

Improving the Efficiency of IP Traceback at the DoS Victim

Siddharth Ramesh, Swaminathan Pichumani, Venkat Chakravarthy

School of Computing, University of Utah,

Salt Lake City, UT 84112

{sramesh, swami, vchakra}@cs.utah.edu

1 Introduction

Denial of Service (DoS) attacks on Internet hosts and network components have been increasing in both frequency and enormity in the past few years causing great monetary damage to the attacked sites, as well as degradation in the browsing quality of Internet users. DoS attacks and its more damaging variant Distributed DoS (DDoS) attacks are also increasingly easy to cause with a host of tools like TFN, TFN2K and Trinoo, publicly available for everyone to download and use. The disturbing part is that the current IP protocol as is, provides little help in identifying the source of such attacks. To make things worse, it helps the attacker mask his real identity using what is known as IP Spoofing, wherein the source address of a packet is changed by the sender to a different one. Since there is no authentication of the IP source addresses at the IP layer, anyone can hide his/her identity behind such spoofed packets.

DoS attacks can be classified into two types - those which use software and protocol bugs/limitations to overcome the victim machine and those which rely on scale of attack to overwhelm the victim. Attacks such as the *Ping of Death*, the *Teardrop* and *buffer-overflow* attacks belong to the first category, while attacks such as *TCP-SYN flood* and *Smurf* attacks are examples of the latter. While patches are available for most of the identified software bugs, attacks which rely on scale have not found any workable solution. And as mentioned earlier, the anonymous nature of IP allows the attackers to get away scotfree thus making law enforcement also tougher.

IP Traceback refers to mechanisms which allow a victim to trace back an IP packet, through the series of routers it traversed, back to the source of the attack. While there have been many IP Traceback solutions proposed ([3, 4, 5]), each of them has their own drawbacks in deployability. But it is almost certain that all IP Traceback mechanisms would need, in some way or the other, help from the underlying network. The strict end-to-end principle for applications, which was one of the major designs on which the Internet was built, will have to be changed in some way.

In this work, we propose a modification to the IP Marking approach suggested in [3] to make reconstruction of the attack path much more efficient. Also, [3] talks about reconstruction of multiple paths, but the algorithm given in the paper reconstructs only a single path of attack. We have modified the algorithm and

changed the encoding scheme to be able to reconstruct multiple paths more efficiently. In addition to the above changes, instead of having a constant marking probability at each router, we introduce the concept of a variable marking probability scheme based on the distance of the router from the victim. This helps in further reducing the convergence time by a considerable amount.

The rest of the paper details our approach as follows. Section 2 reviews the related work done in the field of IP Traceback. Section 3 describes the approach taken by Savage *et.al.* in [3] and identifies the shortcomings in it. We describe our design changes and improvements in Section 4. Section 5 describes our experimental setup. Section 6 provides simulation results evaluating our approach. Finally we conclude in Section 7 with a brief look at future work.

2 Related Work

There have been many proposals in IP traceback. IP Traceback solutions can be classified as *Single packet IP traceback* and *Multiple packets based IP traceback*. As the names suggest, the former allows IP traceback with just a single IP packet which has traversed through the network, while the latter requires a large number of packets to be collected.

One of the foremost schemes in IP traceback was proposed by Savage *et.al* [3]. Their IP Marking approach used in-band signaling using the identifier field in the IP packet itself. Each router marked, with a low probability, packets which passed through it with information about the link edge it was connected to. When the victim was being attacked by a large number of packets, it collected sufficient number of packets to be able to reconstruct the path of the attack. A similar scheme proposed by Bellovin used out-of-band signaling, where the router, with a low probability, sent an ICMP packet to the destination with audit information. As with the previous scheme, the victim reconstructed the path after collecting sufficient number of ICMP packets belonging to a flow.

Snoeren *et.al.* [4] proposed a single packet based traceback mechanism where each router maintained logs on each packet which passed through it. When a packet had to be traced, appropriate extraction techniques were used to recover information from the router logs. However the space occupied by such logs grew very fast, and hence the routers had to keep transferring their logs to larger databases at regular intervals. Snoeren's scheme required a huge network infrastructure for logging, storing and retrieving information.

Schnackenberg *et.al.* [1] proposed the Intruder Detection and Isolation Protocol (IDIP) which required routers involved in the traceback effort to communicate with each other. To support this protocol, various solutions like hardware support and overlay networks had been suggested. But these schemes rely on the longevity of the attack before re-routing can take effect.

3 Background

In [3], Savage *et.al.* describe a traceback mechanism based on probabilistic marking of IP packets at routers. Their approach allows a victim to identify the network path(s) traversed by attack traffic without any support from the Internet Service Providers (ISPs). Also, their traceback solution can be performed only after the attack is completed. Their proposal consists of two algorithms - the Node Sampling and Edge Sampling algorithm. The Node Sampling algorithm has the shortcomings that

- the router order determination is slow and requires a large number of packets
- it fails in the DDoS scenario where multiple paths have to be traced

The Edge Sampling algorithm overcomes the above shortcomings, but requires enormous computation at the victim for reconstruction.

3.1 Edge Sampling Algorithm

In the Edge Sampling algorithm, every router decides to mark a packet with a certain probability p . The marking procedure involves the router writing its IP address into a *start* field and writing a zero in a *distance* field. However, if a router does not decide to mark, it checks the *distance* field. If it is zero, it writes its IP address into an *end* field and increments the *distance* field. And if the *distance* field is not zero, the router just increments that field. When the packet arrives at the victim, the *start* and *end* field contain the routers at the distance contained in the *distance* field. After getting sufficient number of packets, the victim node will know the routers at all distances to the source. Also, it is to be noted that since each router marks with a small probability p irrespective of whether a packet has already been marked or not, there is a higher probability of a receiving a marked packet from a nearer router than a farther one.

3.2 Encoding

The Edge Sampling scheme detailed above requires 72 bits of space in each IP packet for storing two IP addresses (corresponding to the *start* and *end* fields) and one *distance* field. Since the space available in an IP packet is limited, the authors suggest an encoding scheme which dramatically reduces the space required in return for an increase in convergence time and a reduction in robustness to multiple attacks.

The scheme proposed is called Compressed Edge Sampling Algorithm. Each edge is represented by the *exclusive-or* (XOR) of the two IP addresses making up the edge. Any router which decides to mark a packet writes its IP address in an *edge* field and sets the *distance* field to zero. Any router which decides not to mark the packet, checks the *distance* field. If it is zero, it XORs its IP address with the value in the *edge* field and increments the *distance* count by one. If it is not zero, the router just increments the *distance* field.

The above scheme still needs 24 bits of space. To further reduce this, the authors propose that the 32 bit IP address be bit-interleaved with a 32-bit hash of it. This 64 bit quantity is split into many fragments. When

a router decides to mark a packet, it randomly chooses a fragment offset and writes that fragment into the *edge* field and the offset into an *offset* field. If the neighboring router does not decide to mark the packet, it XORs its own fragment at the same offset. The victim reconstructs complete edge ids by XORing the collected fragments in the increasing order of the *distance* field, concatenates them, and uses the 32-bit hash to validate the concatenated result.

Finally 16 bits of space are required by the above encoding scheme, *viz.* 8 bits for *edge fragment*, 3 bits for *fragment offset* and 5 bits for *distance*. The authors have overridden the 16-bit *identification* field of the IP header for this purpose.

3.3 Limitations of the Compressed Edge Sampling Algorithm

The Compressed Edge Sampling algorithm described above has the following limitations

1. IP Fragmentation

The IP identification field is used for fragmentation and reassembly at the IP layer. All fragments of an IP packet should have the same *identification* field for the destination to be able to reassemble the packet. Since this field is used for marking, there are two problems which can arise - a marked packet can be fragmented downstream or a fragment of an IP packet can be marked by an upstream router. Both these scenarios make it impossible for the destination to reassemble the packet. Hence, fragmentation fails.

2. Failure to identify multiple attack paths

Eventhough edge-ids provide enough information to differentiate between multiple paths, the algorithm provided in [3] allows reconstruction of only a single attack path.

3. Increase in time of convergence

The encoding scheme used in the algorithm fragments each edge-id into eight, 8-bit fragments. Hence, if there are k nodes at a distance d , then there should be a total of $8 * k$ fragments from distance d collected at the victim before reconstruction can begin. This results in an increase in the time of convergence.

4. Exponential increase in reconstruction time at the victim

There are eight fragments from each edge-id at each distance d . If there were k nodes at distance d , then corresponding to each of the eight offsets from nodes at distance d , there would be k fragments. This would result in k^8 possible IP addresses which will have to be computed and validated. This is the primary reason why the reconstruction algorithm takes a long time to build the attack-tree in the face of multiple attacks.

4 Design

We have

1. Changed the design of the encoding mechanism

2. Modified the reconstruction algorithm used in [3], and
3. Added a Variable Marking Probability mechanism,

so as to achieve three main goals :

1. Reduce the time of convergence
2. Reduce the complexity of reconstruction at the victim
3. Identify multiple attack paths as in a DDoS attack

4.1 Changes in Encoding

Savage *et.al.* divide the 64-bit edge-id into eight fragments of 8-bits each. Hence, as mentioned in the previous section, the total space used to store the fragment information is 16-bits. They use the 16-bit *identification* field of the IP header to serve this purpose. We propose that the entire 32 bits comprising of *identification* field, *fragmentation flags* and *fragment offset* field be used for encoding. The advantages of using more bits for encoding are reduction in both convergence time and reconstruction complexity at the victim. And the disadvantage of losing the flexibility of IP fragmentation still remains as in Savage's scheme. But [6] says that less than 0.25% of all IP packets are fragmented. Hence this is not a serious issue.

In our encoding scheme, we use the 32-bit IP address concatenated with its 32-bit hash as the basis of an *edge-id*. This 64-bit value is divided into three fragments of 24-bits each, as shown in Figure 1.

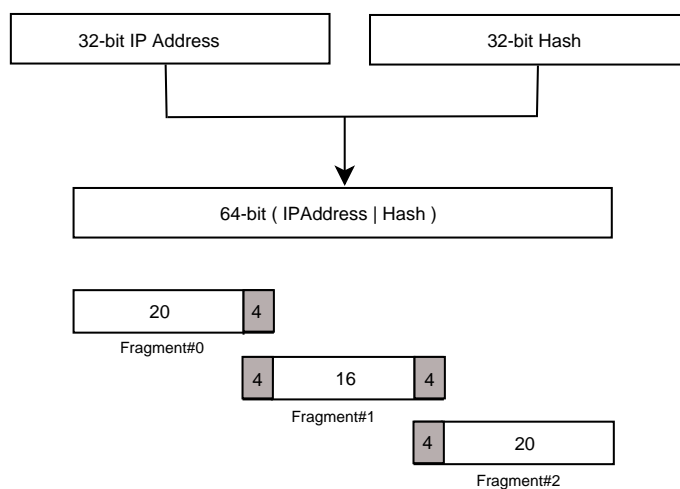


Figure 1: Encoding format

Let IP_m refer to the 32-bit IP address of node m and $h(IP_m)$ be its 32-bit hash. We define $f_{m,j}^d$ to be the j^{th} fragment from node m which is at a distance d from the victim. So,

$$f_{mj}^d = [IP_m | h(IP_m)]_j$$

When a router m , at distance d , decides to mark a packet, it writes f_{mj}^d as the 24-bit value into the *edge-id* field, j into the *offset field* and 0 into the *distance field*. If the next router n does not mark the packet, then it would write $f_{mj}^d \oplus f_{nj}^{d-1}$ into the *edge-id* field and increment *distance* by one. Other routers which do not mark, just increment the *distance* field. Figure 2 shows the encoding of the above three fields in the IP header.

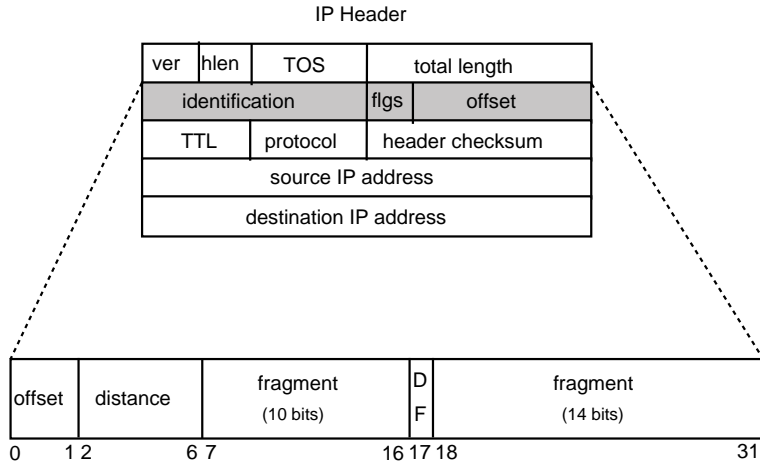


Figure 2: IP Header Encoding

4.2 Modification in the reconstruction algorithm

The reconstruction algorithm is as given below. This allows reconstruction of the entire attack tree even in a DDoS scenario. Our reconstruction algorithm uses all nodes at a distance $n - 1$ when computing possible attack paths through nodes at distance n . This case has not been handled in the algorithm provided in [3].

Path reconstruction algorithm at the victim v :

```

procedure reconstruct
{
  Let B be a three-level hierarchy of buckets indexed by
  distance at the first level, offset at the second and
  fragment at the final level
  Let G be a tree rooted at v
  Let max_bucket := 0
  Let parent[] be a list which stores IP addresses

  recv:

```

```

for each packet w which v receives
  insert_into_bucket(B, w.dist, w.offset, w.frag)
  if (w.dist > max_bucket) then
    max_bucket := w.dist

if(! reconstructable(B)) then
  goto recv

old_parent[] := combine_fragments0( B[0][][ ] )
for each IP address A in old_parent[]
  insert_to_tree(G, parent=G.root, node=A)

for d := 1 to max_bucket
  new_parent[] := combine_fragments( B[d][][ ], old_parent[], G )
  old_parent[] := new_parent[]

return G
}

procedure combine_fragments0( B[0][][ ] )
{
  Let node[] be a list of IP addresses
  for each set of 3 fragments (f0,f1,f2) in B[0][][ ]
    if(bit_overlap(f0, f1, f2)) then
      Let ip_hash := concatenate(f0,f1,f2)
      if (hash_match(ip_hash)) then
        add IP(ip_hash) to node[]

  return node[]
}

procedure combine_fragments( B[d][][ ], old_parent[], G )
{
  Let node[], new_parent[] be lists of IP addresses
  for each P in old_parent[]
    B[d][i=0,1,2][ ] := B[d][i][ ] XOR Pi
    node[] := combine_fragments0( B[d][][ ] )
    for each N in node[]
      insert_into_tree(G, parent=P, node=N)
      add N to new_parent[]
    B[d][i=0,1,2][ ] := B[d][i][ ] XOR Pi

  return new_parent[]
}

```

4.3 Variable Marking Probability (VMP) Scheme

As is evident from the analysis in [3], if all routers mark packets with the same marking probability p , the probability of receiving a marked packet at the victim from a router at a distance d is $p(1 - p)^{(d-1)}$. As can be seen from Figure 3, this probability decreases to an extremely low value for routers at large distances. Because of this imbalance, the victim would have to wait for a long time before it receives packets from all distances and can start reconstruction. To counter this imbalance, each router would have to mark a packet

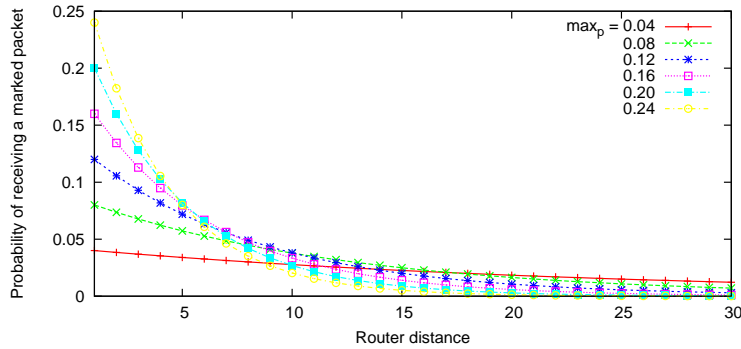


Figure 3: Probability of receiving a marked packet from varying distances for different values of p

with a probability so as to ensure that packets are received from all routers with approximately the same probability irrespective of their distances from the victim. We propose a scheme to do this in an attempt to determine this optimal marking probability function.

In our scheme, each router marks a packet with a probability which is a function of the TTL of the packet. We decided to use an exponential probability function. Let $CurTTL$ be the TTL of the packet at a marking router, and $MaxTTL$ represent the maximum TTL value of an IP packet¹. Let max_p denote the maximum probability with which any router can possibly mark a packet. We use the marking probability formula

$$p = \left(\frac{CurTTL}{MaxTTL} \right)^k * max_p$$

where k is the exponentiating factor.

Figure 4 shows the probability of receiving marked packets from routers at different distances using the above function. We have chosen the value of k to be 7 and plotted the graphs for differing values of max_p . As is evident from the figure, the receiving probabilities have smoothed and are approximately the same across varying distances.

Figure 5 shows the variation in the marking probability using the VMP function. As the distance of the router increases, the marking probability also increases so as to compensate for the distance factor.

¹This is set to 128 in freeBSD 4.7 OS

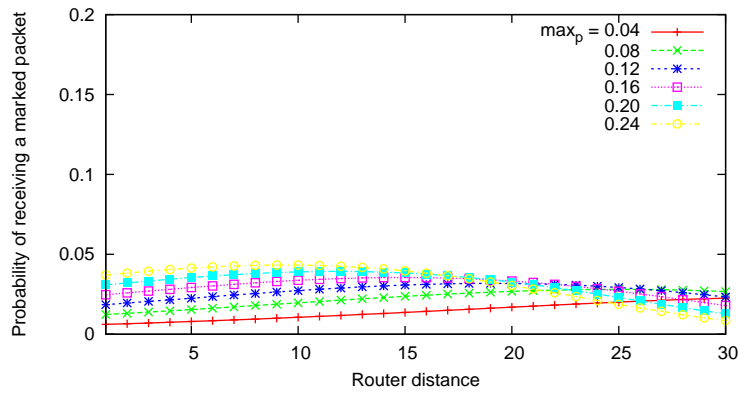


Figure 4: Probability of receiving a marked packet from varying distances using the VMP function for different values of \max_p with $k = 7$

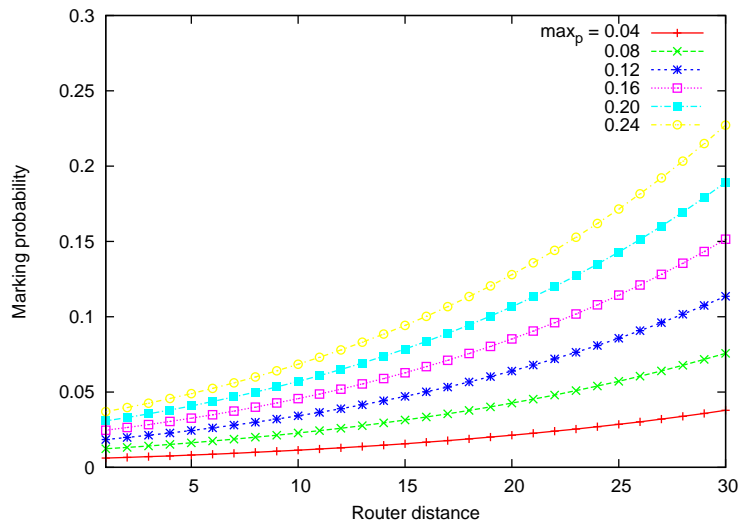


Figure 5: Marking Probability using the VMP function at routers of varying distance for different values of \max_p and $k = 7$

5 Experimental Platform

We have implemented, tested and evaluated our IP traceback scheme on the Emulab testbed setup. Emulab allows us to create different network topologies for testing our scheme. The routers used in our setup are software routers, where the router functionality is a part of the IP code. We use freeBSD 4.7 kernel for our implementation. We modified the router code to include the address stamps at the routers and to reconstruct at the end nodes. When a packet is received on an interface, the packet is placed in the `ip_intr` queue and a software interrupt is generated. This calls the `ip_input()` routine. The `ip_input()` routine then checks any address stamp in the packet. When an address stamp is present it adds it to the reconstruction bucket. When the packet is detected as a transient packet, `ip_input()` routine calls the `ip_forward()` function (when the `ip_forwarding` variable is set). In `ip_forward()` routine, we implement our algorithm for generating the address stamp. We use the in-kernel implementation of MD5 for generating the digests of the IP address. The IP control flow is shown in Figure 6

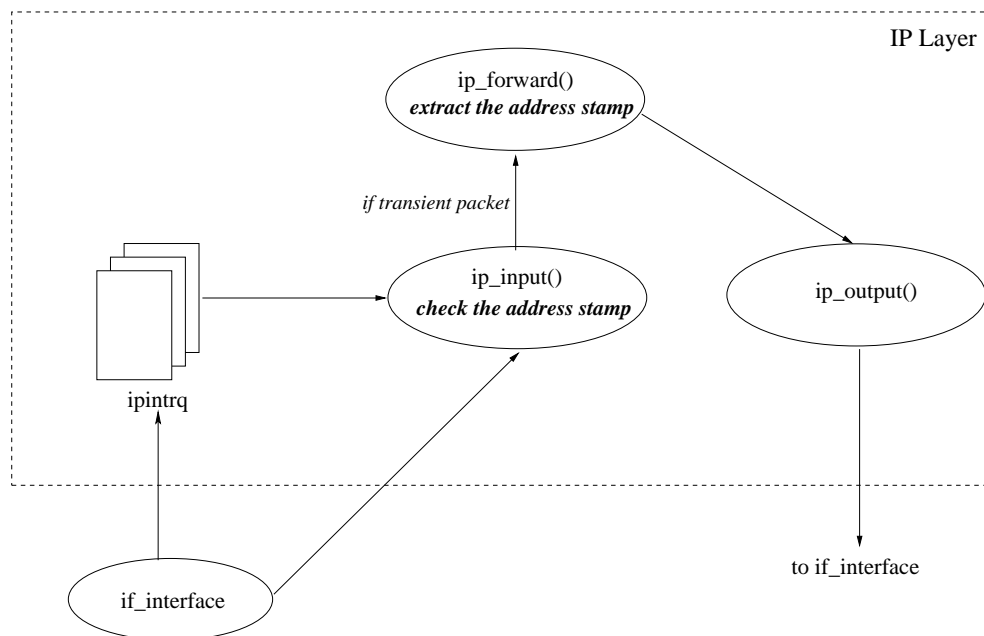


Figure 6: IP Control Flow

We have defined the new traceback functionality as a separate module called `.ADDRESS_STAMP..` This module can be optionally included by editing the `/sys/conf/options` file. When this module is included, the node puts its address stamp (as described in the earlier sections) in the transient packet².

²We use the outgoing interface for the address stamp

6 Evaluation

In this section, we evaluate our proposed modifications in light of the ideas presented in [3].

6.1 Improved Encoding Scheme

We ran simulations to compare the performance of our encoding scheme with the one proposed by Savage *et.al.*

6.1.1 Impact of attack distance

In this experiment, we simulate an attack from a single attacking node. By varying the distance of the attacking node from the victim, we compare the number of packets which are required by the victim to start reconstruction. Figure 7 shows the improvement in the convergence time in our scheme compared to the original one. As can be seen, even for an attacker at a distance of 25, the difference in the number of packets is about 1500.

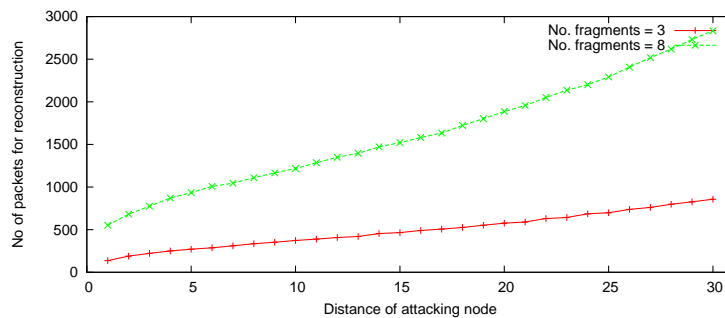


Figure 7: Comparison of Convergence Characteristics based on distance. The marking probability, p , is set to 0.04. Each distance result represents the average of 1,000 independent simulation trials.

6.1.2 Impact of marking probability

In the second simulation, we varied the marking probability p at the routers to see what effect it has on the convergence time in the two schemes. Here we do not use our Variable Marking Probability scheme and hence all routers mark packets with the same marking probability p . The impact of variable marking probability is discussed in Section 6.3.

We used a network with the attacking node 20 hops away from the victim. Figure 8 shows the comparison graph in a single attack path scenario. As p increases the convergence time increases steeply in the original encoding scheme. As was discussed in the previous section, if the attack distance is large, the victim will have to wait for a considerable duration before it can start the traceback. Our solution decreases this time significantly.

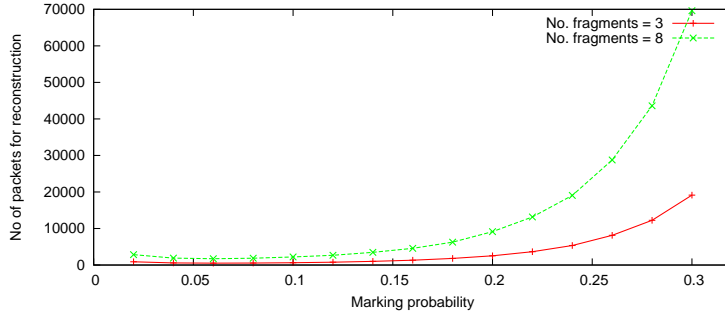


Figure 8: Comparison of Convergence characteristics based on probability. The distance d of the attacker is set to 20. Each probability result represents the average of 1,000 independent simulation trials.

6.1.3 Impact of number of fragments on convergence time

Since there are only three fragments per *edge-id*, the victim has to wait for only $3 * k$ fragments from any distance d compared to $8 * k$ in the Savage *et.al.* algorithm, where there are k routers at a distance d from it. This results in a 62.5% reduction in wait time before the victim can begin reconstruction.

6.1.4 Impact on reconstruction complexity

Also, during reconstruction, the victim would have to only compute k^3 IP addresses and verify them compared to k^8 in the previous case. Eventhough the algorithm is still exponential, simulations show that even when there are only 25 attacking nodes, there is a huge difference as can be seen in Figure 9.

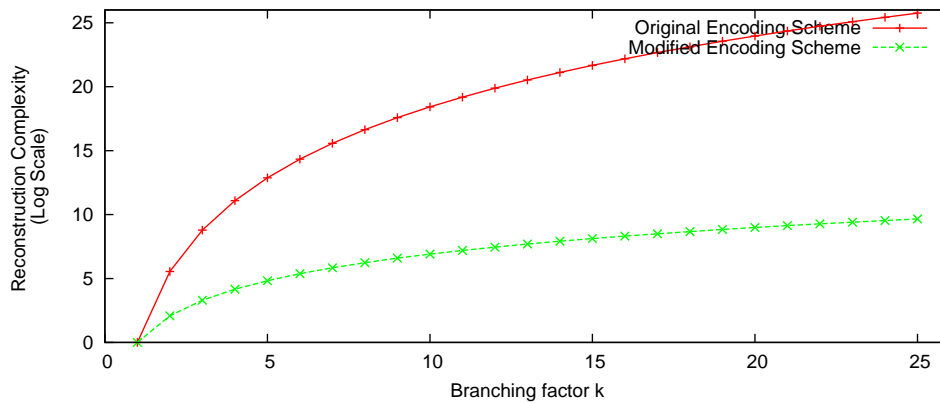


Figure 9: Comparison of the efficiency of the two encoding techniques

6.1.5 Impact of overlapping fragments

Another advantage of this encoding scheme is the additional reduction in reconstruction time provided by the concept of overlapping fragments. To narrow down the number of hash computations, the victim can compare fragments first using the overlapping bits, and then calculate the hash only for a set of matching fragments. Thus the number of hash computations are reduced considerably.

6.2 Modified Reconstruction Algorithm

Our reconstruction algorithm has been modified to work with the changed encoding scheme. The algorithm takes advantage of the encoding scheme and the concept of overlapping fragments to reduce the complexity as well as the time required for reconstruction. We have also ensured that multiple attack paths are reconstructed accurately.

6.3 Variable Marking Probability Scheme

We ran simulations in order to determine the optimal value the marking probability p which would provide further reduction in convergence time. The formula used in the scheme is given here once again.

$$p = \left(\frac{CurrTTL}{MaxTTL}\right)^k * max_p$$

We separately varied the parameters of k and max_p in the above function to obtain the following results. Figure 10 shows the plot of the number of packets required for reconstruction when the value of k is varied for different router distances. Each point on the graph is the average over 1000 independent runs. The parabolic behavior can be explained as follows. For low values of k , the marking probabilities of faraway routers is low requiring a large number of packets from them. For high values of k , the effect reverses and nearby routers mark with a very small probability. From the graph we can deduce that the best convergence is obtained for a value of k ranging between 4 and 5.

Figure 11 plots the number of packets required for reconstruction with changing values of max_p for various attack distances. The value of k chosen for all of these runs is 5. And as with the above case, each plot point represents the average of 1000 simulation runs.

As can be seen, for plots of attack distances at 15 hops and above, the curve is a parabolic function with best convergence occurring at the max_p value of 0.10. Correlating results from Figures 11 and 4, we deduce the reason for the parabolic behavior. For low values of max_p and large attack distances (>15 hops), the number of packets required is high (from Figure 11) because probability of receiving a marked packets from closer routers along the attack path is low, as is evident from Figure 4. Conversely, for high values of max_p and large router distances, the number of packets required is again high, because now, the probability of receiving marked packets from farther routers in the attack path is low.

As the number of hops reduces, the influence of distance overshadows the effect of the using the VMP function. But considering the entire spectrum of router distances, the best convergence is obtained with a max_p value of 0.10.

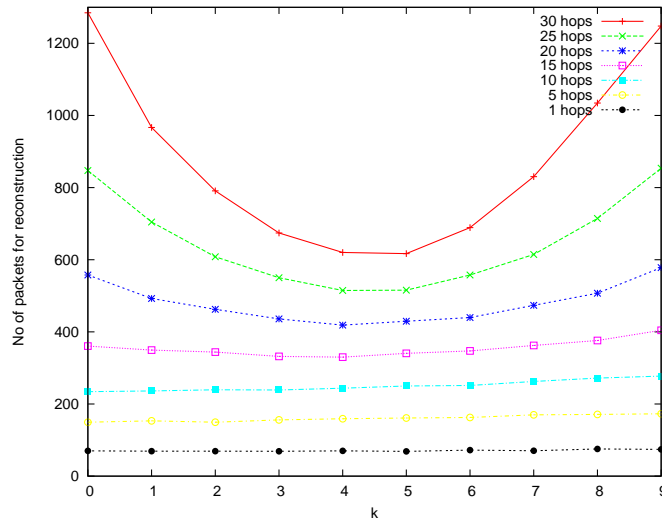


Figure 10: Determining value of k for an optimal Marking Probability p

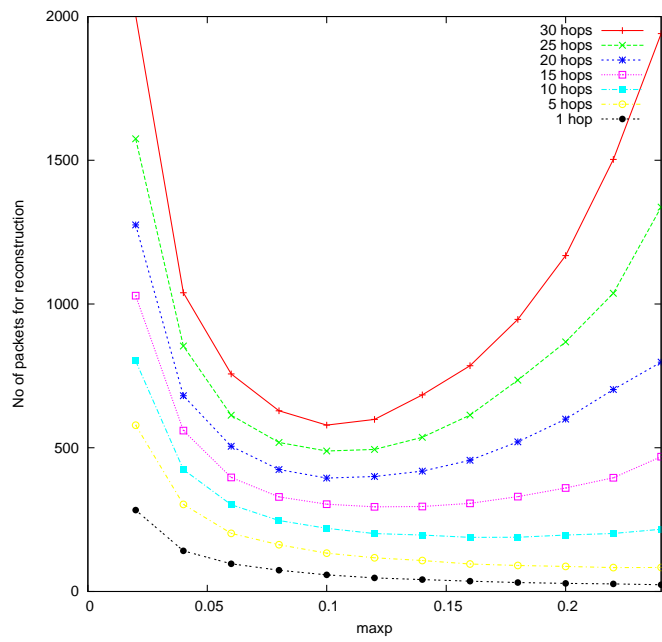


Figure 11: Determining value of \max_p for an optimal Marking Probability p

6.4 IP Fragmentation

One of the main problems faced in the original encoding scheme is that support for fragmentation is lost. To reduce the impact to legacy hosts, two solutions were proposed.

- To handle datagram fragmentation *upstream* from a marking router, prepending a new ICMP `echo reply` header, along with the full edge data was suggested. A separate marking probability q was to be used for this purpose. Although the packet is "lost" from the receiver's standpoint, the edge information is still preserved. We say that, while our encoding technique will also result in "loss" of such data fragments, the victim will still be able to reconstruct the path using such fragmented attack packets.
- To handle fragmentation that occurs downstream from a marking router, one of the solutions was to set the *Don't Fragment* flag on every marked packet. We have implemented this solution by preserving only this flag bit in our encoding mechanism as shown in Figure 2

7 Conclusion and Future Work

In this work, we have modified the encoding scheme and the reconstruction algorithm used in the IP traceback mechanism proposed in [3]. We have also shown that there is a marked reduction in the convergence time and reconstruction complexity. At the same time, we have ensured that our algorithm supports traceback of multiple attack paths. It is also capable of supporting the solutions to the IP fragmentation problem proposed by Savage *et.al.* Another major contribution of this work is the Variable Marking Probability (VMP) Scheme. This provides a further reduction in the convergence time at the victim.

This IP traceback scheme does not need heavy network infrastructure like that required in the Hash-based IP Traceback scheme proposed in [4]. It requires only modest changes in the routers and since these changes occur in the fast path of the router, it will not slow down the router processing capability.

We have identified potential areas where our scheme could be extended. Future work includes making the scheme compatible with IPSec and IPv6 packets. Another area of improvement is addition of high-bandwidth aggregate detection akin to that suggested in [2]. If the victim is able to detect aggregates of DoS flows, then only paths belonging to those aggregates could be reconstructed, further reducing path reconstruction complexity.

References

- [1] Schnackenberg D., Dgahandari K., and Sterne D. Infrastructure for intrusion detection and response. In *Proceedings of First DARPA Information Survivability Conference and Exposition*, 2000.
- [2] R. Mahajan, S. Bellovin, S. Floyd, J. Vern, and P. Scott. Controlling high bandwidth aggregates in the network, 2001.

- [3] Stefan Savage, David Wetherall, Anna Karlin, and Tom Anderson. Practical network support for IP traceback. In *SIGCOMM '00: Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, pages 295–306, New York, NY, USA, 2000. ACM Press.
- [4] Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Stephen T. Kent, and W. Timothy Strayer. Hash-based IP traceback. In *SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications*, pages 3–14, New York, NY, USA, 2001. ACM Press.
- [5] Dawn X. Song and Adrian Perrig. Advanced and authenticated marking schemes for IP traceback. In *Proceedings IEEE Infocomm 2001*, 2001.
- [6] Ion Stoica and Hui Zhang. Providing guaranteed services without per flow management. In *SIGCOMM '99*, pages 81–94, 1999.