RadSec
a secure, reliable RADIUS Protocol

This white paper discusses RadSec, a secure, reliable protocol for proxying RADIUS requests.

1.0 Introduction

This document provides an overview and technical description of the RadSec protocol, which provides secure, reliable RADIUS transport for Radiator and other compliant products.

RadSec allows RADIUS authentication and accounting data to be passed safely across insecure networks such as the internet.

2.0 Rationale and Overview

The RadSec protocol is designed to provide secure, reliable transport of RADIUS requests across insecure networks such as the Internet. It was designed by Open System Consultants in response to some of the shortcomings of the conventional RADIUS protocol.

The RADIUS protocol (RFC 2865) has been in use for many years and has become the de-facto standard for providing AAA (Authentication Authorisation and Accounting) services. In the time since its release, it has revealed a number of serious shortcomings when used in some environments.

Many operators use RADIUS proxying to send conventional RADIUS requests across insecure networks as part of various global roaming or open campus projects. Typical are educational and other institutions which permit their users to roam on the networks of their partners, and arrange to send RADIUS authentication and accounting requests to the user's home RADIUS server, based on the user's Realm. This is called 'Realm-

based proxying', and is usually implemented with the Radiator AuthBy RADIUS clause.

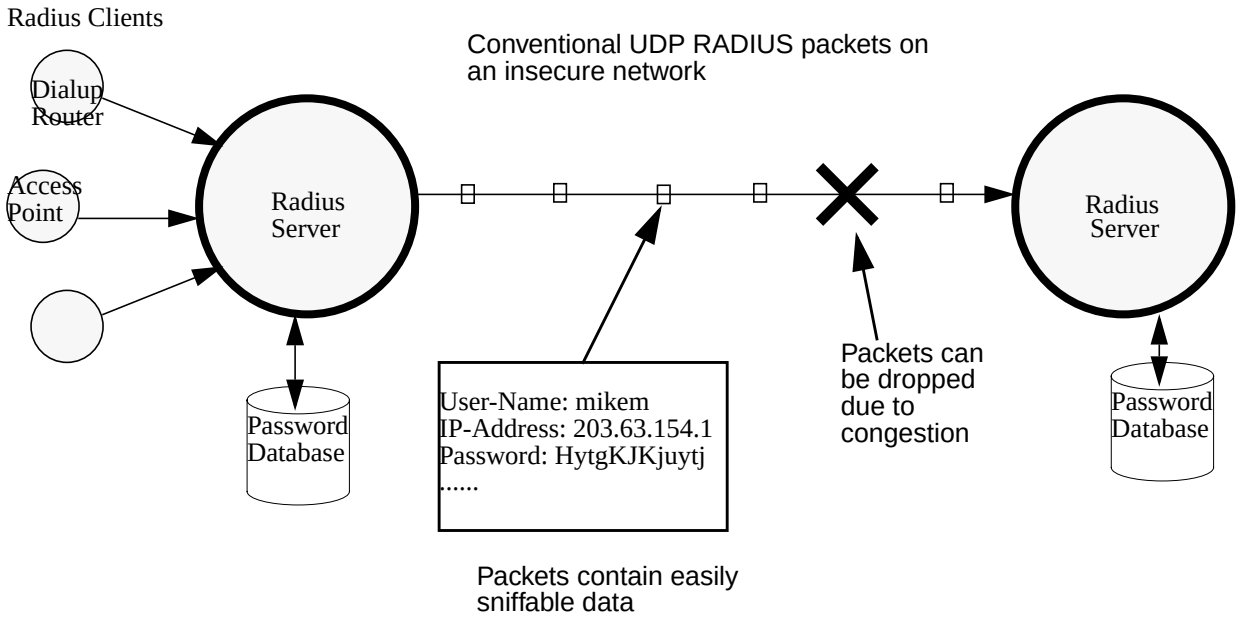
In the past this has forced operators to set up tunnels, IPsec or other mechanisms for their RADIUS requests, or else to brave the insecurities of the Internet.

The data in conventional RADIUS access requests is mostly plaintext, including the user name, IP address, login times etc. The user's password is encrypted with a shared secret, but using a fairly weak encryption algorithm. This means that eavesdroppers can gain valuable information by listening in on conventional RADIUS requests. This is not usually a problem where the RADIUS requests travel over an otherwise secure or private network, but it is a security problem when the RADIUS requests travel across the internet or any other insecure or shared network.

Furthermore, conventional RADIUS uses the unreliable User Datagram Protocol (UDP) for transport. UDP does not guarantee to deliver messages. The RADIUS protocol permits a limited number of retransmissions, but it does not guarantee to deliver requests, and therefore conventional RADIUS requests can sometimes be lost or dropped, especially on a congested network. This can cause inconvenience for users trying to log in, and lost accounting messages can mean lost income for operators.

Also, the conventional RADIUS protocol does not always provide a reliable indication of whether the RADIUS server you are connected to is the one you expect, or that the client that sends a request is really who it claims to be. This means that it is relatively easy to spoof RADIUS clients and servers when using conventional UDP based RADIUS proxying. This can also be used by attackers to gain valuable information about an operator's network and users.

FIGURE 1. Problems with conventional UDP RADIUS proxying



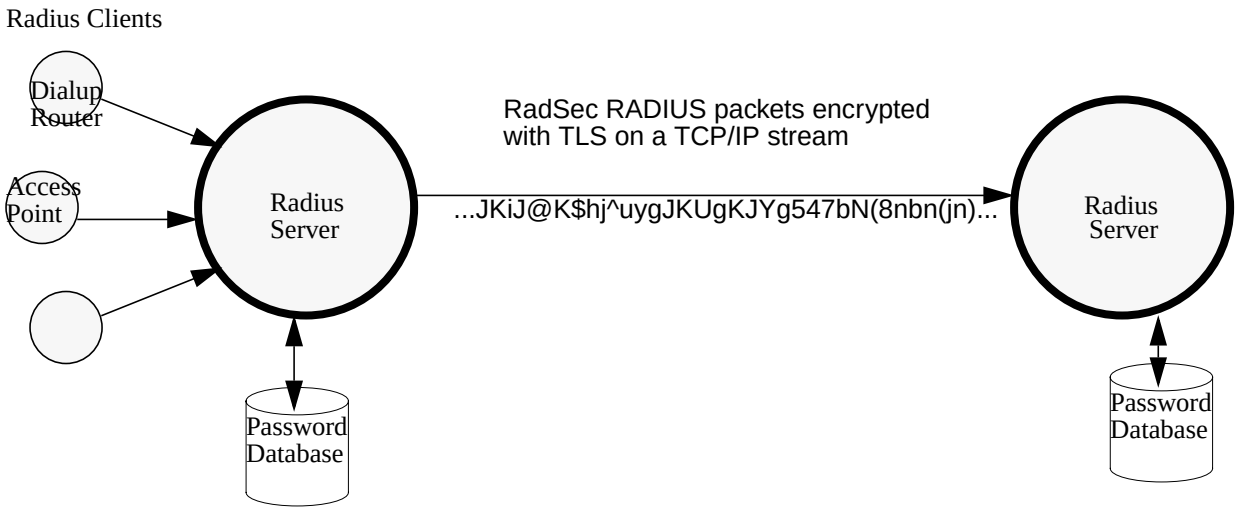
Many of these problems are intended to be solved by the forthcoming Diameter protocol (RFC 3588), which is expected to gradually replace RADIUS some time in the future. Open System Consultants will be making their raDiameter product available in the near future, but RadSec allows operators to alleviate all these problems now, and with a more lightweight solution.

RadSec is based on open, public standards. It uses the TCP/IP stream protocol to carry RADIUS requests, thereby eliminating the random loss of RADIUS requests. Further the RADIUS stream can be optionally encrypted with TLS (Transport Layer Security, a standard data encryption system based on SSL). Even more, it can be configured to provide mutual authentication with Public Key certificates, to prevent spoofing and masquerading of proxy RADIUS servers and clients.

The benefits of using TLS together with TCP/IP RADIUS streams are:

- Can’t be tampered with. Attackers cannot alter or remove RADIUS requests in the stream without the recipient detecting the tampering.
- Can’t be sniffed. All data in every RADIUS request is completely encrypted, so valuable information cannot be sniffed by eavesdroppers.
- Data can’t be lost without knowing. Random packet loss cannot occur without someone knowing about it, even on congested networks.
- Mutual authentication means the client is sure the server it connects to is really who it claims to be. And the server is sure the client is really who they claim to be.

FIGURE 2. RadSec proxying



3.0 Protocol Description

RadSec version 1.

1. RadSec is used to carry RADIUS traffic between 2 cooperating RADIUS servers, or between a RADIUS client and a RADIUS server. In either case, one end of the connection is the RadSec client and the other is the RadSec server.
2. A RadSec server can also be the client of another RadSec server. RadSec compliant instances can connect to multiple other RadSec instances. For some connections it can act as the client, and in others it will act as the server.
3. RadSec servers listen for connections on a well-known TCP/IP port. The default port number is 2083 (IANA official RadSec port number).
4. RadSec clients initiate a RadSec connection by establishing a TCP/IP connection to the address and port number of the RadSec server. The address and port number are expected to be well known by clients intending to connect to a given server (i.e. the server administrator will notify intending client administrators of the address and port number to connect to). TCP/IP keepalives will be enabled for the stream on platforms that support them.
5. After a TCP connection is established, TLS handshaking may commence if client and server are configured to use TLS. TLSV1 is used. RadSec servers are required to present a PKI certificate with a CN (Common Name) identical to the DNS host name corresponding to the address where the server listens. The server MAY be configured to require a PKI certificate from the client. If a valid certificate is not presented, the TLS handshake will fail. Certificates may be from either public or private certificate authorities.

6. If the TCP connection between client and server should fail or be disconnected, or the TLS handshake should fail for any reason, the client will attempt to reconnect to the server at configurable intervals, default 5 seconds.
7. TLS session resumption is not required or supported.
8. Once a stream connection is established (and TLS handshake successfully completed if required), RADIUS requests are sent from the client to the server and replies are sent from the server to the client through the stream. RADIUS packets are encoded on the stream in precisely the same format as a standard RADIUS UDP packet (see RFC 2865), including the request type, identifier and packet length. There are no record separators: the RADIUS packet length field is used to determine the record boundaries.
9. Like conventional RADIUS proxying, every RadSec connection will have a shared secret associated with it, and known to the RadSec Client and Server (regardless of whether or not TLS encryption will be used on the transport). The RADIUS authenticator and any User-Password fields in the RADIUS packets will be encrypted using the connections shared secret in the same was as conventional RADIUS UDP packets. This serves to provide at least the same level of protection as conventional RADIUS when RadSec is used without TLS encryption.
10. Compliant RadSec clients may use either the normal RADIUS packet identifier field or the Proxy-State attribute to match RADIUS replies to requests. It is recommended that Proxy-State be used to avoid the limitations of the 8 bit RADIUS identifier.
11. RADIUS replies may be sent back from the server in a different order to the received requests. There is no guarantee that reply will be sent in response to any particular request, for example when the server proxies the request still further to a home RADIUS server that does not reply.
12. Client may use the User-Name, Realm, or any other combination of RADIUS attributes to determine which RadSec server to send a RADIUS request to. This permits RADIUS servers to implement Realm-based proxying using RadSec to a user's home RadSec/ RADIUS server.
13. It is strongly recommended that TLS encryption be enabled and mutual RadSec client/server authentication required when using RadSec protocol across an insecure network.

4.0 Implementation

Radiator implements the RadSec protocol for proxying requests with the <`Authby RADSEC`> and `<ServerRADSEC>` clauses in the Radiator configuration file.

The `<AuthBy RADSEC>` clause defines a RadSec client, and causes Radiator to send RADIUS requests to the configured RadSec server using the RadSec protocol.

The `<ServerRADSEC>` clause defines a RadSec server, and causes Radiator to list on the configured port and address(es) for connections from `<Authby RADSEC>` clients.

When an `<Authby RADSEC>` client connects to a `<ServerRADSEC>` server, the client sends RADIUS requests through the stream to the server. The server then services

the request in the same was as if the request had been received from a conventional UDP RADIUS client.

Radiator's implementation allows operators to choose between TCP (Transmission Control Protocol) or SCTP (Stream Control Transmission Protocol). See http://www.sctp.org or http://www.networksorcery.com/enp/protocol/sctp.htm for more information on SCTP.

Radiator's implementation allows operators to configure RadSec servers to require client authentication. Server authentication is always performed by RadSec clients. It also allows clients to be configured to specify the name they expect to see as the CN on the server certificate, permitting the use of alternate certificate names by servers.

5.0 Typical Use

The most common use of RadSec is to transport RADIUS requests and replies securely across an unreliable or insecure network. Typical would be an IP carrier who has RADIUS clients at various POPs around the country, and RADIUS servers at their core location. Radiators equipped with RadSec can be set up at each POP in order to carry encrypted RADIUS across the intervening network to their core Radiator servers.

Another common use is to carry RADIUS requests between the RADIUS servers of a group of roaming partners. In this situation, several operators cooperate in providing IP access to each other's users or customers. The operators use each user's realm to determine where to send RADIUS access and accounting requests. Generally this will mean proxying the requests across the Internet to the user's home operator. RadSec can be used to encrypt this inter-operator RADIUS traffic.

It is also reasonable to use `<AuthBy RADSEC>` in almost any other situation where AuthBy RADIUS (i.e. conventional UDP RADIUS proxying) would otherwise be used, provided Radiator or some other RadSec compliant application can be established at each end of the connection.

6.0 Notes

Due to limitations in the implementation of TCP/IP sockets on Microsoft Windows, it is not possible to get carrier grade performance from RadSec when Radiator runs on Microsoft Windows.

Operators can use the CATool Certificate Authority product from Open System Consultants to generate private certificates for RadSec clients and servers. See http://www.open.com.au/catool.