Honeypot Back-propagation for Mitigating Spoofing Distributed Denial-of-Service Attacks *

Sherif Khattab[1], Rami Melhem[1], Daniel Mossé[1], and Taieb Znati[1,2]
[1]Department of Computer Science
[2]Department of Information Science and Telecommunications
University of Pittsburgh, PA 15260
{*skhattab, melhem, mosse, znati*}*@cs.pitt.edu*

Abstract

The Denial-of-Service (DoS) attack remains a challenging problem in the current Internet. In a DoS defense mechanism, a honeypot acts as a decoy within a pool of servers, whereby any packet received by the honeypot is most likely an attack packet. We have previously proposed the roaming honeypots *scheme to enhance this mechanism by camouflaging the honeypots within the server pool, thereby making their locations highly unpredictable. In roaming honeypots, each server acts as a honeypot for some periods of time, or* honeypot epochs*, the duration of which is determined by a pseudo-random schedule shared among servers and legitimate clients.*

In this paper, we propose a honeypot back-propagation *scheme to trace back attack sources when attacks occur. Based on this scheme, the reception of a packet by a roaming honeypot triggers the activation of a DAG of* honeypot sessions *rooted at the honeypot under attack towards attack sources. The formation of this tree is achieved in a hierarchical fashion: first at the Autonomous system (AS) level and then at the router level within an AS if needed. The proposed scheme supports incremental deployment and provides deployment incentives for ISPs. Through ns-2 simulations, we show how the proposed scheme enhances the performance of a vanilla* Pushback *defense by obtaining accurate attack signatures and acting promptly once an attack is detected.*

1 Introduction

The Internet has witnessed a proliferation of services, some of which are publicly accessible, such as google.com and DNS, whereas others, such as subscription-based services, are private to specific user communities. Meanwhile, Distributed Denial-of-Service (DDoS) attacks continue to pose a real threat to Internet services[10, 9, 11], and the arsenal of DDoS attackers is full of different mechanisms[25]. For instance, certain fields of attack packets may be spoofed, or forged, to complicate detection and isolation or to hide attack sources.

Many schemes have been proposed to defend against source-address spoofing DDoS attacks [15, 24, 33, 7, 36, 27, 13]. However, difficulties of widespread deployment reduce the effectiveness of ingress filtering[15]. Whereas traceback schemes [33, 7, 36, 27, 13] can identify the real sources of spoofed attack packets, it is still needed to accurately find these attack packets and to take an action against their sources: to stop them for instance. By enforcing aggregate-based congestion control, the Pushback mechanism is effective to some extent in the containment of DDoS attack traffic. However, in some attack scenarios, such as highly dispersed DDoS attacks, Pushback collaterally damages legitimate traffic sharing paths with attack traffic[24].

Honeypots are physical or virtual machines successfully used as intrusion detection tools to detect worm-infected hosts for example [30, 23, 37]. The *Roaming honeypots* scheme has been proposed as a defense against non-spoofed service-level DoS attacks[21]. Without causing service interruption, it allows the camouflaging of honeypot locations within a pool of server replicas by randomly designating a set of servers to be active over a period of time and using the remaining servers as honeypots. In other words, each server in the pool, in coordination with legitimate clients and remaining peer replicas, assumes the role of a honeypot for specific intervals of time. We call these intervals *honeypot epochs*. Roaming makes it difficult for attackers to identify active servers, thereby causing them to be trapped in the honeypots.

The focus of this paper is on defending *private* services against source-address spoofing DDoS attacks. In private services, legitimate clients are either known a priori, such as in telecommuting to an enterprise network, or they subscribe to the private service to acquire access to the network. In either scenarios, legitimate clients should be able to access the service from *anywhere* in the Internet [18]. To address this problem,

*The authors were supported in part by NSF under grant ANI-0087609.

1-4244-0054-6/06/$20.00

we combine the effectiveness of Pushback mechanism for tracing back and controlling attack traffic to its sources, and the roaming honeypots ability to accurately and promptly detect attack signatures.

The proposed *honeypot back-propagation* is a hierarchical traceback scheme, which effectively traces back to and *stops* sources of attack streams without significant impact on the performance of legitimate traffic streams. The main idea of the scheme is that each roaming honeypot that receives packets initiates a recursive traceback process by alerting Autonomous Systems (ASs) across the path(s) towards attack sources. The alert triggers the AS-level input-debugging [38] process on traffic *destined* for the honeypot, and further propagates honeypot activations upstream towards attack sources. Moreover, within each AS, access routers of attack hosts are identified and filtering rules are installed to drop all traffic destined to the honeypot. The ability of honeypot back-propagation to accurately distinguish attack packets from legitimate ones enables the aggressive action of packet dropping, as opposed to rate limiting, against attack traffic without penalizing legitimate traffic.

As a large number of attack sources participate in a DDoS attack, it becomes increasingly difficult to respond promptly, even when true IP addresses of the attackers are known. This is due to (1) the attack sources may belong to many administrative domains and (2) it may not be possible to block the attack sources based on their IP addresses as these addresses can be assigned to legitimate machines later on.

To address the above issues, widespread deployment and cooperation among ISPs are required. Our honeypot back-propagation provides a high payoff in this regard. First, collateral damage is reduced. Moreover, filtering in honeypot back-propagation is based on the victim's destination address, therefore, it creates relatively few management constraints. This destination-based filtering allows our proposed scheme to scale to a large number of attack sources, and it avoids blocking legitimate machines that get assigned IP addresses previously allocated to attack sources, thus, addressing the second reason of delayed action.

The rest of the paper is organized as follows. In the next section we review some of the related DDoS defenses. Section 3 presents service and attack models. For completeness, we describe the roaming honeypots framework, on which we build honeypot back-propagation in Section 4. Section 5 describes the hierarchical honeypot back-propagation scheme with inter-AS and intra-AS components and discusses some design issues. In Section 6, we describe our ns-2[5] model of the honeypot back-propagation scheme and analyze its benefits. Section 7 concludes the paper.

2 Related Work

In this section, we review some of the related defense systems of the spoofing DDoS attack. If widely deployed, ingress filtering [15] and IPsec [17] can prevent most spoofing attacks. However, the management hassle and per-packet performance overhead are obstacles against widespread adoption of ingress filtering and IPsec, respectively. Maintaining ingress filtering rules can incur an overhead, especially with mobile IP[28] support. In honeypot back-propagation, filtering is based on destination addresses and kicks in only when attacks are detected. Thus, it suffers from less management hassles. The honeypot back-propagation scheme requires light-weight per-packet filtering, and, moreover, this filtering is performed only in the case of attacks.

The SOS architecture [18] tackles the same problem as ours: DoS attack in the context of a private service with predetermined clients. It uses an overlay network to hide the locations of a small number of proxy nodes (servlets) and allows only traffic from these servlets to enter the protected network. In order to gain access to the overlay network, a client has to authenticate itself with one of the replicated access points (SOAPs), which routes each client packet to one of the servlets using hash-based routing. The overhead of the overlay routing can be up to 10 times the direct communication latency[18]. Our work aims at providing a more effective solution by avoiding overlay routing and by taking actions only when attacks occur.

CenterTrack elegantly uses tunneling over an overlay network to determine ingress points of attack traffic [38]. Controlled flooding injects packet floods into the network and detects attack paths based on traffic perturbations[8]. Packet marking (e.g., [33, 7, 36]) and packet logging [31, 35] represent other approaches to the traceback problem. Hierarchical traceback [26] with inter- and intra-domain traceback mechanisms is similar to our hierarchical honeypot back-propagation approach. However, our scheme differs in that honeypot traceback is triggered only upon attack detection.

Traceback information can be used for filtering at the victim. For instance, StackPi is a deterministic packet marking scheme that allows the victim to locally filter attack packets based on the mark field[29].

Pushback propagates an aggregate-based rate-limiting filter upstream from a congested router[16]. Both detection of misbehaving aggregates and assignment of rate limits are done using the Aggregate-based Congestion Control (ACC) mechanism[24]. At each Pushback-enabled router, the rate limit of an aggregate is shared in a max-min fairness fashion among input ports on which traffic matching the aggregate signature is received. To estimate the arrival rate of each input port, a feature similar to input debugging is used to map each packet at the output queue to its corresponding input port. Pushback accepts misbehaving aggregate signatures through special request messages. Honeypot back-propagation scheme can be viewed as a realization of this feature; when a server takes the role of a honeypot, the server's destination address forms the malicious aggregate.

Level-k max-min fairness [41] addresses the drawbacks of the hop-by-hop application of max-min fairness, which can severely punish legitimate traffic sharing aggregate signatures with misbehaving traffic (see Section 6.3.1). Client puzzles at the IP level repre-

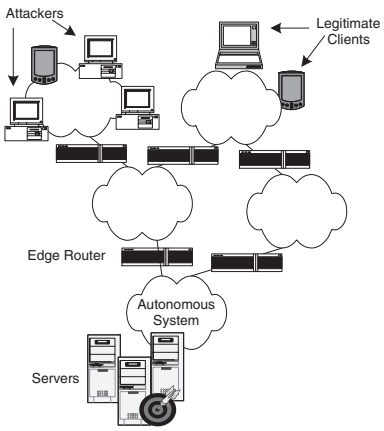

Figure 1. The service consists of a pool of servers. Legitimate clients access the servers from anywhere. Attackers send spoofed attack traffic to the servers.

sent a promising direction in suppressing DDoS attack streams[12, 39]. They can more effective when provided with accurate attack signatures, such as the ones provided by our scheme. Finally, the Mohonk, or mobile honeypots, scheme allows for propagation of honeypot addresses using BGP options [22]. However, these honeypot addresses are simply dark address spaces. Our scheme camouflages honeypot addresses within servers so as to make it difficult for attackers to discover and avoid sending traffic to honeypots.

3 The Spoofing DDoS Attack

We consider a private service as depicted in Figure 1, whereby legitimate clients are either known a priori or acquire access to the service through on-line subscription. Legitimate clients should be able to access the service from anywhere in the Internet. The service is provided by a pool of replicated servers.

A variety of attacks can be staged against the private service. In this study, we assume that attacks can be launched from a number of attack hosts, or zombies, by sending spoofed packets destined for the servers[1]. The attack may result in the following: (1) blocking of legitimate connection requests from reaching the servers and (2) degrading the throughput of both TCP flows from servers to clients as well as data flows from clients into servers. For example, if TCP ACK packets from clients to servers get dropped due to the attack, the throughput of TCP flows is degraded.

4 Roaming Honeypots Background

The proactive server roaming scheme has been proposed in [19], where a prototype of the scheme is evaluated, and has been studied through simulation in [32]. The roaming honeypots scheme leverages proactive server roaming to camouflage honeypot locations within a pool of servers, so as to make it difficult for attackers to direct their traffic away from the honeypots and to avoid detection [21]. The scheme allows for k out of N servers to be concurrently active, whereas the remaining $N-k$ act as honeypots. The locations of the current active servers, and, thus, the honeypots, are changed according to a pseudo-random schedule shared among the servers and legitimate clients. Therefore, legitimate clients always send their service requests to active servers, whereas attack requests may reach the honeypots. The source address of any request that hits a honeypot is blacklisted, so that all future requests from this source are subsequently dropped. The source address is not blacklisted unless a full service handshake is recorded to ensure that it is not spoofed.

The pseudo-random schedule divides time into *epochs*, whereby at the end of each epoch, the set of active servers changes by migrating active connections from one server to another. A long hash chain is generated using a one-way hash function, and used in a backward fashion. The last key in the chain, K_n, is randomly generated and each key, K_i $(0 < i < n)$, in the chain is computed as $H(K_{i+1})$ and used to determine the active servers during epoch i.

Upon subscription to the service, each legitimate client is assigned a roaming key, K_t from the hash chain, with a varying value of t according to each client's trust level and/or other policies. K_t acts as a *time-based* token, whereby it allows the client to track the service up to and including epoch t. Clients also receive the list of servers. When subscription expires, that is, the current service epoch exceeds t, the client may contact the subscription service to acquire a new key.

Roaming honeypots require loose clock synchronization of servers and legitimate clients. That is, the clock shift among system components is bounded by a constant, δ. To maintain loose clock synchronization, clients synchronize their clocks with servers at each service request. However, if a client remains inactive for a long period of time, it contacts the subscription service for resynchronization. At the server side, each service epoch starts earlier by δ at the new servers and ends later by $\delta + \gamma$ at the active servers of the previous epoch, where γ is an estimated communication delay from clients to servers.

5 Honeypot Back-propagation

Our honeypot back-propagation extends the roaming honeypots scheme to defend against source-address spoofing DDoS attacks. As described in the previous section, servers alternate between providing service and acting as honeypots according to a shared pseudo-random schedule [21]. Each server S enters a *honeypot epoch* as soon as it is scheduled to be inactive. During a honeypot epoch, S expects to receive no legitimate traffic, therefore, any packet destined for S is most likely an attack packet. A honeypot epoch

[1]We assume that legitimate clients are not compromised.

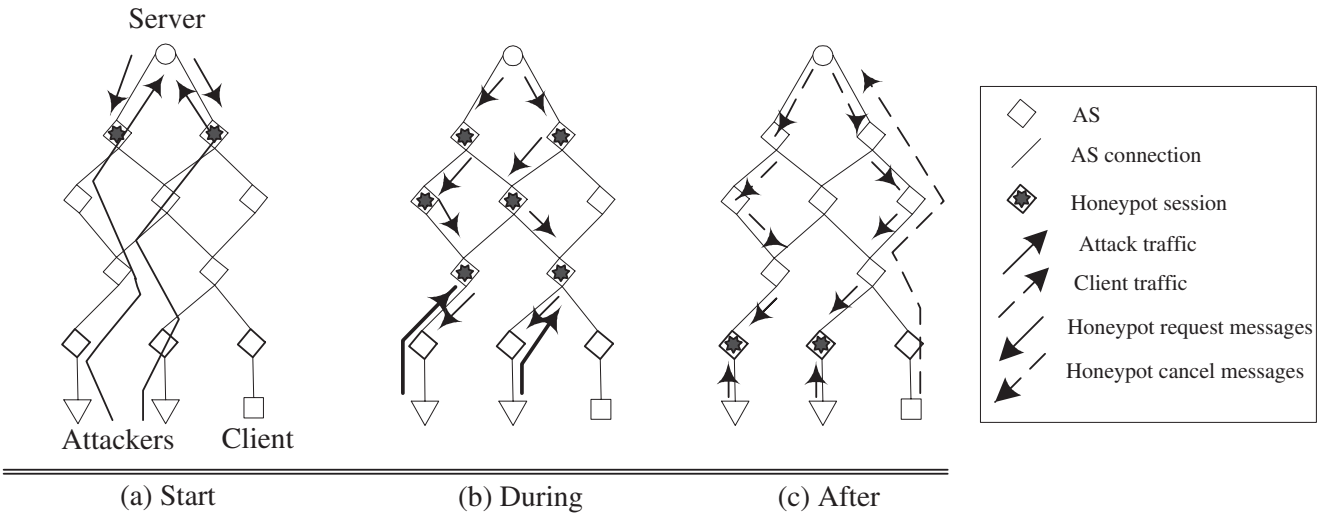

Figure 2. The operation of inter-AS honeypot back-propagation at the start of, during, and at the end of each honeypot epoch. Honeypot sessions are maintained and propagated by honeypot session managers (HSM) inside ASs. (a) While acting as a honeypot and upon reception of attack packets, a server S sends honeypot request messages to the HSM(s) in its home AS(s). When a HSM receives a honeypot request message, it creates a honeypot session. (b) During honeypot epochs, HSMs propagate honeypot sessions upstream towards attack sources. (c) At the end of honeypot epochs, S sends honeypot cancel messages to tear down honeypot sessions except at stub ASs hosting attackers.

ends once S becomes active again. As described earlier, these honeypot epochs are selected in coordination among servers and legitimate clients so as not to cause service interruption.

In other words, honeypot epochs are time windows, in which a server receives a stream of pure attack packets, which we term *honeypot traffic*. In honeypot back-propagation, we trace the honeypot traffic back to its origin(s) and install filters as close as possible to the attack sources.

5.1 Inter-AS Propagation

The basic idea of honeypot back-propagation is that, during its honeypot epochs, back-propagating *honeypot sessions* are created in *ASs* upstream from the server S towards attack sources. During a honeypot session at an AS, packets entering the AS destined for S trigger further propagation of honeypot sessions into the upstream peering ASs from which the packets are received. The back-propagation process stops if no more attack packets are received or when *non-transit* ASs are reached. A non-transit AS does not allow transit traffic from other ASs to pass through. It is connected to either a single ISP, or more than one ISP without allowing traffic from one ISP to pass through on its way to another ISP. Honeypot sessions at non-transit ASs install filtering rules to drop traffic exiting the AS destined for S. Except for the honeypot sessions at non-transit ASs, all other honeypot sessions are torn down at the end of honeypot epochs.

To implement honeypot back-propagation, two main mechanisms are required. The first is needed to identify the ingress points of honeypot traffic so that honeypot sessions get propagated to the corresponding upstream ASs. The second is needed for propagating and tearing down the honeypot sessions.

The first mechanism uses a *honeypot session manager (HSM)*, which is a host connected in the AS network that HSM maintains honeypot sessions and identifies the AS edge routers from which honeypot traffic enters the AS. Upon receiving a honeypot session, ingress honeypot traffic is diverted into the HSM. This can be achieved by sending iBGP route announcement declaring the HSM as the next-hop for ingress traffic destined to S [6]. Upon receiving this route announcement, edge routers forward honeypot traffic into the HSM. However, it is still needed to identify the honeypot traffic ingress points. To achieve this goal, two methods can be used: tunneling and packet marking. In the tunneling approach, Generic Routing Encapsulation tunnels [14] are setup between all edge routers and the HSM; the HSM identifies the ingress points by inspecting from which tunnels the diverted traffic is received. In packet marking, each edge router is assigned a unique identifier such that if there are n such routers, $\lg n$ bits are needed. Edge routers stamp their IDs into the diverted honeypot traffic. Because only the honeypot traffic is marked, the ID field in the IP header can be safely used. This packet marking scheme is similar to the destination-end provider marking scheme in [34].

Once an ingress point is determined by the HSM, a honeypot session is propagated to HSM of the corresponding peering AS. To achieve honeypot session propagation and tear-down, we define two messages:

honeypot request and *honeypot cancel* messages.

Figure 2 illustrates the propagation and tear-down of honeypot sessions along a DAG of ASs routed at the server's AS. As depicted in Figure 2(a), whenever the server S starts a honeypot epoch, it sends a honeypot request message to the HSM(s) of its AS(s). Upon reception of a honeypot request message, the HSM creates a honeypot session, during which the honeypot traffic ingress points are identified as described earlier. When an ingress point x is identified, a honeypot request message is sent to the HSM responsible for the upstream AS connected to x. Figure 2(b) illustrates a snapshot of the scheme *during* a honeypot epoch.

At the end of the honeypot epoch (Figure 2(c)), S sends a honeypot cancel message to the HSM(s) of its AS(s). When a HSM receives a honeypot cancel message, it sends a cancel message to all upstream HSMs to whom it has previously sent a honeypot request message. The HSM of a non-transit AS retains the honeypot session and installs a packet filtering rule at egress edge routers to drop traffic destined for S, otherwise the honeypot session is removed. Therefore, there will be one or more filtering rules applied at non-transit ASs hosting attackers, effectively stopping the attack. However, if legitimate clients are hosted on these non-transit ASs, their traffic will be dropped as well. Also, an ISP (or an enterprise) may be interested in identifying attack hosts within their networks, since the attack traffic sent by these hosts may backfire at the ISP by causing performance degradation inside the ISP network or attacking the ISP's own customers. To handle the identification of attack hosts, we propose intra-AS honeypot back-propagation.

5.2 Intra-AS Propagation

In intra-AS honeypot back-propagation, honeypot sessions at the HSM of an AS are used to further pin down attack hosts. Whereas inter-AS back-propagation requires minimal changes to routers, the intra-AS back-propagation imposes higher loads in the AS routers. The basic tenet of the intra-AS back-propagation is the use of localized traceback within the AS to locate access routers, that is, first-hop routers, of attack hosts. While many proposed traceback mechanisms can achieve this goal, we choose to illustrate the concept using a hop-by-hop traceback scheme.

Recall that honeypot traffic is diverted from edge routers, both ingress and egress, into the HSM during honeypot epochs. Ingress routers are upstream towards attack sources and egress routers are downstream towards the servers. In inter-AS back-propagation, the HSM sends honeypot request messages to the upstream ingress ASs. In intra-AS back-propagation, the HSM sends *local* honeypot request messages to the *egress* routers from which it received diverted traffic. Upon reception of a honeypot request message, the router creates a honeypot session, during which the honeypot traffic input ports are identified using router-level input debugging [38]. When an input port x is identified, a local honeypot request message is sent to the upstream router connected to x provided that local honeypot messages do not cross the AS boundaries. At the end of the honeypot epoch, the HSM sends a honeypot cancel message to the egress routers. When a router receives a honeypot cancel message, it sends a cancel message to all upstream routers to whom it has previously relayed a honeypot request message. Recall that if an access router is reached by the back-propagation process, it is because it is connected to an attack host. Therefore, access routers retain the honeypot sessions and install packet filtering rules to drop traffic destined for S. On the other hand, honeypot sessions in intermediate routers are removed.

5.3 Design Issues

Here, we discuss some design issues involved in honeypot back-propagation.

Message Security To prevent forging of honeypot request and cancel messages, which in itself can lead to DoS attacks, control traffic is encrypted and authenticated using shared keys between ASs, similar to securing BGP sessions. Moreover, in intra-AS back-propagation, messages are sent hop-by-hop and, thus, can be authenticated using the TTL field in the same way as in ACC/Pushback [24].

Incremental Deployment In order for honeypot back-propagation to reach attack hosts and stop the attack, all ASs along the attack path as well as all routers inside attack-hosting ASs should support the scheme. However, partial deployment is possible with some benefits, which increases as more ASs and routers implement the scheme. Partial deployment of inter-AS back-propagation can create gaps between ASs implementing the scheme, which would prevent further propagation of honeypot sessions. To bypass these deployment gaps, we can use a BGP option, in a similar way mobile honeypots are propagated in [22], such that when a HSM fails to propagate honeypot sessions to upstream ASs from which honeypot traffic is received, the HSM broadcasts the honeypot requests using BGP announcements to all upstream ASs. These announcements are propagated until they reach an AS with a HSM from which point normal propagation is resumed.

If an AS does not implement intra-AS back-propagation, it may be the case that the AS contains both attack and legitimate hosts, in which case applying filtering rules will block legitimate traffic as well. However, as mentioned earlier, it is to the ISP's own benefit to apply intra-AS back-propagation in order to detect compromized hosts, which may degrade the ISP performance or even attack its own customers.

False Positives Longitudinal studies of honeypots (e.g., [40]) show that honeypots receive a large amount of benign traffic, such as non-malicious probing, which if not adequately handled, may render the honeypot under attack all the time, causing high unnecessary

overhead of installing and tearing down honeypot sessions. Server-to-server control traffic is another source of false positives. To tolerate false positives, a server during its honeypot epoch does not send a honeypot request message unless either the rate of received traffic exceed a threshold or an attack is detected using other methods (e.g., [4]). Selecting an appropriate threshold is an object of future work.

Protection of the honeypot session manager (HSM) The HSM within an AS is a critical resource in the honeypot back-propagation defense. Therefore, it should be protected from targeted attacks. This protection can be achieved by replication and load balancing of customer traffic on replicated HSMs. The cost of the replication can be amortized over the ISP customers. Another way of blocking external attack traffic to the HSM is to assign it a private IP address (e.g., 10.0.0.1). This way the HSM is not accessible from outside the AS. However, for HSMs to be able to communicate, another mechanism is needed. BGP can be used for this purpose.

6 Simulation Results

The main advantage of honeypot back-propagation stems from the integration of honeypots, which provide accurate and prompt attack signatures. To analyze the effect of honeypot usage, we augment the Pushback scheme [24] with honeypot back-propagation and compare the performance of plain and augmented Pushback. We note that the Pushback framework is flexible enough to allow for the plugging of better attack detection, which our honeypot back-propagation scheme provides. Due to space limitation, we report some of our results; more detailed results can be found in [20].

6.1 ns-2 Model

We modified the Pushback ns-2 module to implement intra-AS honeypot back-propagation. First, because we do not depend on the aggregate-based congestion control, we disabled this feature. We defined a new message type for honeypot requests, and a new session type for honeypot sessions, and we modified the Pushback response to cancel messages in access routers to comply with our scheme. The roaming honeypots module [3] is also extended to send honeypot request and cancel messages, to support roaming with legitimate UDP traffic, and to start and end each honeypot epoch a little bit later and earlier, respectively, to accommodate in-transit legitimate traffic and clock synchronization.

6.2 Simulation Topology and Parameters

Our simulation topology is a tree with hop-count and router-degree distributions shown in Figure 3. This tree represents the network paths from legitimate clients and attack hosts to the servers. Although a tree collected using Internet measurements is more realistic, we use distributions roughly matching those of measured trees (e.g., [1, 2]).

We model five servers behind a bottleneck link of 1 Mb/s capacity. The bottleneck represents the root of the tree topology. Links incident on leaf nodes have a 1 Mb/s capacity and 1 ms propagation delay, whereas links incident on servers are 10 Mb/s in capacity and 1 ms in propagation delay. All other links have a capacity of 10 Mb/s and a propagation delay of 10ms. Although these capacities and propagation delays are not real, their relative values roughly represent relations between access and core links and were used to expedite the simulations.

Results with a network of 100 leaf nodes are reported in this paper, from which we select legitimate clients and attack hosts. Both legitimate clients and attackers send CBR traffic destined for the servers. In the results shown, we vary the number of legitimate nodes while keeping the total legitimate rate at about 90% of the bottleneck capacity (similar results were obtained with lower legitimate loads).

At the start of each periodic epoch, each legitimate client selects one of the three active servers uniformly at random and directs its traffic into it. In the case of Pushback and no-defense experiments, legitimate traffic is uniformly distributed over all five servers. Each attack host picks a server among the five servers uniformly at random and keeps on attacking it. Each simulation run lasts for 1000 seconds. Legitimate traffic starts at time 0, while attack traffic is from 50 to 950 seconds. We measure the throughput of legitimate traffic as a percentage of the total bottleneck link capacity. Figure 4 shows an example of how the legitimate throughput changes with time during one simulation run. At 50 seconds, legitimate throughput suffers from a drop for all schemes. However, for honeypot back-propagation, the legitimate throughput quickly recovers after attack hosts are captured.

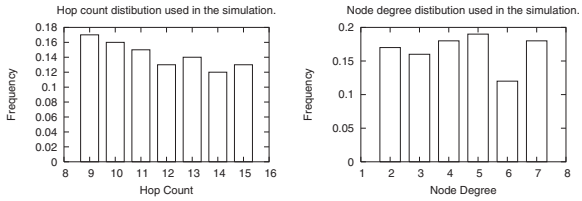

Figure 3. Hop count and node degree distributions for the simulated topology.

6.3 Effect of Attack Parameters

We study the effect of the location of attack nodes in terms of hop distance from victim servers, the number of attackers, and the attack rate per attack host for. Because we are interested in the system behavior under attack, in the next figures we average the client throughput during the attack time (i.e., from 50 to 950 seconds) of each simulation run. Table 1 summarizes the studied parameters and the values we experiment with.

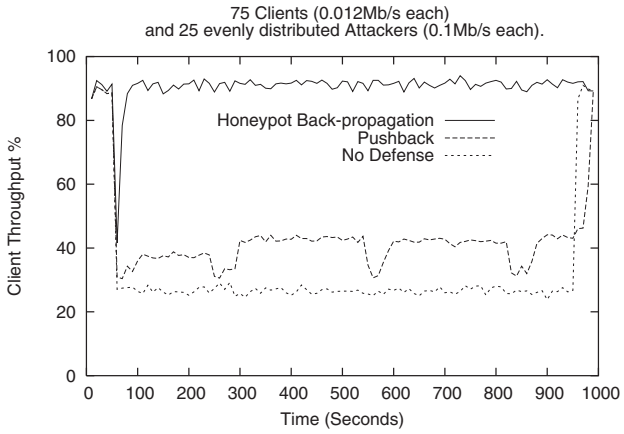

Figure 4. Time plot of one simulation run. Attack is between 50 and 950 seconds.

Parameter(s)	Simulated Values
(Total number of servers, number of active servers)	(5,3)
Bottleneck Link Capacity	1.0 Mb/s
Total Number of Leaf Nodes	100
Epoch length	10 seconds
Total Legitimate Rate	90% of bottleneck capacity
Number of Attackers	25, 50, 75
Attacker Locations	Far, Close, Evenly Distributed
Rate Per Attacker	(0.01, 0.05, 0.1, 0.5) Mb/s

Table 1. Simulation Parameters

6.3.1 Location of Attackers

We consider three scenarios of attacker locations: (a) close attackers, where the attackers are assigned to the closest leaves to the victim servers in the topology tree, (b) far attackers, in which attackers are assigned to the furthest leaves from the servers, and (c) evenly distributed attackers, in which the location of attack nodes are selected uniformly at random over all leaf nodes. In all scenarios, legitimate clients occupy the remaining leaf nodes.

As depicted in Figure 5, as the locations of the attackers get closer to the servers, ACC/Pushback punishes legitimate traffic more. The reason for this behavior is that ACC/Pushback adopts a hop-by-hop max-min fairness allocation of the rate limit among upstream routers without taking into consideration the number of end hosts behind each upstream router. So, for example, the fair share of an end-host connected to a router with another two upstream routers is one third of the rate limit irrespective of the fact that the two routers may have many upstream end-hosts connected through them. The result of this behavior is that as attackers get closer to the victim, their fair share increases, reducing the residual rate for legitimate clients. For close attackers, ACC/Pushback is even worse than no defense at all because it actually protects attack traffic.

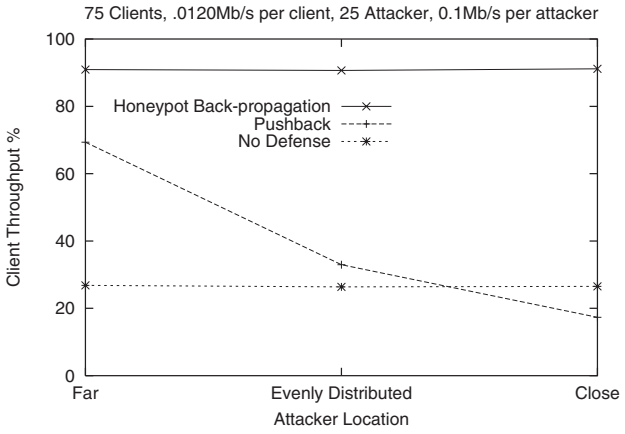

Figure 5. Effect of Attacker Locations.

6.3.2 Number of Attackers

Figure 6 shows that, for ACC/Pushback and for evenly distributed attackers, as the number of attackers increases, the number of attackers close to the victim increases, leading to an increase of the (protected) attack rate, and thus, the negative impact on legitimate throughput consequently increases. However, for far attackers, that is, close legitimate clients (not shown), client throughput is independent of the number of attackers, because in this case ACC/Pushback protects the legitimate traffic of the clients.

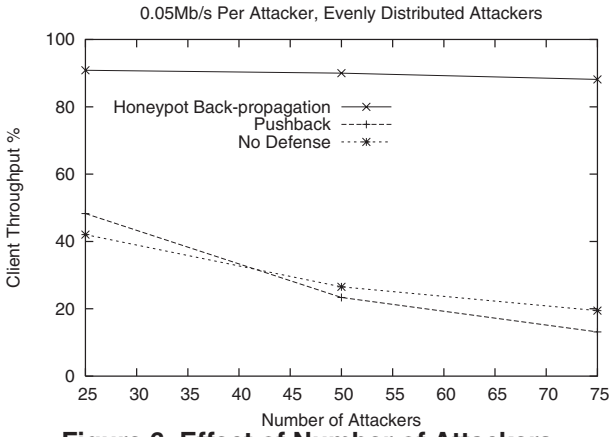

Figure 6. Effect of Number of Attackers.

As long as attacks last for enough time for honeypot back-propagation to reach their sources, which is the case in Figure 6, honeypot back-propagation is independent of the number of attackers. However, for higher number of attackers, some attack hosts may not be reached by honeypot back-propagation with our parameter values. For example, with 500 attackers, if each attacker sends at a rate of 5 Kb/s, the 1 Mb/s bottleneck link will be clogged. With a packet size of 1000 bytes basic scheme, epoch length has to be at least $\geq \frac{1000*8\text{bits}}{5\text{Kb/s}} \cdot h_i \approx 1.6 \cdot h_i$ seconds to reach an attacker h_i hops away from the server, without accounting of the delay of honeypot session propagation. For instance,

to reach an attacker 10 hops away, the epoch length has to be at least 16 seconds.

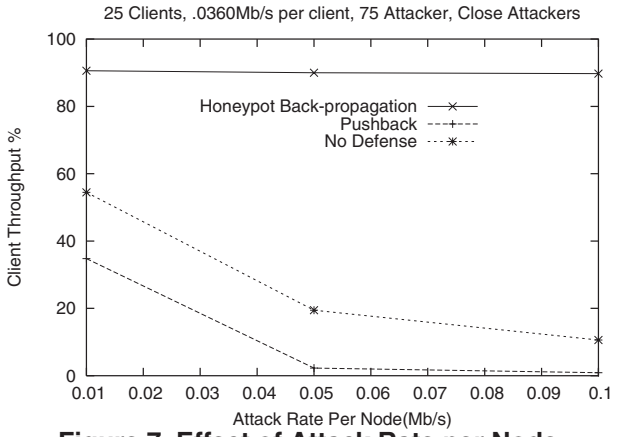

Figure 7. Effect of Attack Rate per Node.

6.3.3 Rate Per Attacker

As shown in Figure 7, for ACC/Pushback and close attackers, as the attack rate per attacker increases, the attack can grab more of the ACC/Pushback-enforced rate limit, resulting in reduction of legitimate throughput. For far attackers, however, legitimate throughput is independent of the individual rate per far attacker, because of ACC/Pushback protection of legitimate traffic of the close clients[20].

7 Conclusion

In this paper we presented *honeypot back-propagation*, a hierarchical traceback defense against DDoS attacks with spoofed source addresses. In coordination with its legitimate clients and other server replicas, a server enters honeypot epochs at times unpredictable to attackers. This allows the honeypots to receive pure attack streams, whereby honeypot back-propagation makes use of this stream to push honeypot sessions towards the attacker without disrupting the service provided to legitimate traffic.

Through ns-2 simulations, we show the feasibility of the honeypot back-propagation scheme and confirm its added benefit to the ACC/Pushback defense. Also, honeypot back-propagation supports incremental deployment, and it incurs a small overhead, since it is activated only during attacks.

References

[1] Cooperative Association for Internet Data Analysis. http://www.caida.org/.

[2] Internet Mapping Project. http://research.lumeta.com/ches/map/.

[3] NetSec Group. http://www.cs.pitt.edu/NETSEC.

[4] Snort. http://www.snort.com.

[5] The Network Simulator - ns-2. http://www.isi.edu/nsnam/ns/.

[6] S. Agarwal, T. Dawson, and C. Tryfonas. DDoS Mitigation via Regional Cleaning Centers. Technical Report RR04-ATL-013177, SPRINT ATL Research, Jan 2004.

[7] S. M. Bellovin, M. Leech, and T. Taylor. ICMP Traceback Messages. In *draft-ietf-itrace-01.txt, internet-draft, October 2001. Expired draft.*

[8] H. Burch and B. Cheswisk. Tracing Anonymous Packets to Their Approximate Source. In *14th Systems Administration Conference, LISA 2000.*

[9] C. C. C. A. CA-2003-04. MS-SQL Server Worm. http://www.cert.org/advisories/CA-2003-04.html, January 2003.

[10] CAIDA. Nameserver DoS Attack October 2002. http://www.caida.org/projects/dns-analysis/oct02dos.xml.

[11] CAIDA. SCO Offline from Denial-of-Service Attack. http://www.caida.org/analysis/security/sco-dos/.

[12] W. chang Feng. The case for TCP/IP puzzles. In *ACM SIGCOMM workshop on Future directions in network architecture*, 2003.

[13] D. Dean, M. Franklin, and A. Stubblefield. An algebraic approach to IP traceback. *ACM Trans. Inf. Syst. Secur.*, 5(2):119–137, 2002.

[14] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina. Generic Routing Encapsulation (GRE). In *RFC 2784*, March 2000.

[15] P. Ferguson and D. Senie. Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing. In *RFC 2827*, May 2001.

[16] J. Ioannidis and S. M. Bellovin. Implementing Pushback: Router-Based Defense Against DDoS Attacks. In *NDSS, 2002.*

[17] S. Kent and R. Atkinson. Security Architecture for the Internet Protocol. In *IETF, RFC 2401*, November 1998.

[18] A. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure Overlay Services. In *ACM SIGCOMM*, 2002.

[19] S. M. Khattab, C. Sangpachatanaruk, R. Melhem, D. Mossé, and T. Znati. Proactive Server Roaming for Mitigating Denial-of-Service Attacks. In *ITRE, 2003.*

[20] S. M. Khattab, C. Sangpachatanaruk, R. Melhem, D. Mossé, and T. Znati. Honeypot Back-propagation for Mitigating Spoofing Distributed Denial-of-Service Attacks. Technical Report TR-04-111, Department of Computer Science, University of Pittsburgh, Sept 2004.

[21] S. M. Khattab, C. Sangpachatanaruk, D. Mossé, R. Melhem, and T. Znati. Roaming Honeypots for Mitigating Service-level Denial-of-Service Attacks. In *ICDCS, 2004.*

[22] B. Krishnamurthy. Mohonk: mobile honeypots to trace unwanted traffic early. In *NetT '04: Proceedings of the ACM SIGCOMM workshop on Network troubleshooting*, pages 277–282, New York, NY, USA, 2004. ACM Press.

[23] J. Levine, R. LaBella, H. Owen, D. Contis, and B. Culver. The Use of Honeynets to Detect Exploited Systems Across Large Enterprise Networks. In *Proceedings of the 2002 IEEE Workshop on Information Assurance and Security.*

[24] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling high bandwidth aggregates in the network. In *ACM SIGCOMM Computer Communication Review*, volume 32, pages 62–73. ACM Press, 2002.

[25] J. Mirkovic and P. Reiher. A Taxonomy of DDoS Attack and DDoS Defense Mechanisms. *ACM SIGCOMM Computer Communication Review*, 34(2):39–53, April 2004.

[26] M. Oe, Y. Kadobayashi, and S. Yamaguchi. An implementation of a hierarchical IP traceback architecture. In *SAINT 2003 Workshop on IPv6 and applications*, Jan 2003.

[27] K. Park and H. Lee. On the effectiveness of probabilistic packet marking for IP traceback under denial of service attack. In *IEEE INFOCOM*, pages 338–347, 2001.

[28] C. Perkins. IP Mobility Support. In *RFC 2002*, October 1996.

[29] A. Perrig, D. Song, and A. Yaar. StackPi: A New Defense Mechanism against IP Spoofing and DDoS Attacks. Technical Report CMU-CS-02-208, School of Computer Science, Carnegie Mellon, Pittsburgh, PA 15213, December 2002.

[30] T. H. Project. *Know Your Enemy.* Addison-Wisley, Indianapolis, IN, 2002.

[31] G. Sager. Security fun with OCxmon and cflowd. Internet2 working group meeting, November 1998.

[32] C. Sangpachatanaruk, S. M. Khattab, T. Znati, R. Melhem, and D. Mossé. A Simulation Study of the Proactive Server Roaming for Mitigating Denial of Service Attacks. In *ANSS, 2003.*

[33] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for IP traceback. In *ACM SIGCOMM, 2000.*

[34] V. Siris and I. Stavrakis. Provider-Based Deterministic Packet Marking against Distributed DoS Attacks. In *SSN*, 2005.

[35] A. C. Snoeren. Hash-based IP traceback. In *ACM SIGCOMM*, pages 3–14, 2001.

[36] D. X. Song and A. Perrig. Advanced and Authenticated Marking Schemes for IP Traceback. In *IEEE INFOCOM*, 2001.

[37] S. Staniford, V. Paxson, and N. Weaver. How to 0wn the Internet in your spare time. In *Proceedings of the 11th USENIX Security Symposium*, 2002.

[38] R. Stone. CenterTrack: An IP Overlay Network for Tracking DoS Floods. In *Proceedings of the 9th USENIX Security Symposium*, 2000.

[39] X. Wang and M. Reiter. Mitigating Bandwidth-Exhaustion Attacks using Congestion Puzzles. In *ACM CCS 2004.*

[40] N. Weiler. Honeypots for distributed denial-of-service attacks. In *Proceedings of WET ICE 2002.*

[41] D. K. Y. Yau, J. C. S. Lui, and F. Liang. Defending Against Distributed Denial-of-service Attacks with Max-min Fair Server-centric Router Throttles. In *IWQoS*, 2002.