Internet Draft
Document: draft-taylor-midcom-diameter-eval-01.txt
Expires: October 2002

Tom Taylor
Nortel Networks
April 2002

Evaluation Of DIAMETER Against MIDCOM Requirements

Status of this Memo

Abstract

This document is submitted as part of the Midcom protocol selection process. It evaluates the suitability of the Diameter protocol as a transport for Midcom. The general conclusions are:

- the Diameter architecture may be too heavy for the Midcom application, although this is a matter for discussion. It is clear in any event that much of the Diameter base is not needed;
- a new application extension to Diameter would have to be defined to meet Midcom's requirements;
- with these reservations, the protocol is a good fit to Midcom requirements.

This version contains added details describing how to use Diameter to meet the requirements.

1. Introduction

1.1 Background

The Midcom Working Group has created a set of requirements for a protocol to control Middleboxes [1]. The Working Group is currently evaluating a number of protocols as a basis for development of the Midcom protocol. Diameter [2] is one of the candidates. This document reports on how well Diameter meets the Midcom requirements, using the template provided in [5].

1.2 Diameter Architecture

Diameter is designed to support AAA for network access. It is meant to operate through networks of Diameter nodes, which both act upon and route messages toward their final destinations. Endpoints are characterized as either clients, which perform network access control, or servers, which handle authentication, authorization and accounting requests for a particular realm. Intermediate nodes perform relay, proxy, redirect, and translation services. Design requirements for the protocol [3] include robustness in the face of bursty message loads and server failures, resistance to specific DOS attacks and protection of message contents, and extensibility including support for vendor-specific attributes and message types.

The protocol is designed as a base protocol to be supported by all implementations, plus extensions devoted to specific applications. Messages consist of a header and an aggregation of "Attribute-Value Pairs (AVPs)", each of which is a tag-length-value construct. The header includes a command code, which determines the processing of the message and what other AVP types must or may be present. AVPs are strongly typed. Some basic and compound types are provided by the base protocol specification, while others may be added by application extensions. One of the types provided in the base is the IPFilterRule, which may be sufficient to express the Policy Rules that Midcom deals with.

Messaging takes the form of request-answer exchanges. Some exchanges may take multiple round-trips to complete. The protocol is connection-oriented at both the transport and application levels. In addition, the protocol is tied closely to the idea of sessions, which relate sequences of message exchanges through use of a common session identifier. Each application provides its own definition of the semantics of a session. Multiple sessions may be open simultaneously.

1.3 Comparison With MIDCOM Architectural Requirements

The Midcom Agent does not perform the functions of a Diameter client, nor does the Middlebox support the functions of a Diameter server. Thus the Midcom application would introduce two new types of endpoints into the Diameter architecture. Moreover, the Midcom

requirements do not at this time imply any type of intermediate node.

A general assessment might be that Diameter meets and exceeds Midcom architectural requirements. The connection orientation may be too heavy for the number of relationships the Middlebox must support: this is a point for discussion. Certainly the focus on extensibility, request-response messaging orientation, and treatment of the session, are all well-matched to what Midcom needs. At this point, MIDCOM is focussed on simple point-to-point relationships, so the proxying and forwarding capabilities provided by Diameter are not needed. Most of the commands and AVPs defined in the base protocol are also surplus to MIDCOM requirements.

2. Detailed Comparison With Requirements

<x.x.x> indicates the requirement documented in section x.x.x of [1].

2.1 Requirements Fully Met

<2.1.1> Ability to establish association between Agent and Middlebox.

Although this is out of scope, the Diameter specification describes several ways to discover a peer. Having done so, a Diameter node establishes a transport connection (TCP, TLS, or SCTP) to the peer. The two peers then exchange Capability Exchange Request/Answer messages to identify each other and determine the Diameter applications each supports.

If the connection between two peers is lost, Diameter prescribes procedures whereby it may be re-established. To ensure that loss of connectivity is detected quickly, Diameter provides the Device-Watchdog Request/Answer messages, to be used when traffic between the two peers is low.

Diameter provides an extensive state machine to govern the relationship between two peers.

<2.1.2> Agent can relate to multiple Middleboxes.

Diameter allows connection to more than one peer (and encourages this for improved reliability). Whether the Diameter connection state machine is too heavy to support the number of connections needed is a matter for discussion.

<2.1.3> Middlebox can relate to multiple Agents.

See previous answer. The Middlebox and Agent play symmetric roles as far as Diameter peering is concerned.

<2.1.4> Deterministic outcome when multiple requests are presented to the Middlebox simultaneously.

Diameter depends partly upon the transport protocol to provide flow control when the server becomes heavily loaded. It also has application-layer messaging to indicate that it is too busy or out of space (DIAMETER_TOO_BUSY and DIAMETER_OUT_OF_SPACE result codes).

<2.1.6> Middlebox status report.

Diameter provides a number of response codes by means of which a server can indicate error conditions reflecting status of the server as a whole. The Disconnect-Peer-Request provides a means in the extreme case to terminate a connection with a peer gracefully, informing the other end about the reason for the disconnection.

<2.1.7> Middlebox can generate unsolicited messages.

The Diameter protocol permits either peer in a connection to originate transactions. Thus the protocol supports Middlebox-originated messages.

<2.1.8> Mutual authentication.

The Diameter base protocol assumes that messages are secured by using either IP Security or TLS. Diameter requires that when using the latter, peers must mutually authenticate themselves.

<2.1.9> Termination of session (connection, in Diameter terminology) by either party.

Either peer in a connection may issue a Disconnect-Peer-Request to end the connection gracefully.

<2.1.10> Indication of success or failure.

Every Diameter request is matched by a response, and this response contains a result code as well as other information.

<2.1.11> Version interworking.

The Capabilities Exchange Request/Answer allows two peers to determine information about what each supports, including protocol version and specific applications.

<2.1.12> Deterministic behaviour in the presence of overlapping rules.

The IPFilterRule type specification, which would probably be used as the type of a Policy Rule AVP, comes with an extensive semantic description which provides for a deterministic outcome, but one which the individual Agent cannot know unless it knows all of the Policy Rules installed on the Middlebox. The IPFilterRule type is defined in [2] as follows:

IPFilterRule

The IPFilterRule format is derived from the OctetString AVP Base Format. It uses the UTF-8 encoding and has the same requirements as the UTF8String. Packets may be filtered based on the following information that is associated with it:

Direction	(in or out)
Source and destination IP address	(possibly masked)
Protocol	
Source and destination port	(lists or ranges)
TCP flags	
IP fragment flag	
IP options	
ICMP types	

Rules for the appropriate direction are evaluated in order, with the first matched rule terminating the evaluation. Each packet is evaluated once. If no rule matches, the packet is dropped if the last rule evaluated was a permit, and passed if the last rule was a deny.

IPFilterRule filters MUST follow the format:

```
       action dir proto from src to dst [options]

       action       permit - Allow packets that match the rule.
                    deny   - Drop packets that match the rule.

       dir          "in" is from the terminal, "out" is to the
                    terminal.

       proto        An IP protocol specified by number.  The "ip"
                    keyword means any protocol will match.
```



```
       src and dst  <address/mask> [ports]

                    The <address/mask> may be specified as:

                    ipno       An IPv4 or IPv6 number in dotted-
                               quad or canonical IPv6 form. Only
                               this exact IP number will match the
                               rule.

                    ipno/bits  An IP number as above with a mask
                               width of the form 1.2.3.4/24. In
                               this case, all IP numbers from
                               1.2.3.0 to 1.2.3.255 will match.
                               The bit width MUST be valid for the
                               IP version and the IP number MUST
                               NOT have bits set beyond the mask.

                               For a match to occur, the same IP
                               version must be present in the
                               packet that was used in describing
                               the IP address. To test for a
                               particular IP version, the bits part
                               can be set to zero. The keyword
                               "any" is 0.0.0.0/0 or the IPv6
                               equivalent.  The keyword "assigned"
                               is the address or set of addresses
                               assigned to the terminal.  For IPv4,
                               a typical first rule is often
```

"deny in ip! assigned"

The sense of the match can be inverted by preceding an address with the not modifier (!), causing all other addresses to be matched instead. This does not affect the selection of port numbers.

With the TCP, UDP and SCTP protocols, optional ports may be specified as:

{port|port-port}[,ports[,...]]

The '-' notation specifies a range of ports (including boundaries).

Fragmented packets that have a non-zero offset (i.e. not the first fragment) will never match a rule that has one or more port specifications. See the frag option for details on matching fragmented packets.

options:

frag Match if the packet is a fragment and this is not

the first fragment of the datagram. frag may not be used in conjunction with either tcpflags or TCP/UDP port specifications.

ipoptions spec
Match if the IP header contains the comma separated list of options specified in spec. The supported IP options are:

ssrr (strict source route), lsrr (loose source route), rr (record packet route) and ts (timestamp). The absence of a particular option may be denoted with a '!'.

tcpoptions spec
Match if the TCP header contains the comma separated list of options specified in spec. The supported TCP options are:

mss (maximum segment size), window (tcp window advertisement), sack (selective ack), ts (rfc1323 timestamp) and cc (rfc1644 t/tcp connection

count). The absence of a particular option may be denoted with a '!'.

established
TCP packets only. Match packets that have the RST or ACK bits set.

setup TCP packets only. Match packets that have the SYN bit set but no ACK bit.

tcpflags spec
TCP packets only. Match if the TCP header contains the comma separated list of flags specified in spec. The supported TCP flags are:

fin, syn, rst, psh, ack and urg. The absence of a particular flag may be denoted with a '!'. A rule that contains a tcpflags specification can never match a fragmented packet that has a non-zero offset. See the frag option for details on matching fragmented packets.

icmptypes types
ICMP packets only. Match if the ICMP type is in the list types. The list may be specified as any combination of ranges or individual types separated by commas. Both the numeric values and the symbolic values listed below can be used. The supported ICMP types are:

echo reply (0), destination unreachable (3),

source quench (4), redirect (5), echo request (8), router advertisement (9), router solicitation (10), time-to-live exceeded (11), IP header bad (12), timestamp request (13), timestamp reply (14), information request (15), information reply (16), address mask request (17) and address mask reply (18).

There is one kind of packet that the access device MUST always discard, that is an IP fragment with a fragment offset of one. This is a valid packet, but it only has one use, to try to circumvent firewalls.

An access device that is unable to interpret or apply a deny rule MUST terminate the session. An access device that is unable to interpret or apply a permit rule MAY apply a more restrictive rule.

An access device MAY apply deny rules of its own before the supplied rules, for example to protect the access device owner's infrastructure.

The rule syntax is a modified subset of ipfw(8) from FreeBSD, and the ipfw.c code may provide a useful base for implementations.

<2.2.1> Extensibility.

Diameter provides a great deal of flexibility for extensions, including allowance for vendor-defined commands and AVPs and the ability to flag each AVP as must-understand or ignorable if not understood.

<2.2.3> Ruleset groups.

Diameter allows message syntax definitions where multiple instances of the same AVP (for example, a Policy Rule AVP whose syntax and low-level semantics are defined by the IPFilterRule type definition) may be present. If a tighter grouping is required, the set of Diameter base types includes the Grouped type. Midcom can choose how to make use of these capabilities to meet the rulreset group requirement when defining its application extension to the Diameter protocol as discussed below.

<2.2.4> Lifetime extension.

The Diameter concept of a session includes the session lifetime, grace period, and lifetime extension. It may make sense to associate the Diameter session with the lifetime of a Midcom Policy Rule, in which case support for lifetime extension comes ready-made.

<2.2.6> Actionable failure reasons.

Diameter provides an extensive set of failure reasons in the base protocol.

<2.2.7> Multiple Agents operating on the same ruleset.

Diameter itself offers no impediment to such an operation. The Midcom application specification must avoid introducing such an impediment.

<2.2.11> More precise rulesets contradicting overlapping rulesets.

Allowed by the IPFilterRule semantics described above.

<2.3.1> Message authentication, confidentiality, and integrity.

Diameter relies on either IPSEC or TLS for these functions.

<2.3.2> Optional confidentiality.

Implementation support of IPSEC ESP in Diameter applications is not optional. Deployment of either IPSEC or TLS is optional.

<2.3.3> Operation across untrusted domains.

The Diameter specification [2] recommends the use of TLS across untrusted domains.

<2.3.4> Mitigation of replay attacks.

Diameter requires that implementations support the replay protection mechanisms of IPSEC.

2.2 Requirements Partially Met

Requirements have been placed here in most cases because it will be necessary to define a Midcom application extension of Diameter, and the satisfaction of the requirements depends on proper definition of the messages and AVPs in that extension.

<2.1.5> Known and stable state.

Diameter documentation does not discuss the degree of atomicity of message processing, so this would have to be specified in the Midcom extension.

<2.2.2> Support of multiple Middlebox types.

Any necessary additional AVPs or values must be specified as part of the Midcom application extension (see <2.2.8> below).

<2.2.5> Mandatory/optional nature of unknown attributes.

Indication of the mandatory or optional status of AVPs is fully supported, provided it is enabled in the AVP definition. No guidance is imposed regarding the return of diagnostic information for optional AVPs.

<2.2.8> Transport of filtering rules.

While Diameter defines the promising IPFilterRule data type (see 2.1.12 above), there is no existing message which would convey this to a Middlebox along with other Midcom-required attributes. A new Midcom application extension of Diameter would have to be defined.

<2.2.9> Mapped port parity.

This capability is not part of the current IPFilterRule type definition. Rather than modify the IPFilterRule type, Midcom could group it with other AVPs which add the missing information.

<2.2.10> Consecutive range of port numbers.

This capability is not part of the current IPFilterRule type definition. Rather than modify the IPFilterRule type, Midcom could group it with other AVPs which add the missing information.

2.3 Requirements Not Met

None.

References

1. R. Swale, P. Mart, P. Sijben, S. Brim, M. Shore, "Middlebox Communications (midcom) Protocol Requirements", draft-ietf-midcom-requirements-05.txt (approved as RFC), November 2001.

2. P. Calhoun, J. Arkko, E. Guttman, G. Zorn, J. Loughney, "Diameter Base Protocol", draft-ietf-aaa-diameter-10.txt, IETF

work in progress, April 2002.

3. P. Calhoun, G. Zorn, P. Pan, H. Akhtar, "Diameter Framework Document", draft-ietf-aaa-diameter-framework-01.txt, IETF work in progress, March 2001.

4. P. Calhoun, W. Bulley, A. Rubens, J. Haag, G. Zorn, D. Spence, "Diameter NASREQ Application", draft-ietf-aaa-diameter-nasreq-09.txt, IETF work in progress, March 2002.

5. M. Barnes, "MIDCOM Protocol Evaluation Template", draft-midcom-protocol-eval-template.txt, March 2002.

Author's Addresses

Tom Taylor
Nortel Networks
1852 Lorraine Ave.
Ottawa, Ontario, Canada K1H 6Z8
Phone: +1 613 736 0961
Email: taylor@nortelnetworks.com