

VULNERABILITIES OF RELIABLE MULTICAST PROTOCOLS

Thomas M. Parks
David A. Kassay
Clifford J. Weinstein

Massachusetts Institute of Technology
Lincoln Laboratory
Lexington, Massachusetts

ABSTRACT

We examine vulnerabilities of several reliable multicast protocols. The various mechanisms employed by these protocols to provide reliability can present vulnerabilities. We show how some of these vulnerabilities can be exploited in denial-of-service attacks, and discuss potential mechanisms for withstanding such attacks.

1 INTRODUCTION

Group communication can be achieved through the use of multiple unicast transmissions. However, this is inefficient because some network links carry multiple copies of the same data. Figure 1 illustrates a situation that could occur when a web server sends live audio to several receivers. Multicast transmission makes more efficient use of network resources. Data is routed such that it never traverses the same network link twice, as shown in Figure 2.

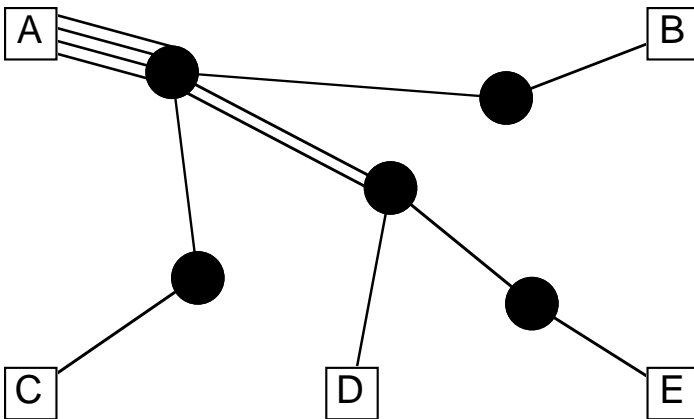


Figure 1: Unicast transmission from a single source (A) to multiple receivers (B, C, D, and E).

Some group communication applications, such as inter-

This work was sponsored by the Defense Advanced Research Projects Agency. Opinions, interpretations, conclusions and recommendations are those of the authors and are not necessarily endorsed by the United States Air Force.

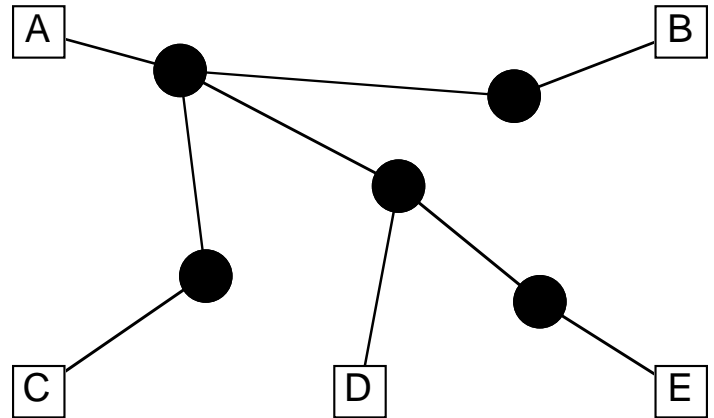


Figure 2: Multicast transmission from a single source (A) to multiple receivers (B, C, D, and E).

active audio/video conferencing, use unreliable multicast transmission. This is because such applications can tolerate some data loss, but cannot tolerate the delays caused by waiting for the retransmission of missing data. Other applications, such as situation awareness and replicated file servers, require reliable multicast transmission. Many different reliable multicast protocols have been developed to meet the differing needs of their applications.

Most reliable multicast protocols are optimized for performance and are robust to common faults, such as lost packets and failure of one or more group members. Few protocols are designed to withstand intentional, malicious attacks. As experimental protocols become candidates for standardization, their vulnerabilities to such attacks will be an important evaluation criterion [1]. We will discuss vulnerabilities of several protocols that allow an attacker to disrupt a communication session. We will also discuss cryptographic and non-cryptographic defense mechanisms.

In an attempt to prevent attacks, network-level security, such as IPSEC [2, 3, 4], could be employed to make every packet transmission secure. But this protection comes at significant cost and cannot protect against all attacks. Encryption can provide privacy to prevent other parties from

observing the data being exchanged in a group communication session. However, authentication and integrity, are the security services required to protect a protocol from disruption. Authentication identifies the origin of a message, and integrity assures that a message is not altered in transit.

Secret-key algorithms, such as keyed MD5 [5], can provide integrity and limited authentication for group communication. All group members use a single, shared key to authenticate packets. This can protect a protocol from certain kinds of attacks from adversaries who do not possess the shared key. However, in a mobile wireless network environment, there is the real possibility that the shared key may be compromised through the loss or capture of one or more group members. Imagine every soldier in an army being provided with a pager, for example. It is likely that several of these small devices would be lost each day.

It is necessary to use public-key algorithms for authentication in a group setting, so that each member has a unique key. The complexity of public-key algorithms severely limits performance. On the order of 10 messages can be signed, and 10 to 100 messages can be verified each second using public-key algorithms*. This may be acceptable when each message is a large image, but it may not be acceptable when each message is an individual network packet.

Rather than crippling the performance of a reliable multicast protocol by applying cryptography in the name of security, we propose that a risk management approach be taken. It is not necessary to make all attacks impossible, but merely to make them sufficiently difficult.

2 RELIABLE MULTICAST

The Transmission Control Protocol (TCP) meets the general requirements for reliable, ordered delivery of packets for unicast transmission. No such general purpose protocol exists for multicast transmission. Reliable multicast communication is important for applications such as multimedia conferencing, replicated file servers, and distributed interactive simulation, among others. Because group communication applications have widely varying reliability requirements, many different reliable multicast protocols have been developed, none of which are dominant standards as TCP is for reliable unicast. Group communication can be one-to-many or many-to-many with small or large group sizes. Some applications require packets to be delivered in order, while others do not. Some applications require stability (the fact that the sender knows that a packet has been received), while others do not. Different protocols provide different levels of reliability appropriate to their particular applica-

*Measurements were made with hardware and software implementations of the DSA and RSA algorithms.

tion.

2.1 SENDER-INITIATED RELIABILITY

A sender-initiated reliability protocol places the burden of loss detection on the sender. A positive acknowledgment (ACK) is required from every receiver for every packet sent. A lost packet is detected when the sender fails to receive an ACK from every receiver within some time limit. When a loss is detected, the packet is retransmitted and the sender again waits for an ACK from every receiver.

Such sender-initiated protocols suffer from ACK implosion [6, 7]: increasing amounts of both network bandwidth and processing time are consumed as the number of receivers in the group increases. Denial-of-service attacks that produce a similar implosion of ICMP echo reply packets have been reported [8].

Some protocols avoid ACK implosion by organizing the group into a logical tree [9, 10] or ring [11, 12] in order to reduce the number of ACKs that must be processed by any one group member. The links connecting the nodes of the tree or ring need not correspond to network links between group members. Both tree and ring protocols require that the group membership be known. Thus, there must also be a protocol for reliably distributing the membership list to all group members.

In the Reliable Multicast Protocol (RMP) [12], members pass a token around a ring. The ring serves several purposes. Only the current token holder acknowledges packets. This eliminates the ACK implosion problem and improves throughput. A global ordering of packets from different sources is determined by a sequence number in each ACK. The circulation of the token completely around the ring indicates that all group members have received all packets up to that point.

2.2 RECEIVER-INITIATED RELIABILITY

A receiver-initiated reliability protocol places the burden of loss detection on the individual receivers. Receivers generate a negative acknowledgment (NAK) when they detect a lost packet. The packet is retransmitted in response to one or more NAKs. NAK implosion is still possible if a large number of receivers lose the same packet. Suppression mechanisms can be used to minimize the number of duplicate NAKs produced when such correlated losses occur. Similar suppression mechanisms can be used to prevent a flood of retransmissions when any member with the appropriate data may respond to a NAK [13, 14].

In the Scalable Reliable Multicast (SRM) [13] protocol, receivers send NAKs to indicate that a packet has been lost. Periodic status messages are emitted by every receiver to

announce the highest sequence number received from each source. These status messages serve as a form of positive acknowledgment. In order to suppress excess NAKs in response to a correlated loss, receivers send their NAK to the entire group. If a receiver hears a NAK for the same lost packet, then that receiver will suppress the sending of its own NAK. Similarly, members responding to a NAK will send their retransmission to the entire group, and will cancel their own retransmission if they hear another one. This will suppress duplicate responses to a NAK.

3 VULNERABILITIES

Our initial experiments have focused on the Reliable Multicast Protocol (RMP) and the Scalable Reliable Multicast (SRM) protocol. However, attacks that exploit specific features of these protocols would likely affect other similar protocols that share these features. Other attacks described here are general in nature and could affect both multicast and unicast transmissions. For each attack, we discuss possible defense mechanisms. Often the solution is to ensure that routers are properly configured and to employ cryptographic authentication.

3.1 FLOODING ATTACK

Flooding is a general, brute force attack in which an adversary simply sends a large amount of data to a particular receiver, or the entire multicast group, to consume resources and degrade the quality of service provided to legitimate group members. It is not necessary for such an attacker to join the multicast group, to be in possession of cryptographic keys, or to have any knowledge of the proper packet format for the multicast protocol. The packets the attacker sends will simply consume network bandwidth and processing time.

3.2 FORGED DATA ATTACK

An attacker who has knowledge of the packet formats for a protocol can send forged data packets to the multicast group. We have implemented such an attack against RMP, which causes corruption of the file being transmitted. These forged data packets can be sent not only from any member of the ring, but from any network host. This attack is possible because this particular implementation of RMP does not verify that the network address in a packet corresponds to the identifier of the appropriate group member. If such consistency checking were performed, then an attacker would have to spoof its address in order for its forged data to be accepted as legitimate. Additionally, digital signatures could be employed to protect even against spoofing. However, this would impose severe throughput limitations, as mentioned

earlier.

3.3 PREMATURE ACK

We have developed an attack against RMP where an outsider that is not a member of the ring sends forged ACKs to the group and disrupts the ordering of packets. The attacker listens to the ACKs being exchanged by legitimate group members to determine the current sequence number, and then sends an ACK to the group for a data packet that has not yet been transmitted. When the corresponding data packet is eventually transmitted, it will be assigned the incorrect sequence number contained in the forged ACK.

3.4 NAK AND RETRANSMISSION SUPPRESSION

In the Scalable Reliable Multicast (SRM) protocol, a receiver multicasts a negative acknowledgment (NAK) to the entire group when it detects that a packet has been lost. When other receivers that have lost the same packet hear that NAK, they suppress the transmission of their own NAK. This prevents a flood of NAKs when correlated packet losses occur, thus saving network bandwidth. However, this efficiency mechanism represents a vulnerability that can be exploited.

By adjusting the time-to-live (TTL) field in the packet header, an attacker can generate NAKs that reach only a subset of the group members. This will suppress NAKs from the members that receive this NAK without triggering a retransmission from the other members of the group. Similarly, an attacker can generate a retransmission with a reduced TTL, which will cancel legitimate retransmissions from other group members without reaching the members that requested the retransmission. These attacks can interfere with retransmissions, which means that they are effective only when losses actually occur. However, losses can be induced through a flooding attack.

4 DEFENSES

A first line of defense is to give receivers the capability to filter out packets received from an attacker. This prevents packets from the attacker from interfering with the operation of the reliable multicast protocol, and minimizes the processing time dedicated to those packets. However, filtering alone does not reduce the network bandwidth consumed, such as in a flooding attack, because packets are simply discarded after they have already been received. In order to conserve bandwidth and prevent congestion, it is necessary to block the transmission of packets close to the source. Version 3 of the Internet Group Management Protocol (IGMP), introduced the ability for members of a multicast group to request such blocking [15].

For either filtering or blocking to be effective, it is necessary to identify the addresses of legitimate group members. In some reliable multicast protocols, such as SRM, group members send periodic messages that contain their identity and status. A simple but effective security measure is to digitally sign these messages. Thus the identity of each group member would be securely bound to a network address.

To circumvent these protective measures, an attacker would have to “spoof” its network address so that its packets appear to come from a legitimate group member. This would force other members of the group to choose between blocking traffic from both the attacker and a legitimate group member or allowing all traffic through. However, if network routers are properly configured to prevent spoofing [16], an attacker could only affect group members on its own local network where there is no intervening router to filter out messages.

A simple authentication protocol, such as keyed MD5 [5], could prevent even this limited spoofing from attackers who do not possess the secret key shared by the group members. If the secret key were compromised, then it would be necessary to distribute a new key, or resort to a more costly public-key authentication protocol in order to completely eliminate the possibility of spoofing.

5 SECURE MULTICAST PROTOCOLS

Secure reliable multicast protocols do exist [17, 18]. These protocols can survive attacks even if up to one-third of the group members have been compromised. They include mechanisms whereby honest group members can detect other group members that exhibit malicious behavior or otherwise fail to properly execute the reliable multicast protocol. However, these protocols rely on digital signatures to authenticate critical control messages. Due to the complexity of public key algorithms, only about 10 digital signatures can be generated, and 10 to 100 signatures verified per second. This cryptographic processing overhead limits the performance and scalability of such protocols. This performance penalty may be acceptable in high security applications.

6 CONCLUSION

The attacks described in this paper have been implemented, and some have been demonstrated to be quite effective. We are conducting controlled experiments on a small heterogeneous wired/wireless network to measure the performance of several reliable multicast protocols in both a benign environment and in the presence of various combinations of these attacks.

We have discussed defense mechanisms ranging from

simple packet filtering to complex authentication protocols. These solutions offer increasing levels of protection at increasing costs. We will enhance some of the reliable multicast protocols discussed in this paper and conduct controlled experiments to measure the performance penalty incurred by these defense mechanisms.

The reliable multicast protocols discussed in this paper lack mechanisms to detect attacks. Adapting such mechanisms from secure multicast protocols is a topic for future work. This will complete the set of tools to protect against attacks, detect attacks, and react to attacks.

REFERENCES

- [1] Allison Mankin, Allyn Romanow, Scott Bradner, and Vern Paxson. IETF criteria for evaluating reliable multicast transport and application protocols. Request for Comments 2357, IETF Network Working Group, June 1998. <<ftp://ftp.isi.edu/in-notes/rfc2357.txt>>.
- [2] Randall Atkinson. Security architecture for the internet protocol. Request for Comments 1825, IETF Network Working Group, August 1995. <<ftp://ftp.isi.edu/in-notes/rfc1825.txt>>.
- [3] Randall Atkinson. IP authentication header. Request for Comments 1826, IETF Network Working Group, August 1995. <<ftp://ftp.isi.edu/in-notes/rfc1826.txt>>.
- [4] Randall Atkinson. IP encapsulating security payload. Request for Comments 1827, IETF Network Working Group, August 1995. <<ftp://ftp.isi.edu/in-notes/rfc1827.txt>>.
- [5] Perry Metzger and William Allen Simpson. IP authentication using keyed MD5. Request for Comments 1828, IETF Network Working Group, August 1995. <<ftp://ftp.isi.edu/in-notes/rfc1828.txt>>.
- [6] Sridhar Pingali, Don Towsley, and James F. Kurose. A comparison of sender-initiated and receiver-initiated reliable multicast protocols. In *SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, May 1994. <<ftp://gaia.cs.umass.edu/pub/Ping94:Multicast.ps.Z>>.
- [7] Brian Neil Levine and J. J. Garcia-Luna-Aceves. A comparison of reliable multicast protocols. *ACM Multimedia Systems Journal*, 6(5), August 1998. <<http://www.cse.ucsc.edu/research/ccrg/publications/brian.mmsj.ps.gz>>.
- [8] “smurf” IP denial-of-service attacks. CERT Advisory CA-98.01, CERT Coordination Center, Septem-

- ber 1998. <ftp://ftp.cert.org/pub/cert_advisories/CA-98.01.smurf>.
- [9] Brian Neil Levine, David B. Lavo, and Garcia-Luna-Aceves J. J. The case for reliable concurrent multicasting using shared ack trees. In *Multimedia*, Boston, Massachusetts, November 1996. ACM. <<http://www.cse.ucsc.edu/research/ccrg/publications/brian.mm96.ps.gz>>.
- [10] Sanjoy Paul, Krishnan K. Sabnani, John C. Lin, and Supratik Bhattacharyya. Reliable multicast transport protocol (RMTP). *IEEE Journal on Selected Areas in Communications*, 15(3):407–421, April 1997. <<http://www.bell-labs.com/user/sanjoy/rmtp2.ps>>.
- [11] Jo-Mei Chang and N. F. Maxemchuk. Reliable broadcast protocols. *ACM Transactions on Computer Systems*, 2(3):251–273, August 1984.
- [12] Brian Whetten, Todd Montgomery, and Simon Kaplan. A high performance totally ordered multicast protocol. Technical Report TR-94-069, International Computer Science Institute, Berkeley, California, December 1994. <<ftp://ftp.icsi.berkeley.edu/pub/techreports/1994/tr-94-069.ps.gz>>.
- [13] Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, and Lixia Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, December 1997. <<http://www-nrg.ee.lbl.gov/floyd/srm-paper.html>>.
- [14] Bikash Sabata, Michael J. Brown, Barbara A. Denny, and Chung-ho Heo. Transport protocol for reliable multicast: TRM. In *IASTED International Conference on Networks*, pages 143–145, Orlando, Florida, January 1996. <<http://www.erg.sri.com/people/sabata/paper/isted.ps>>.
- [15] Brad Cain, Steve Deering, and Ajit Thyagarajan. Internet group management protocol, version 3. Internet draft, IETF Inter-Domain Multicast Routing Working Group, November 1997. <<ftp://ftp.ietf.org/internet-drafts/draft-ietf-idmr-igmp-v3-00.txt>>.
- [16] Paul Ferguson and Daniel Senie. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. Request for Comments 2267, IETF Network Working Group, January 1998. <<ftp://ftp.isi.edu/in-notes/rfc2267.txt>>.
- [17] Michael K. Reiter. Secure agreement protocols: Reliable and atomic group multicast in Rampart. In *Conference on Computer and Communications Security*, pages 68–80. ACM, November 1994. <<http://www.research.att.com/reiter/papers/ccs2.ps.gz>>.
- [18] Kim Potter Kihlstrom, L. E. Moser, and P. M. Melliar-Smith. The SecureRing protocols for securing group communication. In *International Conference on System Sciences*, Kona, Hawaii, January 1998. <<http://alpha.ece.ucsb.edu/ftp/SMP/hicss98.ps.gz>>.