Design and Validation of PATRICIA for the Mitigation of Network Flooding Attacks

Lan Wang, Qishi Wu, Yaoqing Liu
Department of Computer Science
University of Memphis
Memphis, TN
Email: {lanwang, qishiwu, yliu6}@memphis.edu

***Abstract*—A recent trend in Internet denial-of-service attacks is to distribute the attack sources among a large number of compromised computers. To effectively control such attacks, the attack traffic must be stopped at an early stage, which means those edge networks that host the attack sources must be given proper incentives and mechanisms to stop undesirable traffic. We previously proposed an architecture called PATRICIA, where edge networks cooperate to prevent misbehaving sources from flooding traffic in both control and data channels. In this paper, we flesh out the details of the control protocols in PATRICIA and propose an important revision to the previous design to make it more robust against collusion attacks. Furthermore, we present the results from extensive simulation experiments to validate our design.**

***Keywords*-Denial-of-Service attack mitigation, control traffic flooding, collusion attack, packet filtering, capability**

I. Introduction

Large-scale flooding attacks are increasingly feasible due to the huge number of compromised machines around the world. Attackers employ “botnets” that consist of thousands of compromised machines in flooding attacks, email spams, stealing of sensitive information, and other malicious activities [1]. However, the distributed management paradigm of the Internet makes DoS defense a very challenging task.

The goal of our work is to protect legitimate flows from the flooding of attack traffic. We make the following explicit assumptions in this work: (a) an attacker may control a large number of hosts to send traffic at the same time; (b) malicious receivers may collude with malicious senders in order to cause congestion in intermediate nodes as well as to reduce the rate of legitimate flows that pass through those nodes; (c) a legitimate receiver under attack has a mechanism to differentiate good senders from bad senders, which may involve observing the source’s application-layer behavior and/or network-layer behavior. However, we do not assume that this detection mechanism works correctly in all cases. A number of previous papers, e.g. [2], have discussed how this identification may be done. Since our proposed approach can work with any bad-sender identification mechanism, we do not discuss them in detail here.

In [3], we presented a preliminary design of our approach PATRICIA, which allows edge networks to cooperate to prevent misbehaving sources from flooding traffic in both control and data channels. Our approach is based on the belief that *we cannot place trust entirely on hosts, as those hosts controlled by attackers are going to abuse the trust.*

In this paper, we flesh out the details of the control protocols in PATRICIA. Most notably, we have revised the design of the Transmission Channel Establishment Protocol to make it more robust against collusion attacks. Furthermore, we present the results from extensive simulation experiments to validate our design. The detailed protocols are presented in Section III - Section VI and the evaluation results are presented in Section VII.

II. Related Work

DoS attacks have been evolving and those mechanisms that rely solely on observations at the source networks are no longer sufficient in dealing with large DDoS attacks. For example, DoS attacks that use spoofed source addresses can be mitigated by ingress/egress filtering [4] at the source networks or their upstream ISPs. However, a measurement study by Mao et al. [5] shows that source spoofing is no longer commonly employed in DDoS attacks because the attackers control a large number of bots (i.e. there is no need to spoof) and source-spoofed packets can be easily filtered out. Another example is that DoS sources with asymmetric traffic can be detected and thwarted by their own networks using the D-WARD mechanism [6]. However, bots in a large-scale DDoS attack can each send a small volume of non-spoofed traffic to make their traffic symmetric, while the aggregate traffic can still overwhelm the receiver and/or network links. *The above examples illustrate that we need information from the receivers who know more about appropriate application-level behavior and we need collaboration among networks to collect such information.*

Our work is partly inspired by two existing approaches that use information from the receivers: *receiver-based packet filtering* and *capability*. Receiver-based filtering mechanisms, e.g. AITF [7] and StopIt [8], allow receivers to notify the network of their intent to stop certain traffic flows by installing flow-level packet filters in routers. Such mechanisms may not scale well when dealing with large-scale distributed DoS attacks, as routers may need to maintain a large number of filters and complex filtering could significantly slow down packet processing. Anderson *et. al.* proposed the concept of *capability* [9], which was refined in SIFF [10] and TVA [2]. A capability is a form of authorization granted by a receiver to a sender – packets that do not carry correct capability values will be dropped by intermediate routers. Liu et al. proposed a filter-based approach called StopIt and compared it with other approaches [8]. They found that StopIt outperforms capability-based approaches in all but one type of attacks. They also conjectured that combining the two approaches may give a most effective solution.

There are two major challenges for both filtering and capability-based mechanisms: DoS attacks against legitimate control traffic [11] and collusion among malicious hosts [2]. **Our work employs both filtering and capability approaches while addressing their inherent problems**. We maintain filters (i.e. blocking lists) in edge networks to stop the flooding of control traffic, which is more scalable and effective than maintaining the filters in the ISPs. We use the capability approach in the ISPs for its scalability. Moreover, sources and destinations need to obtain endorsements from their own networks, hence creating more hurdles for control traffic flooding and allowing edge networks to detect and prevent collusion at the early stage of data transmission. Another major difference between our design and previous work is the dynamic switching between passive and active traffic regulation, which reduces the overall processing and bandwidth overhead.

III. PATRICIA Design

A. *Summary of Main Features*

Below we summarize the main features of PATRICIA and explain the rationale behind our decisions.

1) Reactive Traffic Regulation: The current Internet allows any host to send packets to any other host. This openness has greatly contributed to the success of the Internet, but also leaves hosts/routers vulnerable to attacks. It would be desirable to design an effective DoS mitigation scheme that does not unduly restrict the openness of the Internet. With the above preference in mind, we designed PATRICIA as a reactive scheme that dynamically switches a host to an *active traffic regulation* mode only when necessary. This reactive approach is viable because the vast majority of Internet hosts are home PCs or regular servers. These machines may receive some constant port scanning traffic, but the traffic volume is usually not enough to overwhelm them except during a major DoS attack.

2) Edge Network Endorsement: To control misbehaving hosts during an attack, we introduce an *endorsement* procedure in edge networks. For a source to communicate with a destination in the *active regulation mode*, both the source and the destination must obtain an endorsement from their own network. and (b) an authorization from the destination, before it can send out any data traffic. Likewise, the destination must obtain an endorsement from its own network in order to authorize the source's data traffic. This endorsement procedure allows edge networks to examine the requests and authorizations issued by their hosts.

3) Edge Network Cooperation: In PATRICIA, each edge network maintains a blocking list (similar to a filter) of misbehaving *local* senders reported by other edge networks. These senders cannot get the endorsement to establish a communication channel with a particular destination (or a set of destinations) for a specified period of time. More specifically, their control messages will not be endorsed by their own edge network and will be dropped by other networks. This feature makes PATRICIA particularly effective against the flooding of control messages (e.g. denial-of-capability attack). Note that an edge network may decide to completely disconnect a local host from its network, if too many reports about the host have been received. However, we would like to emphasize that this is entirely a local decision.

B. *Traffic Regulator*

Each edge network has one or more *traffic regulators*, which use blocking lists and local policies to make endorsement decisions (see Section V for details). For example, the local policy may assign certain important nodes a higher priority in getting endorsements. The traffic regulator can also impose a rate limit on how often each node's control messages can be endorsed, in order to limit the flooding of control messages to other networks. Again, the local policy may specify a higher rate limit for those servers that are expected to send and/or receive a lot of traffic. In addition to providing endorsement service, the traffic regulator activates the protection of local hosts when they are under attack. This process is described in Section IV.

Traffic regulators may become potential targets for attacks. As a precaution, we limit the communication with each traffic regulator to local hosts and routers only. Routers can simply drop any packets originated from non-local source IP addresses but destined to a local traffic regulator. Of course, a local host or router may still attack a traffic regulator, but this is much easier to localize and mitigate than an attack from a remote source. One implication of our restriction is that traffic regulators in different networks cannot directly communicate with each other. However, our design still allows a traffic regulator to indirectly report a misbehaving host to that host's traffic regulator (Section V).

C. *Feasibility*

We have discussed how to incrementally deploy PATRICIA and the deployment incentives for ISPs in our previous paper [3], here we emphasize the feasibility of PATRICIA in terms of its overhead. PATRICIA incurs processing overhead in the active traffic regulation mode. However, only a small percentage of destinations may require active traffic regulation at any given time, since most destinations are in passive mode. Moreover, endorsement is required only for control messages, *not* for data packets. Because the number of control messages is limited by our scheme, the overhead of generating and verifying endorsements should be bounded.

In our current design, an endorsement is in the form of a *public-key signature*. One common concern is the processing overhead of public-key signatures, but advances in both software and hardware continue to enhance processing capability of signatures. For example, a recent implementation of the Rabin-Williams algorithm can achieve 7 μs for a signature verification (1024-bit keys) on a PC with an Intel 2.33GHz Core2 processor [12]. If 5% of a link's capacity is reserved for control messages, and a PATRICIA control message has at least 200 bytes (a Rabin-Williams signature requires 80 bytes, and the IP header requires 20 bytes), the software implementation running on the PC can handle the signature verification for a link faster than 4.6 Gbps. Hardware-based implementations using FPGA or ASIC can handle much faster links. Since the signature verification is not part of data packet processing, it could be performed by dedicated hardware on border routers. In addition, two neighboring networks may establish a trust agreement between each other so that redundant verifications are minimized when a packet crosses the boundary between two neighboring networks. On the other hand, signature generation is slower than verification and a large edge network may need to process requests from many flows. Possible solutions are employing multiple traffic regulators to share the load and using hardware-based signature generators.

Another common concern for using public-key signatures is key distribution. In PATRICIA, however, each edge network needs only one public key and the number of ASes in the Internet is on the order of 10^4 (this number grows slowly). Therefore an edge network may simply download the entire public-key database from the authority that certifies the keys[1].

Hash code operations for generating and verifying authenticators are orders of magnitude faster than public-key signature operations, so routers should be able to handle them fairly easily. Yang et al. [2] conducted a detailed feasibility evaluation of hash code computation.

IV. Traffic Regulation Activation Protocol

The Traffic Regulation Activation Protocol (TRAP) handles the dynamic switching between passive and active regulation. In passive traffic regulation, a host uses a regular IP address, allowing other hosts to send data packets directly to it. Once the host is deemed under attack, it is assigned an address from the *covered IP address* pool and enters the "active traffic regulation" mode. Such dynamic address assignment can be handled by DHCP. In order to send packets to a host with a covered IP address, a sender needs to establish an authorized transmission channel to that host (see Section V).

A. Covered IP Address

A covered IP address could be identified by some subset of the bits in the address, so that a router can instantly recognize whether a packet is sent to a covered IP address. For example, if the value of the last eight bits of an address falls in a globally known range, the address is considered a covered IP address. It is better to use the lower bits of an address for this identification so that each subnet gets one or more covered addresses to assign.

[1]Another possibility is to distribute the keys through the inter-domain protocol BGP in routing updates.

When a host is no longer under attack, its assigned covered IP address should be released back into the pool. This decision could be as simple as allocating addresses for a set amount of time, and then requiring that hosts make another request for an address if the need still exists. A returned covered address should be retained for a set amount of time before being given to a new victim to prevent packets sent to the previous victim being received by the new victim.

In practice, *hosts offering critical services (e.g. DNS servers) may be issued permanent covered IP addresses* as they are primary targets of attacks, so that they do not have to repeatedly switch between passive and active regulation.

B. TRAP Procedure

First, when a node is under a DoS attack, it sends a message to its local traffic regulator to request a covered IP address. When the traffic regulator receives the request, it may apply local policies in the address assignment. For example, the policy may favor hosts that offer known services – these hosts may have a priority in obtaining covered IP addresses and their allocated addresses may have a longer expiration time. In addition, the traffic regulator can make sure that the requester does not have any known security breaches or potential vulnerabilities, for example, by scanning the host. Without these provisions, two colluding hosts can authorize a lot of data traffic between them, exhausting the resources along the traffic path and preventing legitimate traffic from reaching their destinations. In our scheme, *the ability to authorize data traffic is a privilege, not a right automatically given to every host*. We have to ensure that a host is indeed under attack and that it has not been compromised before allowing the host to authorize traffic.

After granting a covered IP address to a host, the traffic regulator immediately notifies the local network's authoritative DNS servers to update the DNS record for this host – all new senders should use this address to communicate with the host. The regulator also instructs the local border routers to drop packets to the old address. *In addition, the host should notify all the hosts that are actively communicating with it* so that they will send requests for authorization to the new address. This notification includes the old address, the new address, a timeout and an endorsement from the Traffic Regulator so that the border routers along the path can verify the legitimacy of the message.

There may still be some new senders trying to reach the old address, as the old DNS record for this host may be cached in some non-authoritative DNS servers. *However, if the above measures are taken, few **legitimate** senders will try to reach the old address after the address change*, based on the following reasoning: (i) if this destination host is a critical server, it should use a permanent covered IP address and its users would not encounter this problem at all; (ii) if the host does not provide any service, there should not be any other legitimate hosts initiating connections to it; and (iii) if the host provides some non-critical service, the owner could use a short expiration time for the DNS record.

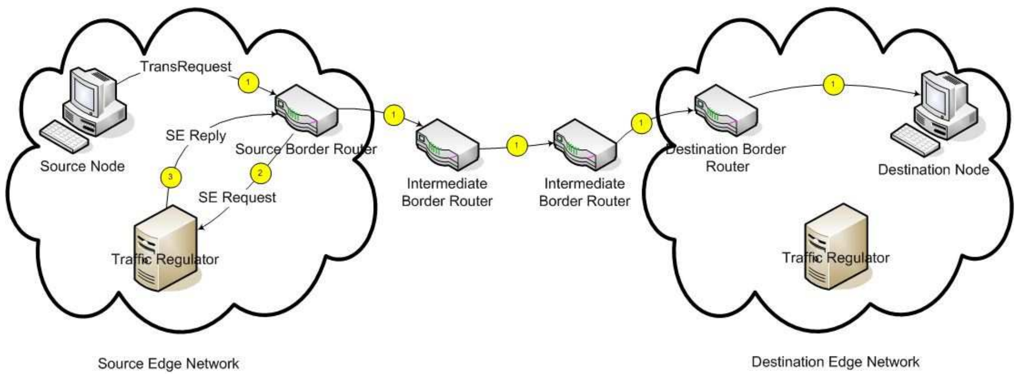


Figure 1. TCEP Message flow on forward path from source network to destination network: (1) transmission request (TransRequest); (2) source endorsement request (SERequest); and (3) source endorsement reply (SEReply).

This approach has already been adopted by many users, as evidenced by the popularity of dyndns.org, which provides dynamic DNS records with an expiration time of 60 seconds. Note that this measure is only intended for new senders – those senders with an ongoing connection with the server have already been notified of the address change. Even if the user does not adopt this measure, the old DNS records will expire eventually.

V. Transmission Channel Establishment Protocol

Before a traffic source can communicate with a destination that uses a covered IP address (i.e. it is protected by active traffic regulation), it needs to use the *Transmission Channel Establishment Protocol (TCEP)* to obtain an authorization from the destination. In this section, we explain how the TCEP messages are used to establish an authorized transmission channel, and how the blocking list is established through the cooperation among edge networks. *Note that the procedure described here is different from our preliminary design in* [3]. The design changes make PATRICIA more robust against collusion attacks (see Section V-E).

A. TCEP Control Messages on Forward Path

As shown in Figure 1, the source first sends a **Transmission Request (TransRequest)** message towards the destination. When the message reaches a border router in the source network, the border router verifies the message's source address using a test similar to the TCP SYN cookie exchange. If the address appears to be authentic, the border router sends to its local traffic regulator a **Source Endorsement Request (SERequest)** message containing the source-destination pair of the proposed transmission channel.

When the traffic regulator receives the SERequest message, it checks its "blocking list" – a list of misbehaving local hosts reported by other edge networks and, for each host, the destinations with which it cannot communicate (see Section V-D). Messages with a source-destination pair on this list are rejected. In addition, the traffic regulator may apply the edge network's local policies, e.g., a priority and a maximum endorsement frequency for each host. These steps make sure that the source host has neither been identified as a DoS source by other networks nor been rejected by its local network's policies.

Once the source's local traffic regulator makes a decision, it sends back a **Source Endorsement Reply (SEReply)** message to the border router. This message contains the decision on whether to endorse the proposed transmission channel and a public-key signature that covers the source address, destination address, the traffic regulator's address, the decision, and a timestamp for preventing replay attacks.

If the traffic regulator's decision is negative, the border router immediately notifies the source that its transmission request has been rejected through a **Transmission Reply (TransReply)** message (see Section V-C for description of this message). Otherwise, the border router includes the decision and the signature in a **TransRequest** message to the destination. The border router and the traffic regulator can set up a persistent secure connection for their communication. With the secure connection, the border router does not have to verify every signature received from the traffic regulator, but subsequent border routers still need to do so.

Along the path from the source network to the destination network, each border router performs two functions after receiving the TransRequest message: *(a) verifying the signature:* the router first retrieves the public key associated with the source address and the traffic regulator from its key database. It then uses the key to verify the signature; *(b) generating authenticator for subsequent data traffic:* the router computes a keyed hash over the source and destination addresses using a local secret key. The hash value is appended to the TransRequest message, and the series of hash values generated by the border routers becomes the *authenticator* for subsequent data messages.

Because the secret key in each border router changes periodically, the authenticator needs to be renewed periodically (but these renewals are carried in data messages, not control messages). If the source misbehaves, the receiver will not return the new authenticator to the source, thereby preventing the source from sending more data traffic. Note that renewing the authorization does not involve any control messages; the new authenticator is carried in data packets. This kind of hash-based authenticator was proposed by Yaar et al. [10] (we omit the details due to space constraints).

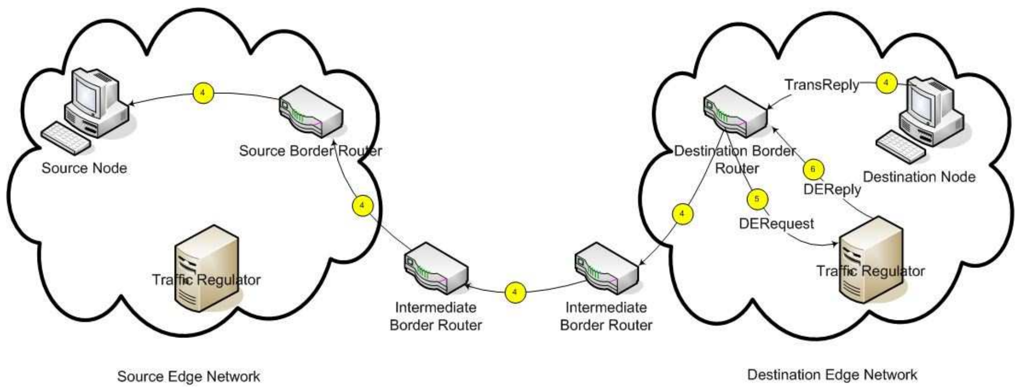


Figure 2. TCEP message flow on reverse path from destination network to source network: (1) transmission reply (TransReply); (2) destination endorsement request (DERequest); and (3) destination endorsement reply (DEReply).

When the TransRequest message reaches the destination's border router, the router removes the authenticator from the message and caches the authenticator. This prevents the destination from colluding with the sender, as we will explain in Section V-E. The border router then forwards the message to the destination. It needs to record its own IP address in this message so that the reply can come back to it (see Section V-C for more information).

B. Receiver Decision

Now we describe what happens after the destination receives the TransRequest message. First, the destination decides whether to accept the request. If the source has never communicated with the destination over an authorized transmission channel before, the destination may simply accept the request, as it cannot use any past history to make its decision. Otherwise, the destination can make an informed decision based on the source's past behavior. Note that PATRICIA can work with any mechanism for identifying misbehaving senders, whether it is based on the source's application layer behavior (e.g. through CAPTCHA [13]), network layer behavior, or an existing blacklist.

C. TCEP Control Messages on Reverse Path

After the destination makes a decision, it sends back a **Transmission Reply (TransReply)** message (see Figure 2). Note that if the destination directly sends a control message back to the source, it may reach a different border router in its local network than the one on the forward path. This is undesirable because the authenticator was cached in the latter. Our solution is to let the destination explicitly send its message to the border router on the forward path (this address is recorded in the TransRequest message), which then forwards the message to the source.

When the message reaches the local border router, the router sends a **Destination Endorsement Request (DERequest)** message to the local traffic regulator with the destination's decision (accept or reject). If the destination's decision is to reject the source, the traffic regulator should not overturn this decision. However, it can overturn an "accept" decision from the destination, for example, if its monitoring statistics show that the source has sent a lot of undesirable traffic before. The traffic regulator's decision also reflects local policies, e.g., how frequently a host can authorize new flows. *Such policies can prevent the destination from accepting too many transmission requests, thus breaking down collusion.* Once the traffic regulator makes a decision, it returns a **Destination Endorsement Reply (DEReply)** message to the border router. This message includes most of the fields in the DERequest, the traffic regulator's decision, and a signature.

The border router places the received information in a **Transmission Reply (TransReply)** message. If the decision is positive, the previously cached authenticator is also included in the message. The border router then sends the message back to the source host. All the participating border routers on the reverse path verify the signature in the message. When the TransReply message reaches the source network's border router, the router verifies the signature and it reports to its local traffic regulator if the decision included in the message is negative (i.e. the transmission request has been rejected). This information is used to set up the blocking list as we will explain in Section V-D. A negative decision should also include a blocking duration (specified by either the destination or the destination's traffic regulator) indicating how long the source needs to wait before attempting to connect to the receiver again. This information is also stored in the blocking list.

The border router then removes the signature from the TransReply message and forwards the message to the source host. When the source receives the TransReply message, it checks the decision included in the message. If the decision is positive, the source host extracts the authenticator from the message and fills it in the IP header's optional portion in its subsequent data packets for routers to verify. When the actual data transfer occurs, each border router verifies the data packets by recomputing the corresponding hash value in the authenticator carried in the IP header.

If the source receives a negative decision, it will wait for a period of time (i.e. the blocking duration) as specified in the TransReply message before sending another TransRequest message. Malicious sources may ignore this timeout value and continue sending more TransRequest messages, but these messages will be endorsed by its local traffic regulator.

D. Blocking List

An important feature of PATRICIA is the **blocking list** maintained at each edge network. This list contains local hosts whose traffic has been deemed undesirable by other hosts or networks. It is established through the cooperation among edge networks, as we briefly described in the previous section. Below we provide more details.

Suppose Host A has been identified by host B or B's network as a potential DoS source (e.g., through a reverse Turing test or traffic monitoring). If A tries to establish a transmission channel with B, B or B's network will reject this request through a Transmission Reply (TransReply) message. When this message arrives at A's network, the border router that receives the message can report the rejected address pair (A, B) to the local traffic regulator. The traffic regulator will add this address pair to its *blocking list* and stop endorsing A's subsequent Transmission Request messages to B for a period of time. This time period could be set by B or B's traffic regulator in the TransReply.

Note that attackers cannot take advantage of this mechanism to block legitimate traffic. Suppose a good source host C wants to communicate with B. If a malicious host D wants to block the traffic from C to B using the blocking list, it has to forge a TransReply message from B to C with a negative decision. However, because D cannot generate a valid signature, its forged TransReply message will be dropped by C's border router.

E. Why let the border routers request for endorsements?

In the previous version of PATRICIA (see our NPSec workshop paper [3]), hosts directly communicate with their local traffic regulators to obtain endorsements. We have changed our design to let border routers communicate with traffic regulators on behalf of sources and destinations, mainly to prevent colluding between the hosts. Below we explain why the previous design may not work well in collusion situations.

Suppose A is a DoS source and B is colluding with it. When A sends a TransRequest mesage to B, B obviously will accept A's request, but its traffic regulator may decide to overturn B's decision (e.g. based on the total amount of traffic A has sent to the other hosts on the network). However, B will not send a TransReply with its traffic regulator's negative decision back to A, since this will put A on the blocking list at A's traffic regulator. In other words, B's withholding of the TransReply message defeats the blocking list mechanism.

In our current design, B only knows its own decision when sending a TransReply message to the source, so B does not have any incentives to withhold the TransReply message. The border router at B's network will communicate with the traffic regulator to obtain the negative decision, which will be delivered to A's network.

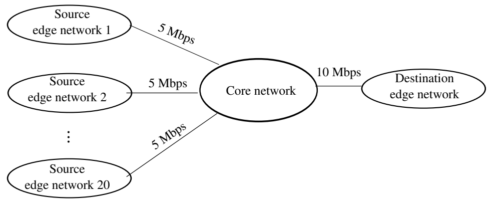


Figure 3. Network topology for simulation.

Another major change in our design is to let B's border router remove the authenticator in A's TransRequest message before delivering it to the destination. Otherwise, B can circumvent our mechanisms by sending the authenticator back to A simply using a data packet (not via a TransReply message which requires the traffic regulator's endorsement), thus allowing A to send "authorized" traffic to B.

On the other hand, our new design does require that B's border router cache the authenticator. Fortunately, the authenticator needs to be cached for only one round-trip time between the border router and a local router (B in this example). Therefore, the cache size is bounded by the maximum number of TransRequest messages that can be received over one local RTT, which should be on the order of 10 ms or less. Note that we limit the control traffic to a small percentage of the total link capacity (see the next section). *For a 10Gbps link, a router needs to cache at most 625 authenticators for that link*, assuming that each TransRequest message is 200 bytes, 5% of the link capacity is allocated to control traffic, and RTT is 20ms.

VI. Traffic Classes

In PATRICIA, an ingress border router divides traffic into four classes: *Control*, *Authorized*, *Regular*, and *Demoted*. Note that internal routers do not process any PATRICIA control messages or verify the authenticators in data packets. They simply forward the packets based on the priorities marked by the border routers. The **Control** class consists of endorsed control messages with relatively low traffic volume. This class has the highest priority to ensure timely delivery of control messages. The **Authorized** class contains data packets with valid authenticators. This class has the second priority. The **Regular** class contains data packets that do not require active traffic regulation and also has the second priority. The **Demoted** class contains data packets without correct authenticators and has the lowest priority. To prevent collateral damage, each class of traffic receives a *dedicated* queue at each router, and the first three classes are allocated their own share of bandwidth. To control flooding attacks within each traffic class, we apply RED-PD [14], a light-weight bandwidth sharing mechanism on a per source AS basis for control traffic, per-source-destination-pair basis for authorized traffic, and per-destination basis for regular traffic.

VII. Performance Evaluation

A. Simulation Setup

Fig. 3 shows our simulation topology with 20 source edge networks and 1 destination edge network. Below we describe the simulation setup for a typical experiment. We vary some of the parameter values to create other simulation scenarios.

Legitimate senders are generated and assigned randomly to one of the source networks with a Poisson arrival rate of 10/s and an exponentially distributed lifetime of 20 seconds on average. Based on the queueing theory of M/M/inf systems, there are on average 200 legitimate senders in the network. The attack starts at time 200s and the attackers arrive at a Poisson rate of 500/s, which means that the number of attackers quickly increases to 1000 in a few seconds. Once generated, these attackers do not leave the network until the simulation ends. Both legitimate senders and attackers send two 500-byte data packets per second. The legitimate senders alone will not cause server overload or congestion, but the total data rate from both categories of senders will overwhelm the server and congest some links.

The receiver processes up to 800 data packets per second with a queue size of 400 packets. If the queue occupancy level exceeds a threshold T_h, the receiver triggers a generic attacker identification mechanism with a certain false positive rate (p) and false negative rate (q). A higher p means incorrectly identifying more legitimate senders as attackers. For example, a p of 0.4 means incorrectly identifying 40% of the legitimate senders as attackers. On the other hand, a higher q means identifying more attackers as legitimate senders. The identification mechanism becomes inactive when the queue occupancy level falls below a threshold T_l and the number of identified attackers per second is less than a threshold N. In our simulation, $T_h = 80\%$, $T_l = 20\%$, $N = 10/s$. p and q are both between 0 and 0.5.

The receiver always accepts the first transmission requests from new senders. However, a sender must renew its authenticator every 2 seconds. This offers a chance for the receiver to reject misbehaving senders. Note that renewing the authorization does not involve any control messages; the new authenticator is carried in data packets. The receiver uses its attacker identification mechanism to determine whether to renew the authorization. If the identification mechanism detects an attacker, the receiver will send back a negative transmission confirmation, putting the sender on the blocking list. If a valid sender receives a negative transmission confirmation or does not get a confirmation message, the sender will try to establish the transmission channel using an exponential back-off mechanism. However, attackers continue to make their attempts at the same rate as before (2 requests per second).

B. Results

We conducted extensive simulations with various attack scenarios and obtained similar results. Here we present a set of representative results. Unless otherwise noted, we use the simulation parameter settings described in Section VII-A.

Figure 4 shows the total data throughput of legitimate users with PATRICIA or without (i.e. the current Internet). When the attack starts at 200s, the legitimate users' aggregate throughput drops from around 1.6Mbps to 800Kbps.

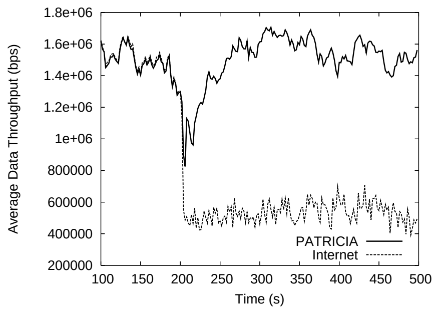


Figure 4. Data throughput of legitimate senders (False Positive Rate = 0.1, False Negative Rate = 0.2).

In the PATRICIA case, their throughput quickly returns to normal after 50 seconds or so. However, without PATRICIA, their throughput drops further to around 500Kbps as the attackers use up the bulk of the server's processing capability.

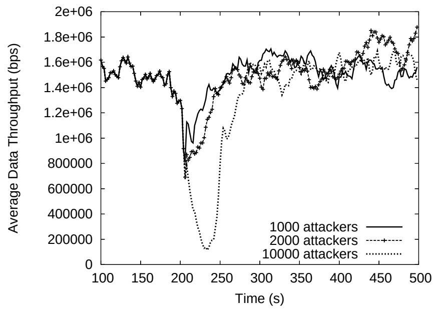


Figure 5. Data throughput of legitimate senders with different number of attackers (False Positive Rate = 0.1, False Negative Rate = 0.2).

To evaluate the performance of PATRICIA under larger attacks, we increase the number of attackers to 2000 and 10000. Figure 5 shows that more attackers can have a higher negative impact on the throughput of legitimate users, but even with 10000 attackers, the total throughput of the legitimate users returns to normal within 100 seconds or so. This is because in PATRICIA, the attackers are quickly identified and their traffic is blocked in their source networks.

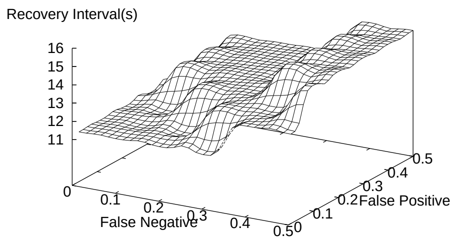
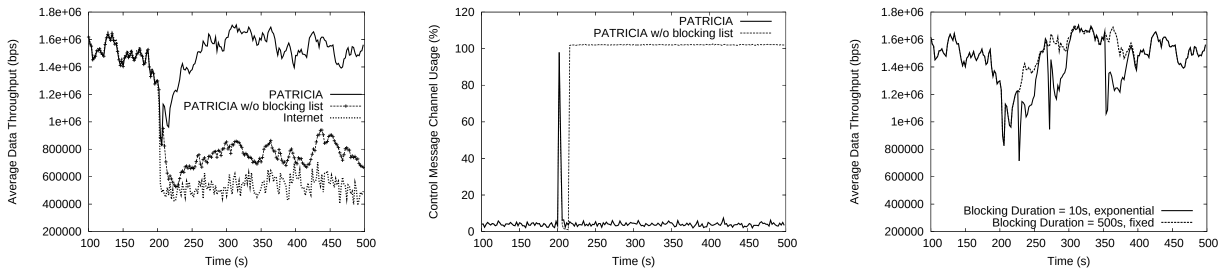


Figure 6. Impact of different attacker identification accuracies on how fast the receiver can recover from the attack.

In order to test how well PATRICIA works with different attacker identification mechanisms, we perform 36 simulations with the values of the false positive rate p and false negative rate q chosen from the set $\{0.01, 0.1, 0.2, 0.3, 0.4, 0.5\}$. We define the *recovery interval* as the duration in which the attacker identification mechanism remains active. A longer recovery interval indicates that it takes the PATRICIA networks more time to suppress the attack. Figure 6 compares the recovery interval under different identification mechanisms. It is clear that higher p



(a) Average throughput of legitimate senders. (b) Congestion of control channel. (c) Effect of different blocking durations.

Figure 7. Effectiveness of blocking list in mitigating control traffic flooding (False Positive Rate = 0.1, False Negative Rate = 0.2).

and higher q result in longer recovery intervals. However, the recovery interval still remains small (about 16 seconds) in the worst case. We also observe that the false negative rate has a slightly bigger impact on the recovery time than the false positive rate.

In the next simulation, we compare the throughput of legitimate senders in three cases: *PATRICIA*, *PATRICA without blocking list*, and *the Internet* (see Figure 7). We make the following observations from Fig. 7(a): (1) the Internet case has the lowest throughput – each legitimate sender receives a throughput of 2 - 3Kbps on average after the attack, even though their sending rate is 8Kbps; (2) PATRICIA performs the best among the three: each sender maintains an 8Kbps throughput after a short recovery period following the attack; (3) without the blocking list, PATRICIA is far less effective because the attackers can keep sending request messages to congest the control channel, as clearly shown in Fig. 7(b). *Our results demonstrate that the blocking list is an effective solution for mitigating control traffic flooding*.

In the above experiments, we set a fixed blocking duration of 500 seconds (i.e. a sender on the blocking list cannot send any messages for 500 seconds), but this long duration could be a problem for a legitimate sender that is incorrectly identified as an attacker. A good solution is to start with a short blocking duration and exponentially increase it every time a particular sender is detected as an attacker. Figure 7(c) shows that this method (with a 10-second initial value) approaches the performance of a fixed 500-second blocking duration 200 seconds after the attack starts.

VIII. Conclusion and Future Work

We presented the PATRICIA architecture and its control protocols for mitigating undesirable traffic. In the PATRICIA architecture, edge networks are involved in flow authorization between hosts to protect legitimate control messages against DoS attacks and mitigate collusion between senders and receivers. Active traffic regulation is only performed when necessary, thus reducing the processing overhead of both control and data traffic. Simulation results show that PATRICIA protects legitimate sources against flooding of both control and data traffic. In our future work, we will investigate approaches to distribute public keys, and refine the protection scheme against attack traffic to obsolete IP addresses. We also plan to develop prototypes of PATRICIA-capable hosts and routers.

Acknowledgment

We thank Dung Dinh Luong for implementing an earlier design of PATRICIA in ns2.

References

[1] J. Markoff, "Attack of the zombie computers is a growing threat, experts say," New York Times, Jan. 2007.

[2] X. Yang, D. Wetherall, and T. Anderson, "A DoS-limiting network architecture," in *Proceedings of the ACM SIGCOMM '05*, 2005, pp. 241–252.

[3] L. Wang, Q. Wu, and D. D. Luong, "Engaging edge networks in preventing and mitigating undesirable network traffic," in *Third Workshop on Secure Network Protocols*, Oct. 2007.

[4] P. Ferguson and D. Senie, "Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing." *RFC 2827*, May 2000.

[5] Z. M. Mao, V. Sekar, O. Spatscheck, J. van der Merwe, and R. Vasudevan, "Analyzing large DDoS attacks using multiple data sources," in *Proceedings of ACM SIGCOMM Workshop on Large-Scale Attack Defense (LSAD)*, 2006.

[6] J. Mirkovic, G. Prier, and P. Reiher, "Attacking DDoS at the source," in *Proceedings of ICNP 2002*, Nov. 2002.

[7] K. Argyraki and D. R. Cheriton, "Active Internet traffic filtering: Real-time response to Denial-of-Service attacks," in *USENIX Annual Technical Conference*, 2005.

[8] X. Liu, X. Yang, and Y. Lu, "To filter or to authorize: Network-layer dos defense against multimillion-node botnets," in *Proceedings of the ACM SIGCOMM '08*, Aug. 2008.

[9] T. Anderson, T. Roscoe, and D. Wetherall, "Preventing Internet Denial-of-Service with capabilities," in *Proceedings of Hotnets-II*, Nov. 2003.

[10] A. Yaar, A. Perrig, and D. Song, "SIFF: A stateless Internet flow filter to mitigate DDoS flooding attacks," in *Proceedings of the IEEE Security and Privacy Symposium*, May 2004.

[11] K. Argyraki and D. R. Cheriton, "Network capabilities: The good, the bad and the ugly," in *Proceedings of the ACM Hot Topics in Networks (HotNets) Workshop*, 2005.

[12] A. Langley, "A rabin-williams signature scheme," http://github.com/agl/rwb0fuz1024/raw/master/rwb0fuz1024.pdf, Aug. 2008.

[13] L. von Ahn, M. Blum, N. Hopper, and J. Langford, "CAPTCHA: Using Hard AI Problems for Security," in *Proceedings of Eurocrypt*, 2003.

[14] R. Mahajan, S. Floyd, and D. Wetherall, "Controlling high-bandwidth flows at the congested router," in *Proceedings of ICNP '01*, Nov. 2001.