification to Hop count Filtering. this modification is based on two scenarios i.e., if the source IP address exists on the internet and if the source IP address is not present on the internet but the network exists. Instead of storing the hop count value before Whenever a packet is received we have saved the final ttl value of a packet and send an Internet Control Message Protocol(ICMP) [27] echo request packet to the source IP address for which we will get an ICMP echo reply packet (in case if IP address host exists in the internet) in which we can extract the final ttl value. Equal final ttl value determines whether they have the same hop count or not then based on the comparison we can determine whether the packet is legitimate or not (see Algorithm 1).

Algorithm 12 Proposed Algorithm for Detecting Spoofed Packets

1: On receiving a packet:
2: store final ttl value which is ft
3: send an ICMP request packet
4: **if the** *replied ICMP packet is of type destination unreachable and code message*
is the host unreachable **then?**
5: Packet is spoofed
6: **else**
7: extract the final ttl value of the receiving ICMP reply packet fr
8: **if** *fr == ft* **then**
9: The packet is not spoofed
10: **else**
11: Packet is spoofed
12: **end if**
13: **end if**

CHAPTER 4. RESULTS AND ANALYSIS

The above algorithm(see algorithm 12) is for several scenarios mentioned below (see 4.1 and 4.2). Worked on some of the topologies (see Fig 3.1 and Fig 3.2). Worked on the IPV4 version of Internet Protocol(IP).

4.1 If the source IP address exists on the Internet

To work on the above scenario one host will be running the above algorithm and the other host will be sending the spoofed and unspoofed packets.

If the spoofed IP or legitimate IP address host exists on the internet and assuming that the spoofed or legitimate IP address or network does not prohibit sending the ICMP echo reply, then the reply for ICMP echo request will be of type ICMP echo reply which can be known by the type and code fields which are present in the ICMP packet, in this case, the type and code field values are 0 and 0. Sent 250 spoofed IP packets of type UDP and TCP and 250 unspoofed IP packets of type UDP and TCP whose IP addresses are not present in the IP2Hopcount table to another host that runs the above algorithm (see algorithm 12). The algorithm was able to detect all the spoofed packets assuming that the attacker and spoofed IP address host don't have the same hop count.

Results for Scenario 4.1				
Type of Filtering Method	Number of spoofed packets sent	Number of packets detected as spoofed	Number of legitimate packets sent	Number of packets detected as legitimate
HCF method	250	250	250	0
Proposed Algo-rithm	250	250	250	250

Table 4.1: Results for Scenario 4.1

4.2 If the source IP address host doesn't exist but the network exists on the internet

To work on the above scenario one host will be running the above proposed algorithm and the other host will be sending the spoofed and unspoofed packets.

If the spoofed IP address host doesn't exist on the internet but the corresponding network exists then the reply for the ICMP request will be of type ICMP destination unreachable message and code message is host unreachable which can be known by the type and code fields of ICMP packets which are 3 and 1. Sent 250 spoofed IP packets of type UDP and TCP and 250 unspoofed IP packets of type UDP and TCP whose IP addresses are not present in the IP2Hopcount table to another host that runs the above algorithm (see algorithm 12). The algorithm was able to detect all the spoofed packets.

Results for Scenario 4.2				
Type of Filtering Method	Number of spoofed packets sent	Number of packets detected as spoofed	Number of legitimate packets sent	Number of packets detected as legitimate
HCF method	250	250	250	0
Proposed Algorithm	250	250	250	250

Table 4.2: Results for Scenario

Chapter 5. CONCLUSION & FUTURE SCOPE

In this thesis, we worked on two of the detection and filtering methods which are Hop Count Filtering and Ingress filtering. After implementing both of these methods we saw the advantages, and disadvantages of both of these methods and also worked on one of the disadvantages of the Hop count Filtering method which is when a new client sends a request to the receiving host that runs the hop count filter process sees it as spoofed one and discards it. To it, we proposed an algorithm that can help to tackle the disadvantage. Even the proposed algorithm fails if the attacker and spoofed IP address have the same hop count Hop count filtering is more of a static filter that only works if the IP address has already been visited and also fails if the attacker and spoofed IP address have the same hop count. Ingress Filtering is more of a dynamic filter that can change filter rules i.e., update or delete the prefixes that can enter into a particular network.

Analysis should be made to see the percentage of spoofed packets that carry the wrong hop count. Further analysis and experimentation are required to be done on real networks to see how effective the proposed algorithm can be and also do a further experiment on the proposed algorithm in a case where the spoofed IP address network doesn't exist on the Internet, experiment when IP packets are flooded and also the time the proposed algorithm takes to process the requests. Analysis should be made on the ingress filtering to see which method of ingress filtering should be applied on a particular link i.e. customer to provider and peering link

Bibliography

[1] Socolofsky, T. and C. Kale, ”TCP/IP tutorial”, RFC 1180, DOI 10.17487/RFC1180, January 1991, ¡https://www.rfc-editor.org/info/rfc1180¿.

[2] Postel, J., ”Internet Protocol”, STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, ¡https://www.rfc-editor.org/info/rfc791¿.

[3] https://www.juniper.net/documentation/us/en/software/junos/interfaces-security-devices/topics/topic-map/security-interface-IPV4-IPV6-protocol.html

[4] https://docs.oracle.com/cd/E19455-01/806-0916/6ja85399u/index.html

[5] https://docs.oracle.com/cd/E19455-01/806-0916/6ja8539be/index.html

[6] http://www.steves-internet-guide.com/IPV4-basics/

[7] https://www.computernetworkingnotes.com/ccna-study-guide/basic-subnetting-in-computer-networks-explained.html

[8] Fuller, V. and T. Li, ”Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan”, BCP 122, RFC 4632, DOI 10.17487/RFC4632, August 2006, ¡https://www.rfc-editor.org/info/rfc4632¿.

[9] https://study-ccna.com/what-is-IP-routing/

[10] https://www.cloudflare.com/en-in/learning/ddos/glossary/IP-spoofing/

[11] Eddy, W., ”TCP SYN Flooding Attacks and Common Mitigations”, RFC 4987, DOI 10.17487/RFC4987, August 2007, ¡https://www.rfc-editor.org/info/rfc4987¿.

[12] Postel, J., ”Transmission Control Protocol”, STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, ¡https://www.rfc-editor.org/info/rfc793¿.

[13] https://www.cloudflare.com/en-in/learning/ddos/udp-flood-ddos-attack/

[14] https://www.cloudflare.com/en-in/learning/ddos/memcached-ddos-attack/

[15] https://www.tutorialspoint.com/memcached/memcached overview.htm

[16] Wang, H., Jin, C. and Shin, K.G., 2007. Defense against spoofed IP traffic using hop-count filtering. IEEE/ACM Transactions on Networking, 15(1), pp.40-53.

[17] Ferguson, P. and D. Senie, ”Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing”, BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, ¡https://www.rfc-editor.org/info/rfc2827¿.

[18] https://spoofer.caida.org/summary.php

[19] Default TTL values in TCP/IP., 2002, [online] Available:

[20] Mopari, I.B., Pukale, S.G. and Dhore, M.L., 2008, December. Detection and defense against DDoS attacks with IP spoofing. In 2008 International Conference on Computing, Communication and Networking (pp. 1-5). IEEE.

[21] Wang, X., Li, M. and Li, M., 2009, December. A scheme of distributed hop-count filtering of traffic. In IET International Communication Conference on Wireless Mobile and Computing (CCWMC 2009) (pp. 516-521). IET.

[22] Mukaddam, A., Elhajj, I., Kayssi, A. and Chehab, A., 2014, May. IP spoofing detection using modified hop count. In 2014 IEEE 28th International Conference on Advanced Information Networking and Applications (pp. 512-516). IEEE.

[23] Baker, F. and P. Savola, ”Ingress Filtering for Multihomed Networks”, BCP 84, RFC 3704, DOI 10.17487/RFC3704, March 2004, ¡https://www.rfc-editor.org/info/rfc3704¿.

[24] Sriram, K., Montgomery, D., and J. Haas, ”Enhanced Feasible-Path Unicast Reverse Path Forwarding”, BCP 84, RFC 8704, DOI 10.17487/RFC8704, Febru- ary 2020, ¡https://www.rfc-editor.org/info/rfc8704¿.

[25] https://arxiv.org/abs/1606.07613v1

[26] https://www.wired.com/story/github-ddos-memcached/

[27] Postel, J., ”Internet Control Message Protocol”, STD 5, RFC 792, DOI 10.17487/RFC0792, September 1981, ¡https://www.rfc-editor.org/info/rfc792¿.

[28] https://ipmininet.readthedocs.io/en/latest/getting started.html

[29] http://mininet.org/

[30] https://scapy.readthedocs.io/en/latest/introduction.html