# Comparison between RADIUS and Diameter

Anna Hosia
Helsinki University of Technology
Telecommunications Software and Multimedia Laboratory

May 28, 2003

**Abstract**

RADIUS is a widely deployed protocol for AAA (Authentication, Authorization, and Accounting) control, while Diameter is a draft planned as its successor. The protocols resemble each other in many ways. For example, their packet formats are quite similar, and they provide support for same kind of AAA mechanisms. However, while RADIUS is a pure client-server protocol, Diameter is more of a peer-to-peer protocol, as also Diameter servers can ask for certain services.

On the transport layer RADIUS uses connectionless UDP, while Diameter utilizes either SCTP or TCP. Diameter's operation is more reliable, mainly because its specification addresses issues such as fail-over procedure and proxy/agent support, while RADIUS specification omits these subjects.

One of Diameter's strengths is that it is backward compatible with RADIUS. Diameter contains also mechanisms for version compatibility support, while RADIUS specification hardly discusses the issue. For instance, Diameter supports error messages while RADIUS does not. Both protocols are designed to be extensible, but Diameter provides more extension mechanisms. Diameter also scales far better than RADIUS, mainly because RADIUS has no provisions for congestion control.

Diameter always uses some kind of transport layer security scheme, such as IPSecurity or TLS, while for example IPSecurity support for RADIUS is optional. This affects mechanisms for entity authentication and overall data security making Diameter a more secure protocol.

## 1 Introduction

RADIUS and Diameter are both protocols for AAA (Authentication, Authorization, and Accounting) control. Nowadays, RADIUS is widely implemented and used, but since it has some severe limitations, Diameter is planned as its successor. [1, 2]

The basic operation of RADIUS and Diameter is similar. They both carry authentication, authorization and configuration information between a Network Access Server (NAS) and a shared Authentication Server. Users wishing to have a link outside contact their NAS, and the NAS asks the server for access. The server has a database containing authentication and configuration information for making decisions. [1, 2] The AAA architecture is presented in Fig. 1.

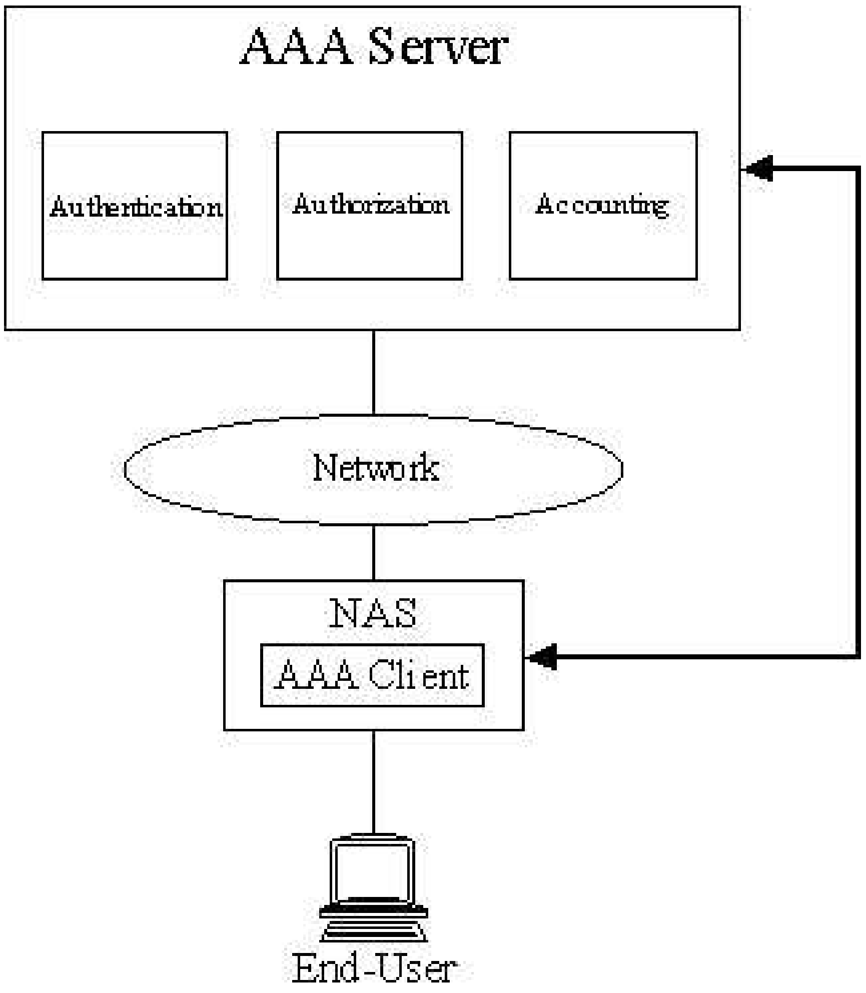

Figure 1: AAA architecture. The AAA client at the network point of entry (NAS) communicates with the AAA server to provide AAA services. [10]

The first RADIUS (Remote Authentication Dial In User Service) RFC was published in 1997, and was originally developed to provide dial-up PPP and terminal server access. However, since those days network access servers and routers have changed enourmously with the rise of new access technologies, such as DSL, Mobile IP, and wireless techniques. This development has put new demands on AAA protocols. [1, 2]

There are two more RADIUS RFCs, which fix faults found in older ones. However, revising all problems of RADIUS - for example the lack of provisions for congestion control - would require major changes to the protocol. Thus IETF AAA Working Group has taken the mission to develop RADIUS a successor, Diameter, which addresses the problems encountered with RADIUS. At the moment Diameter is still an Internet Draft and subject to changes. [1, 2]

As the technological evolution has not stopped, Diameter is designed to be extensible. There is a common base specification that can be used by itself for accounting purposes only. In addition, there are separate applications, which provide extra functionality for different environments. Currently there are two applications, Mobile IPv4 for mobile nodes and NASREQ for PPP/SLIP dial-up and terminal server access environment. [2]

This document compares multiple aspects of RADIUS and Diameter. It starts with an overview section handling for example the AAA mechanisms of the protocols. This is followed by an examination of the suitability of the protocols for evolving, changing environments. After that overall usability issues, such as reliability and scalability, are discussed. Finally, the end of the document is devoted to security issues.

# 2 Protocol overview

As Diameter has evolved from RADIUS, the operation of the protocols resembles each other. They use same kind of packets and support many same AAA functionalities. On the other hand, there are many both minor and major differences, since Diameter's purpose is to improve the operation of RADIUS.

## 2.1 Protocol basics

Diameter extends many functionalities of RADIUS. As a result the packet format has evolved, transport mechanisms have changed, and the overall concept has shifted from client-server towards peer-to-peer.

**Operation paradigm**

RADIUS operates in a pure client-server paradigm, where the NAS acts as client. The RADIUS server does not initiate any messages, but only replies to the messages sent by the clients. RADIUS is also a protocol of stateless nature. Nodes maintain a very limited amount of information - this includes for example message identifiers in order to associate incoming replies with sent requests. [1]

Diameter operation resembles that of RADIUS, as NASes act as Diameter clients to the Diameter server. However, with Diameter any node can initiate a request, which makes Diameter more of a peer-to-peer protocol. Diameter maintains also more state information than RADIUS. It defines the concepts of both transaction and session states, and even so called "stateless agents" have to maintain transaction state information. [2]

**Packet formats**

The basic format of RADIUS and Diameter packets is similar. They include first a fixed-size header, and after that a variable number of attributes, which are generally referred to as AVPs (Attribute Value Pairs). [1, 2]

Both RADIUS and Diameter headers include only basic information necessary to the protocol operation. This includes the code field, which contains the message type, and the message length field. RADIUS has an identifier field and Diameter a so called hop-by-hop identifier, which are both used for matching incoming replies with former requests and for detecting duplicates. RADIUS header contains also an Authenticator field for message authentication. Diameter has a field for version number, application ID, end-to-end identifier, and some command flags. [1, 2]

In both protocols AVPs are used for carrying all specific data. Each AVP consists of attribute code, length, and value, and in Diameter's case also of some control flags and vendor ID. The value field contains information concerning authentication, authorization, or accounting, as well as for example routing. There are some limitations that particular AVPs must or must not be present in certain types of packets. [1, 2]

**Transport issues**

RADIUS and Diameter use different transport protocols for information delivery. RADIUS runs over UDP (User Datagram Protocol), while Diameter can use either SCTP (Stream Control Transmission Protocol) or TCP (Transmission Control Protocol). [1, 2]

UDP is a natural choice for RADIUS, since it is a stateless protocol. UDP is also a lightweight protocol, and it is simple to create multi-threaded RADIUS servers with it. The cost of using UDP is that RADIUS implementations have to create and manage retransmission timers themselves, as UDP provides no retransmission strategy. [1]

Contrary to RADIUS, Diameter runs over reliable transport mechanisms, that is, over SCTP or TCP. At the moment Diameter clients have to support either of them, and servers both. However, it may become mandatory for clients to support SCTP. Both protocols provide acknowledged, error-free, non-duplicated transfer of user data. Thus they offer better service than UDP, but on the other hand, they are heavier to implement and operate. [2, 3]

As typical AAA protocols, RADIUS and Diameter are both application-driven. This means that the rate by which they send messages depends mostly on how quickly the application generates them, not on the size of the congestion window or network settings. With application-driven protocols there is not usually enough traffic to piggyback acknowledgement messages. Thus reliable protocols, such as TCP and SCTP, may double the traffic amount compared with implementations using UDP. [4]

**Proxies and agents**

Both RADIUS and Diameter allow additional nodes on the path between the communicating client and server. With RADIUS these nodes are generally referred to as proxies. [1] With Diameter the term agent is used, because Diameter specifies several agent types, of which "proxy" is one type. [2]

Diameter defines four kinds of agents, which provide relay, proxy, redirect or translation services. Relays forward messages based on routing-related AVPs and realm routing information. Proxies forward messages similarly to relays, but also make policy enforcement decisions. Redirects refer clients to servers, and the actual data are not routed via them. Translation agents perform protocol conversion between Diameter and other AAA protocols, such as RADIUS. [2]

Both protocols allow creating proxy (or agent) chains, where there are several intermediate nodes. It is also possible in both protocols that a node acts as a server to some requests, and as an agent to other kinds of requests. However, there is one important difference concerning proxies in the specifications: The RADIUS specification does not define precisely the behaviour of proxies, and thus it may vary between implementations. Contrary to that, Diameter defines its agents' behaviour explicitly. [1, 2]

## 2.2 AAA mechanisms

Both being usable AAA protocols, RADIUS and Diameter have many similart AAA skills. However, the capabilities of Diameter are sometimes more expansive.

This section is based on RFC 2989, "Criteria for Evaluating AAA Protocols for Network Access" ([5]), and RFC 3127, "Authentication, Authorization, and Accounting: Protocol Evaluation" ([6]) unless otherwise stated. The former describes how protocols can be evaluated, and the latter gives evaluation results for some protocols including RADIUS and Diameter.

**Authentication capabilities**

Both RADIUS and Diameter support authentication using NAIs (Network Access Identifier), CHAP (Challenge Handshake Authentication Protocol), EAP (Extensible Authentication Protocol), and PAP (Password Authentication Protocol); according to RFC 3127 they both meet requirements for using these authentication methods totally. However, RADIUS has some limitations: Its CHAP authentication is subject to dictionary attacks, and it protects clear-text passwords (PAP) only on a hop-by-hop basis. The EAP support in RADIUS is not described in the generic RFC, but in RFC 2869 "RADIUS Extensions". Diameter base protocol offers support for NAIs, while the NASREQ application (see Sec. 1) for CHAP, EAP and PAP.

With RADIUS only clients can make reauthentication requests. This is done by simply sending a new authentication request to the server. However, the server cannot demand reauthentication on demand, as it cannot initiate messages (for server-initiated messages see Sec. 2.3). [1] Diameter instead specifies two message types, Re-Auth-Request and Re-Auth-Answer, which allow also server-initiated reauthentication. [2]

Diameter supports also authorization without authentication, while the support of RADIUS is two-folded. For the time being, RADIUS requires some form of credential in all authorization request messages. On the other hand, it is possible to create new message types to support authorization without authentication in RADIUS.

**Authorization capabilities**

According to RFC 3127 both RADIUS and Diameter offer some support for access rules, authorization restrictions, and filters. However, neither of them meets the requirements concerning this area totally, but for example the abilities of proxy brokers to deny access are good in both protocols.

As with reauthentication, Diameter supports reauthorization on demand better than RADIUS. In RADIUS reauthorization is currently coupled with reauthentication, and thus it suffers from the same problems. In Diameter reauthorization is possible at any time.

Protocols differ also in the ability of the AAA server to ask client NASes to disconnect a session of a particular user for authorization policy reasons. Diameter supports such unsolicited disconnects, while radius doesn't. [1, 2] For more information, see Sec. 2.3.

**Accounting capabilities**

Basic accounting capabilities of RADIUS and Diameter are similar. Both protocols support real-time accounting, meaning that account reporting happens synchronously with events in a time-scale of seconds. Diameter base protocol supports also accounting timestamps and dynamic accounting - that is, accounting for dynamic authentication and authorization. RADIUS Extensions, RFC 2869, supports the same capabilities.

Both protocols also support guaranteed delivery of accounting information with application level acknowledgements, and have extensible accounting records. The delivery mechanisms relates to the retransmission procedure described in Sec. 4.1, and the protocol extensibility is discussed in Sec. 3.2.

## 2.3 Message handling

Diameter and RADIUS systems have different message handling capabilities. With Diameter any of the involved parties can ask for certain services, while only RADIUS clients can initiate communication. Diameter supports also an error reporting scheme, while RADIUS does not.

### Server-initiated messages

RADIUS specification does not support server-initiated messages. RADIUS acts a pure client-server implementation: the client makes a request, and the server replies. [1] However, there is an IETF work in progress, "Dynamic Authorization Extensions to RADIUS", which defines RADIUS server-initiated messages. On the other hand, supporting the draft is optional. [2]

In Diameter support for server-initiated messages is mandatory. This allows the server for example to request to abort service to a particular user or demand reauthentication or reauthorization. [2]

### Silent discarding and error messages

RADIUS does not support error messages. When faults occur, RADIUS simply silently discards packets - drops them without further processing. This applies for example to packets with an unknown code, or packets the length of which is shorter than it says on the length field. In other words, invalid packets are always silently discarded. [1]

Contrary to RADIUS, Diameter has an error reporting mechanism. Diameter messages are silently discarded only, when it is the most suitable way to solve the problem. For example received duplicate answers are still silently discarded. [2]

In general there are two different types of Diameter errors: protocol and application errors. Application error messages are informational, or tell about a transient or permanent failure - or even about success. The error explanation is stored in Result-Code AVP, which is read by the peer the message of which originated the error. Protocol errors differ from application errors in the way that each proxy and relay agent on the transfer path may try to correct the error. That is why the presence of protocol errors is always stated in the protocol header as one of the control bits, the "E" bit, set. [2]

# 3 Suitability for evolving environments

When tested in real environments, needs for new protocol versions or extensions may be discovered. Thus overall compatibility should be considered already in the protocols design phase. The issue is especially important for Diameter, since the protocol is meant to be extensible and compatible with RADIUS.

## 3.1 Compatibility

The version compatibility support of RADIUS is fairly poor. The RADIUS specification simply states, that messages arriving with an invalid Code field are silently discarded, and attributes with an unknown type may be ignored. RADIUS does not support any capability negotiation, and its attributes do not contain information, whether the support for an attribute is necessary. [1] Thus different RADIUS versions will work together only as long as they use same code and type information. As RADIUS does not support even error messages, the entities which do not receive any replies to their requests have no way of knowing whether the other party drops messages of if there is for example network congestion.

Diameter is more developed in compatibility issues than RADIUS. Diameter supports capability negotiation, attribute flags, and error handling. Every time when two Diameter peers establish a transport connection, they must exchange the Capabilities Exchange messages. These allow the peers to discover each other's protocol version number, supported Diameter applications and security mechanisms among other things. [2]

In Diameter every AVP also contains a "mandatory (M) bit", which specifies whether support of the AVP is required. When a party receives a message with an unrecognized mandatory AVP, it sends back an answer specifying the failed AVP. [2]

Diameter header also contains an eight-bit field for protocol version, while RADIUS header doesn't. [1, 2] Currently there is only one Diameter version, but in the future the version field may appear to be important.

Diameter is also designed to be backward compatible with RADIUS. For example Diameter AVP numbers 1 through 255 are reserved for RADIUS attributes, and similarly command codes 0 through 255 are reserved for RADIUS packet type codes. [2]

Diameter specification defines also translation agents (see Sec. 2.1), which provide protocol conversion between Diameter and other AAA protocols. [2] With RADIUS the translation agents must change for example transport layer protocol data units and message formats, and also manage the differences concerning state information.

## 3.2 Extensibility

Both protocols are designed to be extensible. However, Diameter contains for that more mechanisms.

RADIUS RFC states extensibility as one of the key features of the protocol. As RADIUS transactions are comprised of AVPs, extensions are made by creating AVPs. This allows vendors for example to create new authentication methods. [1]

The actual mechanism for these extensions is the use of RADIUS attribute 26 "Vendor-Specific". In this attribute the first four bytes of the value section identify the vendor, and the rest of the value field is used for the vendor-specific information, the exact format of which is defined by the vendor itself. [1]

Extensions must be created carefully so that they do not affect the common operation of the protocol. If a RADIUS server does not recognize a vendor-specific attribute it must ignore it. In that case the client should try to cope without the vendor-specific data even if in a degraded mode. [1]

RADIUS is designed to be extended, when the original base attributes do not offer desired functionality. Diameter instead is designed to be extended even for common applications. The base protocol may be used by itself only for accounting applications - for all other purposes some extension application must be used. [2]

There are several mechanisms to extend base Diameter protocol. These include defining new AVP values or creating new AVPs, creating new authentication/authorization or accounting applications, and defining new application authentication procedures. [2]

While RADIUS extensions are vendor-specific, in Diameter the extensions are for public use. Thus IANA controls the address spaces of AVP values, AVP types, and application identifiers, and for new allocations one must send a request to IANA (Internet Assigned Numbers Authority) with a proper explanation or specification. New command codes can be created only by IETF Consensus. Although Diameter is supposed to be extended, creating new extensions is encouraged only, when there are no existing solutions available. Otherwise the standardization and implementation issues could become too complicated. [2]

One point with affects the protocol extensibility, is the difference in the attribute address spaces of RADIUS and Diameter. In RADIUS the attribute type field is eight bits long allowing only 256 attributes. Thus all the other attributes must be used inside the "Vendor-Specific" attribute limiting somewhat the flexibility of the protocol. In Diameter there are 32 bits for the AVP code allowing over 4 billion different AVPs. [1, 2]

## 4 Protocol usability

Protocol reliability deals with aspects such as the quality and failure detection of transport services. It covers the fail-over to another server, when the primary server is unable to answer. Scalability relates to how many clients a shared AAA server can serve at the same time.

### 4.1 Reliability

RADIUS does not define exact retransmission behaviour. The specification simply states with regard to the Access-Request messages that "if no response is returned within a length of time, the request is re-sent a number of times". [1] Thus the details of the RADIUS retransmission algorithm are implementation-specific, which makes also the protocol reliability vary between implementations. Similarly the RADIUS specification does not contain precise fail-over mechanisms, but the behaviour varies again between implementations. [2] When a RADIUS client finds the server to be down or unreachable, it can forward its request to an alternate server. [1]

In Diameter the transport failure behaviour is far better defined. There is a transport failure algorithm defined in "AAA Transport Profile" ([4]), which all Diameter implementations must support. The protocol has two special messages, Device-Watchdog-Request and Device-Watchdog-Answer to detect transport failures. The Diameter specification also contains explanation of fail-over procedure. [2]

The Diameter fail-over procedure is more precise than in RADIUS. Diameter nodes maintain a pending message queue, which contains sent messages which haven't received an answer yet. After detecting a transport failure, messages in the queue are sent to an alternate agent. This excludes some messages - for example, if the unavailable peer was the fixed destination of some message, there's no point to resend that message. When the failed peer is again available, the transport connection is re-established. This is referred to as failback. [2]

RFC 3127 ([6]) evaluates both RADIUS and Diameter to meet fail-over requirements only partially. For example, Diameter is blamed for not specifying how and when to switch back to the primary server, when it has recovered.

### 4.2 Scalability

One of the main reasons for creating Diameter is the poor scalability of RADIUS. As RADIUS lacks any support for congestion control, it is unsuitable for many network environments.

**Protocol headers**

Both RADIUS and Diameter contain identifier header fields to associate replies with the original requests (see Sec. 2.1). As the identifier field of RADIUS is eight bits long, RADIUS can in principle have only 256 pending request at the same time. For Diameter with its 32-bit field the theoretical maxmimum is over 4 billion. [1, 2]

However, this scalability limitation of RADIUS can be worked around. The identifier is actually used to identify connections between two endpoints described by a 5-tuple (client IP address, client port, UDP, server IP address, server port). Thus one of the well known techniques is to change the source UDP port number of the request. [4]

RADIUS and Diameter headers differ also with regard to alignment rules. With RADIUS header fields are simply put one after other without any special care for alignment. [1] With Diameter the fields are 32-bit aligned, which makes many modern processors able to handle the transactions faster. [2]

**States**

States information can be kept on two different layers - on the transport layer or on the session level. The scalability on the transport layer is proportional to the number of client/server relationships, and on the session level to the number of users. However, maintaining any state information consumes resources. [7]

On the transport layer RADIUS implementations are lighter than Diameter implementations. As RADIUS runs over stateless UDP, the only state information it has to maintain are the identifiers for matching replies with requests. [1] Contrary to that, the stateful transport protocols used by Diameter - TCP and SCTP - require keeping track of window sizes and timers. [2] Thus Diameter's transport layer can be considered less scalable than UDP operation.

On the session level RADIUS does not maintain any real-time states. However, it maintains an off-line "state" for accounting purposes, so that accounting stops and accounting starts can be matched together. [7] Diameter instead maintains much more session level state information. So called stateful agents (for agents see Sec. 2.1) keep track of all authorized active sessions. All agents - also stateless ones - must also keep track of transactions state for fail-over purposes. [2]

**Congestion control**

To avoid congestion, the client should not retransmit a packet to the same server or choose another server until it can be reasonably sure, that previous packet has exited the network on the same path. Thus the client should discover the available bandwidth by examining the response messages, and then self-clock - that is, adjust its transmission rate based on that information. [4]

RADIUS has no provisions for congestion control. That is one reason for why RADIUS may not be suitable for large-scale systems, as it may suffer degraded performance and lose data. [1]

Diameter does not either have any application level support for congestion control. However, it runs over TCP or SCTP, both of which are reliable transport protocols and able to self-clock. The self-clocking and congestion control work fine as long as the client is communicating directly with the server. However, end-to-end self-clocking may be impossible, if there are other AAA agents between the client and the server. With agents such as relays or proxies, there are two separate connections - one between the additional agent and the client, the other between the agent and the server. Thus responses do not flow directly from end to end, and self-clocking is not possible. However, agents such as transport proxies and redirects do allow self-clocking, since there is only one transport connection. [4]

**Shared secrets**

RADIUS always uses shared secrets, and also Diameter may utilize them. The distribution and safe storage of the secrets may present major problems, especially with RADIUS. This topic is discussed in Sec. 5.1.

## 4.3 Other usability issues

### IPv6

The Internet is slowly starting to change over from IPv4 to IPv6. That fact is taken into account in the Diameter design process, but the RADIUS specification is originally made considering IPv4 as the network protocol. However, RFC 3127 ([6]) states, that both RADIUS and Diameter meet the requirements of running over IPv4 and IPv6 totally.

The operation of RADIUS when run over IPv6 is specified in RFC 3162, "RADIUS and IPv6". Using IPv6 requires some new attributes, mainly because IPv6 addresses are longer than IPv4 addresses. Anyway, there are actually few changes, and thus IPv6 does not seem to represent major problems. [8]

The specification of RADIUS over IPv6 also respects the fact that the change from IPv4 to IPv6 is going to be slow. The specification states, that the client may not know in advance, whether the server supports IPv6 or not. The client may also provide IPv6 access natively, using IPv6 within IPv4 tunnels, or using IPv6 over IPv4 without explicit tunnelling. The choice of IPv6 usage has necessarily no effect on RADIUS functions. [8]

Diameter does not have any problem related to the differences of address lengths in IP versions. That is because the Diameter "Address" data type includes the address type, which specifies the actual length of the address. [2]

**Firewalls**

RFC 3127 discusses also the firewall friendliness of the protocols. By firewall friendliness it means not making the firewall look into the packets too deeply, but mainly to work with the mere application port number. In this check RADIUS is stated to be totally compliant with the requirement, as it uses an officially assigned port 1812. RADIUS is also generally known to be operational in environments, where there are firewalls acting as a proxy. However, Diameter meets the requirement only partially. Diameter's problem is the SCTP transport protocol, which is not widely recognized yet. [6] There are no officially assigned TCP and SCTP port numbers for Diameter operation yet. However, they have been requested of IANA. [2]

## 5 Security

AAA nodes are tempting targets for attackers, because they provide network access services. In general, Diameter gives better attention to security related issues than what RADIUS does.

IP Security (IPSec) offers protection against many security threats. All Diameter clients must support IPSec and may support TLS (Transport Layer Security) protocol, and Diameter servers must support both. Diameter implementations must always use some kind of transmission-level security. RFC 3162, "RADIUS and IPv6" ([8]), defines the use of IPSec for RADIUS, but supporting it is not mandatory. [2]

Diameter has also a separate end-to-end security framework, "Diameter CMS Security application", which provides many security services. It offers wider transport-layer protection than mere IPSec or TLS do. It is strongly recommended but not obligatory for the Diameter implementations to support the end-to-end security.

The text mentions shortly some of the most common security attacks and if the protocol specifications offer protection against them. The structure of this section is partly based on reference [7], which is a comparison between older versions of RADIUS, Diameter, and COPS protocols.

## 5.1 Entity authentication

Proper entity authentication - ensuring the identity of the other party involved in communication - offers protection against many common attacks. It enables one to counter attacks based on masquerading, man-in-the-middle, and unauthorized access, and may also help protection against information forgery and denial of service attacks.

Entity authentication can happen on hop-by-hop basis, when adjacent nodes discover each other's identities. The other, more comprehensive alternative is end-to-end authentication, when the identities are solved even if there are proxies or other agents (see Sec. 2.1) on the path between the client and the server.

**RADIUS authentication**

RADIUS entity authentication applies only between two adjacent nodes. Thus there is no end-to-end authentication with proxy RADIUS. RADIUS authentication scheme is based on the concept of shared secrets, where each two directly communicating parties must have a shared secret.

RADIUS authentication is based on using the authenticator header field (see Sec. 2.1). The field is called the Request Authenticator when travelling from the client to the server, and the Response Authenticator the other way round. When PAP authentication is used, the authentication is guaranteed in the following way: The client sends an Access-Request message, and the Request Authenticator contains a random number. The message also contains a User-Password attribute, the value of which is a one-way MD5 hash calculated over the concatenation of the shared secret and the Request Authenticator, finally xored with the user password. When the server replies, the Response Authenticator contains a one-way MD5 hash over the concatenation of the RADIUS packet (where the authenticator field is replaced with the Request Authenticator from the previous Access-Request packet), and the shared secret. Thus the endpoints of the communication can repeat the calculation and verify, that the other party used the right secret (and password). [1]

With RADIUS proxies each two directly communicating entities have their own shared secret. With Access-Request messages the proxy first decrypts the user password by using its shared secret with the client, and then encrypts the password again using the secret with the server. In the other direction proxies must recalculate the Response Authenticator. [1]

**Problems with RADIUS authentication scheme**

The shared secrets scheme used by RADIUS calls forth many problems. While the secrets are stored on the RADIUS servers they must be protected against attacks trying to discover them. Also the distribution of shared secrets must be organized so as not to reveal the secrets to unauthorized parties. The difficulties are emphasized with the large number of secrets to protect, as there must be own secret for each hop in a proxy chain. [1] This results in a large administrative burden, which may turn out as administrators reusing the shared secrets. [2]

The poor quality of the random number a the Request Authenticator may also cause security problems. If the same Request Authenticator and shared secret combination occurs more than once, an attacker may use it for replay attacks. Thus the random numbers of the Request Authenticators should be unique over the lifetime of the shared secret in use. As the same secret may be used over other connections in other geographic regions as well, the Request Authenticator should be both globally and temporally unique. [1]

Besides being unique, the Request Authenticator should also be unpredictable. Otherwise an attacker may send an Access-Request message to the server with a certain Request Authenticator and get an answer. Then if the real client would use the same Request Authenticator, the attacker could replay the former answer masquerading as the server. [1]

**Diameter authentication**

Diameter's hop-by-hop authentication is guaranteed by TLS or IPSec. They both provide security across a transport connection as long as there are no untrusted third party agents on the transport path. In those cases end-to-end security in needed. [2]

Diameter agents complicate the entity authentication procedure. As data is routed via relays or proxies, the former authentication method is not sufficient any more. However, Diameter's end-to-end security framework provide message origin authentication also when there are relays or proxies present. [2] The security offered by translation agents depends on the other protocol as well, and is not covered here.

**Problems with Diameter authentication**

Diameter implementations must support peer authentication between communication endpoints using a pre-shared key. This brings forth same kind of key management problems as RADIUS has. Luckily, as there are other authentication methdods, such as using certificates, available as well, the management problem is smaller in scale. [2]

Diameter security framework uses digital signatures along with digital certificates and/or asymmetric encrypting techniques. The latter includes the problem of distributing public keys, the former the management of certificates - especially the verification of certificate revocations may create a security hole. [9]

## 5.2 Data security

Data security deals with the quality of data transfer. Data integrity check should be provided in order to prevent data corruption. Data confidentiality instead links to eavesdropping and traffic flow analysis - its purpose is to prevent unauthorized parties to gain information about the data.

**RADIUS data security**

As RADIUS packet data are used when calculating the Response Authenticator, it provides an authentication and integrity check for the data. However, as the calculation of the Request Authenticator does not include the RADIUS packet, it does not offer the same protection. RADIUS Extensions (RFC 2869) defines an additional data authentication and integrity mechanisms, but it is only used during Extensible Authentication Protocol (EAP) sessions. [2]

RADIUS support for confidentiality is very limited. All user passwords are always sent encrypted, but that's usually the only encrypted part of the packet. [1] Neither the RADIUS specification nor the RADIUS Extensions provide support for whole packet confidentiality. Also the password protections scheme does not seem to be flawless - at least some in the IETF community are concerned that it might not provide sufficient confidentiality protection. [2]

RADIUS data authentication, integrity and confidentiality support suffers also because of its proxy policy. Because of the hop-per-hop shared secrets and changing identifiers, all proxies must be able to read and modify any message. Proxies also may or may not send Proxy-State attributes from the client side to the remote server, and they may need to modify other attributes to enforce a local policy. Thus the messages may change when travelling through the proxies, which makes the entire encryption difficult. As a result, the user should be able to trust the proxies the packet is visiting. [1]

**Diameter data security**

As Diameter requires using either IPSec or TLS, Diameter data is automatically authenticated, confidential, and replay and integrity protected as long as there are no relays or proxies present. However, when relays or proxies route Diameter messages, they insert and remove routing information. As proxies also implement policy enforcement, the may have to modify other parts of the message as well. Thus in the presence of relays and proxies AVP integrity and confidentiality is guaranteed by using the Diameter end-to-end security framework. [2]

## 5.3 Other security considerations

Neither of the protocols provides direct support against denial of service (DoS) attacks. DoS attacks are typically carried out by flooding the target equipment with bogus traffic. Even if the target could recognize the received data as faulty, the recognition requires certain amount of processing, which may finally submerge the target. However, the design of SCTP protocol, over which Diameter may run, includes some resistance to flooding. [3]

Another issue, which relates to denial of services attacks, is the enabling of failover between AAA agents. Both RADIUS and Diameter support some kind of failover algorithms, which the client starts when it has not received any answers for a certain amount of time. This makes the protocols vulnerable for the following service of denial attack: The attacker may try to swamp the network, so that response packets are dropped, and the client starts its failover procedure. By repeating this attack, the client cycles between different AAA agents without ever getting real service. [4]

Both protocols offer some protection against replay attacks, Diameter better than RADIUS. Diameter requires transmission level security using TLS or IPSec, which guarantees replay protection. [2] However, as IPSec can be used with RADIUS as well to protect it from the replay attacks.

## 6 Conclusion

RADIUS has been in wide-scale use for years, and has thus proven itself to be a workable AAA protocol. However, it suffers from poor scalability, which makes it an unacceptable choice in large networks.

As Diameter is made to address RADIUS flaws, it is superior to its predecessor concerning many issues. Consequently, the further development of RADIUS is hardly reasonable any more, since the outcome would very likely resemble the functionality of Diameter. That is why the resources should be directed in improving Diameter instead.

Despite its better capabilities, Diameter is far from perfect. One of its greatest disadvantages is its complexity, when compared to RADIUS. However, its extensibility allows it to satisfy very different reqirements, and thus when the standardization process is over, Diameter is likely to become the most widely deployed AAA protocol.

## References

[1] C. Rigney, S. Willens, A. Rubens, and W. Simpson. Remote Authentication Dial In User Service (RADIUS). RFC 2865, IETF Network Working Group, June 2000.

[2] Pat R. Calhoun, John Loughney, Erik Guttman, Glen Zorn, and Jarki Arkko. Diameter Base Protocol. Work in Progress, IETF AAA Working Group, December 2002.

[3] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson. Stream Control Transmission Protocol. RFC 2960, IETF Network Working Group, October 2000.

[4] Bernard Aboba and Jonathan Wood. Authentication, Authorization and Accounting (AAA) Transport Profile. Work in Progress, IETF AAA Working Group, January 2003.

[5] B. Aboba et al. Criteria for Evaluating AAA Protocols for Network Access. RFC 2989, IETF Network Working Group, November 2000.

[6] D. Mitton, M. St.Johns, S. Barkley, D. Nelson, B. Patil, M. Stevens, and B. Wolff. Authentication, Authorization, and Accounting: Protocol Evaluation. RFC 3127, IETF Network Working Group, June 2001.

[7] Ronnie Ekstein, Yves T'Joens, Bernard Sales, Olivier Paridaens. AAA Protocols : Comparison between RADIUS, DIAMETER and COPS. Internet draft (expired), IETF AAA Working Group, April 2000.

[8] B. Aboba, G. Zorn, and D. Mitton. RADIUS and IPv6. RFC 3162, IETF Network Working Group, August 2001.

[9] Pat R. Calhoun, Stephen Farrell, and William Bulley. Diameter CMS Security Application. Internet draft (expired), IETF AAA Working Group, March 2002.

[10] Christopher Metz. AAA PROTOCOLS: Authentication, Authorization, and Accounting for the Internet. IEEE Internet Computing online, 2001.