2009 IEEE International Advance Computing Conference (IACC 2009)
Patiala, India, 6–7 March 2009

Mitigating DDoS attack and Saving Computational Time using a Probabilistic approach and HCF method

Biswa Ranjan Swain
Department of Computer Science & Engg.
National Institute of Technology
Rourkela, Orissa, India
E-mail: swain.biswaranjan@gmail.com

Bibhudatta Sahoo
Department of Computer Science & Engg.
National Institute of Technology
Rourkela, Orissa, India
E-mail: bdsahu@nitrkl.ac.in

Abstract—**This paper discusses the mitigation of Distributed Denial of Service (DDoS) attack and as well preservation of computational time on wireless network. The DDoS effects upon the QoS in the loss of bandwidth and the resources available at the server. The uncertainty of Distributed denial of Service attack can be best simulated with the help of probabilistic model. The simple Hop Count method was used to calculate the hop count from the Time-To-Live (TTL) field of each packet. In this approach calculation of hop count for each packet is not required to detect the malicious packet. The number of packet we need to examine depends upon the probabilistic approach. This method mitigates the DDoS attack by reducing computational time and memory during the processing of a packet.**

Keywords- HCF Method, DDoS Attack, Time-To-Live

I. INTRODUCTION

Today the installation of wired medium is reasonable difficult than the wireless medium. And also the reliability in wireless medium is high than wired medium. Mobility in wireless medium is outstanding. In wireless medium we know the bandwidth is very limited so simple DDoS attack is also possible. Here in this type of medium the attacker needs to overflow the buffer at the server side to make it unable to provide service[3]. By this DDoS attack the processing performance of the server is degraded. So many mechanisms had been developed to mitigate this but many of those were not applied at the end system side. Our mechanism is applied at the server side. Distributed Denial of Service (DDoS) attacks are virulent, relatively new type of attack on the availability of Internet services and resources[1]. DDoS attackers infiltrate large numbers of computers by exploiting software vulnerabilities, to set up DDoS attack networks. These unwitting computers are then invoked to wage a coordinated, large scale attack against one or more victim systems. As specific countermeasures are developed, attackers enhance existing DDoS attack tools, deriving new techniques[2]. So it would be desirable to develop comprehensive DDoS solutions that defend against known & future DDoS attack variants. As detecting the attack is an uncertainty problem here so we have developed a probabilistic approach to detect the attack.

A mathematical mode of designing a system provides an opportunity for representing the system and also getting the performance report successively as this system is an uncertain system.

This paper is organized into following six sections. Section II describes the related work done till now. Section III describes the method of Hop Count by TTL. Section IV describes our probabilistic approach to find the number of packets being malicious among a massive number of packets. Section V describes how to mitigate the attack using our probabilistic approach with HCF method, however we have not simulated through any simulation environment.

II. RELATED WORK

Researchers have used the distribution of TTL values seen at servers to detect abnormal spikes due to DDoS traffic[4]. However we have not come across any scheme which takes the probabilistic approach and the hop count method to mitigate the DDoS attack. In this section we will reflect some lessons regarding the mitigation of this attack.

Several filtering solution which must execute on IP routers, have been proposed to prevent spoofed IP packets from reaching intended victims. The most straightforward scheme is ingress filtering, which blocks spoofed packets at edge routers, where address ownership is relatively unambiguous, and traffic load is low[5]. However, the success of ingress filtering hinges on its wide-deployment in IP routers. Most ISPs are reluctant to implement this service due to administrative overhead and lack of immediate benefit to their customers.

In the existing Hop Count Filtering mechanism, they have checked for each packet to calculate the hop count so that they can be sure about the maliciousness of the packet[4] and can accept or discard the packet. They look for all the packets reach at the server.

III. TTL BASED HOP COUNT FILTERING METHOD

A. The Method

The end-system approach protects Internet servers with sophisticated resource management to servers. This approach provides more accurate resource accounting, and fine-grained service isolation and differentiation. However, without a mechanism to detect spoofed traffic, spoofed packets will share the same resource principals and code paths as legitimate requests. While a resource manager can confine the scope of damage to the particular service under attack, it cannot sustain the availability of that service. In stark contrast, the server's ability to filter most, if not all, spoofed IP packets can help sustain service availability even under DDoS attacks. Since filtering spoofed IP packets is orthogonal to resource management, it can also be used in conjunction with advanced resource-management schemes.

Although an attacker can forge any field in the IP header, he cannot falsify 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 TTL field of the IP header, since each intermediate router decrements the TTL value by one before forwarding it to the next hop. The difference between the initial TTL (at the source) and the final TTL value (at the destination) is the hop-count between the source and the destination[4]. By examining the TTL field of each arriving packet, the destination can infer its initial TTL value, and hence the hop-count from the source. Here we assume that attackers cannot sabotage routers to alter TTL values of IP packets that traverse them.

Since hop-count information is not directly stored in the IP header, one has to compute it based on the TTL field of the IP header[4]. TTL is an 8-bit field in the IP header, originally introduced to specify the maximum lifetime of IP packets in the Internet. During transit, each intermediate router decrements the TTL value of an IP packet by one before forwarding it to the next-hop router. The final TTL value when a packet reaches its destination is therefore the initial TTL subtracted by the number of intermediate hops (or simply hop-count). The challenge in hop-count computation is that a destination only sees the final TTL. It would have been simple if all operating systems (OSs) use the same initial TTL, but in practice, there is no consensus on the initial TTL value. Furthermore, Since the OS for a given IP address may change at any time, we cannot assume a single static initial TTL value for each IP address.

B. *The Algorithm*

```
Algorithm 1: Boolean HCF(packet)
    for each packet:
        extract the final TTL T and IP address S;
        infer the initial TTL $T_0$;
        compute the hop count $H_c$=T-$T_0$;
        index S to get the stored hop count $H_s$;
        if($H_c \neq H_s$)
            packet is spoofed;
        else
            packet is legitimate;
```

Algorithm 1: Hop Count Filtering Algorithm

The inspection algorithm extracts the source IP address and the final TTL value from each IP packet. The algorithm infers the initial TTL value and subtracts it from the final TTL value to obtain the hop-count. Then, 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 classified as spoofed.

IV. A Probabilistic Approach To Find Out The Number Of Packet Being Malicious Among Massive Number Of Packets

The limited availability of radio spectrum is always the bottleneck in a wireless network[3]. Although transmission rates in wireless networks are much lower than those in wired networks, potential bandwidth depletion DDoS attacks are still feasible if attackers liberately coordinate a large population of mobile zombie devices to send out flooding traffic, which can easily consume all spectrum resources or at least significantly reduce the capacity of communication channels available to normal traffic.

In wireless bandwidth depletion DDoS attacks, attackers need to make the buffer overflow probability [3] close to **1** i.e. to make the total bandwidth consumption by all the traffic greater than the channel capacity.

So we have developed a probabilistic approach to know the no. of packets to be malicious.

Suppose the no. of packets arrive at the server with a poisson's distribution 'λ'.

Suppose each packet arrives at the server being malicious with probability 'p' or non-malicious with probability '1-p'.

So the joint probability of 'n' packets among the total traffic are malicious and 'm' packets among the traffic are non-malicious is as follows,

Let $N1$ denote the no. of malicious packet.

$N2$ denote the no. of non-malicious packet.

Also let N=$N1$+$N2$ be the total no. of traffic arrive at the server.

Now conditioning on N gives,

$$P\{N1=n,N2=m\}=\sum_{i=0}^{\infty}P\{N1=n,N2=m|N=i\}P\{N=i\}$$

Because $P\{N1=n,N2=m|N=i\}=0$ when $i=m+n$ the preceding equation yields that,

$$P\{N1=n,N2=m\} = P\{N1=n,N2=m|N=n+m\}(\lambda^{n+m}/n+m)$$

Probability of 'n' packets to be malicious is,

$$P\{N1=n,N2=m\} = ({}^{n+m}C_n\, p^n(1\text{-}p)^m)e^{-\lambda}(\lambda^{n+m}/n+m)$$

$$=((m+n)!/m!n!)\, p^n(1\text{-}p)^m e^{-\lambda}(\lambda^{n+m}/n+m)$$

$$=((m+n)!/m!n!)\, p^n(1\text{-}p)^m e^{-\lambda p}e^{-\lambda(1-p)}(\lambda^{n+m}/n+m)$$

$$=e^{-\lambda p}(\lambda p^n/n!)\, e^{-\lambda(1-p)}(\lambda(1\text{-}p)^m/m!)$$

Because the preceding joint probability mass function factors into two products, one of which depends upon 'n'.

So,

$$P\{N1=n\}=\sum_{m=0}^{\infty} P\{N1=n,N2=m\}$$
$$=e^{-\lambda p}(\lambda p^{n}/n!)$$

This derivation shows the probabilities of 'n' packets among the total packets with poisson's distribution of arrival rate 'λ' are malicious.

V. Mitigation Using Hop Count Method

After calculating the probability of some no. of packets being malicious we can filter out that many no. of packets from the given no. of packets. After checking each packet when we will reach at the desired no. of packets, we will let other packets to enter the server. The checking operation can be done using Hop Count method. This hop count can be done through the TTL field of the packet which can't be changed by any attacker. So that when this hop count of a packet will be differentiated from the desired no. of hop count we can say the packet is malicious. By applying this method we can say the overhead of processing and memory will decrease. But we have to compromise to some extent over here in comparison to simple HCF method.

```
Algorithm 2: proposed_algorithm
    for given 'λ', 'p', 'm+n'
    calculate 'n' such that P(n)=1;
    integer count=0;
    for each value of count upto 'n'
      for each packet 'i'
        if (count ≠ n)
          status= HCF(i);
        if status is 'spoofed'
            count++;
            drop the packet;
        else
            allow the packet;
        end if;
      end if;
      allow all the rest packets upto m+n
```

Algorithm 2: Proposed algorithm for mitigation of DDoS.

Our algorithm says, *for the given values of poisson's arrival rate of packets 'λ' ,error probability of each packet 'p', and a definite value of 'm+n', we can calculate the value of 'n' by specifying the probability of 'n' to 1. This value of 'n' can be calculated from the developed probabilistic equation.* Then we examine each and every packet reaching to the server to calculate the hop count and to identify the maliciousness. Whenever we find one malicious packet we are increasing the count value so that the value of count will go maximum upto 'n' whenever the value of count will reach 'n', we will simply allow other packets upto 'm+n' without checking them.

Conclusion

In simple HCF we were checking each and every packet and let them enter to the server on the confirmation of their non-maliciousness. But in our approach we calculate the probability of number of packets being malicious and according to that we are mitigating the attack. In HCF 90% of erroneous packets were dropped[4] but in our case it may happen that 80% to 85% of erroneous packet will be dropped. In HCF the computation and memory overhead is there but, in our approach some how the overhead is less as we are not checking all of the packets. We have not yet simulated this approach in any simulation environment.

References

[1] Douligeris, C. Mitrokotsa, A. *"DDoS Attacks and Defense Mechanisms: A Classification"*, Proceedings of 3rd IEEE International Symposium, 14-17 Dec 2003, pp- 190-193.

[2] Spetch, Stephen M. Lee, Ruby B. *"Distributed Denial of Service: Taxonomies of Attacks, Tools, and Countermeasures"*, Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems, 2004 International Workshop on Security in Parallel and Distributed Systems, September 2004, pp. 543-550.

[3] Huang, Q. Kobayashi, H. Liu, B. *"Modeling of Distributed Denial of Service Attacks in Wireless Networks"*, Computers and signal Processing, 2003, 28-30 Aug. 2003, pp- 41- 44.

[4] Jin, C. Wang, H. Shin, Kang G. *"Hop-Count Filtering: An Effective Defense Against Spoofed Traffic"*, IEEE Transactions on Dependable and Secure Computing, 2004.

[5] Lipson, H.F. *"Tracking and tracing Cyber-Attacks: Technical Challenges & Global Policy Issues"*, CMU/SEI-2002-SR-009, November 2002.