A Dynamic Stateful Multicast Firewall

Shen Li*†, Vijay Sivaraman*†, Alex Krumm-Heller†, Craig Russell†
*School of Electrical Engineering and Telecommunications, University of New South Wales, Australia
†ICT Centre, CSIRO, Australia

***Abstract*—Enterprises are faced with the challenge of enabling IP multicast applications without exposing their network to multicast denial-of-service attacks. Current practice is to use firewalls and manually configure them on a per-multicast-session basis. This imposes a high work-load on the network administrator, and severely reduces flexibility for end-users. In this paper, we propose and demonstrate a simple yet powerful multicast firewall algorithm that can, under most conditions, automatically distinguish unsolicited multicast packets and drop them to protect the network from denial-of-service attacks. Inspired by the "stateful" operation of unicast firewalls, our multicast firewall blocks unsolicited multicast packets by maintaining state information on multicast group membership and unicast interactions. We prototype our algorithm as a plug-in to Linux NetFilter, and present performance and scalability results from testing on a high-quality multicast video platform coupled with synthetic traffic from a network tester. Based on the prototype, we believe that it is feasible to build multicast firewalls that can, without manual intervention, and with minimal performance impact, protect the network against multicast attacks.**

I. Introduction

The IP multicast service model, introduced by Deering in 1990 for efficient multipoint communication [1], offers two key benefits: efficient use of bandwidth for multipoint communication, and indirection of a group address which allows for network-level rendezvous and service discovery. In spite of the mixed success that IP multicast has had in actual deployments, we are finally witnessing the emergence of applications that can greatly benefit from IP multicast support: examples include massive multiplayer online games (MMOGs) [2], IP TV [3], and large-scale collaborative audio-visual platforms such as the AccessGrid [4]. Probably the largest hinderance to wide-area deployment of IP multicast by ISPs has been the complexity of managing inter-domain multicast routing, as illustrated by a recent article [5]. The routing complexity is apparent in the list of multicast protocols that a Cisco router advertises: PIM-SM, PIM-DM, BiDir-PIM, PIM-SSM, AutoRP, MBGP, MSDP, and IGMPv1,v2,v3, while still lacking MASC/AAP and BGMP.

This paper considers IP multicast from an enterprise organization's point-of-view, wherein security has been the major obstacle to widespread deployment. Native multicast allows any Internet host to transmit to a multicast group, and this creates the potential for denial-of-service (DoS) attacks in which one or many malicious hosts in the Internet can overwhelm an enterprise network by transmitting unsolicited traffic on a multicast group to which one or more hosts in the enterprise network have subscribed. An example of a multicast DoS attack is the Ramen worm [6] in 2001. Enterprises today protect against multicast DoS attacks by deploying firewall devices (such as Linux NetFilter or Cisco PIX) configured with explicit rules to permit only desired multicast sessions. These rules need to be configured manually by the network administrator on a per-session basis (since multicast group addresses may not be known apriori), imposing a high burden on the personnel and causing frustration to end-users. These factors have led several organisations to expose their multicast environment outside the enterprise firewall, or to entirely abandon efforts to benefit from such applications.

Though the general problem of securing multicast traffic has been addressed before, no satisfactory solution has emerged that protects against multicast DoS attacks. The IETF multicast security (MSEC) working group is developing mechanisms for encrypting and authenticating multicast messages, but do not address DoS attacks. The IETF source specific multicast (SSM) working group is developing efficient routing layer solutions for single-source multicast groups; explicit knowledge of the source allows routers to drop attack packets from malicious sources. However, SSM requires enhancements to the routing infrastructure, and may take many years to be widely deployed. Proprietary protocols by which end-hosts can signal their multicast session requirements to the firewall have been proposed, but require special signalling software and can open up the firewall to potential abuse.

In this paper we develop a simple yet novel firewall algorithm that automatically blocks unsolicited multicast traffic. Drawing inspiration from the "stateful" operation of unicast firewall algorithms, our approach is based on the observation that receivers of multicast traffic usually have unicast interaction with the sources of the multicast traffic. Maintaining state on the unicast interactions, together with multicast group memberships, can help the firewall determine if an incoming multicast packet is solicited or not. We implement our proposal as a simple plug-in to NetFilter on a Linux PC, and test its performance on our high-quality multicast video platform together with synthetic attack traffic. Our firewall is shown to be effective in blocking multicast DoS attacks, with negligible performance penalty in terms of packet latency and firewall CPU load.

The rest of the paper is organised as follows: section II reviews existing mechanisms for multicast security. In section III we describe our stateful firewall solution, and discuss various design trade-offs. Section IV outlines our prototype implementation on Linux, and section V demonstrates performance results from experiments conducted on our test-bed. The paper is concluded in section VI.

II. Background

This section reviews existing proposals for multicast security, and discusses their effectiveness in blocking DoS attacks.

A. IETF MSEC

The IETF Multicast Security (MSEC) working group [7] was chartered to standardize protocols for securing group communication over the Internet. The main focus is on ensuring privacy and authenticity of multicast messages using cryptographic techniques, and an architecture is developed whereby each group has a single trusted entity called the group controller that performs key management. Though protection against multicast DoS attacks is listed as a desirable goal in the MSEC charter, there is no realistic mechanism for this, short of the firewall knowing all the cryptographic keys and performing authentication on all received multicast packets, which would put an excessively high burden on the firewall.

B. IETF SSM

The complexity of choosing a rendezvous point in wide-area multicast routing protocols motivated Holbrook and Cheriton [8] to propose a restricted service model suitable for applications (such as IP TV) that require delivery from a single, often well-known source. This model, now being standardised by the IETF Source Specific Multicast (SSM) working group [9], associates a multicast "channel" with both a group and source IP address, whereby the identity of the source is explicitly exposed so that the routing tree can be rooted at the source. Consequently, routers will drop packets from other (malicious) sources, solving the denial-of-service attack problem at the routing layer. Though router vendors and ISPs are pursuing SSM enthusiastically, it requires upgrades to the routing infrastructure (to support IGMPv3, routing protocol extensions, source-specific forwarding of muticast packets, etc.), and may take several years to be widely adopted.

C. Proprietary Solutions

Proprietary solutions have been proposed by researchers from HP Labs [10]. In the **multicast proxy** approach, a proxy server is located outside the firewall. Clients inside the organisation authenticate with the proxy and issue their request for multicast traffic. The proxy server joins the multicast group on the client's behalf. When the multicast data stream arrives, the proxy converts it into unicast for transmission across the firewall to the requesting client. This approach has the drawback that it requires installation and maintenance of proxy devices. Moreover, the unicasting of the multicast stream to each of the individual receivers does not scale well when there are many internal receivers to a multicast group.

Another solution explored in [10] is the use of a **private protocol** by which the multicast application dynamically notifies the firewall of its group membership (this notification can be extended to include information about the senders on the multicast group). However, this method comes with many caveats: first, the application and firewall would need to implement this private protocol. Second, the firewall can no longer remain a transparent device (it's IP address will have to be revealed by the network administrator to all hosts within the enterprise), and this can open up the firewall to misuse by internal users. Third, in some (albeit rare) cases the firewall may not have an IP address (such as when it is a bridge firewall). These drawbacks make this approach undesirable for practical deployment.

III. A Dynamic Stateful Multicast Firewall

A. The Basic Idea

To understand the proposed solution, consider how unicast sessions transit a firewall without network administrator intervention. A firewall is like a valve, namely it allows all traffic to go out (from the enterprise to the Internet) but is very selective in the traffic it allows in (from the Internet to the enterprise). Thus a unicast session initiated from the outside (Internet-side) of a firewall to a host on the inside (enterprise-side) of the firewall will be dropped at the firewall unless the network administrator explicitly configures a rule to allow it in. However, when a host from the inside requests a session, the firewall forwards it out and notes that a request was sent (i.e. maintains "state"). When the response arrives, the firewall correlates the response with the original request and permits the packet in. The "stateful" nature of firewalls allow unicast traffic to flow in both directions without requiring network administrator involvement, and is based on the principle that if a user within the enterprise has initiated a session with an external party, traffic from that external party's application should be admitted.

So how come the same idea cannot be directly applied to multicast traffic? The first problem is that multicast traffic is addressed to a *group* address rather than to individual end-hosts, so the firewall cannot deduce the external recipients of a multicast stream by inspecting the egress multicast traffic. A second problem is that traffic on a multicast group is often one-way, *i.e.* a receiver inside the enterprise might want to watch a multicast video stream without necessarily generating video itself. The firewall therefore needs more information to help it distinguish solicited from unsolicited traffic incoming multicast traffic. Our idea is based on the premise that many multicast applications have unicast interaction with the sources of the multicast data. As examples, video conferencing tools such as the AcessGrid platform, CSIRO's Virtual Tearoom, Microsoft's conference XP, and Avaya's Collaborative Videoconferencing all have a unicast signalling channel for managing the multicast data. By maintaining state on the unicast traffic interactions as well as multicast memberships, the "stateful" firewall can in most cases distinguish solicited from unsolicited multicast traffic.

We explain this more formally using the notation below:

- $\mathcal{C}$ denotes the set of internal hosts, i.e. "client" IP addresses located within the organisation.
- $\mathcal{G}$ denotes the set of multicast groups to which one or more internal hosts are subscribed.
- $\mathcal{S}$ denotes the set of external (Internet) hosts, that could be potential "sources" transmitting on a multicast group.

Our multicast firewall maintains state regarding the following two relations: first, the $\mathcal{C}$-$\mathcal{G}$ relation, i.e. membership of internal hosts to multicast groups, and second, the $\mathcal{C}$-$\mathcal{S}$ relation, i.e. unicast interactions that internal hosts have had with external senders (indicative of the external hosts that internal hosts "trust"). When a multicast packet addressed to group g from source s arrives at the firewall, it allows the packet to enter only if there exists an internal client $c \in \mathcal{C}$ which has both joined the group g and has had unicast interaction with source s. This is discussed in more detail next as we describe how the various relations are built.

B. Building the Client-Group ($\mathcal{C}$-$\mathcal{G}$) Relations

Building the $\mathcal{C}$-$\mathcal{G}$ relations involves the firewall knowing the set of multicast groups that internal clients are subscribed to. Three possible ways are discussed below.

1) Client is also a Sender on the Group: In an interactive multi-party multicast session, one would expect to be able to determine participation of a client in a multicast group by snooping for packets generated by that client addressed to that multicast group. However, this approach fails not just for one-way multicast applications (such as IP TV), but also for many interactive multicast applications is use today. Unfortunately, inter-domain multicast routing protocol complexity [5] has made it common practice on platforms such as the AccessGrid to have a single sender in each multicast group, in effect requiring as many multicast groups to be set up as the number of participants. The resulting one-way flow of traffic makes it problematic to use egress multicast traffic as an indicator of group memberships.

2) Snooping Routing Protocol Messages: Another way by which a firewall (particularly if it is also the border router) could deduce membership of multicast groups is from the routing protocol messages that construct the multicast tree. However, these routing protocol messages only carry information on the group itself, and not on the clients subscribed to that group. This does not suffice for building the desired $\mathcal{C}$-$\mathcal{G}$ relation, though it can assist in a weaker form of protection where only the set $\mathcal{G}$ of subscribed groups is used.

3) IGMP snooping: The IGMP (Internet Gateway Management Protocol) [11] provides direct information on membership of multicast groups on a LAN. Host applications that wish to join/leave a multicast group send IGMP join/leave messages to their designated router. In addition, the router periodically queries the LAN for membership of multicast groups. If the firewall is integrated with the router, it would be actively participating in IGMP and would hence know the membership of hosts in its connected LANs. A bridged firewall (that does not participate in routing) can deduce the same information by passively snooping for IGMP messages. A major drawback of this scheme is that it is restricted to scenarios where the enterprise network is a layer-2 network (LAN). If the enterprise has an internal routed network with multiple LANs, the firewall would not see IGMP messages on LANs not directly connected to it. This approach may therefore work well only for small enterprises with flat networks, and alternative schemes are needed for large enterprises that have routed networks.

C. Building the Client-Source ($\mathcal{C}$-$\mathcal{S}$) Relations

Much like a stateful unicast firewall, our multicast firewall records unicast communication that internal clients have with external sources. Some approaches are discussed below.

1) Snooping SIP Messages: The IETF SIP (Session Initiation Protocol) working group [12] is developing a text-based protocol for initiating interactive communication sessions between users, such as for audio, video, chat, interactive games, virtual reality, etc. Multicast applications are increasingly using SIP as the signalling mechanism. Snooping the unicast SIP messages generated by an internal client can help the multicast firewall in deducing the external sources from which legitimate multicast traffic can be expected. This approach will however not work with legacy multicast applications that use proprietary signalling message formats.

2) Snooping All Unicast Packets: In the absence of a common unicast signalling message format across all applications, one could have the firewall deduce the $\mathcal{C}$-$\mathcal{S}$ relation by snooping *all* unicast traffic egressing the enterprise network. This approach runs the risk that the firewall will permit an external host running a "trustworthy" unicast application to launch multicast attacks on a subscribed group. However, we believe the benefits of generality (i.e. not being tied into a specific signalling protocol) outweight the risks of this approach. The other concern could be the complexity of maintaining state on all unicast interaction, but we note that unicast firewalls today are built to cope with this.

D. Inferring the Source-Group ($\mathcal{S}$-$\mathcal{G}$) Relations

As described earlier, our multicast firewall algorithm allows a multicast packet from source s addressed to group g to enter only if there is a client that is subscribed to multicast group g and has had unicast interaction with source s. Stated notationally, an (s, g) packet enters only if there exists $c \in \mathcal{C}$ such that the (c, g) and (c, s) relations exist. The $\mathcal{S}$-$\mathcal{G}$ relations are therefore inferred from the $\mathcal{C}$-$\mathcal{G}$ and $\mathcal{C}$-$\mathcal{S}$ relations. The determination of $\mathcal{S}$-$\mathcal{G}$ relations is done for each incoming multicast packet (we call this the *query* operation), and therefore needs to be efficient. At the same time, these relations are updated whenever the $\mathcal{C}$-$\mathcal{G}$ or $\mathcal{C}$-$\mathcal{S}$ relations change, and the corresponding *update* operation also needs to be efficient. These performance trade-offs are discussed next.

E. Performance Trade-offs

Let $n_{\mathcal{C}}$, $n_{\mathcal{S}}$, and $n_{\mathcal{G}}$ denote respectively the size of the sets $\mathcal{C}$, $\mathcal{S}$, and $\mathcal{G}$ respectively. The $\mathcal{S}$-$\mathcal{G}$ relations could be precomputed or determined on-the-fly, as discussed next.

1) Pre-Computing the $\mathcal{S}$-$\mathcal{G}$ Relations: The $\mathcal{C}$-$\mathcal{G}$ relations could be stored in an $n_{\mathcal{C}} \times n_{\mathcal{G}}$ matrix where an entry is non-zero if the client corresponding to its row is related to the group represented by its column. A matrix could similarly store the $\mathcal{S}$-$\mathcal{C}$ relations, and the $\mathcal{S}$-$\mathcal{G}$ relations could be deduced from the product of the above two matrices.

With small number of active multicast groups, the storage space requirement scales linearly with the number of internal and external hosts. Queries require $O(1)$ time, while update operations (e.g. when a client talks to a previously unknown source) are more expensive. Sparse matrices require reduced storage at the expense of increased time for queries.

2) Dynamically Determining the $\mathcal{S}$-$\mathcal{G}$ Relation: The $\mathcal{S}$-$\mathcal{G}$ relation can be computed on-demand at the firewall, as and when multicast packets arrive at its external interface. In the matrix representation above, this would correspond to computing the (s, g) element of the $\mathcal{S}$-$\mathcal{G}$ relation matrix, where s denotes the source of the arriving packet and g the group it is addressed to. This operation requires $O(n_{\mathcal{C}})$ time, but practical implementations may use techniques that rely on the matrices being sparse. The advantage of dynamically computing the $\mathcal{S}$-$\mathcal{G}$ relation is that updates to the underlying $\mathcal{S}$-$\mathcal{C}$ and $\mathcal{C}$-$\mathcal{G}$ relation matrices do not entail upfront recomputations.

IV. Implementing the Dynamic Stateful Firewall

We implemented and tested our dynamic firewall as a plug-in to Linux NetFilter, which is a packet filtering platform in the Linux 2.4.x and 2.6.x kernel series. We wrote a kernel module that maintains the state required for our dynamic multicast firewall. Rules were inserted into the NetFilter's rule-table to direct all incoming multicast packets to our module, as also all outgoing unicast traffic. Note that our module does not need to snoop incoming unicast traffic. Within the module, data structures are maintained to hold information about the set of active multicast groups $\mathcal{G}$, the set of receivers (internal hosts) $\mathcal{C}$, and the set of sources (external hosts) $\mathcal{S}$. Our current proof-of-concept implementation maintains these sets in a combination of hashed tables, with linked lists used to resolve collisions. In the future we intend to optimise the implementation using more sophisticated data structures.

The $\mathcal{C}$-$\mathcal{G}$ relations are derived by snooping IGMP messages. This suffices for our testing scenarios in which the internal network is switched and has no routers. We further clarify here that our firewall operates as a bridge (*i.e.* no routing functionality), and routing is performed by a router connected to its external interface. When our firewall sees an IGMP report indicating that a client has joined a group, the data structures for the two are linked together with pointers. When it is detected that clients have left multicast groups (via explicit IGMP host leave messages or via inactivity timeout), the associated pointers are removed.

The $\mathcal{C}$-$\mathcal{S}$ relations are derived by observing all egress unicast traffic. When an internal host interacts with a previously unknown external host, a new node is created for the external host and pointers used to link the two. As in the previous explanation, these relations are removed after an inactivity timeout (of the order of a few minutes).

The kernel firewall rules are configured so all multicast packets arriving on the external interface are passed on to our module. For a multicast packet from source s addressed to group g, our module searches its data structure for group g in set $\mathcal{G}$, and for source s in set $\mathcal{S}$. If either is unknown, the packet is dropped. Otherwise, it scans through the clients $c \in \mathcal{C}$ that are members of group g till it finds one that has a relation to s. If such a c is found, the packet is admitted into the enterprise network, otherwise it is dropped.

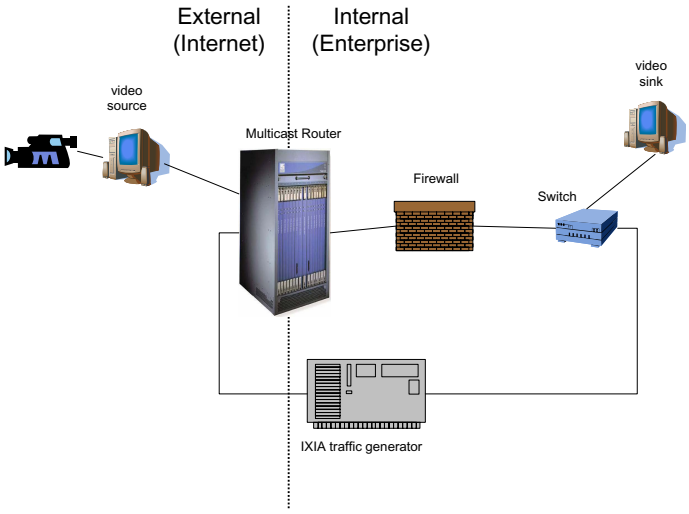
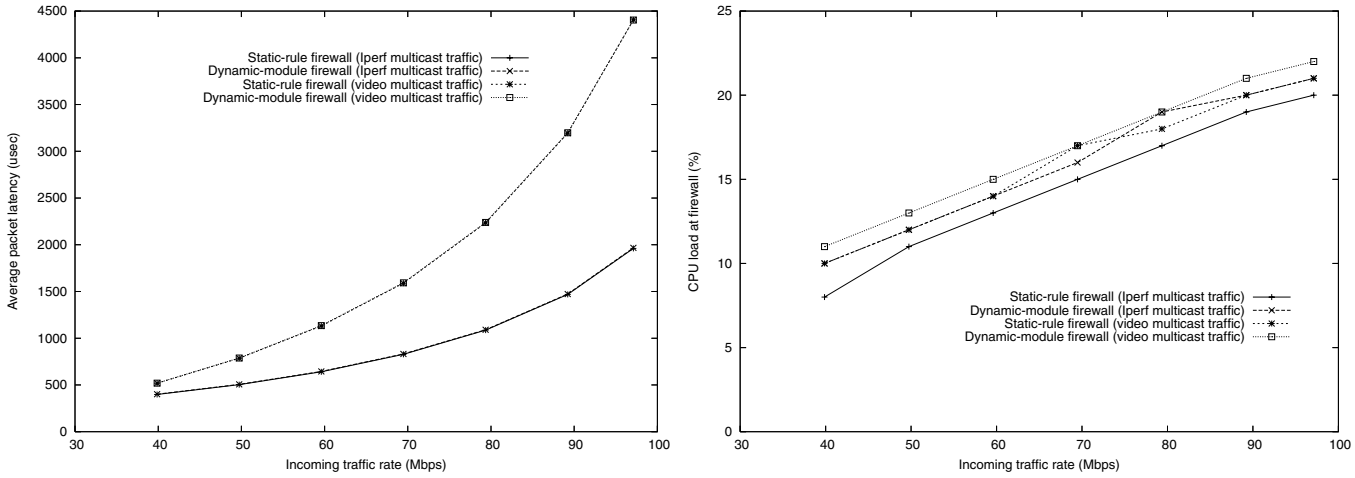


Fig. 1. Topology for Experiments

We make two important observations here: first, that the network administrator can override our module in a natural way by configuring explicit static rules to permit/deny specific multicast sessions (this may be desirable for certain important sessions or where there is no unicast signalling interaction). Packets for these sessions will then be handled by the kernel rule-table, and will not even be seen by our module. Second, as a fall-back option for multicast applications that do not have periodic unicast signalling with the source(s), an end-user can start ping session(s) to the source(s), which will cause the multicast firewall to build the appropriate state.

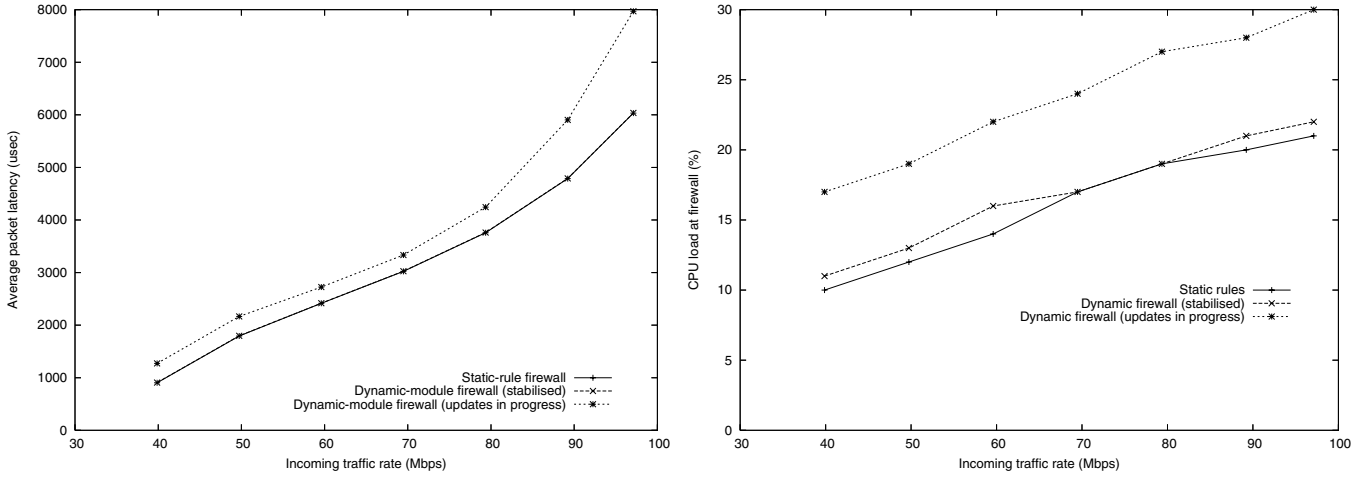
V. Performance Evaluation

We tested our implementation on our laboratory test-bed with topology as shown in 1. The firewall device is a Dell desktop computer with a 2.8 GHz Pentium-IV processor and 512 MBytes of RAM. It was running the Red Hat Enterprise Linux 4 Update 1 (kernel version 2.6.9-11EL) distribution. We made the firewall operate in bridging mode (*i.e.* no routing), so as to isolate firewall functionality from routing issues that are beyond the scope of this paper. The firewall has two 100 Mbps Ethernet interfaces, the external one connected to a Nortel Passport 8600 series router running the PIM-SM multicast routing protocol. The router is in turn connected to a PC generating a multicast video stream, and to a port of the IXIA traffic generator. The internal interface of the firewall is connected via a layer-2 switch to a PC that sinks the video traffic, as well as to a port on the IXIA traffic monitor. The IXIA is a specialised hardware traffic generator and analyser. It is capable of generating arbitrary packet streams, including unicast, multicast, IGMP, etc., and analysing packet statistics including data rates, latencies, etc. at high data rates. Our tests below rely on its capabilities to emulate a large number of multicast sessions and multicast DoS attack scenarios.



(a) (b)

Fig. 2. Scenarion I [multicast-performance]: (a) packet latency and (b) firewall CPU load as a function of traffic load.



(a) (b)

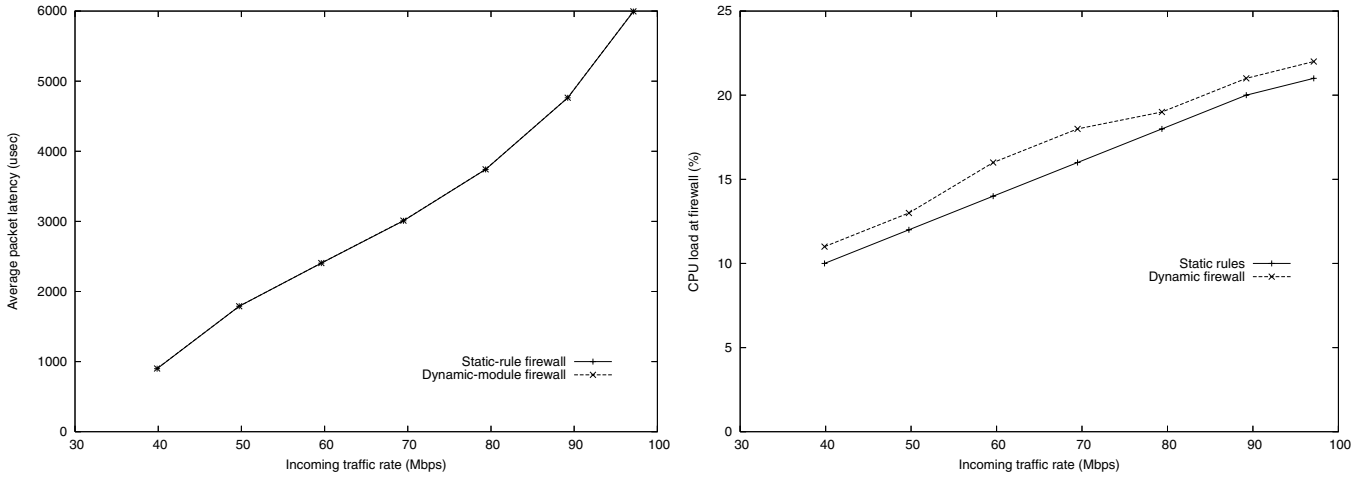
Fig. 3. Scenarion II [multicast-scaling]: (a) packet latency and (b) firewall CPU load as a function of traffic load.

We conduct three sets of experiments designed to measure performance, scaling and attack-resilience of our prototype implementation. In each scenario we not only verify correctness of our firewall (namely that it does not unduly block wanted multicast traffic or permit unwanted traffic), but also quantify its performance in terms of the latency for incoming unicast traffic, and CPU load at the firewall device itself.

Scenario I [multicast-performance]: In this scenario, a 30 Mbps multicast video stream flows from the external to the internal PC, accompanied by a low-rate unicast heartbeat. After verifying that our dynamic firewall correctly admits the multicast traffic, we introduce unicast flows from the IXIA's external port to the internal one via the firewall. The unicast traffic rate is varied, such that the total traffic rate on the firewall's interface varies from 40 to 100 Mbps, and the latency for the unicast traffic is measured. The objective is to quantify if our module affects the enterprise's unicast download latency in the presence of multicast traffic. The baseline for comparison in all our experiments is where explicit static rules are configured to admit the multicast sessions.

Figure 2(a) plots the packet latency for incoming traffic as a function of traffic load for two cases: (i) when the multicast traffic is synthetically generated using Iperf, a network traffic generation and measurement tool, and (ii) when a live video session (in DV format, similar to what we use in our AccessGrid platform) streams the multicast traffic. For both these cases, the end-to-end packet latency is plotted for our dynamic firewall module as compared to the baseline that uses static rules; the curves overlap showing that the dynamic operation incurs no performance penalty. Figure 2(b) shows the corresponding CPU load at the firewall device. Our dynamic firewall module causes no more than around 1% or so higher load on the CPU, which is quite a small price to pay for freedom from manual configuration.

Scenario II [multicast-scaling]: In this scenario we test the scaling capability of our scheme by having the IXIA emulate a large number of multicast groups and external sources. To add to the 30 Mbps video stream, the internal Ixia port is made to join 50 multicast groups (address range 239.1.2.3 to 239.1.2.52) and emulate 250 internal clients (address range



(a) (b)

Fig. 4. Scenarion III [multicast-attack]: (a) packet latency and (b) firewall CPU load as a function of traffic load.

$10.2.0.0/24$) conversing with 8000 external sources (address range $10.1.0.0/19$). The base-line performance is measured by configuring the firewall with static rules to permit the above sessions. We then remove the static rules, load our dynamic firewall module, and repeat the experiment. We distinguish the case when the $\mathcal{C}$-$\mathcal{S}$ and $\mathcal{C}$-$\mathcal{G}$ relations have already been established (we call this the stabilised case) from the transient case when the relations are being established – the latter includes the performance impact caused due to heavy "update" operations on the relations.

Figure 3(a) shows packet latency while 3(b) shows associated CPU load. We see that the stabilised performance curves overlap with the base-line case, indicating that the query operations hardly incur a computational overhead. However, when the relations are being established (transient), the performance degradation is clearly visible. This indicates that "updates" are expensive, probably due to the memory allocattion calls when relations are being created. Insight from this scenario is leading us to re-evaluate our design to minimise sytem memory allocation in the next version of our implementation.

Scenario III [multicast-attack]: Our third scenario simulates multicast denial-of-service (DoS) attacks, originating from the external IXIA port, in addition to the legitimate traffic described in the previous scenario. We only consider the case where the attackers (from IP range $10.1.128.1/19$) send packets to groups that clients in the enterprise have subscribed to (otherwise the multicast routing protocols would not deliver the attack packets to the enterprise anyway). We now quantify if the attacks cause performance degradation at the firewall (CPU load) or for internal clients (packet latency) as the intensity of the attack packets (and hence load on the firewall interface) varies. Figure 4(a) shows that the packet latency is hardly distinguishable from the base-line where the firewall uses static rules, and figure 4(b) shows the associated CPU load to be only marginally higher. From this we conclude that our firewall dynamically allows solicited traffic without incurring much perceptible performance cost even under reasonably severe DoS attacks.

VI. Summary and Future Work

In this paper we have proposed, prototyped, and tested a first-of-a-kind multicast firewall algorithm that dynamically determines solicited from unsolicited multicast traffic. This eliminates the need for manual configuration of firewall rules on a per-multicast-session basis, while protecting the enterprise network against multicast DoS attacks. We prototyped our idea as a simple plug-in module to the Linux NetFilter framework, and showed that it is easily implementable and almost as efficient in terms of packet latency and CPU load as having static rules. We believe that enterprises eager to use emerging multicast applications could hugely benefit from such a dynamic stateful multicast firewall.

Our future work is improving the performance of our prototype implementation by reducing memory allocation system calls, and investigating new data structures and algorithms that provide appropriate trade-offs in space and time complexity for typical enterprise traffic patterns.

References

[1] S. Deering and D. Cheriton. Multicast Routing in Datagram Internetworks and Extended LANs. *ACM Transactions on Computer Systems*, 8(2):85–110, May 1990.
[2] MMOGCHART. *http://www.mmogchart.com*.
[3] Microsoft TV: IPTV Edition. *http://www.microsoft.com/tv/IPTVEdition.mspx*.
[4] The Access Grid. *http://www.accessgrid.org*.
[5] S. Ratnasamy, A. Ermolinskiy, and S. Shenker. Revisiting IP Multicast. In *Proceedings of ACM Sigcomm*, Pisa, Italy, Sep 2006.
[6] LWN.net. Multicast Impacts from the Ramen Worm, 2001. *http://lwn.net/2001/0125/security.php3*.
[7] IETF Multicast Security (MSEC) working group. *http://www.ietf.org/html.charters/msec-charter.html*.
[8] H. Holbrook and D. Cheriton. IP Multicast Channels: Express Support for Single-Source Multicast Applications. In *Proceedings of ACM Sigcomm*, Cambridge, MA, Sep 1999.
[9] IETF Source-Specific Multicast (SSM) working group. *http://www.ietf.org/html.charters/ssm-charter.html*.
[10] L. Oria. Approaches to Multicast over Firewalls: An Analysis. Technical Report HPL-IRI-1999-004, HP Labs, Aug 1999. *http://www.hpl.hp.com/techreports/1999/HPL-IRI-1999-004.html*.
[11] W. Fenner. Internet Group Management Protocol, version 2. RFC 2236. *http://www.ietf.org/rfc/rfc2236.txt*.
[12] IETF Session Initiation Protocol (SIP). *http://www.ietf.org/html.charters/sip-charter.html*.