Schema for the Remote Access Dialin User Service (RADIUS)

1. Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

To learn the current status of any Internet-Draft, please check the "id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ds.internic.net (US East Coast), nic.nordu.net (Europe), ftp.isi.edu (US West Coast), or munnari.oz.au (Pacific Rim).

The distribution of this memo is unlimited. It is filed as <draft-ietf-asid-ldapv3-radius-00.txt>, and expires May 1, 1998. Please send comments to the authors.

2. Abstract

This document defines a schema for the Remote Access Dialin User Service (RADIUS). This schema makes it possible to integrate a RADIUS server with an LDAP-based directory service, making it possible for an organization to maintain a single store of user information. This consolidation is desirable since it results in a reduction in the administrative workload, and eliminates the need to synchronize across multiple user information stores.

3. Introduction

Today enterprises are looking to simplify the process of user administration. With the advent of HR and personnel management systems, information on a user is typically entered at the time of hiring, is retained until termination. If an LDAP-based directory is also deployed within the enterprise, this necessitates synchronization of the HR or personnel database with the LDAP-based directory in order to maintain consistency.
Should the enterprise then seek to deploy NAS devices or layer 2 tunneling solutions, there may be a need to add a RADIUS server or if extended security is required, a backend security server. Each of these may require their own store of user information.

The possibility of operating multiple stores of user information is not appealing, since this may require rekeying of information or synchronization between multiple stores. Maintaining multiple stores also results in increased maintenance costs, and raises concerns about inconsistency and replication delays. In order to avoid this problem, it is desirable to consolidate stores of user information. One way this can be achieved is to make it possible for a RADIUS server to store its user information in an LDAP-based directory.

This document is one of three related specifications which describe how a RADIUS server may be integrated with an LDAP-based directory service. Reference [12] describes a schema designed for tracking sessions in progress. Such information can be useful for a variety of purposes including security incident response; simultaneous usage control; or monitoring of connection quality, login time, packet size or bandwidth usage. Due to the frequency of changes to this data, dynamic attributes must be employed, as described in [9].

Reference [13] describes an LDAP extension supporting validation of user credentials submitted during PPP authentication. This makes it possible for the RADIUS server to validate user credentials received in the Access-Request packet. We believe that such an extension is needed since current LDAP security mechanisms do not support PPP authentication methods. In addition, requiring a BIND and UNBIND for each authentication results in an unacceptable level of overhead. Rather than having to do a BIND and UNBIND for every authentication, as described in [13], when the RADIUS service starts up, it binds to the LDAP-based directory service using a special account, and when the RADIUS service shuts down, it unbinds. In between, during the time that the RADIUS service is running, the LDAP extension for PPP authentication is used to validate user credentials.

This document defines an LDAP schema for the Remote Access Dialin User Service (RADIUS). The RADIUS protocol, described in [6]-[8], supports authentication, authorization and accounting for dialup users. To date, RADIUS servers have stored and retrieved user data in a variety of ways, including databases and flat files. The goal of the schema described in this document is to make it as easy as possible to add support for LDAP-based directory services to existing RADIUS server implementations. In order to allow RADIUS servers to interoperate with a variety of LDAP-based directory services, this schema is designed to be implementable on a wide range of directory service implementations. This requires avoiding reliance on features that have not been widely implemented, such as multiple inheritance.
3.1. Objects and attributes

The RADIUS schema defined in this document requires support for several new classes: radiusProfileClass, radiusPolicyClass, radiusDictionaryClass, and eapDictionaryClass. The radiusProfileClass is used to store RADIUS attributes relevant to groups of users. The radiusPolicyClass is used to describe conditions under which a given profile may be applied. The radiusDictionaryClass is used to store the RADIUS Dictionary corresponding to a particular profile. This along with the vendorId attribute allows RADIUS profile objects to be self-describing. The eapDictionaryClass is used to store a list mapping EAP Types to names.

The attributes in the radiusProfileClass fall into three categories: attributes present in the Access-Request, attributes present in the Access-Reply, and attributes that do not travel over the wire. Attributes present in the Access-Reply are stored in the directory so that the RADIUS server can retrieve them and include them in the Access-Reply. Note that only static attributes present in Access-Reply need be stored in the directory; attributes which are computed on the fly can be recreated as needed.

When stored in the directory, attributes included in the Access-Request are used to represent conditions required for an Access-Accept to be sent. For example, a callingStationId attribute can be stored in the directory in order to indicate that the Calling-Station-Id attribute should have a certain value for a given user.

Attributes that do not travel over the wire also typically represent information which is useful in determining whether to return an Access-Accept. For example, a timeOfDay attribute can be stored in the directory in order to restrict a user’s access by time of day.

The attributes in the radiusPolicyClass represent conditions which may be evaluated in order to determine whether a given profile should be applied. For example, rather than just restricting access, it may be desirable to give users different Session-Time or Port-Limit attributes depending on the time of day. Similarly, it may be desirable to require that modem callers do callback or call from a particular callingStationId, while this may not make sense for users connecting over a VPN. If this functionality is needed then policy objects will need to be created, and attributes such as timeOfDay or portType will be set.

3.2. Profiles

The RADIUS schema defined in this document is designed to support attributes which apply to individual users, as well as attributes which apply to groups of users.

In the early versions of this document, it was envisaged that all attributes would be contained within the user object. However, this approach allows a user to only have a single set of attributes. This
is problematic in the case that the enterprise were to attempt to deploy RADIUS servers from more than one vendor, or where use of different profiles is required according to the situation.

Furthermore, since in many cases RADIUS attributes are shared by large groups of users, storing attributes in every user object creates unnecessary difficulty in changing the attributes of all members of a group, and requires replication of large amounts of redundant data.

As a result, the notion of a RADIUS profile was introduced. Within this document, we allow profiles to include pointers to other profiles or to a series of policies, so that profiles may form a linked list. This allows a hierarchy of profiles to be provided. More specific attributes override more general ones. Since many RADIUS parameters are expected to be identical for a group of users, typically the user object contains either a small number of attributes (perhaps only a radiusProfilePointer or radiusPolicyPointers) while subsequent RADIUS profiles contain more attributes.

In the schema that follows we allow (but do not require) RADIUS attributes to be stored in the user object. The user object also must contain either a RADIUSProfilePointer or a RADIUSPolicyPointer. These are distinguished names pointing to RADIUS Profile Objects or RADIUS Policy Objects, respectively. Similarly, RADIUS profiles may themselves contain a RADIUSProfilePointer or a RADIUSPolicyPointer.

3.2.1. Example

All BIGCO employees are required to use token card authentication, and thus in the company profile the authenticationType attribute is set to only allow EAP, and the EAP-Type attribute is set for BIGCO’s token card type. BIGCO also sets up a marketing profile providing a session-Timeout value of 30 minutes, a portLimit of one, and framedIpAddress set to indicate dynamic allocation. However, Fred requires a static IP address, and thus his user object will contain a framedIpAddress attribute as well as a radiusProfilePointer to the Marketing profile.

3.3. Policy support

The schema described in this document provides for the unconditional application of a profile to a user via the RADIUSProfilePointer attribute. The RADIUSProfilePointer is a distinguished name pointing to a RADIUS profile. This pointer is used when a given profile is to be applied in all circumstances (unconditional).

However it may be desirable to have profile A apply to a user in one set of circumstances, and profile B apply in another set of circumstances. Where a profile needs to be applied based on a set of conditions, a user object or profile object may contain one or more RADIUS-PolicyPointer attributes. The RADIUSPolicyPointer is a distinguished name pointing to a RADIUS Policy Objet. The RADIUSPolicyPointerClass contains attributes reflecting the conditions under which a given
A RADIUS profile is to be applied, along with a RADIUSProfilePointer to the RADIUS profile itself that should be applied when the conditions hold. Since a RADIUSPolicyPointer need only be used when there are multiple conditions to evaluate, the RADIUSPolicyPointer attribute is multi-valued. A RADIUSPolicyPointer may be an attribute in the user object or in a RADIUS profile.

Although a relatively simple RADIUS Policy Object is presented in this schema, more complex versions are possible. For example, it has been suggested that regular expressions be allowed in policy attributes so as to provide more advanced matching capabilities.

3.3.1. Example

Let us assume that BIGCO wishes to offer dialin access to their domestic sales force, as well as VPN access to contractors and to finance employees travelling overseas. In order to consistently manage and account for the use of their NAS devices and Layer 2 tunnel servers (PPTP/L2F/L2TP), BIGCO has chosen to adopt the RADIUS protocol. However, given the large number of employees and contractors that need to be managed, BIGCO desires a RADIUS solution integrated with their existing LDAP-based directory service. This will allow the network administrator to edit the user’s RADIUS attributes with the same user-interface as they use to edit other user attributes, and will eliminate the need to maintain multiple stores of user information.

As part of this service offering, BIGCO may wish to implement a number of access policies. For example, in order to make sure that high speed dialin access is available to the sales force when they need it, BIGCO may wish to restrict use of the ISDN ports to sales personnel only during the hours of 9-5, and permit the use of multilink. Since contractors are only to be given access to a selected group of machines, BIGCO may wish to apply a filter to their traffic. Since travelling finance users often access highly confidential information over the VPN, BIGCO may wish to require that these users authenticate via a smartcard, and use only 128-bit encryption so as to provide for extended security. For security reasons, BIGCO may wish to restrict contractors and finance users to a single login at a time.

In certain cases, BIGCO may also wish to implement policies that depend on the type of port or network that the user is connecting to. For example, if the user is connecting via dialup, then it may be appropriate to include tunnel attributes within the Access-Accept, so as to set up a tunnel for the user. However, if the user is already connected via a tunnel, this would not be necessary. Similarly, if BIGCO only has a limited number of ISDN ports available, it may be desirable to set a shorter Session-Timeout or Idle-Timeout on these ports, or to set Port-Limit to one so as to not allow multi-link. The schema defined in this document permits the administration of these and other policies.
3.4. Caching

Reference [14] describes a simple caching scheme for LDAP-based directories. A new operational attribute, ttl, is defined which specifies the maximum time an object may remain in the cache. Such a caching scheme is particularly beneficial for the schema presented in this document, since it is expected that profiles and policies will apply to large numbers of users. The first time the RADIUS server encounters a pointer to a given profile or policy, the profile or policy will be retrieved from the directory and cached. Subsequently, the profile or policy may be retrieved from the cache, speeding the retrieval process. As a result, it is to be expected that caching should result in a substantial performance gain.

Since the schema presented in this document is designed to be used across several directory implementations and the operational ttl attribute is not yet widely implemented, a ttl attribute is included in the radiusProfileClass and radiusPolicyClass. As in [14], the ttl attribute denotes the number of seconds that an entry may remain in the cache before becoming stale. A value of 0 implies that the object must not be cached.

3.5. Extensibility

Today vendors distinguish their RADIUS servers by a variety of means, including the range of supported attributes (standard and vendor-specific), and the breadth of policies that may be represented. As a result, while it is desirable to provide a common base set of classes and attributes which all RADIUS schemas will share, RADIUS server capabilities differ substantially from implementation to implementation, and a successful RADIUS schema definition must support this differentiation.

The schema described in this document provides support for most of the attributes defined in [6]-[8], but does not include vendor-specific attributes or attempt to cover the full range of policies that might be desirable. Thus we expect that vendor extensions will be required in most cases.

Vendor differentiation can be achieved via two methods: adding attributes to the base RADIUS profile and policy classes, or creating subclasses inheriting from the base classes. Adding attributes to the base class is recommended in cases where the new attributes to be added do not conflict with those described in this document or in [6]-[8].

Where conflicts do not arise, the vendorId attribute within the RADIUS profile should be set to zero, indicating that the profile supports IETF standard RADIUS. As with many RADIUS server implementations, a RADIUS dictionary capability is supported through the RADIUS Dictionary Class, allowing the RADIUS profile to be self-describing. The goal is to allow attributes to be added without having to require an update to the RADIUS server code. Note however that the RADIUS
dictionary is only designed to describe attributes that are sent on the wire; non-wire attributes (such as authenticationType) are not included, and thus such attributes typically cannot be supported without code changes.

Creating a sub-class is desirable in cases where conflicts are possible. Such conflicts can arise for example, when vendors have defined attributes which conflict with the standard RADIUS attribute space described in [6]-[8]. In this case, the vendorId attribute should be set to the SMI Vendor Code, indicating that the profile is specific to a given vendor, and contains potentially conflicting elements. Since a RADIUS server searching for a profile with objectclass=radiusProfile-Class will encounter both base class profiles and subclasses, the vendorId attribute is critical in allowing an implementation to differentiate the profiles it can understand from those that it cannot. Typically an implementation will only wish to work with profiles whose vendorId is either zero (IETF RADIUS) or set to their own SMI Vendor Code. As with addition of attributes to the base class, when attributes are added to a subclass, the RADIUS Dictionary class should also be modified to allow the subclass to be self-describing.

Since it is conceivable that RADIUS servers from two vendors may be deployed simultaneously, both desiring to store objects in the same LDAP-based directory service, and each implementing their own profile subclass, a method must be provided to allow a user to have more than one set of RADIUS profile and policy objects. This can be achieved by allowing the radiusProfilePointer or radiusPolicyPointer to point to a container object rather than pointing to an object itself. The RADIUS server would then search the container for a RADIUS profile or policy with an appropriate vendorId.

In order to prevent name conflicts, it is recommended that vendors adding their own attributes prepend a suffix to all attribute names. The IETF Schema Working Group has announced its intention to manage suffix allocation in order to avoid name conflicts. Such precautions are particularly necessary when dealing with attributes which may appear in the Access-Request. Note that such attributes are included in the schema in order to express conditions under which an Access-Accept may be sent, not in order to provide guidance about what the RADIUS server should return in the Access-Accept. Since vendors may wish to provide sophisticated facilities for expressing such conditions, it is likely that there will be disagreement on how these conditions should be formulated. However, rather than redefining existing attributes, vendor should create their own attributes using suffixes for conflict avoidance.

To illustrate how extensibility features may be used, the additional attributes supported by a hypothetical BIGCO Profile Class are included.
4. Object definitions

The RADIUS schema includes definition of the following objects:

- RADIUS Profile Class
- RADIUS Policy Class
- RADIUS Dictionary Class
- EAP Dictionary Class

4.1. RADIUS Profile Class

\[
\text{( radiusProfileClass } 1
\text{ NAME 'radiusProfile'
}\text{ SUP profile}
\text{ PARENT (country $ organization $ organizationalUnit $}
\text{ locality $ container)}
\text{ STRUCTURAL}
\text{ MUST (}
\text{ cn }
\text{ MAY ( serviceType $ framedProtocol $ framedIPAddress $}
\text{ framedIPNetmask $ framedRouting $}
\text{ filterId $ framedMTU $ framedCompression $}
\text{ loginIPHost $ loginService $ loginTCPPort $}
\text{ callbackNumber $ callbackId $ framedRoute $}
\text{ framedIPXNetwork $ sessionTimeout $ idleTimeout $}
\text{ terminationAction $ calledStationId $ callingStationId $}
\text{ loginLATService $ loginLATNode $ loginLATGroup $}
\text{ framedAppleTalkLink $ framedAppleTalkNetwork $}
\text{ framedAppleTalkZone $ portLimit $ loginLATPort $}
\text{ tunnelType $ tunnelMediumType $ tunnelServerEndpoint $}
\text{ tunnelPrivGroup $ arapFeatures $ arapZoneAccess $}
\text{ arapSecurity $ passwordRetry $ prompt $ ttl $}
\text{ eapType $ sessionsAllowed $ authenticationType $}
\text{ vendorId $ timeOfDay $ radiusDictionary $}
\text{ radiusProfilePointer $ radiusPolicyPointer)
\text{ )}
\]

4.2. RADIUS Policy Class

\[
\text{( radiusPolicyClass } 1
\text{ NAME 'radiusPolicy'
}\text{ SUP policy}
\text{ PARENT (country $ organization $ organizationalUnit $}
\text{ locality $ container)}
\text{ STRUCTURAL}
\text{ MUST (}
\text{ cn $ radiusProfilePointer}
\text{ )}
\text{ MAY ( portType $ nasPrefixes $ clientPrefixes $}
\text{ timeOfDay $ ttl $ vendorId}
\text{ )}
\]
4.3. RADIUS Dictionary Class

(radiusDictionaryClass 1
 NAME 'radiusDictionaryClass'
 SUP top
 PARENT (country $ organization $ organizationalUnit $ locality $ container)
 STRUCTURAL
 MUST {
     cn $ dictionaryEntry
 }
)

4.4. EAP Dictionary Class

(eapDictionaryClass 1
 NAME 'eapDictionaryClass'
 SUP top
 PARENT (country $ organization $ organizationalUnit $ locality $ container)
 STRUCTURAL
 MUST {
     cn $ dictionaryEntry
 }
)

4.5. BIGCO Profile Class

As described earlier, the base classes may be extended by attribute addition, subclassing, or both. An example of the subclassing approach is illustrated below. Here the bigcoProfileClass is created as a sub-class of the radiusProfileClass and adds several attributes, each of which uses bigco as a suffix to avoid name collisions.

(bigcoProfileClass 1
 NAME 'bigcoProfile'
 SUP radiusProfileClass
 PARENT (country $ organization $ organizationalUnit $ locality $ container)
 STRUCTURAL
 MUST {
 }
 MAY { bigcoAllowedPortType $ bigcoEncryptionType $ bigcoBapRequired $ bigcoBapLinednLimit $ bigcoBapLinednTime $ bigcoDynDirServer
 }
)
5. Attribute definitions

5.1. New Attribute Types Used in the RADIUS Profile Class

(radius radiusProfileClass 6
 NAME 'serviceType'
 DESC 'The service to be provided to the user. Values include: Login(1), Framed(2), Callback Login(3), Callback Framed(4), Outbound(5), Administrative(6), NAS Prompt(7), Authenticate Only(8), Callback NAS Prompt(9)'
 EQUALITY integerMatch
 SYNTAX 'INTEGER'
 SINGLE-VALUE
)

(radius radiusProfileClass 7
 NAME 'framedProtocol'
 DESC 'For Framed service, the protocol to be provided to the user. Values include PPP(1), SLIP(2), ARAP(3), Gandalf(4), Xylogics(5)'
 EQUALITY integerMatch
 SYNTAX 'INTEGER'
 SINGLE-VALUE
)

(radius radiusProfileClass 8
 NAME 'framedIPAddress'
 DESC 'IP address to be assigned to the user in dotted decimal notation'
 EQUALITY caseIgnoreIA5Match
 SYNTAX 'INTEGER'
 SINGLE-VALUE
)

(radius radiusProfileClass 9
 NAME 'framedIPNetmask'
 DESC 'Netmask to apply to the user in dotted decimal notation'
 EQUALITY caseIgnoreIA5Match
 SYNTAX 'IA5String(128)'
 SINGLE-VALUE
)

(radius radiusProfileClass 10
 NAME 'framedRouting'
 DESC 'Routing method for the user. Values include None(1), Send(2), Listen(3), Send & Listen(4)'
 EQUALITY integerMatch
 SYNTAX 'INTEGER'
)
SINGLE-VALUE

(r radius radiusProfileClass 11
 NAME 'filterId'
 DESC 'String representing the filter list for the user'
 EQUALITY caseIgnoreIA5Match
 SYNTAX 'IA5String{128}''
)

(r radius radiusProfileClass 12
 NAME 'framedMTU'
 DESC 'Maximum Transmission Unit for the user'
 EQUALITY integerMatch
 SYNTAX 'INTEGER'
 SINGLE-VALUE
)

(r radius radiusProfileClass 13
 NAME 'framedCompression'
 DESC 'Compression protocol to be used on the link. Values include: None(1), VJ compression(2), IPX header compression(3)'
 EQUALITY integerMatch
 SYNTAX 'INTEGER'
)

(r radius radiusProfileClass 14
 NAME 'loginIPHost'
 DESC 'System with which to connect the user in dotted decimal notation'
 EQUALITY caseIgnoreIA5Match
 SYNTAX 'IA5String{128}''
)

(r radius radiusProfileClass 15
 NAME 'loginService'
 DESC 'Service to be used to connect the user to the login host. Values include Telnet(1), Rlogin(2), TCP Clear(3), PortMaster(4), and LAT(5)'
 EQUALITY integerMatch
 SYNTAX 'INTEGER'
 SINGLE-VALUE
)

(r radius radiusProfileClass 16
 NAME 'loginTCPPort'
 DESC 'The TCP port with which the user is to be connected'
 EQUALITY integerMatch
 SYNTAX 'INTEGER'
 SINGLE-VALUE
)
( radius radiusProfileClass 19
  NAME 'callbackNumber'
  DESC 'Number to be called'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String(128)'
  SINGLE-VALUE
)

( radius radiusProfileClass 20
  NAME 'callbackId'
  DESC 'Name of place to be called'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String(128)'
  SINGLE-VALUE
)

( radius radiusProfileClass 22
  NAME 'framedRoute'
  DESC 'Routes to be plumbed for the user'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String(128)'
)

( radius radiusProfileClass 23
  NAME 'framedIPXNetwork'
  DESC 'IPX Network number to be configured for the user'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String(128)'
  SINGLE-VALUE
)

( radius radiusProfileClass 27
  NAME 'sessionTimeout'
  DESC 'Per-session time limit in seconds. After this expires, the action specified in Termination-Action is taken'
  EQUALITY integerMatch
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)

( radius radiusProfileClass 28
  NAME 'idleTimeout'
  DESC 'The maximum number of consecutive seconds of idle connection allowed before session termination'
  EQUALITY integerMatch
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)

( radius radiusProfileClass 29
  NAME 'terminationAction'
)
DESC 'Action taken when specified service is completed. Values include Default(1) or RADIUS-Request(2)'
EQUALITY integerMatch
SYNTAX 'INTEGER'
SINGLE-VALUE
)

(radius radiusProfileClass 30
 NAME 'calledStationId'
 DESC 'Indicates that access is only allowed to these numbers'
 EQUALITY caseIgnoreIA5Match
 SYNTAX 'IA5String{128}''
)

(radius radiusProfileClass 31
 NAME 'callingStationId'
 DESC 'Indicates that access is restricted and only allowed from this phone number.'
 EQUALITY caseIgnoreIA5Match
 SYNTAX 'IA5String{128}''
)

(radius radiusProfileClass 34
 NAME 'loginLATService'
 DESC 'Identity of the LAT service to use'
 EQUALITY caseIgnoreIA5Match
 SYNTAX 'IA5String{128}''
 SINGLE-VALUE
)

(radius radiusProfileClass 35
 NAME 'loginLATNode'
 DESC 'The node with which the user is to be automatically connected by LAT'
 EQUALITY caseIgnoreIA5Match
 SYNTAX 'IA5String{128}''
 SINGLE-VALUE
)

(radius radiusProfileClass 36
 NAME 'loginLATGroup'
 DESC 'The LAT group codes which this user is authorized to use'
 EQUALITY caseIgnoreIA5Match
 SYNTAX 'IA5String{128}''
 SINGLE-VALUE
)

(radius radiusProfileClass 37
 NAME 'framedAppleTalkLink'
 DESC 'The AppleTalk network number which should be used for the user'
 EQUALITY caseIgnoreIA5Match
)
SYNTAX 'INTEGER'
SINGLE-VALUE
)

(radius radiusProfileClass 38
  NAME 'framedAppleTalkNetwork'
  DESC 'The AppleTalk network number which
      the NAS should probe to allocate an
      AppleTalk node for the user'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)

(radius radiusProfileClass 39
  NAME 'framedAppleTalkZone'
  DESC 'The name of the Default AppleTalk Zone'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String(128)'
  SINGLE-VALUE
)

(radius radiusProfileClass 62
  NAME 'portLimit'
  DESC 'Maximum number of ports to be provided'
  EQUALITY integerMatch
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)

(radius radiusProfileClass 39
  NAME 'loginLATPort'
  DESC 'The Port with which the user is to
       connected by LAT'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String(128)'
  SINGLE-VALUE
)

(radius radiusProfileClass 64
  NAME 'tunnelType'
  DESC 'String representing the type of tunnel to
      be set up, of the form Tag: Value. Values
      include PPTP(1), L2F(2), L2TP(3), ATM-P(4),
      VTP(5), AH(6), IP-IP(7).'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String(128)'
)

(radius radiusProfileClass 65
  NAME 'tunnelMediumType'
  DESC 'String representing the medium for the tunnel to
       run over, of the form Tag: Value. Values
       include IP(1), X.25(2), ATM(3), Frame Relay(4).'
EQUALITY caseIgnoreIA5Match
SYNTAX 'IA5String{128}'
}

(radius radiusProfileClass 67
 NAME 'tunnelServerEndpoint'
 DESC 'String representing the address of the tunnel
 server, of the form Tag: Value. The format
 of the value field depends on the tunnelMediumType
 attribute'
 EQUALITY caseIgnoreIA5Match
 SYNTAX 'IA5String{128}'
)

(radius radiusProfileClass ?
 NAME 'tunnelPrivateGroupId'
 DESC 'String representing the Private Group Id for the
 tunnel, of the form Tag: Value.'
 EQUALITY caseIgnoreIA5Match
 SYNTAX 'IA5String{128}'
)

(radius radiusProfileClass 71
 NAME 'arapFeatures'
 DESC 'This is a compound string containing info that
 the NAS should send to the user in the ARAP
 feature flags packet'
 EQUALITY caseIgnoreIA5Match
 SYNTAX 'IA5String{128}'
 SINGLE-VALUE
)

(radius radiusProfileClass 72
 NAME 'arapZoneAccess'
 DESC 'This field controls access to ARAP zones.
 Values include
 Only allow access to default zone(1),
 Use zone filter inclusively(2),
 Use zone filter exclusively (4)'
 EQUALITY integerMatch
 SYNTAX 'INTEGER'
 SINGLE-VALUE
)

(radius radiusProfileClass 73
 NAME 'arapSecurity'
 DESC 'This field contains an integer
 specifying the security module signature,
 which is a Macintosh OSTYPE'
 EQUALITY integerMatch
 SYNTAX 'INTEGER'
 SINGLE-VALUE
)
( radius radiusProfileClass 75
  NAME 'passwordRetry'
  DESC 'This is an integer specifying the number of password retry attempts to permit the user'
  EQUALITY integerMatch
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)

( radius radiusProfileClass 76
  NAME 'prompt'
  DESC 'This attribute is used only in RADIUS Access-Challenge packets and indicates if the NAS should echo the user’s response as entered. Values include No Echo (0), or Echo(1).'
  EQUALITY integerMatch
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)

( radius radiusProfileClass 256
  NAME 'ttl'
  DESC 'Time in seconds that this profile may remain in a profile cache.'
  EQUALITY integerMatch
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)

( radius radiusProfileClass 257
  NAME 'eapType'
  DESC 'Allowable EAP types, in order of preference. If this attribute has a value, EAP must be included in the allowable authentication types.'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String{128}'
  SINGLE-VALUE
)

( radius radiusProfileClass 258
  NAME 'sessionsAllowed'
  DESC 'This attribute indicates the number of simultaneous sessions allowed for this user.'
  EQUALITY integerMatch
  SYNTAX 'INTEGER'
  SINGLE-VALUE
)

( radius radiusProfileClass 259
  NAME 'authenticationType'
  DESC 'Allowable authentication types (EAP, CHAP, PAP, MS-CHAP, etc.) in order of preference. If an attribute isn’t included, it isn’t allowed.'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String{128}'
)
5.2. New Attribute Types Used in the RADIUS Policy Class

(radius radiusPolicyClass 1
 NAME 'portType'
 DESC 'Bitstring representing port types for which access is allowed. Values include Async(1), Sync(2), ISDN Sync(3), V.120(4), V.110(5), Virtual(6) and ATM(7). If not present, any type is allowed.'
)
EQUALITY integerMatch
SYNTAX 'INTEGER'
SINGLE-VALUE

(radius radiusPolicyClass 2
 NAME 'nasPrefixes'
 DESC 'String representing originating nasIPAddress
 prefixes to which this profile will apply.
 If not present, this profile applies to all
 prefixes.'
 EQUALITY caseIgnoreIA5Match
 SYNTAX 'IA5String{128}'
 SINGLE-VALUE
)

(radius radiusPolicyClass 3
 NAME 'clientPrefixes'
 DESC 'String representing originating RADIUS Client
 prefixes to which this profile will apply.
 If not present, this profile applies to all
 prefixes.'
 EQUALITY caseIgnoreIA5Match
 SYNTAX 'IA5String{128}'
 SINGLE-VALUE
)

5.3. New Attribute Types Used in the RADIUS Dictionary Class

(radius radiusDictionaryClass 1
 NAME 'dictionaryEntry'
 DESC 'A dictionary entry in the RADIUS dictionary,
 of the form Attribute-Number:[Vendor-Type:]ldapDisplayName:Type.
 Vendor-Type may only be present with Attribute-Number=26
 (Vendor Specific).'</n
 EQUALITY caseIgnoreIA5Match
 SYNTAX 'IA5String{128}'
)

5.4. New Attribute Types Used in the BIGCO Profile Class

(bigco bigcoProfileClass 262
 NAME 'bigcoAllowedPortType'
 DESC 'This attribute is a bitstring representing
 port types to which access is allowed. Values
 include Async(1), Sync(2), ISDN Sync(3),
 V.120(4), V.110(5), and Virtual(6).'</n
 EQUALITY bitStringMatch
 SYNTAX 'BitString'
 SINGLE-VALUE
)
6. Security issues

Integration of a RADIUS server with an LDAP-based directory service can result in a variety of security threats, including:

- Rogue LDAP-servers
- Theft of passwords
Inappropriate use

These threats are discussed in turn.

6.1. Rogue LDAP servers

Were a rogue LDAP server to respond to queries from the RADIUS server and have its responses accepted, it is possible that users could gain inappropriate access to the network. In order to protect against this, the conversation between the RADIUS server and the LDAP-based directory service SHOULD be mutually authenticated via SSL/TLS or IPSEC.

6.2. Theft of passwords

RADIUS servers supporting PAP authentication SHOULD provide for confidentiality of packets sent to and from the LDAP server. This can be achieved using SSL/TLS or IPSEC ESP.

6.3. Inappropriate use

This schema is intended for use by a RADIUS server integrating with an LDAP-enabled directory. This schema SHOULD NOT be used by devices looking to access the directory directly.

LDAP-enabling of devices would introduce several security problems. As described in [13], LDAP-enabling a RADIUS server requires that the RADIUS server be given permissions to access a user’s RADIUS objects and attributes. If the dynamic attributes described in [12] are supported, then the RADIUS service must also be able to write those attributes to the DS. As a result, the administrator of the RADIUS server should exercise care to ensure that the RADIUS account password is not compromised. If at all possible, the RADIUS server should be physically secured.

In contrast, LDAP-enabling of devices requires that devices be given these access-rights. This can be achieved by making the devices members of a group, and giving the group access rights to this portion of the schema. However, such an expansion of access rights is undesirable, since while RADIUS servers can often be physically secured, widely deployed devices typically cannot be.

Furthermore, direct use of LDAP across a WAN typically requires that LDAP pass through a firewall. This is problematic since LDAP-based directories can be used to store a wide variety of data, much of it sensitive. Thus without implementing an LDAP proxy to limit access only to appropriate portions of the schema, it is difficult to enforce security. Since humans are notoriously lax in administration of access rights, an attacker obtaining a device password would typically also obtain access not only to RADIUS attributes for every user, but to other information as well.
Beyond the security issues, LDAP-enabling of devices increases the size of the device binaries, and may in some cases introduce dependencies in the device boot sequence that can be problematic.

7. Acknowledgments

Thanks to Steven Judd, Ashwin Palekar, David Eitelbach, Narendra Gidwani and Donald Rule of Microsoft for useful discussions of this problem space.

8. References


9. Authors’ Addresses

Bernard Aboba
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Phone: 425-936-6605
EMail: bernarda@microsoft.com