Enabling Session Initiation in the Presence of Middleboxes

Sergio Lembo
Telecommunications and Multimedia Laboratory
Helsinki University of Technology
P.O.Box 5400
FI-02015 Espoo FINLAND
first.last at tml.hut.fi

Jani Heikkinen
Telecommunications and Multimedia Laboratory
Helsinki University of Technology
P.O.Box 5400
FI-02015 Espoo FINLAND
first.last at tml.hut.fi

ABSTRACT

This paper presents the design and implementation of a SIP proxy that enables inbound session initiation in the presence of network address translators. Furthermore, the implementation is used to evaluate a recently proposed IETF Internet-Draft. As a result, two problems were identified. This paper proposes a solution for these problems.

Categories and Subject Descriptors

C.2 [**COMPUTER- COMMUNICATION NETWORKS**]: Network Protocols

Keywords

SIP, NAT, Multiprocess Architecture, TLS, DTLS

1. INTRODUCTION

In the recent past, a significant amount of research has been concentrating on the network address translation (NAT) traversal issues. The usage of private address spaces and thus the existence of NATs introduces several problems in the face of inbound requests among several Internet protocols. These problems concern the security, performance, and availability domains. Moreover, in the wireless domains, the last hop of the request path is typically over untrusted, limited bandwidth radio link.

In the case of Session Initiation Protocol (SIP), our preliminary research showed that the recently proposed IETF SIP Outbound Internet- Draft [1] aimed to solve a number of security and availability problems by extending RFC3261. The draft proposes to create authenticated flows between an User Agent Client (UAC) and a proxy during the SIP registration phase. Flows remain in time by using keep-alives sent over the same flow from the UAC to the proxy.

Thus, the contribution of this paper is to evaluate and analyze the proposed draft by designing and implementing a SIP proxy conforming to the draft. During this process, we identified two major problems which can make a SIP request or a response to be forwarded to a wrong client. In this paper, we propose solutions for these problems.

The structure of this paper is as follows. Section 2 presents briefly the design of the proxy and discusses flows, flow tokens, challenges, and solutions to them. Section 3 describes in more detail the implementation aspects.

*CoNEXT'*07, December 10-13, 2007, New York, NY, U.S.A.

2. DESIGN

One of the requirements of the SIP outbound Internet-Draft is to allow incoming SIP requests to be forwarded to clients behind a NAT. This is enabled by the proxy through the reuse of client-initiated flows and flow-tokens.

2.1 Flows

In this paper, we call the connections of the connection-oriented transport protocols or bidirectional streams of datagrams of connectionless transport protocols simply as flows. A flow is created by a client through registration procedure which includes user authentication. Hence, the flow is implicitly authenticated since a confidential, integrity protected flow-token signals successful authentication of the user that is authorized to use a particular SIP address of record.

An incoming request to a client is initially routed to a particular proxy via the SIP routing mechanisms. At the proxy, a flow to the client is identified by a flow-token. The flow-token will be present in future incoming requests. Thus, these simple, integrity protected tokens are unique identifiers that allow forwarding the request over the correct flow to the recipient. The client will beforehand create several flows to increase the availability of the client in the case of flow failure.

According to the Internet-Draft, the flow-token must contain information to uniquely identify a flow. This information can include the transport protocol, and source and destination IP addresses, and port numbers.

2.2 Challenges

Even when a flow-token identifies uniquely a flow, it must also contain enough information to validate the flow in order to prevent the request to be forwarded to a wrong client, for example, after a flow failure.

A request can be forwarded to a wrong client in a case when the flow for the first client terminates and a new client forms a similar flow with the same origin IP address and port. Similarly, SIP responses are normally routed to clients through the use of Via headers. This header does not contain enough information to verify the correct recipient of the response by the proxy. From the security point of view, IP addresses as identifiers can not be trusted in any case.

2.3 Solutions

Although the draft favors flow-tokens avoiding a state, protocols like TCP, TLS, and DTLS (RFC4347) introduce state in terms of connections or security associations. Nevertheless, a solution to previous problems is to generate a random such that the combination of an IP, port and the random creates a unique identifier.

In order to find a correct flow for an incoming request, the proxy can use the IP and port stored in the flow-token of the request. Furthermore, the proxy utilizes the random to verify that the flow in question is the correct one and not a flow that unluckily appeared later in time with the same IP and port.

The second problem was related to the SIP responses and the lack of enough information in the Via header to find and verify the correct flow. A Via header is composed and added to the message when the proxy forwards a SIP request and it is later present in the response for such request. A proposed solution for this problem consists on adding a flow-token in the URI part of the Via header. Responses over a flow will use the flow-token information to forward the response to the destination.

3. IMPLEMENTATION

We considered the solutions discussed above and implemented an experimental proxy that follows the requirements specified in SIP Outbound Internet-Draft.

The design of the proxy follows a multiprocess architecture in which each process will be in charge to maintain its own, dedicated, flow. Thus, we distribute the load of the processing among several processes and isolate flow failures to them. By creating one process for every flow, we are also able to establish a one-to-one relationship between flow-tokens and processes.

The actual relationship between flow-tokens and processes is through a local socket listening in every process. The local socket is identified by the same flow-token identifying the flow.

When a SIP message containing a flow-token arrives to the proxy, first we use the flow token to establish a connection to the listening local socket. In this way we are able to reach the process in charge of the desired flow. Then, once in this process, we verify that the random retrieved from the flow-token and the random stored in the process match. In a case of success, we validate that the desired flow is the correct one and proceed to forward the SIP message over the flow.

3.1 TCP-TLS and DTLS Flows

We have also implemented transport layer security support for the proxy. In the case of TCP or TLS flows, a main process will handle the initial connection and immediately create a child process that will be in charge of the flow.

As mentioned above, besides the connection handling for the flow, the child process also contains a local socket used to reach the process itself. Thus, there are two possible events. First, a message arriving over the flow: it will be received and sent to the application layer where it will be processed. Secondly, a message arriving over the local socket: such message is the result of a SIP request arriving at the proxy, with destination over the flow and containing a flow-token. In addition, there could be an event due to expiration of a countdown timer. If the process does not receive a suitable keep-alive message, a countdown timer will expire and cause the process to end, thus terminating the flow.

In the case of DTLS, the processing is done in two stages which differ from the TCP and TLS processing. First, the datagrams including the DTLS records are received by a main process. This process will multiplex the datagrams to correct subprocesses that actually handle the cryptographic processing of the DTLS protocol.

4. CONCLUSIONS

We identified two problems that can make SIP requests or responses be forwarded over a wrong flow and proposed solutions for these problems. We are currently planning experimental analysis of effectiveness of the proxy implementation.

5. REFERENCES

[1] C. Jennings and R. Mahy. Managing Client Initiated Connections in the Session Initiation Protocol (SIP). Internet-Draft draft-ietf-sip-outbound-10, IETF, Jul 2007.