

## RADIUS

### Securing Public Access to Private Resources

By Jonathan Hassell  
October 2002  
ISBN 0-596-00322-6,  
206 pages

## Excerpts from Chapter 9, New RADIUS Developments

Up to this point, I've covered the contents and specifications of the original RADIUS RFC drafts. Since those drafts were approved and published, new advancements in technology have mandated some revisions to those RFCs, particularly in the areas of tunnel support and new security technologies. In this chapter, I'll cover these updates and how they might affect your current implementation or any changes you will make in the future.

## The Apple Remote Access Protocol

The Apple Remote Access Protocol (ARAP) sends traffic based on the AppleTalk protocol across PPP links and ISDN switched-circuit networks. ARAP is still pervasive in the Apple market, although the company is attempting to transition into an Apple-specific TCP stack for use over a PPP link. ARAP support is typically found in most RADIUS client gear, and RADIUS now supports authenticating based on the ARAP protocol.

ARAP authentication typically takes one to two steps, as follows:

1. The first step is basically a mutual authentication with an exchange of random numbers signed with a *key*, which happens to be the user's password. The RADIUS client challenges and authenticates the dial-in client, and the dial-in client challenges and authenticates the RADIUS client challenges. First, the RADIUS client sends random numbers of 32 bits to the dial-in client inside an ARAP `msg_auth_challenge` packet. Then, the dial-in client uses his password to encrypt the two random numbers sent by the RADIUS client with DES. The dial-in client sends the result back in a `msg_auth_request` packet. Finally, the RADIUS client unencrypts the message based on the password it has on record for the user and verifies the random numbers are

intact. If so, it encrypts the challenge from the dial-in client and sends it back in a `msg_auth_response` packet.

2. The RADIUS client may initiate a second phase of authentication using optional *add-in security modules*, which are small pieces of code that are run on both ends of the connection and provide read and write access across the link. Some security token vendors use these add-ins to perform their own proprietary authentication.

There are some caveats to integrating ARAP and RADIUS based on the way ARAP is designed. Namely, ARAP transmits more security profile information after the first phase completes but before the second phase of authentication begins. The profile information is contained within a single attribute and is a series of numeric characters relating to passwords. Even so, challenge responses and this new profile information must exist at times that may seem a bit non-standard. But it *is* the standard.

To allow an ARAP-based client access to the resources the RADIUS server is protecting, an `Access-Request` packet must be issued on behalf of the ARAP client. This process takes place as one would imagine: the RADIUS client with the ARAP protocol generates a challenge based on a random number and, in response from the end-user client, receives the challenge and the username. The relevant data is then forwarded to the RADIUS server inside a standard RADIUS `Access-Request` packet. The data that is transplanted is as follows: `User-Name`, `Framed-Protocol` with a value of 3 for ARAP, `ARAP-Password`, and any other pertinent information like `Service-Type`, `NAS-IP-Address`, `NAS-Id`, `NAS-Port-Type`, `NAS-Port`, `NAS-Port-Id`, `Connect-Info`, and others. Note that only one of the `User-Password`, `CHAP-Password`, or `ARAP-Password` attributes needs to be present in the `Access-Request` packet. Any `EAP-Messages` attributes supercede any of those attributes' presence in a packet.

The authentication then takes place. If the RADIUS server doesn't support ARAP, it should return an `Access-Reject` message. If it does, then in order to authenticate the user it should verify the user response using the challenge, found in the first eight octets of the request authenticator, and the associated response, found in the first eight octets of the `ARAP-Password` attribute. The resulting `Access-Accept` message, if this information is verified and the user is indeed successfully authenticated, should be formatted in the following manner:

- The `ID` and `Response Authenticator` fields should be included as per the normal RADIUS standard.
- The `Service-Type` attribute should be `Framed-Protocol`.
- The `Framed-Protocol` attribute should be set to a value of 3, signifying ARAP.
- The `Session-Timeout` attribute should be set to the maximum connect time. It can also be set for unlimited time by either setting the value to -1 or not including the attribute in the packet. All machines party to the transaction will consider the absence of attribute to mean the user is allowed unlimited connect time.
- The `ARAP-Challenge-Response` attribute should be set to a value of eight octets of the response to the challenge. The RADIUS server forms this by finding the challenge from the last eight octets of the `ARAP-Password` attributes and performing DES encryption using the password of the user as the key.

- The `ARAP-Features` attribute contains information that the RADIUS client should encapsulate in a feature flags packet in the ARAP protocol.
- A single `Reply-Message` of up to 253 text characters can be included.
- `Framed-AppleTalk-Network` may be included.
- `Framed-AppleTalk-Zone`, of up to 32 characters, may also be included.

Zones are an interesting concept that deserve some commentary. The ARAP protocol maps this concept of zones, which designate access privileges a user may have that correspond with an area or a set of resources. A Zone Access Flag is defined along with a list of configured zones; this access flag specifies how the user is allowed to access the zones in the list. For example, the user might be able to use only the default zone's resources, or he may be allowed to use only the zones in an accompanying list, or he may be allowed to see all zones *except* those in his list. The RADIUS client gear handles zones by using configured filters with the same names as the ARAP zones. It uses the `ARAP-Zone-Access` attribute, which contains an integer similar to the zone access flag. The name of the zone is then transplanted by the RADIUS client into pertinent RADIUS packets using the standard RADIUS `Filter-ID` attribute.

## The Extensible Authentication Protocol

EAP is supported in the new RADIUS extensions and allows for new authentication types to be used over links running on the PPP protocol. Authentication schemes such as public key, smart cards, one-time passwords, Kerberos, and others are supported over PPP when EAP is used. To support EAP, RADIUS includes two new attributes--`EAP-Message` and `Message-Authenticator`--that are described in this section.

Typically, the RADIUS server acts as an intermediary between the client and a backroom proprietary security and authentication server. It normally encapsulates the EAP packets within a standard RADIUS packet, using the `EAP-Message` attribute, and then transmits them back and forth between the two machines. This lets the RADIUS server talk to the other proprietary authentication server using a standard protocol that requires no special modifications on the RADIUS server. It can still fully support standard RADIUS requests with reduced overhead.

A typical EAP over RADIUS transaction occurs in a standard format, which is outlined here:

1. The dial-up client and the RADIUS client gear negotiate the use of EAP within their specific link control protocol--this is most commonly PPP.
2. The RADIUS client then sends an `EAP-Request/Identity` message to the client unless its identity has been verified through some other means, such as callback or caller ID.
3. The dial-up client then responds with an `EAP-Response/Identity` message.

4. The RADIUS client gear receives this response from the client and forwards it to the RADIUS server inside a standard RADIUS `Access-Request` using the `EAP-Message` attribute.
5. The RADIUS server responds with a standard `Access-Challenge` packet that contains an `EAP-Message` attribute. The `EAP-Message` attribute contains a full EAP packet.
6. The RADIUS client gear unwraps the encapsulated EAP message and forwards it to the dial-up client.

The authentication continues as many times as needed until either an `Access-Reject` message or an `Access-Accept` message is received. If the EAP transaction follows these typical steps, then the RADIUS client gear will never have to manipulate an EAP packet. Of course, the world is not always as simple as that; as such, there are a couple of caveats to this scenario.

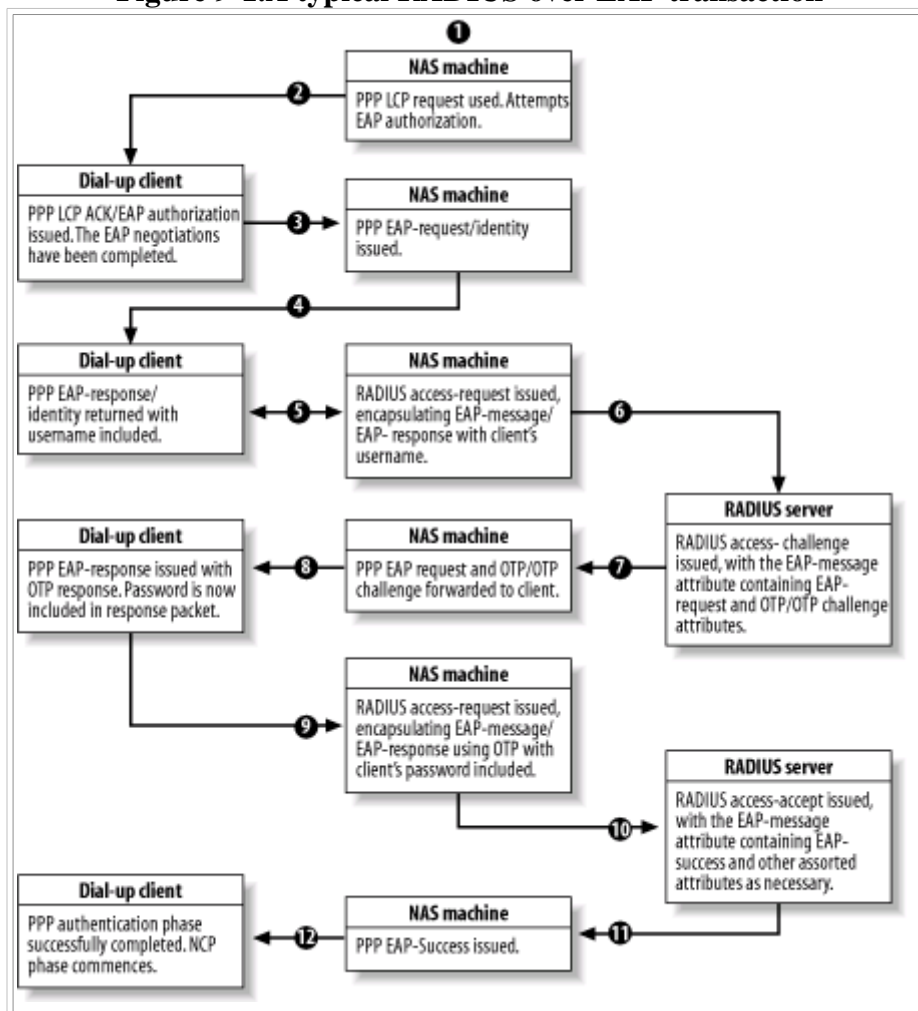
First, you must permit proxy capability between RADIUS machines that may not be compliant with the EAP protocol. If the RADIUS client machine sends the `EAP-Request/Identity`, as described previously in step two, the RADIUS client has to copy the relevant information into the appropriate fields in a standard RADIUS packet. For example, the contents of the `EAP-Response/Identity` packet need to be copied into the `User-Name` attribute. As well, the `NAS-Port` or `NAS-Port-ID` attributes should be included in the `Access-Request` packet, and the `NAS-Identifier` and `NAS-IP-Address` attributes *must* be included. (All of this must, of course, be present in the subsequent packets exchanged between the proxy machines and the original, EAP-compliant RADIUS server.) All of this is to facilitate accounting.

Second, the RADIUS client may not always issue the `EAP-Request/Identity` packet. Primarily, this occurs when the identity of a user has been verified through some other means, such as a callback and caller-ID value. You can tell this has happened when the `Called-Station-ID` or `Calling-Station-ID` attributes are present in a RADIUS packet (these are required to be present when identity is verified through their use). In this case, the RADIUS client sends a standard `Access-Request` packet to the RADIUS server with an `EAP-Message`; inside that message is the `EAP-Start` instruction, which is indicated by sending an `EAP-Message` attribute with a length of two octets. This method, although convenient, is not within the RADIUS specification as per RFC 2865, and there are problems with this approach when proxies are in use.

## Examples of an EAP Conversation

[Figure 9-1](#) is an example of a transaction between a dial-up client, a RADIUS client box, and an EAP-compliant RADIUS server using the OTP authentication scheme. (This is the same example used in the RADIUS Extensions RFC, number 2869. It has been remodeled for clarity and potentially better visual comprehension.)

Figure 9-1.A typical RADIUS over EAP transaction



## Potential Uses

EAP has a lot of potential, although its current uses are limited since most of the protocols it uses to talk with a backend security authenticator are proprietary, largely due to a lack of standardization. However, RADIUS over EAP compensates for some of the security deficiencies of the core RADIUS protocol's design, as discussed at length in Chapter 8, and is recommended for use where appropriate.

## Tunneling Protocols

With the advent of work-from-home strategies and the branch-office concept becoming ever more popular, the dependence on access to corporate networks and privatized ISPs has become stronger. There exists a way to use a sort of tunnel to log in to corporate network over the Internet and access that network's resources as though you were locally attached to it. Although discussing tunnels is beyond the scope of this book, RADIUS does support a variety of tunneling protocols, both voluntary and compulsory. New RADIUS attributes were introduced with RFC 2868 that provide support for this emerging technology.

As well, private ISPs and even some corporate IT data centers want to be able to account for the use of their service for accounting, billing, and auditing purposes. RADIUS accounting, of course supporting the AAA model as discussed in Chapter 1, is an obvious way to collect this data, especially with the new tunneling-support attributes, some modifications to the `Acct-Status-Type` attribute, and some entirely new attributes specifically focused at RADIUS accounting.

The new values for the `Acct-Status-Type` attribute are listed in [Table 9-1](#).

**Table 9-1: New values per RFC 2867 for Acct-Status-Type**

Value	Name	Description	Also requires
9	Tunnel-Start	Marks the creation of a tunnel with another end point.	User-Name, NAS-IP-Address, Acct-Delay-Time, Event-Timestamp, Tunnel-Type, Tunnel-Medium-Type, Tunnel-Client-Endpoint, Tunnel-Server-Endpoint, Acct-Tunnel-Connection
10	Tunnel-Stop	Marks the destruction of a tunnel with another node.	User-Name, NAS-IP-Address, Acct-Delay-Time, Acct-Input-Octets, Acct-Output-Octets, Acct-Session-ID, Acct-Session-Time, Acct-Input-Packets, Acct-Output-Packets, Acct-Terminate-Cause, Acct-Multi-Session-Id, Event-Timestamp, Tunnel-Type, Tunnel-Medium-Type, Tunnel-Client-Endpoint, Tunnel-Server-Endpoint, Acct-Tunnel-Connection, Acct-Tunnel-Packets-Lost
11	Tunnel-Reject	Marks the rejection of an attempt to establish a tunnel with another node.	User-Name, NAS-IP-Address, Acct-Delay-Time, Acct-Terminate-Cause, Event-Timestamp, Tunnel-Type, Tunnel-Medium-Type, Tunnel-Client-Endpoint, Tunnel-Server-Endpoint, Acct-Tunnel-Connection
12	Tunnel-Link-Start	Marks the creation of a tunnel link; for those protocols that support multiple links per tunnel.	User-Name, NAS-IP-Address, NAS-Port, Acct-Delay-Time, Event-Timestamp, Tunnel-Type, Tunnel-Medium-Type, Tunnel-Client-Endpoint, Tunnel-Server-Endpoint, Acct-Tunnel-Connection
13	Tunnel-Link-Stop	Marks the destruction of a tunnel link; for those protocols that support multiple links per tunnel.	User-Name, NAS-IP-Address, NAS-Port, Acct-Delay-Time, Acct-Input-Octets, Acct-Output-Octets, Acct-Session-Id, Acct-Session-Time, Acct-Input-Packets, Acct-Output-Packets, Acct-Terminate-Cause, Acct-Multi-Session-Id, Event-Timestamp, NAS-Port-Type, Tunnel-Type, Tunnel-Medium-Type, Tunnel-Client-Endpoint, Tunnel-Server-Endpoint, Acct-Tunnel-Connection, Acct-Tunnel-Packets-Lost
14	Tunnel-Link-Reject	Marks the rejection of an attempt to establish a tunnel link; for those protocols that support multiple links per tunnel.	User-Name, NAS-IP-Address, Acct-Delay-Time, Acct-Terminate-Cause, Event-Timestamp, Tunnel-Type, Tunnel-Medium-Type, Tunnel-Client-Endpoint, Tunnel-Server-Endpoint, Acct-Tunnel-Connection

The new tunnel-accounting attributes are integrated with the rest of the RADIUS extensions attributes in the next section.

Back to: [Sample Chapter Index](#)

Back to: [RADIUS](#)

[O'Reilly Home](#) | [O'Reilly-Partnerbuchhandlungen](#) | [Bestellinformationen](#) | [Kontakt](#)  
[Datenschutz](#) | [Über O'Reilly](#)

© 2001, O'Reilly & Associates, Inc.  
[webmaster@oreilly.de](mailto:webmaster@oreilly.de)