

## IP Multicast and Firewalls

### Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

### Copyright Notice

Copyright (C) The Internet Society (1999). All Rights Reserved.

### 1. Abstract

Many organizations use a firewall computer that acts as a security gateway between the public Internet and their private, internal 'intranet'. In this document, we discuss the issues surrounding the traversal of IP multicast traffic across a firewall, and describe possible ways in which a firewall can implement and control this traversal. We also explain why some firewall mechanisms - such as SOCKS - that were designed specifically for unicast traffic, are less appropriate for multicast.

### 2. Introduction

A firewall is a security gateway that controls access between a private administrative domain (an 'intranet') and the public Internet. This document discusses how a firewall handles IP multicast [1] traffic.

We assume that the external side of the firewall (on the Internet) has access to IP multicast - i.e., is on the public "Multicast Internet" (aka. "MBone"), or perhaps some other multicast network.

We also assume that the \*internal\* network (i.e., intranet) supports IP multicast routing. This is practical, because intranets tend to be centrally administered. (Also, many corporate intranets already use multicast internally - for training, meetings, or corporate announcements.) In contrast, some previously proposed firewall mechanisms for multicast (e.g., [2]) have worked by sending \*unicast\* packets within the intranet. Such mechanisms are usually inappropriate, because they scale poorly and can cause excessive network traffic within the intranet. Instead, it is better to rely

upon the existing IP multicast routing/delivery mechanism, rather than trying to replace it with unicast.

This document addresses scenarios where a multicast session is carried - via multicast - on both sides of the firewall. For instance, (i) a particular public Mbone session may be relayed onto the intranet (e.g., for the benefit of employees), or (ii) a special internal communication (e.g., announcing a new product) may be relayed onto the public Mbone. In contrast, we do not address the case of a roaming user - outside the firewall - who wishes to access a private internal multicast session, using a virtual private network. (Such "road warrior" scenarios are outside the scope of this document.)

As noted by Freed and Carosso [3], a firewall can act in two different ways:

- 1/ As a "protocol end point". In this case, no internal node (other than the firewall) is directly accessible from the external Internet, and no external node (other than the firewall) is directly accessible from within the intranet. Such firewalls are also known as "application-level gateways".
- 2/ As a "packet filter". In this case, internal and external nodes are visible to each other at the IP level, but the firewall filters out (i.e., blocks passage of) certain packets, based on their header or contents.

In the remainder of this document, we assume the first type of firewall, as it is the most restrictive, and generally provides the most security. For multicast, this means that:

- (i) A multicast packet that's sent over the Internet will never be seen on the intranet (and vice versa), unless such packets are explicitly relayed by the firewall, and
- (ii) The IP source address of a relayed multicast packet will be that of the firewall, not that of the packet's original sender. To work correctly, the applications and protocols being used must take this into account. (Fortunately, most modern multicast-based protocols - for instance, RTP [4] - are designed with such relaying in mind.)

### 3. Why Multicast is Different

When considering the security implications of IP multicast, it is important to note the fundamental way in which multicast communication differs from unicast.

Unicast communication consists of a 'conversation' between an explicit pair of participants. It therefore makes sense for the security of unicast communication to be based upon these participants (e.g., by authenticating each participant). Furthermore, 'trust' within unicast communication can be based upon trust in each participant, as well as upon trust in the data.

Multicast communication, on the other hand, involves a arbitrary sized, potentially varying set of participants, whose membership might never be fully known. (This is a feature, not a bug!) Because of this, the security of multicast communication is based not upon its participants, but instead, upon its \*data\*. In particular, multicast communication is authenticated by authenticating packet data - e.g., using digital signatures - and privacy is obtained by encrypting this data. And 'trust' within multicast communication is based solely upon trust in the data.

#### 4. Multicast-Related Threats and Countermeasures

The primary threat arising from relaying multicast across a firewall is therefore "bad data" - in particular:

- (i) damaging data flowing from the Internet onto the intranet, or
- (ii) sensitive data inadvertently flowing from the intranet onto the external Internet.

To avert this threat, the intranet's security administrator must establish, in advance, a security policy that decides:

- (i) Which multicast groups (and corresponding UDP ports) contain data that can safely be relayed from the Internet onto the intranet. For example, the security administrator might choose to permit the relaying of an MBone lecture, knowing that the data consists only of audio/video (& to safe ports).
- (ii) Which multicast groups (and corresponding UDP ports) will not contain sensitive internal information (that should therefore not be relayed from the intranet onto the Internet). This, of course, requires placing trust in the applications that internal users will use to participate in these groups. For example, if users use an audio/video 'viewer' program to participate in an MBone session, then this program must be trusted not to be a "Trojan Horse". (This requirement for "trusted applications" is by no means specific to multicast, of course.)

Once such a security policy has been established, it is then the job of the firewall to implement this policy.

## 5. What Firewalls Need to Do

In short, a firewall must do three things in order to handle multicast:

- 1/ Support the chosen multicast security policy (which establishes particular multicast groups as being candidates to be relayed),
- 2/ Determine (dynamically) when each candidate group should be relayed, and
- 3/ Relay each candidate group's data across the firewall (and then re-multicast it at the far end).

These three tasks are described in more detail in the next three sections.

Note that because a firewall is often a convenient place to centralize the administration of the intranet, some firewalls might also perform additional administrative functions - for example, auditing, accounting, and resource monitoring. These additional functions, however, are outside the scope of this document, because they are not specifically \*firewall\*-related. They are equally applicable to an administrative domain that is not firewalled.

## 6. Supporting a Multicast Security Policy

As noted above, a multicast security policy consists of specifying the set of allowed multicast groups (& corresponding UDP ports) that are candidates to be relayed across the firewall. There are three basic ways in which a firewall can support such a policy:

- 1/ Static configuration. The firewall could be configured, in advance, with the set of candidate groups/ports - for example, in a configuration file.
- 2/ Explicit dynamic configuration. The set of candidate groups/ports could be set (and updated) dynamically, based upon an explicit request from one or more trusted clients (presumably internal). For example, the firewall could contain a 'remote control' mechanism that allows these trusted clients - upon authentication - to update the set of candidate groups/ports.
- 3/ Implicit dynamic configuration. The set of candidate groups/ports could be determined implicitly, based upon the contents of some pre-authorized multicast group/port, such as a "session directory". Suppose, for example, that the security policy decides that the default Mbone SAP/SDP session directory [5] may be relayed, as well as any sessions that are announced in this directory. A 'watcher' process, associated with the firewall, would watch this directory, and use its contents to

dynamically update the set of candidates.

Notes:

- (i) Certain ranges of multicast addresses are defined to be "administratively scoped" [6]. Even though the firewall does not act as a true multicast router, the multicast security policy should set up and respect administrative scope boundaries.
- (ii) As noted in [2], certain privileged UDP ports may be considered dangerous, even with multicast. The multicast security policy should check that such ports do not become candidates for relaying.
- (iii) Even if sessions announced in a session directory are considered automatic candidates for relaying (i.e., case 3/above), the firewall's 'watcher' process should still perform some checks on incoming announcements. In particular, it should ensure that each session's 'group' address really is a multicast address, and (as noted above) it should also check that the port number is within a safe range. Depending on the security policy, it may also wish to prevent any \*locally\* created session announcements from becoming candidates (or being relayed).

## 7. Determining When to Relay Candidate Groups

If a multicast group becomes a candidate to be relayed across the firewall, the actual relaying should *\*not\** be done continually, but instead should be done only when there is actual interest in having this group relayed. The reason for this is two-fold. First, relaying a multicast group requires that one or both sides of the firewall join the group; this establishes multicast routing state within the network. This is inefficient if there is no current interest in having the group relayed (especially for Internet->intranet relaying). Second, the act of relaying an unwanted multicast group consumes unnecessary resources in the firewall itself.

The best way for the firewall to determine when a candidate group should be relayed is for it to use actual multicast routing information, thereby acting much as if it were a real 'inter-domain' multicast router. If the intranet consists of a single subnet only, then the firewall could listen to IGMP requests to learn when a candidate group has been joined by a node on this subnet. If, however, the intranet consists of more than one subnet, then the firewall can learn about candidate group memberships by listening to "Domain Wide Multicast Group Membership Reports" [7]. Unfortunately, this mechanism has only recently been defined, and is not yet used by

most routers.

Another, albeit less desirable, way for the firewall to learn when candidate multicast groups have been joined is for the firewall to periodically 'probe' each of these groups. Such a probe can be performed by sending an ICMP ECHO request packet to the group, and listening for a response (with some timeout interval). This probing scheme is practical provided that the set of candidate groups is reasonably small, but it should be used only on the intranet, not on the external Internet. One significant drawback of this approach is that some operating systems - most notably Windows 95 - do not respond to multicast ICMP ECHOs. However, this approach has been shown to work on a large, all-Unix network.

Another possibility - less desirable still - is for each node to explicitly notify the firewall whenever it joins, or leaves, a multicast group. This requires changes to the node's operating system or libraries, or cooperation from the application. Therefore this technique, like the previous one, is applicable only within the intranet, not the external Internet. Note that if multicast applications are always launched from a special "session directory" or "channel guide" application, then this application may be the only one that need be aware of having to contact the firewall.

What makes the latter two approaches ("probing" and "explicit notification") undesirable is that they duplicate some of the existing functionality of multicast routing, and in a way that scales poorly for large networks. Therefore, if possible, firewalls should attempt to make use of existing multicast routing information: either IGMP (for a single-subnet intranet), or "Domain Wide Multicast Group Membership Reports".

In some circumstances, however, the client cannot avoid contacting the firewall prior to joining a multicast session. In this case, it may make sense for this contact to also act as a 'notification' operation. Consider, for example, an RTSP [8] proxy associated with the firewall. When the proxy receives a request - from an internal user - to open a remote RTSP session, the proxy might examine the response from the remote site, to check whether a multicast session is being launched, and if so, check whether the multicast group(s) are candidates to be relayed.

## 8. Relaying Candidate Groups

The actual mechanism that's used to relay multicast packets will depend upon the nature of the firewall. One common firewall configuration is to use two nodes: one part of the intranet; the other part of the external Internet. In this case, multicast packets would be relayed between these two nodes (and then re-multicast on the other side) using a tunneling protocol.

A tunneling protocol for multicast should *\*not\** run on top of TCP, because the reliability and ordering guarantees that TCP provides are unnecessary for multicast communication (where any reliability is provided at a higher level), yet would add latency. Instead, a UDP-based tunneling protocol is a better fit for relaying multicast packets. (If congestion avoidance is a concern, then the tunnel traffic could be rate-limited, perhaps on a per-group basis.)

One possible tunneling protocol is the "UDP Multicast Tunneling Protocol" (UMTP) [9]. Although this protocol was originally designed as a mechanism for connecting individual client machines to the MBone, it is also a natural fit for use across firewalls. UMTP uses only a single UDP port, in each direction, for its tunneling, so an existing firewall can easily be configured to support multicast relaying, by adding a UMTP implementation at each end, and enabling the UDP port for tunneling.

### Notes:

- (i) When multicast packets are relayed from the intranet onto the external Internet, they should be given the same TTL that they had when they arrived on the firewall's internal interface (except decremented by 1). Therefore, the internal end of the multicast relay mechanism needs to be able to read the TTL of incoming packets. (This may require special privileges.) In contrast, the TTL of packets being relayed in the other direction - from the external Internet onto the intranet - is usually less important; some default value (sufficient to reach the whole intranet) will usually suffice. Thus, the Internet end of the multicast relay mechanism - which might be less trusted than the intranet end - need not run with special privileges.
- (ii) One end of the multicast tunnel - usually the intranet end - will typically act as the controller (i.e., "master") of the tunnel, with the other end - usually the Internet end - acting as a "slave". For security, the "master" end of the tunnel should be configured not to accept any commands from the "slave" (which will often be less trusted).

## 9. Networks With More Than One Firewall

So far we have assumed that there is only one firewall between the intranet and the external Internet. If, however, the intranet has more than one firewall, then it's important that no single multicast group be relayed by more than one firewall. Otherwise (because firewalls are assumed to be application-level gateways - not proper multicast routers), packets sent to any such group would become replicated on the other side of the firewalls. The set of candidate groups must therefore be partitioned among the firewalls (so that exactly one firewall has responsibility for relaying each candidate group). Clearly, this will require coordination between the administrators of the respective firewalls.

As a general rule, candidate groups should be assigned - if possible - to the firewall that is topologically closest to most of the group members (on both the intranet and the external Internet). For example, if a company's intranet spans the Atlantic, with firewalls in New York and London, then groups with mostly North American members should be assigned to the New York firewall, and groups with mostly European members should be assigned to the London firewall. (Unfortunately, even if a group has many internal and external members on both sides of the Atlantic, only one firewall will be allowed to relay it. Some inefficiencies in the data delivery tree are unavoidable in this case.)

## 10. Why SOCKS is Less Appropriate for Multicast

SOCKS [10] is a mechanism for transparently performing unicast communication across a firewall. A special client library - simulating the regular 'sockets' library - sits between applications and the transport level. A conversation between a pair of nodes is implemented (transparently) as a pair of conversations: one between the first node and a firewall; the other between the firewall and the second node.

In contrast, because multicast communication does not involve a conversation between a pair of nodes, the SOCKS model is less appropriate. Although multicast communication across a firewall is implemented as two separate multicast communications (one inside the firewall; the other outside), the *same* multicast address(es) and port(s) are used on both sides of the firewall. Thus, multicast applications running inside the firewall see the same environment as those running outside, so there is no need for them to use a special library.

Nonetheless, there has been a proposal [11] to extend SOCKS V5 to support multicast. This proposal includes two possible modes of communication:

- (i) "MU-mode", uses only \*unicast\* communication within the intranet (between the firewall and each internal group member), and
- (ii) "MM-mode", which uses unicast for client-to-firewall relay control, but uses \*multicast\* for other communication within the intranet.

As noted in section 2 above, "MU-mode" would be a poor choice (unless, for some reason, the intranet does not support multicast routing at all). If multicast routing is available, there should rarely be a compelling reason to replace multicast with 'multiple-unicast'. Not only does this scale badly, but it also requires (otherwise unnecessary) changes to each application node, because the multicast service model is different from that of unicast.

On the other hand, "MM-mode" (or some variant thereof) \*might\* be useful in environments where a firewall can learn about group membership only via "explicit notification". In this case each node might use SOCKS to notify the firewall whenever it joins and leaves a group. However, as we explained above, this should only be considered as a last resort - a far better solution is to leverage off the existing multicast routing mechanism.

It has been suggested [11] that a benefit of using multicast SOCKS (or an "explicit notification" scheme in general) is that it allows the firewall to authenticate a client's multicast "join" and "leave" operations. This, however, does not provide any security, because it does not prevent other clients within the intranet from joining the multicast session (and receiving packets), nor from sending packets to the multicast session. As we noted in section 3 above, authentication and privacy in multicast sessions is usually obtained by signing and encrypting the multicast data, not by attempting to impose low-level restrictions on group membership. We note also that even if group membership inside the intranet could be restricted, it would not be possible, in general, to impose any such membership restrictions on the external Internet.

## 11. Security Considerations

Once a security policy has been established, the techniques described in this document can be used to implement this policy. No security mechanism, however, can overcome a badly designed security policy. Specifically, network administrators must be confident that the multicast groups/ports that they designate as being 'safe' really are

free from harmful data. In particular, administrators must be familiar with the applications that will receive and process multicast data, and (as with unicast applications) be confident that they cannot cause harm (e.g., by executing unsafe code received over the network).

Because it is possible for an adversary to initiate a "denial of service" attack by flooding an otherwise-legitimate multicast group with garbage, administrators may also wish to guard against this by placing bandwidth limits on cross-firewall relaying.

## 12. Summary

Bringing IP multicast across a firewall requires that the intranet first establish a multicast security policy that defines which multicast groups (& corresponding UDP ports) are candidates to be relayed across the firewall. The firewall implements this policy by dynamically determining when each candidate group/port needs to be relayed, and then by doing the actual relaying. This document has outlined how a firewall can perform these tasks.

## 13. References

- [1] Deering, S., "Host Extensions for IP Multicasting", STD 5, RFC 1112, August 1989.
- [2] Djahandari, K., Sterne, D. F., "An MBone Proxy for an Application Gateway Firewall" IEEE Symposium on Security and Privacy, 1997.
- [3] Freed, N. and K. Carosso, "An Internet Firewall Transparency Requirement", Work in Progress.
- [4] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.
- [5] Handley, M. and V. Jacobson, "SDP: Session Description Protocol", RFC 2327, April 1998.
- [6] Meyer, D., "Administratively Scoped IP Multicast", BCP 23, RFC 2365 July 1998.
- [7] Fenner, B., "Domain Wide Multicast Group Membership Reports", Work in Progress.
- [8] Schulzrinne, H., Rao, A. and R. Lanphier, "Real Time Streaming Protocol (RTSP)", RFC 2326, April 1998.

- [9] Finlayson, R., "The UDP Multicast Tunneling Protocol", Work in Progress.
- [10] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D. and L. Joned, "SOCKS Protocol Version 5", RFC 1928, April 1996.
- [11] Chouinard, D., "SOCKS V5 UDP and Multicast Extensions", Work in Progress.

#### 14. Author's Address

Ross Finlayson,  
Live Networks, Inc. (LIVE.COM)

EMail: [finlayson@live.com](mailto:finlayson@live.com)  
WWW: <http://www.live.com/>

## 15. Full Copyright Statement

Copyright (C) The Internet Society (1999). All Rights Reserved.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet Standards process must be followed, or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

## Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.

