DAMIT: When you realize you're under attack. (DDoS Attack Mitigation and IP Traceback)

Ranjana Addanki, Kevin Boutarel, Hussein Nagree

Abstract

Distributed Denial of Service (DDoS) attacks are an extremely common and dangerous threat in the modern internet. While a lot of effort has been made to stop these attacks in the servers that are targeted, there is a lot of scope for detecting, mitigating, or stopping these attacks within the network itself. In this paper, we present DAMIT, a network protocol for mitigating DDoS attacks, as well as retracing malicious attack traffic within a network. DAMIT uses programmable switches to tag packets with a custom header, along with marking and rate limiting techniques, which helps network administrators detect and suppress DDoS attacks directed at servers hosted in the network. A traceback tool can easily move around the network, as needed, and attach itself to any switch in the network to take advantage of tagged packets to trace the source of attack traffic. This process can run before or even during an attack and retrace attack paths between 80% and 100% accuracy as well as finding switches within 0-2 hops of the border switch. DAMIT provides flexibility to network administrators, the ability to be used in networks consisting of up to hundreds of routers, and protect servers of various capabilities from DDoS attacks.

1 Introduction

Today's Internet's infrastructure is very vulnerable to Distributed Denial of Service (DDoS) attacks from attackers with well-equipped tools to degrade performance and disable network services. In the last several years, Internet DDoS attacks have increased in frequency and severity. On October 26th, 2016, the servers of Dyn, a company in control of a large amount of the DNS infrastructure, were victim of the "Mirai botnet" attack, making users unable to access certain websites on the Internet, like Twitter, Netflix, and many more [1]. Approximately, one hundred thousand "Internet of Things" (IoT) devices were infected [9].

In a Denial of Service (DoS) attack, a server is flooded with fake traffic, causing it to crash, to degrade its performance, or to make it unable to respond to legitimate traffic. DDoS attacks are the most common type, where an attacker sends traffic to a victim from different sources. This makes it hard to distinguish legitimate traffic from attack traffic, and almost impossible to prevent an attack by just blocking a few source IP addresses. Additionally, vulnerabilities in the IP Protocol enable attackers to easily fake the source of an IP packet. With techniques like IP spoofing, encapsulation, and NAT forwarding, it becomes difficult to determine the location of attacking machines. It also prevents in-network security techniques, like ingress filtering, from stopping the attack. Mitigating attack traffic is only a small step in defending against DoS attacks. A much larger accomplishment and harder step is to identify the entities responsible of such attacks [11].

To find the true source of an attacker, it's important to identify a technique for finding the origin of the IP packets without relying on the source IP address field in the packet header. This mechanism is called IP traceback. It provides systems the capability to trace a single packet back to its origin [4]. While there are many approaches for IP traceback, a popular mechanism is packet marking. The main idea is to record network path information inside the packets. Once a packet traverses a router, it writes its identification information (e.g., IP address) into a header field of the packet [11]. This marking can be done either probabilistically or deterministically. By putting together the information from all the packets, the victim can then reconstruct the attack path back to the adversary.

Creating rigorous protection against DDoS attacks is a hard problem to solve for potential victims. To build firewalls or other defensive systems requires some knowledge of computer security. However, victims are often small companies, hosting web servers, that do not employ security experts. Thus, they usually lack the technical know-how to build proper secure systems. Some users may be unable or unwilling to spend the resources required to build and maintain such a system despite knowing the risks of an unsecure system.

To remedy these concerns, this paper proposes the DAMIT (DDoS Attack Mitigation and IP Traceback) system. DAMIT fulfills two important needs. First, it lets the network act on behalf of the victim, by monitoring its incoming traffic and taking action upon detecting an attack. Second, the system collects information about attackers by finding the attack path taken through the network, which enables the location of potentially compromised machines or botnets. With DAMIT, ISPs and network operators can provide protection against DDoS attacks, hold unsecure users more accountable, and offer better reliability to their clients. The latter benefits by not wasting valuable resources towards implementing security procedures while operators provide better services to more users, and potentially amortize their costs. DAMIT enables the network to detect attacks closer to the source; this lowers the internal bandwidth by not carrying heavy attack traffic throughout the network.

2 Design

2.1 P4 Implementation

2.1.1 A Brief Overview of P4

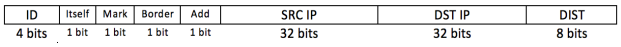
P4 allows a user to program protocol-independent packet processors [10][13]. The programmability comes in three ways: the creation of custom protocol headers, the parsing of existing and new protocol headers, and the customization of match-action tables.

As a packet arrives at a P4 switch (a switch created by a P4 program), it first gets parsed into the defined headers specified in the program. The packet headers, and metadata associated with the packet, traverse the ingress and egress pipelines containing match-action tables. Each table will read various fields of the packet headers or metadata and an action will be executed according to the entries in the tables. These entries will be entered at compile time. Between the two pipelines, a buffer is put in place which allows incoming and outgoing packets to be cloned. These cloned packets can recirculate, going through both pipelines again, or simply the egress pipeline.

The DAMIT protocol utilizes heavily these three programmable aspects of P4, along with the possibility of cloning packets.

2.1.2 DAMIT Protocol Header

The custom DAMIT protocol header is the basis of the whole system. It is inserted between the link layer (Ethernet, etc.) and the network layer (IPv4, IPv6, etc.), and is defined in the following way in the P4 program:



The first 4 bits are used to represent the *id* of the header. This is always set to 0. It is set this way so that the parser can distinguish between parsing the network protocol header or the DAMIT protocol header. The following 4 bits are considered the flags of the protocol and will be discussed in more detail in the following section. The *src* and *dst* fields are used for the ids of the current switch and the next hop switch, from the perspective of the switch adding the header. IPv4 addresses can be inserted into these fields; however, with the flexibility of P4, they could easily be changed to represent their IPv6 counterparts (which would use 128 bits) or some other unique id. The *dist* field shows the number of hops needed to reach the destination from the src switch. Since this field is only 8 bits long, the maximum length of the path of the network will be 256 hops. Again, this can be modified easily to suit larger or smaller networks. With this custom defined header as the foundational basis, switches can now be integrated and become core components of the DAMIT system.

2.1.3 Enabling IP Traceback using P4 switches

Each switch will play an active role in retracing attack paths from malicious traffic. Each P4 switch in the network effectively emulates a router. This is important because these switches need to know where

to send traffic based on destination IP addresses. The network administrator can assign a unique id to each switch, by keeping a mapping of these in order to translate the path taken within the network (or simply allow a switch's IP address to represent its id). These ids will be used when switches populate the src and dst fields of the protocol header. Two additional bits of information also need to be added to each switch. The first is a specific bit "border" that is set to 1 if the router is a border router of the network (i.e. connected to routers/hosts outside of the network) or 0 otherwise. The second bit is set to 1 if the switch is currently attached to the traceback tool. As of right now, retracing attack traffic cannot be done entirely in the data plane and requires the help of an external tool to be attached to switches of interest. Future work discusses incorporating the traceback process entirely in the data plane. With all of this information set for each switch, they can now start to receive and send out packets.

Upon receiving a packet, the following logic is executed at the ingress point:

- With probability 1/P (where P is an adjustable parameter), reset or add a new header:
  - If no header is present on the packet, add a header with id set to 0.
  - Set the itself and mark bits to 0, and the border bit to the value saved in the switch.
  - Set the add bit to 1. This tells the next switch that the previous switch added or reset the header and it can therefore add itself as the destination of the edge (assuming it doesn't reset the header itself).
  - Set the src field to be the switch's id, and the dist field to 0.
- Otherwise, if a header is already present:
  - If the add bit is set to 1, then set the dst field to be the switch's id, and increment the dist field.
  - If the add bit is set to 0, increment the dist field.
  - If the itself bit is set to 1, change it to 0.
  - If none of these are true, no operation should be done on the packet's header.

A potential security measure can be incorporated into the ingress point if this protocol became popular. Packets entering the network already containing the DAMIT header would indicate that attackers are forging packets to tamper with the traceback results, by modifying the actual distance of the attack path. Such packets could be dropped upon detection.

Since the DAMIT system should not affect actual traffic, packets exiting the network (either at border routers or routers connected to hosts and servers in the network) need to remove the DAMIT header, if present, before sending out the packet. However, if the packet is being sent to the traceback tool, it will be sent as-is and the header will not be removed.

This algorithm could be extended to fit not only P4 switches, but any programmable switch or even hardware capable to reproduce these possible steps.

2.1.4 Marking and Rate Limiting

The most straightforward way to stop active attack traffic is to apply rate limiting techniques. Additional information can be set at each switch to implement these. Upon receiving a certain number of packets, the switch can decide between two actions: marking or rating.

When marking, the switch will set the mark bit of the header to be 1. This will indicate, further down the network, that this packet reached at some point in the path a marking switch. Of course, if the DAMIT header did not exist before this point, the switch will need to add it and set itself as the src of the edge, so that in case it would not have been added later on, it will at least contain this edge. If at some point the switch decides to rate limit the traffic, it can simply drop packets that go over a certain threshold.

The thresholds for the marking and rate limiting procedures can be stored within counters or registers. One of these should be used per distinct destination and only one is necessary to complete the two actions. This container should be reset after a certain period of time so that actual traffic can keep flowing through the network. For P4 [10], counters are impractical since there is no current method of reading them. Registers are a perfect substitute, but require the user to manually increment the register upon receiving a packet. The timing metadata is not completely incorporated into the behavioral model of P4,

but future work can be done to incorporate this into the model.

These two techniques allow network administrators the flexibility to set up their networks based on their requirements. Switches can decide to mark and rate limit packets. Certain links will see their packets get marked after some percentage of the link is used and then dropped if saturation of the link is met. Marking and not rate limiting provides the benefit to examine which edges are getting a lot of traffic towards a specific destination.

Additionally, the traceback tool has the ability to handle both types of packets: marked and unmarked. The former allows the traceback process to focus on heavy attack traffic towards the victim and not handle potentially benign attacks or even regular traffic.

2.2 IP Traceback

Extensive past work on the IP Traceback problem has shown that while this is not an easy problem, there exist multiple algorithms to solve it. The IP Traceback algorithm used in DAMIT is presented in *Practical Network Support for IP Traceback* [2]. The paper offers two possible ways to solve this problem. The first method requires finding all the nodes traversed in the attack path, while the second method requires finding all possible edges. The latter, as explained in the paper, is more efficient and robust against attackers.

The traceback tool is a program outside of the network which can retrace attack paths for a single destination IP address of interest. Not only can it retrace the attack path of a single attacker, but also the paths of multiple attackers simultaneously. To do so, it maintains a graph and analyses the DAMIT header to insert the edge appropriately, until it has either found a complete path from the attacker to the victim, or a certain number of packets has been reached. This threshold can vary based on the requirements of the network administrator. Additionally, this process can be a proactive or reactive solution by starting before or even during an attack and complete in both cases. To make this program work, the traceback tool attaches itself to one of the switches (usually the switch that is connected to an inner host or server). The switch must then clone any incoming packet that is destined to that host or server and not remove the DAMIT header. The traceback process might not be able to find a complete path if the packet threshold was hit. However, since the DAMIT system allows any switch to have this tool attached to it, this tool has the ability to move around the network as needed. It can detach itself from one switch and attach itself to another. Updates to both switches could make it appear as if the traceback tool had been set there in the beginning.

The traceback process explained in *Practical Network Support for IP Traceback* was subtly modified by not checking if edges had the correct length since forged packets can easily be dropped as they enter the network. Additionally, this paper enhanced the algorithm by incorporating the ability to execute it at any node within the network, not only at the victim.

3 Evaluation

All of our code was developed on a Linux virtual machine [12]. The VM was running Ubuntu 64-bit v14.04.01. The code used and written in P4 was from v1.1.1. Mininet was the best choice to simulate a complete network topology and traffic. Not only does it have ease of use because the code base is in Python, but P4 was very elegantly integrated with Mininet. The version of Mininet was 2.3.0d1 and the traceback program was written in Python v2.7.

This following section will cover a first look at how to configure Mininet to run the DAMIT system on different topologies, then a description and evaluation of experimental results obtained from running a variety of tests.

3.1 Network Configuration

Mininet needs three important pieces of information to implement a network topology: the number of switches, the number of hosts, and the links between pairs of nodes in the network.

Our program running Mininet allows a user to define any topology beforehand to be passed via a certain file. An example of this can be seen in Figure 2 which is the equivalent representation of the topology in Figure 1 [2]. The file must begin by stating the number of switches contained in the topology. The next two lines show the number of hosts that will be present in this network. Outer hosts will be hosts that are not contained within our network while inner hosts will be. Therefore, any switch connected

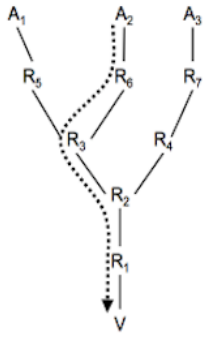


Figure 1: Example of network topology, where A1, A2, A3 are attackers, Ri are routers in the network, and V is the victim.

```
1 topo.txt
  1 switches 7
  2 outer_hosts 3
  3 inner_hosts 1
  4 o1 s5
  5 o2 s6
  6 o3 s7
  7 s5 s3
  8 s6 s3
  9 s7 s4
 10 s3 s2
 11 s4 s2
 12 s2 s1
 13 s1 i1
```

Figure 2: Example of network topology description

to an outer host will be considered a border router. Also, any switch connected to an inner host will have a traceback tool attached to it. The following lines of the file represent links to add to the network. Currently, our Mininet program works best under the condition that the topology is a minimum spanning tree. Future work could be extended to incorporate more varied forms.

Once a topology has been defined, the P4 switches require a behavioral executable and a JSON-compiled version of the P4 program to get them running. These switches populate their tables through a command-line interface. Modifications have been made to the original CLIs to incorporate the possibility to pass updates directly via the command line without needing to pass them via a file. This allows the code to be more modular and removes the overhead of creating extra temporary files when updating the P4 switches. While some entries might be specific to each switch, a set of default commands can be used to work on every switch. These instructions usually represent default actions that tables should take when a table miss occurs, or entries that need no parameters passed to a certain action. An example of such a file is shown in Figure 3.

```
1 default.txt
  1 table_set_default ipv4_lpm _drop
  2 table_set_default forward _drop
  3 table_set_default clone_for_trace _no_op
  4 table_set_default drop_table _drop
  5 table_add clone_for_trace clone_pkt 1 => 250
  6 table_set_default set_clone _no_op
  7 table_add marking set_mark 1 =>
  8 table_set_default modification _no_op
  9 table_add modification inc_tag 1 0 0 =>
 10 table_add modification remove_self 1 1 1 =>
 11 table_set_default send_frame _drop
```

Figure 3: Example of list of default commands to install on each switch

3.2 Running tests

3.2.1 IP Traceback

a

```
----------------------
Attack Path: c090010a <---> 0a00000a
Total packets received: 29
Total packets tagged: 4
Percentage packets tagged: 0.14

Complete path found:
Border router: 78787806
Path: ['c090010a', '78787806', '78787803', '78787802', '78787801', '0a00000a']
Edges (src <---> dst : dist): 5
c090010a <---> 78787806 : 4
78787806 <---> 78787803 : 3
78787803 <---> 78787802 : 2
78787802 <---> 78787801 : 1
78787801 <---> 0a00000a : 0
----------------------
```

b

```
----------------------
Attack Path: c090000a <---> 0a00000a
Total packets received: 56
Total packets tagged: 7
Percentage packets tagged: 0.12

Complete path found:
Border router: 78787805
Path: ['c090000a', '78787805', '78787803', '78787802', '78787801', '0a00000a']
Edges (src <---> dst : dist): 5
c090000a <---> 78787805 : 4
78787805 <---> 78787803 : 3
78787803 <---> 78787802 : 2
78787802 <---> 78787801 : 1
78787801 <---> 0a00000a : 0
----------------------
```

c

```
----------------------
Attack Path: c090020a <---> 0a00000a
Total packets received: 68
Total packets tagged: 10
Percentage packets tagged: 0.15

Complete path found:
Border router: 78787807
Path: ['c090020a', '78787807', '78787804', '78787802', '78787801', '0a00000a']
Edges (src <---> dst : dist): 5
c090020a <---> 78787807 : 4
78787807 <---> 78787804 : 3
78787804 <---> 78787802 : 2
78787802 <---> 78787801 : 1
78787801 <---> 0a00000a : 0
----------------------
```

Figure 4: Output of IP Traceback process from Experiment described in 3.2.1

The traceback tool implemented for these tests is a Python program that uses NetworkX, a third-party

Python library used for working with graphs, to retrace attack paths of a singular or multiple attackers to a destination IP address.

When implementing the topology from Figure 1 into the DAMIT system, the output obtained from the traceback program can be seen in Figure 4. Each attacking host, Ai, was assigned an IPv4 address of the form 192.144.0a.10, where a equals i-1. This is equivalent to its hexadecimal form of c0900a0a. Each P4 switch, Ri, was assigned an id of the form 120.120.120.0i, which leads to 7878780i. Finally, the victim host was assigned the IPv4 address of 10.0.0.10 which is the same as 0a00000a. The output of the trace shows the number of packets received and needed to complete the path from source to destination and how many were tagged with the DAMIT header among them. Additionally, it shows whether the complete or a partially complete path was found, along with the connected nodes of said path. Finally, it shows the total number of edges found along with the src and dst ids of the router edges along the way including the dist field found.

Even if a partial attack path was retrieved, since the DAMIT system allows any switch the possibility to have the traceback tool attached to it, the tool is able to move around the network as needed. This allows the tool to find the border router of the malicious traffic and possibly complete the attack path by running the process from the new switch to the same destination.

3.2.2 Finding the perfect P

Finding the best probability 1/P of adding/resetting the DAMIT header was a simple but important goal. This probability would lead to implementing solutions to DDoS attacks closest to the attacker but also show the different links that are carrying the heavy load of the traffic within the network.

Using our Mininet implementation described earlier, linear topologies were created with variable path lengths, ranging from 5 to 50 switches with increments of 5. The attacker is connected to router R1 whereas the victim is connected to router Rn, where n is the total path length. The same assignment scheme for the router ids and the IPv4 addresses was used as described above. Three different series of tests were made with P values of 10, 25, and 50. The test involved pinging the victim from the attacker and record the output of the traceback program. It was given a threshold of 100 packets to find the complete path between the attacker and the victim. The test was repeated 5 times for each path length and each values of P, leading to a total of 150 tests ran. The complete data recorded can be found at our GitHub repository [12]. Two important graphs resulted from these tests.

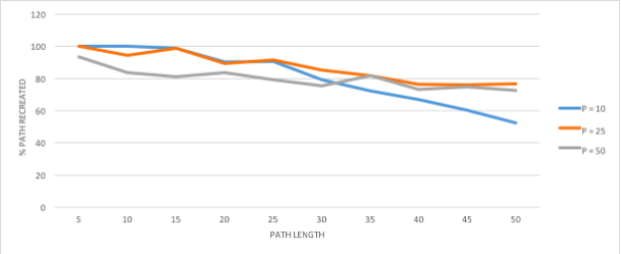


Figure 5: Plot of percentage of path recreated vs length of a path in the network

Figure 5 shows the percentage of the path recreated versus the length of the path. The lines for P = 10 and P = 25 follow the same trend until the length of the path exceeds 25 switches. After that, P = 10 drops down until reaching 50% for a path length of 50 switches, while P = 25 is able to recreate around 80% of the attack path. The trend of the P = 10 line is understandable since the switches closer to the victim will have a higher chance of tagging packets, thus hindering the ability to retrieve edges that occurred beforehand. As a contrast, the lines for P = 25 and P = 50 differ until a path length of 35 switches where they follow the same trend until the end. The switches with probability P = 50 are not able to retrace the attack path since so few packets will actually get tagged. This reduces the chance of a packet being tagged by any edge along the path. This plot favors P = 25 to have the best capability of retracing the complete attack path.

Figure 6 shows the farthest router id of the path found on average versus the path length. Since the attacking host was connected to router R1, then finding the id closest to 1 would show that we have found a router that is closest to the attacker. This information can then be used to implement rate limiting techniques to stop DDoS attacks closer to the source. The graph shows that all three values of P are the same until a path length that exceeds 30 switches. After

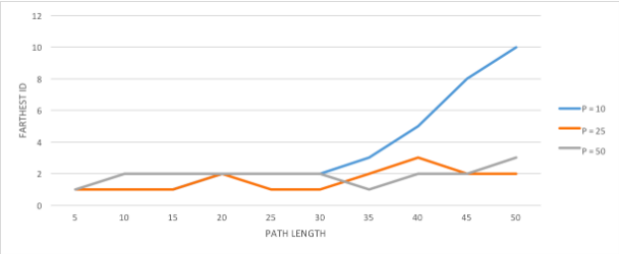


Figure 6: Plot of farthest id found on the path vs length of a path in the network. Routers are numbered from 1 to N, where N is the length of the path. Finding router 1 means finding the border router through which the packet came into the network

that, P = 25 and P = 50 are very close together in finding routers very close to the attacker, around 2 hops away from the actual border router. However, the P = 10 line is unable to find routers that are within 10 hops of the border router for an attack path of 50 switches. The logic explained above also explains this trend for a smaller probability of tagging packets.

Therefore, the tests show that assigning a probability of 1/25 to each switch will produce the best results needed to recreate the complete attack path of malicious traffic as well as finding routers extremely close to the attackers.

4 Related Work

There have been many papers that have surveyed multiple IP traceback schemes. There are four different types of techniques: packet marking, link testing, ICMP traceback, ingress filtering.

4.1 Packet Marking

Packet marking techniques insert data needed for traceback into an IP packet header. This marks the routers the packet traversed from the attack source to the destination. These marks can then be used to reconstruct a path in order to deduce the source(s) of packets and the path of the traffic [4]. The following two sections will detail the current packet marking schemes.

4.1.1 Probabilistic Packet Marking

Attacks generally comprise of many packets and while each marked packet represents only a sample of packets, the combined sampled packets can allow the victim to reconstruct the entire path. Thus, their solution is to probabilistically mark packets with partial path information as they arrive at routers to allow the victim to reconstruct the attack path [2]. It encodes this information in a 16-bit fragment ID field in the IP header [4]. To ensure that the path traversed by each packet is fully reported, a set of routers sample packets, and use the field to report the last sampled router to the recipient. Thus, in situations of DoS attacks where there is a single attacker, the observed frequencies of router addresses directly yield the set of routers on the path, the most frequently observed addresses correspond to the router closest to the attacker, and vice versa [1].

However, this approach doesn't address distributed attacks because the practical implementation has difficulty in grouping fragments together. Thus, the probability of misattributing an edge increases when there are many attackers in play. There is also an issue of path validation because some number of packets sent by the attacker are unmarked by intervening routers. This makes it difficult for the victim to identify these packets versus genuine marked packets. Therefore, an attacker could insert "fake" edges by carefully manipulating the identification fields in the packets it sends [2]. Furthermore, the path reconstruction process requires high computational work especially when there are many sources. Likewise, when there are many attack sources, the possible rebuilt path branches are useless to the victim because of the high false positives [4].

4.1.2 Deterministic Packet Marking

Another stream of packet marking methods is known as Deterministic Packet Marking (DPM). In this algorithm, each packet is marked when it enters the network. The marking is a partial address of the IP address of the interface closest to the source of the packet on the edge ingress router. It does so by splitting the IP address in two parts and populates the ID field with either the first 16 bits or the last 16 bits, and mark the reserved flag with 0 if it is the first part and 1 if it is the second part. This randomness prevents attackers from sending exactly every other packet to

the victim, which prevents an attacker from creating a situation where the victim only has a part of one address.

An issue with deterministically tagging these packets is the address fragmentation/reassembly problem. Enough packets must be collected to reconstruct the attack path. Another issue with this method is that if the attacker frequently spoofs the source address with a different value each time, this approach to tagging may become void [3].

4.1.3 Flexible Deterministic Packet Marking

Another packet marking scheme is known as Flexible Deterministic Packet Marking (FDPM). This utilizes various bits (or marks) in the IP header, which allows it to have flexible lengths depending on the network protocol used. When an IP packet enters the protected network, it is marked by the interface closest to the source of the packet on an edge ingress router. The source IP addresses are stored in these marking fields and this mark will not be overwritten by other routers this packet traverses. The source IP addresses can be reconstructed at any point within the network by the victim. Since the marks don't increase in size, no additional bandwidth is consumed [4].

4.2 Link Testing

This technique starts from the router closest to the victim and interactively tests its upstream links until they determine which one is used to carry the attacker's traffic. This method assumes that the attack remains active for the duration of the trace. Thus, this is infeasible for attacks that are detected after the fact, attacks that occur intermittently, or attacks that modulate their behavior in response to a traceback [2].

Many routers include a feature called input debugging that allows an operator to filter particular packets on some egress port and determine which ingress port they arrived on. First the victim must determine that it is being attacked and develop an attack signature. After that it contacts the network operator and communicates its signature. The operator then installs an input debugging filter on the victim's egress port which will identify which router the packet came from. This will recursively repeat itself until it reaches the attacker or it leaves the ISP's border. Once it reaches the border, the victim must contact the ISP to continue the recursive process [2][6]. Even if there are automated tools to conduct this trace, there is a lot of management overhead. This method requires a lot of communication and coordination with network operators at multiple ISPs at different times. This can cause the traceback to be slow or impossible to complete [2][5].

4.3 ICMP Traceback

This technique makes use of ICMP, a protocol used by network devices to send error messages and operational information. Bellovin proposes a scheme that every router samples packets, with a low probability (about 1/20,000), and sends an ICMP traceback message to the destination, along with that packet's contents. This should include information about the adjacent routers along the path to the destination. During a DoS attack, the victim can use these messages to reconstruct the attack path [2][8].

Though this method is similar to the packet marking techniques, it has several disadvantages. These ICMP packets are very different and may be filtered or rate limited differently from normal traffic. This scheme also requires a key distribution infrastructure to deal with the problem of attackers sending false ICMP Traceback messages [2].

4.4 Ingress Filtering

An approach to address the problems of anonymous attacks is to configure routers to block packets that arrive with illegitimate source addresses [2 [7]. This means each router needs to be able read each packet's source address and determine what a legitimate and illegitimate address is. This is not feasible in all networks; just mainly in customer networks or at the border of Internet Service Providers where address ownership is relatively ambiguous and traffic load is low. Furthermore, the overhead of ingress filtering becomes prohibitive on high speed links. Another problem with this type of solution is that its effectiveness depends on widespread deployment. Even if this was widely deployed, attackers could still forge addresses from the hosts within a valid customer network even if the host space is smaller [2].

5 Conclusions

The DAMIT system is a fully functional system which relies on programmable switches to stochastically tag packets with a custom defined header throughout the network. These switches also have the ability to apply DDoS mitigation techniques such as marking and rate limiting of attack traffic. An external traceback tool is used to retrace malicious traffic traversing the network. This process can recreate 80-100% of attack paths as well as find routers within 0-2 hops away from the border routers. Additionally, this tool can move around the network to possibly complete the rest of the attack path and find a closer router at the edge of the network. By detecting attacks closer to the source, DAMIT lowers the internal bandwidth of the network providing better services to clients, and locates potentially compromised external hosts and botnets.

References

[1] M. Grossglauser and J. Rexford. Passive Traffic Measurement for IP Operations. In *The Internet as a Large Scale Complex System*, 2002

[2] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical Network Support for IP Traceback. In *Proceedings of the conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM '00)*. ACM, New York, NY, USA, 295-306

[3] A. Belenky and N. Ansari. IP Traceback With Deterministic Packet Marking. *IEEE Communications Letters*. 7(4), April. 2003

[4] A. Parvathi and G.L.N. JayaPradha. An IP Trace back System to Find the Real Source of Attacks. In *International Journal of Computer Trends and Technology*. 2(1), 2011

[5] J. Glave. Smurfing Cripples ISPs. *Wired Technology News*, Jan. 1998

[6] R. Stone. CenterTrack: An IP Overlay Network for Tracking DoS Floods. In *Proceedings of the 2000 USENIX Security Symposium*, Denver, CO, July. 2000

[7] P. Ferguson and D.Senie. Network Ingresss Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing. *RFC 2267*, Jan. 1998

[8] S.M.Bellovin. ICMP Traceback Messages. Internet Draft: draft-bellovin-itrace-00.txt, Mar. 2000

[9] Woolf, N. DDoS attack that disrupted internet was largest of its kind in history, experts say. *The Guardian*, Oct. 2016

[10] http://p4.org/

[11] C. Gong and K. Sarac. A More Practical Approach for Single-Packet IP Traceback using Packet Logging and Marking. *IEEE Transactions on Parallel and Distributed Systems*. 2008;19(10):1310âĂŞ1324

[12] https://github.com/kboutarel/DAMIT

[13] Pat Bosshart, Dan Daly, Glen Gibb, Martin Izzard, Nick McKeown, Jennifer Rexford, Cole Schlesinger, Dan Talayco, Amin Vahdat, George Varghese, David Walker. P4: Programming Protocol-Independent Packet Processors. *ACM Sigcomm Computer Communications Review (CCR)*. Volume 44, Issue 3 (July 2014)