

Short Paper: Schemes for Enhancing the Denial-of-Service Tolerance of SRTP

Sachin Garg
Avaya Labs Research
Basking Ridge, NJ, USA
sgarg@avaya.com

Navjot Singh
Avaya Labs Research
Basking Ridge, NJ, USA
singh@avaya.com

Timothy Tsai
Sun Microsystems
Santa Clara, CA, USA
Timothy.Tsai@sun.com

Abstract

Secure Real-time Transport Protocol (SRTP) provides confidentiality, authentication, integrity and replay protection for secure media transport in VoIP. However, the overhead of HMAC-SHA1 incurred per packet makes SRTP susceptible to flooding based Denial-of-Service attack. In this paper, we present a class of schemes to increase the DoS tolerance in SRTP. The central idea is to add a light-weight authentication mechanism on top of SRTP. This mechanism is used to efficiently discard illegitimate packets early on in the face of a DoS attack. Analysis shows that substantially larger traffic flood can be handled with the proposed enhancements.

1. Introduction

Real-time Transport Protocol (RTP) [3] used as the basis for media transport in VoIP is susceptible to several attacks, including snooping, injection of forged content, and packet replay. To address these concerns, Secure Real-time Transport Protocol (SRTP) [1] was recently standardized by IETF. SRTP provides message confidentiality, authentication, integrity checking and replay protection for RTP traffic on a per-packet basis. The standard specifies AES encryption of the RTP payload and a message authentication hash of the header and the encrypted payload using HMAC-SHA1 [2]. The HMAC is truncated to a 32-bit authentication tag and appended to the packet before transmission. The receiver repeats the HMAC computation on the received header+payload and compares it to the tag. A match not only confirms data integrity but also authenticates the sender.

One potential concern for SRTP is the overhead imposed by the authentication hash. The HMAC-SHA1 based tag must be computed for each packet before authentication can be determined. Thus, one possible DoS attack is to bombard a target with a series of forged RTP packets, each of which contains an improper authentication tag. We start by giving

a concrete example to motivate the need for enhancing DoS tolerance in SRTP.

DoS Example: The main SHA-1 hash computation includes hashing $L + 3$ blocks of 512 bits, where $L = \lceil \frac{M}{64} \rceil$ and M is the number of bytes in the RTP header and the payload. For G.711 speech coding at 8KHz, each RTP packet contains 160 bytes of payload and 12 bytes header. Thus $L = \lceil \frac{172}{64} \rceil = 3$ implying a total 6 SHA1 hash operations per packet. Each hash computation involves a total of 740 32-bit logical and arithmetic operations. On a 60Mhz processor, $740 \times 6 = 4440$ operations alone, not counting control flow instructions, would take approximately $76\mu s$. An attacker can bombard the victim device with forged RTP packets with the sole intent to consume processor cycles by invoking the authentication process. Theoretically, a flood rate of 13157 packets per second would ensure that the CPU is completely consumed in only authentication. Assuming minimum size frames, this would amount to less than 5 Mbps bandwidth consumed of the 10/100 Mbps Ethernet pipe. The rest of the paper is organized as follows. Section 2 describes two schemes, which significantly reduce the processor cycles consumed by forged packets. In Section 3, the schemes are compared to basic SRTP vis-a-vis their performance. Finally, Section 4 concludes the paper.

2. Enhancing DoS Tolerance (SRTP+)

This section describes the two schemes, which we shall collectively refer to as SRTP+. The central idea behind each is to utilize a first-level, cheap message authentication that does not require HMAC of the entire header and payload. Once this authentication passes, normal SRTP authentication and integrity checking is applied. When the device is under attack, the added first-level authentication efficiently discards illegitimate packets. Under normal operation, however, if all packets are legitimate, a minimal, but constant overhead is imposed. Both schemes append a tag to each packet. However, they differ in the manner in which the authentication tag is generated and verified for each RTP packet.

2.1. Secret PRN sequence (Scheme 1)

In this scheme the authentication tag is a number in a cryptographically secure pseudorandom number sequence. Each packet is assigned a pseudorandom number based on its sequence number such that successive sequence numbers in RTP packet correspond to successive pseudorandom numbers in the pseudorandom number sequence. The endpoints of the RTP communication must know the pseudorandom number corresponding to each sequence number, both for generating the authentication tag for the outgoing packet and for verifying the tag of an incoming packet.

Figure 1 shows the communications between two endpoints. Before any RTP packet is sent, the seed for the PRNG is exchanged in a secure manner. The exchange of SRTP+ PRNG keys can be piggybacked on top of the secure exchange of keys for SRTP. After the keys are exchanged, a 32-bit SRTP+ authentication tag is attached to each RTP packet. This authentication tag is transmitted in cleartext, since knowledge of the authentication tag for one packet cannot yield the tag for succeeding packets. If packets ar-

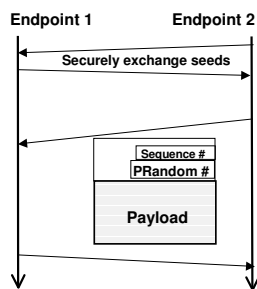


Figure 1. SRTP+ Exchange for Scheme 1

rive out of order or are dropped, the sequence number of the packet will not be the expected number. In that case, the PRNG must iterate multiple times until the pseudorandom number for the correct sequence number is calculated. For example, if the last received sequence number was 1000 and the sequence number of the newly received packet is 1004, then the PRNG must iterate four times to calculate the pseudorandom number corresponding to sequence number 1004. If the packets have indeed arrived out of order, the pseudorandom numbers for the sequence numbers 1001, 1002, and 1003 must be remembered because the PRNG can only calculate numbers going forward from the current number. Thus, a sliding window of calculated pseudorandom number must be maintained. When the authentication tag for an incoming packet matches the pseudorandom number for the lowest sequence number in the window, the sliding window can be shifted forward to discard the verified packet. The size of the sliding window should be based on the largest forward skip in sequence numbers

that can be tolerated by the underlying media codec. With a 30 millisecond packetization interval, and assuming delay tolerance of 150 ms, this would amount to a window size of 5 packets. One advantage of using a sliding window is that pseudorandom numbers for future packets can be calculated in advance, which allows the computationally intensive portion of the authentication process to be shifted away from the time when a packet arrives. We note that this Scheme does not enhance tolerance of man-in-the-middle attacks, where the attacker can intercept and/or modify the RTP packets.

2.2. Secret random number chaining (Scheme 2)

In this scheme, avoiding the need for SHA1 altogether, the sender calculates in advance a series of random numbers and uses one of these numbers as the authentication tag for each packet. The authentication tag is sent in cleartext, but the random numbers for the next N packets are encrypted (e.g., included in the SRTP encrypted payload). The receiver stores the N random numbers after decrypting the payload. These random numbers correspond to the sequence numbers for the next N expected packets and are compared to the authentication tags for succeeding packets for authentication. Before the first packet can be authenticated, the first N random numbers must be sent to the receiver, possibly during the SRTP key exchange. Figure 2 shows the communications between the two endpoints.

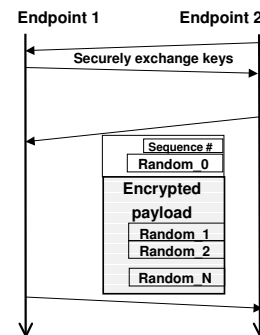


Figure 2. SRTP+ Exchange for Scheme 2

The main advantage of this scheme is the extremely low overhead of verifying authentication, which requires a simple arithmetic comparison. However, additional bytes need to be encrypted at the sender and decrypted at the receiver.¹ Furthermore, these encrypted bytes increase the size of the RTP payload.

¹If the last block of the block cipher encryption involves a number of cleartext bits that is less than the block size, the unused bits can be used to hide the N random numbers. Unfortunately, the default 160-byte payload for G.711 encoding leaves no unused bits when used with 128-bit AES blocks.

Table 1. Measured Performance for SRTP+ Schemes

	1.5 GHz Pentium IV			866 MHz Pentium III			450 MHz Sparc		
	μ s	overhead	speedup	μ s	overhead	speedup	μ s	overhead	speedup
Scheme1	2.04	26%	3.8	2.59	28%	3.5	5.95	26%	3.8
Scheme2 (128-bit key)	0.47	6%	16.5	0.80	9%	11.5	2.83	12%	8.0
SRTP	7.75			9.19			22.68		

3. Performance Analysis

To gain an understanding of the anticipated performance improvements from each SRTP+ scheme, we implemented versions of the schemes on several systems. (1) a 1.5 GHz Pentium IV system with 256 MB of RAM running Linux 2.4.7-10, (2) an 866 MHz 2-CPU Pentium III system with 512 MB of RAM running Linux 2.4.20-20.9, and (3) a 450 MHz 4-CPU Sparc system with 4 GB RAM. All test programs executed on a single CPU, and memory contention was not an issue. The open-source beecrypt-3.1.0 package was used for the SHA1 and HMAC-SHA1 implementations. Table 1 shows the measured performance for each SRTP+ scheme as well as the performance for SRTP. These numbers only show the performance for the portion of the run-time overhead imposed by the random number generation for Scheme 1, the AES encryption of one 128-bit block for Scheme 2, and the HMAC-SHA1 operation for SRTP with a 172-byte RTP packet.

Several observations can be made from Table 1. Both schemes offer a performance improvement relative to SRTP. The speedup is more than an order of magnitude for Scheme 2. For SRTP and a 172-byte RTP packet, $L=\text{ceil}(172/64)=3$, and $N=L+1=4$. As expected, the run-times for SRTP are approximately 4 times that for Scheme 1. Scheme 2 is much faster than the other schemes largely because the 128-bit AES block cipher encryption operates on a smaller input block compared to the 512-bit SHA1 hash operation.

When an IP endpoint is under attack with forged packets, the proposed schemes in SRTP+ will determine that the packet is fake resulting in a discard. Normal SRTP processing including authentication plus integrity checking will not be performed, thus avoiding that overhead. As seen under the "Speedup" column of Table 1, for the first experimental system, the fake packet detection speedups for Schemes 1 and 2 are 3.8 and 16.5 respectively. In other words, Scheme 2 can handle approximately 16.5 times higher ingress traffic before the CPU gets overwhelmed compared to basic SRTP. From the example in Section 1, this would mean ingress traffic of up to 80Mbps.

When the device faces only legitimate traffic, a packet undergoes the SRTP+ authentication followed by the basic SRTP authentication and integrity checking. The total overhead consists of the SRTP+ overhead in addition to the

SRTP overhead. As seen under the "Overhead" column of Table 1, for the first experimental system, Schemes 1 and 2 consume 26% and 6% more CPU respectively. It is possible to minimize the overhead of SRTP+, while retaining its DoS speedup if the SRTP+ schemes can be turned on only if an attack is underway. Since, the expected ingress traffic at an RTP based device is well-known, a substantial deviation from the known behavior such as a sudden increase in ingress traffic rate is a solid indicator that an attack is in progress. This intrusion detection mechanism can be utilized to turn-on and turn-off SRTP+ schemes. A perfect, zero time detection mechanism will eliminate the overhead. In reality, a heuristic which strikes a balance between not being too aggressive to account for network induced traffic variations and not too passive to detect the start and end of an attack efficiently is needed. The design of such a heuristic is not within the scope of this paper.

4. Conclusion

The key point brought forth by this paper is that simple, practical techniques can be used to significantly enhance the DoS tolerance in SRTP. We presented two variations of adding a light-weight authentication mechanism on top of SRTP which are used to efficiently discard illegitimate packets in the face of an attack. The measurements show that with SRTP+, a device can handle malicious traffic, the rate of which is more than an order of magnitude higher than what can be handled by basic SRTP. Further, smart implementation of SRTP+, when combined with attack detection, can minimize the overhead incurred under normal operation.

References

- [1] M. Baugher, D. McGrew, E. Carrara, M. Naslund, and K. Norrman. The secure real-time transport protocol. IETF RFC 3711. <<http://www.ietf.org/rfc/rfc3711.txt?number=3711>>.
- [2] H. Krawczyk, M. Bellare, and R. Canetti. Hmac: Keyed-hashing for message authentication. IETF RFC 2104, Feb. 1997. <<http://www.ietf.org/rfc/rfc2104.txt?number=2104>>.
- [3] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. Rtp: A transport protocol for real-time applications. IETF RFC 3550, July 2003. <<http://www.ietf.org/rfc/rfc3550.txt?number=3550>>.