Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Computer Science 10 (2012) 507 – 516

Procedia Computer Science

The 3rd International Conference on Ambient Systems, Networks and Technologies (ANT-2012)

A Four-Step Technique for Tackling DDoS Attacks

Hakem Beitollahi and Geert Deconinck

Katholieke Universiteit Leuven, Electrical Engineering Department, Kasteelpark Arenberg 10, Leuven, Belgium
{hakem.beitollahi and geert.deconinck}@esat.kuleuven.be

Abstract

This paper proposes a novel feedback-based control technique that tackles distributed denial of service (DDoS) attacks in four consecutive phases. While protection routers close to the server control inbound traffic rate and keeps the server alive (phase 1), the server negotiate with upstream routers close to traffic sources to install leaky-buckets for its IP address. The negotiation continues until a defense router on each traffic link accepts the request (phase 2). Next, the server through a feedback-control process adjusts size of leaky-buckets until inbound traffic locates in a desired range (phase 3). Then through a fingerprint test, the server detects which port interfaces of defense routers purely carry good traffic and subsequently asks corresponding defense routers to remove the leaky-bucket limitations for those port interfaces. Additionally, the server amends size of leaky-buckets for the defense routers proportional to amount of good traffic that each one carries (phase 4). Simulation-based results shows that our technique effectively, defenses a victim server against various DDoS attacks such that in most cases more than 90% of good inbound traffic reaches the server while the DDoS attack has been controlled as well.

Keywords: Denial of service attacks, Network security, Rate limiting

1. Introduction

DDoS attack is a distributed challenge such that bogus packets arrive to the victim from several (e.g., thousands or hundreds of thousands) distributed points in the Internet (i.e., attack: many to one). Consequently, DDoS attack needs a distributed solution which several nodes should cooperate to tackle a DDoS attack (i.e., defense: many to many). Distinguishing DDoS packets from legitimate packets is more accurate when done near the victim server or at the victim server. In most DDoS attacks, the victim server can easily detects DDoS packets as these packets have invalid and bogus payload. On the other hand, the best place to parry the attack is the nodes close to traffic sources because (1) by installing a simple leaky-bucket and adjusting the size of leaky-buckets the large volume of traffic is controlled (2) as at nodes close to traffic sources legitimate traffic is more probable to not be mixed with the attack traffic on the same port interfaces of the routers, less number of nodes need filtering action which it is a high-cost action (3) as any defense node experiences a small part of traffic, the overhead on nodes (from both computation resources and memory point of views) is low. In fact, the closer to the victim server, the worse to tackle DDoS attacks.

One of the main challenges for cooperative techniques is how to find location of attackers and then rate-limit traffic at points close to the source of attacks. To achieve this goal, various IP traceback techniques

1877-0509 © 2012 Published by Elsevier Ltd.
doi:10.1016/j.procs.2012.06.065

based on packet marking (PPM techniques) [1, 2, 3] have been proposed to find location of attackers. These techniques assume that all routers in the Internet mark packets (either probabilistically or deterministically). Hence, a victim server can construct attack paths by collecting significant number of packets. The main challenges of these techniques are: (1) Universal deployment: these techniques need a universal deployment in the Internet to be effective, because all routers in the Internet should mark packets. (2) Scalability: these techniques severely suffer from scalability. Reference [4] shows that the best traceback scheme proposed can only effectively trace fewer than 100 attackers. (3) Cost: all routers in the Internet are involved in deploying these techniques. (4) Survivability of the victim server: all these techniques assume that the victim server can survive during constructing the attack tree which takes few minutes; however, this is not true assumption in realistic.

There are also some other cooperative techniques that are not based on traceback mechanisms. Mahajan et al. [5] studied recursive pushback of max-min fair rate limits, starting from the congested routers toward upstream routers. The most serious challenge with this technique is that legitimate traffic is severely punished as resource sharing is started at points close to the victim server and then recursively pushback to upstream routers. Other secondary challenges with this technique are cost, overhead to routers, no solving mechanism to protect traffic of innocent hosts that share the same paths with the attackers. Yau et al. [6] viewed DDoS attacks as a resource management problem and proposed to distribute bottleneck resource as max-min fashion between level-k routers. K-MaxMin is the improved version of PushBack technique and it could solve some challenges of the PushBack technique. However, there are still some challenges with K-MaxMin. For instance, K-MaxMin is beaten by a meek attack where attackers send traffic rate at the range of legitimate users. K-MaxMin also is beaten by incremental-step attack where attackers incrementally add new zombie machines to the attack process. K-MaxMin also similar to PushBack has no mechanism to protect traffic of innocent hosts that share same path with the attacks. K-MaxMin also assume that all routers of level k (the routers which are K hops away upstream from the victim server) accept to cooperate; but it couldn't be a true assumption.

DefCOM [7] is another cooperative technique that makes an overlay between those routers that participate to detect and stop DDoS attacks. DefCOM installs classifier filters on nodes near traffic sources to distinguish attack packets from legitimate packets and asks other overlay node to rate-limit traffic. D-WARD technology [8] is used as classifer nodes in DefCOM. However, D-WARD is a very high expensive technique that burden large overheads to routers which we believe most routers reject to install firewalls such as D-WARD.

This paper proposes a novel cooperative technique that is based on a feedback-control strategy. Our scheme tackles DDoS attacks in four consecutive phases namely: control phase, negotiation phase, stabilization phase and processing phase (this is why it is called the four-step technique). This technique overcomes all challenges that we enumerated above. The preferences of the our scheme over previous techniques are as follows. (1) It does not require universal deployment; it can be implemented in the todays Internet infrastructure; it is important to know that IP traceback techniques such as PPM techniques are not practical in today's Internet. (2) Unlike PPM techniques, it does not have scalability problem. (3) Unlike PPM techniques, in the four-step architecture, the victim server survives because a protection layer protects the victim server from downing. However, in PPM techniques we cannot have such layer because with this layer the construction of attack trees may take several hours. (4) Unlike the Pushback technique which has the high cost, the cost of our techniuqe is low because only routers of protection and defense layers install leaky-bucket for IP address of the victim server. (5) Unlike the Pushback mechanism that traffic of legitimate users severely punished, in the four-step architecture we do the best-effort to reach traffic of legitimate users without rate-limit to the victim server. (6) Unlike the K-MaxMin technique, the four-step techniuqe fairly adjusts size of leaky-buckets for defense routers proportional to the amount of good traffic that each one carries. (7) Our techniuqe more effectively protects legitimate traffic from rate-limiting than K-MaxMin.

The rest of this paper is organized as follows. Section 2 presents the models. Section 3 discusses the architecture of the four-step technique. Section 4 evaluates the technique through simulation and finally, Section 5 concludes the paper.

2. Models

System model: The total workload that a server desires to receive at a time denotes by L, while the total traffic received at the server at a time is denoted by f. During attack, we desire the victim server receives workload in a range close to the desirable rate, i.e., $L - d \leq f \leq L + d$, where d is a small constant (e.g., 5% of L). The parameter d also is called tolerance parameter which means that the server can tolerate even if it receives d loads beyond L.

Network model: The network is modeled as a directed graph where nodes represent either routers or hosts (traffic sources and servers) and edges represent the links between nodes.

Routers: Any router has several input port interfaces that receives traffic from them. Any port interface is labeled with a local unique number that indentifies the port interface. We call this number the port interface identifier (PID). For more precisely, if a port interface is connected to multiple nodes (e.g., routers or hosts) via a broadcast link-layer channel (such as in a LAN or a hub), then the router assigns virtual PIDs to virtual port interfaces through MAC addresses. In this paper, we assume all routers are trusted.

3. The architecture of the four-step technique

In the four-step technique, two layers of routers are involved in the defense procedure. While a protection layer keeps the victim server alive, the victim server starts negotiation with upstream routers close to traffic sources to compose the defense layer. The fingerprint test which is done through defense routers assists the victim server to have an accurate view about each router of the defense layer and more precisely about each port interface of a defense router. Then the victim server knows the leaky-buckets for which port interfaces should be removed and subsequently which defense routers should remove all leaky-buckets. Additionally, the victim server is able to how fairly amend size of leaky-buckets for the rest of defense routers proportional to the good traffic that each one carries. Our architecture also is able to save traffic of those legitimate users that share the same port interfaces with attackers. To achieve these goals, the four-step technique acts in four consecutive phases as follows:

3.1. Control phase

The goal of the control phase is to control the traffic rate via upstream routers before it overwhelms the bandwidth of the victim server. Before discussing this phase, the following notes are important. (1) The DDoS traffic at the victim server does not reach its maximum instantaneously, but it only after a few seconds [9], because attacker's machines are located at different spots on the globe, implying different latencies. (2) When the rate of bogus packets increases slightly and passes a threshold, a DoS attack can be detected early by the victim server. The above facts provide an opportunity for the victim server to control the traffic rate via upstream routers, which are called protection routers, before the traffic is converged to overwhelm the victim server's bandwidth.

Now, let us explain the protection routers and discuss which routers are protection routers. We consider all edge routers of the ISP that hosts the victim server which include both customer edge routers and inter-provider edge routers as protection routers except the customer edge router via which the victim server connects to the ISP. We denote these routers by C in this paper. The number of protection control routers is denoted by N_c. As all these routers are under the control of a single administrator, the victim server can easily determine the value of N_c in the pre-attack stage. We assume that during normal conditions, the victim server has negotiated with its ISP administrator about the task of protection routers and the administrator has accepted the request of the victim server.

The procedure of the control phase is as follows. The victim server divides L by N_c and then forwards $r_c = L/N_c$ to the protection routers. r_c is the desired rate-limit amount for protection routers. When the protection routers receive this rate-limit amount from the victim server and authenticate the request, they install a leaky bucket of size of r_c for the IP address of the victim server and limit outgoing traffic to the victim server to this rate. Also, the victim server forwards a period value (T) to the protection routers in which each protection router must announce the victim server the total traffic volume that it has toward the victim server. The control phase ensures that the victim server is capable of initiating the defense commands,

because its bandwidth is never saturated. In other words, the traffic load at the victim server is light and the victim server can easily issue control packets and easily receive feedback packets from both defense and protection routers.

3.2. Negotiation phase

In order to save traffic of legitimate users, the victim server should negotiate with upstream routers close to traffic sources to rate-limit traffic for it by installing leaky-buckets for its IP address. The closer defense routers to traffic sources are the better performance is achieved. We note that customer edge routers of different ISPs that have traffic toward the victim server are farthest routers from the victim server and closest routers to traffic sources; therefore, through a traceback mechanism a request for rate-limiting is sent to all customer edge routers of different ISPs that have traffic toward the victim server. In the negotiation phase, anytime a router rejects to install the leaky-bucket for the victim server, the request is passed to all downstream neighbor routers of that particular router through which traffic for the victim server passes. This procedure is continued until a router on each traffic link accepts to install leaky-bucket for the victim server. The routers which accept to install leaky-buckets for the victim server are called defense routers.

The traceback procedure is as follows. Suppose all routers know the public key of their neighbor routers and the customer edge router of the ISP, through which the victim server connects the ISP knows the public key of the victim server. In order to start the traceback mechanism, the victim server generates a message which represents the rate-limit request, signs the message with its private key and sends it to the first router, i.e., customer edge router. The router authenticates the message, signs the message with its private key and sends it to all neighbor nodes. Upon reception of the message by a router, the router first authenticates the message, executes a monitor process aiming at verifying whether traffic for the victim server passes through it. Upon the router detects that the traffic for the victim server is passing through it, it signs the message with its private key and sends it to its neighbor nodes. On the other hand, if the monitor process lasts for a predefined time interval (e.g., 3 seconds) and the router did not detect any packet with the destination address of victim server during this interval, the router terminates the monitor process and does not broadcast the message to its neighbor nodes. This process is continued until the message reaches all customer edge routers of different ISPs that have traffic toward the victim server.

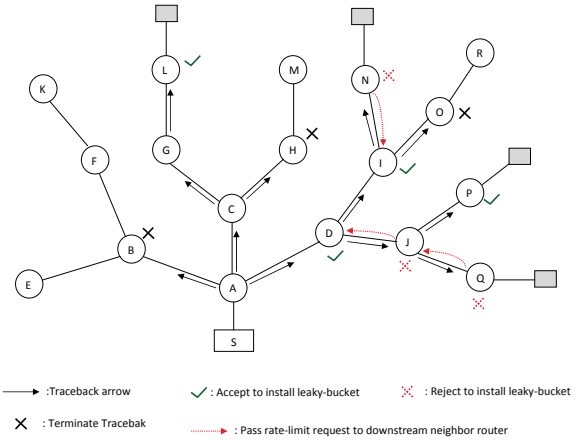

Fig. 1: The procedure of traceback during the negotiation phase

An example: Consider the network graph depicted in Figure 1. The victim server S sends the rate-limit request to router A. Router A broadcasts the message to its neighbor nodes (B, C and D). Router B detects that no traffic passes through it with the destination address of S and thereby does not broadcast the message to its neighbor nodes. Router C and D detect traffic with the destination address of S and each one broadcasts the message to its neighbor nodes. Similarly router G forwards the message to router L while router H does not forward the message to router M as it does not observe any traffic with the destination

address of S. Router I also broadcasts the message to routers N and O and similarly router J broadcasts the message to routers P and Q. As can be seen all customer edge routers that have traffic toward S, i.e., L, N, P and Q receive the message. Now assume that routers L and P accept to install a leaky-bucket for S; but routers N and Q reject to do that. Router N signs the rate-limit request with its private key and passes it to node I. Router I authenticates the packet and accepts to install leaky-bucket for S. Router Q also signs the rate-limit packet with its private key and passes it to node J. Suppose that node J also rejects to install leaky-bucket for S; thereby node J signs the request and passes it to router D. Finally, suppose node D accepts to install leaky-bucket for S.

Stamp the packets: As can be seen in the example depicted in Figure 1, the traffic passing through routers P and I passes through router D as well. In such cases, the traffic of routers I and P must not be rate-limited again in router D. To achieve this goal, any defense router, such as routers I and P, stamps the packets that it allows to pass by embedding 16 zero bits in the IP identification field. On the other hand, defense routers except customer edge routers do not rate-limit the packets which have valid stamp in their IP identification field. For instance, router D does not rate-limit the packets which have 16 zero bits in the IP identification field. The IP identification field is used for fragmentation. However, previous study [1] shows that only 0.25% of packets need fragmentation. It is possible that attackers forge IP identification field and fill out the field with zero pattern. Because of this the victim server asks all customer edge routers that carry traffic toward the victim server to check IP identification field of the destined packets toward it and if the field has non-zero pattern leave it unchanged, but if the field has the zero pattern, overwrite it with a non-zero random pattern. Checking IP identification field and overwriting it with a random non-zero pattern is a cheap operation such that we believe and assume that all customer edge routers that have traffic toward the victim server accept to cooperate.

3.3. Stabilization phase

The goal of the stabilization phase is that the victim server regulate size of leaky-buckets at defense routers such that any defense router sends maximum amount of traffic toward the victim server with this condition that (1) total traffic at the server be between $L - d$ and $L + d$ and (2) no traffic is dropped at the protection routers (i.e., we amend r_c as well). In this case, in the next phase we can receive maximum amount of samples (packets) from defense routers and we are sure that all samples have arrived at the server (i.e., no sample has been dropped at protection routers). The size of a leaky-bucket at defense routers is shown by r_s. To achieve the goal of stabilization phase, the victim server through a feedback-control process amends size of leaky-buckets as follows.

1. The victim server chooses a small initial value for r_s. This amount is randomly selected because in the next round, the amount of r_s is amended appropriately.
2. As discussed above (the control phase), the leaky-bucket size for protection routers is set to L/N_c; i.e., $r_c(1) = r_c(2) = ... = rc(N_c) = r_c = L/Nc$.
3. At the beginning, the victim server selects a small value for T (e.g., $T = 2$ seconds), the period at which any protection router must inform the victim server the total amount of traffic toward it (after processing phase, the victim server amends T to a larger value such as 5 seconds).
4. The victim server multicasts the leaky-bucket size of r_s to defense routers. The victim server also forwards the pair of (r_c, T) to protection routers.
5. Any protection router, say protection router i, and defense router when receives the size of the leaky-buckets, $r_c(i)$ and r_s, respectively, first authenticates it and then installs a leaky-bucket for the IP address of the victim server.
6. Any protection router announces the victim server total traffic volume that it has toward the server with period of T.
7. If the sum of traffic volume reported by protection routers locate between $L - d$ and $L + d$ (i.e., $L - d \leq f \leq L + d$), then the victim server is in stabilization phase and we go to step 10. Otherwise, the victim server amends the size of leaky-buckets for defense routers (r_s) as $r'_s = \frac{r_s \times L}{W(r_s)}$; where r'_s, r_s, $w(r_s)$ and L show amended leaky-bucket size for defense routers, the current leaky-bucket size for

defense routers, the sum of traffic volume toward the victim server reported by protection routers and the desired traffic rate at the victim server, respectively.

8. The victim server also amends amount of r_c for protection routers appropriately. The value of r_c that has been determined for each protection router in the previous step, might be a) more than the reported traffic volume (w_c) in some protection routers, b) equal to traffic volume in some protection routers and c) less than traffic volume in some other protection routers. We can measure extra capacity in each protection router as $r_c - w_c$ if w_c is less than r_c. Sum of all extra capacities is considered as available capacity. In a loop, this available capacity is equally divided among those protection routers that their w_c is bigger than r_c. If some available capacity left and no router needs more capacity, it is divided equally among all protection routers. The loop terminates when available capacity reaches zero or no more protection routers need extra capacity.
9. The server forwards the new amount of r_s to defense routers and forwards $r_c(i)$ to router i of protection routers for $i = 0$ to $i = N_c$. Now the algorithm returns to step 5. The algorithm may then continue in the loop that iteratively adjusts r_s to an appropriate value.
10. In this step, the amount of W, total traffic volume reported by protection routers, has located between $L - d$ and $L + d$; i.e, $L - d \leq W \leq L + d$. In this case, the victim server adjusts the final value of r_c for each protection router according to step 10. The victim server also adjusts T to a higher value, say 5 seconds and then forwards new pair of $(r_c(i), T)$ to each protection router. The algorithm returns to step 5. The algorithm remains in this loop as long as DDoS attack exist.

3.4. Processing phase

The goal of this phase is to precisely isolate good traffic from attack traffic at each defense router at the port interface granularity. This phase, itself, includes 3 steps as follow.

Step 1: Identifying which port interfaces of defense routers purely carry good traffic

First, we should find a simple and cheap mechanism to show that an arrived packet at the victim server has passed through which defense router. Next, we should find another simple and cheap mechanism to show a packet has been entered a defense router through which port interface. Let us first discuss the first issue. The simplest and cheapest technique to understand that a packet comes from which defense router is that we ask defense routers to append 32 bits of hash of their IP address to the end of the payload of packets that they allow to pass the leaky-buckets. We note that defense routers do not need to append 32 bits hash of their IP addresses to dropped packets! They only append 32 bits of hash of their IP addresses to the end of outgoing packets toward the victim server (output of leaky-buckets). Three important questions arise here: (1) is this possible, considering the MTU (Maximum Transmission Unit) bound? (2) Can attackers cheat the victim server by appending hash of forged IP addresses to the end of packets? (3) What should we do if attackers send packets with the size of MTU restriction?

Answering the first question: The MTU is the maximum packet size that a router can pass without fragmenting it. For Ethernet, the MTU size is 1500 bytes. References [1, 10, 11] show that less than 0.25% of packets need fragmentation. The reference [12] shows that the length of more than 80%, 85% and 90% of packets in the Internet is less than 1000 bytes, 1200 bytes and 1400 bytes, respectively. Another study [13] shows that the web request length is 300 bytes with a probability of 0.8 and 1100 bytes with a probability of 0.2. All the above evidences show that there is enough available space for a defense router to append 32 bits hash of its IP address to the end of the outgoing packets. When a defense router appends 32 bits hash of its IP address to a packet, we say so the packet has fingerprint of a defense router.

Answering the second question: An attacker cannot cheat the victim server by appending a hash of forged IP address to the end of a packet because the attacker can always append a hash of forged IP address to the end of a packet before the defense routers. In fact, defense routers are always the last nodes that append hash of their IP address to the end of the outgoing packets.

Answering the third question: We should thanks the attackers if they send their packets with the size of MTU because then their packets can easily be recognized. As discussed above, the size of most of the requests from the legitimate users is less than MTU. In this case, we can ask defense routers to simply

drop packets for which their total length is more than (MTU-64 bits). We note that any packet should have 64 bits available space for our mechanism. The first 32 bits are used to append the hash of IP address of the corresponding defense router and the last 32 bits should be empty to show that the appended hash IP address is the hash IP address of defense routers and not the hash IP address of attackers. An alternative approach for the packets that have size of more than MTU - 64 bits is as follows. The victim server can ask defense routers to log fingerprint of those outgoing packets toward it that their size is more than MTU - 64 bits, and periodically (e.g., every t seconds) send log files for it. Again, we note that defense routers do not log fingerprint of dropped packets, they only log fingerprints of those destined packets to the server that are allowed to pass the routers (output of leaky-buckets). To log fingerprint for a packet, the victim server asks defense routers log a portion (e.g., 160 bits) of the packet as a fingerprint for that packet. On the other hand, when the victim server issued the logging command for defense routers, it also starts to log fingerprints of incoming packets which have a size of more than MTU - 64 bits. The victim server classifies incoming packets and their fingerprints into two categories: good category and attack category. When the victim server receives the log files from the defense routers, for each item of the log file of each defense router, it searches the item in both categories of good and attack. When processing all items of the log file of a defense router was finished, the victim server is able to see that which packets come from that defense router and additionally which packets belong to the good category and which belong to the attack category.

Now let us discuss how to determine which input port interfaces of a defense router carry destined traffic toward the victim server. CAIDA's (Cooperative Association for Internet Data Analysis) Skitter study [14] and another study [15] shows that 98.5% of Internet routers have fewer than 64 working interfaces (including both physical and virtual ports). Hence, allocating 6 bits to show a port interface identifier (PID) is enough. There is a field (8 bits) in the IP header that is called type of service (TOS); the majority of routers ignore to check content of this field [3]. Consequently, to determine that a packet enters a defense router through which port interface, the victim server asks defense routers to copy the PID on which receives a packet through it in the TOS field of that packet.

Analyzing the fingerprints (appended to the packets or in log files) and TOS field of packets assists the victim server to detect which port interfaces of defense routers purely carry good traffic. It also assists the victim server to detect how much good traffic a defense router carries. ***Definition 1:*** those port interfaces which purely carry legitimate traffic are called good port interfaces. Those defense routers that all of their port interfaces are good are called good defense router. ***Definition 2:*** those port interfaces which carry both legitimate and attack traffic are called likely port interfaces and legitimate users along these port interfaces are called poor users. Additionally, those defense routers that carry both legitimate and attack traffic are called likely defense routers. ***Definition 3:*** those port interfaces that totally carry attack traffic are called unlikely port interfaces and similarly those defense routers that totally carry attack traffic are called unlikely defense routers.

Upon identifying a good defense router, the victim server issues a command to the defense router to remove leaky-buckets for all port interfaces and pass traffic without rate-limit. Then, the victim server asks likely defense routers to remove leaky-buckets for good port interfaces and pass traffic coming from those port interfaces without rate-limit. Afterward, in order to rectify these changes, the victim server amends value of r_s and r_c appropriately as discussedin in Section 3.3.

Step 2: Protect traffic of poor users from rate-limiting

The fingerprint test assists the victim server to recognize which packets come from which defense routers. Hence, the victim server can easily detect poor users and their traffic and also it easily detects which defense routers carry traffic of which poor users. To protect traffic of poor users from rate-limiting, the victim server acts as follow. the victim server generates a good traffic signature for traffic of each poor user based on the user's packets' header parameters and then asks the corresponding likely defense router to assign the leaky-bucket capacity to the traffic which matches this signature and then assign the remaining of the leaky-bucket capacity (if anything left) to other traffic. The signature for traffic of a poor user is a pair of source address and source port. The victim server also informs the defense routers, the port interfaces (PIDs) on which they receive traffic of each poor user through it. This also simplifies the task of defense routers.

The probability that an attacker could randomly generate a packet with a specific pair of source address and source port and then the packet exactly passes the corresponding defense router through the corresponding port interface is negligible. Moreover, any defense router needs to assign a very small part of its memory to such signatures because good users are scattered across the globe and any defense router carry traffic of a subset of poor users. For instance, if 500 poor users are along one defense router, that defense router needs to assign around 3KB of its memory to these signatures.

Step 3: Fairly amending size of leaky-buckets

Equal leaky-bucket capacity for likely defense routers and unlikely defense routers is not fair, because likely defense routers carry both legitimate and attack traffic, while unlikely defense routers, only carry attack traffic. Even some likely defense routers may carry more good traffic than other likely defense routers. In fact, the fingerprint test provides the opportunity for the victim server to detect which defense router carries which amount of good traffic. A likely defense router may need more capacity to pass legitimate traffic. In other words, legitimate traffic rate at likely defense routers might be bigger than the leaky-bucket capacity. While, in unlikely defense routers, the leaky-bucket capacity is consumed only by attack traffic. So, we must fairly amend size of leaky-buckets for defense routers proportional to the amount of good traffic that they carry. Let us discuss how the victim server can fairly amend size of leaky-buckets for defense routers. Suppose after removing leaky-bucket limitation for good defense routers, the available bandwidth for the remaining defense router is B_w. We can assign 10% of B_w to unlikely defense routers and 90% of B_w to likely defense routers. 10% of B_w is equally distributed between unlikely defense routers. For instance, suppose there are n unlikely defense routers; thereby the leaky-bucket capacity for each unlikely defense router would be $B_w/(10 \times n)$. On the other hand, 90% of B_w is divided between likely defense routers proportional to the amount of good traffic that they carry. Suppose total arrival traffic from all poor users is G and assume likely defense router i carries g_i good traffic. Therefore, the leaky-bucket capacity for each likely defense router is calculated as $r_{s_i} = \frac{9 \times g_i}{10 \times G} \times B_w$.

4. Simulation results

To examine system performance of the four-step technique, we conduct experiments using the ns2 simulator. We construct a global network topology from real traceroute data. The traceroute data set is obtained from the Internet mapping project at Lumeta [16], the collected data of February 8th 2006. All paths start from a single router, which is the victim server's edge router in the simulation. This edge router has a degree (number of input port interfaces) of six. The resulting graph has a total of 5472 nodes, of which 2006 are hosts. These hosts have been scattered across 200 distinct customer networks. In our simulation, we assume, these 2006 hosts access the victim server either as an attacker or a legitimate user. Our analysis show that the average router's degree in the graph is 16 (i.e., number of port interfaces) and the lower bound and upper bound is 1 and 32, respectively. We choose routers 2 hops away from the victim server as edge routers of the victim server's ISP. These edge routers are protection routers. Number of protection routers is 10. The victim server's link bandwidth is 100 Mbps. We perform experiments for various attack scenarios: diffuse attacks, sparse attacks, meek attacks and incremental step attacks. To examine the performance of the four-step technique, we replicate experiments with K-MaxMin and compare our scheme with K-MaxMin.

4.1. Diffuse attacks

In this experiment, 60% of hosts assume to be legitimate users (i.e., 1203 users) and the remaining 40% are attackers (i.e., 803 users). Legitimate users and attackers evenly distributed across the network. In this experiment, legitimate users and attackers send packets to the victim server at a rate from the range [20, 80] kbps and [200, 800] kbps, respectively. Figure 2.a shows percentage of the bandwidth that has been occupied by good traffic versus time. When there is no attack, on average 60% of the victim server's bandwidth is occupied by good traffic; while when there is an attack against the victim server, good traffic can only occupy 20% of the bandwidth. When the four-step technique is used for defense, system is stabilized at time equal to 8 seconds which at this time 70% of good traffic is survived; upon stabilization, the fingerprint and processing phase is started; thereby after elapsing 24 seconds from the beginning of attack, more than

96% of good traffic is survived; i.e., 57.8% of victim server's bandwidth is occupied by good traffic. While, in the K-MaxMin case, maximum 65% of good traffic is survived which can be obtained after elapsing 140 seconds of the beginning of the attack. As can be seen in the figure, our scheme saves good traffic much better (96% vs. 65%) and very faster (8 seconds vs. 140 seconds) than K-MaxMin approach.

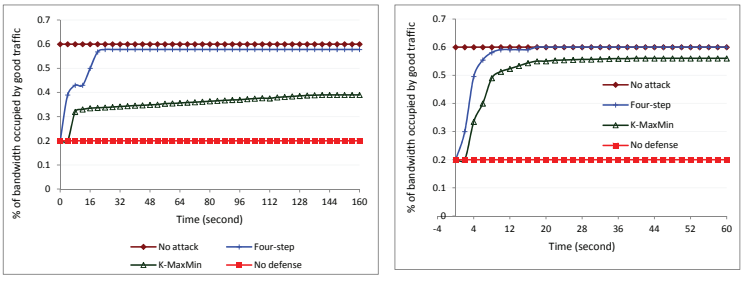

Fig. 2: Percentage of the bandwidth occupied by good traffic during a) diffuse attack, b) Sparse attack.

4.2. Sparse attacks

In this experiment also 60% of hosts are legitimate users and 40% of hosts are attackers, but while legitimate users are evenly distributed across the network, attackers cumulate in few places. It means that 1203 legitimate users are evenly across 200 customer networks but 803 attackers distributed only across 30 customer networks. Legitimate users and attackers send packets towards the victim server from the same range as previous experiment. Figure 2.b shows percentage of the bandwidth occupied by good traffic when the victim server is under sparse attack. With the four-step technique, 100% of good traffic is survived after elapsing 18 seconds of the beginning of attack, while with K-MaxMin about 93% of good traffic is survived after elapsing 40 seconds of the beginning of attack. As can be seen, both four-step and K-MaxMin are promising techniques against sparse attacks.

4.3. Meek attacks

Attackers may resort to stealthy techniques to avoid attack detection by sending traffic at the range of legitimate users. In this case, attackers must compromise large number of zombie machines to make such attack. In this experiment, 30% of hosts assume to be legitimate users and 70% of hosts assume to be attackers. All users are evenly distributed across the network. Both legitimate users and attackers send traffic at a rate from the range [40, 100] kbps. Figure 3.a shows percentage of the bandwidth occupied by good traffic versus time. As the figure shows, with our scheme 93% of good traffic survives after elapsing 20 seconds of the beginning of attack; while with K-MaxMin only 28% of good traffic survives. The interesting point is that survival ratio of K-MaxMin even is lower than when there is no defense mechanism.

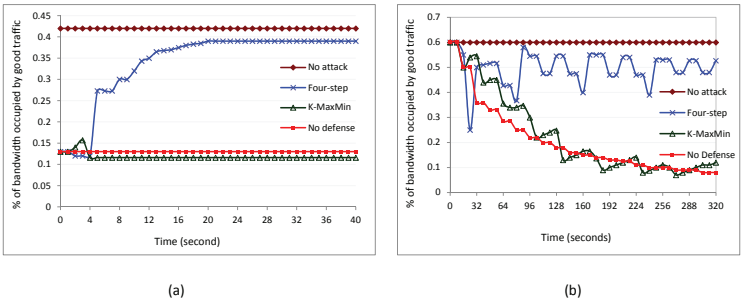

Fig. 3: Percentage of the bandwidth occupied by good traffic during a) meek attack, b) incremental step attack.

4.4. Incremental step attacks

Sophisticated attackers may attempt degrade performance of the system by incremental step attacks in which an attacker does not activate all attack machines at once. The attacker first activates n machines, then after elapsing each τ seconds, activates n more machines. The number of activated machines at each time can be the same or different. The frequency of activating new machines can be fast or slow. The plausible problems with these attacks are that they might generate additional control load, from calculating new rate-limit amount point of view and perhaps increase collateral damage to good traffic. Due to page limit we only discuss an incremental step attack with one frequency rate (every 10 seconds 5 new zombie machines are activated). Legitimate users send traffic at a rate from the range [35, 115] kbps and attackers send traffic at a rate from the range [400, 1200] kbps. Figure 3.b shows percentage of the bandwidth occupied by good traffic during this incremental step attack. As can be seen in the figure, performance of the system with K-MaxMin significantly decreases with this attack. Interestingly, at time 130 onward, no defense system has the same performance as K-MaxMin has. The four-step technique is much better than K-MaxMin encountering this sophisticated attack; though its performance is slightly decreased. In fact, incremental step attacks are more danger against rate-limit approaches that have been proposed for controlling DDoS attacks.

5. Conclusion

This paper proposes a novel and systematic cooperative defense mechanism based on cooperation of the victim server with two types of upstream routers: (1) edge routers of its ISP (i.e., protection routers) and (2) routers close to traffic sources (i.e., defense routers). The four-step technique tackles DDoS attacks in four consecutive phases which are called control phase, negotiation phase, stabilization phase and processing phase, respectively. The performance of our scheme is evaluated for four attack models using simulation and also it is compared with K-MaxMin. The results show that the four-step technique is significantly better than K-MaxMin and it can effectively protect a victim server from DDoS attacks.

References

[1] S. Savage, D. Wetherall, A. R. Karlin, T. E. Anderson, Network Support for IP Traceback, IEEE/ACM Transactions on Networking 9 (3) (2001) 226–237.
[2] M. Goodrich, Efficient Packet Marking for Large Scale IP Traceback, in: Proceedings of the 9th ACM conference on Computer and communications security, 2002, pp. 117–126.
[3] D. Dean, M. Franklin, A. Stubblefield, An algebraic approach to ip traceback, ACM Transactions on Information and System Security 5 (2).
[4] M. Sung, J. Xu, J. Li, L. Li, Large-Sacle IP traceback in High-Speed Internet: Practical Techniques and Infromation-Theoretic Foundation, IEEE/ACM Transactions on Networking 16 (6).
[5] R. Mahajan, S. M. Bellovin, S. Floyd, Controlling high bandwidth aggregates in the network, ACM SIGCOMM Computer Communication Review 32 (3) (2002) 62–73.
[6] D. Y. Yau, J. C. S. Lui, F. Liang, Y. Yam, Defending against Distributed Denial-of-Service Attacks with Max-Min Fair Server-Centric Router Throttles, IEEE/ACM Transactions on Networking 13 (1) (2005) 29–42.
[7] J. Mirkovic, M. Robinson, P. Reiher, Forming Alliance for DDoS Defenses, in: Proceedings of the New Security Paradigmes Workshop, 2003.
[8] J. Mirkovic, P. Reiher, D-WARD: A Source-End Defense against Flooding Denial-of-Service Attacks, IEEE Transactions on Dependable Computing 2 (3) (2005) 216–232.
[9] M. Long, C. Wu, J. Hung, Denial of Service Attacks on Network-Based Control Systems: Impact and Mitigation, IEEE Transactions on Industrial Informatics 1 (2) (2005) 85–96.
[10] K. Claffy, S. McCreary., Sampled Measurements from June 1999 to December 1999 at the AMES Inter-exchange Point, Personal Communication (January 2000).
[11] I. Stoica, H. Zhang, Providing Guaranteed Services Without Per Flow Management, in: Proceedings of the 1999 ACM SIGCOMM Conference, Boston, MA, USA, 1999, pp. 81–94.
[12] Y. Cheng, V. Ravindran, A. Leon-Garcia, Internet Traffic Characterization Using Packet-Pair Probing, in: Proceedings of 26th IEEE International Conference on Computer Communications (INFOCOM), Anchorage, AK, 2007, pp. 1766–1774.
[13] B. A. Mah, An Empirical Model of HTTP Network Traffic, in: Proceeding of Infocom'97, Kobe, Japan, 1997, p. 592600.
[14] The Cooperative Association for Internet Data Analysis (Skitter Project), http://www.caida.org/tools/measurement/skitter/, visited in August 2011.
[15] B. Al-Duwairi, T. E. Daniels, Topology Based Packet Marking, in: Proceedings of International Conference on Computer Communication Networks (ICCCN'04), 2004, pp. 146–151.
[16] Lumeta, http://www.lumeta.com/research/, visited on September 2011.