Securing QoS: Threats to RSVP Messages and Their Countermeasures*

Tsung-Li Wu
E.C.E. Department
North Carolina State University
Raleigh, NC 27695

S. Felix Wu
Zhi Fu, He Huang
Computer Science Department
North Carolina State University
Raleigh, NC 27695

Fengmin Gong
ANR
MCNC
RTP, NC 27709

ABSTRACT: In this paper, we study one type of *DoQoNS (Denial of Quality of Network Service)* attacks: **attacks directly on the resource reservation and setup protocol**. Particularly, we have studied and analyzed the RSVP protocol. Two contributions are: First, we performed a security analysis on RSVP which demonstrates the key vulnerabilities of its distributed resource reservation and setup process. Second, we proposed a new secure RSVP protocol, *SDS/CD (Selective Digital Signature with Conflict Detection)*, which combines the strength of *attack prevention* and *intrusion detection*. SDS/CD resolves a fundamental issue in network security: how to protect the integrity, in an End-to-End fashion, of a target object that is mutable along the route path. As a result, we will show that SDS/CD can deal with many *insider* attacks that can not be handled by the current IETF/RSVP security solution: *hop-by-hop Authentication*.

1 Introduction

Attacks on quality of network services (DoQoNS) can be classified into two different types: control flow attacks (*e.g.,* killer reservation in RSVP [1] [2]) and data flow attacks (e.g., resource stealing in RSVP/DiffServ [3]). First an intruder can attack directly on the signaling/control protocol for network resource reservation and connection setup, which results in the victim link conceptually unavailable to other users. Second, even if the resource reservation process is successful, the intruder can still attack the dataflow such that some or all the reserved resources are not spent appropriately.

Under the RSVP framework, some of the RSVP objects remain unchanged from sender to receiver (or from receiver to sender) under normal operation (e.g., `Tspec(`$PATH$`)`, `Filter_spec(`$RESV$`)`, or `Flow_spec(`$RESV$`)` under the unicast case), and others may be updated by the intermediate RSVP routers (e.g., `Adspec(`$PATH$`)`, or `Flow_spec(`$RESV$`)` under the multicast case). The integrities of these mutable RSVP objects are hard to protect, since they can be modified by a bad router the same way the legitimate router can.

*This work is supported by the U.S. Department of Defense Advanced Research Projects Agency under contract DABT63-97-C-0045.

In this paper, we focus ONLY on the first type of *DoQoNS (Denial of Quality of Network Service)* attacks: **attacks directly on the resource reservation and setup protocol**. we perform a security analysis on RSVP which demonstrates the key vulnerabilities of its distributed resource reservation and setup process. We also propose a new secure RSVP protocol, *SDS/CD (Selective Digital Signature with Conflict Detection)* for RSVP, which combines the strength of *attack prevention* and *intrusion detection*. The security strength and practicality of this new protocol are demonstrated. And, we will also show that SDS/CD can deal with many *insider* attacks that can not be handled by the current IETF/RSVP security solution: *hop-by-hop authentication* [4].

2 Attacker's Objectives and Classification

We consider four objectives for the RSVP QoS attackers, they are defined as follows.

1. Unnecessary/Suboptimal Reservation (**UR**): A user wants to reserve X units of resources but the attacker causes the system to reserve Y units, while X is significantly different from Y.

2. Degradation of Network Utilization (**DNU**): The network system, as a whole, have enough resources to support a set of QoS requests, S_{req}. But, the attacker can, for instance, interfere with the reservation protocol such that the network can only support a small subset of S_{req}, $S_{partial}$.

3. Reserved QoS Degradation (**RQoSD**): Even the resources along the path has been reserved and maintained successfully, attackers can use the reserved resource unauthorizedly.

4. Denial of QoS Request (**DQoSR**): The attacker can potentially intercept or drop all or some of the reservation messages such that the QoS reservation and channel setup can be failed or maliciously delayed in a persistent way.

For RSVP, we define three classes of attackers: $Insider_{RSVP}$, $Outsider_{OnPATH}$, and $Outsider_{Other}$. An $Insider_{RSVP}$ is an RSVP-enabled router on the reservation path between a sender and a receiver. $Outsider_{OnPATH}$, on the other hand, is an RSVP-disabled router also on the reservation path. And $Outsider_{Other}$ are routers or end-hosts which are not on the reservation path.

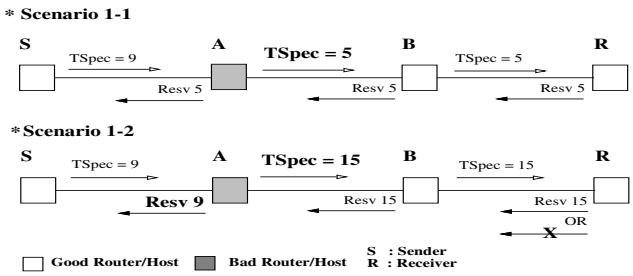

Figure 1. Insider attack, scenarios 1-1 and 1-2

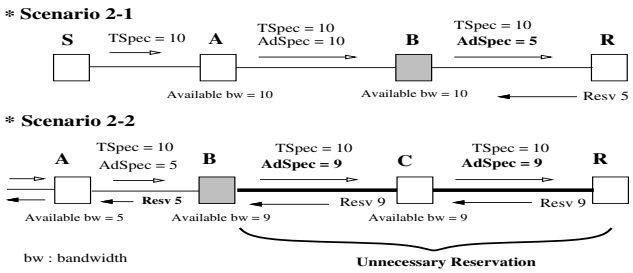

Figure 2. Insider attack, scenarios 2-1 and 2-2

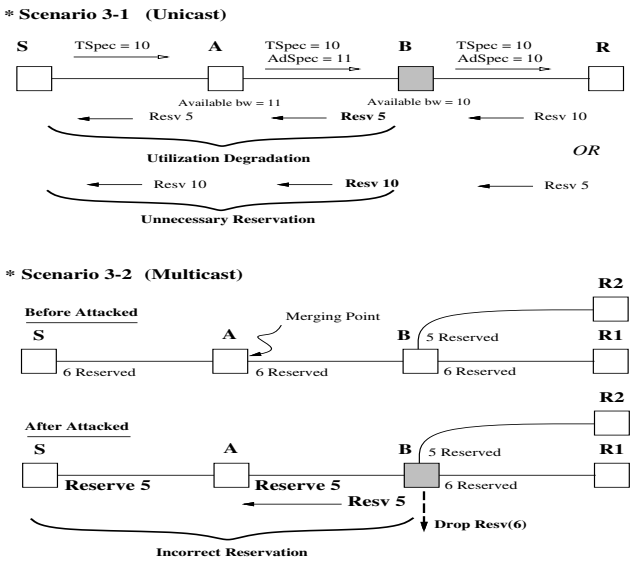

Figure 3. Insider attack, scenarios 3-1 and 3-2

3 Eight Attack Scenarios: The Problems

Some RSVP control message attacks performed by these three classes of attackers are presented in this section.

Scenarios 1-1 and 1-2, `Tspec(`$PATH$`)` Tampering Attack: A vicious $Insider_{RSVP}$ router can silently modify the `Tspec(`$PATH$`)` in the incoming PATH messages. The receiver could make a wrong decision in reservation due to the faulty `Tspec(`$PATH$`)` information. If `Tspec(`$PATH$`)` is modified to a lower value (*i.e.,* the attacker performs a `Lower` operation on `Tspec(`$PATH$`)`), the QoS that the receiver is going to enjoy might be degraded. On the other hand, if an attacker performs a `Higher` operation on `Tspec(`$PATH$`)`, then the receiver, depending on its local policy, may decide not to reserve the resources because the service looks more expensive (assuming some pricing mechanism is in place). In both cases, receivers cannot enjoy the QoS in a level that they should have while there are still plenty of resources available. Furthermore, an aggressive attacker could modify both `Tspec(`$PATH$`)` and `Rspec(`$RESV$`)` objects to trick the receiver to make a much bigger (but unnecessary) reservation. See Figure 1.

Scenarios 2-1 and 2-2, `Adspec(`$PATH$`)` Modification: The `Adspec(`$PATH$`)` object is passed to the local traffic control module for `Adspec(`$PATH$`)` update. Conceptually, the `Adspec(`$PATH$`)` value represents the lowest resource availability among all the RSVP-enabled routers on the path. A vicious RSVP-enabled router can modify the value of `Adspec(`$PATH$`)`, which implies that all the RSVP-enabled routers downstream may update the `Adspec(`$PATH$`)` incorrectly. Finally, the receiver may reserve a lower or higher level of QoS. See Figure 2.

Scenarios 3-1 and 3-2, see Figure 3, `Rspec(`$RESV$`)` and `Tspec(`$RESV$`)` modification: These two RESV parameters [5] are supposed to remain untouched unless a merging is needed while multicasting. A vicious RSVP-enabled router (or sometimes even an $Outsider_{OnPATH}$) increases or decreases them unlegitimately such that either the RSVP routers upstream make unnecessary reservation or QoS degradation in receiver could happen.

Fake `TearDown(`$RESV$`)`: An attacker can generate a fake `TearDown(`$RESV$`)` message to teardown other reservations, such that it may have a very good chance to reserve the resources. Finally, both RSVP signalling operations and dataflow can be damaged by an attacker performing dropping attacks.

4 Selective Digital Signature and Conflict Detection (SDS/CD): The Solution and Evaluation

In order to deal with $Insider_{RSVP}$ attacks, we propose to use a detection algorithm with modified end-to-end authentication. We separate the target RSVP objects into two different groups: *constant* or *mutable*. For the constant part of the RSVP messages, the source or originator of the target object (*e.g.,* $Sender_{Alice}$ or $Receiver_{Bob}$) will digitally sign the object with its own private key. This prevents any $Inside_{RSVP}$ from tampering with the signed objects.

The SDS/CD protocol is described step by step as follows.

1. $Sender_{Alice}$ selectively and digitally signs the `Tspec(`$PATH$`)` (constant) part of the PATH message with its private key.

2. Upon receiving a PATH message, $Receiver_{Bob}$ verifies the integrity of `Tspec(`$PATH$`)` using $Alice$'s public key, and then makes resources request accordingly.

3. $Receiver_{Bob}$ sends back RESV message **piggybacking** with, additionally, the `Adspec(`$PATH$`)` object it just received (historical `Adspec(`$PATH$`)`). Both the piggybacked

Adspec ($PATH$) and the Rspec ($RESV$) in the RESV message are selectively and digitally signed with Bob's private key.

4. Upon receiving the RESV message, the intermediate RSVP-enabled router $Router_{Chris}$ verifies if the piggybacked (and signed) Adspec ($PATH$) is equivalent to or lower than the last Adspec ($PATH$) value it forwarded downstream. If it is not, a conflict has been detected and Chris raises an alarm to its local policy decision point (PDP). The PDP will decide whether the RESV message should be dropped or not. Also, it will decide whether it is appropriate to flood a TearDown ($RESV$) message downstream.

In this case, the SDS/CD protocol at $Router_{Chris}$ detects that some vicious or faulty RSVP-enabled router downstream has reported an invalid Adspec ($PATH$) value toward one of the receivers.

5. At an intermediate RSVP-enabled router, if a merge from multiple RESV messages is necessary, the router picks the RESV message with the largest Rspec ($RESV$) value and forwards it toward $Sender_{Bob}$ (upstream).

6. When an RESV message arrives at $Sender_{Alice}$, the sender verifies if the Rspec ($RESV$) is signed by one of the legitimate receivers. If it is not, the reservation should be cancelled. And, if it has been signed by a legitimate receiver, then that particular receiver is accountable for this reservation request.

7. The Rspec ($RESV$) being received by the $Sender_{Alice}$ is "historical" (*i.e.,* no more merge should be performed). $Sender_{Alice}$ can digitally sign the Rspec ($RESV$) and piggybacks this signed historical Rspec ($RESV$) with the next refreshing PATH message.

8. Upon receiving the refresh PATH message, the intermediate RSVP-enabled router $Router_{Chris}$ checks if the piggybacked (and signed) Rspec ($RESV$) is lower than the last Rspec ($RESV$) value it forwarded upstream. If it is, a conflict has been detected and Chris will raise an alarm to its local policy decision point (PDP). The PDP will decide whether the PATH message should be dropped or not. Also, it will decide whether it is appropriate to flood a TearDown ($PATH$) message upstream.

In this case, the SDS/CD protocol at $Router_{Chris}$ detects that some vicious or faulty RSVP-enabled router upstream had chosen a smaller Rspec ($RESV$) value in a merge operation and forwarded to the sender.

For the evaluation, the following table shows the bad effects (the acronyms are defined as in section 2), which can be caused by the 8 attacks.

Attacks	1-1	1-2	2-1	2-2	3-1 & -2	4	dropping
UR		V		V	V		
DNU	V	V	V	V	V		
RQoSD					V	V	V
DQoSR						V	V

The next table summarizes that the effectiveness of *hop-by-hop authentication* vs. *SDS/CD* in handling 8 attacks. Please note that, in Attack 3-2 (Figure 3), even with the hop-by-hop authentication, an $Outsider_{OnPATH}$ can drop the $RESV(6)$ message from $R2$ to router B to achieve the same result as B (an RSVP-enabled router) is compromised.

Countermeasure	Hop-by-Hop	SDS/CD
Attack 1-1	$Outsider_{OnPATH}$	$Insider_{RSVP}$
Attack 1-2	$Outsider_{OnPATH}$	$Insider_{RSVP}$
Attack 2-1	$Outsider_{OnPATH}$	$Insider_{RSVP}$
Attack 2-2	$Outsider_{OnPATH}$	$Insider_{RSVP}$
Attack 3-1	$Outsider_{OnPATH}$	$Insider_{RSVP}$
Attack 3-2	$Outsider_{Other}$	$Insider_{RSVP}$
Attack 4	$Outsider_{OnPATH}$	$Insider_{RSVP}$
Dropping	$Outsider_{Other}$	$Outsider_{Other}$

5 Related Proposed Protocols and Remarks

[4] defined an INTEGRITY object to be carried in RSVP message in order to provide the information required for hop-by-hop integrity checking. Some concerns in implementing this mechanism include the overheads of performing at least two message digest computations per RSVP message in each router and exchanging RSVP Authentication Key regularly, and the limition of hop-by-hop authentication being unable to prevent insider attacks. For the 8 attacks we present to demonstrate the vulnerabilities of RSVP in supporting network QoS. [4] is only useful in preventing $Outsider_{OnPATH}$ (and in one case, only preventing $Outsider_{Other}$, which is the weakest attack model).

Our SDS/CD protocol addresses a fundamental issue in network security: *how can we protect, in an End-to-End fashion, the integrity of an object that is mutatable along the route path?* SDS/CD resolves this issue by delaying the integrity protection process until the target object becomes *history* and letting the intermediate RSVP-enabled routers, which could have modified the target object's value, to detect the conflicts. We showed that, under SDS/CD, many types of $Insider_{RSVP}$ attacks can be handled in a flexible, secure and efficient way.

SDS/CD still can not handle the RSVP message dropping attacks, which probably are not *preventable*. We believe that the combination of various prevention and detection schemes together will hopefully provide a strong solution in protecting network QoS.

Acknowledgments

Liang Li and Steve Nadas from IBM/RTP have generously provided us a beta implementation of RSVP running on IBM's 2210 routers, which we used to perform some of our experiments.

References

[1] L. Zhang, et al. RFC 2205, Sep. 1997.
[2] M. Talwar. RSVP Killer Reservations. Internet Draft, Nov. 1998.
[3] S. Black, et al. RFC2475, Dec. 1998
[4] F. Baker, B. Lindall, and M. Talwar. RSVP Cryptographic Authentication. Internet Draft, Nov. 1998.
[5] J. Wroclawski. RFC2210, Sep. 1997.