Path-coupled signaling for NAT/Firewall traversal

Miquel Martin*, Marcus Brunner*, Martin Stiemerling*, Ali Fessi†
*NEC Europe Ltd. - {miquel,brunner,stiemerling}@netlab.nec.de
†University of Tuebingen; Work performed at NEC Europe Ltd. - fessi@netlab.nec.de

***Abstract*— Complex protocols tend to negotiate secondary flows on the application layer. This, in the general case, prevents Firewalls and NATs from allowing or routing them, and communication becomes impossible. In this paper, we describe the requirements and design of an end-application triggered, path-coupled signaling protocol for NAT/Firewall traversal. Finally, we show and discuss a performance evaluation, based on our implementation of the protocol.**

I. Introduction

As protocols and networks continue to grow in complexity, certain obstacles still represent a hindrance. In this paper, we deal with Firewall access control and Network Address Translation (NAT) issues for those protocols that carry flow information above the network and transport layer.

Let us take a protocol such as SIP as an example. During a voice call establishment, the peers will offer each other an IP address and port number where audio is expected[7]. Because of the multi flow nature of the protocol, this information is not deterministic, and therefore can not be pre-configured on any Firewall or NAT (from now on, a middlebox). As a consequence, the "unexpected" audio stream that will follow is likely to be dropped at the Firewall, or be deemed unroutable through a NAT. Note that this is not SIP specific, but rather common to any protocol that negotiates secondary flows in the application layer. The trap is thus set: a middlebox in its basic form, being aware of the network and transport layers alone, cannot configure itself according to application layer information.

A number of solutions exist to overcome this problem, each with its own set of pros and cons. Two of the most important are *Application Level Firewalls and NATs* and the *Midcom*[1] protocol. In the first, a module is installed into the Firewall to make it application layer aware. In spite of its simplicity, this solution is specific to each application protocol, and is therefore not generic. In the latter, a Midcom agent remotely configures the middleboxes according to the application layer protocol. This requires updated knowledge on the network topology about the data path, as well as full control on the behavior of all the involved middleboxes.

The root of the problem is therefore, the inclusion of flow information in unprocessed layers, and that, with a per protocol variability. The aim of this work is to provide a robust and complete solution that enables nowadays complex protocols to work through increasingly complex network topologies. To this end, we propose the use of a unique, end-application initiated, path-coupled signaling protocol.

For such a solution to be viable, standardization is a must. With this in mind, this work has been shaped around the framework of the IETF NSIS Working group[2], [3]. In this context, we have identified the problems involved in middlebox signaling, designed[4] and implemented a working prototype, which is currently under standardization within the NSIS WG.

II. Problem Description

The problems of multi flow protocols that negotiate traffic flows in the application layer have already been outlined in Section I. In this section we will work out the specifics.

A. Firewall access for negotiated flows

The rule of thumb in Firewall design involves initially closing all accesses and subsequently opening them on a per need basis. This, of course, cannot be done if the flow information is not available at configuration time, as is the case for negotiated flows. Therefore, what we consider a secure Firewall, will block protocols of these characteristics.

B. NAT routability

A Network Address Translator[5] acts as the middleman between two address spaces, translating back and forth as required. Usually, one of the sides of the NAT is the publicly reachable Internet, and the other is a private network. The addresses used in the private side[6] cannot be reached directly from the outside, and so, the NAT needs to keep a state for the connections initiated there. This allows the replies to be routed back in.

This means that the initiator of any traffic flow must be located within the private network. The reason is that traffic for that private network will all be addressed to the NATs public address, and state will be the only way to route it inwards. This brings up two problems: first, the hosts in the private network might offer the private IP as expected target for the data stream, and the public side host would be unable to route it there. Second, even if the first problem was solved, the NAT is unable to decide who within the private network should receive the stream.

C. Network topology issues

Given the complexity of nowadays networks, any number of middleboxes in any order could be traversed for any end-to-end communication. This makes it difficult to have some sort of controller signaling all the appropriate middleboxes, since this would require knowing all the routes in the Internet, and the state of each of the machines.

Moreover, different flows might follow different paths. If we try to signal all the middleboxes in the path from A to B, our signaling might travel a different path than the data, and so, what configurations were prepared would be useless. For instance, SIP traffic often does not follow the same path as the audio stream.

Even for the same kind of traffic, the path from A to B is not necessarily that from B to A (routing asymmetry), which potentially means that a different number of middleboxes should be signaled, some on the forward and some in the backwards direction, as seen in Figure 1.

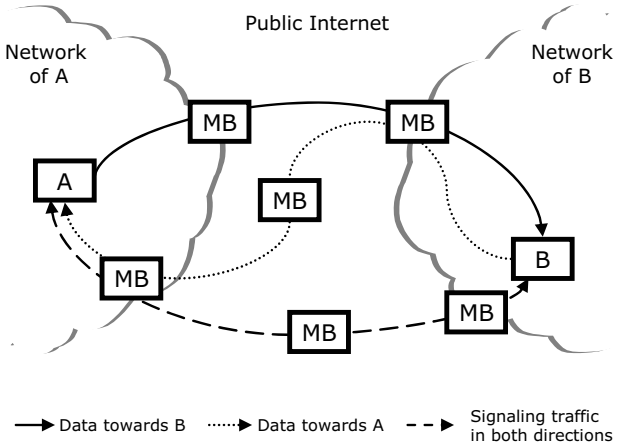

Fig. 1. routing asymetry example

D. Middleboxes and Application layer information

Unless all middleboxes on the data path understand each of the application level protocols, communication cannot be successful. The stress here is on the necessity to take design and implementation actions *for each application layer protocol*.

Because of the great number of such protocols, standardization would be a tall order, and human effort should be greatly divided, even though the necessities of all the protocols are basically the same: routing or access granting of a certain traffic flow.

III. Path-coupled signaling approach

A working solution to the aforementioned problems calls for a unified signaling protocol to configure middleboxes independently of the specific application signaling protocol.

In such scheme, the burden of initiating such signaling would befall the application, since it is there that the precise flow routing/access necessities are known. With application specificity thus solved, standard, well proven security mechanisms could be built into this unified protocol.

Topology issues could be addressed by ensuring the signaling protocol travel the same path as the data. This path-coupled approach would assure that signaling reaches all the necessary boxes.

Finally, middleboxes would need to become aware of *one and only one* of these application layer protocols: the unified signaling protocol.

Within the NSIS framework, all these requirements can be met by creating a NAT/Firewall service layer protocol (NSLP) traveling on top of a common NSIS transport layer (NTLP)[3]. Prior work on path-coupled signaling had been performed in TIST[8].

A. Protocol operations

This NAT/Firewall NSLP, as described here, must basically fulfill two functions:

- Allow a given flow[1] to traverse a Firewall.
- Install a forwarding state for a given flow in a NAT.

1) Firewall Scenario: The case of Firewall traversal is rather straightforward. A properly secured signaling message must be sent prior to the data. This message instructs the Firewalls along the path to allow the data flow to traverse the Firewall. Note that the signaling traffic itself must be allowed to traverse in all cases.

Once this state is installed in the Firewalls along the path, the data will be able to traverse end to end without obstacles.

2) NAT Scenario: The NAT scenario is more complex. Sending a message along with the data is not enough, since there is no way to point to a host in a private network with the current IP addressing scheme.

Therefore, prior action on the private side is required. To this end, the host that expects to receive a data stream must signal towards the data source to inform the NATs on the way that if a given flow is received, it should be forwarded towards it. This provides the data receiver with a publicly reachable address, which is to be offered as destination for the data flow.

What remains is analogous to the Firewall case, except that the flow information in the signaling message (now, from the data source to the data destination) is used to locate the previously installed forwarding state, and both the signaling and the data message are forwarded inwards into the private network accordingly.

Note that the problem of routing asymmetry does not apply to the first forwarding state installation, since the data is forced to travel through that particular NAT (route pinning).

B. On path mechanism

For this to work, the signaling traffic must travel along the same path as the data traffic. Within the NSIS framework, such signaling is expected to travel on a common transport layer (NTLP), which should be handled as data in each routing point. This will already be the default in most cases, and in the exceptions, a rule might have to be set accordingly.

In the case of load balancing systems, for instance, the signaling messages might be statistically sent on a different path, in which case the application would have to signal again and potentially keep as many paths as simultaneously open branches in the load balancing system.

[1] A flow is defined at least by source and destination IP addresses and ports

C. Scalability and stability

A soft state mechanism is in place to tear down a path[2] after a period of inactivity. To guarantee this, refresh messages are sent periodically, and the actual tear down occurs when a timeout since the last refresh message occurs, or when an explicit tear down message is sent.

The refresh interval is negotiated in an IPv6 MTU discovery mechanism fashion, with an initial value set and decreased if necessary by each of the middleboxes on the path. It is important to bear in mind that cooperation from *all* of the said middleboxes is required for the communication to succeed.

D. Topology independence

The protocol must work in exactly the same way, independently of whether NATs, Firewalls or both, are present. This is to comply with the requirement that the end hosts must have no prior knowledge of the network they are in.

For this reason, the protocol sequence is modeled in such a way that hosts behave always in exactly the same fashion. This is as follows

1) The data receiver host sends a reservation message in the direction of the data sender. All the NATs in the data path, if any, remember the reservation.
2) The last NAT[3], replies to the data receiver with a message containing its outermost address and port allocated in the reply, which is publicly reachable.
3) The data receiver awaits the reply message to a certain timeout. It then offers the allocated address or its own address as destination for the data.
4) The data sender sends an state creation message towards the data destination address. This packet traverses all Firewalls installing the necessary packet filters, and activates all the NAT Forwarding states.
5) The data receiver sends an acknowledge message back to the data sender.
6) The data sender sends the data to the data destination. This path is now cleared and the communication succeeds.

This behaviour is shown in Figure 2.

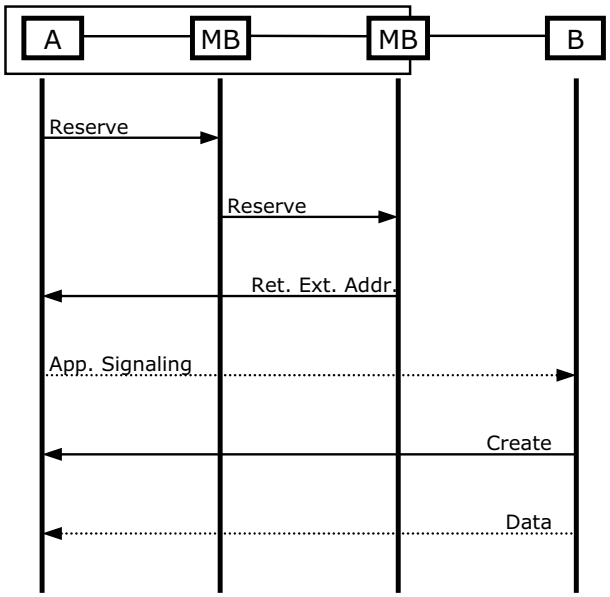

Fig. 2. Signaling flow example

E. Security considerations

While Firewalls have a clear security functionality, one must not forget that NATs are also used to obscure network topology, which is sometimes regarded as an extra security feature. Since the aim of this signaling is that of traversing such security mechanisms, this must be done in a controlled, secure way.

A two-layer security approach is thus required. On the top layer, each middlebox must have a policy based decision engine, which has the final word in state installation. Such policies can be based on such varied concepts as time of the day (office hour limited access), characteristics of the flow, safe port ranges or Firewall load.

On the lower layer, the protocol must provide the parameters required by the decision engine. For this reason, the protocol must ensure the integrity of the signaling packets. Possible access lists and accounting mechanisms will also require authentication and non-repudiation. Finally, note that privacy is out of scope, since the only information contained within the packets is flow information. Any attacker that can eavesdrop the signaling packet will also be able to see the flows en-route, and so, no new information is discovered. Even though a race-condition vulnerability might arise, it is up to the application layer protocol to secure itself. These requirements present a number of inherent problems, namely, the need for public key cryptography in the absence of a PKI and the on-path mutable data.

Finally, if separate efforts are done for each application protocol, the security of the system will be that of the weakest of them all. A unified protocol, allows for a join effort at security design and test.

1) Absence of a PKI: The security requirements call for a public key cryptography approach. This is no easy task without a proper infrastructure for key exchange and verification. For that reason, we have to work on probable trusts, such us a middlebox trust in the machines within its network. Such approach is far from complete, and so, information on the integrity of the received data must be passed over to the decision engine, which will then issue a verdict. Lower sensitivity operations, such as Access to standard VoIP port ranges, for instance, might be granted without further verification in a demilitarized zone.

[2]A path is understood here as the related Firewall rules and NAT forwarding states along the data path

[3]The last NAT is that which knows that its outside interface is publicly reachable

2) Mutable data: The need for data integrity collides with the presence of mutable fields in the signaling packets. A flow description whose path we want to open, will change as it traverses a NAT, and so, the signaling packet must be updated to reflect the address translation. This means that sensitive information will be altered on-path, and any previous signatures of this information will become invalid. For this reason, trusting the information on the source of a flow, once this traverses a NAT, requires trust in the said NAT.

The security mechanisms in the protocol are thus a pending issue. Yet another reason for this is the fact that the transport layer for NSIS is still in definition phase at the time of writing, and information on the security mechanisms that it will provide is unclear.

IV. Evaluation

The path-coupled protocol has been implemented in C on a Linux platform, using the Netfilter architecture[9] as a Firewall and NAT implementation, and libIPQ as a packet reader/injector. Given the unavailability of a NTLP definition, the protocol runs over raw IP packets.

We performed tests concerning correctness with SIP. The test bed included two SIP phones and several middleboxes in-between implementing different functionalities (Firewall or NAT). Because the SIP Phones are unable to signal NSIS off the box, an NSIS gateway was implemented, that understands SIP packets and generates the appropriate NSIS signaling messages. When path-coupled signaling is enabled, the middleboxes along the path are successfully configured to allow the traversal of the RTP flows between the SIP phones. The phone calls succeed in all the scenarios.

Besides, we conducted performance tests to verify the efficiency of path-coupled signaling in large networks. The test bed included two PCs, A and B, where A initiates signaling towards B, and a third PC in the middle with a middlebox functionality. The middlebox intercepts the signaling messages from A to B and processes them. The processing includes fetching the signaling message from the IP stack, parsing it, creating a signaling session with a session-ID[4] and installing the packet filters as requested in the signaling message.

Figure 3 shows the time required to process the signaling messages depending on the number of the signaling sessions already running. Since this delay depends heavily on the implementation of the packet filter, we realized tests both with and without installing packet filters (normal and dry-run plots) to evaluate the performance of the path-coupled signaling approach independently of the underlying packet filter/NAT implementation.

Both graphs show linear behavior, since both the signaling processing and the Netfilter Firewall implementation use linked lists to store the packet filters. The normal plot (installing packet filters) shows a moderate performance due to the non-optimized insertion of packet filters in the Linux Netfilter implementation. Nevertheless, the dry-run (without installing packet filters) shows a good performance. On the test machine, for up to 4,000 sessions, the delay does not exceed 1ms.

[4]The session-ID can be used for future control operations e.g. refresh

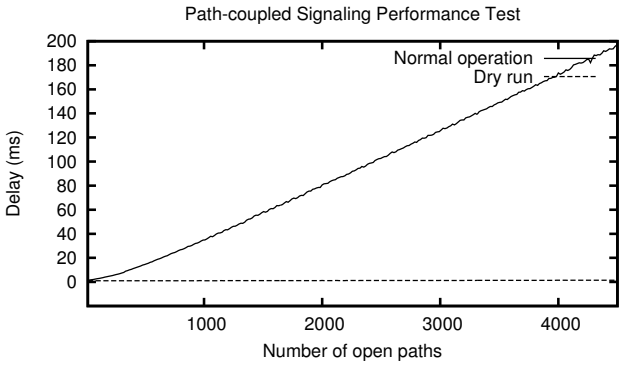

Fig. 3. Performance on a Pentium 333MHz, 256MB RAM

We realized a number of other performance tests with different approaches for the middlebox configuration problem such as the Application Level Firewawll and the Midcom approaches described in section I, Figure 4 shows the performance measurements results for the different approaches. The graphs show that the path-coupled approach is the most efficient one for middlebox configuration in large networks.

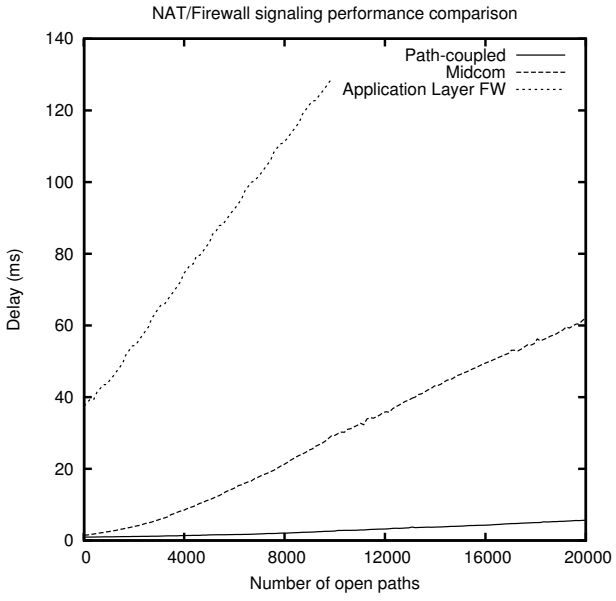

Fig. 4. Simulation results on a Pentium 333MHz, 256MB RAM

V. Related work

In the area of signaling protocols, the path-coupled approach followed by this design resembles that of RSVP. Still, RSVP, initially designed for QoS, cannot be easily shaped to perform the aforementioned tasks, and introduces complex issues, such

as multicast support, which are not required for this kind of signaling.

On the Firewall/NAT signaling area, Application Level Firewalls and Midcom offer simpler solutions, at the cost of not solving a number of important scenarios. Finally, long-standing protocols such as SOCKS[10] are also end application initiated, but do not account for middleboxes beyond the network boundaries known to the end application.

VI. Conclusion and future work

In this paper we have listed the problems of current NAT/Firewall control protocols and their increasing problems with current and future networks. The increasing complexity of both protocols and network topologies calls for a unified middlebox signaling protocol. The requirements outlined in this document show that a path-coupled signaling protocol for NAT/Firewall traversal is a sound proposal, which enables communication in all problem scenarios, even in scenarios where other protocols fail. The performed tests show the extensibility and efficiency of the implemented solution. A drawback compared with the other solutions is that both types, end hosts and middleboxes, must be changed to run this path-coupled protocol. Future work deals with migration aspects for the introduction of this new protocol.

In the future, work is left is in the standardization process, especially in including the feedback of peer reviews. At least as important an issue is the remaining work in security, where attack scenarios and its countermeasures have to be thoroughly accounted for.

This work is partially funded by EU IST FP6 Project ENTHRONE No. 507637.

References

[1] P. Srisuresh et. al., *Middlebox communication architecture and framework*, Request for Comments: 3303, August 2002.

[2] R. Hancock et. al., *Next Steps in Signaling: Framework*, Work in progress: draft-ietf-nsis-fw-06, June 2005.

[3] H. Schulzrinne and R. Hancock, *GIMPS: General Internet Messaging Protocol for Signaling*, Work in progress: draft-ietf-nsis-ntlp-03, July 2005.

[4] M. Stiemerling, H. Tschofenig, M. Martin and C. Aoun, *NAT/Firewall NSIS Signaling Layer Protocol (NSLP)*, Work in progress: draft-ietf-nsis-nslp-natfw-03, July 2004.

[5] P. Srisuresh and M. Holdrege, *IP Network Address Translator (NAT) Terminology and Considerations*, Request for Comments: 2663, August 1999.

[6] Y. Rekhter et. al., *Address Allocation for Private Internets*, Request for Comments: 1918, February 1996.

[7] J. Rosenberg, H. Schulzrinne et. al., *SIP: Session Initiation Protocol*, Request for Comments: 3261, June 2002.

[8] Shore, M., *The TIST (Topology-Insensitive Service Traversal) Protocol*, Work in progress: draft-shore-tist-prot-00.txt, May 2002.

[9] Rusty Rusell and Harald Welte, *Linux netfilter Hacking HOWTO*, http://www.netfilter.org/documentation/HOWTO/netfilter-hacking-HOWTO.html, February 2002.

[10] M. Leech et. al., *SOCKS Protocol Version 5*, Request for Comments: 1928, March 1996.