Packet Monitoring Approach to Prevent DDoS Attack in Cloud Computing

Vikas Chouhan & Sateesh Kumar Peddoju

Electronics & Computer Engineering Department,Indian Institute of Technology Roorkee, Roorkee-247667, India

Abstract - In cloud environment, cloud servers providing requested cloud services, sometimes may crash after receiving huge amount of requests. This is exactly what happens in a denial of service (DoS) attack. It prevents the authentic clients from getting service. DoS attack is accompanied by IP Spoofing so as to hide the source of flooding and to make every request look different.

In this paper, we present an approach for packet monitoring in Cloud Environment to prevent DDoS attacks. This new approach of Hop Count Filtering provides a network independent and readily available solution to prevent DoS attack in Cloud environment. Also, this method decreases the unavailability of cloud services to legitimate clients, reduces number of updates and saves computation time. The presented approach is simulated in CloudSim toolkit environment and corresponding results are then produced.

Keywords - *Cloud Computing, TTL, IP, Hop Count, Denial-of-Service.*

I. INTRODUCTION

Cloud computing can be defined as a new style of computing in which dynamically scalable and often virtualized resources are provided as a services over the Internet. Advantages of the cloud computing technology include cost savings, high availability, and easy scalability [1].

DoS attacks do not wish to modify data or gain illegal access, but instead they target to crash the servers and whole networks, disrupting legitimate users' communication. DoS attacks can be launched from either a single source or multiple sources. Multiple-source DoS attacks are called distributed denial-of-service (DDoS) attacks [2].

When the operating system notices the high workload on the flooded service, it will start to provide more computational power to cope with the additional workload. The attacker can flood a single, system based address in order to perform a full loss of availability on the intended service [3, 4].

These attacks are a type of Flooding Attack [2, 5], which basically consist of an attacker sending a large number of nonsense requests to a certain service, which is providing various services under cloud. As each of these requests has to be handled by the service implementation in order to determine its invalidity, this causes a certain amount of workload per attack request, which in the case of a flood of requests usually would cause a Denial of Service to the server hardware [2].

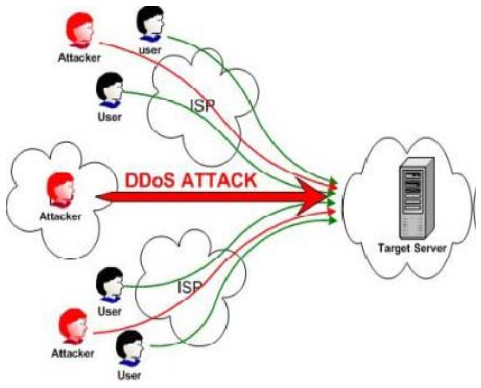


Fig. 1 : DDoS attack [2]

II. HOP-COUNT COMPUTATION

Since hop-count information is not directly stored in the IP header, one has to compute it based on the Time-to-live (TTL) field. TTL is an 8-bit field in the IP header, originally introduced to specify the maximum lifetime of each packet in the Internet. Each intermediate router decrements the TTL value of an in-transit IP packet by one before forwarding it to the next-hop [6, 7].

A. *Extract final value of TTL*

When a Packet reaches its destination and extracting its TTL field value, this value is known as final TTL. The challenge in hop-count computation is that a destination only sees the final TTL value. It would have been simple had all operating systems (OSs) used the same initial TTL value, but in practice, there is no consensus on the initial TTL value. Furthermore, since the OS for a given IP address may change with time, we cannot assume a single static initial TTL value for each IP address [6].

B. *Investigate the initial value of TTL*

According to [6], most modern OSs uses only a few selected initial TTL values, 30, 32, 60, 64, 128, and 255. Only a few Internet hosts are apart by more than 30 hops, thus one can determine the initial TTL value of a packet by selecting the smallest initial value in the set that is larger than its final TTL. For example, if the final TTL value is 112, the initial TTL value is 128, the smallest of the two possible initial values, 128 and 255. Thus, given the final TTL value one can find the initial TTL value. Initial TTL values can be calculated as follows [8]:

Initial TTL=32 if final TTL <=32

Initial TTL =64 if 32<final TTL<=64

Initial TTL =128 if 64<final TTL <=128

Initial TTL =255 if 128<final TTL <=255

C. *IP2HC Table*

The IP2HC table [8] is a mapping between Source IP Address of a packets and stored hop count for that IP Address. It is a structure with Source IP address serving as index to match the hop count information.

III. PROPOSED ALGORITHM TO PREVENT DOS ATTACK

The proposed algorithm, uses the hop count filtering mechanism, and provides a clear idea of implementation so that it can be used in Cloud environment to prevent DoS attacks.

The algorithm requires continuous monitoring of packets travelling over the network in the Cloud, and thus, we extract SYN flag, TTL and source IP information from these monitored TCP/IP packets. The algorithm recognises four cases for each captured packet in the whole operation.

i. If SYN flag is set and source IP address exist (Syn=1 and Src=1) in IP2HC table then calculate hop-count by using TTL value of IP packet. Now check if the hop-count matches with the stored hop-count, if not, then update source hop-count field of table for that source IP address.

ii. If SYN flag is set and source IP address does not exists (Syn=1 and Src=0) in the IP2HC table then calculate hop-count and add a new entry for the Source IP address with the corresponding hop count in the IP2HC table.

iii. If SYN flag is not set and source IP address exists (Syn=0 and Src=1) in IP2HC table then calculate hop-count and if this hop count does not matches the stored hop count entry in the IP2HC table for the corresponding source IP address, then packet is spoofed, else the packet is legitimate.

iv. If SYN flag is not set and source IP address does not exists (Syn=0 and Src=0) in IP2HC table then it means that the packet is spoofed, because every legitimate IP address having a valid TCP connection will have its entry in the IP2HC table.

The inspection algorithm extracts the source IP address and the final TTL value from each IP packet [6]. The algorithm infers the initial TTL value and subtracts the final TTL value from it to obtain the hop-count. The source IP address serves as the index into the table to retrieve the correct hop-count for this IP address. If the computed hop-count matches the stored hop-count, the packet has been “authenticated” otherwise; the packet is likely spoofed [6].

ALGORITHM -1

Consider the following notations:

synflag = Syn bit of TCP packet.

mcount =malicious packet counter.

Tf= final value of TTL.

Ti=initial value of TTL.

```
Initialize mcount=0;
  For each packet
  Set TTL = ExtractFinalValueOfTTL( );
          //get time-to-leave field of IP packet
  Set srcIp = ExtractSourceIP( );
          //get source IP address from IP packet
  Set synflag = ExtractSynBit( );
         //get Syn flag value from TCP packet
```

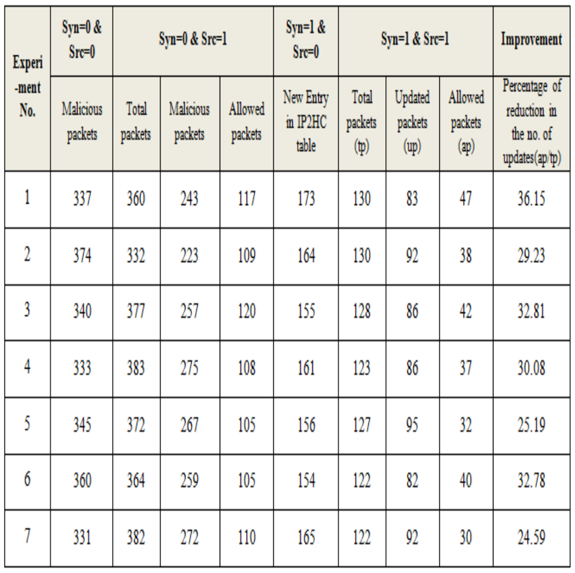
```
If (synflag is set)
{
   If (establish_tcp_connection( ))
           //true when connection established
   {
       If ( srcIp is exist in IP2HC table )
       {
          ComputePacket ( srcIp , TTL , synflag);
            // function call which filter the spoofed packet
       }
       else //new connection packet
       {
          Hc=ComputeHopCount( TTL );
             //get hop-count value
          NewEntryInTable(srcIp,Hc);
             //Add entry into IP2HC table
       }
   }
   else
   {
         // ignore packet
   }
}
else //synflag is not set
{
    If ( srcIp exist in IP2HC Table)
    {
        ComputePacket ( srcIp , TTL, synflag );
          // function call which filter the spoofed packet
    }
    else
    {
        'drop the packet' //Packet is spoofed
        mcount++;    // increment in malicious packet by 1
      }
}
ComputePacket ( string srcIp , int Tf , boolean synflag)
{
    Hc=ComputeHopCount( Tf ); //get hop-count value
    Hs=RetreiveStoredHopCount(srcIp);
        //get stored hop-count value
    If ( Hc != Hs )
    {
        if( synflag is set)
        {
           UpdateTable ( srcIp , Hc);
              //update hop-count value in IP2HC table
        }
        else
        {
           'drop the packet' //Packet is spoofed
           mcount++;
             // increment in malicious packet by 1
        }
    }
    else
    {
        'allow the packet' // packet is legitimate
    }
}

int ComputeHopCount( int Tf )
{
    Set Ti= InvestigateInitialTTL(Tf);
    return Ti - Tf; //return hop-count value
}
```

IV. SIMULATION RESULTS

We simulated our algorithm on CloudSim toolkit having an arrival rate of 1000 packets per/sec at cloud server. Experimental results are shown in Table I. Various conditions discussed in Table I include pair of SYN flag (Syn) and source IP Address (Src) to provide information of packet. A value of Syn=0 represent SYN flag not set and Syn=1 represents SYN flag is set. Similarly, Src indicate the presence of source IP Address in IP2HC table. A value of Src=0 represent entry does not exist and Src=1 represents entry exists. First experiment consists of 580 (337+243, see Table I) malicious packets, and 173 new entries and only 83 entries are updated. In contrast, the packets which require an update in the table are 130 (Syn=1 & Src=1). So, the effective (in fact, reduced) number of updates are 47 (130-83). The total reduction in the number of updates in the table is 30.15% (total allowed packets/ total packets), which is a considerable amount of improvement over the conventional method.

TABLE I : RECEIVED NUMBER OF PACKETS



Experi -ment No.	Syn=0 & Src=0	Syn=0 & Src=1			Syn=1 & Src=0	Syn=1 & Src=1			Improvement
	Malicious packets	Total packets	Malicious packets	Allowed packets	New Entry in IP2HC table	Total packets (tp)	Updated packets (up)	Allowed packets (ap)	Percentage of reduction in the no. of updates(ap/tp)
1	337	360	243	117	173	130	83	47	36.15
2	374	332	223	109	164	130	92	38	29.23
3	340	377	257	120	155	128	86	42	32.81
4	333	383	275	108	161	123	86	37	30.08
5	345	372	267	105	156	127	95	32	25.19
6	360	364	259	105	154	122	82	40	32.78
7	331	382	272	110	165	122	92	30	24.59

For computation time simulation the sample inputs are taken as arrival rate 'A' and various results has been analyzed and presented in Table II.

TABLE III : SAMPLE INPUTS

Sample	Sample input (Arrival rate in packets/sec)	Computation time (in ms)
1	1000	20
2	2000	38
3	4000	26
4	6000	25
5	8000	44
6	9000	62
7	10000	68

The graph in (Figure. 2) show how our proposed approach saves the possible computation time. In case of samples 2, 3 and 4 there is variation. Sample 2 needs more time then sample 3 and 4 because it depends on receiving field of packets. Computation time is relevant factor for performance measurement of cloud network and it improves processing power of cloud server and minimizes loss of available resources.

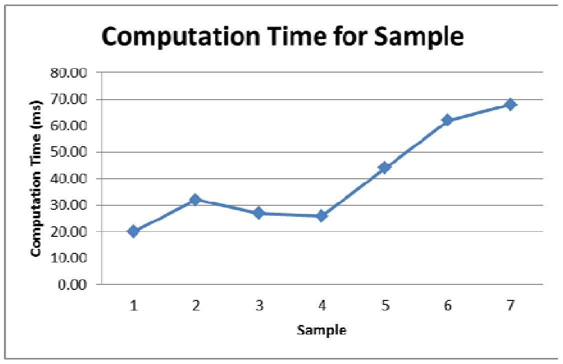


Fig. 2 : Graph showing computation time

IV. CONCLUSIONS

Cloud Computing is gaining popularity, but with the widespread usage of cloud the issue of cloud security is also surfacing. One of the major threats to Cloud security is Distributed Denial of Service Attack (DDoS) or simply Denial of service attack (DoS). To improve resource availability of resources, it is essential to provide a mechanism to prevent DDoS attacks. One of the methods for prevention is Hop Count Filtering method (HCF). This paper presented a version of Hop Count filtering method which not only detects malicious packets but also includes update of IP to Hop count Table (IP2HC) with a mechanism that reduces the number of updates and thus saves computation time by analyzing SYN flag of TCP protocol.

REFRENCES

[1] B. Furht and A. Escalante, Handbook of Cloud Computing: Springer, 2010, pp. 3-11.

[2] D. GARG, "DDOS Mitigation Techniques-A Survey," in International Conference on Advanced

Computing, Communication and Networks, 2011.UACEE '11, pp. 1302-1309.

[3] P. A. R. Kumar and S. Selvakumar, "Distributed Denial-of-Service (DDoS) Threat in Collaborative Environment - A Survey on DDoS Attack Tools and Traceback Mechanisms," in Advance Computing Conference, 2009. IACC 2009. IEEE International, 2009, pp. 1275-1280.

[4] P. S. Mann and D. Kumar, "A Reactive Defense Mechanism based on an Analytical Approach to Mitigate DDoS Attacks and Improve Network Performance," International Journal of Computer Applications, vol. 12-No.12, pp. 43-46, January 2011.

[5] S. T. a. K. Levitt, "Detecting spoofed packets," in Proceedings of The Third DARPA Information Survivability Conference and Exposition (DISCEX III) '2003, Washington, D.C., 2003.

[6] W. Haining, et al., "Defense Against Spoofed IP Traffic Using Hop-Count Filtering," Networking, IEEE/ACM Transactions on, vol. 15, pp. 40-53, 2007.

[7] I. B. Mopari, et al., "Detection and defense against DDoS attack with IP spoofing," in Computing, Communication and Networking, 2008. ICCCn 2008. International Conference on, 2008, pp. 1-5.

[8] N. Venkatesu, et al., "An Effective Defense Against Distributed Denial of Service in GRID," in Emerging Trends in Engineering and Technology, 2008. ICETET '08. First International Conference on, 2008, pp. 373-378.

◈◈◈