DDoS Attack Mitigation Using Random and Flow-Based Scheme

Bansidhar Joshi, Bineet Joshi and Kritika Rani

1 Introduction

Distributed denial-of-service attacks are a critical threat to the internet. DDoS attackers generate a huge amount of requests to victims through compromised computers (zombies), with the aim of denying normal service or degrading of the quality of services [1]. Internet was primarily designed to facilitate communication and was not designed in a way to give security to such communications. Hence, the network has a lot of scope of being attacked. DoS attacks are growing at a rapid speed, and they are becoming distributed and highly sophisticated. The astounding fact to know is that a DDoS attack can be launched at a minimum of $20. With the help of botnet, DDoS attack can be implemented by sending a few packets by each compromised system. It, hence, becomes difficult to differentiate between legitimate and illegitimate traffic. There are many schemes to tackle these attacks like DPM, PPM, FDPM, and information theory. Several detection schemes exist. Like a technique which prioritizes packets based on a score which estimates its legitimacy given the attribute values it carries and based on this score, selective packet discarding is carried out [2].

B. Joshi (✉) · K. Rani
Jaypee Institute of Information Technology, Noida, India
e-mail: bansidhar.joshi@jiit.ac.in

K. Rani
e-mail: kritika.rani17@gmail.com

B. Joshi
Swami Rama Himalayan University, Dehradun, India
e-mail: bineetjoshi@gmail.com

R. K. Shukla et al. (eds.), *Data, Engineering and Applications*,
https://doi.org/10.1007/978-981-13-6351-1_11

Another method was the one in which traffic is analyzed only at the edge routers of an Internet service provider (ISP) network [3, 4]. This framework is able to detect any source-address-spoofed DDoS attack, no matter whether it is a low-volume attack or a high-volume attack. We will be discussing about information theory and linear packet marking technique in this paper and its advantages over conventional DPM and PPM. We will also talk about some of the techniques from which we can prevent our system to be a part of the botnet. Entropy can measure the variations of randomness of flows on a given local router [5]. Based on such variations, a DDoS attack can be detected. If the difference between the entropy and mean is greater than a threshold value, then there is an attack.

2 Related Work

2.1 Node Append

This is the simplest method of marking to trace the attacker. In this method, we continue appending the information about the router in the header of the packet. The disadvantage is the overhead on the packet header and insufficient space to accommodate so many routers [5].

2.2 Node Sampling

This was introduced to solve the problem of storage overhead. The packets are marked depending on the probability chosen at random [5]. Either 0 or 1, can be chosen, so the probability is 0.5. A node once marked is not marked again. The problem lies when it becomes difficult to know which node will mark the packet and a high number of packets are required to trace back the attack.

2.3 Edge Sampling

Authors sent the packet with some probability p and overcome the shortcoming in node sampling about the distance from the destination by introducing d. When the packet is marked, it is updated to 0 and is incrementally increased with every decrease in ttl [5]. Once a packet is marked, it is not marked again. $p(1-p) \wedge (d-1)$ where d is the number of hops away from the destination.

Let x be the number of packets then

$$E(X) = ln(d)/p(1 - p) \wedge (d - 1) \tag{1}$$

The disadvantage of this technique is that we need many packets to trace the route and in case of multiple attackers, this technique is not that robust.

2.4 Probabilistic Packet Marking

This is basically the extension of edge sampling. There is not enough space in the packet header to store the 32-bit address of the router. The value of the router in the fields is fragmented so that it is less used by the router. It can even be applied after or during the attack with no additional router storage problems [6] as in case of logging.

2.5 Deterministic Packet Marking

This is almost similar to PPM but as in PPM, the authors mark the packets based on a probability, and in deterministic packet marking, we mark all the packets passing through the network [2]. Using IP address of the router, packets are marked. The problem occurs in case of IP spoofing, where the attackers can spoof their IP address making it difficult to reach the correct router.

2.6 Linear Packet Marking

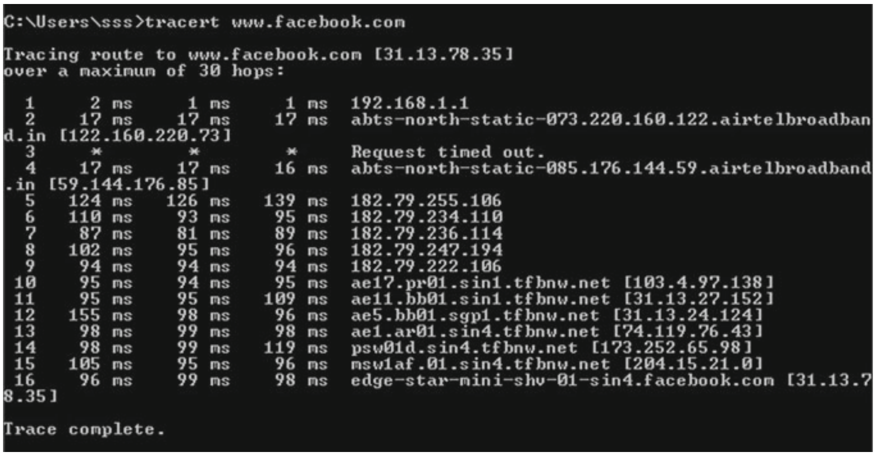
This scheme needs the number of packets almost equal to the hops traversed by the packet [3]. In this trace back scheme, the ID field value is divided by the number of hops. The remainder, thus, obtained is the value of the hop that will mark the packets. Like all packet marking schemes, it also needs an attack to be alive and trace backing to the source itself creates a DoS while performing the attack. In this scheme, the authors use TTL value to decide the hop count and the value in the ID field at the router to decide whether to mark the packet or not.

2.7 *Entropy Variation*

Entropy can measure the variations of randomness of flows on a given local router [7]. Based on such variations, a DDoS attack can be detected. In case of an attack, the actual source can be found by submitting the requests to upstream routers and finally reaching the source. The advantage lies in the fact that it does not require the attack to be alive, can differentiate between legitimate and illegitimate users efficiently, and does not require any marking. The problem in this technique comes in case of trace back when a request is made to all upstream routers in order to identify the source.

3 Analysis of a Random and Flow-Based Model

We found out that the maximum hops that a packet travels before reaching the destination is 25. A simulation is done using the tracert command and the same is shown in Figs. 1 and 2. The algorithm that we are using for detecting whether a DoS or a DDoS is happening or is it a surge in the legitimate traffic is explained in Algorithm 1, and the algorithm we are using for marking the packet for successful IP trace back is explained in Algorithm 2.



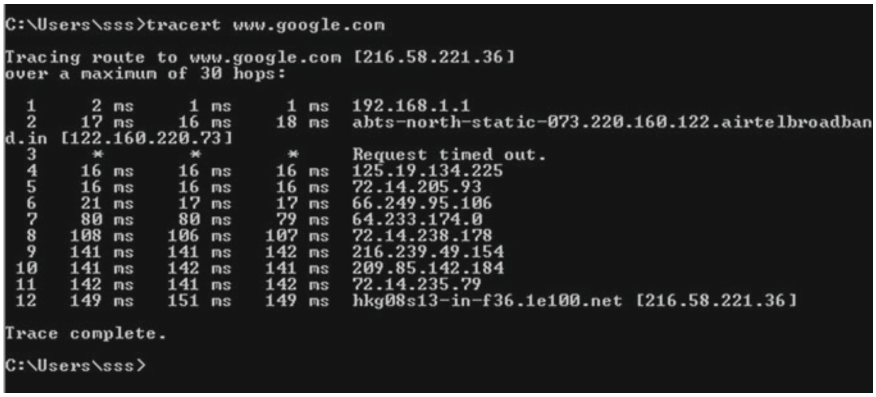
```
C:\Users\sss>tracert www.facebook.com

Tracing route to www.facebook.com [31.13.78.35]
over a maximum of 30 hops:

  1     2 ms     1 ms     1 ms  192.168.1.1
  2    17 ms    17 ms    17 ms  abts-north-static-073.220.160.122.airtelbroadban
d.in [122.160.220.73]
  3     *        *        *     Request timed out.
  4    17 ms    17 ms    16 ms  abts-north-static-085.176.144.59.airtelbroadband
.in [59.144.176.85]
  5   124 ms   126 ms   139 ms  182.79.255.106
  6   110 ms    93 ms    95 ms  182.79.234.110
  7    87 ms    81 ms    89 ms  182.79.236.114
  8   102 ms    95 ms    96 ms  182.79.247.194
  9    94 ms    94 ms    94 ms  182.79.222.106
 10    95 ms    94 ms    95 ms  ae17.pr01.sin1.tfbnw.net [103.4.97.138]
 11    95 ms    95 ms   109 ms  ae11.bb01.sin1.tfbnw.net [31.13.27.152]
 12   155 ms    98 ms    96 ms  ae5.bb01.sgp1.tfbnw.net [31.13.24.124]
 13    98 ms    99 ms    98 ms  ae1.ar01.sin4.tfbnw.net [74.119.76.43]
 14    98 ms    99 ms   119 ms  psw01d.sin4.tfbnw.net [173.252.65.98]
 15   105 ms    95 ms    96 ms  msw1af.01.sin4.tfbnw.net [204.15.21.0]
 16    96 ms    99 ms    98 ms  edge-star-mini-shv-01-sin4.facebook.com [31.13.7
8.35]

Trace complete.
```

Fig. 1 Tracing the route of Facebook.com



```
C:\Users\sss>tracert www.google.com

Tracing route to www.google.com [216.58.221.36]
over a maximum of 30 hops:

  1     2 ms     1 ms     1 ms  192.168.1.1
  2    17 ms    16 ms    18 ms  abts-north-static-073.220.160.122.airtelbroadban
d.in [122.160.220.73]
  3     *        *        *     Request timed out.
  4    16 ms    16 ms    16 ms  125.19.134.225
  5    16 ms    16 ms    16 ms  72.14.205.93
  6    21 ms    17 ms    17 ms  66.249.95.106
  7    80 ms    80 ms    79 ms  64.233.174.0
  8   108 ms   106 ms   107 ms  72.14.238.178
  9   141 ms   141 ms   142 ms  216.239.49.154
 10   141 ms   142 ms   141 ms  209.85.142.184
 11   142 ms   141 ms   142 ms  72.14.235.79
 12   149 ms   151 ms   149 ms  hkg08s13-in-f36.1e100.net [216.58.221.36]

Trace complete.

C:\Users\sss>
```

Fig. 2 Tracing the route of Google

3.1 Algorithm Used for This Work Are

Algorithm 1: To determine the threshold time after which we monitor if there is an attack.

Start from 1 time unit after which we will check for an attack and increase the time exponentially. The next time interval after which we monitor the attack will be 2 units. So the time intervals are increasing like this 1, 2, 4…

A threshold value is formed based on past experiment and trends keeping in mind (based on different kinds of networks for different purposes like in an airline website, No. of visits during festivals or holidays increases to many folds):

Different months
Days
Seasons (summer, winter, spring, rainy, etc.) Festivals (can be included in months)
Weekly

On reaching this threshold value, we start increasing the time interval additively instead of exponentially, i.e., now, time interval increases by 1 time unit…

$$k, k + 1, k + 2 \ldots$$

If there as an attack at any point, mitigate it, reduce the threshold value to half of the time interval, and reduce the time to 1 unit again and repeat the process.

Algorithm 2: To detect an attack due to high number of packets being sent by a few attackers, local router is the router taken into consideration. Upstream routers are the routers just above the local router. A flow is defined as fij <ui, dj, t>: ui is an upstream router, dj is the destination address, and t is the current time.

The algorithm is as follows [8]:

1. Initialize C0, δ0 with some experimental value.
2. Identify the flows f1, f2, f3 … fn and set count number of each flow to zero, x1 = 0, x2 = 0, x3 = 0 …, xn = 0 (count number is basically the number of times a particular flow (having same upstream router and timestamp) occurs in a given time).
3. When time interval used to monitor the traffic (as described in the above algorithm) is over, calculate the probability distribution and entropy variation as follows:

$$pj = xj * (\sum_{j=1}^{n} x) - 1 \tag{2}$$

$$h = -\sum_{j=1}^{n} pj \log(pj) \tag{3}$$

4. Save x1, x2 … xn and H(F).
5. If there is no dramatic change of the entropy variation H(F), |H(F)-C| <= δ update mean and the standard variation is

$$C[t] = \sum_{j=1}^{n} \alpha j * C[t-j], \qquad \sum_{j=1}^{n} \alpha j = 1, \tag{4}$$

$$\delta[t] = \sum_{j=1}^{n} \beta j * \delta[t-j], \qquad \sum_{j=1}^{n} \beta j = 1, \tag{5}$$

6. Go to step 2.
7. If |H(F)-C| > δ, then the attack is possible.
8. Now if pi * log(pi/qj) > 0.2 (where qj is another flow), then it is a legitimate traffic. Otherwise, it is an attack.

Versi	H.	TOS(8)	Total length(16)	
Identification(16)			Flags	Fragmentation Offset(12)
Time to Live(8)		Protocol (8)	Header Checksum(8)	
32-bit destination address				

Fig. 3 IPv4 header

Algorithm 3: We are implementing the algorithm based on the assumption that a packet on an average takes only 25 hops to reach the destination [6]. The steps of marking are as follows: on the arrival of a packet, we will apply our packet marking algorithm. Then, based on the algorithm, we will see to mark the packet or leave it unmarked and finally forward it on the network [2]. We use ID field of the ipv4 to store the unique identification number of the router, as well as the maximum hop distance from the destination. A diagram of the complete header is given in Fig. 3. Distance is calculated using the ttl value of the packet that tells the maximum time the packet can stay on the network.

```
1.  If packet is not marked
2.      Select random number'x' between 1 and 25
3.      Dist away= Dist – ttl
4.      If((packet ID % x) + 1 == identification number of a router)
5.          Insert router address into packet header
6.          Set MRN = 0
7.      Endif
8.  Else
9.      Increment MRN
10. End if
```

We simulated our detection Algorithm 1 using Wireshark. We edited our files by converting them into text format using editcap, modifying the header using tcprewrite, used multithreading to make the capturing and detection modules run together, and tcpreplay to resend the packets. Combining all such concepts in C programming on a Linux platform, we were able to implement our algorithm. The calculations are shown below. The detailed algorithm is explained in Fig. 4.

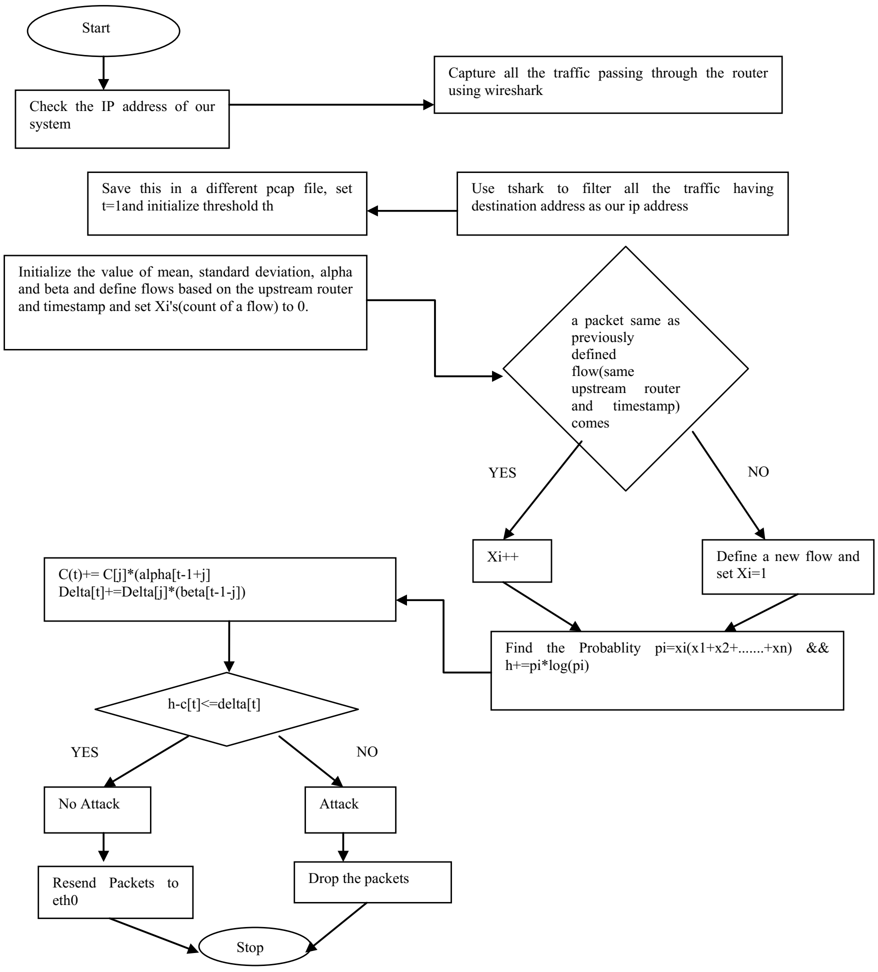

Start
Check the IP address of our system
Capture all the traffic passing through the router using wireshark
Use tshark to filter all the traffic having destination address as our ip address
Save this in a different pcap file, set t=1and initialize threshold th
Initialize the value of mean, standard deviation, alpha and beta and define flows based on the upstream router and timestamp and set Xi's(count of a flow) to 0.
a packet same as previously defined flow(same upstream router and timestamp) comes
YES
NO
Xi++
Define a new flow and set Xi=1
Find the Probablity pi=xi(x1+x2+.......+xn) && h+=pi*log(pi)
C(t)+= C[j]*(alpha[t-1+j]
Delta[t]+=Delta[j]*(beta[t-1-j])
h-c[t]<=delta[t]
YES
NO
No Attack
Attack
Resend Packets to eth0
Drop the packets
Stop

Fig. 4 Flow diagram

The simulation of Algorithm 2 is shown using ns2. The changes are made in the header of the trace file. The data needed is saved in .tr file and is extracted for analyzing using the awk script on the desired columns.

Case 1

```
X1 = 5          X2 = 4   X3 = 6   X4 = 4
P1 = 5/21                P2 = 6/21                P3 = 6/21
                P4 = 4/21
H+=Pi * log(Pi)
P1 * log(P1) = -0.148    P2*log(P2) = -0.155
P3*log(P3) = -0.155        P4*log(P4) = -0.137
H = -0.595       c[0] = 0.2         delta[0] = -0.4
H - c[t] <= delta[t]
H – c[0] <= delta[0]
-0.795 <= -0.4   true
```

Therefore, no attack.

Case 2

```
X1 = 100        X2 = 4            X3 = 2
P1 = 0.943      P2 = 0.037       P3 = 0.018
H+=Pi * log(Pi)
H = -0.1083      c[0] = 0.2         delta[0] = -0.4
H - c[t] <= delta[t]
H – c[0] <= delta[0]
-0.308 <= -0.4   false
```

Therefore, attack.

4 Conclusion and Future Work

4.1 Conclusion

The conclusions drawn from the paper are that DoS and DDoS attack are a great threat to the internet. Packet marking techniques are costly to implement and are difficult to predict at an early stage. Moreover, we need the attack to be alive to trace back to the origin of the attack. The entropy variation technique is thus a relatively new technique for the detection of DDoS and DoS attack and there is a steep learning curve for the whole organization, for both users and administrators. Packet marking techniques, on the other hand, are more mature, well documented, and easy to understand.

For the people who are not familiar with the technology used for information theory should adopt to packet marking techniques for the reason stated above and wait for the information theory to become more mature.

4.2 Future Work

In future, we would like to implement these techniques on the public and private cloud setup and compare the results of our algorithm with the results of the existing techniques. We would also check the scalability and the fault tolerance of the algorithm in both the types of cloud that is public as well as private cloud.

References

1. Sagar. A., Joshi B. K., Mathur, N.: A study of distributed denial of service attack in cloud computing (DDoS). In: Edition on Cloud and Distributed Computing: Advances and Applications, vol. 2 (2013)
2. Belenky, A., Ansari, N.: IP traceback with deterministic packet marking. Commun. Lett. IEEE **7**(4), 162–164 (2003)
3. Saurabh, S., Sairam, A.S.: Linear and Remainder: Packet Marking for Fast IP TraceBack. IEEE. 978-1-4673-0298-2/12/ (2012)
4. Joshi, B., Joshi, B., Rani, K.: Mitigating data segregation and privacy issues in cloud computing. In: Proceedings of International Conference on Communication and Networks: ComNet 2016, vol. 508, pp. 175. Springer (2017)
5. Savage, S., Wetherall, D., Karlin, A., Anderson, T.: Network support for IP traceback. IEEE Trans. Netw. **9**(3) (2001)
6. Goodrich, M.T.: Probabilistic packet marking for large-scale IP traceback. IEEE/ACM Trans. Netw. **16**(1), 15–24 (2008)
7. Yu, S., Zhou, W., Doss, R.: Information theory based detection against network. behavior mimicking DDoS attacks. IEEE Commun. Lett. **12**(4) (2008)
8. David, J., Thomas, C.: DDoS attack detection using fast entropy approach on flow-based network traffic. Procedia Comput. Sci. **5**, 30–36 (2015)