[RGKSt01] *Utz Roedig, Manuel Görtz, Martin Karsten, and Ralf Steinmetz;* **RSVP as Firewall Signalling Protocol;** Proceedings of the 6th IEEE Symposium on Computers and Communications, July 2001.

RSVP as Firewall Signalling Protocol

Utz Roedig[1], Manuel Görtz[1], Martin Karsten[1], Ralf Steinmetz[1,2]
[1] Industrial Process and System Communications, Darmstadt University of Technology, Germany
[1] German National Research Center for Information Technology, GMD IPSI, Darmstadt, Germany
Email: {Utz.Roedig|Manuel.Goertz|Martin.Karsten|Ralf.Steinmetz}@KOM.tu-darmstadt.de

Abstract

Within a global networked environment, security aspects have become more and more important and access control at network borders is considered essential. For this purpose firewall systems are used which provide a well-established security mechanism to restrict the exchanged traffic to a certain subset of users and applications. In order to cope with the increasing demand for new applications, a firewall must be flexible and extensible to support such new applications and their protocols. RSVP is a dynamic signalling protocol, which has been invented to negotiate resource requirements between end systems and a packet-based communication network. In this paper, we investigate the interoperation of RSVP with a firewall system in order to support new applications in a generic way. We show how the resulting system flexibility allows for a variety of employment scenarios and incremental deployment of such a technology. We back up our claims by describing a prototype that we have implemented.

1. Introduction

A firewall examines all network traffic between the connected networks. Only data that is explicitly allowed to, as specified by a security policy, is able to pass through [1]. In addition to the inspection of data flows, some firewalls also hide the internal network structure of an organization. From the Internet, the only visible and therefore attackable network system is the firewall. This is achieved by the use of Network Address Translation (NAT) [2] mechanisms. The appropriate treatment of the data that is demanded by the security policy, depend on the application and/or the protocols that are used. Therefore, the firewall has to be instructed on how the data has to be treated for each application. Two general problem domains have to be solved for each application, according to the basic functionality of a firewall:

- Opening and closing paths through the firewall based on security checks.
- Appropriate implementation of the NAT functionality.

The tasks of a firewall are well defined, but there are many possible firewall architectures to fulfil them. Firewalls may consist of packet filters, "stateful filters", proxies or a combination of all these components. In addition, the applications itself may interact explicitly with a firewall to support the firewall to fulfil their tasks.

RSVP, initially designed and described in [3], has been specified by the IETF [4] to carry reservation requests for packet-based, stateless network protocols such as IP (Internet Protocol). In the RSVP model, senders inform RSVP-capable routers and receivers about the possibility for reservation-based communication by advertising their services via PATH messages. PATH messages carry the sender's traffic specification (TSpec) and follow exactly the same path towards receivers as data packets. Receivers initiate reservations by replying with RESV messages. They contain a TSpec and a reservation specification (RSpec) and establish the reservation.

2. Motivation and Outline

An explicit interaction between applications and firewalls is necessary, when a general method to add support for new applications within a firewall needs to be available. Especially for the integration of complex and dynamic applications, like multimedia applications, such a general method is desired. To build the interaction between the application and the firewall normally a "firewall framework" is used. In this paper, we show that the existing RSVP architecture can be used to build such a firewall framework. The resulting firewall framework has not to be defined and implemented from scratch, but instead a lot of already existing work can be reused. To explain this fact in detail, we give a general model of a firewall framework. This allows us to show which parts of a firewall framework are already covered by RSVP and which parts have to be added or modified. As we show, modifications of RSVP are not necessary for basic scenarios. RSVP can be "used" as firewall framework.

The next section gives a definition of a firewall framework. In Section 4, we show how RSVP is used to build such a firewall framework. Afterwards, we discuss the necessary boundary condition, the security of RSVP itself. In Section 6, we clarify the functionality of our approach by describing the overall signalling process on the basis of an example application. Section 7 gives an overview of related work and links our new approach to previous work. We close the paper with a summary of the presented work.

3. Firewall Architectures

Before we define how RSVP can be used as generic firewall framework, we clarify the difference between a firewall using a framework and a firewall that does not. Then we identify which parts are necessary within such a firewall framework in general.

3.1 Definition of a Basic Firewall

A firewall has to support a set of basic tasks, to properly handle application communication. These basic tasks can be classified in the following manner:

T 1. Opening and closing paths through the firewall based on (security) checks.

- **T 1.1** The firewall checks which hosts are involved in the communication (e.g. by determining source and destination IP-addresses).
- **T 1.2** The firewall has to determine the service used (e.g. by analyzing the UDP or TCP ports).
- **T 1.3** The firewall has to know about the communication dependencies and has to adjust its configuration dynamically (e.g. relationship of several flows that form one session).
- **T 1.4** The firewall determines the users that are involved in the communication (e.g. by authentification of a user within a proxy).

T 2. Implementation of the NAT functionality

- **T 2.1** NAT is implemented by modifying the IP-Addresses in layer 3 and for TCP and UDP, the ports in layer 4.
- **T 2.2** For some applications, modifications of the higher layers are necessary. (e.g if IP-addresses are negotiated and submitted in high layers during communication).

In the next sections, we use the term *basic firewall* to refer to a firewall or firewall system which is capable to execute the tasks listed above. Not all of these tasks are performed by all types of firewall components (filters, stateful filters, proxies). For example a packet filter does not support the Task 1.4 and NAT. Standard firewalls however are able to execute all of the mentioned tasks. Some of these firewalls might provide additional tasks (e.g. virus checking) but these are not included in our definition of a *basic firewall*. Every firewall, using a framework or not, should support the tasks of a *basic firewall*.

3.2 Grouping of Logical Parts

For some of the described tasks of a *basic firewall* a firewall needs some information, that is normally not available to network nodes. Therefore, the firewall has to interact with the application or its protocols. Figure 1 shows the logical parts that can be grouped to build different levels of interaction between application and firewall.

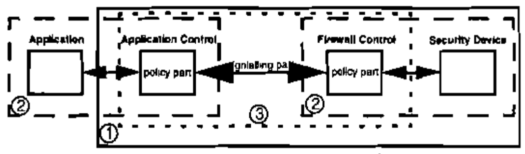

Figure 1 - logical parts of a firewall

The part *Application Control* is responsible to extract the information (e.g. owner of the data) from the applications communication that is necessary for the firewall. This information are then passed to the *Firewall Control*. The *Firewall Control* is responsible to handover this information to the security device (e.g. a packet filter device). The appearance of this information depend on the used security device type (e.g. commands to configure a specific packet filter device).

Combination ① in Figure 1 shows a "state of the art" firewall, where the application does not interact directly with the firewall to help the firewall performing its tasks. In this case, all logical parts are located on the firewall itself. The *Application Control* is designed as protocol parser within the firewall. The information extracted by the *Application Control* is then passed via function call to the *Firewall Control* which is also located on the firewall. The *Firewall Control* then executes the appropriate firewall configuration commands.

When a firewall framework ② is used, the *Application Control* is part of the application. The necessary information is passed by the application to the *Application Control*, so that a parser is not necessary. Then the information is passed to the *Firewall Control* which is located on the firewall. Since the application is running on a different machine than the firewall, a mechanism to transport the information over the network is needed. The *Firewall Control* then executes the appropriate firewall configuration commands.

It is also possible to group the logical parts in other ways. Combination ③ in Figure 1 shows an intermediate proxy between the application and the firewall.

The main difference between combination ① and ② is given by the generality of the approaches. In the first approach, the firewall needs to interoperate with a parser. This parser must have knowledge of the internal structure of the application protocols, and therefore a parser for each application is necessary. The first approach could be considered to be more secure, due to the granularity of the possible checks in the specialized parser and the missing interaction with other network components. In the second approach, the firewall support has to be integrated in the application, but the firewall has not to be modified for every new application. There is a trade off between security and generality, which has to be considered.

A firewall framework consists of the following two main parts:

- The transportation of the necessary information between applications and the firewall. We refer to this part of a framework as the *signalling-part*.
- The definition, how these informations are created in the applications (*Application Control*) and how these informations are used in the firewall (*Firewall Control*). We refer to this part of the framework as *policy-part*.

Both parts have to interact in a way that the resulting framework can handle the tasks of a *basic firewall*. The design of each part has an impact on the design of the other part.

3.3 Security Concerns

If a firewall framework is used to replace specialized firewall components, like an application level gateway, there is some functionality that could not be covered. An application level gateway or proxy compromises specialized parsers for the supported applications. These parsers are capable to perform security checks on the application layer. For example an FTP application level gateway can filter certain ftp commands like GET or PUT, an HTTP application level gateway can remove ActiveX or java applications from the transferred documents. There is a trade-off between "extended security features" and "general application support".

4. An RSVP Based Firewall Framework

Firewalls can use RSVP in two different ways. First, a firewall is also a router and therefore could use RSVP in the same way as normal QoS aware routers. In case that the flow's packets are not simply forwarded but checked by some application code on the firewall, the resources to be reserved are not only network resources but local components as well. Hence, mechanisms such as CPU scheduling and memory management must be considered [5]. This usage of RSVP is related to classical QoS which also considers the specific behavior of a security component. The details of network and local resource management are out of the scope of this paper. The second method, how a firewall can use RSVP, is to use it as signalling protocol for security QoS requests. A desired security level can be part of a QoS specification [5] as well. This method of using RSVP is discussed within this paper.

4.1 Definition

To avoid the need of implementing and installing a new parser everytime a new protocol has to cooperate with the firewall. we propose the "Firewall RSVP Framework". We suggest to use RSVP to transport the necessary information between applications and the firewall. RSVP is used in this framework to implement the signalling part and the policy part.

The application has to announce every upcoming stream by sending a standard RSVP PATH message which represents the *Application Control* in the *policy-part*. The message contains a description of the upcoming flow, e.g. protocol, source, destination. These information are used by the firewall to implement the appropriate mechanisms, representing the *Firewall Control* within the *policy-part*. Because these steps are performed for each flow, the firewall does not have to analyze flow dependencies or flow characteristics of an application session to support the tasks 1.1 to 1.3. Thereby, the *signalling-part* uses the standard RSVP signalling mechanisms, as they are also used for QoS purpose. We define this operation mode as the passive mode where existing information in the RSVP messages is interpreted for security purposes within the firewall. To be able to fulfil also the necessary Task 1.4, the signalling part has to use an *active mode*. Within the RSVP PATH message an additional object has to be inserted by the application, which contains information about the flow's owner. This can be achieved by using the already defined POLICY_DATA-object.

To match the NAT requirements specified in task 2. (Section 3.1), the *active mode* might be used as well, but this is not discussed within this paper.

As we have shown, two "operation modes" have to be distinguished. Both modes can be used within a RSVP firewall framework. When the *passive mode* is used, the standard RSVP signalling can be employed to control a firewall supporting Tasks 1.1 to 1.3. To control a firewall which supports all tasks of a *basic firewall*, the *active mode* has to be used. To use the *active mode*, the RSVP signalling has to be enhanced whieh is well feasible due to the modular design of RSVP.

This paper investigates both operation modes. We show how these two operation modes can be used to build a generic framework for the integration of new applications within firewalls. Thereby, we primarily focus on the firewall problem domain "Opening and closing paths". The problem domain "NAT functionality" is considered but not be discussed in detail in this paper.

4.2 Preconditions

If RSVP is used as a firewall framework, some preconditions have to be considered.

At least two of the involved nodes (one communication endpoint and the firewall) have to support the RSVP protocol. Compared to other frameworks, as described above, this results in the same implementation effort, but also additional features are available. If external entities (applications, devices,...) can influence the configuration of the firewall, a *policy overlay* is necessary. This means that an administrator of the firewall should be able to narrow the possible configuration changes that are signalled. For example, an administrator wants to be sure that negotiated flows are only enabled when they use not well-known ports. This ensures that even in a worst case scenario a basic firewall policy remains active. If RSVP is transporting security related information, which is true when the firewall configuration is based on the RSVP message content, it has to ensure that the protocol itself is secure. Because this is the most important precondition, we discuss this issue in detail in Section 5.

When these preconditions are fulfilled, the following mechanisms have to be implemented to enable RSVP for the targeted purpose.

4.3 Details

Figure 2 describes a simple scenario where an intranet is protected by a firewall which uses the RSVP framework. We refer to this scenario to describe the details of the framework.

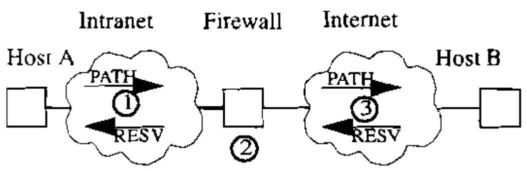

Figure 2 - RSVP session establishment

If Host A intends to initiate a flow to Host B it announces this upcoming flow by sending a PATH message to HOST B (①). The firewall, which is capable to take part in the RSVP signalling, processes and forwards the RSVP information to host B. Host B replies with a RESV message (③). This message is also processed by the firewall and forwarded to host A.

4.3.1 Policy Part - Application Control. The applications have to announce all flows that are used by RSVP. Therefore each application has to use the standard RSVP mechanisms. If the firewall also requires user authentification for the flows, the RSVP implementation also has to support the POLICY_DATA-object as described in [6]. Applications have to implement a standard RSVP signalling interface to provide the *Application Control*.

4.3.2 Policy Part - Firewall Control. To implement this part of the framework, the firewall has to take part in RSVP signalling. This could be achieved by adding a standard RSVP daemon to the firewall. The used RSVP daemon has to be slightly modified to be able to support the following capabilities:

If host A initiates the communication (as described above), the firewall uses the information in the PATH message to implement a filter rule for the upcoming flow. If host B initiates the communication, host B sends the PATH message and host A responds with the RESV message. In this case the information within the RESV message is used by the firewall to implement the appropriate filter rule. As described in Section 5, security mechanisms can be used to authenticate the generator of RSVP messages. Because of the hop by hop characteristic of RSVP, also the security mechanisms are hop based. Therefore the firewall can only trust RSVP messages that are passed by the host directly without passing an RSVP hop and are passed from a hop that is trusted by the firewall. These conditions can only be satisfied by signed messages, containing firewall signalling information, from internal and therefore trusted hosts.

Consequently either the PATH or the RESV message are used to update the firewall configuration, depending on who is the internal sender. A *ConnectionCache* is necessary in the firewall, to control if the internal host has confirmed the communication request. The *ConnectionCache* is a table which holds entries for each connection. It stores source and destination addresses and ports as well as the used protocol, the direction (internal or external) and the status of the connection. The RSVP implementation within the firewall has to distinguish internal and external generated messages

An RSVP message (PATH and RESV), generated by the application to announce an upcoming flow, usually has the appearance shown in Figure 3.

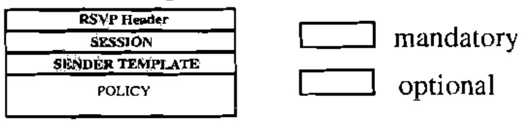

Figure 3 - RSVP message

The protocol number, the destination port and destination address are included in the SESSION object. The source port and source address are included in the SENDER_TEMPLATE object. Information about the user that is responsible for the flow is included in the POLICY_DATA object. The usage of the POLICY_DATA object is currently not available in most RSVP implementations, but it is defined within the RSVP standards. The necessary information is extracted from the RSVP message and processed by the firewall in the following manner:

- The information about the upcoming flow are extracted from the RSVP message.
- The informations are compared with the firewall security policy. The firewall checks, if the desired communication endpoints are valid and the user is authorized for communication. (Task 1.1,1.2,1.4 of the *basic firewall*. Task 1.3 is not performed by the firewall, this task is now performed by the application).
- The firewall checks whether the negotiated communication parameters match with the *policy overlay*.
- The firewall creates the filter rules using the information

from the RSVP packets. Depending on the used firewall type, several rules might be necessary to enable one flow.

Rules that are not needed anymore have to be removed properly. RSVP provides a soft state characteristic. The applications have to take care of the implemented QoS reservations. They have to refresh their reservations periodically by sending appropriate messages. If the refresh messages are not sent, the implemented reservations in the network nodes are removed. This feature is also used within the firewall to remove rules that are not needed anymore. Alternatively applications might send a TEAR message to remove the implemented QoS parameters explicitly. In this case, the TEAR message is used by the firewall to remove the rules.

4.3.3 Signalling Part. The signalling part is implemented by using an RSVP implementation (in the application and on the firewall) that supports basic security as defined in Section 5. The signalling part uses the standard RSVP signalling mechanisms.

4.4 Limitations and open Questions

The additional signalling between applications and the firewall brings up performance questions. It is shown in [7] that the RSVP signalling is performant enough to be used even for large number of concurrent sessions.

This paper does not cover the problem domain "NAT functionality", which has to be considered. For NAT functions, additional RSVP objects have to be designed to signal appropriate information between application and firewall. There are also other fields which have to be investigated, e.g. multicast, real-time operation, extensibility.

5. Security Mechanisms in RSVP

Security can be divided into three major issues that can be distinguished and for all of them solutions within RSVP exist.

- **Authentication:** In [8] an identity representation for RSVP is introduced. This information is placed in the POLICY_DATA object.
- **Integrity:** In [9] the optional INTEGRITY object is defined, which carries information to ensure the integrity of the RSVP message.
- **Confidentiality:** IPsec [10] can be used in conjunction with RSVP to provide confidentiality.

Authentication and Integrity must be provided to be able to perform the *basic firewall* Tasks (Section 3.1). Confidentiality is not necessary, because an intruder can either look at the RSVP message content or the subsequent communication. Communication relations can be determined by an intruder anyway. If the POLICY_DATA object and the INTEGRITY object is used, RSVP can be considered secure and capable to carry information to configure a firewall. A more detailed analysis of the security mechanisms in RSVP regarding firewall control can be found in [11].

6. Communication Example

To make the previous described concept more vivid, a real-world examples is explained in this section. In the chosen example, RealAudio reflects a currently often used application.

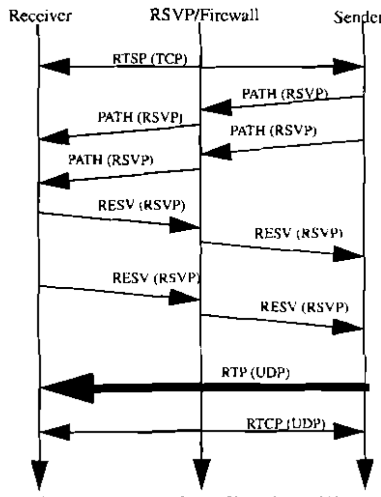

Figure 4 - RealAudio signalling

The signalling between a client, the firewall and a server for a RealAudio communication is shown in Figure 4 and described below. The communication starts with an RTSP [12] connection from the client to the well-known port (554) on the server. This connection is used to negotiate dynamic parameters, e.g. ports, for the upcoming media data transmission using RTCP/RTP [13] over UDP. With our suggestion to use RSVP for signalling, the following situation arise:

- The receiver establishes a RTSP connection via TCP to the server through the firewall, using a predefined rule.
- After negotiating the parameters for the media streams, the sender initiates the data transmission. For this purpose the sender sends two PATH messages.
- The informations in the PATH messages are inserted in the *ConnectionCache* (untrusted external sender).
- The receiver replies with two RESV messages.
- The firewall receives and extracts sender and receiver port and address and the underlying protocol from the message. If the POLICY_DATA object is present, the user can be identified as well.
- The integrity of the message is checked as well as the authentication. If the verification succeeds, a new rule is created and the parameters for the connection are updated in the *ConnectionCache*.
- The arriving UDP streams match now the criteria in the rulesets from the firewall.
- After closing the RealAudio session, the specific firewall rules time out or are removed (TEAR message).

The use of RSVP allows to get all necessary information for the creation, installation and deletion of temporary firewall rules.

7. Related Work

There are some frameworks available, that can be used to handle applications within a firewall in a generic way. These frameworks are based on a signalling between the applications and the firewall. Some of these frameworks are currently investigated and proposed in the standardization organizations [16] [17] [18]. An other well known method, SOCKS is described in [15]. Most of these frameworks try to cover both firewall problem domains ("Opening and closing paths" and "NAT functionality"), which makes sense because these problem domains mostly show up together and are related to the same physical device.

Our approach is also based on a signalling between the communication endpoints (applieations) and the firewall. The differences between our approach and the described existing solutions are the following. An existing protocol is used, namely RSVP. In the *passive mode*, information that is already available is used. No additional effort at the client side is necessary to solve the firewall problem domain "Opening and closing paths". Our approach is more general, because it is not application specific and can deal with "modern" applications. Our approach is more integrating, because two related problems can be solved (QoS and QoP) using one framework.

8. Conclusion

We have shown that RSVP can be used as generic firewall framework. Compared to other existing frameworks this approach is based on already established and used methods. By combining both approaches, RSVP signalling and explicit firewall control, both benefit. Firewalls will benefit if they are able to provide a generic interface, that is well known and is already accepted by application developers. RSVP benefits by providing an additional feature and becoming thereby a more integrating signalling approach. Security can be seen as a QoS parameter and it is sound to use the same protocol to signal all QoS parameters. We have shown that RSVP already provides a lot of mechanisms that are also necessary to build a firewall framework. Few extensions to RSVP are necessary to also cover the task "firewall control".

9. References

[1] W. Cheswiek and S. Bellovin. Firewalls and Internet Security, Addison Wesley, 1994.

[2] K. Evang and P. Franci. RFC 1631: The IP Network Address Translator (NAT) Status: INFORMATIONAL, 1994

[3] L. Zhang, S. Deering, D., S. Shenker and D.Zappala. RSVP-A New Resource ReSerVation Protocol. IEEE Network Magazine, 7(5):8–18, September 1993.

[4] R. Braden, L. Zhang, S. Berson, S. Herzog and S. Jamin. RFC 2205: Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification. Standards Track RFC, September 1997.

[5] R. Steinmetz and L. Wolf. Quality of Service: Where are We?. In Proceedings of the IFIP Fifth International Workshop on Quality of Service (IWQOS '97), New York, NY, USA, May 1997.

[6] S. Herzog. RFC 2750: RSVP Extensions for Policy Control. Standards Track RFC, January 2000.

[7] M. Karsten, J. Schmitt and R. Steinmetz. Implementation and Evaluation of the KOM RSVP Engine. In Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'2001). IEEE, April 2001. To appear.

[8] S. Yadav, R. Yavatkar, R. Pabbati, P. Ford, T. Moore and S. Herzoh. RFC 2752: Identity Representation for RSVP. Standards Track RFC, January 2000.

[9] F. Baker, B. Lindell and M. Talwar. RFC 2747: RSVP Cryptographic Authentication. Standards Track RFC, January 2000.

[10] S. Kent, R. Atkinson. RFC 2401: Security Architecture for the Internet Protocol. Standards Track RFC, November 1998.

[11] U. Roedig, M. Görtz, M. Karsten, R. Steinmetz. RSVP as Firewall Signalling Protocol. Technieal Report 6, KOM, December 2000.

[12] H. Schulzrinne, A. Rao, R. Lanphier. RFC 2326: Real Time Streaming Protocol (RTSP). Standards Track RFC, April 1998.

[13] H. Schulzrinne, S. Casner, R. Frederick, V. Jacobson. RFC 1889: RTP: A Transport Protocol for Real-Time Applications. Standards Track RFC, January 1996.

[14] U. Roedig, R. Ackermann, C. Rensing, and R. Steinmetz. A Distributed Firewall for Multimedia Applications. In Proceedings of the Workshop "Sicherheit in Mediendaten", September 2000.

[15] M. Leeeh, M. Ganis, Y. Lce, R. Kuris, D. Koblas and L. Jones. RFC 1928: SOCKS Protocol Version 5, Standards Track RFC, March 1996.

[16] J. Kuthan and J. Rosenberg. Fircwall Control Protocol Framework and Requirements. draft-kuthan-fcp-01.txt, June 2000.

[17] S. Mercer, A. Moilitor, M. Hurry and T. Ngo. Internet Draft draft-rfced-inf-mercer-00.txt, H.323 Firewall Control Interface (HFCI). June 1999.

[18] TIPHON Working Group.VoIP Firewall Control Protocol (FCP), 2000. http://www.etsi.org/tiphon/.