Network Working Group
Request for Comments: 2903
Category: Experimental

C. de Laat
Utrecht University
G. Gross
Lucent Technologies
L. Gommans
Enterasys Networks EMEA
J. Vollbrecht
D. Spence
Interlink Networks, Inc.

August 2000

Generic AAA Architecture

Status of this Memo

Copyright Notice

Abstract

This memo proposes an Authentication, Authorization, Accounting (AAA) architecture that would incorporate a generic AAA server along with an application interface to a set of Application Specific Modules that could perform application specific AAA functions. A separation of AAA functions required in a multi-domain environment is then proposed using a layered protocol abstraction. The long term goal is to create a generic framework which allows complex authorizations to be realized through a network of interconnected AAA servers.

Table of Contents

1. Introduction

The work for this memo was done by a group that originally was the Authorization subgroup of the AAA Working Group of the IETF. When the charter of the AAA working group was changed to focus on MobileIP and NAS requirements, the AAAarch Research Group was chartered within the IRTF to continue and expand the architectural work started by the Authorization subgroup. This memo is one of four which were created by the subgroup. This memo is a starting point for further work within the AAAarch Research Group. It is still a work in progress

and is published so that the work will be available for the AAAarch subgroup and others working in this area, not as a definitive description of architecture or requirements.

The authorization subgroup of the AAA Working Group proposed an "AAA Authorization Framework" [2] illustrated with numerous application examples [3] which in turn motivates a proposed list of authorization requirements [4]. This memo builds on the framework presented in [2] by proposing an AAA infrastructure consisting of a network of cooperating generic AAA servers communicating via a standard protocol. The protocol should be quite general and should support the needs of a wide variety of applications requiring AAA functionality. To realize this goal, the protocol will need to operate in a multi-domain environment with multiple service providers as well as entities taking on other AAA roles such as User Home Organizations and brokers. It should be possible to combine requests for multiple authorizations of different types in the same authorization transaction. The AAA infrastructure will be required to forward the components of such requests to the appropriate AAA servers for authorization and to collect the authorization decisions from the various AAA servers consulted. All of this activity is perfectly general in nature and can be realized in the common infrastructure.

But the applications requiring AAA services will each have their own unique needs. After a service is authorized, it must be configured and initialized. This will require application specific knowledge and may require application specific protocols to communicate with application specific service components. To handle these application specific functions, we propose an application interface between a generic AAA server and a set of one or more Application Specific Modules (ASMs) which can carry out the unique functionality required by each application.

Since the data required by each application for authentication, authorization, or accounting may have unique structure, the standard AAA protocol should allow the encapsulation of opaque units of Application Specific Information (ASI). These units would begin with a standard header to allow them to be forwarded by the generic infrastructure. When delivered to the final destination, an ASI unit would be passed by a generic AAA server across its program interface to an appropriate ASM for application specific processing. Nevertheless, it remains a goal of the design for information units to be encoded in standard ways as much as possible so as to enable processing by a generic rule based engine.

The interactions of the generic AAA server with the Application Specific Modules and with each other to realize complex AAA functions is explored in section 2. Then, in section 3, we attempt to further organize the AAA functions into logical groups using a protocol layering abstraction. This abstraction is not intended to be a reference model ready to be used for protocol design. At this point in the work, there are numerous questions that need to be addressed and numerous problems that remain to be solved. It may be that an abstraction other than layering will prove to be more useful or, more likely, that the application layer will require some substructure of its own.

Finally, in section 4, we show how the security requirements identified in [4] can be met in the generic server and the Application Specific Modules by applying security techniques such as public key encryption or digital signatures to the Application Specific Information units individually, so that different stakeholders in the AAA server network can protect selected information units from being deciphered or altered by other stakeholders in an authentication, authorization, or accounting chain.

2. Generic AAA Architecture

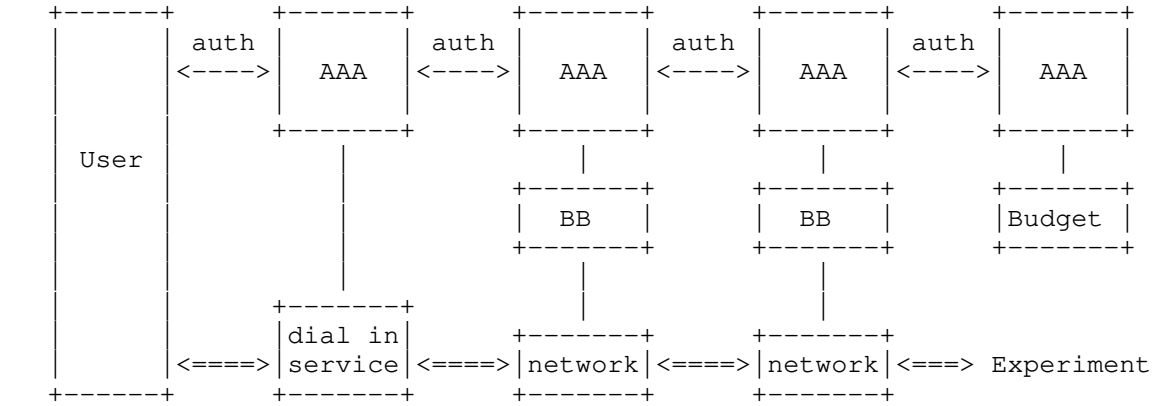
For the long term we envision a generic AAA server which is capable of authenticating users, handling authorization requests, and collecting accounting data. For a service provider, such a generic AAA server would be interfaced to an application specific module which manages the resource for which authorization is required. Generic AAA components would also be deployed in other administrative domains performing authorization functions.

2.1. Architectural Components of a Generic AAA Server

2.1.1. Authorization Rule Evaluation

The first step in the authorization process is for the user or an entity operating on the user's behalf to submit a well-formatted request to an AAA server. A generic AAA server has rules (logic and/or algebraic formulas) to inspect the request and come to an authorization decision. The first problem which arises is that Application Specific Information (ASI) has to be separated from the underlying logic for the authorization. Ideally the AAA server would have a rule based engine at this point which would know the logic rules and understand some generic information in the request, but it would not know anything about application specific information except where this information can be evaluated to give a boolean or numerical value. It should be possible to create rules that refer to

data elements that were not considered when the application was created. For example, one could request to do a remote virtual control room experiment from home using a dialin provider. The request would only be successful if the dialin access server allows it and if there is bandwidth available (bandwidth broker) and if the experimenter has the money to pay for it (E-Commerce). Possibly the people who specified the bandwidth broker protocol did not think of combining quality of service with a network service authorization in a single AAA request, but this generic model would allow it.



user <-> dialin <-> backbone with BB <-> <remote experiment>

Fig. 1 -- Example of a Multi Domain Multi Type of Server Request

2.1.2. Application Specific Module (ASM)

Ultimately an AAA server needs to interact with an application specific module (ASM). In a service provider, the ASM would manage resources and configure the service equipment to provide the authorized service. It might also involve itself in the authorization decision because it has the application specific knowledge required. A user home organization (UHO) may require ASMs as well, to perform application specific user authorization functions. For example, a UHO ASM might be required to access certain application specific databases or interpret application specific service level specifications.

Whatever the role of an administration relative to an authorization decision, the capabilities of the generic AAA server and the interface between it and the ASMs remains the same. This interface may be an Application Program Interface (API) or could even be a protocol based interface. In this model, however, the application

specific module is regarded as as separate architectural component from the generic AAA server. As such, it must be addressable and must therefore be part of a global naming space.

2.1.3. Authorization Event Log

For auditing purposes, the generic server must have some form of database to store time-stamped events which occur in the AAA server. This database can be used to account for authorizations which were given, but it can also be used in rules. One can imagine rules in which an authorization is only given if some other event was logged in the past. With the aid of certificates, this database could support non-repudiation.

2.1.4. Policy Repository

A database containing the available services and resources about which authorization decisions can be made and the policy rules to make them is also needed. Here too, the naming space for the services and resources is important since they must be addressable from other AAA servers to be able to build complex authorization requests.

2.1.5. Request Forwarding

Due to the multiple administrative domain (multi-kingdom) nature of the AAA problem, a mechanism to forward messages between AAA servers is needed. The protocol by which two AAA servers communicate should be a peer-to-peer protocol.

2.2. Generic AAA Server Model

With the implementation of the above mentioned components, the AAA server would be able to handle AAA requests. It would inspect the contents of the request, determine what authorization is requested, retrieve the policy rules from the repository, perform various local functions, and then choose one of the following options to further process each of the components of the request:

a) Let the component be evaluated by an attached ASM.

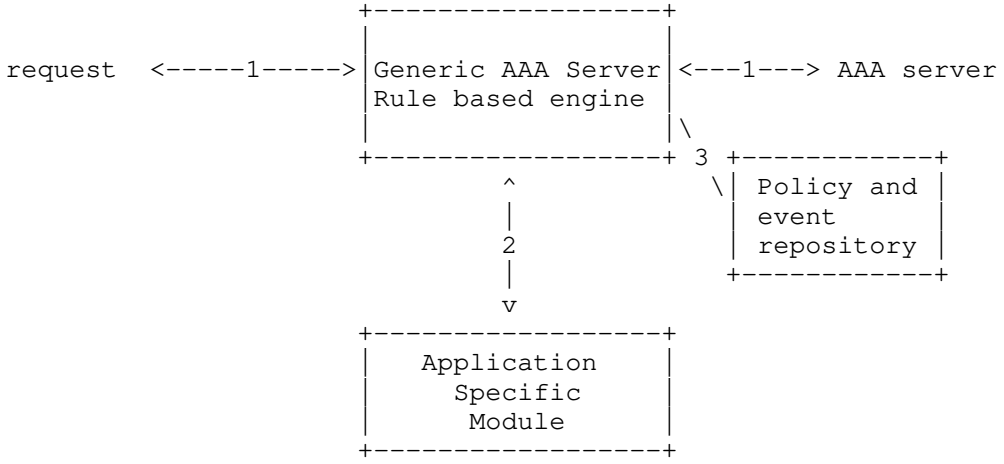
b) Query the authorization event log or the policy repository for the answer.

c) Forward the component(s) to another AAA server for evaluation.

In the following sections we present the generic model.

2.2.1. Generic AAA Server Interactions

Figure 2 illustrates a generic AAA Server with connections to the various architectural components described above. In this model, the user or another AAA server contacts the AAA server to get authorization, and the AAA server interacts with the service. The request is sent to the AAA server using the future AAA protocol. The server interacts with the service via a second protocol which we have labeled as type "2" in the figure. We say no more of the type 2 protocol than that it must support some global naming space for the application specific items. The same holds for the type 3 communication used to access the repository.



The numbers in the links denote types of communication.

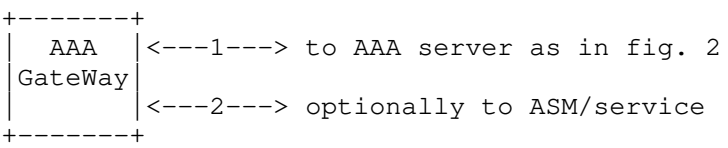
Fig. 2 -- Generic AAA Server Interactions

2.2.2. Compatibility with Legacy Protocols

Because of the widespread deployment of equipment that implements legacy AAA protocols and the desire to realize the functionality of the new AAA protocol while protecting the investment in existing infrastructure, it may be useful to implement a AAA gateway function that can encapsulate legacy protocol data units within the messages of the new protocol. Use of this technique, for example, would allow Radius attribute value pairs to be encapsulated in Application Specific Information (ASI) units of the new protocol in such a way that the ASI units can be digitally signed and encrypted for end-to-end protection between a service provider's AAA server and a home AAA server communicating via a marginally trusted proxy AAA server. The service provider's NAS would communicate via Radius to the service

provider's AAA server, but the AAA servers would communicate among themselves via the new AAA protocol. In this case, the AAA gateway would be a software module residing in the service provider's AAA server. Alternatively the AAA gateway could be implemented as a standalone process.

Figure 3 illustrates an AAA gateway. Communication type 4 is the legacy protocol. Communication type 1 is the future standard AAA protocol. And communication type 2 is for application specific communication to Application Specific Modules (ASMs) or Service Equipment.



The numbers in the links denote types of communication.

Fig. 3 -- AAA Gateway for Legacy AAA Protocols

2.2.3. Interaction between the ASM and the Service

In a service provider, the Application Specific Module (ASM) and the software providing the service itself may be tightly bound into a single "Service Application". In this case, the interface between them is just a software interface. But the service itself may be provided by equipment external to the ASM, for example, a router in the bandwidth broker application. In this case, the ASM communicates with the service via some protocol. These two possibilities are illustrated in figure 4. In both cases, we have labeled the communication between the ASM and the service as communication type 5, which of course, is service specific.

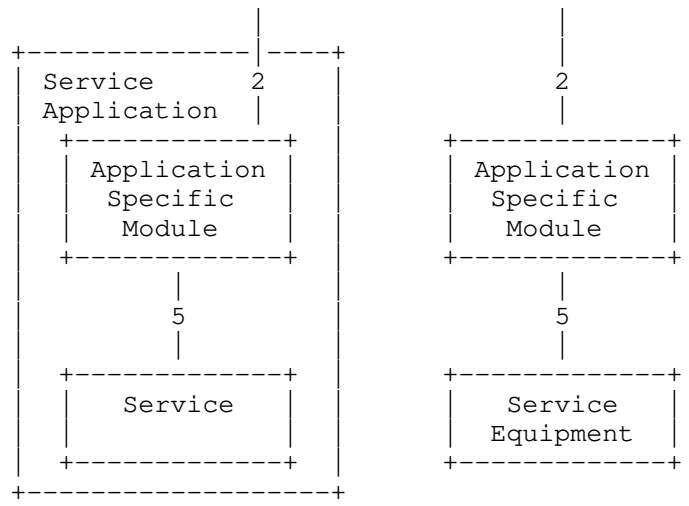
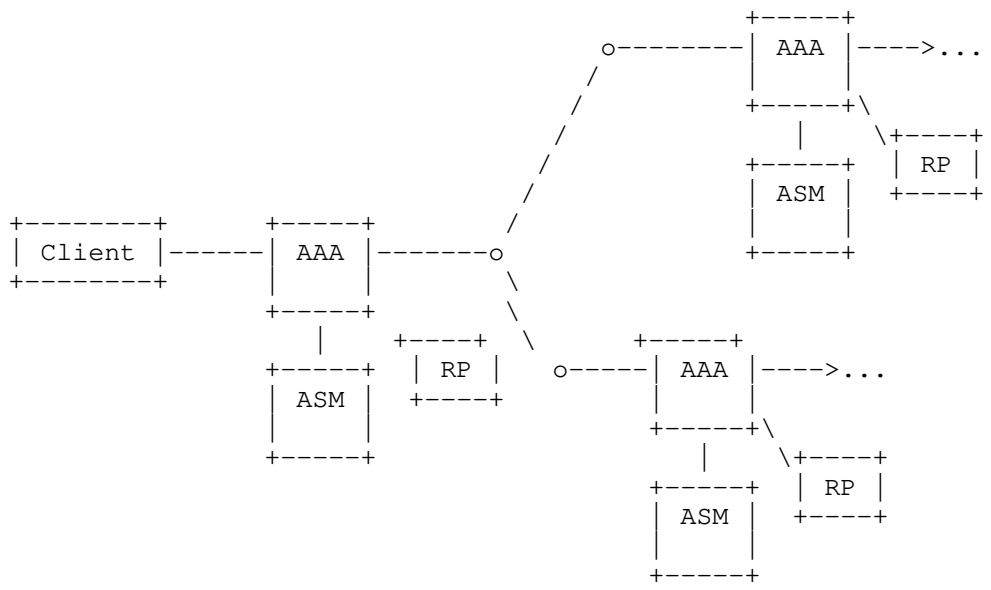


Fig. 4 -- ASM to Service Interaction (two views)

2.2.4. Multi-domain Architecture

The generic AAA server modules can use communication type 1 to contact each other to evaluate parts of requests. Figure 5 illustrates a network of generic AAA servers in different administrative domains communicating via communication type 1.



The AAA servers use only communication type 1 to communicate.
ASM = Application Specific Module
RP = Repository

Fig. 5 -- Multi-domain Multi-type of Service Architecture

2.3. Model Observations

Some key points of the generic architecture are:

1) The same generic AAA server can function in all three authorization models: agent, pull, and push [2].

2) The rule based engine knows how to evaluate logical formulas and how to parse AAA requests.

3) The Generic AAA server has no knowledge whatsoever about the application specific services so the application specific information it forwards is opaque to it.

4) Communication types 1, 2, and 3 each present their own naming space problems. Solving these problems is fundamental to forwarding AAA messages, locating application specific entities, and locating applicable rules in the rule repositories.

5) A standard AAA protocol for use in communication type 1 should be a peer-to-peer protocol without imposing client and server roles on the communicating entities.

6) A standard AAA protocol should allow information units for multiple different services belonging to multiple different applications in multiple different administrative domains to be combined in a single AAA protocol message.

2.4. Suggestions for Future Work

It is hoped that by using this generic model it will be feasible to design a AAA protocol that is "future proof", in a sense, because much of what we do not think about now can be encoded as application specific information and referenced by policy rules stored in a policy repository. From this model, some generic requirements arise that will require some further study. For example, suppose a new user is told that somewhere on a specific AAA server a certain authorization can be obtained. The user will need a AAA protocol that can:

1) send a query to find out which authorizations can be obtained from a specific server,

2) provide a mechanism for determining what components must be put in an AAA request for a specific authorization, and

3) formulate and transmit the authorization request.

Some areas where further work is particularly needed are in identifying and designing the generic components of a AAA protocol and in determining the basis upon which component forwarding and policy retrieval decisions are made.

In addition to these areas, there is a need to explore the management of rules in a multi-domain AAA environment because the development and future deployment of a generic multi-domain AAA infrastructure is largely dependent on its manageability. Multi-domain AAA environments housing many rules distributed over several AAA servers quickly become unmanageable if there is not some form of automated rule creation and housekeeping. Organizations that allow their services to be governed by rules, based on some form of commercial contract, require the contract to be implemented with the least

possible effort. This can, for example, be achieved in a scalable fashion if the individual user or user organization requesting a service is able to establish the service itself. This kind of interaction requires policy rule establishment between AAA servers belonging to multiple autonomous administrative domains.

3. Layered AAA Protocol Model

In the previous section, we proposed the idea of a generic AAA server with an interface to one or more Application Specific Modules (ASMs). The generic server would handle many common functions including the forwarding of AAA messages between servers in different administrative domains. We envision message transport, hop-by-hop security, and message forwarding as clearly being functions of the generic server. The application specific modules would handle all application specific tasks such as communication with service equipment and access to special purpose databases. Between these two sets of functions is another set of functions that presumably could take place in either the generic server or an ASM or possibly by a collaboration of both. These functions include the evaluation of authorization rules against data that may reside in various places including attributes from the authorization request itself. The more we can push these functions down into the generic server, the more powerful the generic server can be and the simpler the ASMs can be.

One way of organizing the different functions mentioned above would be to assign them to a layered hierarchy. In fact, we have found the layer paradigm to be a useful one in understanding AAA functionality. This section explores the use of a layered hierarchy consisting of the following AAA layers as a way of organizing the AAA functions:

Application Specific Service Layer
Presentation Service Layer
Transaction/Session Management Service Layer
Reliable/Secure Transport Service Layer

Nevertheless, the interface between the generic AAA server and the ASMs proposed in the previous section may be more complex than a simple layered model would allow. Even the division of functionality proposed in this section goes beyond a strict understanding of layering. Therefore this paper can probably best be understood as the beginnings of a work to understand and organize the common functionality required for a general purpose AAA infrastructure rather than as a mature reference model for the creation of AAA protocols.

In our view of AAA services modeled as a hierarchy of service layers, there is a set of distributed processes at each service layer that cooperate and are responsible for implementing that service layer's functions. These processes communicate with each other using a protocol specialized to carry out the functions and responsibilities assigned to their service layer. The protocol at service layer n communicates to its peers by depending on the services available to it from service layer n-1. The service layer n also has a protocol end point address space, through which the peer processes at service layer n can send messages to each other. Together, these AAA service layers can be assembled into an AAA protocol stack.

The advantage of this approach is that there is not just one monolithic "AAA protocol". Instead there is a suite of protocols, and each one is optimized to solve the problems found at its layer of the AAA protocol stack hierarchy.

This approach realizes several key benefits:

- The protocol used at any particular layer in the protocol stack can be substituted for another functionally equivalent protocol without disrupting the services in adjacent layers.

- Requirements in one layer may be met without impact on protocols operating in other layers. For example, local security requirements may dictate the substitution of stronger or weaker "reliable secure transport" layer security algorithms or protocols. These can be introduced with no change or awareness of the substitution by the layers above the Reliable/Secure Transport layer.

- The protocol used for a given layer is simpler because it is focused on a specific narrow problem that is assigned to its service layer. In particular, it should be feasible to leverage existing protocol designs for some aspects of this protocol stack (e.g. CORBA GIOP/CDR for the presentation layer).

- A legacy AAA protocol message (e.g. a RADIUS message) can be encapsulated within the protocol message(s) of a lower layer protocol, preserving the investment of a Service Provider or User Home Organization in their existing AAA infrastructure.

- At each service layer, a suite of alternatives can be designed, and the service layer above it can choose which alternative makes sense for a given application. However, it should be a primary goal of the AAA protocol standardization effort to specify one mandatory to implement protocol at the AAA Transaction/Session Management (AAA-TSM) service layer (see section 3.4).

3.1. Elements of a Layered Architecture

At each layer of a layered architecture, a number of elements need to be defined. These elements are discussed in the following sections.

3.1.1. Service Layer Abstract Interface Primitives

The service layer n is assumed to present a program interface through which its adjacent service layer n+1 can access its services. The types of abstract program service primitives and associated parameters exchanged across the boundary between these service layers must be specified.

3.1.2. Service Layer Peer End Point Name Space

Each service layer is treated as a set of cooperating processes distributed across multiple computing systems. The service layer must manage an end point name space that identifies these peer processes. The conventions by which a service layer assigns a unique end point name to each such peer process must be specified.

3.1.3. Peer Registration, Discovery, and Location Resolution

Along with defining an end point name space, a service layer must also specify how its peers:

- announce their presence and availability,

- discover one another when they first begin operation, and

- detect loss of connectivity or service withdrawal.

It is also necessary to specify what mechanisms, if any, exist to resolve a set of service layer specific search attributes into one or more peer end point names that match the search criteria.

3.1.4. Trust Relationships Between Peer End Points

Once an end point has established its initial contact with another peer, it must decide what authentication policy to adapt. It can trust whatever authentication was done on its behalf by a lower service layer or, through a pre-provisioning process, implicitly trust the peer, or else go through an authentication process with its peer. The supported mechanisms for establishing a service layer's end point trust relationships must be specified.

3.1.5. Service Layer Finite State Machine

To the extent that a service layer's internal states are externally visible, the layer's behavior in terms of a Finite State Machine (FSM) should be specified. Events that can drive the FSM state transitions may include:

- service layer n+1 interface primitive requests
- protocol data unit arrivals from peer service layer n end points received through the layer n-1 access point
- service layer n-1 interface primitives (e.g. call backs or interrupts)
- timer expirations

3.1.6. Protocol Data Unit Types

Each service layer defines a lexicon of protocol data units (PDUs) that communicate between the layer's peer processes the information that controls and/or monitors that service layer's distributed state and allows the service processes of that layer to perform their functions. Embedded in the PDUs of each layer are the PDUs of the higher layers which depend on its services. The PDUs of each service layer must be specified.

3.2. AAA Application Specific Service Layer

AAA applications have almost unlimited diversity, but imposing some constraints and commonality is required for them to participate in this generic AAA architectural framework. To satisfy these constraints, participating AAA applications would derive their application specific program logic from a standardized "Authorization Server" abstract base object class. They would also support an "Authorized Session" object class. An Authorization Session object instance represents an approved authorization request that has a long-lived allocation of services or resources. The generic AAA architecture could be extended to include other abstract base object classes in the future (e.g. Authorization Reservation, Authentication Server, etc.). How to implement the derived Authorization Server class's public methods for a given problem domain is entirely up to the application. One technique might be to place a software "wrapper" around an existing embedded application specific service to adapt it to the standardized Authorization Server object paradigm. The major Authorization Server class methods are:

- Publish an advertisement that describes the Authorization Server's service attributes and its application specific service layer end point address. Once the Authorization Server has registered, peer processes can discover its presence or send messages addressed to it.

- Application Specific Authorization Decision Function (AS-ADF) method takes a User's application specific authorization request and returns a decision of approve, deny, or conditionally approve with referral to another stakeholder. In the latter case, the application may create a reservation for the requested services or resources. This method represents the "condition" side of a policy rule's condition/action pair.

- Commit a service or set of resources to a previously conditionally approved authorization decision. For those authorization requests that have a long-term lifecycle (as opposed to being transactions), this method mobilizes a reservation into an Authorized Session object instance. This method represents the "action" side of a policy rule's condition/action pair.

- Cancel a previously conditionally approved Authorization request. This method releases any associated reservations for services or resources.

- Withdraw the Authorization Server's service advertisement.

A key motivation for structuring an AAA application as an Authorization Server object instance is to separate the generic authorization decision logic from the application-specific authorization decision logic. In many cases, the application can be divorced from the AAA problem altogether, and its AAA responsibility can be assigned to an external rules based generic AAA Server. (The idea is similar to that of a trust management policy server as defined in [5].) This would facilitate a security administrator deploying AAA policy in a central repository. The AAA policy is applied consistently across all users of the applications, resources, and services controlled by the AAA server. However, it is recognized that for many problem domains, there are unique rules intrinsic to the application. In these cases, the generic AAA Server must refer the User's authorization request to the relevant Application Specific Module.

3.3. Presentation Service Layer

The presentation service layer solves the data representation problems that are encountered when communicating peers exchange complex data structures or objects between their heterogeneous

computing systems. The goal is to transfer semantically equivalent application layer data structures regardless of the local machine architecture, operating system, compiler, or other potential inter-system differences.

One way to better understand the role of the presentation layer is to evaluate an existing example. The Generic Inter-ORB Protocol (GIOP) and its Common Data Representation (CDR) is a presentation service layer protocol developed by the Object Management Group (OMG) industry consortium. GIOP is one component within the Common Object Request Broker Architecture (CORBA). Peer Object Request Brokers (ORB) executing on heterogeneous systems use GIOP to invoke remote CORBA object interface methods. GIOP encodes an object method's input and output parameters in the Common Data Representation (CDR). While there are other presentation service layer protocols in the industry, GIOP in combination with CDR represents a mature, comprehensive solution that exhibits many of the presentation service layer requirements that are applicable within the AAA protocol model.

In the context of Internet access AAA protocols, RADIUS and its successors use the Attribute Value Pair (AVP) paradigm as the presentation service layer encoding scheme. While such an approach is versatile, it is also prone to becoming splintered into many ad hoc and vendor specific dialects. There is no structure imposed or method to negotiate the constraints on which AVPs are combined and interpreted for a given conversation in a consistent way across AAA protocol implementations or problem domains. At run-time, it can be hard for the communicating peers to negotiate to a common inter-operable set of AVPs.

To avoid this pitfall, a primary presentation service layer responsibility is the ability to let peers negotiate from a base Authorization Server object class towards a commonly understood derived Authorization Server object class that both presentation service layer peers have implemented for their application specific problem domain. This negotiation implies a requirement for a globally registered and maintained presentation service layer hierarchy of Authorization Server object class names.

3.4. AAA Transaction/Session Management Service Layer

The AAA Transaction/Session Management (AAA-TSM) service layer is a distributed set of AAA Servers, which typically reside in different administrative domains. Collectively they are responsible for the following three services:

Authentication -- Execute the procedure(s) needed to confirm the identity of the other parties with which the AAA TSM entity has a trust relationship.

Authorization -- Make an authorization decision to grant or deny a User's request for services or resources. The generic rules based policy engine described earlier in this document executes the authorization decision function. When the User's request is instantaneous and transient, then its authorization approval is treated as an ephemeral transaction. If the authorization approval implies a sustained consumption of a service or resources, then the request is transformed into an Authorized Session. For the duration of the Authorized Session's lifetime:

- its state may be queried and reported, or
- it may be canceled before service is completed, or
- the service being delivered may be modified to operate under new parameters and conditions, or
- the service may complete on its own accord.

In each of these cases, the AAA-TSM service layer must synchronize the Authorized Session's distributed state across all of those AAA Servers which are implementing that specific Authorized Session.

Accounting -- Generate any relevant accounting information regarding the authorization decision and the associated Authorized Session (if any) that represents the ongoing consumption of those services or resources.

The peer AAA servers and their AAA-TSM end points exchange AAA-TSM messages to realize these AAA functions. A central AAA-TSM concept is that there is a set of one or more AAA Server stakeholders who are solicited to approve/disapprove a User request for application layer services. The AAA-TSM service layer routes the User's request from one stakeholder to the next, accumulating the requisite approvals until they have all been asked to make an authorization decision.

The AAA Servers may also do User authentication (or re-authentication) as part of this approval process. The overall flow of the routing from one stakeholder to another may take the form of the "push", "pull", or "agent" authorization models developed in [2]. However, in principle, it is feasible to have an arbitrary routing path of an AAA-TSM authorization request among stakeholders. Once the final approval is received, the AAA-TSM service layer commits the requested service by notifying all of those stakeholders that require

a confirmation (i.e. turn on a pending reservation and do a transaction commit). Alternatively, any stakeholder among those on the consent list can veto the authorization request. In that case, all stakeholders who previously approved the request and had asked for a confirmation are told that the request has been denied (i.e., cancel reservation and do a transaction rollback).

The AAA-TSM authorization request payload must carry its own "Context State", such that when an AAA server receives it, there is sufficient information that it is essentially self-contained. Embedding the Context State within the AAA-TSM message provides two benefits. First, the message can be immediately processed with respect to the AAA Server's local policy, and this minimizes or altogether avoids the need for the AAA Server to exchange additional AAA-TSM messages with its peers to complete its piece of the overall authorization decision. The other benefit is that the AAA Server minimizes the amount of state information resources that it commits to a user's pending request until it is fully approved. This helps protect against denial of service attacks.

One can envision many possible message elements that could be part of the Context State carried within an AAA-TSM request message:

- AAA-TSM session identifier, a unique handle representing this authorization request. All AAA servers who participate in a request's approval process and its subsequent monitoring throughout its Session lifetime refer to this handle.

- permission lists stating which AAA Servers are allowed to modify which parts of the message.

- User's authorization request, encoded as a presentation layer PDU.

- User authentication information, (e.g. an X.509 public key certificate).

- User credentials information, or else a pointer to where that information can be found by an AAA server. An example of such credentials would be an X.509 attributes certificate.

- the list of AAA Server stakeholders who have yet to be visited to gain full approval of the User's authorization request. Each element in that list contains a presentation layer message encoding how the user authorization request should be evaluated by its application specific Authorization Decision Function (ADF).

- the current position in the list of AAA Server stakeholders to be visited.

- a list of those AAA servers which have already conditionally approved the User's authorization request, but which have predicated their approval on the request also completing its approval from those stakeholders who have not yet seen the request. Each element in the list has a digital signature or comparable mechanism by which their approval can be subsequently verified.

- an expiration time stamp, expressed in a universally understood time reference, which sets a lifetime limit on the AAA-TSM message's validity. This offers some replay attack protection, and inhibits messages from circulating indefinitely seeking the completion of a request's approval.

- a message payload modification audit trail, tracing which parties introduced changes into the User's authorization request terms and conditions.

- an AAA-TSM message integrity check, computed across the whole message rather than its individual elements, and signed by the most recent AAA-TSM layer end point process to modify the AAA-TSM message before its transmission to its AAA-TSM peer. This function may be delegated to the underlying Reliable Secure Transport layer connection to that destination peer.

3.5. AAA-TSM Service Layer Program Interface Primitives

The AAA-TSM service layer and its adjacent presentation service layer communicate across their boundary through a set of program interface primitives. A key design goal is to keep these primitives the same regardless of the higher level AAA application, analogous to a callable "plug-in". The two service layers are responsible for coordinating their state information. This responsibility includes all of the pending Authorization requests and the Authorization Sessions that they are both controlling and monitoring. The initial contact between these two layers is through an abstract object that is called an AAA-TSM Service Access Point (SAP). A particular service instance between these two layers is realized in an abstract object that is called an Authorized Session. The presentation service layer invokes AAA-TSM interface primitives against an AAA-TSM SAP.

The AAA-TSM service layer interface primitives can be broadly characterized as follows:

- Register a presentation end point address identifier and its associated set of attributes to a service access point.

- Send a presentation layer message to a specified destination presentation layer peer end point address.

- Receive a presentation layer message from another presentation layer end point address. A receive operation may select a specific originating presentation layer end point address from which the message is expected, or receive a message from any presentation layer peer.

- The AAA-TSM service layer calls an application specific authorization decision function, which returns a condition code expressing an approval, denial, or partially approves with a referral to another AAA Server.

- AAA-TSM service layer tells the presentation layer to commit an earlier partially approved authorization request.

- Cancel an earlier partially approved authorization request (i.e. rollback).

- The presentation service layer notifies the AAA-TSM service layer that it has terminated an in-progress Authorized Session.

- AAA-TSM service layer notifies the presentation service layer that another presentation service layer peer has terminated an Authorized Session.

- Un-register a presentation service layer end point address.

3.6. AAA-TSM Layer End Point Name Space

The AAA-TSM service layer end point name space is the N-tuple formed by concatenating the following components:

- AAA Server's Reliable/Secure Transport layer end point address

- AAA-TSM authorization request serial number, a unique durable unsigned integer generated by the AAA Server who first receives the User's authorization request.

Some AAA applications may require that each assigned AAA-TSM transaction serial number be stored in persistent storage, and require that it be recoverable across AAA Server system re-boots. The serial number generation algorithm must be guaranteed unique even if the AAA Server does a re-boot.

3.7. Protocol Stack Examples

The layering paradigm makes it possible to use the most appropriate syntax for each application for encoding the Application Specific Information units of that application. This encoding would take place at the presentation layer. Similarly the application layer can recognize the semantics specific to each application. Figure 6 illustrates some possible AAA protocol stacks.

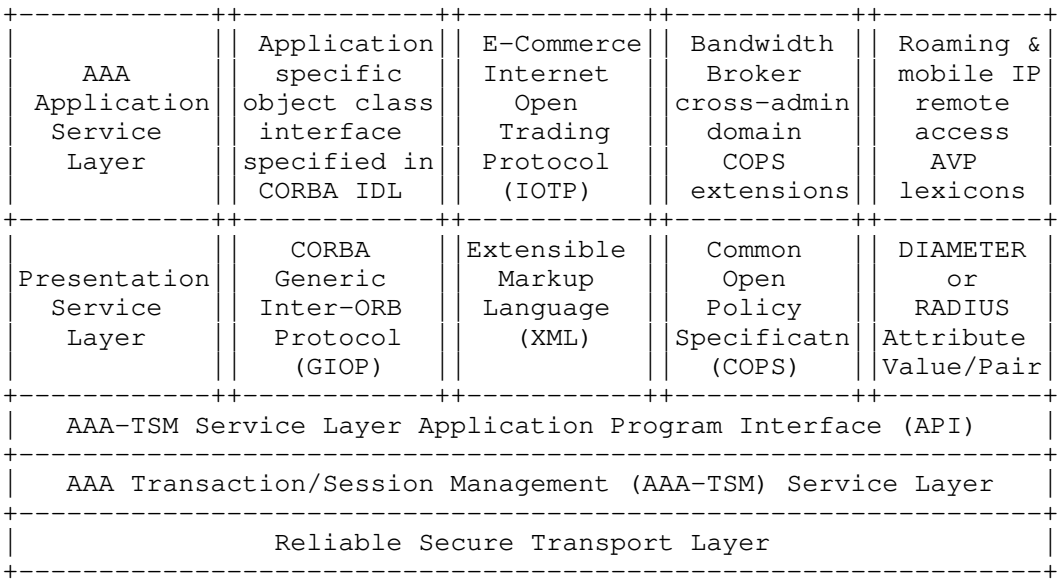


Fig. 6 -- Possible AAA Protocol Stacks

4. Security Considerations

Security considerations for the framework on which the work described in this memo is based are discussed in [2]. Security requirements for authorization are listed in section 2.2 of [3].

This memo identifies a basic set of AAA functions that are general in nature and common to many different AAA applications. We propose that a standard set of security mechanisms should be defined as part of a base AAA protocol which would include such things as public key encryption and digital signatures that could be applied to individual information units within an AAA message. Security with this granularity is needed to meet the end-to-end security requirement specified in section 2.2.7 of [3] because a single AAA message may

contain multiple information units each generated by AAA servers from different administrative domains and destined to AAA servers in different domains.

In addition, it may be necessary to encrypt or sign an entire AAA message on a hop-by-hop basis. This could be handled by a standard, lower layer protocol such as IPSEC. If so, then certain auditing requirements will have to be met so that it can be established later that the messages relative to some specific session ID were, in fact, protected in a particular way. Alternatively, hop-by-hop security mechanisms may be built into the base AAA protocol itself.

Glossary

Application Specific Information (ASI) -- information in an AAA protocol message that is specific to a particular application.

Application Specific Module (ASM) -- a software module that implements a program interface to a generic AAA server which handles application specific functionality for an AAA protocol message.

Service Provider -- an organization which provides a service.

User -- the entity seeking authorization to use a resource or a service.

User Home Organization (UHO) -- An organization with whom the User has a contractual relationship which can authenticate the User and may be able to authorize access to resources or services.

References

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

[2] Vollbrecht, J., Calhoun, P., Farrell, S., Gommans, L., Gross, G., de Bruijn, B., de Laat, D., Holdrege, M. and D. Spence, "AAA Authorization Framework", RFC 2904, August 2000.

[3] Vollbrecht, J., Calhoun, P., Farrell, S., Gommans, L., Gross, G., de Bruijn, B., de Laat, C., Holdrege, M. and D. Spence, "AAA Authorization Application Examples", RFC 2905, August 2000.

[4] Farrell, S., Vollbrecht, J., Calhoun, P., Gommans, L., Gross, G., de Bruijn, B., de Laat, C., Holdrege, M. and D. Spence, "AAA Authorization Requirements", RFC 2906, August 2000.

[5] Blaze, M., Feigenbaum, J., Ioannidis, J. and A. Keromytis, "The KeyNote Trust-Management System Version 2", RFC 2704, September 1999.

Authors' Addresses

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

George M. Gross
Lucent Technologies
184 Liberty Corner Road, m.s. LC2N-D13
Warren, NJ 07059
USA

Phone: +1 908 580 4589
Fax: +1 908-580-4991
EMail: gmgross@lucent.com

Leon Gommans
Enterasys Networks EMEA
Kerkplein 24
2841 XM Moordrecht
The Netherlands

Phone: +31 182 379279
email: gommans@cabletron.com
or at University of Utrecht:
l.h.m.gommans@phys.uu.nl

John R. Vollbrecht
Interlink Networks, Inc.
775 Technology Drive, Suite 200
Ann Arbor, MI 48108
USA

Phone: +1 734 821 1205
Fax: +1 734 821 1235
EMail: jrv@interlinknetworks.com

David W. Spence
Interlink Networks, Inc.
775 Technology Drive, Suite 200
Ann Arbor, MI 48108
USA

Phone: +1 734 821 1203
Fax: +1 734 821 1235
EMail: dspence@interlinknetworks.com

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.