Defending Network-based Services Against Denial of Service Attacks

Jinu Kurian and Kamil Sarac
(Corresponding author: Jinu Kurian)

Department of Computer Science, University of Texas at Dallas
Richardson, TX 75080, USA (Email: jinuk@student.utdallas.edu)

(Received June 14, 2008; revised and accepted Dec. 12, 2008)

Abstract

Over the last decade, several value-added services have been proposed for deployment in the Internet. Many of these services (e.g. IP Multicast) are *stateful* services introducing state maintenance overhead into the network for their operation. This characteristic makes these services vulnerable to a specific type of denial-of-service (DoS) attacks called *state overload attack*. In this paper, we examine state overload attacks in value-added services and in particular IP multicast. We describe why these attacks are possible and present two solutions to prevent them. In both cases, we describe the solutions, evaluate their overhead, and outline incremental deployment strategies for their deployment.

Keywords: Multicast security, state overload attacks, value-added services, denial-of-service, DoS defense

1 Introduction

Over the last decade, several *value-added* services have been proposed for deployment in the Internet. These include IP multicast [1], QoS support [15], content distribution networks [8], and denial of service (DoS) defense mechanisms [13] among others. These services provide users and service providers with an array of added capabilities. They also provide ISPs with an opportunity to provide a new set of services to draw additional revenue from their users. However, compared to the stateless nature of the traditional best-effort IP forwarding, some of the above-mentioned value-added services introduce additional state and processing overhead into the network. If the state introduced is not carefully controlled, it can be subject to abuse which can be used to cripple the service. The state overhead for example can be a means to launch DoS attacks on the service and its users. In this paper, we aim to demonstrate the vulnerability of stateful services and in particular IP multicast to DoS attacks. We examine why these attacks are possible and present two solutions to prevent them. One high level lesson that we take from this study is the realization of the difficulties in introducing value-added network services without creating a significant level of additional overhead and security vulnerability for the network and its users.

IP multicast was one of the first value-added services to be developed and partially deployed in the Internet [1]. Despite the well known advantages of IP multicast in supporting multi-receiver network applications, the deployment of IP multicast in a large scale has been limited due to various architectural and security flaws in existing protocols [2, 12, 14]. More recently, the popularization of the IPTV application has triggered a new synergy in the market and facilitated more deployment of the IP multicast service in the Internet [17].

The standard protocol used today to build and maintain multicast trees is Protocol Independent Multicast (PIM) [3]. In PIM, multicast receivers cause the generation of group join requests which are forwarded towards the source of the multicast group. In response to these join requests, multicast enabled routers in the path upstream to the source create and maintain state entries in forwarding state buffers. If these finite state buffers are exhausted due to excessive state, future join requests cannot be processed and may be denied. This stateful nature of the join mechanism makes the multicast service vulnerable to DoS attacks called *state overload* attacks against multicast-enabled routers [14].

Based on the intended target of the attack, state overload attacks can be one of two types. In a *directed end-system attack*, the target is an end-system or its subnet. The objective is to thwart the end-system from sourcing or receiving multicast content. By overloading the state buffers at routers in its vicinity, a DoS attack can be executed against a multicast source (e.g. an Internet TV station), thereby preventing new customers from joining and receiving data. The second type of attack is called *infrastructure attack* and is of a much larger scale. In this attack, the target is the network infrastructure itself; e.g., a group of core routers in the multicast network backbone. Since the core routers are required to maintain state for potentially all multicast groups in their networks, an attack on them can be carried out for almost any source-destination pair the attacker desires. If successful, the infrastructure attack may have an impact on a large number of multicast users competing for the limited state buffers in the target domain. Both types of attacks are relatively easy to launch and can significantly impact the availability of multicast service at their targets.

In this paper, we propose two proactive solutions to defend against state overload attacks in IP multicast. The objective of these solutions is to ensure that multicast enabled routers create state only if necessary, thereby preventing state overload with unwanted state in the routers. The first solution proposes some modifications to the PIM protocol to eliminate the vulnerability in the protocol that makes these attacks possible. This solution is comprehensive and eliminates the problem completely. While the solution is partially deployable, it is heavyweight and will require a transition period during which non-deploying domains will continue to be vulnerable to attack. Our second solution aims to rectify this. The solution proposes a distributed overlay infrastructure to validate join requests before they are processed by the routers. The advantage of this solution is that it can be deployed without any modifications required in the current PIM protocol or multicast infrastructure. We evaluate the added overhead and the effectiveness of both approaches using a combination of simulation and implementation. Based on our evaluations we observe that our solutions are highly effective in preventing state overload attacks.

The rest of this paper is organized as follows. Section 2 summarizes how the PIM-Join mechanism works and why it is vulnerable to attack. Section 3 describes related work in the area. Section 4 describes the first solution called the enhanced join method, including its operation, partial deployment strategies and evaluation results. Section 5 describes the second solution called the overlay based indirection method, its architecture and operation, a discussion of some security and partial deployment concerns, and evaluation results. Finally, Section 6 concludes the paper.

2 Problem Description

2.1 PIM-Join Mechanism

PIM supports two types of join operations: (1) shared tree joins, and (2) source specific joins. Shared tree joins are used to establish a shared tree between the receivers and a pre-selected router, called the Rendezvous Point (RP). Since the RP is a domain-local router, the PIM-Join message and the state created are also local. Hence, state overload attacks using shared tree joins can have a localized effect. In source specific joins, the receivers join directly to the multicast source, S, of a group, (S,G). Since S can be located anywhere in the Internet, attacks via source specific joins are extremely potent, impacting potentially any local or remote victim sites. Therefore, we focus on source specific join attacks in this paper.

In PIM when a receiver R desires to join a multicast group (S,G), it creates an IGMP INCLUDE message for the desired group and sends it to its designated router DR(R). Upon receiving the INCLUDE message, DR(R) creates a new Join(S,G) message and forwards it towards the designated router of the source, DR(S). All the routers between DR(R) and DR(S) create forwarding state for the (S,G) group as the PIM-Join message propagates towards S. Routers forward PIM-Join messages on their shortest path interface towards the source. This interface is called the *incoming interface* (*iif*) or *reverse path forwarding* interface (Int_{RPF}) for the group. For each *iif* entry, the router also includes all the interfaces from which it received PIM-Join messages in an *outgoing interface list* (*oif*) for the group. A router needs to create new forwarding state for a PIM-Join for each distinct (S,G) group. In source specific multicast, a single source can support up to 2^{24} multicast groups, all of which can be used to generate distinct Join(S,G_i) messages.

The forwarding state created is "soft", i.e., it expires if no refreshing PIM-Join messages arrive from downstream. Each state entry is associated with an *entry-timer* (ET) and each interface in the *oif* is associated with an *oif-timer* (OT). If the ET expires and the *oif* becomes empty, the router sends a PIM-Prune(S,G) message on its *iif* interface for the group and leaves the multicast tree. When an upstream router receives a PIM-Prune message on an interface, *i*, in *oif*, it removes *i* from *oif*. Alternatively, if no refreshing PIM-Join message arrives on *i* during a Join Hold Time period (the default is 260 seconds), OT_i expires and *i* is removed from *oif*. The protocol is visually presented in Figure 1.

2.2 State Overload Attacks

There are two vulnerabilities in the PIM protocol that are exploited in a state overload attack. The first vulnerability is in the PIM-Join procedure which requires that DR(R) issue a PIM-Join message without verifying whether the source or the group requested in the PIM-Join message exist. The second vulnerability is that the protocol requires the creation of state information in response to any PIM-Join message for its correct operation. The PIM-Join(S,G) message (legitimate or bogus) propagates towards S creating forwarding state at all routers on the R-to-S path. Since the routers do not have any mechanism to verify the validity/existence of the source or the group, they will maintain the (S,G) state as long as R, who could be an attacker, sends refresh messages.

Consider an attack scenario that proceeds in rounds. For the first round of 260 seconds (or the local default Join Hold Time), the attacker generates distinct bogus PIM-Join messages to create unwanted state information in the routers. In subsequent rounds, it sends refresh messages to continue to maintain the state at the routers. If there are multiple attackers, the amount of state information maintained at routers could be prohibitively large. For example, in a directed end system attack, if there

```
1 /* Consider a router $R_j$ */
2 On receiving PIM-Join(S,G) on an interface $i$
3     IF (S,G) state exists and ($i \notin oif$) THEN
4         $oif = oif \cup \{i\}$
5     IF (S,G) state NOT exists THEN
6         Create (S,G) state with $iif = \text{Int}_{RPF}(S)$; $oif = \{i\}$
7         IF NOT DR(S) THEN
8             Forward PIM-Join(S,G) on $\text{Int}_{RPF}(S)$

9 On receiving Prune(S,G) on an interface $i$ or OT for $i$ expires
10    Remove $i$ from $oif$ of (S,G)
11    IF $oif$ is empty and ET expires THEN
12        Send PIM-Prune(S,G) on $\text{Int}_{RPF}(S)$
13        Remove (S,G) state from forwarding state table
```

Figure 1: Source specific joins in PIM

are 5000 zombies (attackers), with each zombie issuing 10 separate PIM-Join requests per minute, at the end of the first round, the routers at the target site will need to store more than 200 000 different multicast entries. Similarly, an infrastructure attack can be launched with attackers choosing highly used paths to target routers at the core of the network. In this case, the attacks are less targeted and the state created would potentially be distributed among multiple core routers. Considering the same attack parameters as before, in the worst case, a core router may end up storing up to 200 000 different entries. In both attacks, the number of created entries can be large.

The above discussion suggests that a solution to the problem must involve a verification of the validity of the source and the groups being subscribed to in a PIM-Join message. We expect that the routers are suitably provisioned to handle legitimate PIM-Join requests so that joins to legitimate groups will not create an attack. In the rest of this paper, we present two approaches to include such a verification mechanism in the multicast join process. To simplify the initial discussion, we assume that attacks which may involve sources and receivers cooperating with each other to overcome the verification mechanism are not possible. We will address these attacks in more detail in Section 4.3.

3 Related Work

One basic idea to defend against state overload attacks is to have DR(R) rate limit joins and prunes originating from end-hosts in its subnet [6, 14]. Rate limiting can be performed based on several factors [14] like the number of PIM-Join requests an end-host/subnet generates, whether the host is spoofing its address, whether the host is willfully malicious or has a history of good behavior etc. Also, rate limiting can be performed on the number of PIM-Join requests generated from the DR as a whole or be specific to a single host or subnet, a group address, source-group address or source-receiver pair etc. Rate limiting can be effective against state overload attacks that involve one or more attacking hosts within the same subnet. However, the factors to be taken into account to perform rate limiting are difficult to measure and can lead to false positives which acerbate the problem. Furthermore, it may not be effective if the attack is sufficiently distributed to overcome locally enforced rate limitations.

Defending against state overload attacks has been partially addressed in a few previously proposed solutions. Gronvall [5] suggests the use of Bloom filters to reduce the amount of state that is required to be maintained at multicast routers for join forwarding. MAFIA [11] is a multicast management solution which aims to strengthen multicast security through access control and traffic filtering. MAFIA utilizes group membership information available at different locations in a network for this purpose. One important location is the intra-domain network boundary. MAFIA servers deployed at intra-domain boundaries can monitor and limit outgoing PIM-Join requests. For inbound protection the MAFIA server can rate limit or filter incoming data traffic.

The Multicast Control Protocol (MCOP) aims to control user access to multicast services in the intra-domain [9]. MCOP uses access control lists to limit multicast traffic and IGMP messages. These access control lists can be remotely controlled allowing operators in the local or remote domains to limit undesired traffic from the MCOP domain. MCOP by itself is a protocol which allows for the communication between two remote entities, the Multicast Control Server (MCS) and the Multicast Control Client (MCC). MCS is a database which maintains information about valid groups, receivers and sources for multicast groups. The MCC can query the MCS to obtain information about the validity and permissions of a receiver or source for access controlled filtering.

Our second solution is an overlay based architecture that builds on the techniques used in MCOP and MAFIA. The main difference lies in the fact our solution is targeted directly at preventing state overload attacks at the inter-domain scale. We also ensure that the infrastructure by itself is protected from DoS attacks. Additionally, subverting nodes in the network to launch attacks is not possible due to the independent decisions made by each domain the join path. Our first solution is a novel approach which aims to eliminate the problem completely. It achieves this by enhancing the PIM protocol to make it more secure and hence prevents these attacks completely.

4 Solution 1: Enhanced Multicast Joins

Our first solution is a comprehensive approach which aims to eliminate the vulnerability in the PIM-Join procedure that is exploited in a state overload attack. Our objective is to ensure that every router can verify the validity of any PIM-Join message it receives. Before creating any state, every router in the forwarding path will individually verify the validity of the source and the group being subscribed to in the join. This ensures that bogus PIM-Join messages sent by malicious receivers cannot create unwanted state in the routers.

During join forwarding, routers do not create any forwarding state, but instead add the requisite state information to the PIM-Join message before sending it upstream towards the source. Each on-tree router, R_j, appends this state information as a nonce, say N_j, to the end of a nonce block in a new Join(S,G,N) message. The state information added includes the incoming interface, i_j, of the PIM-Join message and a secure hash of all the locally added state, H_j (a detailed description of the state added is deferred to Section 4.2).

If the source and the group in the Join(S,G,N) message are valid, the accumulated state information is returned by DR(S) in a new JoinACK(S,G,N) message. Each router, R_j, in the return path *individually* verifies the JoinACK(S,G,N) by recomputing the secure hash H_j with the relevant state information in the nonce N_j. This ensures that the received JoinACK(S,G,N) is a valid acknowledgment of the Join(S,G,N) that R_j had previously forwarded upstream. Once the verification is complete, R_j creates a forwarding entry for (S,G) with i_j as the *oif* and Int_{RPF}(S) as the *iif* for the group. Once the JoinACK reaches and is verified by DR(R), the join process is complete (see Figure 2).

Note here that as soon as the JoinACK is generated by DR(S), multicast data can be sent downstream along with the JoinACK. The JoinACK creates the multicast tree as it propagates downstream and the multicast data can flow along the newly created tree. As is the case with the unmodified PIM-Join process, data will not flow on the tree until a PIM-Join message is received at the designated router of the source. Therefore, there will be no loss of data due to the additional messaging required

4.1 PIM Router Operation Under the Modified Join Procedure

We now present a detailed description of operation of a router after the proposed modifications to the PIM-Join mechanism. The protocol is visually presented in Figure 3. A modified PIM router R_j can accept two types of PIM-Join messages, the legacy Join(S,G) message and the modified Join(S,G,N) message. On receiving a Join(S,G) or a Join(S,G,N), R_j checks to see if it already has preexisting state for the (S,G) group. If it does and the incoming interface i upon which the message is received is already in the *oif*, the *entry-timer* and *oif-timer* for the (S,G) entry are refreshed. If the state exists, but the interface i is not in the *oif*-list, i is added to the *oif*-list. In the case of a Join(S,G,N), a JoinACK is sent downstream along i. This JoinACK is required for the routers downstream to complete their join process in the modified protocol.

In both Join(S,G) and Join(S,G,N) cases, if the (S,G) entry does not exist, it implies a join request for a new multicast tree. In this case, R_j checks to see if it is the designated router DR(S) for the (S,G) group. If it is the designated router, R_j checks to see if the (S,G) group is valid, and if so creates a state entry for the group and adds i to the *oif*-list for the entry. In the case of a Join(S,G,N) it additionally returns a JoinACK along the interface i. If R_j is not the designated router, it creates a nonce N_j with the required information (see Section 4.2), and appends it to the incoming Join(S,G) or Join(S,G,N) message. The modified Join(S,G,N) message is sent upstream towards the source on Int_{RPF}(S). Note here that unlike in a normal join procedure, no state is created during join forwarding. Also, as can be seen from the above description, the proposed modifications do not introduce any backward compatibility problems as the routers can process both the existing Join(S,G) and the proposed Join(S,G,N) messages.

The modified PIM router R_j can also accept a JoinACK(S,G,N) message from an upstream router. Similar to a PIM-Join in the legacy PIM protocol, the JoinACK message creates (S,G) state and initializes the *entry-timer* and *oif-timer* for the state entry. Before creating the state, R_j verifies (see Section 4.2) the nonce N_j at the end of the nonce block in the JoinACK message. Once the nonce is verified and the state is created, it removes its nonce N_j from the end of the nonce block and forwards the JoinACK(S,G,N) downstream (if the nonce block is not empty) along the interface specified as the incoming interface i in the JoinACK. Finally, the Prune(S,G) message on an interface i removes i from the (S,G) state entry at the receiving router. If after the removal of i the *oif*-list becomes empty, the router can remove itself from the multicast tree. For this purpose it sends a Prune(S,G) upstream towards the source on Int_{RPF}(S) to remove itself from the tree.

The state diagram of a router operating with our modified PIM protocol is shown in Figure 4. It has two

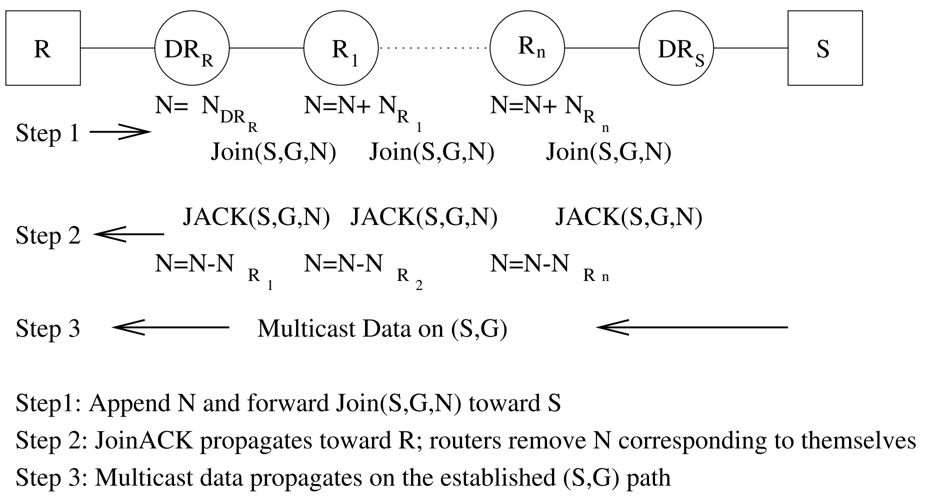

R
DR_R
R_1
R_n
DR_S
S
Step 1
N= N_{DR_R}
N=N+ N_{R_1}
N=N+ N_{R_n}
Join(S,G,N) Join(S,G,N) Join(S,G,N)
Step 2
JACK(S,G,N) JACK(S,G,N) JACK(S,G,N)
N=N-N_{R_1} N=N-N_{R_2} N=N-N_{R_n}
Step 3
Multicast Data on (S,G)
Step1: Append N and forward Join(S,G,N) toward S
Step 2: JoinACK propagates toward R; routers remove N corresponding to themselves
Step 3: Multicast data propagates on the established (S,G) path

Figure 2: Modified PIM-Join procedure

```
1  /* At a router $R_j$ */
2      On receiving Join(S,G) on an interface $i$
3      IF (S,G) state exists at and ($i \notin oif$)THEN
4          $oif = oif \cup \{i\}$
5      IF (S,G) state NOT exists THEN
6          IF $R_j$ = DR(S) THEN
7              IF (S,G) group is valid
8                  Create (S,G) state with $iif$ = Int$_{RPF}$(S); $oif = i$
9          ELSE
10             Compute N$_j$ and Append N$_j$ to N
11             Send Join(S,G,N) on interface $k$ = Int$_{RPF}$(S) toward S

12     On receiving Join(S,G,N) on an interface $i$
13     IF (S,G) state exists at and ($i \notin oif$) THEN
14         $oif = oif \cup \{i\}$ AND Send JoinACK(S,G,N) on $i$
15     IF (S,G) state NOT exist THEN
16         IF $R_j$ = DR(S) THEN
17             IF (S,G) group is valid
18                 Create (S,G) state with $iif$ = Int$_{RPF}$(S); $oif = i$
19                 Send JoinACK(S,G,N) on $i$
20         ELSE
21             Append N$_j$ to N
22             Send Join(S,G,N) on interface $k$ = Int$_{RPF}$(S)

23     On receiving JoinACK(S,G,N) on an interface $k$ for a Join(S,G,N)
24     IF k = Int$_{RPF}$(S) THEN
25         Recompute and verify N$_j$
26         IF (S,G) state NOT exists
27             Create (S,G) state with $oif = i$; $iif$ = Int$_{RPF}$(S)
28             Modify N = N - N$_j$
29             Send JoinACK(S,G,N) on $i$

30     On receiving Prune(S,G) on an interface $i$ or OT for $i$ expires
31     Remove $oif = oif - \{i\}$
32     IF $oif$ is empty THEN
33         Send Prune(S,G) on Int$_{RPF}$(S)
34         Remove (S,G) state from forwarding state table
```

Figure 3: Router operation for modified PIM joins

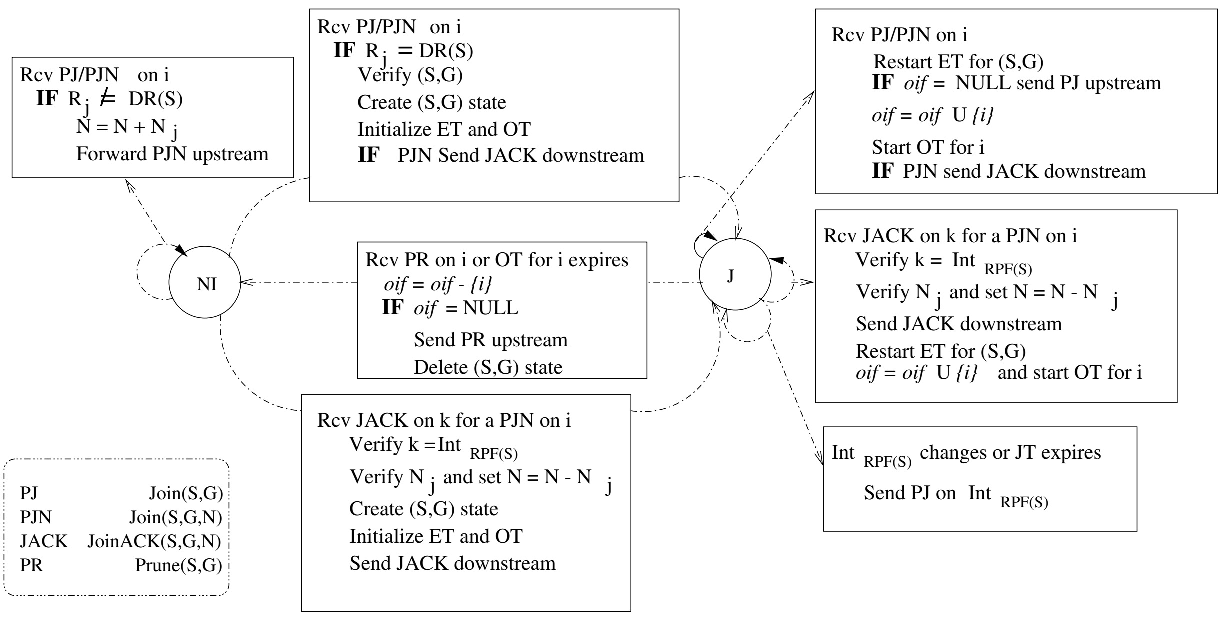


Figure 4: State machine of a router, R_j, with the modified PIM protocol

states: No-Info (NI) and Joined (J). In the NI state, the router has no knowledge of the existence of a group, i.e., it maintains no (S,G) state about the group. In the J state the router maintains a forwarding entry for (S,G), and keeps an *entry-timer* (ET) and an *oif-timer* (OT) for each interface in *oif* for (S,G). We briefly discuss the state transitions for the routers below:

NI State. In this state, there are two events causing router R$_j$ to take different actions:

(1) Receiving JoinACK(S,G,N) on interface, k, for a Join(S,G,N): Verify k = Int$_{RPF}$(S); verify N$_j$ and update N = N - N$_j$; forward JoinACK(S,G,N) if required; create (S,G) state; initialize ET and OT timers; and move to J state.

(2-a) $R_j = DR(S)$ and receives Join(S,G)/Join(S,G,N): Verify the validity of (S,G) (by checking with S); create (S,G) state; initialize ET and OT timers; in the case of Join(S,G,N), send a JoinACK(S,G,N) on the interface in *oif*; and move to J state.

(2-b) $R_j \neq DR(S)$ and receives Join(S,G)/Join(S,G,N): Add relevant state, N$_j$, to the incoming join message and forward a Join(S,G,N) message toward S on Int$_{RPF}$(S).

J State. In this state there are four events causing router R$_j$ to take different actions:

- Receiving Join(S,G)/Join(S,G,N) on interface, i: Refresh its ET for (S,G); set *oif* = *oif* $\cup\{i\}$ and start OT for i. In addition, on Join(S,G,N), send a JoinACK(S,G,N) on i.
- Receiving JoinACK(S,G,N) on interface, k, for a Join(S,G,N) on i: Verify $k = Int_{RPF}(S)$; verify N_j and update $N = N - N_j$; forward JoinACK(S,G,N) on interface i toward R_{j-1}. Set *oif* = *oif* $\cup\{i\}$ and start OT for i.
- $Int_{RPF}(S)$ change or join timer expiry: Send Join(S,G) on $Int_{RPF}(S)$. Here R_j issues a Join(S,G) rather than Join(S,G,N) as the (S,G) is already verified prior to the creation of (S,G) state at R_j. In this case, $Int_{RPF}(S)$ could change due to a unicast routing change. A join timer is used to trigger the transmission of periodic refresh messages upstream.
- Receiving Prune(S,G) on interface, i, or OT for i expires: Set *oif* = *oif* - $\{i\}$. If *oif* = $\emptyset$, send Prune(S,G) on Int$_{RPF}$(S); remove (S,G) state; and move to NI state.

4.2 Authenticating Joins

In the modified join procedure, every router, R$_j$, adds a nonce, N$_j$, to a Join(S,G,N) message. When the JoinACK(S,G,N) is returned, R$_j$ retrieves N$_j$ for verification and then creates the forwarding state for (S,G). N$_j$ has to carry the requisite information which will allow R$_j$ to create the forwarding state for (S,G). It should also carry path information to ensure that the JoinACK(S,G,N) is returned downstream along the same path as the original join upstream. Additionally, the nonce has to be secure against modification, brute force, and replay attacks during its valid duration.

Routers can create this nonce by including (1) the 16-bit incoming interface, i, for the incoming join, (2) the lower order 16 bits of the IP address of the downstream

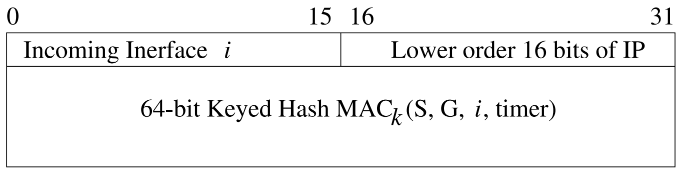

0
15
16
31
Incoming Inerface i
Lower order 16 bits of IP
64-bit Keyed Hash MAC_k(S, G, i, timer)

Figure 5: Nonce, N_j, added at router, R_j

router forwarding the join, and (3) a keyed-hash or MAC of the group address, the incoming join interface, i, and an ascending counter T (see Figure 5). The nonce created is bound to a specific group, (S,G), and interface, i. The nonce, N_j, is then appended to the end of the nonce block, N, in the join message before it is forwarded upstream. The combination of 16-bit interface ID, i, and the lower order 16 bits of the IP address of the downstream router enable the current router, R_j, to derive the reverse path for the JoinACK when it is received later from an upstream router. Note here that the JoinACK will always take the reverse of the original join path, irrespective of the network level path. This is because the path taken by the original join is included in the nonce. The JoinACK will follow the reverse of this recorded path back to R. A keyed hash like HMAC-SHA or HMAC-MD5 can be used to create a 64-bit hash string in the nonce. The hash creation can be done without modification and without noticeable overhead in most routers. The secret key, k, is randomly generated by the router and can be varied at a rate slower than the clock to provide added security against brute force attacks on the nonce.

On the return path, when a router, R_j, receives a JoinACK(S,G,N), it first performs an RPF check on the incoming interface of the JoinACK(S,G,N). The RPF check amounts to a lightweight authentication of the upstream router. The RPF check along with the presence of a timer in the hash prevents replay attacks using the same nonce. R_j then extracts N_j from the head of N. N_j includes i and (S,G). The router uses the extracted information and its current (or last few) keys to create a 64-bit keyed hash of (S,G), i, and the current (or last few) counter values. It then authenticates the JoinACK by comparing this hash with the one in N_j that arrived with the JoinACK. After this verification is complete, the router proceeds with the rest of the join process as described in Section 4.1.

4.3 Discussion

We now briefly comment on some important issues and questions about our proposed solution. One naive solution to state overload attacks may involve the designated multicast router DR(R) at a receiver site R to send a probe towards DR(S) to verify the existence of a remote source S or a group (S,G). This may not be effective all the time. Instead of issuing an IGMP-based group join request, an attacker can establish PIM Neighborhood relation with its DR router and can send a Join(S,G) request to get around this protection mechanism. Also this method cannot prevent an attacker from injecting bogus state information into the channel path. An effective solution in this case requires all the routers to verify the existence of the source S or the group (S,G) by sending probes to the source S. But this by itself creates a flooding attack toward the source site S and hence cannot be used effectively.

The modified PIM protocol effectively prevents a receiver from overloading routers with bogus (S,G) states. This feature however assumes that the sender is not maliciously sourcing bogus groups. In this case, malicious senders and receivers can co-operate with each other to launch an infrastructure attack. To protect against malicious sources, the modified protocol can be combined with source authentication mechanisms such as MAFIA [11] at the source's domain. MAFIA as we discussed earlier in Section 3 operates by maintaining information about valid groups in each domain. By itself, this validity information provided by MAFIA may not be effective against state overload attacks due to reasons discussed for the naive solution above. But, using MAFIA as an additional component in our solution can ensure the legitimacy of both the source and group being subscribed to in the join. So, collaboration between malicious receivers and senders can be prevented.

Another issue is related to packet fragmentation. Since, in our solution, routers append information to the forwarded join messages, fragmentation possibilities should be considered carefully. The standard PIM-Join message, which includes a single group subscription, is 24 bytes. In our approach, each on-tree router appends 12 bytes to the join message. Therefore, the size of a modified join message is `24 +(12*n)` where `n` is the number of routers on the path. Assuming `n=40` which is a safe estimate for the Internet today, the maximum size of a modified join message is 504 bytes. Given the fact that IP requires a minimum MTU of 576 bytes, the proposed approach should not cause any fragmentation.

Another issue is the possible loss of JoinACK packets in the network. In such a situation, the routers from the point of loss in the reverse path downstream will not create state entries while state entries will be created upstream. In this scenario, if the receiver does not issue a new join request, the created states will be dissolved upon timeout. If the receiver issues a new join request before the timeout, the JoinACK will be returned by the first router with an established state entry upstream and the join procedure will be completed as normal.

4.4 Partial Deployment Scenario

Our proposed solution requires all PIM routers to be updated to support the modified join operation. In this section, we consider a method which can provide a temporary solution to ISPs supporting our protocol when the neighboring domains do not support the modifications. In this discussion, we refer to routers with the updated PIM protocol as modified routers and the routers employing

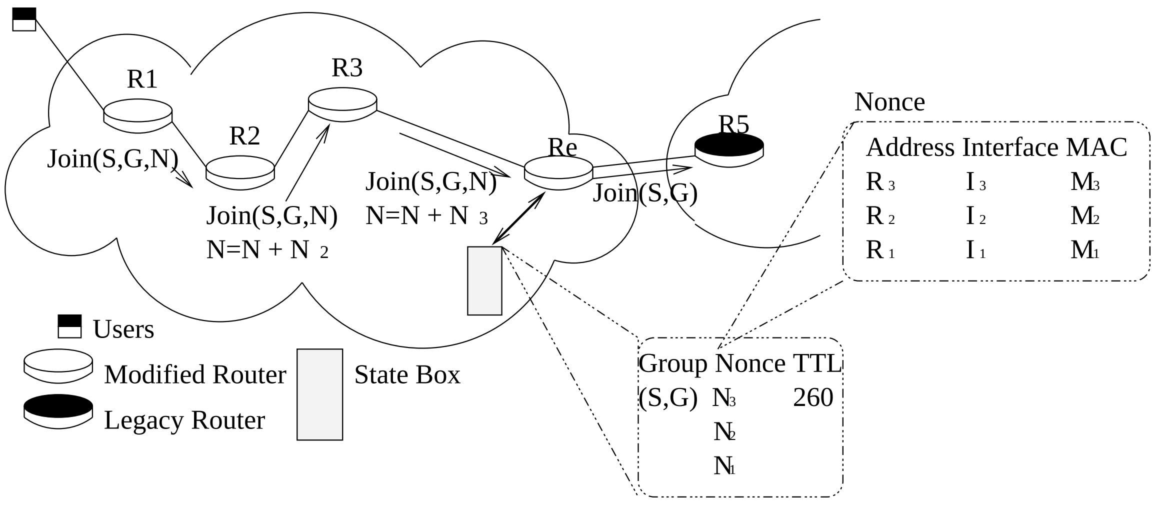


Figure 6: Partial deployment case

the non-updated PIM version as legacy routers. For our protocol to function properly, the modified routers require a valid JoinACK message from their upstream neighbors. Downstream routers can be legacy routers without affecting the protocol which will function normally in the domains with modified routers. If a modified router is a domain edge router having a PIM neighborhood relationship with a legacy router of a neighboring domain, it will not be able to receive JoinACK messages.

To deal with such cases, we introduce a proxy-based approach for an ISP supporting our proposed protocol. In this approach, the ISP can deploy *state boxes* at the edges of its domain. The state boxes are high capacity storage devices capable of handling large amounts of data. When an edge router, R_e, of the domain detects (as a result of periodic PIM Hello message exchange) that its next hop neighbor in the neighboring domain is a legacy router, it removes and forwards the accumulated nonce information from the join messages to the local state box. This state is maintained for a short duration (e.g., 260 seconds), and is indexed under the appropriate (S,G) value of the incoming join message. The edge router, R_e, then forwards an unmodified join message upstream towards the source.

If the source is valid and is transmitting regularly, its data will flow down the established path to the edge router, R_e. R_e verifies with the state box if an entry for this (S,G) exists in it. If an entry exists, R_e retrieves the state information from the local state box and issues a JoinACK with the stored state information downstream, thereby establishing forwarding state in the routers in its domain. This state caching operation is visually presented in Figure 6. Until the (S,G) group is verified as valid and the JoinACK is issued, all multicast data for the group from upstream will be redirected by R_e for temporary storage at the state box. After the JoinACK is issued, the buffered multicast data for the (S,G) group is retrieved from the state box and sent downstream along the newly established multicast path towards the receiver. If the source is invalid or has not transmitted for a long period of time, the state is dissolved at the state box to reclaim the state buffer occupied by this state.

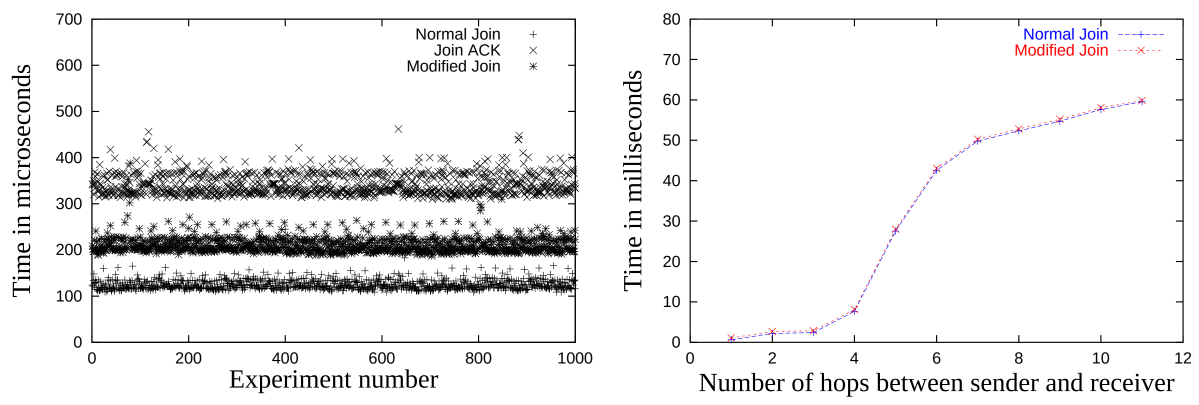
In this solution, the state boxes are possible points of attack if the attack volume is excessively high. Considering the numbers used in Section 2 and with a worst-case scenario of 40 hops between the attackers and R_e, this could amount to `2.6M*504=48MB` of state information at the end of 260 seconds. Common storage devices are available today with capacities up to several terabytes, hence overflowing a state-box with excess state is implausible. The state buffers used to buffer the multicast data can create another possible point of attack. However, unlike the state box, the amount of multicast data that needs to be buffered is unrelated to the volume of attack traffic. It depends only on the number of legitimate groups in the domain (note that spurious groups will produce no multicast data to be buffered). A circular buffer which stores the data only for a limited period of time is sufficient for most multicast applications like streaming video and other real time traffic.

4.5 Evaluations

In this section, we evaluate the overhead introduced by our modified PIM protocol and its performance under DoS attacks.

Processing overhead at a router: We use a Linux-based router to measure the processing overhead in computing and verifying the nonce in the modified joins. For this measurement, we use the sample implementation of HMAC-MD5 from RFC 2104 [7]. For the unmodified version of the PIM protocol, we use the implementation available in the Linux kernel.

Our metric for comparison is the total time taken from reception of a join message to processing and forwarding it upstream. For the modified protocol, we also measured the time taken from reception of a JoinACK from



(a) Processing overhead at a router. (b) Latency perceived by receiver.

Figure 7: Processing overhead and latency of network layer solution

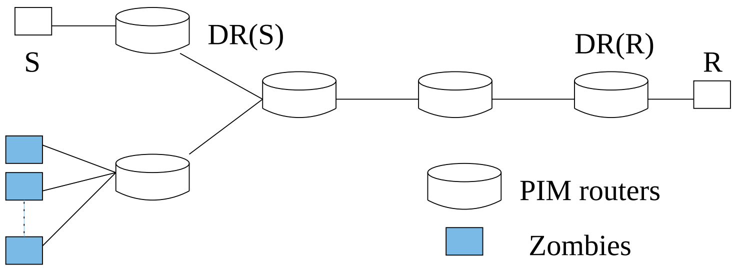
upstream to verify it and send it downstream. The join requests are generated at a rate of 5000 requests/second and the averaged time taken over 5 seconds was measured. Each experiment was repeated 1000 times. Figure 7(a) shows that per-node processing overhead is about 4 times higher for a modified PIM-Join as compared to a normal PIM-Join. However, from an end-user's point of view, the perceived latency is a more important metric as it indicates the overall performance impact of the modified protocol. The perceived latency includes additional components like queuing and propagation delay at routers. To evaluate the end-to-end latency as perceived by a user, we performed NS2 [4] (ns-2.26) simulations to compare the delay incurred in the modified and unmodified cases. The metric of interest is the total time from the user issuing a PIM-Join request until it starts receiving multicast data from the group. Here, we assume that as soon as the PIM-Join request (modified or unmodified) arrives at the source site, the source starts sending multicast data.

In our simulations, we used the 90th percentile values from the previous experiments as the nodal processing time incurred at on-tree PIM routers. Figure 7(b) presents the results of our simulations. As can be seen from the figure, the total latency, as perceived by an end-user, is virtually identical for both cases. This result is because, in a network, the inter-nodal latency, which is on the order of milliseconds, becomes much more significant than the per-node processing overhead, which is on the order of microseconds. As a result, the end-user perceives very little difference in the delay introduced by these two protocols.

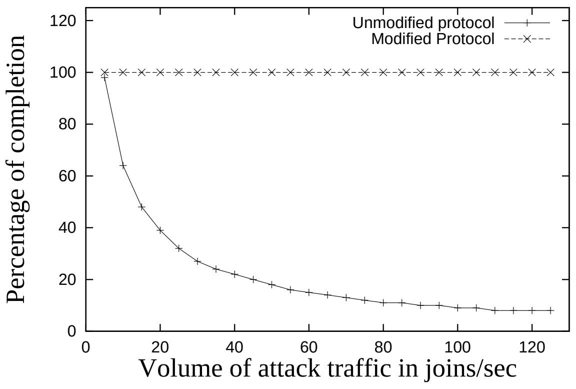
Percentage of completed joins under attack: To evaluate the resistance of the modified protocol to state overload attacks, we performed various experiments in NS2 [4] (ns-2.26) using a simulated network topology (Figure 8(a)). In our experiments, users attempt to join a remote group while the routers on the path are subject to state overload attacks of varying magnitudes. The evaluations are performed under the assumption that there is no loss in the network because the objective of the attack is not to congest the network but to overload the routers. In addition, we assume that the designated router at the source site can distinguish between legitimate and malicious joins based on the group address in the join message. As we showed in Section 2, with a distributed attack generated by 5000 zombies, routers in the vicinity of the victim may need to store as many as 200 000 entries. To simplify our simulation, we consider attacks of smaller magnitudes (25 zombies at its peak) but use a smaller state buffer threshold (i.e., the number of entries a router can accommodate before it starts dropping new requests) to acerbate the effect of the attack.

In the simulations, a legitimate user issues join requests at the rate of 5 joins/sec while the attack traffic is varied from 0 to 125 joins/sec. The results displayed are for a state buffer threshold value of 200 entries. The metric used is the percentage of completed legitimate joins. Figure 8(b) shows the results for the modified and the unmodified protocol. As can be seen in the unmodified protocol, the percentage of completed joins decreases exponentially as the rate of attack increases because the legitimate and the attack traffic compete for the same limited buffer space. The modified protocol meanwhile maintains a 100% completion rate because only legitimate joins create state.

Performance comparison with MCOP and MAFIA: The primary advantage of our solution when compared to prior solutions like MCOP and MAFIA is in its distributed nature. Our solution avoids the need for a centralized or partially centralized database which tracks legitimate sources and groups. Additionally since each router independently makes its own decision, compromising a single router has minimal effect on the operation of the protocol. With MAFIA or MCOP, if the edge router is compromised it can let in PIM-Joins without verification. Additionally, since non-border routers perform no verification, traffic injection of bogus PIM-Joins cannot be detected. In the rest of this section we compare the performance of MCOP (both MCOP and MAFIA rely on access control and can be expected to



(a) Simulated network topology.



(b) Percentage of completed joins.

Figure 8: Effectiveness on a simulated network topology

perform similarly) with the modified PIM-Join protocol.

Under normal operating circumstances, assuming that the network latencies dominate the overall validation time, a Join message in MCOP is required to wait at the border router while the request is validated. This wait duration consists of the network delay between the border router and the centralized database and the processing time to loop up the request. Assuming an average of two hops from all the border routers, MCOP joins can incur an additional delay in the order of 2-5 msec (as opposed to microseconds in our solution) to validate the user's request. The experiments performed by Miikka Tammi [16] corroborate these numbers.

Under attack scenarios, especially in the case of inter-domain attacks, MCOP and MAFIA provide little protection. This is because these solutions are catered towards intra-domain scale multicast groups and hence cannot be easily extended to inter-domain scale. MCOP or MAFIA in an inter-domain scale would require a centralized database that keeps track of all source-group pairs in the Internet which is impractical. A decentralized database brings with it all the problems related to distributed synchronization between databases.

To compare the performance of MCOP with our protocol, we performed a series of experiments in NS2 [4] (ns-2.26) using a similar network topology as before (Figure 8(a)). However we group the routers into four domains so that each domain consists of 25% of all the routers and MCOP is simulated in nodes in all domains. Routers in the source-end domain thus reject all join requests where the source is unknown.

Figure 9 shows the results of our simulations. As can be seen from the simulations, MCOP offers little protection in the case of inter-domain attacks. This is because nodes in the domains other than the source have no information about the validity of the source and are hence forced to process all requests, bogus or otherwise. The legitimate user who shares a domain with the attacker faces heavy losses in successful join requests because they are dropped in routers at the other domains in the path to the source domain.

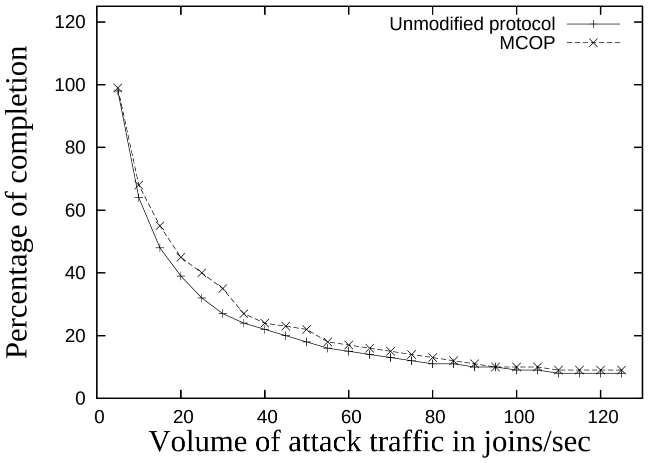


Figure 9: Percentage of completed joins using MCOP

Our evaluations demonstrate that our modified PIM protocol is highly effective in preventing state overload attacks. In addition, the processing overhead required in routers is higher, but it does not cause a noticeable performance degradation for the end user.

This approach thus provides a comprehensive solution to the problem with minimal performance impact for the user. But, the modifications required for its deployment would require a transition period during which non-deploying domains will be vulnerable to attack. Next, we discuss our second solution which aims to rectify this transition problem and provide a more easily deployable solution. The approach we have adapted is to use an overlay network to ensure that it requires minimal changes for its deployment.

5 Solution 2: Overlay Based Indirection

Our second solution proposes an overlay based architecture aimed at providing a practical solution to state overload attacks. The central idea behind this solution is to ensure that the DR(R) propagates join requests only if the source and the group being requested in the join message are known to be valid. This prevents the possibility of state overload attacks with bogus join messages. Unlike the first solution, this method is a stop-gap rather than a complete fix; its utility stems from the fact that it is easily deployable and requires no modifications to the PIM protocol or PIM routers for their functioning.

5.1 Architecture

The architecture required is shown in Fig. 10. It consists of three components in each multicast domain: 1) Overlay nodes, 2) Verification boxes (VB), and 3) Indirection boxes (IB). One or more overlay nodes are deployed per domain and are configured statically or dynamically with a database of valid (S,G) pairs within their domain. The database can optionally include added information like valid receivers for the group, the number of groups each source is allowed to support, the number of receivers per domain/subnet. This added information can be used to provide intelligent join request filtering and access control if required for group sources. Associated with an overlay node is a domain name which can be used by other nodes to resolve its IP address. Domain names can be created such that a source address S can be translated to a domain name using a simple bootstrap or mapping algorithm. To maintain connectivity of the overlay network, overlay nodes in neighboring domains establish neighborhood relationships with each other. A routing protocol is also established on top of the overlay network to provide packet forwarding between nodes.

Verification boxes (VB) are deployed by the ISPs and network administrators at the edges of their domains. A VB is responsible for monitoring and filtering all incoming and outgoing PIM control messages from its domain. To perform inbound filtering, VBs are configured to maintain information about the valid (S,G) pairs in their domains. Inbound control messages to invalid groups in its local domain are filtered by the VB at the edges before entering the domain. Incoming transit and outgoing messages are also verified to ensure that the final destination is valid. To verify outgoing messages and transit messages, every VB is configured to join an intra-domain source specific channel (P, D) to the overlay node (not shown in the figure) in its domain. The overlay node periodically multicasts information about valid (S,G) groups in other domains it has learned. Based on the available information, the VB filters all outgoing PIM control messages to ensure that the remote destination of the message is known to be valid. Validity information is cached as long as data flows on the (S,G) channel. If there is no flow of data, the information expires unless it is renewed by the overlay node in the source domain.

Indirection Boxes (IB) are co-located with the designated router (DR) in IP multicast domains. They use IGMP snooping [10] to detect and redirect IGMP traffic from hosts in their domains to themselves. All IGMP messages are buffered in the IBs until the intended recipients are verified as valid. They establish mutually authenticated sessions with their local overlay node for this purpose. Once the IGMP message is verified as legitimate, the protocol operation is allowed to continue as normal.

5.2 Operation

We will now describe the operation of the above mentioned components of our architecture and how they effectively prevent state overload attacks. When a receiver R (see Figure 10) issues an IGMP INCLUDE to its DR(R) to join a channel (S,G), the IB in its subnet I(R), detects and redirects the IGMP message to itself. I(R) creates a verification request with the required state information and sends it to its closest overlay node P(R). If P(R) cannot verify the request by itself, it looks up the overlay node P(S) of the (S,G) group requested and forwards a REQUEST message to the remote node. The REQUEST is forwarded along the overlay network to the remote destination node.

P(S) upon receiving the REQUEST message consults its locally maintained database to verify if the group being requested is valid. Once it is verified if the group is legitimate or not (and other verifications like permissions of the receiver if required are completed), P(S) creates a REPLY message with the state information in the REQUEST message and the status of the verification request to return to P(R). As the REPLY message travels downstream through the overlay network to P(R), overlay nodes in the path cache the information provided in the REPLY message.

Every overlay node in the path, upon receiving a REPLY message, multicasts this information to all VBs on the source-specific group (P,D). If the request is valid, P(R) on receiving the REPLY message, directs the requesting IB, I(R) to continue with the join forwarding. I(R) retrieves and sends the previously cached IGMP INCLUDE message to DR(R) which generates the join message as usual. The join as it propagates is verified by each VB in the upstream path to the source. When the join reaches DR(S) the join procedure is complete.

5.3 Security and DoS Resistance of the Architecture

In this section we consider the security and DoS resistance of the overlay architecture and its individual components. These are important considerations to ensure that the introduction of the new architecture does not create any new security flaws which can be exploited by an attacker to launch new attacks on the architecture and hence the service it seeks to protect.

Indirection boxes (IB) are limited in their traffic scope to users from their subnet and the local overlay node. This makes them easy to protect against DoS attacks

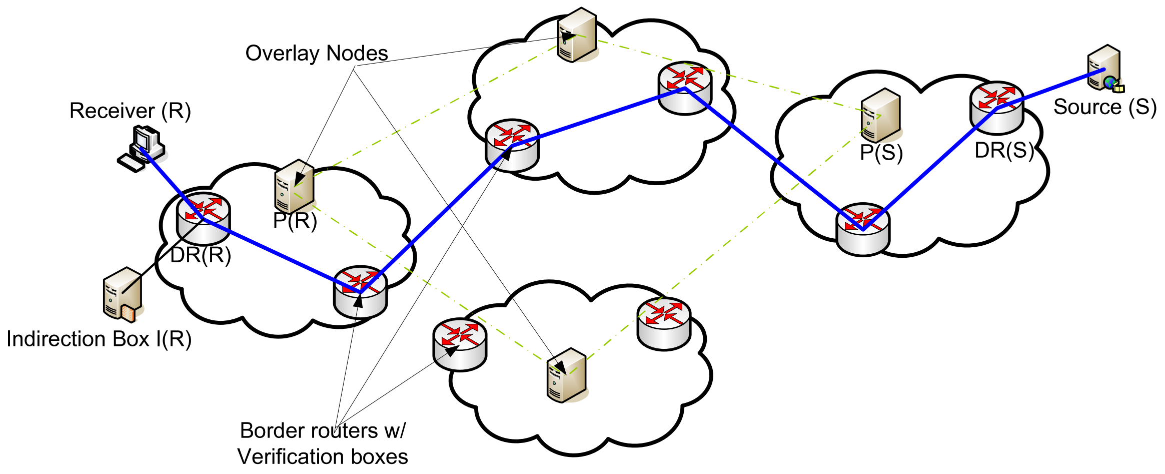


Figure 10: Architecture required for overlay based solution

from outside the domain. Furthermore, even a successful attack on the IB has a limited impact and affects only a small set of users. The IB uses IPSec in its transactions with its local overlay node. This ensures that an attacker cannot easily masquerade as an IB to generate bogus verification requests to the overlay node or generate join requests which have not been verified as valid. Even if the attacker is able to generate bogus join requests, they will not be forwarded by the verification boxes.

Verification boxes (VB) are open only to multicast traffic; so all unicast traffic to it can be filtered out before it reaches the VB at the edges of domains using router based access control lists (ACLs). This makes unicast based DoS attacks implausible. Since the VBs are required to maintain state about valid (S,G) pairs in their domains and remote domains which have been approved, an attacker might attempt to launch a state overload attack against them. Such an attack however is highly unlikely to succeed. As opposed to the router based state overload attacks, only valid groups will create state in the VBs. Additionally, VBs can be equipped with high capacity storage devices which are practically impossible to overload with state alone (see Section 4.4).

Overlay nodes are limited in their scope to traffic from their neighboring overlay nodes and IBs and VBs from within its domain. An attacker can attempt to launch unicast based DoS attacks with the aim of either disrupting the connectivity to the overlay node or exhausting its computing resources and/or buffer space and memory. We expect that overlay nodes will be deployed by ISPs at the core of the domain. So, connectivity to the overlay nodes will be through high bandwidth links which cannot be easily disabled by intra-domain DoS attacks. Additionally, filtering of unwanted traffic at the edges of the domains is trivial because the legitimate traffic from outside the domain is expected only from neighboring overlay nodes. Unicast based DoS attacks on the overlay nodes are thus implausible.

In another type of attack, an attacker in a remote domain can force her local overlay to continuously issue verification requests to an overlay node in another domain. If the number of these verification requests is large and sufficiently distributed, the remote overlay could be overwhelmed. A number of measures can be adapted to protect against these attacks. The querying overlay can enforce rate limits on the number of verification requests on a per-host or per-domain basis. Additionally, by caching and pre-fetching validity information, the number of verification requests that need to be sent can be minimized. Other measures to prevent these attacks include resource redundancy in the number of overlay nodes per domain and allowing any overlay node to reply to a verification request if it has the relevant information requested.

5.4 Partial Deployment

Partial deployment is an important concern in any new architectural proposal. In our solution, if the source domain is non-deploying, verification of the validity of the (S,G) pair requires some modifications to the architecture. In this case, to verify the validity of the (S,G), the overlay node in the neighboring domain to the source can issue a join request to the group. If the source and group are valid and regularly transmitting, the node will start receiving multicast data. This can be used as a validity check for the presence of the group. If the source and receiver domains have both deployed the architecture, the solution will provide complete protection to the deploying domains irrespective of the status of other domains. To maintain connectivity, traffic between overlay nodes can be tunnelled across non-deploying domains.

Finally, if a transit join request arrives at the edge of a deployed domain, and the (S,G) is not known to be valid, the ingress VB cannot always drop this request in the presence of partial deployment. If the request arrives from a non-deploying domain, the validity of the join message needs to be verified before creating state in the transit domain. For this purpose, the join request can be tunnelled to the exit VB of the deployed domain (without creating state) which forwards the join upstream as usual. If the (S,G) is valid and regularly transmitting, the VB will start receiving multicast data. At this stage ingress VB can be directed to forward the join request as usual. The join request will establish states within the deployed domain and operation can continue as normal. This solution is similar to partial deployment modifications for the first solution described in more detail in Section 4.4.

5.5 Evaluations

In this section we evaluate the overhead introduced due to the overlay based approach and the effectiveness of the protocol in defending against state overload attacks.

Latency introduced during join procedure: To evaluate the added latency introduced for the verification process prior to the join, we model the operation of the IB and overlay nodes as applications in NS2 [4] (ns-2.26). From an end-user's point of view, the metric of interest is the perceived latency as it indicates the overall performance impact of the solution. The perceived latency includes additional components like queuing and propagation delay at routers and is measured as the total time from the user issuing a PIM-Join request until it starts receiving multicast data from the group. Here, we assume that as soon as the PIM-Join request arrives at the source site, the source starts sending multicast data. For the overlay based method, in addition to the above, the latency also includes the time for the source to send the verification request to its local overlay node and get a verified reply from the remote overlay node through its local overlay node. In our evaluations we assume that the time required to perform database lookups is negligible compared to the overall end-to-end latency.

The topology used for our evaluations is a 10 AS hierarchial Waxman with 50 nodes in each AS. One of the nodes in each AS is randomly selected to act as an overlay node. To measure the latency introduced, a source node is configured to send a request message to its local overlay node. Upon receiving the request message, the local overlay node sends the request to its neighboring overlay node and so on upstream until the request reaches the overlay node of the receiver. The message is then sent downstream along the same path until it reaches the source overlay node and back to the original source node. At this point the source generates a join request and sends it upstream to the receiver. For each run of the simulation, more hops are added between the source and destination nodes and the overall latency perceived by the user is measured in each case. In our topology, the average hop length between neighboring overlay nodes is approximately two. Additionally, the source and destination nodes are chosen to be one hop away from their respective overlay nodes.

The results of our simulation are displayed in Figure 11(a). As can be seen from the results, in the overlay based approach, the verification process causes an increase in latency by a factor of around 2 for the join procedure. Note that this added latency is a one-time overhead during the initial join procedure only. Once the join is successfully completed the added latency overhead due to the solution is minimal.

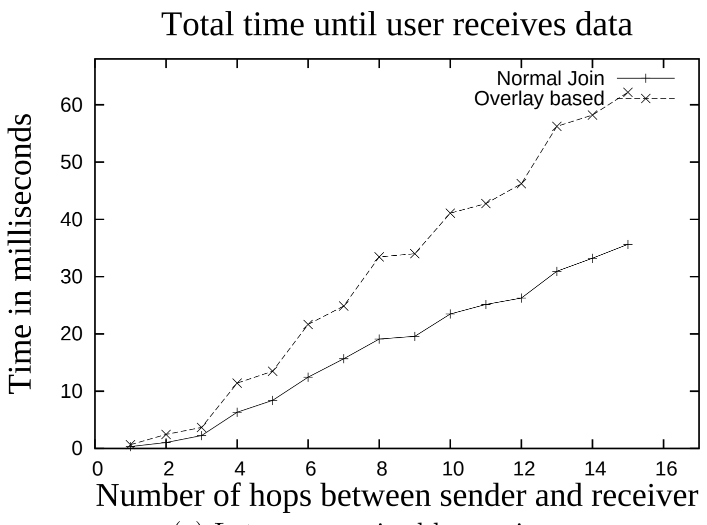
Percentage of completed joins under attack: To measure the DoS resistance of the proposed approach to state overload attacks, we measure the percentage of legitimate joins completed by a receiver operating with and without the added protection. The routers in the receiver's join path to the source are subject to state overload attacks of varying magnitude. We use the same topology (Figure 8(a)) and metrics as in the experiments for the first solution. The results of our experiments are displayed in Figure 11(b). As can be seen in the normal PIM protocol, the percentage of completed joins decreases exponentially as the rate of attack increases because the legitimate and the attack traffic compete for the same limited buffer space. The overlay based method meanwhile maintains a 100 % completion rate because only legitimate joins create state.

Our evaluations demonstrate that the overlay based approach is highly effective in preventing state overload attacks. The overlay based solution is easily deployable and requires no changes in the network for its deployment. However, it has some drawbacks that make it a temporary fix rather than a complete solution. One of the drawbacks of this solution is that the overlay architecture can be subject to DoS attacks using the verification request process. We described these attacks and some measures to prevent such attacks earlier in Section 5.3. Additionally, unlike the first solution, the architecture masks the vulnerability rather than fixing it completely.

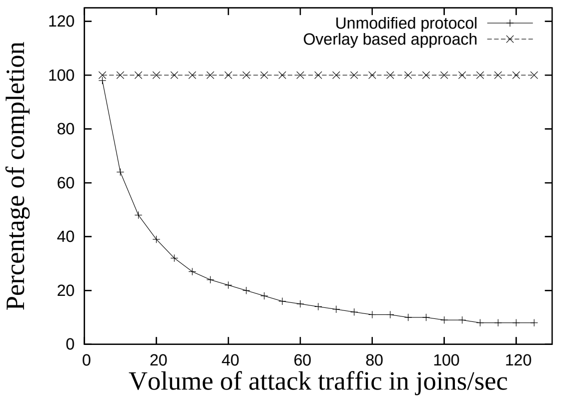
6 Conclusion

DoS attacks pose a serious problem to the health and security of value-added services in the Internet. In this paper, we have examined DoS attacks, called state overload attacks, for a specific service, IP multicast. Since the attacks exploit an inherent weakness in the PIM protocol, we proposed two solutions to make it more secure against these attacks. The modifications provide an effective solution against DoS attacks while creating minimal performance loss or latency for the end user. Our solutions can be incrementally deployed in the inter-domain with some modifications and can provide an equally effective defense in the partial deployment case as compared to the full deployment case.

Our study of state overload attacks in multicast demonstrate the importance of careful protocol design and the intrinsic difficulty in designing secure protocols. In particular a high level lesson we have taken away is the difficulty of introducing value-added services in the Internet without introducing added overhead and new security



(a) Latency perceived by receiver.



(b) Percentage of completed joins.

Figure 11: Performance of overlay based solution

vulnerabilities. In our first solution, we have provided a blue-print for building secure protocols for value-added services. In particular we demonstrate that the overhead introduced for the operation of value-added protocols can be effectively controlled without taking away from the protocols functionality and without creating new vulnerabilities that can be exploited to bring down the service.

References

[1] K. Almeroth, "The evolution of multicast: From the MBone to inter-domain multicast to Internet2 deployment", *IEEE Network*, vol. 14, no. 1, pp. 10–20, Jan. 2000.

[2] C. Diot, B. Levine, B. Lyles, H. Kassem, and D. Balensiefen, "Deployment issues for the IP multicast service and architecture", *IEEE Network*, vol. 14, no. 1, pp. 78–88, Jan. 2000.

[3] D. Estrin et al., *Protocol Independent Multicast Sparse-Mode (PIM-SM): Protocol Specification*, Internet Engineering Task Force (IETF), RFC 2362, June 1998.

[4] K. Fall and K. Varadhan, *Ns-2 Notes and Documentation.* (http://www.isi.edu/nsnam/ns)

[5] B. Gronval, *Scalable Multicast Forwarding.* (http://www.acm.org/sigcomm/ccr/archive/2002/jan02/CCR-SC01-Posters/BjornGronvall.ps)

[6] M. Handley and A. Greenhalgh, "Steps towards dos resistant internet architecture", *Proceedings of ACM SIGCOMM Workshop on FDNA*, Portland, OR, USA, pp. 49–56, Aug. 2004.

[7] H. Krawczyk, M. Bellare, and R. Canetti, *HMAC: Keyed-hashing for Message Authentication*, Internet Engineering Task Force (IETF), RFC 2104, Feb. 1997.

[8] B. Krishnamurthy and C. Wills, "On the use and performance of content distribution networks", *Proceedings of SIGCOMM Internet Measurement Workshop*, San Fransisco, USA, pp. 169–282, Nov. 2001.

[9] R. Lehtonen, J. Soini, J. Majalainen, and H. Vatiainen, *MCOP Operation for First Hop Routers*, Internet Engineering Task Force (IETF) draft, work in progress, June 2004.

[10] K. McCloghrie, D. Farinacci, and D. Thaler, *Internet Group Management Protocol MIB*, Internet Engineering Task Force (IETF), RFC 2933, Oct. 2000.

[11] K. Ramachandran and K. Almeroth, "MAFIA: A multicast management solution for access control and packet filtering", *Proceedings of MMNS*, Belfast, Ireland, pp. 64–77, Sep. 2003.

[12] S. Ratnasamy, A. Ermolinskiy, and S. Shenker, "Revisiting IP multicast", *Proceedings of ACM SIGCOMM*, Pisa, Italy, pp. 25–26, Sep. 2006.

[13] S. Savage, D. Wetherall, A. Karlin, and T. Anderson, "Practical network support for IP traceback", *Proceedings of ACM SIGCOMM*, Stockholm, Sweden, pp. 295–306, Aug. 2000.

[14] P. Savola, R. Lethonen, and D. Meyer, *PIM-SM Multicast Routing Security Issues and Enhancements*, Internet Engineering Task Force (IETF), RFC 4609, Aug. 2006.

[15] S. Shenker and J. Wroclawski, *General Characterization Parameters for Integrated Service Network*, Internet Engineering Task Force (IETF), RFC 2215, Sep. 1997.

[16] M. Tammi, *Implementation and Performance Evaluation of Multicast Control Protocol*, Master's thesis, Tampere University of Technology, 33720 Tampere, Finland, May 2004.

[17] Y. Xiao, X. Du, J. Zhang, F. Hu, and S. Guizani, "Internet protocol television (IPTV): The killer application for the next-generation internet", *IEEE Communications Magazine*, vol. 45, no. 11, pp. 126–134, Nov. 2007.

Jinu Kurian is a 3rd year Phd student in the Department of Computer Science, University of Texas at Dallas. He received his M.S in Telecommunication Engineering from the Department of Telecommunications Engineering, University of Texas at Dallas in 2006. His research interests are in network security, denial of service defense, service overlay networks and secure authentication mechanisms.

Kamil Sarac is an Assistant Professor of computer science at the University of Texas at Dallas. He obtained his Ph.D. degree in computer science from the University of California Santa Barbara in 2002. His research interests include network and service monitoring and Internet measurements; multicast communication; overlay networks; and energy efficiency in ad hoc networks. He is a member of ACM and IEEE.