VOL. 6, NO. 12, DECEMBER 2011 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences


www.arpnjournals.com

A NOVEL SCHEME FOR DETECTING AND PREVENTING SPOOFED IP ACCESS ON NETWORK USING IP2HP FILTER

N. Arumugam[1] and C. Venkatesh[2]
[1]Anna University of Technology, Coimbatore, Tamilnadu, India
[2]Faculty of Engineering, EBET Group of Institutions, Kankayam, Tamilnadu, India
E-mail: nanptc@gmail.com

ABSTRACT

Denial of Service (DoS) attacks presents a serious problem for internet communications. It simply floods the link of the victim server with a large amount of packets leading to a high rate of packet drops for legitimate users. In general, DoS attacks are not exposed but the threat is common. The problem is aggravated when the attackers spoof their IP addresses. Defense against IP spoofing is a dominant and many approaches that could diminished the spoofing problem. Since the destination based forwarding paradigm of the Internet Protocol, IP address spoofing is both simple and very effective in evading both prevention and detection. The straightforward method of installing simple filters without proper validation at border routers is rendered inefficient by IP spoofing. The attacker can choose randomly an IP address as the source for different packets and thus make the detection method infeasible. Therefore, detecting and preventing packets with spoofed source address has been actively pursued in the research community. Many existing solutions to this problem are IP trace back, packet marking, authentication methods etc. Among these, this paper proposes a solution based on request verification cum filtering technique near the victim server. An experimental result shows that the proposed method eliminates most of the spoofed packets with moderate memory and time consumption.

Keywords: IP spoofing, hop count, packet transfer time, searching algorithm.

1. INTRODUCTION

In the current situation, the internet is an essential part of our everyday life and many important and essential services like banking, shopping, transport, health, and communication are partly or completely dependent on the internet [1]. IP spoofing is commonly associated with malicious network activities, such as Distributed Denial of Service (DDoS) attacks which block legitimate access by either exhausting victim server's resources or saturating stub networks access links to the internet. Most DDoS attacking tools spoof IP addresses by randomizing the 32-bit source-address field in the IP header which hide attacking sources and dilutes localities in attacking traffic [2].

To prevent DDoS attacks many solutions, proposed by many researchers have taken two distinct approaches, they are either a router-based or host-based. The router-based approach installs defense mechanisms inside IP routers to trace the sources of attack [3, 4, 5] or detect and block attacking traffic [6, 7, 8, 9, 10]. However, these router-based solutions require not only router support but also coordination among different routers and networks. In contrast to the router-based approach, the host-based approach can be deployed immediately. Moreover, end systems should have a much stronger incentive to deploy defense mechanisms than network service providers.

This paper proposes a trivial scheme that validates incoming IP packets at the victim internet server. The fundamental idea is to utilize inherent network information such as the number of hops and packet transfer time of the received packet takes to reach its destination. The remainder of the paper is organized as follows. Section 2 presents the related work, section 3 demonstrates the proposed methodology, section 4 discuses the different phases of proposed methodology, section 5 discuses experiment analysis and section 6 presents the conclusion of the proposed methodology.

2. RELATED WORKS

To detect and prevent DDoS attacks, there are two distinct approaches they are router-*based* and *host*-based. The router-based approach installs detection mechanisms inside IP routers to trace the source(s) of attack [11], or detect and block attacking traffic [12]. However, these router-based solutions require not only router support but also coordination among different routers and networks and wide-spread deployment to reach their potential. In contrast to the router-based approach, the host-based approach can be deployed immediately. Moreover, end systems should have a much stronger incentive to deploy defense mechanisms than network service providers. The current host-based approaches protect an internet server either by using sophisticated resource-management schemes [13], or by significantly reducing the resource consumption of each request to withstand the flooding traffic such as SYN cookies [14] and Client Puzzle [15]. Existing host-based solutions work at the transport-layer and above, and cannot prevent the victim server from consuming CPU resource in servicing interrupts from spoofed IP traffic. At high speed, incoming IP packets generate many interrupts and can drastically slow down the victim server [16]. Therefore, the ability to detect and filter spoofed packets at the IP layer without any router support is essential to protection against DDoS attacks.

3. PROPOSED METHODOLOGY

This section describes a preamble of the normal TCP connection establishment and then discussed the proposed methodology concepts for IP spoofing detection in detail.

3.1. Three way handshake-normal TCP connection

When making a TCP connection one host send a TCP packet with a SYN flag bit on in order to request connection establishment. Immediately up on receiving, the packet destination host sends back a TCP packet with ACK and SYN flags bits on to accept the request and to establish the connection in the reverse direction. The negotiation ends when the first host sends back an ACK packet. Logically TCP tries to make both upstream and downstream connection separately to achieve full duplex transmission. This process is called 3-way handshake [17]. Figure-1 shows the illustrative of three way handshake.

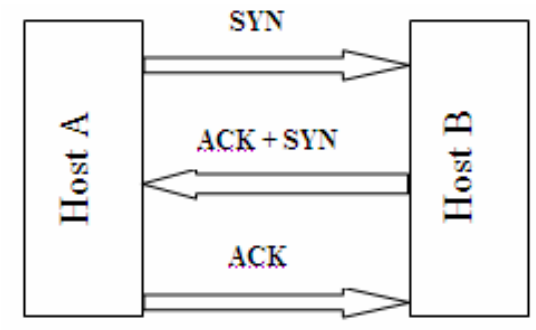

Figure-1. 3-way handshake.

The ACK packet is sent back immediately after the SYN packet is received. The ACK packet can be regarded as an ECHO packet of the SYN packet. Therefore the difference between the time when the SYN packet is sent and the time when the ACK packet returns can be used as the approximation of the packet transfer time between two hosts.

3.2. Proposed lightweight scheme

In this paper, a lightweight scheme has been proposed that validates incoming request before it reaches the protected server without using any cryptographic methodology. The fundamental idea is to utilize inherent network information that each packet carries. The inherent network information we use here is the number of hops and packet transfer time (PTT) of a request packet takes to reach its destination. Since there is a strong correlation among hop count and PTT, this paper considered these two metrics for the IP spoofing detection process. In internet an attacker can forge any field in the IP header except the number of hops an IP packet takes to reach its destination, which is solely determined by the Internet routing infrastructure. The hop-count information is indirectly reflected in the Time-to-Live (TTL) field of the IP header, since each intermediate router decrements the TTL value by one before forwarding a packet to the next hop. Figure-2 shows pictorial representation of the proposed methodology.

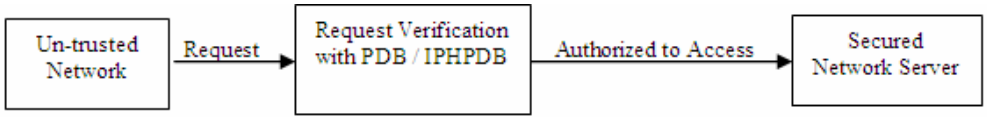

Figure-2. Proposed methodology.

Instead of normal procedure, in the proposed method request is authorized to access the server based on request validation and verification. The graphical representation of the proposed methodology is shown in Figure-3. It represents how an IP request is validated and tested in detail. The validation and testing process is simply comparing the request details with the stored values in a data base called IPHP DB (IP Hop count Packet Transfer Time Data base). It consists of source IP address along with its appropriate hop count, PTT value. This technique is straight forward at the same time the implementation is not a simple one. An obvious solution is to collect the IP address and the relevant hop count value when there is no DDoS attack. There are many problems with this approach. First, during a DDoS attack the victim sees a large number of previously unseen address and all of them are considered spoofed because they are not in the database. Second due to frequent routing changes, the hop count value is most likely change leads to false positive. A better approach to building the database is to add the IP address, hop count value and packet transfer time (PTT) for each TCP session separately after the TCP handshake is completed. This guarantees the integrity of the tuple (IP address, hop count, PTT) will used to detect the spoofing of IP address.

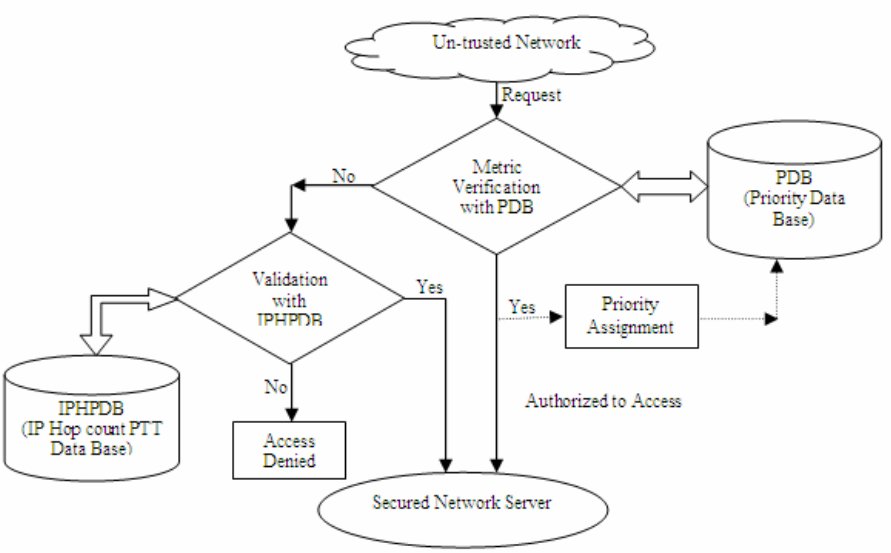

Figure-3. IP spoofing detection mechanism.

4. PHASES OF PROPOSED METHODOLOGY

This section describes different phases of the proposed methodology to detect and filter out the bogus request and also explain the functions of each phases are to be implemented. Figure-4 shows the block diagram approach of different phases of detection procedures.

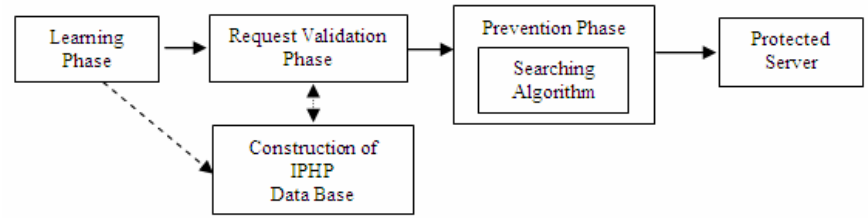

Figure-4. Different phases of fake IP detection.

4.1. Learning phase

During this phase the inherent information carries by the request packet are registered in a database called as IPHP DB (IP Hop count Packet Transfer Time Data base) is placed before the server. It contains the successfully connected request packet details like source IP and the respective hop counts with PTT. It is assumed that during normal connection no attacks are happening there, hence the respective source IP and its hop count, PTT values can be registered in the mapping table. This record is later used to verify each incoming packet and filter out the spoofed ones. The learning phase continues for a sufficient time to allow most of the database to be filled up. Figure-5 shows the pictorial representation to collect the IP packet metrics.

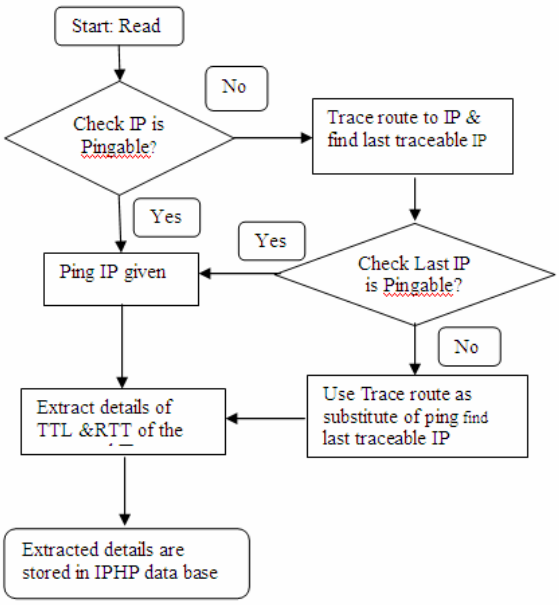

Figure-5. The pictorial representation to collect the requested IP packet metrics.

4.1.1 Building of IPHP data base

Building an accurate IPHP data base is critical thus the following methods were implemented to construct and update the data base:

a) Initialize the data base
b) Up-to-date the data base

4.1.1(a) Initialization of the IPHP data base

Initially the server administrator have to collect the metrics of requested source IP details such as IP address, hop-count values, packet transfer time (PTT) of each request. At the beginning, IPHP data base is developed with metrics based on the assumptions that successful request are not spoofed. The preliminary collection period should be long enough to ensure good accuracy even at the very beginning and the duration should depend on the amount of daily traffic the server is receiving. For a popular site, the collection period of a few days could be sufficient, while for a lightly loaded site, a few weeks might be more appropriate.

4.1.1(b) Updating the IPHP data base

The data base must be kept up-to-date as hop-counts of existing IP addresses change. The hop-count from a client to a server could change as a result of relocation of networks, routing instability, or temporary network failures. Some of these events are transient and therefore can be omitted, but longer-term changes in hop-count must be captured.

According to [18], the hop count number is started with 4 and the maximum number of hops is 30. Hence multiple IP addresses may have the same hop count values. Unfortunately an attacker may have the same hop count values as that of spoofed IP address. It is cautious to examine hop count distributions at various locations in the internet. The most critical aspect in initializing and updating the mapping table is to ensure that only valid mappings are stored in the table. At the time of initialization of this table the administrator have to provide provision for new entries whenever new legitimate IP is signed. As hop count from client to server could change due to router instability or network failure, the table should be up to date updated. This update function will execute after a fixed time span and it will change the entry only after the completion of three way hand shaking of TCP connection.

4.1.2. Hop count computation

The number of hops a packet takes to reach its destination reflected in the Time-to-Live (TTL) field of the IP header and also each intermediate router decrements the TTL value by one before forwarding a packet to the next hop. Since hop count information is not directly stored in the IP header and it might compute from the final TTL value. TTL indicates the time in which a packet can exist on the network [19]. It is defined to prevent a packet from circling on the network and it is decremented by one when passing through one router. Hence it is possible to calculate hop count from the TTL value. TTL is an 8-bit field in the IP header, originally introduced to specify the maximum lifetime of each packet in the Internet.

There are two methods to measure the hop count from a host. One is an active measurement and the other is a passive measurement. The first method is to use ICMP ECHO packets. In most cases this gives an accurate hop count. However applying this method to thousand hosts is not realistic because sending lots of ICMP packets is not recommended as a measuring method. The second method is simply to subtract the TTL of a received IP packet from its initial value. This can be done without sending any sample packets and therefore is ideal of measuring the hop counts of many hosts. However in order to use this method the initial TTL values should be known in advance.

Hop count = (initial TTL) - (TTL)

4.1.3. Problem with initial values of TTL

According to RFC 1700, the recommended initial TTL value is 64. However this rule is often ignored on the real internet. Swiss Academic and Research Network (SWITCH) have researched initial TTL values of different OS (Operating Systems). As a result there are six initial TTL values: 30, 32, 60, 64, 128 and 255. The packet whose initial TTL value 255 and the initial TTL value 128 can be distinguished from other easily. However it is more difficult to assess packets whose TTL values are less than 60 or 64. The same problem occurs to the packets with TTL value less than 30. The popular OS like Microsoft Windows, Linux and Free BSD are using 32 and 64 as initial values. Hence the following formula to convert TTL to hop count,

Hop Count =	
32-TTL	TTL<=32
64-TTL	TTL<=62
128-TTL	TTL<=128
255-TTL	TTL<=255

4.1.4 Packet transfer time

It is also necessary to establish a method to measure packet transfer time between the measuring point and the target host. Since there is no time stamp field in IP header, it is impossible to calculate packet transfer time by simply analyzing the packet header. The most accurate method to measure transfer time is to use measurement software which sends and receives sample packets. However, this method can be used only for specific hosts by executing a measurement program and therefore it is not suitable to collect packet transfer times of a number of hosts. Another way is to send an ICMP ECHO packet and count the time till its reply return from the host. This method seems to work well but it is very tedious to sending many ICMP packets and also the round trip time of ICMP packets differ from IP packets.

Another passive method is to capture and analyze TCP handshaking packets. This method makes it possible to collect roundtrip times of a large number of hosts without sending any unnecessary packets. When making a

TCP connection one host send a TCP packet with a SYN flag bit on in order to request connection establishment. Immediately up on receiving the packet destination host sends back a TCP packet with ACK and SYN flags bits on to accept the request and to establish the connection in the reverse direction. The negotiation ends when the first host sends back an ACK packet. Logically TCP tries to make both upstream and downstream connection separately to achieve full duplex transmission. This process is called 3-way handshake [20].

We used active method of measurement using two different measurement tools: Ping and Trace route. The tools utility uses TCP to measure the PTT between a host and a web server. Every Ping generates 'n' sequences of TCP request-response and reports the measured PTT. Trace route also measures the number of hops (or) routers between the host and server. The PTT and the number of hops were recorded as shown in the Table-1.

4.2. Validation phase

A strong linear correlation has been observed between hop count and packet transfer time. The graphical representation shown in Figures 6-7 confirmed the association of two metrics. Hence this paper considered these metrics along with the respective source IP for the detection of spoofed IP. During this phase the requested IP details are compared with the IPHP data base .Based on the testing result the request is permitted to access the server or not. To minimize the searching time an additional data base denoted as PDB is developed. It contains the details of the successfully connection established request details. Figure-6 shows another plotted line graph of PTT on the y-axis and hops count on the x-axis. At both measuring points, a correlation of hop count and packet transfer time can be observed. In the graph, it is clear that packet transfer time increases as the hop count grows. This establishes the fact that there is a relationship between PTT latency and hops count [21].

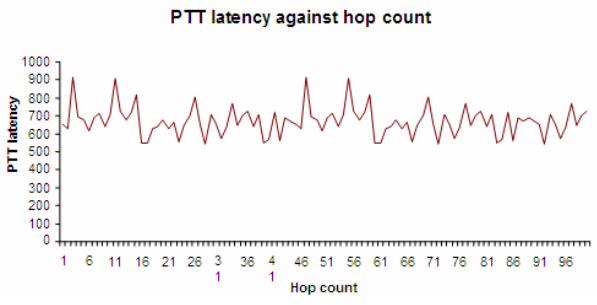

Figure-6. Graph of PTT latency against hop count.

According to [22], the RTT depends on the distance to the destination as well as the bandwidth and congestion along the path. The round trip time (RTT) as a function of hop count. For each destination the most frequent path is chosen and put into a hop-count bin based on its length. From these bins, the 10^{th}, 50^{th}, and 90^{th} percentiles of RTT are calculated.

4.3 Prevention phase

This phase decides whether the request is allowed to access the server or not. In other words successfully tested request in the validation phase is allowed to access the server if not the request is drop out. Basically in this phase the features of requested IP are matched with the IPHP data base, based on the result the request is treated as legitimate or a fake request.

4.3.1 Address lookup and switching process

Both validation and prevention phase datagram's source IP address is used as a key for identify the genuineness of the request otherwise called as address lookup. Once the verification information is retrieved, the proposed line of defense scheme can permit the request to access the protected server. This process is called as *switching*. One of the problems of the proposed method is the searching time of IP look up and it is unavoidable. But it can be minimized by proper methodology. One of the appropriate searching methods is proposed to reduce the searching time. To reduce the time complexity for searching process many lookup algorithms available. One such is the Elevator-Stairs Algorithm. It provides a total search time of O (w/k + k) by indexing hash Table to Practical Algorithm to Retrieve Information Coded in Alphanumeric (PATRICIA), where w is the length of the IP address and k is the level of tree. Elevator Stairs Algorithm uses linear search at the k-level is modified to binary search at the k-level of tree. At the k^{th}-level, non branching nodes are added to jump k levels of tree which reduces the time for searching in the tree. It provides a better search time over the existing Elevator-Stairs Algorithm, by accomplishing a two-way search in the tree.

4.3.2 Elevator stairs algorithm

The basic concept of Elevator Stairs algorithm is a binary tree similar to a scenario wherein a tall building has an elevator that stops only at certain intermediate floors. A passenger desiring to reach any floor from the topmost floor of the building can take the elevator up to the nearest possible upper floor and then reach the destination floor by taking the stairs. In this tree, traversed path of any two nodes are compressed through the removal of non-branching nodes between them. IP lookup is initialized by searching the longest matching prefix in the k-level tree, which is build from tree by adding non-branching nodes.

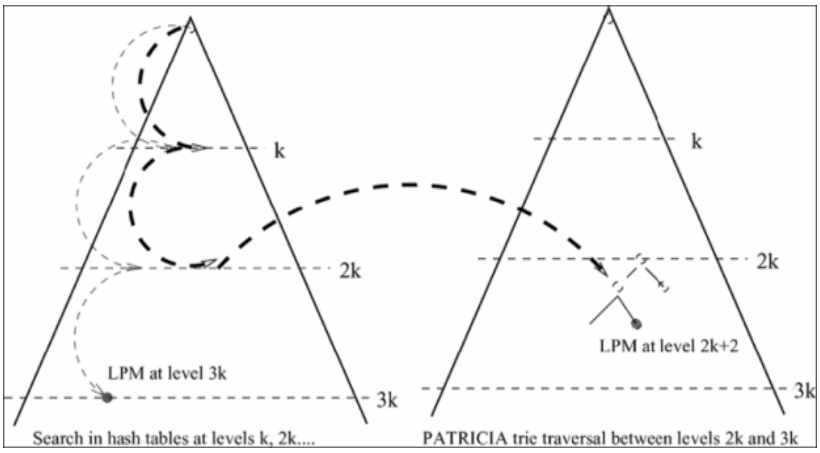

Figure-7. Searching in Elevator-Stairs algorithm.

As a next step, multiple bit key corresponding to the matching prefix is determined. Then a search of the hash table at the levels k, 2k, 3k, is performed, where the value of k is a constant between zero and maximum length of the IP address. If it matches with the level it searches for the destination node in the unmodified tree using linear search. The Elevator-Stairs algorithm uses optimal *O (N)* memory and achieves better lookup and update times than other methods with similar memory requirements. Figure-7 shows the searching in elevator-stairs algorithm. The log*W*-Elevators algorithm gives *O* (log *W*) lookup time [23]. The space required for a binary tree is *O (NW)* where *N* is the number of strings and *W* is the maximum length of a string. The space can be reduced to *O (N)* by compressing each maximal non branching path into one edge, which called PATRICIA tree. Lookup time in the k-level of the PATRICIA is of the order O (w/k). Total search time of the algorithm is of the order O (w/k+k).

4.3.3 Priority data base (PDB)

We can further minimize the searching time, based on the past successfully permitted request IP metrics accumulated in a separate data base called a priority data base (PDB) which is placed before IPHP database. This data base is constructed by assumption that once successfully established connections are guaranteed to be non-spoofed. Initially each request is taken care by the PDB before starting the look up validation. Successful request can directly permit to access the server without further validation process.

4.3.4 Mismatches counter (MMC)

To keep the record of unauthorized access, we use a counter called Mismatches Counter (MMC), which counts the number of packets whose identification, cannot be matched. This includes both packets with incorrect IP identification as well as packets from unknown source addresses that are not recorded in the mapping Table. When the MC value becomes greater than a threshold, it is considered as a signal of DoS/DDoS attack. The value of MC is reset to zero after fixed intervals to ensure that the cumulative results over a long duration is not considered as the indication of attack by mistake.

4.4 Filtering phase

Using the techniques and criteria discussed above, filtering procedure is described below. Any packet received by the edge router is authorized to permit the request according to the following rules:

a) Successful completion of the requested source IP metrics during the verification phase.
b) If the source IP address and its hop count of the packet exist but the PTT value does not match, then it is a partially spoofed packet and the request is accepted.
c) 3) If the source IP address and its PTT values of the packet exist but the hop count PTT value does not match, then it is a partially spoofed packet and the request is accepted.
d) If the source IP address does not appear in the IPHP data base then the packet is considered as a new request, accepted to access the server.

5. EXPERIMENTAL ANALYSIS

To finds out the correlation coefficient *r* between the metrics hop count and the packet transfer time of the server to each request, a small test bed consisting of server /client environment running on Pentium(R) 4CPU 3.0-GHz machine with 1-GB RAM using java jdk 1.6 as a front end tool and Microsoft SQL-Server 2000 as a back end tool.

5.1 Computing correlation coefficient

To calculate the correlation among hop count and PTT, a server machine of 250 GB HDD, 1 GB RAM hp core2 duo 2.3 MHz with operating system of EL4 Red hat Linux was connected to the internet from our institution and the data were collected using the TCP Ping and TCP Trace route tools on February 11, 2011. A TCP Ping tool on the host machine connected to the Internet through our department network was run to ping a particular web server and the PTT was observed and recorded. Immediately after that, a TCP Trace route was also run to

the same web server and the hops count was observed and recorded. Table-1 shows the details of PTT and hops count values obtained from the experiments. Similarly Figure-8 shows the Scatter graph of PTT latency against hops count and describing the linear relationship between the two variables.

Table-1. Measured RTT and hops count.

	Hops	**RTT** (ms)
www.annuniv.edu	8	46.506
www.britanni.co.in	8	43.968
www.cosel.com	6	50.283
www.tpub.com	6	65.743
www.google.com	7	23.691
www.iit.com	5	44.186
www.iitm.ac.in	5	15.944
www.isro.com	5	17.257
www.msn.com	5	15.184
www.nokia.com	5	16.010
www.panasonic.co.in	11	55.033
www.pepsicoindia.co.in	6	58.917
www.rcom.co.in	11	50.790
www.sify.com	9	42.512
www.sony.com	4	26.152
www.titanworld.com	5	69.403
www.yahoo.com	5	50.672
www.niit-tech.com	5	44.393

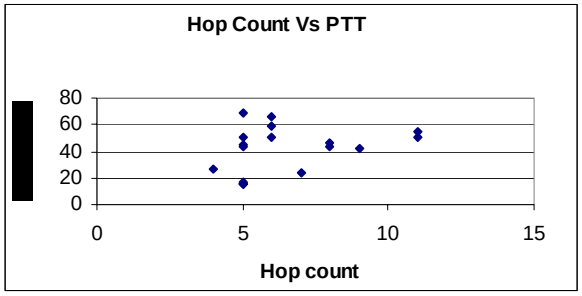

Figure-8. PTT and hops count.

5.2 Correlation coefficient

The correlation coefficient is a measure of linear association between two variables. The value of the correlation coefficient ranges from 1.00 to -1.00. A value of 0.0 indicates that there is absolutely no relationship between the X and Y variables. The strength of the relationship between the X and Y variables increases as the value of r approaches 1.00 and -1.00. Perfect correlation occurs if r equals either 1.00 (perfect positive) or -1.00 (perfect negative). Positive correlation coefficients indicate that an increase in the value of the X variable results in an increase in the value of the Y variable. Negative correlation coefficients indicate that an increase in the value of the X variable results in a decrease in the value of the Y variable. We have calculated the correlation coefficient (r) using the formulae as below:

Correlation(r) = $[N\Sigma XY-(\Sigma X)(\Sigma Y)/\sqrt{([N\Sigma X^2-(\Sigma X)^2][N\Sigma Y^2-(\Sigma Y)^2])}]$

where

Numbers of values or elements are:

X = First score
Y = Second score
ΣXY = Sum of the product of first and second scores
ΣX = Sum of first scores
ΣY = Sum of second scores
ΣX^2 = Sum of square of first scores
ΣY^2 = Sum of square of second scores

From the sequence of experiment value we only take 6 numbers of events and calculated the correlation coefficient value r as below. This paper choose 6 event out of 18 events hence N = 6 then,

$\sum X = (4+5+5+6+8+11) = 39 = 6.5$
$\sum Y = (26.152+44.393+69.403+50.283+46.506+55.033) = 291.78 = 48.63$
$\sum XY = 325.$

Correlation coefficient (r)
$= [N\Sigma XY-(\Sigma X)(\Sigma Y) / \sqrt{([N\Sigma X^2-(\Sigma X)^2][N\Sigma Y^2-(\Sigma Y)^2])}]$
$\Sigma X^2 = 47.83$
$\Sigma Y^2 = 2531.87$

Correlation co-efficient = (r) = 0.92236. Thus it is high enough to say that there is a strong correlation between hop count and packet transfer time. Hence this paper considered these two metric along with the IP address for detecting fake IP access.

5.3 Evaluation of the proposed methodology

The proposed technique is experimented with the simulated server/client environment using java jdk 1.6 as a front end tool and Microsoft SQL-Server 2000 as a back end tool. During the normal TCP connection establishment, few metrics of the requested source IP details such as hop count, packet transfer time are registered in a database called IPHP DB (IP Hop count Packet Transfer Time data base). These metrics are calculated using the tracert command. It is a program that shows the route over the network between two systems, listing all the intermediate routers a connection must pass through to get to its destination. Instead of normal client / server request response, in the proposed experiment setup the requisite source IP detail are validated. Based on the validation result, permission is provided to the requisite access the protected server. Hence there is an additional time is required for each request to access the server and it is unavoidable. Let us assume the request response time between the clients / server in normal condition is t ms.

But the proposed light weight scheme takes t+Δt ms, where Δt is an additional time taken for the validation process. Table-2 shows the server response time for a request between the normal and the proposed method.

Table-2. Request and response time of the server.

Scheme	Without spoofing verification	With spoofing verification
Response time in ms	373ms	1125ms

For normal procedure the request response time between the server/client took 373ms. Whereas the proposed method took slightly more time due to the checking process of the originality of the request. Hence the establishment time of the proposed method took nearly 1125ms. Figure-9 shows the graphical representation of the difference between the normal and proposed IP spoofing verification method.

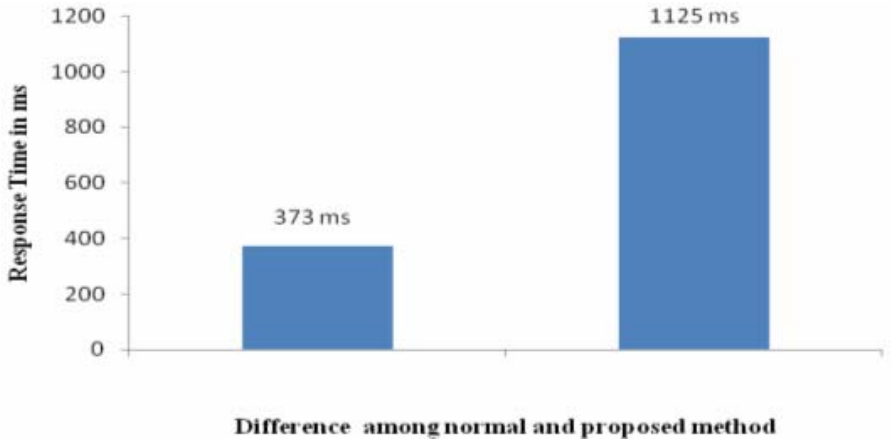

Figure-9. Server response time between normal and proposed method.

Since the metrics of each request is validated with the data base naturally the time consumption by the proposed method is high. To minimize the validation time this paper builds the metrics of the requested source IP details in an elevator-stairs data structure. Since the searching time of elevator-stairs algorithm is O (log W) leads to reduction in the validation time. To confirm this by an experimental evaluation, the data structure is developed for 100 requested IP address as node and the corresponding searching time is also calculated.

Table-3. Request and response time of the server.

Scheme	Number of nodes	Normal searching	Proposed searching
1.	20	3.864	0.368
2.	40	4.83	0.46
3.	60	5.796	0.552
4.	80	7.728	0.738
5.	100	9.66	0.92

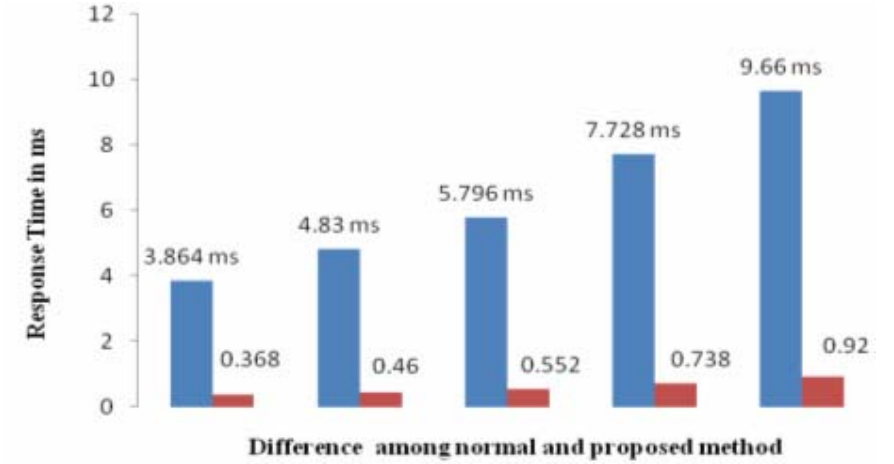

Figure-10. Searching time difference between normal and proposed methods.

Table-3 shows the searching time of normal and proposed method for different number of IPs. The graphical representation as in the Figure-10 shows the difference in searching time among the existing and the proposed method. It is confirm that the proposed method required minimum searching time than normal method.

6. CONCLUSIONS

This paper presents an IP2HP filtering scheme that detects and discards spoofed IP request to access a protected network server resource. To detect the spoofed request, hop count and packet transfer time of the requested source IP is compared with the existing data base placed near the protected server. By default these metrics are stored in a data structure during the learning state. The metrics used for the validation is indirectly reflected in the Time-to-Live (TTL) field of the IP header, the attacker cannot falsify the number of hops an IP packet takes to reach its destination, which is solely determined by the Internet routing infrastructure. And also an attacker does nothing even knows its design. Using this concept this paper discards the spoofed IP request by validating the metrics of requested IP at the victim server. In the mean time a moderate amount of searching time and memory storage are required during the learning stage. This problem is also minimized by deploying a binary tree data structure instead of a mapping table. Hence the proposed method eliminates most of the spoofed packets with moderate memory and time consumption.

REFERENCES

[1] Yao Chen, Shantanu Das, Pulak Dhar, Abdulmotaleb El Saddik and Amiya Nayak. 2008. Detecting and Preventing IP-spoofed Distributed DoS Attacks. International Journal of Network Security. 7(1): 70-81.

[2] Haining Wang, Cheng Jin and Kang G. Shin. 2007. Defense against Spoofed IP Traffic Using Hop-Count Filtering. IEEE/ACM Transactions on Networking. 15(1), February.

[3] S. Savage, D. Wetherall, A. Karlin and T. Anderson. 2000. Practical network support for IP trace back. In: Proc. ACM SIGCOMM, pp. 295-306.

[4] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent and W. T. Strayer. 2001. Hash-based IP trace back. In: Proc. ACM SIGCOMM, pp. 3-14.

[5] D. Song and A. Perrig. 2001. Advanced and authenticated marking schemes for IP trace back. In: Proc. IEEE INFOCOM, 2: 878-886.

[6] P. Ferguson and D. Senie. 1998. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. RFC 2267, Jan.

[7] J. Ioannidis and S. M. Bellovin. 2002. Implementing pushback: Router-based defense against DDoS attacks. In: Proc. NDSS'2002, San Diego, CA, Feb.

[8] A. D. Keromytis, V. Misra and D. Rubenstein. 2002. SOS: Secure overlay services. In: Proc. ACM SIGCOMM, pp. 61-72.

[9] J. Li, J. Mirkovic, M. Wang, P. Reiher and L. Zhang. 2002. Save: Source address validity enforcement protocol. In: Proc. IEEE INFOCOM, pp. 1557-1566.

[10] K. Park and H. Lee. 2001. On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law Internet. In: Proc. ACM SIGCOMM, pp. 15-26.

[11] M. Sung and J. Xu. 2002. IP traceback-based intelligent packet filtering: A novel technique for defending against Internet DDoS attacks. In: Proc. IEEE ICNP, pp. 302-311.

[12] W. Lee and S. J. Stolfo. 1998. Data mining approaches for intrusion detection. 7' USENIX Security Symposium, San Antonio, TX. pp. 79-93, January.

[13] I. B. D. Cabrera *et al.* 2001. Proa+-ctive Detection of Distributed Denial of Service Attacks using MIB Traflic Variables - A Feasibility SNdy. IFIPi EEE Int. Symp. On Integrated Network Management, Seattle, W.A, May.

[14] Y. Huang and J. M Pullen. 2001. Countefmg Denial uf-Sewice anacks Using Congestion Triggered Packet Sampling and Filtering. Proc. IO' ICCCN, Anzona, USA, Oct.

[15] T.M. Gil and M Poleto. 2001. MULTOPS: a data-structure for bandwidth attack detection. USENlX Security Symposium Washington, DC. pp. 23-38, Aug.

[16] Haining Wang. 2007. Member, IEEE, Cheng Jin, and Kang G. Shin, Fellow, IEEE. Defense against Spoofed IP Traffic Using Hop-Count Filtering. IEEE/ACM Transactions on Networking. 15(1), February.

[17] S. M. Bellavin. 2001. ICMP traceback messages. Internet Draft.

[18] C. Banos. 2000. A proposal for ICMP traceback messages. Internet Draft, Sept.

[19] 2009. http://www.opus1.com on 12th October.

[20] K. Fujii and S. Goto. Correlation between Hop Count and Packet Transfer Time. APAN/IWS2000, http://netresearch.ics.uci.edu/kfujii/iws2000.pdf

[21] W. Richard Stevens. 1994. TCP/IP Illustrated. Vol. 1. Addison Wesley Longman, Inc.

[22] Friday Yakubu1 and Shehu M.T. *et al*. Correlation between Latency and Hop Count. Department of Mathematics, Ahmadu Bello University, Zaria.

[23] http://www.caida.org/tools/measurement/skitter/RSSAC/ accessed on 30th April, 2011.

[24] Rama Sangireddy and Natsuhiko Futamura *et al*. 2005. Scalable, Memory Efficient, High - Speed IP Lookup Algorithms. IEEE/ACM Transactions on Networking. 13(4), August.