AAA Working Group
INTERNET-DRAFT
Expires in six months

S. Farrell
Baltimore Technologies
J. Vollbrecht
Merit Network, Inc.
P. Calhoun
Sun Microsystems, Inc.
L. Gommans
Cabletron Systems EMEA
G. Gross
Lucent Technologies
B. de Bruijn
Interpay Nederland B.V.
C. de Laat
Utrecht University
M. Holdrege
Lucent Technologies
D. Spence
Merit Network, Inc.
October 1999

AAA Authorization Requirements
<draft-ietf-aaa-authorization-reqs-01.txt>

Status of this Memo

Abstract

This document specifies the requirements that AAA protocols must meet in order to support authorization services in the Internet. The requirements have been elicited from a study of a range of applications including mobile-IP, roamops and others.

Table Of Contents

1. Introduction

<<Comments are in angle-brackets like this. Where text was added late in the editing process, requirements are marked "late entrant".>>

The key words "MUST", "REQUIRED", "SHOULD", "RECOMMENDED", and "MAY" in this document are to be interpreted as described in [RFC2119].

This document is one of a series of three documents prepared by the AAA WG authorization subgroup dealing with the authorization requirements for AAA protocols. The three documents are:

AAA Authorization Framework [FRMW]
AAA Authorization Requirements (this document)
AAA Authorization Application Examples [SAMP]

The process followed in producing this document was to analyze the requirements from [SAMP] based on a common understanding of the AAA authorization framework [FRMW]. This document assumes familiarity with both the general issues involved in authorization and, in particular, the reader will benefit from a reading of [FRMW] where, for example, definitions of terms can be found.

2. Requirements

Requirements are grouped under headings for convenience; this grouping is not significant.

Definitions and explanations of some of the technical terms used in this document may be found in [FRMW].

Each requirement is presented as a succinct (usually a sentence or two) statement. Most are followed by a paragraph of explanatory material, which sometimes contains an example. Fully described examples may be found in [SAMP].

The requirements presented are not intended to be "orthogonal", that is, some of them repeat, or overlap, with others.

2.1 Authorization Information

2.1.1 **Authorization decisions MUST be able to be based on information about the requestor, the service/method requested, and the** operating environment (authorization information). AAA protocols are required to transport this information.

This simply states the requirement for a protocol and an access decision function, which takes inputs, based on the requestor, the resource requested and the environment.

2.1.2 **It MUST be possible to represent authorization information as sets of attributes. It MAY be possible to represent authorization** information as objects.

This states that authorization information must be decomposable into sets of attributes. It is not intended to imply any particular mechanism for representing attributes.

2.1.3 **It MUST be possible to package authorization information so that the authorization information for multiple services or** applications can be carried in a single message in a AAA or application protocol.

This states that a protocol, which always required separate AAA messages/transactions for each service/application, would not meet the requirement. For example, it should be possible for a single AAA message/transaction to be sufficient to allow both network and application access.

2.1.4 **Standard attributes types SHOULD be defined which are relevant**

to many Internet applications/services (e.g. identity information, group information, ...)

There are many attributes that are used in lots of contexts, and these should only be defined once, in order to promote interoperability and prevent duplication of effort.

2.1.5 Authorization decisions MUST NOT be limited to being based on identity information, i.e. AAA protocols MUST support the use of non-identifying information, e.g. to support role based access control (RBAC).

Authorization based on clearances, roles, groups or other information is required to be supported. A AAA protocol that only carried identity information would not meet the requirement.

2.1.6 Authorization data MAY include limits in addition to attributes which are directly "owned" by end entities.

This states that some attributes do not simply represent attributes of an entity, for example a spending limit of IRú1,000 is not an intrinsic attribute of an entity. This also impacts on the access decision function, in that the comparison to be made is not a simple equality match.

2.1.7 It MUST be possible for other (non-AAA) protocols to define their own attribute types, which can then be carried within an authorization package in a AAA or application protocol.

This states that the attributes that are significant in an authorization decision, may be application protocol dependent. For example, many attribute types are defined by [RFC2138] and support for the semantics of these attributes will be required. Of course, only AAA entities that are aware of the added attribute types can make use of them.

2.1.8 It SHOULD be possible for administrators of deployed systems to define their own attribute types, which can then be carried within an authorization package in a AAA or application protocol.

This states that the attributes that are significant in an authorization decision, may be dependent on a closed environment. For example, many organizations have a well-defined scheme of seniority, which can be used to determine access levels. Of course, only AAA entities that are aware of the added attribute types can make use of them.

2.1.9 It SHOULD be possible to define new attribute types without central administration and control of attribute name space.

A centralized or distributed registration scheme of some sort is needed if collisions in attribute type allocations are to be

avoided. However a AAA protocol which always requires use of such a centralized registration would not meet the requirement. Of course, collisions should be avoided where possible.

2.1.10 It MUST be possible to define attribute types so that an instance of an attribute in a single AAA message can have multiple values.

This states that a protocol which does not allow multiple instances of an attribute in a message/transaction would not meet the requirement. For example it should be possible to have a "group" attribute which contains more than one groupname (or number or whatever).

2.1.11 If MUST be possible to distinguish different instances of the same authorization attribute type or value, on the basis of "security domain" or "authority".

This recognizes that it is important to be able to distinguish between attributes based not only on their value. For example, all NT domains (which use the English language) have an Administrators group, an access decision function has to be able to determine to which of these groups the requestor belongs.

2.1.12 AAA protocols MUST specify mechanisms for updating the rules which will be used to control authorization decisions.

This states that a AAA protocol that cannot provide a mechanism for distributing authorization rules is not sufficient. For example, this could be used to download ACLs to a PDP.

Note that this is not meant to mean that this AAA protocol mechanism must always be used, simply that it must be available for use. In particular, storing authorization rules in a trusted repository (in many cases an LDAP server) will in many cases be used instead of such a AAA protocol mechanism. Neither does this requirement call for a standardized format for authorization rules, merely that there be a mechanism for transporting these.

2.1.13 The AAA protocol MUST allow for chains of AAA entities to be involved in an authorization decision.

This states that more than one AAA server may have to be involved in a single authorization decision. This may occur either due to a decision being spread across more than one "domain" or in order to distribute authorization within a single "domain".

2.1.14 The AAA protocol MUST allow for intermediate AAA entities to add their own local authorization information to a AAA request or response.

This states that where more than one AAA entity is involved in an authorization decision each of the AAA entities may manipulate the

AAA messages involved either by adding more information or by processing parts of the information.

2.1.15 AAA entities MAY be either be deployed independently or integrated with application entities.

This states that the AAA entities may either be implemented as AAA servers or integrated with application entities.

2.1.16 The AAA protocol MUST support the creation and encoding of rules that are to be active inside one AAA server based on attributes published by another AAA server. The level of authorization of the requesting AAA Server MAY govern the view on attributes.

This states that one AAA entity may have to distribute authorization rules to another, and that the AAA entity that receives the rules may only be seeing part of the story.

<<late entrant>>

2.1.17 AAA protocols MAY have to support the idea of critical and non-critical attribute types.

This is analogous to the use of the criticality flag in public key certificate extensions.

2.1.18 A AAA protocol MUST allow authorization rules to be expressed in terms of combinations of other authorization rules which have been evaluated.

For example, access may only be granted if the requestor is member of the backup users group and not a member of the administrator's group. Note that this requirement does not state which types of combinations are to be supported.

2.1.19 It SHOULD be possible to make authorization decisions based on the geographic location of a requestor, service or AAA entity.

This is just an example of an authorization attribute type, notable because it requires different underlying implementation mechanisms.

2.1.20 It SHOULD be possible to make authorization decisions based on the identity or the equipment used by a requestor, service or AAA entity.

This is just an example of an authorization attribute type, notable because it may require different underlying implementation mechanisms (if IPSec isn't available).

<<late entrant>>

2.1.21 When there are multiple instances of a given attribute, there must be an unambiguous mechanism by which a receiving peer can

determine the value of specified instance.

2.2 Security of authorization information

2.2.1 It MUST be possible for authorization information to be communicated securely in AAA and application protocols. Mechanisms that preserve authenticity, integrity and privacy for this information MUST be specified.

This states that there must be a well-defined method for securing authorization information, not that such methods must always be used. Whether support for these mechanisms is to be required for conformance is left open. In particular, mechanisms must be provided so that a service administrator in the middle of a chain cannot read or change authorization information being sent between other AAA entities.

2.2.2 AAA protocols MUST allow for use of an appropriate level of security for authorization information. AAA protocols MUST be able to support both highly secure and less secure mechanisms for data integrity/confidentiality etc.

It is important that AAA protocols do not mandate too heavy a security overhead, thus the security mechanisms specified donÆt always need to be used (though not using them may affect the authorization decision).

2.2.3 The security requirements MAY differ between different parts of a package of authorization information.

Some parts may require confidentiality and integrity, some may only require integrity. This effectively states that we require something like selective field security mechanisms. For example, information required to gain access to a network may have to be in clear, whilst information required for access to an application within that network may have to be encrypted in the AAA protocol.

2.2.4 AAA protocols MUST provide mechanisms that prevent intermediate administrators breaching security.

This is a basic requirement to prevent man-in-the-middle attacks, for example where an intermediate administrator changes AAA messages on the fly.

2.2.5 AAA protocols MUST NOT open up replay attacks based on replay of the authorization information.

For example, a AAA protocol should not allow flooding attacks where the attacker replays AAA messages that require the recipient to use a lot of CPU or communications before the replay is detected.

2.2.6 AAA protocols MUST be capable of leveraging any underlying peer entity authentication mechanisms that may have been applied - this

MAY provide additional assurance that the owner of the
authorization information is the same as the authenticated entity.

For example, if IPSec provides sufficient authentication, then it must be possible to omit AAA protocol authentication.

2.2.7 End-to-end confidentiality, integrity, peer-entity-authentication, or non-repudiation MAY be required for packages of authorization information.

This states that confidentiality, (resp. the other security services), may have to be provided for parts of a AAA message, even where it is transmitted via other AAA entities. It does allow that such a AAA message may also contain non-confidential, resp. the other security services), parts. In addition, intermediate AAA entities may themselves be considered end-points for end-to-end security services applied to other parts of the AAA message.

2.2.8 AAA protocols MUST be usable even in environments where no peer entity authentication is required (e.g. a network address on a secure LAN may be enough to decide).

This requirement (in a sense the opposite of 2.2.6), indicates the level of flexibility that is required in order to make the AAA protocol useful across a broad range of applications/services.

2.2.9 AAA protocols MUST specify "secure" defaults for all protocol options. Implementations of AAA entities MUST use these "secure" defaults unless otherwise configured/administered.

This states that the out-of-the-box configuration must be "secure", for example, authorization decisions should result in denial of access until a AAA entity is configured. Note that the interpretation of "secure" will vary on a case-by-case basis, though the principle remains the same.

2.3 Time

2.3.1 Authorization information MUST be timely, which means that it MUST expire and in some cases MAY be revoked before expiry.

This states that authorization information itself is never to be considered valid for all time, every piece of authorization information must have associated either an explicit or implicit validity period or time-to-live.

2.3.2 AAA protocols MUST provide mechanisms for revoking authorization information, in particular privileges.

Where the validity or time-to-live is long, it may be necessary to revoke the authorization information, e.g. where someone leaves a company. Note that this requirement does not mandate a particular

scheme for revocation, so that it is not a requirement for blacklists or CRLs.

2.3.3 A set of attributes MAY have an associated validity period - such that that the set MUST only be used for authorization decisions during that period. The validity period may be relatively long, (e.g. months) or short (hours, minutes).

This states that explicit validity periods are, in some cases, needed at the field level.

2.3.4 Authorization decisions MAY be time sensitive. Support for e.g. "working hours" or equivalent MUST be possible.

This states that the AAA protocol must be able to support the transmission of time control attributes, although it does not mandate that AAA protocols must include a standard way of expressing the "working hours" type constraint.

2.3.5 It MUST be possible to support authorization decisions that produce time dependent results.

For example, an authorization result may be that service should be provided for a certain period. In such cases a AAA protocol must be able to transport this information, possibly as a specific result of the authorization decision, or, as an additional "termination of service" AAA message transmitted later.

2.3.6 It MUST be possible to support models where the authorization information is issued in well in advance of an authorization decision rather than near the time of the authorization decision.

This is required in order to support pre-paid (as opposed to subscription) scenarios (e.g. for VoIP).

2.3.7 It SHOULD be possible to support models where the authorization decision is made in advance of a service request.

This is for some applications such as backup, where actions are scheduled for future dates. It also covers applications that require reservation of resources.

2.3.8 A AAA mechanism must allow time stamp information to be carried along with authorization information (e.g. for non-repudiation).

The PKIX WG is developing a time stamp protocol, which can be used as part of a non-repudiation solution. In some environments it may be necessary that certain AAA protocol messages are timestamped (by a trusted authority) and that the timestamps are forwarded within subsequent AAA messages.

2.4 Topology

2.4.1 AAA protocols MUST be able to support the use of the push, pull and agent models.

This states that a protocol that only supported one model, say pull, would not meet the requirements of all the applications. The models are defined in [FRMW].

2.4.2 In transactions/sessions, which involve more than one AAA entity, each æhopÆ MAY use a different push/pull/agent model.

For example, in the mobile IP case, a "foreign" AAA server might pull authorization information from a broker, whereas the broker might push some authorization information to a "home" AAA server.

2.4.3 AAA Protocols MUST cater for applications and services where the entities involved in the application or AAA protocols belong to different (security) domains.

This states that it must be possible for any AAA protocol message to cross security or administrative domain boundaries. Typically, higher levels of security will be applied when crossing such boundaries, and accounting mechanisms may also have to be more stringent.

2.4.4 AAA protocols MUST support roaming.

Roaming here may also be thought of as "away-from-home" operation. For example, this is a fundamental requirement for the mobile IP case.

2.4.5 AAA protocols SHOULD support dynamic mobility

Dynamic mobility here means that a client moves from one domain to another, without having to completely re-establish e.g. whatever AAA session information is being maintained.

2.4.6 An authorization decision MAY have to be made before the requestor has any other connection to a network.

For example, this means that the requestor canÆt go anywhere on the network to fetch anything and must do requests via an application/service or via an intermediate AAA entity. The AAA protocol should not overexpose such a server to denial-of-service attacks.

2.4.7 AAA protocols MUST support the use of intermediate AAA entities which take part in authorization transactions but which donÆt "own" any of the end entities or authorization data.

In some environments (e.g. roamops), these entities are termed brokers (though these are not the same as bandwidth brokers in the QoS environment).

2.4.8 AAA protocols MAY support cases where an intermediate AAA entity returns a forwarding address to a requestor or AAA entity,

in order that the requestor or originating AAA entity can contact another AAA entity.

This requirement recognizes that there will be routing issues with AAA servers, and that this requires that AAA protocols are able to help with such routing. For example, in the mobile IP case, a broker may be required, in part to allow the foreign and home AAA servers to get in contact.

2.4.9 **It MUST be possible for an access decision function to discover the AAA server of a requestor. If the requestor provides** information used in this discovery process then the access decision function MUST be able to verify this information in a trusted manner.

This states that not only do AAA servers have to be able to find one another, but that sometimes an application entity may have to find an appropriate AAA server.

2.5 Application Proxying

2.5.1 **AAA protocols MUST support cases where applications use proxies, that is, an application entity (C), originates a service** request to a peer (I) and this intermediary (I) also initiates a service request on behalf of the client (C) to a final target (T). AAA protocols MUST be such that the authorization decision made at T, MAY depend on the authorization information associated with C and/or with I. This "application proxying" must not introduce new security weaknesses in the AAA protocols. There MAY be chains of application proxies of any length.

Note that this requirement addresses application layer proxying - not chains of AAA servers. For example, a chain of HTTP proxies might each want to restrict the content they serve to the "outside". As the HTTP GET message goes from HTTP proxy to HTTP proxy, this requirement states that it must be possible that the authorization decisions made at each stage can depend on the user at the browser, and not say, solely on the previous HTTP proxyÆs identity. Of course there may only be a single AAA server involved, or there may be many.

2.5.2 **Where there is a chain of application proxies, the AAA protocol flows at each stage MAY be independent, i.e. the first hop may use** the push model, the second pull, the third the agent model.

This simply restates a previous requirement (no. 2.4.7), to make it clear that this also applies when application proxying is being used.

2.6 Trust Model

2.6.1 AAA entities MUST be able to make decisions about which other AAA entities are trusted for which sorts of authorization information.

This is analogous to a requirement in pubic key infrastructures: Just because someone can produce a cryptographically correct public key certificate doesnÆt mean that I should trust them for anything, in particular, I might trust the issuer for some purposes, but not for others.

2.6.2 AAA protocols MUST allow entities to be trusted for different purposes, trust MUST NOT be an all-or-nothing issue.

This relates the packaging (no. 2.1.3) and trust (no. 2.6.1) requirements. For example, a AAA entity may trust some parts of an authorization package but not others.

2.6.3 A confirmation of authorization MAY be required in order to initialize or resynchronize a AAA entity.

This states that a AAA entity may need to process some AAA protocol messages in order to initialize itself. In particular, a AAA entity may need to check that a previous AAA message remains "valid", e.g. at boot-time.

2.6.4 A negation of static authorization MAY be required to shut down certain services.

This is the converse of 2.6.5 above. It means that a AAA entity may be "told" by another that a previous AAA message is no longer "valid". See also 2.3.2 and 2.7.6.

2.6.5 It MUST be possible to configure sets of AAA entities that belong to a local domain, so that they are mutually trusting, but so that any external trust MUST be via some nominated subset of AAA entities.

This states that for efficiency or organizational reasons, it must be possible to set up some AAA servers through which all "external" AAA services are handled. It also states that it must be possible to do this without over-burdening the "internal-only" AAA servers with onerous security mechanisms, just because some AAA servers do handle external relations.

2.6.6 Intermediate AAA entities in a chain MUST be able to refuse a connection approved by an earlier entity in the chain.

For example, in mobile IP the home network may authorize a

connection, but the foreign network may refuse to allow the connection due to the settings chosen by the home network, say if the home network will refuse to pay.

2.6.7 It SHOULD be possible to modify authorization for resources while a session is in progress without destroying other session information.

For example, a "parent" AAA server should be able to modify the authorization state of sessions managed by a "child" AAA server, say by changing the maximum number of simultaneous sessions which are allowed.

2.7 Not just transactions

2.7.1 Authorization decisions MAY be context sensitive, AAA protocols MUST enable such decisions.

This states that AAA protocols need to support cases where the authorization depends, (perhaps even only depends), on the current state of the system, e.g. only seven sessions allowed, seventh decision depends on existence of six current sessions. Since the context might involve more than one service, the AAA protocol is likely to have to offer some support.

2.7.2 AAA protocols SHOULD support both the authorization of transactions and continuing authorization of sessions.

This states that AAA entities may have to maintain state and act when the state indicates some condition has been met.

2.7.3 Within a single session or transaction, it MUST be possible to interleave authentication, authorization and accounting AAA messages.

This states, that e.g. a session may have to use initial authentication, authorization and accounting AAA message(s), but also have to include e.g. re-authentication every 30 minutes, or a continuous "drip-drip" of accounting AAA messages.

2.7.4 Authorization decisions may result in a "not ready" answer.

This states that yes and no are not the only outcomes of an authorization decision. In particular, if the AAA entity cannot yet give a decision, it might have to return such a result. This is analogous to how public key certification requests are sometimes handled in PKI management protocols.

2.7.5 A AAA entity MAY re-direct a AAA request that it has received.

This states that if entity "a" asks "b", then "b" may say: "don't ask me, ask 'c'". This is analogous to HTTP re-direction (status code 307).

2.7.6 AAA protocols SHOULD allow a AAA entity to "take back" an authorization.

The expectation is that AAA protocols will support the ability of a AAA entity to signal an application or other AAA entity that an authorization (possibly previously granted by a third AAA entity) is no longer valid.

2.8 Administration

2.8.1 It MUST be possible for authorization data to be administered on behalf of the end entities and AAA entities.

This requirement indicates that administration of AAA has to be considered as part of protocol design - a AAA protocol, which required all AAA entities act independent of all other AAA entities, would not meet the requirement.

2.8.2 Centralizable administration of all features SHOULD be supported.

It should be possible (if it meets the domain requirements) to centralize or distribute the administration of AAA.

2.8.3 AAA protocols SHOULD support cases where the user (as opposed to an administrator) authorizes a transaction.

For example, a user might want to control anti-spam measures or authorize things like a purchase. In such cases, the user is acting somewhat like an administrator.

2.8.4 One AAA entity MAY create authorization rules for another AAA entity.

This is required to properly support delegation of authority, however when allowed, this must be able to be done in a secure fashion.

2.8.5 AAA protocols SHOULD support failure recovery when one AAA entity in a chain of AAA entities that maintain state about a session fails.

For example, in a network access situation it may be required that a AAA server which has crashed be able to determine how many sessions are in progress, in order to make the "next" authorization decision.

2.8.6 It SHOULD be possible for a AAA entity to query the authorization state of another AAA entity.

This may be required as part of a failure recovery procedure.

2.8.7 AAA protocols MUST be able to support "hot fail-over" for

server components without loss of state information.

This states that AAA protocols must be able to support cases where, when a server is no longer operable, a secondary server can

automatically be brought "live" without losing important state information.

2.9 Bytes on-the-wire

2.9.1 **Authorization separate from authentication SHOULD be allowed when necessary, but the AAA protocols MUST also allow for a single** message to request both authentication and authorization.

AAA protocols have to allow a split between authentication and authorization so that different mechanisms are used for each. This states that sometimes both types of information need to be carried in the same message.

2.9.2 **In order to minimize resource usage (e.g. reduce roundtrips) it MUST be possible to embed AAA PDUs into other protocols.**

This states that the AAA protocol authorization packages must be defined so that they can also be carried in other protocols. For example, depending on AAA protocol header information in order to reference an authorization package could cause a protocol to fail to meet the requirement.

2.9.3 **A AAA protocol MAY provide mechanisms for replication of state information.**

This can be required e.g. to support resiliency in cases where hot fail-over is required. Note that AAA protocols are of course, subject to normal protocol design requirements to do with reliability, no single-point-of-failure etc even though these are not all specified here.

2.9.4 **A AAA protocol SHOULD allow the possibility for implementation of a gateway function between the AAA protocol and other legacy** AAA related protocols.

For example, some form of support for [RFC2138] as a legacy protocol is very likely to be required. Of course, the use of such a gateway is almost certain to mean not meeting some other requirements, (e.g. end-to-end security), for transactions routed through the gateway. There is no implication that such gateway functionality needs to be a separate server.

2.9.5 **A AAA protocol MUST be able to support use of a wide range of primitive data types, including RFC2277.**

For example, various sized, signed and unsigned integers, possibly including multi-precision integers will almost certainly need to be

transported. Floating point support according to ANSI IEEE 754-1985 may also be required.

2.9.6 A AAA protocol transport SHOULD support being optimized for a long-term exchange of small packets in a stream between a pair of hosts.

NASes typically have a high number of transactions/second, so the AAA protocol MUST allow the flow of requests to be controlled in order for the server to make efficient use of it's receive buffers.

2.9.7 A AAA protocol MUST provide support for load balancing.

In the event that a peer's cannot receive any immediate requests, the AAA protocol MUST allow for an implementation to balance the load of requests among a set of peers.

2.10 Interfaces

2.10.1 It SHOULD be possible that authorization data can be used for application purposes.

For example, in web access, if the authorization data includes a group name, mechanisms to make this data available to the application so that it can modify the URL originally requested are desirable.

2.10.2 It SHOULD be possible that authorization data can be used to mediate the response to a request.

For example, with web access the clearance attribute value may affect the content of the HTTP response message.

2.10.3 AAA protocols SHOULD be able to operate in environments where requestors are not pre-registered (at least for authorization purposes, but possibly also for authentication purposes).

This is necessary to be able to scale a AAA solution where there are many requestors.

2.10.4 AAA protocols MUST be able to support a linkage between authorization and accounting mechanisms.

Motherhood and apple-pie.

2.10.5 AAA protocols MUST be able to support accountability (audit/ non-repudiation) mechanisms.

Sometimes, an authorization decision will be made where the requestor has not authenticated. In such cases, it must be possible that the authorization data used is linked to audit or other accountability mechanisms. Note that this requirement does not call

for mandatory support for digital signatures, or other parts of a non-repudiation solution.

2.11 Negotiation

2.11.1 AAA protocols MUST support the ability to refer to sets of authorization packages in order to allow peers negotiate a common set.

Given that peers may support different combinations of authorization attribute types and packages, the requirement states that protocol support is required to ensure that the peers use packages supported by both peers.

2.11.2 It MUST be possible to negotiate authorization packages between AAA entities that are not in direct communication.

This states that where, e.g. a broker is involved, the end AAA servers might still need to negotiate.

2.11.3 Where negotiation fails to produce an acceptable common supported set then access MUST be denied.

For example, a server cannot grant access if it cannot understand the attributes of the requestor.

2.11.4 Where negotiation fails to produce an acceptable common supported set then it SHOULD be possible to generate an error indication to be sent to another AAA entity.

If negotiation fails, then some administrator intervention is often required, and protocol support for this should be provided.

2.11.5 It MUST be possible to pre-provision the result of a negotiation, but in such cases, the AAA protocol MUST include a confirmation of the "negotiation result".

Even if the supported packages of a peer are configured, this must be confirmed before assuming both sides are similarly configured.

2.11.6 For each application making use of a AAA protocol, there MUST be one inter-operable IETF standards-track specification of the authorization package types that are "mandatory to implement".

This requirement assures that communicating peers can count on finding at least one IETF specified inter-operable AAA protocol dialect provided they are doing authorization for a common application specific problem domain. This does not preclude the negotiation of commonly understood but private AAA protocol authorization package types (e.g. vendor specific).

2.11.7 It SHOULD also be possible to rank AAA negotiation options in order of preference.

This states that, when negotiating, peers must be able to indicate preferences as well as capabilities.

2.11.8 The negotiation mechanisms used by AAA protocols SHOULD NOT be vulnerable to a "bidding-down" attack.

A "bidding-down" attack is where an attacker forces the negotiating parties to choose the "weakest" option available. This is analogous to forcing 40-bit encryption on a link. The requirement highlights that protocol support is needed to prevent such attacks, for example by including the negotiation messages as part of a later MAC calculation, if authentication has produced a shared secret.

<<late entrant>>

2.11.9 A peer MUST NOT send an attribute within an authorization package or attribute that was not agreed to by a prior successful negotiation. If this AAA protocol violation occurs, then it MUST be possible to send an error indication to the misbehaving peer, and generate an error indication to the network operator.

<<late entrant>>

2.11.10 A peer MUST declare all of the sets of the authorization packages that it understands in its initial negotiation bid message.

3. Security Considerations

This document includes specific security requirements.

This document does not state any detailed requirements for the interplay with authentication, accounting or accountability (audit). A AAA protocol, which meets all of the above requirements, may still leave vulnerabilities due to such interactions. Such issues must be considered as part of AAA protocol design.

4. References

[FRMW] Vollbrecht et al., "AAA Authorization Framework", draft-ietf-aaa-authz-arch-00.txt, October 1999.

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", RFC 2026, BCP 9, October 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, March 1997.

[RFC2138] Rigney, C., et al., "Remote Authentication Dial In User Service (RADIUS)", RFC2138, April 1997.

[RFC2277] Alvestrand, H., "IETF Policy on Character Sets and Languages", RFC2277, January 1998.

[SAMP] Vollbrecht et al., "AAA Authorization Application Examples", draft-ietf-aaa-authz-samp-00.txt, October 1999.

Author's Addresses

Stephen Farrell
Baltimore Technologies
61/62 Fitzwilliam Lane
Dublin 2,
IRELAND
Phone: +353-1-647-7300
Fax: +353-1-647-7499
stephen.farrell@baltimore.ie

John R. Vollbrecht
Merit Network, Inc.
4251 Plymouth Rd., Suite 2000
Ann Arbor, MI 48105
USA
Phone: +1 734 763 1206
Fax: +1 734 647 3745
EMail: jrv@merit.edu

Pat R. Calhoun
Network and Security Research
Center, Sun Labs
Sun Microsystems, Inc.
15 Network Circle
Menlo Park, California, 94025
USA
Phone: +1 650 786 7733
Fax: +1 650 786 6445
EMail: pcalhoun@eng.sun.com

Leon Gommans
Cabletron Systems EMEA
Kerkplein 24
2841 XM Moordrecht
The Netherlands
Phone: +31 182 379278
Email: gommans@cabletron.com

George M. Gross
Lucent Technologies
184 Liberty Corner Road, m.s.
LC2N-D13
Warren, NJ 07059
USA
Phone: +1 908 580 4589
Fax: +1 908 580 7430
Email: gmgross@lucent.com

Betty de Bruijn
Interpay Nederland B.V.
Eendrachtlaan 315
3526 LB Utrecht
The Netherlands
Phone: +31 30 2835104
Email: betty@euronet.nl

Cees T.A.M. de Laat
Physics and Astronomy dept.
Utrecht University
Pincetonplein 5,
3584CC Utrecht
Netherlands
Phone: +31 30 2534585
Phone: +31 30 2537555
EMail: delaat@phys.uu.nl

Matt Holdrege
Lucent Technologies
1701 Harbor Bay Pkwy.
Alameda, CA 94502
USA
Phone: +1 510 747 2711
Email: holdrege@lucent.com

David W. Spence
Merit Network, Inc.
4251 Plymouth Rd., Suite 2000
Ann Arbor, MI 48105
USA

Phone: +1 734 615 2630
Fax: +1 734 647 3745
EMail: dwspence@merit.edu

Full Copyright Statement