Locating Network Domain Entry and Exit point/path for DDoS Attack Traffic

Vrizlynn L. L. Thing[1,2], *Student Member, IEEE,* Morris Sloman[2], *Member, IEEE,* and Naranker Dulay[2], *Member, IEEE*
vlt,mss,nd@doc.ic.ac.uk
Institute for Infocomm Research[1], Imperial College London[2]

***Abstract*—A method to determine entry and exit points or paths of DDoS attack traffic flows into and out of network domains is proposed. We observe valid source addresses seen by routers from sampled traffic under non-attack conditions. Under attack conditions, we detect route anomalies by determining which routers have been used for unknown source addresses, to construct the attack paths. We consider deployment issues and show results from simulations to prove the feasibility of our scheme. We then implement our Traceback mechanism in C++ and more realistic experiments are conducted. The experiments show that accurate results, with high traceback speed of a few seconds, are achieved. Compared to existing techniques, our approach is non-intrusive, not requiring any changes to the Internet routers and data packets. Precise information regarding the attack is not required allowing a wide variety of DDoS attack detection techniques to be used. The victim is also relieved from the traceback task during an attack. The scheme is simple and efficient, allowing for a fast traceback, and scalable due to the distribution of processing workload.**

***Index Terms*—Distributed Denial of Service, IP Traceback.**

I. Introduction

CURRENT networks do not usually perform any form of authentication on the source IP address. This is exploited by many Distributed Denial of Service (DDoS) attacks [1], where the attackers use spoofed source addresses to hide their identity and location. Some service providers do perform ingress filtering at access routers to check for valid source IP , but this is not completely effective. The Spoofer Internet-wide active measurement Project [2] concluded in 2005 that approximately 360 million addresses and 4600 autonomous systems were vulnerable to spoofing, and concerted attacks employing spoofing remain a serious concern. In 2006, backscatter analysis [3] was conducted where attack traffic using IP spoofing was captured. It shows that over a period of 3 years from 2001 to 2004, 22 collected distinct traces revealed 68,700 attacks on over 34,700 distinct Internet hosts. Traceback mechanisms [4]–[14] have been proposed to trace the true source of the attackers to stop the attack at the point nearest to its source in order to reduce waste of network resources and to try to determine the identity of the attackers to be able to prosecute or take other actions against them.

In infrastructure traceback schemes, the network is responsible for generating and evaluating traceback state information to construct the attack graph of the routers through which attack traffic is passing. In IP Logging [4], intermediate routers log the invariant portion of the IP header (20 bytes) and the first 8 bytes of the payload of all IP packets. Hashing is then performed on the 28-byte information obtained above, followed by a Bloom filter processing to reduce the storage requirement. The logs are retrieved from various routers when traceback for the path taken by any single IP packet is initiated. Given a copy of an attack packet, and an approximate time of receipt, it is, in theory, possible to generate a similar hash and search router logs to determine the attack graph. However, the overheads of generating and storing even a 28 byte hash can be rather high so IP Logging is not done in most networks.

In the end-host traceback schemes, potential victim hosts maintain the traceback state information. In IP Marking [5], [9]–[11], [13], [14], intermediate routers along the path taken by the packets mark their addresses into the packet with a predefined probability. The victim of the attack can then examine the information found in the attack packets so as to construct the attack path. In ICMP Traceback (ITrace) [6], an intermediate router probabilistically generates an ITrace message for each IP packet it processes, and sends it to the same final destination of the IP packet. The victim of the attack can therefore use the ITrace messages to construct the attack path. Various enhancements have been proposed to ITrace to improve performance [7], [8], [12], which are discussed in more detail in Section VIII. However, all the above-mentioned traceback schemes require that the attack packets are distinguishable from legitimate packets. This is due to the need for the identification of an attack signature in the packets to initiate and perform traceback. IP marking and ITrace (and its variants) also require changes to be made to the routers to allow for participation in the traceback process.

Standard routing protocols perform packet forwarding based on the destination IP address in the packets so packets belonging to a particular source-destination pair follow a relatively static path as routing tables are not updated very frequently under normal conditions. When an attacker spoofs a legitimate user's source address, the packet may pass through routers which are not on the normal source-destination routing path and this anomaly can be used to determine the attack path. Our method builds and maintains caches of valid source addresses for routers in the network from sampled traffic under non-attack conditions. Under attack conditions, we determine which routers have been used for unknown source addresses to construct the attack graph within an administrative domain. We propose two approaches of our scheme: Network Segmentation Based (NSB) and Strategic Points Based (SPB). In the NSB approach, the network is segmented, routers in each segment are assigned to the distributed White List (WL) Caching Device and the Traceback Manager consolidates information received from these devices to generate the attack graph. In the SPB approach, instead of covering the whole network, traffic sampling is performed in strategic routers in the network where incoming and outgoing traffic will definitely traverse. Therefore, the number of sampling points and packets are reduced, resulting in reduction of processing workload and overhead traffic. Thus the Traceback Manager and WL caching device functionalities can co-exist in the same system, eliminating the need for information dissemination. We have also defined an extension for inter-domain support to identify the network point nearest to the attack source. The strengths of this scheme are its scalability due to the distribution of processing workload and speed due to the simple computation for the attack graph construction. The elimination of the need to make modifications to the routers, victim and data packets to support traceback, unlike existing techniques, allow the scheme to be "non-intrusive". In addition, we analysed the scheme, proposed enhancements to it and considered the deployment issues.

Section II of the paper presents the design objectives and key assumptions. Section III describes our traceback scheme. Section IV explains the deployment considerations. Simulations and experiments conducted are presented in Section V and VI, respectively. Section VII elaborates on security threats and the limitations of our approach, with Section VIII discussing related work and comparisons with existing techniques, followed by the conclusions in Section IX.

II. Design Objectives and Key Assumption

A. Design Objectives

Our design is based on the following objectives:

- As changes to the Internet infrastructure raise conformance issues, modifications to the Internet infrastructure should not be required.
- Methods, such as IP Marking [5], [9]–[11], [13], [14], require information to be placed in the original data packets. However, as there is no unused field in the IP packets, fields reserved for other purposes, such as the identification field for fragmentation, have to be used instead. This could result in overwriting existing information and packet corruption, so changes to the original data packets should not be required.
- During a DDoS attack, the victim would be overwhelmed from the attack traffic. Therefore, no additional "burden" should be placed on the victim when performing traceback during an attack.
- Simple and fast algorithms for tracing of routers carrying attack traffic are necessary to identify the furthest attack source points when an attack is ongoing so as to carry out mitigation. [15] showed that 50% and 80% of attacks last for less than 10 and 30 minutes, respectively.
- Traceback mechanisms are triggered by attack detection mechanisms. Existing schemes require precise information on the attack packets, such as attack signatures, to differentiate between legitimate and attack packets and retrieve traceback information from the latter. As brute-force DDoS attacks could flood the victim with seemingly legitimate traffic, identifying an attack signature of the data packets may not be possible so should not be a prerequisite for the traceback methods.

B. Key Assumption

Our design makes the key assumption that end-to-end routes are relatively stable. Analysis of 40,000 end-to-end route measurements conducted using repeated "traceroutes" between 37 Internet sites, is reported in [16]. Two distinct views of route stability, prevalence and persistence, were studied. Prevalence refers to the probability that a certain route is encountered (if a route is observed, how probable are we to observe it again in the future). Persistence refers to routes remaining unchanged over a long period of time (if a route is observed at time t, how long before it may change). In [16], routes were reduced to three different levels of granularity, namely host (each route as a sequence of Internet hostnames), city (as a sequence of geographical cities), and AS (as a sequence of Autonomous Systems). Prevalence of a dominant route (that is, it appears most often) is computed as the ratio of the number of times the dominant route is observed to the total number of traceroutes measuring a particular path. The median value of prevalence is 82%, 97% and 100% at host, city and AS granularity respectively. Therefore, in general, it was concluded that Internet paths were strongly dominated by a single route. It was shown that the time periods over which routes persist demonstrate a wide variation, ranging from seconds to days. However, over 60% of the Internet paths had routes persisting for either days or weeks.

Routing stability based on data captured from the National Internet Measurement Infrastructure (NIMI) and a set of 189 public traceroute servers was studied in [17]. Of the NIMI paths, 78% always exhibited the same route, and 86% of the routes had a prevalence of 90% or higher. For the public servers, the corresponding figures are 73% and 85% respectively. It was shown that very often, routes persist for at least a day, but in general, 1/3 of the Internet routes and 1/6 of the NIMI routes are short-lived.

A study in 2002 [18] investigated whether routing fluctuations caused by the instability of the small fraction of Internet routes affect a significant portion of the Internet traffic. It was concluded that the vast majority of Internet routing instability stems from only a small number of unpopular destinations. Popular destinations, which are responsible for the bulk of the Internet, were shown to have remarkably stable routes lasting days or weeks at times, probably due to the fact that they have reliable and well-managed connections to the Internet.

The above studies showed that Internet routes exhibit relatively high stability in terms of prevalence and persistence. This satisfies the requirement in our scheme that Internet routes would not change erratically and frequently such that the cached information in the white list becomes obsolete. Therefore, we can assume that Internet routes taken by the data packets under normal conditions are generally stable.

III. Network Domain Entrypoint/path Determination

Standard routing protocols perform packet forwarding based on the destination IP address in the packets so packets belonging to a particular source-destination pair follow a relatively static path as routing tables are not updated very frequently under normal conditions. When an attacker spoofs a legitimate user's source address, the packet may pass through routers which are not on the normal source-destination routing path and this anomaly can be used to determine the attack path.

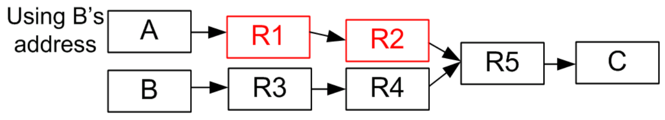

Fig. 1: Route Anomaly

In Figure 1, we show that if node A spoofs node B's address, an "incorrect" path via anomalous intermediate routers can be detected as B to C traffic should flow through R3 and R4, not R1 and R2. By performing source IP address validation checks on whether transit packets are supposed to arrive at particular routers, these packets could be identified as from legitimate or illegitimate users, with a low false positive rate (studied in Section 6). Therefore, even in the event that DDoS attacks constitute seemingly legitimate packets, they would still be traceable.

In our scheme, the routers in the network send sampled transit traffic flow information, using standard flow sampling and reporting mechanisms such as Netflow [19], PSAMP [20]–[22], and IPFIX [23], [24], to their assigned White List (WL) caching device. The flow information includes the source address and port, destination address and port in the original data packets, and the packet's next hop address. In the cache, each record will consist of the above fields, the address of the router that sent the export and time of receipt.

The WL caching devices will update the white lists for the routers during the learning stage (that is, when there is no ongoing DDoS attack). Therefore, spoofed source addresses are prevented from being included in the caches. We assume a DDoS attack would be detected using mechanisms such as TCP SYN flood [25], or MULTOPS [26].

During the attack, traffic sampling at the routers continues and this information is sent to the WL caching devices. However, the white list generation and updates are suspended upon attack detection. The WL caching devices search for mismatches between the sampled traffic and cache data (that is, flows from previously seen sources going through wrong routers which indicate spoofed addresses), and generate partial attack graphs which are sent to the Traceback Manager to generate the full attack graphs. We present two approaches of our traceback scheme, Network Segmentation Based (NSB) and Strategic Points Based (SPB), in the next two sections.

A. Network Segmentation Based (NSB) Approach

In the NSB approach, the network in an administrative domain is divided into segments. Each segment of routers is assigned a WL caching device. During an attack, the Traceback Manager queries the WL caching devices by requesting them to check for specific source/destination address pairs. The WL caching devices send information to the Traceback Manager as to whether the flows with the specified source/destination address pairs have passed through the routers they are in charge of, and if these flows are expected or anomalous. This approach is useful in the case of DDoS attacks whereby the attack signature or attack pattern is identifiable. The detection mechanisms are signature-based and are able to distinguish between legitimate and attack traffic. They are then able to provide information regarding the suspicious source/destination address pairs. However, in the event that attack traffic constitutes seemingly legitimate packets, an attack signature would not be present. This shortcoming is similar to the existing traceback mechanisms which require distinguishing between legitimate and attack packets to conduct tracing. Another problem is the wide range of spoofed addresses and that the chosen source/destination address might not have been captured by all the routers during sampling. Therefore, a set of suspicious source/destination address pairs has to be determined. Nevertheless, this method allows for fast mismatch checking in the event that such attack information is available.

Another solution is to rely on the continuous arrival of flow exports from the routers at the WL caching devices during an attack stage to perform traceback. The WL caching devices will perform checking to identify traffic flows which are not supposed to arrive at the routers they are in charge of (that is, performing router address, traffic source address and destination address matching checks against the white lists). The WL caching devices will then construct partial attack graphs based on these observed anomalies and send them to the Traceback Manager. The Traceback Manager will proceed to perform the complete attack graph generation.

In this case, some routers may see packets from new sources that are not in their white lists which are legitimate requests rather than attack packets. However, such legitimate requests would constitute a relatively minimal percentage of mismatches in comparison to the attack traffic. In the case of attack traffic going through a router, an excessive high number of mismatches would be observed due to the wide range of spoofed addresses and high volume of attack traffic. However, if the attack traffic does not pass through a particular router, the observed number of unknown source addresses due to new legitimate requests will be comparatively small. Our solution is to set a percentage threshold on the number of 'unknown' source addresses seen by a particular router to take into consideration new legitimate requests. The percentage threshold affects the false positives and varies among different organisations. The chosen value or range should depend on the network services provided by the organisations and their normal traffic profile of the arrival rate of new legitimate requests.

We present an example scenario in Figure 2, where we assume the legitimate and attack traffic enter the network through different ingress routers so as to simplify the explanation. We will present simulation scenarios of mixed

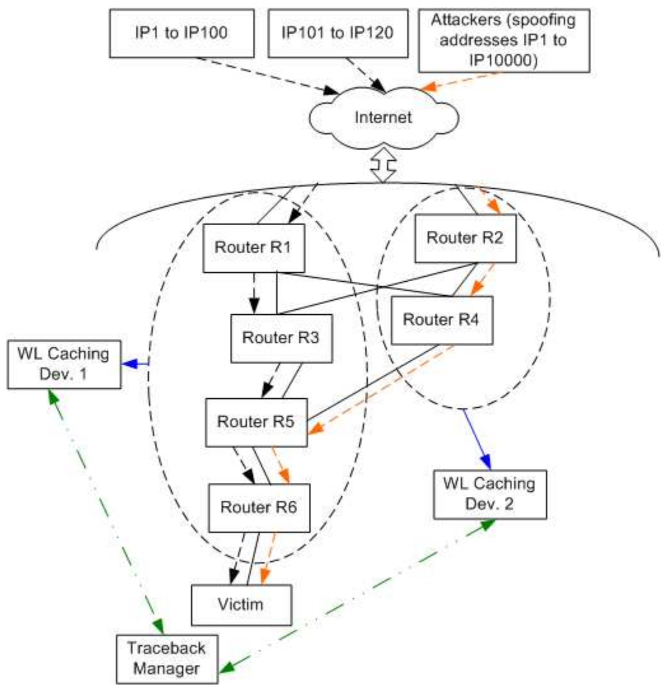

Fig. 2: Network Segmentation Based Approach

traffic coming in through ingress routers, in Section 7.6. As shown, the legitimate traffic is coming from addresses IP1 to IP120. Of these, IP1 to IP100 have visited the site before the attack, whereas IP101 to IP120 are new legitimate requests. Therefore, the WL Caching Device 1 would only have records of IP1-IP100/victim address pairs. Although, mismatches were also observed by WL Caching Device 1 corresponding to routers R1 and R3 "being asked" to forward packets belonging to flows from unknown sources, the number of these mismatches is extremely small.

In the example, the attackers are spoofing the source address of the packets using a wide address range from IP1 to IP10000. Therefore, R2, R4, R5 and R6 would see a sharp rise in the number of new traffic flows. WL Caching Device 1 and 2 will observe a high number of mismatches for the sampled traffic from these routers to the victim. They will then construct the partial attack graphs of R5–>R6 and R2–>R4 respectively, and send these graphs to the Traceback Manager for forming the complete attack graph. In addition, we assume that the Traceback Manager has knowledge of the network topology and the entry and exit points of the network.

B. Strategic Points Based (SPB) Approach

One of the main goals of conducting traceback is to locate the points closest to the attack sources so as to carry out mitigation such as effective filtering or rate-limiting. Therefore, instead of having coverage of all routers within a domain such as a campus network, it would suffice to identify the strategic points, where incoming and outgoing traffic will definitely traverse, and perform monitoring on them instead. To pinpoint the strategic points, we firstly classify attackers into internal (for example, zombies within the victim network) and external attackers. To trace external attackers, the strategic points to perform monitoring or traffic sampling would be at the ingress routers. However, for the internal attackers, we have to know the network topology. We group the internal nodes as intermediate routers and access routers. Monitoring is conducted on the group of access routers.

By reducing the number of routers participating in the traffic sampling and flow exporting, the workload and overhead traffic is significantly reduced. This is a very important enhancement considering that traceback is to be performed during the occurrence of a DDoS attack whereby the victim's network is under heavy load. Another advantage of this approach is that due to the small number of routers involved, a single Traceback Manager with built-in WL caching device functionalities could be in charge of the whole network, therefore consolidating the information storage and processing at a central point. This would allow faster processing and a global view of the traffic flows in the domain, making it easier to identify anomalous flows.

IV. Deployment Considerations

Our method allows for inter-domain support for both the NSB and SPB approaches. In the case of NSB, after construction of the attack graph at the Traceback Manager, it is able to identify the entry point/s of the attack traffic flows into the network. It would then communicate with the Traceback Manager/s of those networks with connecting traffic to these entry point/s, sending them information of the victim's IP address. Assuming these networks can perform traceback, the same attack graph construction process will be carried out at these networks and the completed graph will be sent back to the Traceback Manager at the victim network. This facilitates tracing to the nearest possible point to the source of the attack. In the case of SPB, the similar function performed by the WL caching device is built into the Traceback Manager. Instead of the whole attack graph, the Traceback Managers of the co-operating networks only report the ingress points in their networks, where attack traffic flows are detected. Therefore, SPB would be a more feasible solution as co-operating networks would not have to disclose their internal network topology.

Our traceback method is non-intrusive, in that it does not require changes to the routers assisting in the traceback process. Built-in traffic sampling/monitoring and exporting tools in routers such as Netflow [19], PSAMP [20]–[22] and IPFIX [23], [24] are used to sample and report the required information to the WL caching devices. If such tools are not built into the routers, we can instead make use of monitoring devices by installing them along the network paths.

If the learning process is not suspended in time, records of the attack traffic flow might make it into the white list, thereby corrupting it. The decision as to when to stop the learning process is dependent on the DDoS attack detection mechanism, as it triggers traceback. A solution is to create a separate buffer for the white list. Records of sampled traffic are first written in to the buffer. The interval for the buffer to confirm entries into the white list cache would then be based on the attack detection speed. For example, if the attack detection mechanism takes x secs to detect an attack and the triggering delay (that is, time to inform Traceback Manager of the attack) takes y secs, the buffer flushing interval would be x+y secs for the SPB approach. For the NSB approach, we would also need to take into account the time taken for the Traceback Manager to inform the WL caching devices of the attack.

Another important issue is when do we reactivate the learning process, as we have to make sure that only legitimate traffic is present. Therefore, the detection mechanism which detects the attack and triggers traceback or the response mechanism responsible for mitigating the attack has to ensure that the attack has stopped or successfully been mitigated, to trigger the reactivation of the learning process.

The size of the white list is an important issue to be considered during deployment. For NSB, although all routers are to be monitored, we distribute the work load across multiple WL caching devices. In SPB, the number of routers to be monitored is significantly cut down to allow the use of a single WL cache in the Traceback Manager. As an estimation of the white list size, we referred to [27] which shows that Amazon.com experienced 630,000 visitors in a single hour on its busiest day in 2003. By having a white list cache for a protected server in an IPv4 network, each record would take up 8 bytes of storage (that is, 4 bytes for the source address and another 4 bytes for the router). This converts to 2.4MB of storage for a white list containing the past half hour of records.

V. Simulations

We have carried out simulations to study the performance of our traceback method. Due to the advantages of SPB over NSB approach, we implemented the SPB approach in ns-2 [28] and investigated the performance of the SPB approach during an attack.

During the learning phase, nodes generate legitimate traffic to the target/victim and the Traceback Manager builds the white list. When the attack traffic is started, the white list updating is suspended and traceback is started, but the legitimate nodes continue to generate traffic at a probability (to simulate random traffic). The attackers spoof the source addresses in the attack packets based on a range which also includes the legitimate nodes.

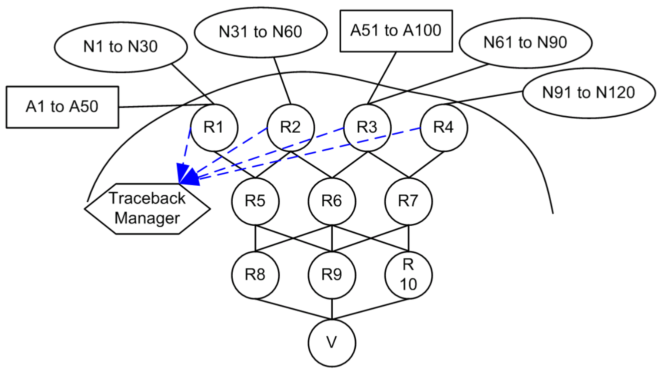

Fig. 3: Traceback using SPB Approach

The network topology is shown in Figure 3. We have 100 attackers (A1 to A100) and 120 legitimate nodes (N1 to N120). The attackers send attack traffic with randomly spoofed addresses in the range of 1 to 10000 (which includes the addresses of the legitimate nodes). The strategic points are R1, R2, R3 and R4, which are the entry points to the network. The links from the legitimate nodes and the attackers into the network are set to 10Mbps with a propagation delay of 30ms to reflect the Internet delays. The internal links are set to 100Mbps with a propagation delay of 10ms.

During the learning phase, each legitimate node N1 to N100, sent traffic to the victim V, at the rate of 5 pkts/sec. R1 to R4 sampled this traffic with probability 0.01 and sent sampled data to the Traceback Manager. The learning period was 20 secs. We ran 3 sets of simulations where the attack started at the 20th sec with rates of 20, 50 or 100 pkts/sec, per attack node. During the 1.5 sec long attack, all legitimate nodes (including N101 to N120 which were simulating new legitimate requests) generated traffic with a "decide to send" probability of 0.5[1] at a rate of 5 pkts/sec per node.

Attack Rate pkts/sec	t ms	0.5 sec	1 sec	1.5 sec (attack stopped)	1.6 sec
20	R1(1) R3(1)	R1(5) R3(4)	R1(11) R3(13)	R1(15) R3(17) R4(1)	Same
50	R1(1) R3(1)	R1(15) R3(10) R4(1)	R1(24) R3(26) R4(1)	R1(34) R3(29) R4(2)	R1(36) R3(41) R4(2)
100	R1(1) R3(2)	R1(23) R3(24)	R1(43) R3(56)	R1(69) R3(91) R2(1)	R1(73) R3(95) R2(1)

TABLE I: Mismatched packets

R1 and R3 were successfully detected to be carrying attack traffic. Table I shows the statistics of the number of mismatch packets traversing the routers detected by the Traceback Manager. The time stated is from the start of the attack and the results are displayed as RX(Y), where X refers to the router's ID and Y refers to the number of mismatch packets detected. The time, t, taken to first detect mismatch packets for both R1 and R3, was 140ms, 80ms and 70ms for attack rates of 20, 50 and 100 pkts/sec, respectively. At t ms, a total of 3, 3 and 4 sampled packets were received by the Traceback Manager, of which 2, 2 and 3 were mismatch packets, for the attack rates of 20, 50 and 100 pkts/sec, respectively.

The results show that there were false positives detected. R2 (for attack rate of 100 pkts/sec) and R4 (for both attack rates of 20 and 50 pkts/sec) were detected to be carrying attack traffic due to mismatch packets detected. These mismatch packets were sampled from the new legitimate traffic not found in the white list. We also observe that as time progresses, false positives started appearing (e.g. 1 mismatch packet for R4 at 0.5 sec when attack rate is 50 pkts/sec). However, the difference between the number of mismatch packets sampled for R1,R3 and R2,R4 widens too. At 0.5 sec, the smallest-

[1]Legitimate traffic during an attack is generated at 5 pkts/sec. However, a random generator is used to determine whether to generate each packet, with a probability of 0.5.

gap ratio (worst case) was 1/10. At 1.6 sec (measurement taken at 1.6 sec to wait for packets due to propagation delay even though attack was stopped at 1.5 sec), the smallest-gap ratio was 1/15, 1/18 and 1/73 for attack rates of 20, 50 and 100 pkts/sec. Threshold values can be set so that these false positives are ignored.

The traceback mechanism was implemented in C++ and deployed in an experimental testbed within the EU funded Diadem Firewall project [29]. Even though a major Internet service provider was one of the partners, we were unable to install the system within a large operational network as the service provider would not permit any DDoS attacks in their network. Therefore, a rather unrealistic small scale test was conducted to test the feasibility of the system and that it could be implemented to work with the router monitoring elements.

The traceback scheme was designed to be used with and triggered by DDoS attack detection mechanisms, so as to perform attack traffic entry and exit point/path locating. To conduct more realistic and large scale experiments, we studied, designed and implemented advanced DDoS attack tools, and attack detection and response mechanisms. The system which incorporated the modules to carry out widely observed DDoS attacks (e.g. TCP SYN flooding) [30], the attack detection modules [31], [32] and our traceback module, is called the DARE (**D**DoS **A**daptive **R**espons**E**) system. Therefore, using DARE, we were able to conduct more realistic experiments in the Emulab environment [33] as described in the next section.

VI. Experiments

We ported DARE with the integrated Traceback module to the Emulab machines to carry out the experiments in this section. DARE is a distributed adaptive DDoS mitigation system composed of modular traffic flow monitoring and aggregating tools, attack detectors and responses, and system managers. Emulab is a large-scale network testbed, giving researchers in the fields of networking and distributed systems a wide range of environments to develop, debug, and evaluate their systems. It allows researchers to specify an arbitrary network topology, giving a controllable environment, including PC nodes with full ”root” access, running an operating system of choice.

The Traceback mechanism was evaluated based on the following criteria.

- Correct execution of the triggering function of Traceback (that is, interoperability with the attack detection module)
- High accuracy or correctness; Traceback should return correct results as to which routers are forwarding attack traffic
- High speed of the tracing, which means returning results within seconds is desirable

In our experiments, we carried out TCP SYN attacks as these are one of the most prominent DDoS attacks [30] and typically use address spoofing, presenting the need for Traceback.

Figure 4 shows the experimental network topology and location of the deployed modules. Nodes 10, 39, 21 and 27 are edge routers at the source networks (due to the

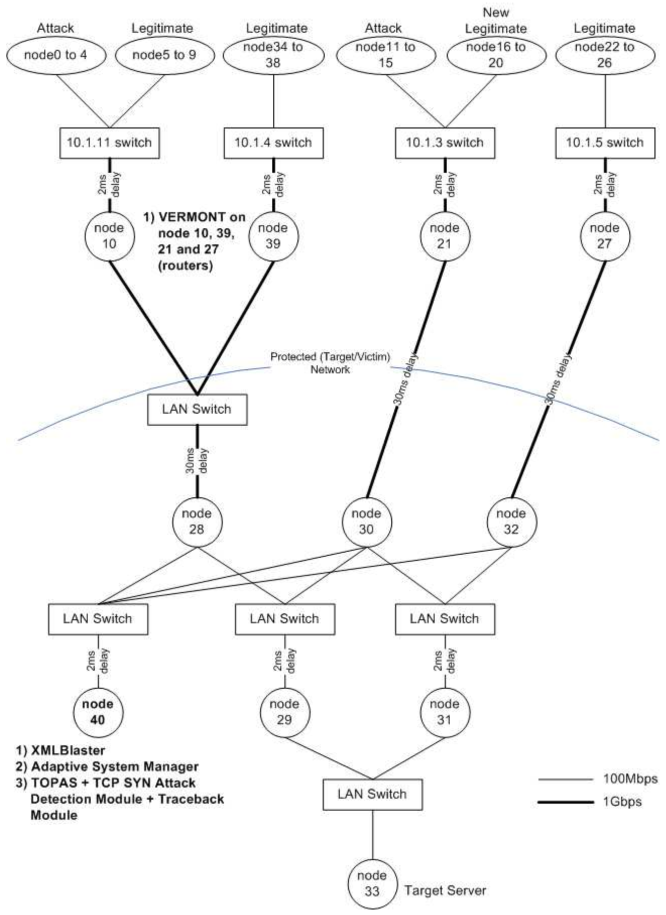

Fig. 4: Experimental Network Topology

TCP SYN attack detection module implemented in DARE, being a source-end based detection mechanism). The Versatile montoring toolkit (VERMONT), which are network monitors configured to capture and/or aggregate traffic packets [34] are deployed on these edge routers.

The traffic packets or flow information from VERMONT is then exported to the Traffic Flow and Packet Analysis System (TOPAS) [35], a framework for the reception and real-time analysis of the network packets and flow information (on Node 40). TOPAS subscribes to flow information essential for DARE's detection modules (also on Node 40) to carry out analysis to detect various attacks in parallel.

VERMONT modules at the source networks send information on outgoing SYN and incoming SYN&ACK packets to the TCP SYN attack detector for analysis. In a TCP SYN attack, the difference between the number of the outgoing SYN and incoming SYN&ACK packets will be very noticeable. The TCP SYN detector in DARE incorporates a Bloom-based hashing technique to allow the detection of both conventional and a new variant of TCP SYN attacks, by only counting validated SYN&ACK packets [31]. It also incorporates a more effective Bloom-based Filter to drop attack packets while protecting legitimate ones at the edge routers.

The Traceback module is triggered after an attack is detected, to perform traceback to identify the closest locations from the attack sources. The module depends on flow information received from VERMONT modules to update its WL during non-attack situations and to check for route anomalies in source and destination IP address pairs during attacks.

The Adaptive System Manager (ASM) is the system coordinator in DARE. Alert messages regarding attacks are sent to the ASM, which then dispatches appropriate responses to the response modules, such as the Traceback module. Results and status information from the response modules are also sent to the ASM for attack status logging.

The communications of the alert and response in DARE (between the ASM, detection modules and response modules) are through Intrusion Detection Message Exchange Format(IDMEF) event messages — an IETF standard which defines an XML message format for intrusion alerts [36]. In DARE, the event distribution between the modules is based on XMLBlaster [37], a publish-subscribe server, where modules connect to it and subscribe to the events they are interested in. When an event is generated, the originating module publishes the event to the XMLBlaster server which in turn forwards it to those modules that have subscribed to the type of events (based on event topics). The use of the XMLBlaster for event distribution allows DARE to scale easily when additional detection and response modules, or even ASMs, are added.

The network topology was designed and structured in a way to consider a mixture of scenarios:

- Two networks (10.1.11 and 10.1.4), with one having a mixture of both attack hosts and legitimate hosts, while the other one having just legitimate hosts. The two networks have different source exit points but share the same network entry point (Node28) into the protected network. After attack detection, Traceback should successfully identify Node 10 as forwarding attack traffic. As the TCP SYN attack is not mounted from the 10.1.4 network, Traceback should not return its exit point (Node 39) as an attack traffic forwarding router.
- A network (10.1.3) having a mixture of both attack hosts and new legitimate hosts, with all traffic exiting from the same source network point and entering the target network through the same entrance point (Node30). The exit point at the 10.1.3 network should be identified by Traceback as forwarding attack traffic.
- A network (10.1.5) having purely legitimate hosts, with all the traffic exiting from the same exit point at the source network and entering through the Node32 entrance point into the target network. In this case, Traceback should not identify the exit point at the source network as forwarding attack traffic.

In the topology (Figure 4), Node0 to Node9 were in the 10.1.11 network, Node34 to Node38 in the 10.1.4 network, Node11 to Node20 in the 10.1.3 network and Node22 to Node26 in the 10.1.5 network. Node0 to Node4, and Node11 to Node15 were attack hosts, while Node5 to Node9, Node34 to Node38, and Node22 to Node26 were hosts sending legitimate traffic (including setting up TCP connections for sending data) to the target server. Node16 to Node20 were hosts configured to send legitimate traffic to the target server after the attack has started so as to emulate new legitimate service requests during an ongoing attack. The 2ms delay on the internal links was set to emulate traffic propagation delays within internal networks. The 30ms delay on the external links was to emulate Internet traffic propagation delays. Internal networks were connected through 100Mbps Ethernet links, while external links were 1Gbps Ethernet.

Multicast servers were implemented and deployed on all the source hosts to listen for experimental control instructions, for example, when to start and stop sending traffic and requests, what rate to send them at and where to send them to.

Here, we provide an overview of DARE's basic operations. When there is no on-going attack, the Traceback module performs WL learning from received flow records. When a new flow is observed, it is entered into a non-committed buffer and resides in the buffer for at least 30 secs. When no DDoS attack is observed after this period, the flow is then committed into the WL as a valid legitimate record.

The TCP SYN Attack Detector subscribes to the SYN and SYN&ACK records to perform bloom-based checking before updating the counts of the packets received and compute the attack indicator value using the CUSUM algorithm [38] to decide if there is an occurrence of a TCP SYN attack. When an attack is detected, the detector generates an IDMEF Alert message about the attack to the XMLBlaster server under an appropriate topic. The ASM and the Traceback response module that have subscribed to the message topic receive the IDMEF Alerts from the XMLBlaster server. Triggering of Traceback and logging are performed by the ASM after parsing and processing the message.

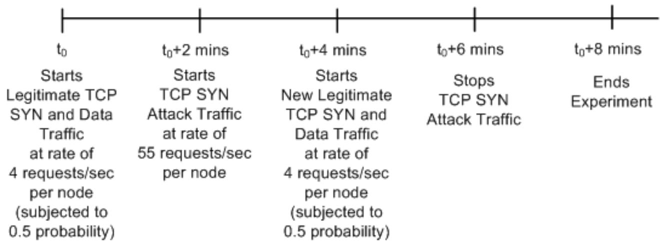
t_0
Starts Legitimate TCP SYN and Data Traffic at rate of 4 requests/sec per node (subjected to 0.5 probability)
t_0+2 mins
Starts TCP SYN Attack Traffic at rate of 55 requests/sec per node
t_0+4 mins
Starts New Legitimate TCP SYN and Data Traffic at rate of 4 requests/sec per node (subjected to 0.5 probability)
t_0+6 mins
Stops TCP SYN Attack Traffic
t_0+8 mins
Ends Experiment

Fig. 5: TCP SYN Attack Experiment Timeline

In the TCP SYN Attack experiments, our Emulab topology specification allocated PC3000 type machines to Node10, Node39, Node21 and Node27, which are running VERMONT. PC3000 machines have 3GHz 64bit Xeon processor, 800MHz FSB and 2GB 400MHz DDR2 RAM. We also specified the allocation of a PC2400c2 type machine to Node40, which is running the XMLBlaster server, the ASM, and TOPAS with the detection and response modules. PC2400c2 machines have 2.4GHz 64bit Intel Core2 Duo E6600 processsors and 2GB 800MHz DDR2 RAM. The rest of the nodes are automatically allocated with either the PC3000 or PC850 type machines. PC850 machines have 850MHz PentiumIII processor and 512MB PC133 SDRAM. PC3000 and PC850 machines are running the Fedora Core4 standard Linux Operating System and PC2400c2 is running the Fedora Core6 standard Linux Operating System.

Figure 5 shows the timeline of the TCP SYN Attack experiments. In a real scenario, legitimate traffic can go on for hours, days or even months without encountering any attack. In our experiments, we allocated 2 mins for the legitimate traffic so that Traceback could learn legitimate traffic flows and update its WL. As the Traceback buffer was set to a delay of 30 secs before committing new records, two minutes of legitimate traffic should be sufficient time for the WL update process. The attack was set to last for 4 mins, with 2 mins of overlap with new legitimate traffic. Research showed that 60% and 80% of DDoS attacks last for less than 10 and 30 minutes, respectively [3]. Therefore, the system should be able to complete traceback within a short time, in this case, within the first 2 mins. The last 2 mins of overlap with new legitimate traffic were used to emulate new connection requests. All the legitimate traffic was stopped 2 mins after the attack to emulate a real legitimate traffic scenario, where transmissions carry on even after an attack.

In the experiments, the legitimate hosts started sending TCP SYN requests to the target server at a rate of around 4 pkts/sec per host (precise packet rate relied on the time delay between sending of each packet and was dependent on each individual system processing speed) and a probability of 0.5[2], at time t_0. At time t_0+2 mins, the attack hosts started sending TCP SYN requests with randomly spoofed source IP addresses to the target server at a rate of around 55 pkts/sec. At time t_0+4 mins, the new legitimate hosts started sending new TCP SYN requests to the target server at a rate of around 4 pkts/sec per host and a probability of 0.5, during the attack. This traffic was therefore newly observed and had not been ”learnt” by the Traceback module. At time t_0+6 mins, the attack was stopped and at time t_0+8 mins, we stopped the experiments.

The Traceback module was configured to only commit records into the WL after a buffer clearance period of at least 30 secs to ensure that the detection module has time to detect an incoming attack, to prevent attack traffic records entering the WL. When an attack is detected, the detector module will send a notification message through the XMLBlaster, to the ASM and the Traceback module to stop the learning process and compute the Traceback results. The ASM will perform attack logging. After the ASM receives the Traceback results, identifying the nodes forwarding attack traffic, the ASM triggers the filtering at the specific edge routers.

We ran a few rounds of the experiments and the experimental results were consistently close. We present two sets of the results as follow.

At time t_0, when the legitimate TCP SYN and data traffic arrived, the Traceback module started to learn the legitimate traffic. At time t_0+2 mins (t_1), the attack hosts started sending TCP SYN flooding packets with randomly spoofed source IP addresses to the target server. The attack was detected by the TCP SYN Attack Detector, and a notification message was sent to the ASM and the Traceback module, at 6 secs after the start of the attack (t_1+6 secs). The Traceback results, identifying Node21 (refer to Figure 4) as the router forwarding the attack traffic, was sent to the ASM at t_1+8 secs. Further Traceback results, identifying Node10 as the other router forwarding attack traffic, was sent to the ASM at t_1+12 secs. Therefore, traceback was accomplished correctly for both attack traffic flows in 2 secs and 6 secs.

In the second experiment, the attack was detected by the TCP SYN Attack Detector, at 5 secs after the start of the attack (t_1+5 secs). The Traceback results, identifying Node10 as the router forwarding the attack traffic, was sent to the ASM at t_1+7 secs. Further Traceback results, identifying Node21 as the other router forwarding attack traffic, was sent to the ASM at t_1+11 secs. Therefore, the correct traceback results were returned in 2 secs and 6 secs.

As shown in both experiments, the Traceback mechanism was able to efficiently and accurately identify the routers forwarding attack traffic, within seconds. In addition, Node39 and Node27, which were not forwarding attack traffic, were not identified by the Traceback.

A. *Discussion*

In Section V, we discussed the ns-2 simulations we performed on the Traceback module. We knew which routers were forwarding the attack traffic and the first mismatch packet passing through the routers was detected within msecs in the simulations. The simulations were carried out to determine how fast the first mismatch packet was captured and how to distinguish between attack and new legitimate packets mismatches to return correct traceback results by the setting of thresholds.

In the experiments, we do not assume knowledge of which routers were forwarding attack traffic as that defeats the purpose of tracing back in a real scenario. The correct traceback results returned in 2 secs to 6 secs in the two experiments were determined based on threshold checks so that the router forwarding new legitimate traffic would not be detected as one with attack traffic traversing it. Mismatched packets at the router for the target server have to exceed a count of 50 and must also exceed 70% of all the mismatched packets at all the forwarding routers before being identified as forwarding attack traffic. The threshold count of 50 mismatched packets is to ensure that initial burst of new traffic (exceeding the 70% threshold due to initial count of 0 packets) would not result in erroneous traceback results. In the simulations, we observed that such a threshold allows differentiation between attack traffic and new legitimate traffic as flooding attack traffic will exceed new legitimate traffic by a large amount. Although lowering the percentage will allow a much faster and correct traceback, we decided that 70% is a much safer setting with not much compromise on the traceback speed.

VII. Threat Analysis and Limitations

In this section, we consider the security issues and limitations of our method.

A. *Security Considerations*

1) Spoofing of IPFIX packets: An attacker might impersonate a monitoring element to send IPFIX packets to the

[2]Although legitimate traffic is generated at a configured packet rate, it takes into consideration a 0.5 probability by incorporating a random generator to determine whether to generate each packet. This is to introduce randomness into the legitimate traffic generation.

collector to inject spoofed addresses into the white list. Subsequent DDoS attack packets, using these addresses, can traverse entry points without being identified by the Traceback mechanism. However, the attacker would need to know the path the attack packets will be taking and also what the identification numbers or IP addresses of the monitoring elements are. Another objective might be to corrupt the white list so as to let Traceback wrongly identify an entry point which is allowing in legitimate traffic and cause self-inflicted DoS. IPFIX packets from illegitimate sources should be prevented from being entered into the white list by using authenticated associations between the monitoring elements and the collector.
2) Spoofing traffic during learning: Before launching an attack, an attacker might send traffic with spoofed source addresses at a normal rate in order to get these into the white list. These IP addresses could then be used in future attacks and would not be detected as anomalies as they are in the white list. A solution would be to monitor for bi-directional flows with established connections. In this case, only such flow records would be committed to the white list.
3) Man-in-the-middle attacks: An on-path attacker could also act as a Man-in-the-middle to change the contents of IPFIX packets to insert spoofed addresses in the white list or corrupt it. Traceback would then fail as a result. To prevent packet modification, Message Authentication Codes to validate the integrity of the contents of IPFIX packets can be used.
4) Eavesdropping attacks: On-path attackers can eavesdrop unencrypted traceback traffic to determine what addresses are going through which entry points. Attackers could then be given legitimate addresses to use as spoofed addresses. The attackers would have to be chosen so that the traffic always enters via the “correct” entry point. Encryption such as IPSec ESP, could be used to guard the confidentiality of the data.
5) Replay attacks: Studies have shown end-to-end Internet routes to be relatively stable. However, in the event of router failures, new routes would be chosen for packet delivery instead. Therefore, the white list would be updated by the IPFIX packets. An attacker might attempt to perform a replay of old IPFIX packets and cause the white list to be corrupted. This threat would result in legitimate traffic coming in from new routes being detected as attacks and cause a self-inflicted DoS. Protection against replay attacks can be achieved by the use of timestamps or nonce to verify that an IPFIX packet has been freshly generated. Alternatively, IPSec could also prevent replay by the use of dynamic keying if support for automatic key management is present.
6) Resource depletion attack on collector: Malicious flooding of the collector with IPFIX packets to deplete processing resources can be prevented if the collector accepts IPFIX packets only from known authenticated monitoring elements.
7) DDoS on Traceback’s components: Attackers might carry out direct DDoS attacks on the Traceback’s components, such as the monitoring elements and the collector. DDoS detection and response mechanisms could be used to protect these vital components. Alternatively, techniques such as port hopping [39] could be used to switch between port numbers at predefined time intervals. Since security associations between the monitoring elements and the collector would already have been set up, it makes the computation of the current port number feasible. Ports not in use could be closed while the one which is dynamically computed and allocated could be used for communication.

B. Limitations

1) Speed of detection mechanism: When the detection mechanism detects the occurrence of a DDoS attack, Traceback is triggered to stop the learning process and start the tracing back to the entry points where the attack traffic is coming in. If this is not done quickly enough, the white list would be corrupted with the information from the attack traffic. Therefore, new entries are entered into a buffer in the Traceback mechanism before being allowed into the white list. New entries are allowed to be transferred into the white list only after a time delay to ensure that no attack traffic is present during this time interval.
2) Speed of traceback: As Traceback is performed in real-time (during the occurrence of an attack), it has to obtain the results before the attack is over. If the attack is too short, Traceback would be unable to complete if the speed of tracing is not fast enough. A solution would be to implement logging to store the attack information to allow for traceback in the event of short attacks, for accountability purposes and even for further post-mortem analysis.
3) Across administrative borders: After Traceback within the victim’s network domain has been performed, the results could be passed on to the adjoining administrative domain, from which the attack traffic is coming. The results could be used for that domain to carry out further tracing back or simply for informative purpose. The forms of communications to be used could be secured emails, phone calls or authenticated and encrypted packets to protect the integrity and confidentiality of the data. The choice of the form of communication would depend on the time of the attack, if near real-time informing is required and whether an automated information or triggering system and secure link has been set up between the two networks.

VIII. Related Work

In the IP logging scheme, the network routers log the passage of all IP packets. The key challenge here lies in the potentially huge amount of information storage requirement. For example, if a router were to log all the packets in its entirety, each OC-192 link at 1.25 GB/s at the router requires 75 GB of storage for a 1-minute query buffer. The storage requirement quickly becomes prohibitive as the number of router links increases. One solution, SPIE (Source Path Isolation Engine) [4], has been proposed for IP version 4. The mechanism is designed to identify the true source of a particular IP packet given a copy of the packet to be traced and an approximate time of receipt. In order to take care of the transformation of packets as they are routed from source to destination, the mechanism identified the invariant portions of the 20-byte IPv4 header. The fields that are susceptible to changes include: TOS (Type of Service), TTL (Time to Live), Checksum and Options field. The logging is based on the invariant portion of the IP header and the first 8 bytes of payload. Based on the statistics collected, the 28-byte prefix described above results in a rate of collision of approximately 0.00092% in WANs and 0.139% in LANs. To further reduce the storage requirement, instead of storing the entire 28-byte prefix, hashing is performed, followed by Bloom filter processing [40]. This reduces the memory storage requirement in the router to 0.5% of link bandwidth per unit time. The disadvantage is that using both the packets' digest (instead of the full packet) and hashing to reduce storage requirement increases the risk of incurring false positives.

In IP marking [5], [9]–[11], [13], [14] schemes, the intermediate routers mark the IP packets with additional information so that the victim can use them to determine the attack path. Approaches proposed include node append, node sampling and edge sampling. The node append mechanism is similar to the IP Record Route Option [41], in that the addresses of successive routers traversed by IP packets are appended to the packets. The victim can thus traceback the source of such attack packets easily. However, this method introduces very high overhead in terms of router processing and packet space. The node sampling approach reduces such overhead by the probabilistic marking of IP packets. The edge sampling approach, as its name implies, marks an edge of the network topology, traversed by the IP packets, instead of just the node. As the IP marking algorithms put the marking information in the Identification field of the IP header, IP marking has an inherent disadvantage in that it "corrupts" the packets by making changes to them during transit and also affects the processing of the IP packets (for example, the Identification field is used for fragmentation purpose). The standardization of the IP marking schemes is thus unlikely due to this problem.

In the ICMP Traceback mechanism [6]–[8], [12], a new ICMP message type, ICMP Traceback (ITrace), is designed to carry information on routes that an IP packet has taken. IP Marking requires overloading some fields in the IP header, which raises the backward protocol compatibility problem. ICMP Traceback utilizes out-of-band messaging to achieve packet tracing and therefore overcomes IP marking's problem. As an IP packet passes through a router, an ICMP Traceback message (ITrace) is generated with a low probability of about 1/20000 for the IP packet and sent to the same destination. Assuming that the average maximum diameter of the Internet is 20 hops, this probability value is to set the upper bound to the net increase in the traffic overhead to 0.1%. This ITrace message is then sent randomly, with a certain probability, to the destination or to the origin of the IP packet. In the event of a DDoS attack, the destination node can then use it to traceback the attack path. The disadvantage of this scheme is that additional traffic overhead will be incurred due to the traceback messages propagating along the routes to the victim. During an attack, this additional load would be undesirable.

In addition, all these existing traceback schemes require wide-spread changes to and deployment on Internet routers. Unless standardization is in place, it will be a long and difficult process for everyone to decide on the scheme to implement. These schemes also rely heavily on the detection mechanism not just to trigger traceback but also to provide them with the original packet for its route to be traced. This requires the detection mechanism to identify attack packets or an attack signature. The IP marking and ICMP Traceback schemes also rely on the victim to receive and construct the traceback path information. This might be a burden on the victim which is already under a DDoS attack. IP marking also require changes to be made to the original data packets and overwriting essential fields might corrupt the original packets.

In contrast, our approach provides a means for performing traceback in a non-intrusive way. Changes to the Internet routers are not required. Constraints are not placed on the detection mechanism to provide it with precise information regarding the attack. The logging and computation tasks are shifted to the WL caching devices and Traceback Manager, relieving the victim from additional burden. Changes to the original data packets are also not required. As the learning phase is conducted before the attack, once the attack is detected, mismatch checking can be conducted at once to determine routers carrying attack traffic. Our algorithm is also simple and efficient, allowing for a fast generation of the attack graph and is scalable due to the distribution of processing workload.

IX. Conclusion

In this paper, we proposed a new non-intrusive traceback technique based on the rationale that packets relating to a particular source-destination flow follow a relatively static path through routers. If an attacker spoofs a legitimate user's address, an "incorrect" path can then be detected. Our traceback mechanism builds caches of valid source addresses (white list generation) for routers at distributed WL caching devices, performs the construction of the attack graph within an administrative domain, and provides an extension for inter-domain support to identify the network point nearest to the attack source.

We proposed two approaches to the Non-Intrusive IP Traceback scheme, namely the Network Segmentation Based (NSB) and Strategic Points Based (SPB) approach. The first approach divides the network into different segments with a WL caching device responsible for each. The second cuts down the number of routers to be monitored and only focuses on the strategic points within the network. Therefore, SPB is able to achieve the traceback objective while reducing the work load and overhead. Simulations were conducted based on SPB performing traceback to locate routers carrying attack traffic. The sampling rate was set to 0.01. Routers forwarding attack packets were successfully located by our traceback scheme in 140ms, 80ms and 70ms for attack rate of 20, 50 and 100 pkts/sec, respectively. We also observed that as the attack rate increases, the detection is faster and the difference in the number of mismatch packets from attack and new legitimate traffic increases due to differences in generation rate. This allows a threshold to be set to ignore low rates of new legitimate traffic.

We implemented our traceback scheme in C++ and integrated it as a response module in the DARE system. The system is deployed in Emulab to perform more realistic experiments to measure the accuracy and speed of our traceback scheme. We conducted a few sets of experiments with close results and presented two sets of them. In a TCP SYN attack scenario, we were able to obtain correct traceback results and achieve traceback speed of 2 secs and 6 secs. The results showed that our scheme is able to achieve traceback accuracy and efficiency in a near real attack scenario.

Due to the differences in the way our system and the other existing traceback techniques are triggered, quantitative analysis and comparison are not practical. However, we presented a qualitative analysis comparing our scheme with other traceback techniques. Our approach is non-intrusive, not requiring any changes to be made to the Internet routers and precise information regarding the attack is not required so we can use a wide variety of DDoS attack detection techniques. The logging and computation tasks are shifted to the WL caching devices and Traceback Manager, and therefore relieving the victim from additional burden. Changes to the original data packets are also not required. As the learning phase is conducted before the attack, once the attack is detected, mismatch checking can be conducted at once to determine routers carrying attack traffic. Our algorithm is also simple and efficient, allowing for a fast generation of the attack graph and is scalable due to the distribution of processing workload.

Acknowledgment

We gratefully acknowledge the support from the EU funded DIADEM Distributed Firewall FP6 IST-2002-002154. This research continued through participation in the International Technology Alliance sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence. We would also like to thank Emulab for providing the facility for the experiments.

References

[1] K. J. Houle and G. M. Weaver, "Trends in denial of service attack technology," CERT Coordination Center, http://www.cert.org/archive/pdf/DoS_trends.pdf, Tech. Rep., 2001 Oct. 2001.

[2] R. Beverly and S. Bauer, "The spoofer project: Inferring the extent of source address filtering on the internet," in *USENIX SRUTI: Steps to Reducing Unwanted Traffic on the Internet Workshop*, Jul. 2005, pp. 53–59.

[3] D. Moore, C. Shannon, D. J. Brown, G. M. Voelker, and S. Savage, "Inferring Internet denial-of-service activity," *ACM Transactions on Computer System (TOCS)*, vol. 24, no. 2, pp. 115–139, May 2006.

[4] A. C. Snoeren, C. Partridge, L. A. Sanchez, C. E. Jones, F. Tchakountio, S. T. Kent, and W. T. Strayer, "Hash-based IP traceback," in *ACM Sigcomm*, Aug. 2001, pp. 3–14.

[5] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical network support for IP traceback," in *ACM Sigcomm*, Aug. 2000, pp. 295–306.

[6] S. Bellovin, M. Leech, and T. Taylor, "ICMP traceback messages," in *IETF Internet Draft, Version 4*, Feb. 2003 (Work in progress).

[7] H. C. J. Lee, V. L. L. Thing, Y. Xu, and M. Ma, "ICMP traceback with cumulative path, an efficient solution for IP traceback," in *International Conference on Information and Communications Security*, vol. 2836. Springer Lecture Notes in Computer Science, Sept. 2003, Oct. 2003, pp. 124–135.

[8] V. L. L. Thing, H. C. J. Lee, M. Sloman, and J. Zhou, "Enhanced ICMP traceback with cumulative path," in *61st IEEE Vehicular Technology Conference*, Stockholm, Sweden, May 2005.

[9] D. Song and A. Perrig, "Advanced and authenticated marking scheme for IP traceback," in *IEEE INFOCOMM*, Apr. 2001, pp. 878–886.

[10] D. Dean, M. Franklin, and A. Stubblefield, "An algebraic approach to ip traceback," in *Network and Distributed System Security Symposium*, Feb. 2001, pp. 3–12.

[11] A. Yaar, A. Perrig, and D. Song, "FIT: Fast Internet traceback," in *IEEE Infocomm*, Mar. 2005.

[12] A. Mankin, D. Massey, C.-L. Wu, S. F. Wu, and L. Zhang, "On design and evaluation of "intention-driven" ICMP traceback," in *IEEE International Conference on Computer Communication and Networks*, Oct. 2001, pp. 159–165.

[13] K. Park and H. Lee, "On the effectiveness of probabilistic packet marking for IP traceback under denial of service attack," in *IEEE INFOCOMM*, Apr. 2001, pp. 338–347.

[14] A. Yaar, A. Perrig, and D. Song, "Pi: A path identification mechanism to defend against DDoS attacks," in *IEEE Symposium on Security and Privacy*, May 2003, pp. 93–109.

[15] D. Moore, G. M. Voelker, and S. Savage, "Inferring Internet denial-of-service activity," in *Usenix Security Symposium*, San Diego, CA, Aug. 2001, pp. 9–22.

[16] V. Paxson, "End-to-end routing behavior in the Internet," in *ACM Sigcomm*, Aug. 1996, pp. 25–38.

[17] Y. Zhang, V. Paxson, and S. Shenker, "The stationarity of Internet path properties: Routing, loss, and throughput," in *ACIRI Technical Report*, 2000.

[18] J. Rexford, J. Wang, Z. Xiao, and Y. Zhang, "BGP routing stability of popular destinations," in *ACM SIGCOMM IMW (Internet Measurement Workshop)*, Nov. 2002.

[19] Cisco IOS Netflow, in *http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html*.

[20] B. Claise, "Packet sampling (PSAMP) protocol specifications," in *IETF Internet Draft, Version 3*, Oct. 2005 (Work in progress).

[21] N. Duffield, "A framework for packet selection and reporting," in *IETF Internet Draft, Version 10*, Jan. 2005 (Work in progress).

[22] T. Zseby, M. Molina, N. Duffield, S. Niccolini, and F. Raspall, "Sampling and filtering techniques for IP packet selection," in *IETF Internet Draft, Version 7*, Jul. 2005 (Work in progess).

[23] B. Claise, "IPFIX protocol specification," in *IETF Internet Draft, Version 19*, Sept. 2005 (Work in progress).

[24] G. Sadasivan, N. Brownlee, B. Claise, and J. Quittek, "Architecture for IP flow information export," in *IETF Internet Draft, Version 9*, Aug. 2005 (Work in progress).

[25] H. Wang, D. Zhang, and K. G. Shin, "Detecting SYN flooding attacks," in *IEEE INFOCOMM*, 2002.

[26] T. M. Gil and M. Poletto, "MULTOPS: a data-structure for bandwidth attack detection," in *10th USENIX Security Symposium*, Feb. 2001.

[27] K. Regan, "Holiday e-tail sales set records despite performance woes," in *E-Commerce Times, http://www.ecommercetimes.com/story/32491.html*, Dec. 2003.

[28] The Network Simulator (ns-2), in *http://www.isi.edu/nsnam/ns*.

[29] "Diadem firewall," in *http://www.diadem-firewall.org*.

[30] Arbor Networks, "Worldwide Infrastructure Security Report," http://www.arbornetworks.com/report, Sept. 2007.

[31] V. L. L. Thing, M. Sloman, and N. Dulay, "Enhanced TCP SYN attack detection," in *IEEE/IST Workshop on Monitoring, Attack Detection and Mitigation (MonAM)*, Toulouse, France, Nov. 2007.

[32] V. L. L. Thing, "Adaptive response system for distributed denial-of-service attacks," in *PhD Thesis, Computing Department, Imperial College London, UK*, Aug. 2008.

[33] Emulab, in *http://www.emulab.net*.

[34] Versatile Monitoring Toolkit (Vermont), in *http://vermont.berlios.de*.

[35] G. Munz and G. Carle, "Real-time analysis of flow data for network attack detection," in *IFIP/IEEE Symposium on Integrated Management (IM)*, 2007.

[36] H. Debar, D. Curry, and B. Feinstein, “The intrusion detection message exchange format (IDMEF),” IETF RFC 4765, Mar. 2007.

[37] xmlBlaster, in *http://www.xmlblaster.org/*.

[38] B. E. Brodsky and B. S. Darkhovsky, *Nonparametric Methods in Change-point Problems*. Kluwer Academic Publishers, 1993.

[39] H. C. J. Lee and V. L. L. Thing, “Port hopping for resilient networks,” in *60th IEEE Vehicular Technology Conference*, Los Angeles, California, USA, Sept. 2004.

[40] B. H. Bloom, “Space/time trade-offs in hash coding with allowable errors,” *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, Jul. 1970.

[41] J. Postel, “Internet protocol,” in *IETF RFC 791*, Sept. 1981.