CluB: A Cluster Based Framework for Mitigating Distributed Denial of Service Attacks

Zhang Fu, Marina Papatriantafilou, Philippas Tsigas

Chalmers University of Technology, 42196 Gothenburg Sweden. Email: {zhafu,ptrianta,tsigas}@chalmers.se

ABSTRACT

Distributed Denial of Service (DDoS) attacks are threats not only for the direct targets but also for the core of the network. They are also hard to detect in advance, hence methods to deal with them need to be proactive. By building on earlier work and improving on distribution of control aspects, we propose a Cluster Based framework, which is called *CluB*, to mitigate DDoS attacks; the method balances the effectiveness-overhead trade-off by addressing the issue of granularity of control in the network. *CluB* can collaborate with different routing policies in the network, including contemporary datagram options. We estimate the effectiveness of the framework and also study a set of factors for tuning the granularity of control.

Categories and Subject Descriptors

C.2 [**Computer–Communications Networks**]: Security and Design; C.2.1 [**Network Architecture and Design**]: Network Communications

General Terms

Security, Design

Keywords

Cluster-Based, Distributed Denial of Service, Granularity of Control

1. INTRODUCTION

Distributed Denial of Service (DDoS) attacks can be so powerful that they can easily aggregate high volume malicious traffic and deplete the computing resources or bandwidth of the potential targets. Preventing or mitigating DDoS attacks is quite hard, since the key feature of today's networks is openness –any pair of Internet end points can communicate freely. This leads to the lack of effective network infrastructures for distinguishing and dropping malicious traffic in real time. When the attacker controls millions of *zombie* machines, it can easily launch a big volume of malicious traffic by having each of the zombie machines contribute only a small volume of packet flow. This is challenging not only for the target of the attack, but also for the network, as large volumes of illegitimate traffic share the same network resources as legitimate traffic and can furthermore cause congestion phenomena and performance degradation. Although there is large amount of literature on congestion control [5], the methods do not apply well to the DDoS situation, as they did not take malicious traffic into consideration. Considering malicious traffic, it is desirable to prevent it completely from consuming network resources. This is essentially the motivation of the recent research work on this theme (cf. section 2), focusing on controlling the malicious traffic as close to the attacker(s) as possible.

Ideally, in an imaginary world, if we have a global network monitor which can observe and control the flow between any pair of hosts, DDoS attacks could be mitigated by having this monitor identify every potential DDoS attack and stop the corresponding traffic as early as possible. However, it is well-understood that such a centralized countermeasure is practically impossible due to huge implied overhead —it would actually resemble a DoS attack itself! On the other hand, a completely distributed, low-overhead solution, where every entity has only local information, can have limited effect in stopping malicious traffic at some distance from the target. Viewing the problem from this perspective, we observe a trade-off in the achievable protection level of the network and the efficiency/overhead of the protecting framework, depending on the granularity of control.

Roughly speaking, to balance this trade-off, we would wish to have some distribution of control, with some sort of distributed authority present for coordination purposes. This observation led us to think of an analogy in real-life; namely the *exit and entry control* problem between countries in the world. A citizen of one country needs a passport and the corresponding visa to go to another country. The passport is a permission that the local country allows this person to go abroad and recognizes this person as a good citizen. The visa is the permission from the destination country to allow this person to enter.

Inspired by the above idea, we propose *CluB*: a Cluster-Based framework against DDoS attacks. In *CluB* the network consists of a set of clusters (in the Internet, these can be e.g. Autonomous Systems (AS), or neighborhoods of ASes; in cloud computing, these can be different clouds to provide secure routing services.) Packets need permissions to exit,

SAC'11 March 21-25, 2011, TaiChung, Taiwan.

enter, or pass-by different clusters. Several challenges need to be addressed, including what is the right level of distribution so that this can be feasible and scalable, how the permissions are issued, how the permission-control is carried out, how the permissions are implemented so that they are hard to be faked, and what is the level of protection that can be achieved. In section 3 we describe solutions in *CluB*, addressing the above questions. Briefly, the properties of *CluB* are that each of the sending permissions is *path-independent* within the clusters, the packets forwarded through clusters are *checked in a distributed way* and only some particular routers keep the states of valid traffic flows. We also suggest implementation options, involving the appropriate cryptographic tools, and periodic updates of components in the architecture as well. This limits the possibility of a powerful attacker to obtain means to launch direct attacks to the checking entities. In section 4, we analyze the proposed framework and show the guarantees of *CluB* in filtering malicious traffic. Furthermore, in section 5, we study how to choose the size of clusters, aiming at balancing the granularity-of-control trade-off. We further discuss the proposed framework in section 6, and give conclusion and future work in section 7.

2. RELATED WORK

Solutions proposed to defend DoS attacks can be mainly categorized into *reactive solutions* and *proactive solutions*. Generally speaking, reactive solutions usually rely on the targets to detect the attacks, and identify the malicious traffic by the path through which it is forwarded [21, 20, 16, 4, 15]. Proactive solutions aim at mitigating or preventing the DDoS attacks before they happen. Some proposals of proactive defense are based on *Secure Overlay Networks* [14, 22]. The basic idea is to use a secure overlay network as a proxy for the server that needs to be protected. Communication between clients and the server is via secure overlay nodes, the legitimate traffic is spread and the corresponding service becomes DDoS-resilient. But this mechanism is expensive for deployment. It is only worth using this mechanism to protect some very important services.

Capability-based mechanisms [2, 23, 24] against DDoS were proposed as an alternative. The essential component in this kind of solutions is a token (capability) which indicates the validity of the messages. Every router along the path will check this capability. Every capability is *path-dependent*, which implies that if the path is broken, the capability should be regenerated. Also, intermediate routers do not have the right to decide which traffic can pass through them. This implies that malicious hosts can allow each other to send traffic and flood a part of the network. Another important issue is that attackers can flood capability-request packets to the request channel to prevent the legitimate requests from being delivered. The attack against the initial capability requests was referred to as *Denial of Capability* (DoC) [3], i.e. capabilities need to be used together with mechanisms against DoC [18, 12].

Other related work includes the methods for traffic control across different organization boundaries using visas in [8] and the Platypus routing architecture [19]. Although the basic metaphor is similar (i.e. use of visa-like permissions), the mechanisms presented in these two papers are not designed to solve the DDoS problem. So naturally, the challenges mentioned in the introduction, which are necessary to be addressed in order to apply the intuitive metaphor into solving the DDoS problem, are not addressed in that work.

3. THE *CluB* FRAMEWORK

In this section, we will first give the system model, and in section 3.2 and 3.3 the key components and the basic protocol of *CluB* will be shown.

3.1 System Model

Network *nodes* (routers or end-hosts) are organized into disjoint clusters. All end entities (or hosts) are associated with clusters where they are located. Clusters that forward traffic between other clusters have *backbone routers* which do the job of forwarding transit traffic. We also assume that the deviation of the clock values of any pair of different nodes is bounded in a small range; this helps synchronize the periodic updates; the algorithm can also be extended to work for bounded clock drifts (implying unbounded deviation of clock values, as described in [10]). In this paper, bounded offsets are assumed for simplicity of the presentation. The attacker is modeled as an *adaptive adversary* which can eavesdrop and launch a bounded number of directed attacks (i.e. flood packets to a set of specific hosts/routers). There is an assumed bound of the time that it takes for the adversary to get the information on a possible target (e.g. some important router) and launch a directed (distributed) attack to it. We call this *exposure delay*.

We say a router is *compromised*, if the attacker can break into the router's system and get all its information. For the sake of the presentation, we first consider that the attacker cannot control the behavior of the compromised routers. In section 6, we will present and discuss mechanisms against malicious behavior of the compromised routers.

Depending on whether the source and the destination of packets are within the same cluster, traffic can be classified into *intra-cluster traffic* and *inter-cluster traffic*. Controlling intra-cluster traffic, due to the limited number of hosts and paths in one cluster, can be done e.g. via flow filtering based on fairness[5]. Each cluster can adopt its own method to prevent intra-cluster traffic flooding. In this paper, we concentrate on controlling inter-cluster traffic.

3.2 Key Components in The Framework

Each cluster C_i mainly controls three types of inter-cluster traffic **I)** *outbound traffic* i.e. from C_i to other clusters, **II)** *inbound traffic* i.e. from other clusters to C_i; **III)** *transit traffic* i.e. traffic that passes through C_i and goes to another cluster. In order to go out/in or pass through a cluster, packets need to have some permissions which are called *authentication tokens*. To control the validity of the authentication tokens, in each cluster there are designated *Egress Checking Routers* (*ECR*), *Ingress Checking Routers* (*ICR*). Transit traffic is controlled by *backbone routers*. Tokens and the checking routers are changed periodically. Section 3.2.2 and 3.2.3 show the details.

3.2.1 Coordination Authorities

Every cluster needs a coordination entity, which can be implemented in a centralized manner by a single node, protected by a secure overlay as in [14, 22]. We present the proposed solution with a single coordinator per cluster. This coordinator is publicly trusted and well protected, e.g. via a secure overlay as mentioned above, to avoid turning it to a

DoS attack target. The coordinator maintains security policies of its cluster and cooperates with coordinators in other clusters. Generally, each coordinator has the duty for the following tasks:
(I) It decides whether to allow a host in the cluster to send outbound traffic out of this cluster, or to allow a host of another cluster to send inbound and transit traffic to/via its cluster. The coordinator grants the *authentication code* of the cluster to the hosts of the approved requests.
(II) It generates new authentication codes for the cluster periodically; and gives the codes to the corresponding routers for checking the validity of traffic.
(III) As mentioned in the preceding paragraphs, not only the tokens, but also the routers for checking them are changed periodically. It is the coordinator that appoints the checking routers for each period of time. We describe this process in more detail in section 3.2.3.

Considering potential attacks to the permission requesting stage, these can be referred to the DoC problem (cf. section 2) and can be mitigated in two levels: In intra-cluster level, a secure overlay can be used to protect the coordinator, hence mitigating the potential attempts by the attacker to prevent the requests of legitimate hosts from being received by the coordinator. In inter-cluster level, coordinators may control the sending rate when they forward requests, according to policies agreed among clusters.

3.2.2 Authentication Tokens

When a packet is being forwarded in/via a cluster, the latter could be source, intermediate or destination cluster. Hence *CluB* has three kinds of authentication tokens for legitimate packets, which are computed using the corresponding authentication codes:
Outbound-authentication token is used for packets going out of the source cluster. This kind of token is generated with the *outbound-authentication code* of the source cluster.
Inbound-authentication token is used for packets being forwarded within the destination cluster. This kind of token is generated with the *inbound-authentication code* of the destination cluster.
Transient-authentication token is used for packets passing through intermediate clusters. Each token is generated with the *transient-authentication code* of the corresponding intermediate cluster.

Every cluster has it own authentication codes, and all authentication codes are changed periodically. A host that wants to send packets to another cluster, in order to generate valid authentication tokens for its packets, it needs the current outbound-authentication code of the local cluster, the current inbound-authentication code of the destination cluster, and the current transit authentication code of each intermediate cluster. Considering that a host may get the authentication codes and share them with other hosts, which is referred as *colluding*, the authentication codes are given in a hash format binding with specific IP addresses and checking routers. When the host gets authentication codes, it will generate authentication tokens for every outbound packet in the following way:

$$HASH\left(hash \oplus SeqNumber||hash\right), \qquad (1)$$

where $hash = HASH\left(SrcIP||DesIP||AuthCode||PID||RID\right)$ which is given by the coordinator, $SeqNumber$ is the packet's sequence number, "||" denotes concatenation, PID is the period-number, RID is the checking router's identity and $HASH$ can be an one-way hash function, such as MD5, SHA.

In each period, each outbound packet of this host has a unique sequence number. Generating authentication tokens with packet sequence numbers is to prevent the attacker from "replaying" legitimate packets.

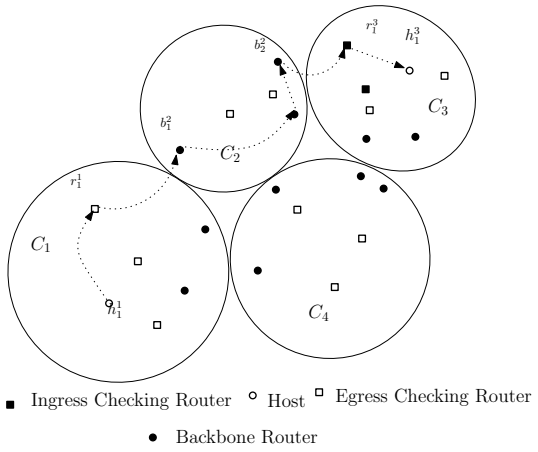


Figure 1: Example for describing *CluB*'s architecture and packet forwarding

3.2.3 Egress/Ingress Checking Routers

In *CluB* each cluster has one or more ordinary routers that act as the logical exit control of the cluster, called the *Egress Checking Routers* (ECRs). All the packets generated from this cluster to other clusters are supposed to be routed via some ECRs where the packets' outbound-authentication tokens are checked.

In addition to ECRs, *Ingress Checking Routers* (ICRs) also exist in each cluster. All the packets whose destinations are in this cluster are supposed to pass through some ICRs where their inbound-authentication tokens are checked.

In the beginning of every time period, the cluster coordinator computes the new ECRs and ICRs for this period using a pseudo-random function, and then sends a *notification message* to every router in the cluster. There could be more than one checking routers, considering fault-tolerance and load balancing issues. The notification message to a specific router only tells whether this router is a checking router or not in the current time period. If it is, the outbound/inbound authentication code for the current time period is included in the notification message. Since a notification message should only be readable by the corresponding receiver, the coordinator encrypts it with the receiver's public key and patches its digital signature in it. Due to the possible time offsets and packet delivery latency, each checking router has to serve a number of time units longer than a time period.

Besides, recall that we have the *backbone router(s)* in each cluster, responsible for checking packets in transit. These are assumed to be part of the underlying routing protocol (e.g. they can be the same as the backbone routers in the autonomous systems) or can be appointed in *CluB*.

3.3 The Basic Protocol

The protocol consists of three main parts, namely *per-*

Algorithm 1 Algorithm for a router (can be ECR router) processing an outbound packet.

```
initiate isECR;
AuthCode ← (Outbound authentication code);
PID ← the number of current period;

--When receive an outbound packet PK_out
routerECR ← PK_out.ECR;
/*check whether it should check this packet*/
if(isECR =true & myID == routerECR)
  if(checked before) drop PK_out;
  else
    SeqNum ← PK_out.SeqNumber;
    hash ← HASH(srcIP||desIP||AuthCode||PID||myID);
    if(HASH(hash ⊕ SeqNum||hash) == PK_out.AuthToken)
        PK_out.outBit = 1;
        route the packet out of the cluster;
    else drop PK_out;
else if(isECR != true & myID == routerECR)
  drop PK_out;
else if(isECR = true & myID != routerECR)
  if(!PK_out.outBit)
      route PK_out to routerECR;
  else if(PK_out is received directly from a host)
      drop PK_out;
  else route the packet out of the cluster;
```

mission requesting, *packet encapsulation*, *packet forwarding*, which are explained in detail below. Algorithm 1 shows the pseudocode for ECRs and ordinary routers processing outbound packets. The pseudocode for ICRs and inbound packet processing is symmetric.

3.3.1 Permission Requesting

Before a host sends packets to other clusters, it should send a request to the coordinator in the local cluster. Upon receiving the request, the coordinator checks whether this host misbehaved before according to local policy, e.g. blacklisting as using in the capability-based mechanisms. If everything is OK, the coordinator forwards this request to the coordinator of the destination cluster, while each coordinator of the intermediate clusters also gets it. They too, decide whether to grant the permissions to the host according to their local policies. If everything is OK, the destination coordinator provides the address of one of its ICRs and its inbound-authentication codes; similarly, all the intermediate coordinators provide their transit authentication codes. After the local coordinator gets the (encrypted) reply messages, it hashes each of the authentication codes as well as the outbound-authentication code (cf. section 3.2.2). Then the local coordinator will put these hash values and the addresses of ECR and ICR into one message and return it to the requesting host, encrypted using the host's public key.

3.3.2 Packet Encapsulation

After successfully getting the reply, the requesting host can generate the authentication tokens (cf. section 3.2.2) for its packets. In *CluB*, the regular packets have more header fields than the normal (e.g. IP) packets. These extra fields form a data structure, called *routing vector*. The first entry of the vector contains the address of an ECR of the source cluster and the cluster's outbound authentication token. The last entry of the vector contains one of the ICRs of the destination cluster and corresponding inbound-authentication token. The intermediate entries contain the corresponding transit-authentication tokens. Each entry also contains one bit, called *checking bit*, indicating whether the packet has been checked by the checking router.

We can use an example to illustrate the routing vector. As shown in Figure 1, host h_1^1 wants to send packets to h_1^3 in cluster C_3. The routing vector is like this:

$$\left\{\left\langle C_1, r_1^1, 0, tok_1\right\rangle, \left\langle C_2, backbone, 0, tok_2\right\rangle, \left\langle C_3, r_1^3, 0, tok_3\right\rangle\right\},$$

where tok_1, tok_2, tok_3 are the outbound, transit and inbound authentication tokens, and r_1^1 is an ECR in C_1, r_1^3 is an ICR in C_3. Initially, all the checking bits are set to 0. We describe how the packets are routed from h_1^1 to h_1^3 in the next section.

3.3.3 Packet Forwarding

An outbound packet is forwarded through the source cluster, intermediate clusters and destination cluster. Each authentication token is checked by the corresponding checking routers. If everything is valid, the packet will reach its destination. Consider the example illustrated by Fig. 1: A packet from h_1^1 to h_1^3 will be first routed to router r_1^1 for checking the outbound authentication token, then it will be routed out of cluster C_1, with next hop C_2. In C_2 the packet will be forwarded by the backbone router(s) where the transit authentication token is checked. After passing through C_2, the packet will enter into C_3 and be routed to router r_1^3 (ICR). After r_1^3 checks the inbound authentication token, the packet can be finally routed to its destination h_1^3. Below we detail some important steps of the forwarding process using our running example. For the sake of space limitation, we only show the forwarding process within the source cluster, the situations for intermediate and destination clusters are analogous.

When a non-checking router gets an outbound packet (it can tell that from the source and destination addresses), it checks whether this packet has been already checked. If the packet has not been checked by some ECR in the current cluster (i.e. its checking bit in the first entry of is 0), then the packet will be routed to this ECR. If the checking bit is 1, then the packet will be routed to exit the cluster When an ECR receives an outbound packet, it checks whether the router's identity in the first entry of the packet's routing vector is equal to its own identity. If not, then it will behave as an ordinary router. If yes and it did not check this packet before, it will check the validity of the packet, i.e. redo the computing for the outbound authentication token and compare it with the corresponding value contained in the packet; if they match, then the router will flip the checking bit and route the packet towards out of the cluster. The packets having invalid tokens will be dropped. Identifying the replayed packets needs the checking routers to keep states for the received packets, which would be a big overhead. To reduce this overhead, each ECR/ICR can also employ a *Bloom Filter* [17]. Details about how to use Bloom Filters are presented in an extended version of this paper [11].

Given that the attacker may not know which router is ECR, it may guess and fill an arbitrary router's IP in the routing vector. So if a non-ECR router receives an outbound packet and finds its IP address in the first entry of the routing vector, it will drop the packet. Routing a regular outbound packet from its host to its assigned ECR can be done using common routing protocols, such as RIP and OSPF.

Hop count h	1	2	3	4	5	6	≥ 7
AS distribution, p_h	0.04%	8.25%	38.55%	38.12%	12.7%	2.25%	0.29%

Figure 2: Probability that an AS resides exactly h hops away from a reference AS [6]

Dealing with malicious packets whose checking bits are set: The adversary may let the compromised hosts send packets whose checking bits are 1. Such packets might reach the destination without being checked. Since the checking bits should be changed only by the corresponding checking routers, if a router receives a packet whose checking bits are 1 directly from a host, this packet is definitely malicious and will be dropped. Given the assumption that the routers do not misbehave, this mechanism can prevent the malicious packets from avoiding the checking process. In section 6, we extend the method and discuss solutions against malicious behavior of compromised routers.

Token-Refreshing: Since the ECRs, ICRs and all authentication codes change periodically, a host has to renew them periodically, thus repeats the initial steps. Considering the cost in the requesting process, we let the coordinator of each cluster give the information about its ICRs, inbound and transit authentication codes to the coordinators of other clusters. So upon receiving a refresh request from a host who got the sending permission before, the coordinator does not have to forward the request to the other coordinator(s) that may be involved. If the host did not misbehave so far, new information will be granted to the host.

4. ANALYSIS AND EVALUATION

In this section, we analyze the proposed methods from two perspectives. First, the effectiveness of filtering distributed malicious traffic is studied, also in connection to the capability-based mechanism. Second, we analyze the influence of malicious packets flooding to legitimate traffic.

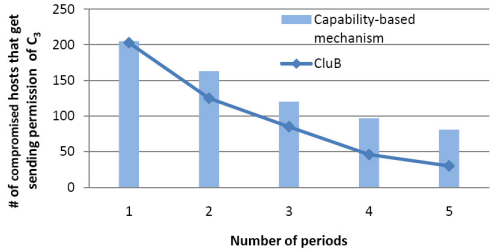
4.1 Filtering Effectiveness

We consider that the attacker, who does not know any of the authentication codes, randomly piggybacks some hash values in the routing vector. The expectation of the amount of malicious traffic reaching the target is estimated below.

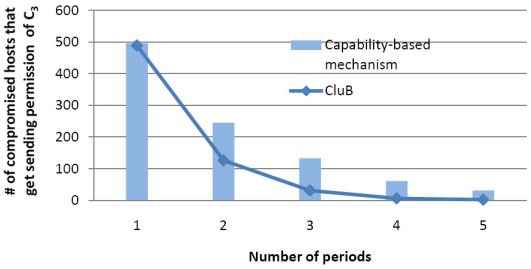
PROPOSITION 1. *Assume that the proportion of the malicious traffic that is generated at the clusters h hops (at cluster-level) away from the target is λ_h. Suppose the total amount of the malicious traffic is M, then the expectation of the amount of malicious traffic reaching the target is: $\sum_{h=1}^{\delta} \lambda_h \cdot M \cdot p_{in} \cdot p_{eg} \cdot p_{tr}^{h-1}$, where p_{in}, p_{eg}, p_{tr} are the probabilities that the attacker finds out one cluster's outbound, inbound and transit authentication code respectively and δ is the diameter of the network in cluster-level.*

To reflect on what the above implies, let us consider a case where the clusters in *CluB* are autonomous systems (AS) in the Internet. Suppose also that the amount of the malicious traffic generated from each AS to a specific AS is the same, and $p_{in} = p_{eg} = p_{tr} = p$. We use the data of neighbor ASes distribution of a specific AS mentioned in [6], which is shown in Fig. 2. Then the proportion of the malicious traffic that can reach the target is estimated as: $\sum_{h=1}^{6} p^{h+1} p_h$, where p_h is the probability that an AS resides exactly h hops away from the given AS. Here we choose the diameter as 6, since most of the ASes reside within 6 hops away from the reference AS. If each kind of authentication code has 32 bits, then the probability to guess out each of the authentication codes is $\frac{1}{2^{32}}$. Due to the time offsets and packet delivery latency, at any given time there could be more than one codes valid for each kind of authentication tokens. Let x be the number of outbound/inbound/transit authentication codes valid in each cluster at a given time. So in this example, $p = 1 - \left(1 - \frac{1}{2^{32}}\right)^x$. Considering that usually the time offsets and the packet delivery latency are quite small, e.g. we choose $x = 2$, then the percentage of the malicious traffic that can reach the target is smaller than 10^{-18}.

Comparison with the capability-based mechanism: To put the filtering properties of *CluB* in perspective, it is worthwhile to study them in connection to the capability-based method. Assuming *CluB* can use the same procedure of black-listing as that used in the capability-based mechanisms, we can study the potential effect of the difference between the two distributions of control. In the case study, we



(a) Each compromised host in cluster C_1 has probability 0.2 to attack each of the other clusters



(b) Each compromised host in cluster C_1 has probability 0.5 to attack each of the other clusters

Figure 3: Filtering effectiveness comparison between *CluB* and the capability-based mechanism. The number of compromised hosts in cluster C_1 (in fig.1) that can attack hosts in cluster C_3 in different periods is shown.

created 4 clusters with topology as shown in Fig.1. There are 1000 hosts in cluster C_1, all of which are considered as compromised. In every simulation period, each of the compromised hosts attacks each of the other three clusters with probability 0.2 (shown in Fig. 3(a)) or 0.5 (shown in Fig. 3(b)). Initially, none of the compromised hosts are prevented from sending packets to the other three clusters. We count the number of compromised hosts that can send traffic from cluster C_1 to cluster C_3 for 5 periods. From Fig.3, we can see that this number drops faster in *CluB* compared with the capability-based mechanism. This is because when some compromised hosts attack cluster C_2 and C_4 before attacking cluster C_3, they cannot send traffic to cluster C_3 later, since they are blacklisted in C_2 and C_4 and cannot get the

transit authentication codes. In the capability mechanism, since the responsibility for issuing capabilities is assigned to the end-server, those compromised hosts (who attacked C_2 and C_4 before) can still get capabilities from C_3 and send traffic to it as long as they are not blacklisted in C_3.

4.2 Packets Flooding to The Checking Routers

By keeping the length of the periods less than the exposure delay, we may prevent the adversary from attacking the checking routers directly. For the unlikely event that an attacker is able to launch a directed attack in a shorter time, we would like to know the harm that this may cause, i.e. to analyze the effect of the directed flooding attacks to the checking routers. We show this for the outbound packets flooding; the situation of inbound packets flooding is symmetric. Proofs of lemmas are omitted here due to space constraints and can be found in [11]. Let us first give the following definition:

DEFINITION 1. *The* pass-through probability P_{E-th} *is the probability that a legitimate outbound packet is processed by the ECR which is indicated in the routing vector of the packet.*

Next we will analyze the pass-through probability when the attacker launches outbound packets flooding to the ECRs. Consider that there are Φ ECRs in a cluster and each ECR can process B outbound packets per time unit. If an ECR has to process more than B outbound packets per time unit, then it randomly processes B packets from them (by sampling over small time intervals). When the attacker knows the information about the current ECRs in the cluster, it may control the compromised hosts to attack ψ ECRs, $\psi \leq \Phi$. We use s_i, $1 \leq i \leq \psi$, to denote the number of malicious outbound packets that are sent to the i^{th} ECR that is under attack. And we assume that $\forall i, s_i \geq 1$. For the sake of the analysis, we assume that in each time unit, each ECR receives the same number of legitimate outbound packets. We use Q to denote this number. Here to avoid non-trivial options, we assume $Q \leq B < Q + s_i$.

Given the i^{th} ECR that is under attack, the pass-through probability for a legitimate outbound packet assigned to this ECR is $\frac{B}{Q+s_i}$. When a legitimate outbound packets is sent to an ECR that is not under attack, the pass-through probability of this packet is 1. Since each ECR receives the same amount of legitimate outbound packets, we assume the probability that a legitimate outbound packet is send to a specific ECR is $\frac{1}{\Phi}$. We have the following lemma:

LEMMA 1. *For fixed* Φ *and* ψ*, the pass-through probability is:* $P_{E-th} = \frac{\Phi-\psi}{\Phi} + \frac{1}{\Phi}\sum_{i=1}^{\psi}\frac{B}{Q+s_i}$.

We use m to denote the number of compromised hosts in the cluster. Each compromised host can at most send b outbound packets per time unit. It can be shown that the attacking strategy to minimize P_{E-th} with ψ ECRs under attack is to let $s_i = \frac{mb}{\psi}$ for all $i \in \{1, \cdots \psi\}$. Furthermore, when the capacity of each ECR for processing the outbound packets is just enough for processing the legitimate outbound packets, that is $B = Q$, the attacker's option for minimizing P_{E-th} is to attack all the ECRs, i.e. $\psi = \Phi$, and $\forall i : s_i = \frac{mb}{\Phi}$. The corresponding P_{E-th} is $\frac{\Phi B}{mb+\Phi B}$. When each ECR has more capacity for processing outbound packets, that is $B > Q$ (i.e. there is over provision in each ECR's capacity by $B - Q$), the strategy to minimize P_{E-th} for the attacker is to attack $\frac{mb\left(\sqrt{\frac{B}{B-Q}}-1\right)}{Q}$ ECRs. Note that, since ψ is at most Φ, if $\frac{mb\left(\sqrt{\frac{B}{B-Q}}-1\right)}{Q} > \Phi$ then P_{E-th} is minimized when $\psi = \Phi$. We have the following lemmas as summary:

LEMMA 2. *If* $B = Q$*, then* $P_{E-th} \geq \frac{\Phi B}{mb+\Phi B}$.

LEMMA 3. *If* $B > Q$*: if* $\frac{mb\left(\sqrt{\frac{B}{B-Q}}-1\right)}{Q} \leq \Phi$*, then* $P_{E-th} \geq \frac{B-\sqrt{B(B-Q)}}{Q}$*; if* $\frac{mb\left(\sqrt{\frac{B}{B-Q}}-1\right)}{Q} > \Phi$*, then* $P_{E-th} \geq \frac{\Phi B}{\Phi Q+mb}$.

Plug in some real parameters, suppose each ECR can handle $1M$ outbound packets per second and the number of legitimate outbound packets sent to each ECR is also $1M$, that is $B = Q = 1M$. If there are 10 ECRs and the attacker can at most flood $10M$ packets per second, that is $mb = 10M$, then $P_{E-th} \geq 0.5$. Now suppose each ECR can handle more packets per second, say $B = 1.1M$, then $P_{E-th} \geq 0.55$; if now we have more ECRs, say 20, then $P_{E-th} \geq 0.768$.

5. SIZE OF CLUSTERS

Optimizing the granularity of clusters involves many factors. In this section, we will discuss the factors that play significant roles in choosing the granularity of control.

Security and Processing Load.

Note that if some of the ECRs/ICRs are compromised, then the outbound/inbound authentication code may be revealed. If we assume that each router has a certain probability to be compromised, then each cluster should not have many ECRs(ICRs), since this may increase the probability of the revelation of the authentication codes. Given a router, we use η to denote the probability it is compromised (assuming that each router has the same probability to be compromised). Then the probability that at least one of the ECRs is compromised is:

$$P_{comp} = 1 - (1 - \eta)^{\Phi}, \tag{2}$$

If we want to control P_{comp} under a threshold, say Γ, then $\Phi \leq \log_{1-\eta}(1 - \Gamma)$.

However, one cluster should have enough ECRs to handle its outbound traffic. If we use SHA-1 as the hashing function for the authentication tokens, according to Crypto++5.6.0 benchmarks [1], it can be processed at 153 MiB/sec on a 1.83 GHz Inter Core2 machine. Some off-the-shelf routers such as Cisco 7600 series have processors of 1.2 GHz, taking a conservative estimation that they can process the outbound packets with speed 50 MiB/Sec. If the volume of legitimate outbound traffic is 10 Gbps, then there should be $\frac{10Gbps}{50MiB/Sec} \approx 25$ ECRs. This is a constraint that we have to consider, since generally the amount of outbound traffic grows with the size of the cluster.

Traffic Stretch.

In the source cluster, an outbound packet is forwarded out of the source cluster via one of the ECRs. The shortest path length from the source to the neighbor clusters via one ECR is usually bigger than the shortest path length (which is the distance) between the source and the boundary of clusters, unless the ECR is on the shortest path. We use *traffic stretch* to measure this phenomenon. Here we define the stretch for the outbound traffic being forwarded in the source cluster (definition for inbound traffic is symmetric).

DEFINITION 2. Traffic stretch *of an outbound packet is the ratio between the shortest path length from the source of*

the packet to the next cluster via one ECR and the original shortest path length from the source to the next cluster.

Generally, the more ECRs a cluster has, the less the average traffic stretch will be. This is because more ECRs imply higher probability that the shortest path length via one of the them is close to the original shortest path length. The relationship between the traffic stretch and the number of ECRs depends on the intra-cluster topology. To illustrate this issue, we present a study on several intra-cluster topologies. In particular, we illustrate the traffic stretch as a function of the density of ECRs in the cluster. The first topology is a balanced tree (height of 4 and each internal node has 6 child nodes, 1555 nodes in total). The second topology is a (33×33) grid. The third topology is a power-law topology (with 1000 nodes). The fraction of ECRs among all the routers of the cluster is between 0.001 and 0.03. Figure 4 depicts the stretch of different intra-cluster topologies. Figure 5 shows the stretch of power-law topologies with different sizes. In both figures, the average stretch is decreased when the fraction of ECRs grows. It is interesting to see that in both figures, the decreasing speed of the average stretch is quite fast before the fraction of ECRs reaches 0.005; and after the fraction of ECRs reaching 0.01 the decreasing speed of the average stretch is quite slow. Given the trade-off between security and the traffic stretch, one could use such diagrams to decide on the ECR density (e.g. with the studied system settings, the optimal fraction of ECRs is between 0.005 to 0.01.)

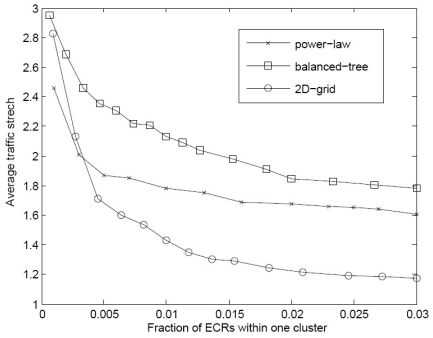


Figure 4: Average traffic stretch for routing packets via ECRs.

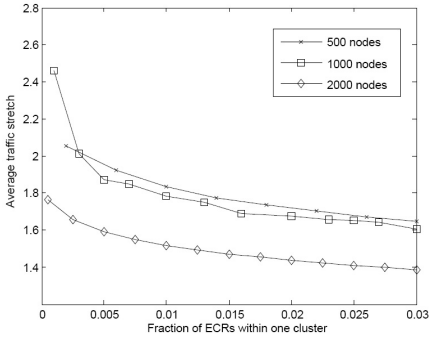


Figure 5: Average traffic stretch for routing packets via ECRs in the clusters of power-law topology.

Path Failures.

The forwarding path between one cluster to another may change in the cluster level due to the link failures or routing policy changes, then the sending hosts have to require transit authentication codes of the new clusters in the path. Next we will discuss the relation between the cluster sizes and the probability of path changing as a result of link failures.

We assume that the number of physical links between two neighbor clusters is proportional to the size of the clusters, e.g. $f(n) = \log\log n$, to denote the number of physical links between two neighbor clusters (we assume that every cluster has the same size n). We say the *connection* between two neighbor clusters fails when all the physical links between them fail. Given a physical link, we use θ to denote the probability that this link fails (we assume that each link has the same probability to fail). We say that a path fails when there exists at least one pair of two consecutive clusters in the path whose connection fails. We use P_{path-f} to denote the probability that a path fails, L is the path length in cluster level. Hence, we have:

$$P_{path-f} = 1 - Pr[\text{no connections fail in the path}]$$

$$= 1 - \Pi_{i=0}^{L-1}\left(1 - Pr[\text{connection between } c_i \text{and } c_{i+1} \text{fails}]\right)$$

If assume that every cluster has the same size, we have:

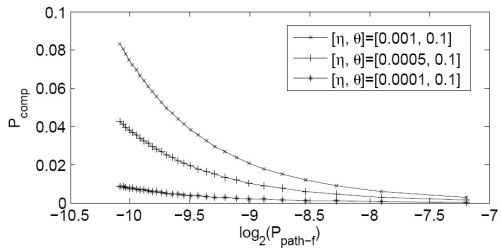
$$P_{path-f} = 1 - \left(1 - \theta^{f(n)}\right)^{L}. \quad (3)$$

Note that generally the length of a path decreases when the size of the clusters increases (for example if we merge two neighbor clusters together then the length of the path is $L-1$) and since a^x is a monotonically decreasing function when $0 < a < 1$ and $x > 1$, the value of $\left(1 - \theta^{f(n)}\right)^{L}$ will increase when n grows. Previous work (such as [9]) shows that the AS-level topology is complied to power-law. Chung et al. [7] showed that in common Internet graphs, where the topology is power-law with powers ranging from 2.1 to 2.45, the average distance between two ASes is $\log\log(N)$, where N is the number of autonomous systems. Since a cluster can be naturally mapped to one or a group of autonomous systems, if we assume the topology of cluster-level also follows power-law, then L can be estimated as $\log\log(N_C)$, where N_C is the number of clusters. Figure 6 illustrates the trade-off between P_{comp} and P_{path-f} using formula 2 and 3. We let the total number of routers equal to 100000, n ranges from 100 to 3000. From Figure 6 we observe that P_{comp} decreases when P_{path-f} increases, this is because increasing the size of a cluster will increase the amount of outbound traffic; then more ECRs/ICRs are needed, which leads to the decrease of P_{comp}. Fig. 6(a) illustrates the trade-off with fixed value of θ and varying η; while Fig. 6(b) illustrates the trade-off with fixed value of η and varying θ.

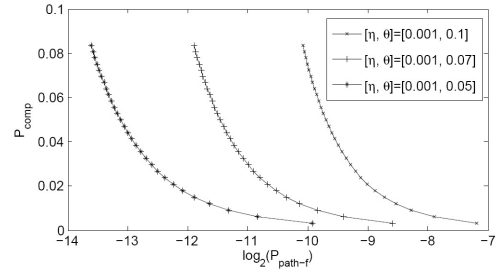
6. FURTHER DISCUSSION

Taking algorithm 1 into consideration, if the compromised routers can set the checking bits to 1 in the malicious packets, these packets can be forwarded to the destination without actually being checked. In this more aggressive threat model, to make sure that each packet is checked, we may use signatures. When a checking router finishes checking a valid packet, it fills its signature in the packet header. When other routers get this packet, they verify this signature, and the packet will be forwarded accordingly.

However, a compromised router can pretend to be a checking router and gives its signature to non-legitimate packets, then these packets can be forwarded by other routers. To



(a) η ranges from 0.001 to 0.0001, θ is fixed to 0.1



(b) η ranges from 0.1 to 0.05, θ is fixed to 0.001

Figure 6: Trade-off between the probability that at least one ECR is compromised and the probability of path changing in cluster level. X axis presents the logarithmic value of P_{path-f}.

solve this problem, the coordinator can broadcast the checking router list for the previous period in the notification messages. Now, the identifications of the compromised routers who pretended to be checking routers can be revealed and they can be black-listed. To reduce the computing overhead induced by digital signatures, we may use probabilistic verification: the router tries to verify each signature with probability σ_h which is inversely proportional to h (the number of hops between the current router to the checking router). It is also possible to use one time signature in conjunction with a hash tree to achieve fast verification and a small size of the public key as used in [13].

7. CONCLUSION AND FUTURE WORK

CluB is a proactive framework proposed here for mitigating DDoS attacks. In this paper we estimate the effectiveness of the framework as well as the impact of some potential attacks against the framework. *CluB* offers the possibility of adjusting the granularity of traffic control in the network, thus providing the possibility to tune the trade-off between overhead and effectiveness in stopping malicious traffic. This paper studies some important factors involved in this trade-off, including cluster size, density of ECRs/ICRs, confidentiality risk, processing load, traffic stretch, path connectivity and failures. A useful continuation can be a holistic study that investigates the impact of distribution level of the coordinating authority, period length, clock drifts, signature methods. It is also worth investigating the option of different clusters adopting different policies, which can be modeled as a game.

Acknowledgments

The research leading to these results has received funding from the Swedish Civil Contingencies Agency (MSB) and from European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No.257007.

8. REFERENCES

[1] Crypto++ 5.6.0 benchmarks http://www.cryptopp.com/benchmarks.html, 2009.

[2] T. Anderson, T. Roscoe, and D. Wetherall. Preventing internet denial-of-service with capabilities. *SIGCOMM Comput. Commun. Rev.*, 34(1):39–44, 2004.

[3] K. Argyraki and D. Cheriton. Network capability: The good, the bad and the ugly. In *In Proceedings of Workshop on Hot Topics in Networks (HotNets-IV)*, November 2005.

[4] K. Argyraki and D. R. Cheriton. Active internet traffic filtering: real-time response to denial-of-service attacks. In *Proceedings of the annual conference on USENIX Annual Technical Conference*, pages 10–10. USENIX Association, 2005.

[5] J.-Y. L. Boudec. Rate adaptation, congestion control and fairness: A tutorial, EPFL, December 2000.

[6] Y. Chen, K. Hwang, and W.-S. Ku. Collaborative detection of DDoS attacks over multiple network domains. *IEEE Trans. Parallel Distrib. Syst.*, 18(12):1649–1662, 2007.

[7] F. Chung and L. Lu. The average distances in random graphs with given expected degrees. *Internet Mathematics*, 1:15879–15882, 2002.

[8] D. Estrin, J. Mogul, and G. Tsudik. Visa protocols for controlling interorganizational datagram flow. *Selected Areas in Communications, IEEE Journal on*, 7(4):486–498, May 1989.

[9] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM '99*, pages 251–262. ACM, 1999.

[10] Z. Fu, M. Papatriantafilou, and P. Tsigas. Mitigating distributed denial of service attacks in multiparty applications in the presence of clock drifts. In *Proceedings of IEEE SRDS*, pages 63–72. IEEE Computer Society, 2008.

[11] Z. Fu, M. Papatriantafilou, and P. Tsigas. *CluB*: A cluster based method for mitigating distributed denial of service attacks, Technical Report 2009-09, Chalmers University of Technology, 2009. www.cse.chalmers.se/ ~ zhafu/CluB.pdf.

[12] Z. Fu, M. Papatriantafilou, P. Tsigas, and W. Wei. Mitigating denial of capability attacks using sink tree based quota allocation. In *Proceedings of SAC 2010*, pages 713–718. ACM, 2010.

[13] Y.-C. Hu, A. Perrig, and M. Sirbu. Spv: secure path vector routing for securing bgp. In *SIGCOMM '04*, pages 179–192. ACM, 2004.

[14] A. D. Keromytis, V. Misra, and D. Rubenstein. SOS: secure overlay services. *SIGCOMM Comput. Commun. Rev.*, 32(4):61–72, 2002.

[15] X. Liu, X. Yang, and Y. Lu. To filter or to authorize: network-layer DoS defense against multimillion-node botnets. In *SIGCOMM '08*, pages 195–206. ACM, 2008.

[16] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker. Controlling high bandwidth aggregates in the network. *SIGCOMM Comput. Commun. Rev.*, 32(3):62–73, 2002.

[17] A. B. I. M. Mitzenmacher. Network applications of bloom filters: A survey. In *Internet Mathematics*, pages 636–646, 2002.

[18] B. Parno, D. Wendlandt, E. Shi, A. Perrig, B. Maggs, and Y.-C. Hu. Portcullis: protecting connection setup from denial-of-capability attacks. In *SIGCOMM '07*, pages 289–300. ACM, 2007.

[19] B. Raghavan and A. C. Snoeren. A system for authenticated policy-compliant routing. *SIGCOMM Comput. Commun. Rev.*, 34(4):167–178, 2004.

[20] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical network support for ip traceback. *SIGCOMM Comput. Commun. Rev.*, 30(4):295–306, 2000.

[21] D. X. Song and A. Perrig. Advanced and authenticated marking schemes for ip traceback. In *IEEE INFOCOM 2001.*, volume 2, pages 878–886 vol.2, 2001.

[22] A. Stavrou and A. D. Keromytis. Countering dos attacks with stateless multipath overlays. In *Proceedings of ACM CCS*, pages 249–259, New York, NY, USA, 2005. ACM.

[23] A. Yaar, A. Perrig, and D. Song. SIFF: A stateless internet flow filter to mitigate DDoS flooding attacks. *IEEE Security and Privacy Symposium*, page 130, 2004.

[24] X. Yang, D. Wetherall, and T. Anderson. A DoS-limiting network architecture. In *SIGCOMM '05*, pages 241–252. ACM, 2005.