RADIUS Extensions Working Group
Internet-Draft
Intended status: Experimental
Expires: April 27, 2009

S. Winter
RESTENA
M. McCauley
OSC
S. Venaas
UNINETT
K. Wierenga
Cisco

October 24, 2008

TLS encryption for RADIUS over TCP (RadSec)
draft-ietf-radext-radsec-02

Status of This Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

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."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on April 27, 2009.

Abstract

This document specifies security on the transport layer (TLS) for the RADIUS protocol [RFC2865] when transmitted over TCP [I-D.dekok-radext-tcp-transport]. This enables dynamic trust relationships between RADIUS servers.

Table of Contents

1. Introduction

The RADIUS protocol [RFC2865] is a widely deployed authentication and authorisation protocol. The supplementary RADIUS Accounting specification [RFC2866] also provides accounting mechanisms, thus delivering a full AAA solution. However, RADIUS is experiencing several shortcomings, such as its dependency on the unreliable transport protocol UDP and the lack of security for large parts of its packet payload. RADIUS security is based on the MD5 algorithm, which has been proven to be insecure.

The main focus of RadSec is to provide a means to secure the communication between RADIUS/TCP peers on the transport layer. The most important use of RadSec lies in roaming environments where RADIUS packets need to be transferred through different administrative domains and untrusted, potentially hostile networks. An example for a world-wide roaming environment that uses RadSec to secure communication is "eduroam", see [eduroam].

There are multiple known attacks on the MD5 algorithm which is used in RADIUS to provide integrity protection and a limited confidentiality protection. RadSec wraps the entire RADIUS packet payload into a TLS stream and thus mitigates the risk of attacks on MD5.

Because of the static trust establishment between RADIUS peers (IP address and shared secret) the only scalable way of creating a massive deployment of RADIUS-servers under control by different administrative entities is to introduce some form of a proxy chain to route the access requests to their home server. This creates a lot of overhead in terms of possible points of failure, longer transmission times as well as middleboxes through which authentication traffic flows. These middleboxes may learn privacy-relevant data while forwarding requests. The new features in RadSec obsolete the use of IP addresses and shared MD5 secrets to identify other peers and thus allow the dynamic establishment of connections to peers that are not previously configured, and thus makes it possible to avoid intermediate aggregation proxies. The definition of lookup mechanisms is out of scope of this document, but an implementation of a DNS NAPTR lookup based mechanism exists and is described as an example lookup mechanism in Appendix A.

1.1. Requirements Language

In this document, several words are used to signify the requirements of the specification. The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in

RFC 2119. [RFC2119]

1.2. Terminology

RadSec node: a RadSec client or server

RadSec Client: a RadSec instance which initiates a new connection.

RadSec Server: a RadSec instance which listens on a RadSec port and accepts new connections

2. Normative: Transport Layer Security for RADIUS over TCP

2.1. TCP port and packet types

The default destination port number for RadSec is TCP/2083. There are no separate ports for authentication, accounting and dynamic authorisation changes. The source port is arbitrary.

2.2. Connection Setup

RadSec nodes

1. establish TCP connections as per [I-D.dekok-radext-tcp-transport]

2. negotiate TLS sessions according to [RFC5246] or its predecessor TLS 1.1. The following restrictions apply:

   * The authentication MUST be mutual, i.e. both the RadSec server and the RadSec client authenticate each other.

   * When using X.509 certificates, RadSec servers SHOULD indicate their acceptable Certification Authorities as per section 7.4.4 of [RFC5246] (see Section 3.1 (1))

   * When using X.509 certificates, the TLS Extension "Trusted CA Indication" from [RFC5246] or its TLS 1.1 predecessor SHOULD be used to indicate trusted CAs for the client (see Section 3.1 (2))

   * When using X.509 certificates, certificate validation is performed as per [RFC5280] or its predecessor. The client MAY perform additional checks to accomodate for different trust models.

   * The client MUST NOT negotiate cipher suites which only provide integrity protection.

* The cipher suite TLS_RSA_WITH_3DES_EDE_CBC_SHA MUST be supported.

* The cipher suites TLS_RSA_WITH_AES_128_CBC_SHA and TLS_RSA_WITH_RC4_128_SHA SHOULD be supported. (see Section 3.2 (1))

3. start exchanging RADIUS datagrams. Note Section 3.3 (1)). The shared secret to compute the (obsolete) MD5 integrity checks and attribute encryption MUST be "radsec" (see Section 3.3 (2)).

2.3. RADIUS Datagrams

Authentication, Accounting and Authorization packets are sent according to the following rules:

RadSec clients handle the following packet types from [RFC2865], [RFC2866], [RFC5176] on the connection they initiated (see Section 3.3 (3) and (4)):

o send Access-Request

o send Accounting-Request

o send Status-Server

o send Disconnect-ACK

o send Disconnect-NAK

o send CoA-ACK

o send CoA-NAK

o receive Access-Challenge

o receive Access-Accept

o receive Access-Reject

o receive Accounting-Response

o receive Disconnect-Request

o receive CoA-Request

RadSec servers handle the following packet types from [RFC2865], [RFC2866], [RFC5176] on the connections they serve to clients:

- o receive Access-Request
- o receive Accounting-Request
- o receive Status-Server
- o receive Disconnect-ACK
- o receive Disconnect-NAK
- o receive CoA-ACK
- o receive CoA-NAK
- o send Access-Challenge
- o send Access-Accept
- o send Access-Reject
- o send Accounting-Response
- o send Disconnect-Request
- o send CoA-Request

3. Informative: Design Decisions

This section explains the design decisions that led to the rules defined in the previous section.

3.1. X.509 Certificate Considerations

(1) If a RadSec client is in possession of multiple certificates from different CAs (i.e. is part of multiple roaming consortia) and dynamic discovery is used, the discovery mechanism possibly does not yield sufficient information to identify the consortium uniquely (e.g. DNS discovery). Subsequently, the client may not know by itself which client certificate to use for the TLS handshake. Then it is necessary to for the server to signal which consortium it belongs to, and which certificates it expects. If there is no risk of confusing multiple roaming consortia, providing this information in the handshake is not crucial.

(2) If a RadSec server is in possession of multiple certificates from different CAs (i.e. is part of multiple roaming consortia), it will need to select one of its certificates to present to the RadSec client. If the client sends the Trusted CA Indication, this hint can

make the server select the appropriate certificate and prevent a handshake failure. Omitting this indication makes it impossible to deterministically select the right certificate in this case. If there is no risk of confusing multiple roaming consortia, providing this indication in the handshake is not crucial.

(3) When using X.509 certificate validation, peer validation always includes a check on whether the DNS name or the IP address of the server that is contacted matches its certificate. When a RadSec peer establishes a new connection (acts as a client) to another peer, the following rules of precedence are used during validation:

- If the client expects a certain fully qualified domain name (FQDN) and the presented certificate contains both at least one instance of the subjectAltName field with type dNSName and a Common Name, then the certificate is considered a match if any one of those subjectAltName fields matches the expected FQDN. The Common Name field is not evaluated in this case.

- If the client expects a certain fully qualified domain name (FQDN) and the presented certificate does not contain any subjectAltName field with type dNSName, then the certificate is considered a match if the Common Name field matches the expected FQDN.

- If the client expects a certain IP address and the presented certificate contains both at least one instance of the subjectAltName field with type iPAddr and a Common Name, then the certificate is considered a match if any one of those subjectAltName fields matches the expected IP address. The Common Name field is not evaluated in this case.

- If the client expects a certain IP address and the presented certificate does not contain any subjectAltName field with type iPAddr, then the certificate is considered a match if the Common Name field matches the expected IP address.

(4) If dynamic peer resolution is used, the above verification steps may not be sufficient to ensure that a connecting peer is authorised to perform user authentications. In these cases, the trust fabric cannot depend on peer authentication methods like DNSSEC to identify RadSec nodes. The RadSec nodes also need to be properly authorised. Operators of a RadSec infrastructure should define their own authorisation trust model and apply this model to the certificates after they have passed the standard validity checks above. Current RadSec implementations offer varying degrees of versatility for defining criteria which certificates to accept.

3.2. Ciphersuites and Compression Negotiation Considerations

RadSec implementations need not necessarily support all TLS ciphersuites listed in [RFC5246]. Not all TLS ciphersuites are supported by available TLS tool kits and licenses may be required in some cases. The existing implementations of RadSec use OpenSSL as cryptographic backend, which supports all of the ciphersuites listed in the rules in the normative section.

The TLS cphersuite TLS_RSA_WITH_3DES_EDE_CBC_SHA is mandatory-to-implement according to [RFC5246] and thus has to be supported by RadSec nodes.

The two other ciphersuites in the normative section (TLS_RSA_WITH_RC4_128_SHA and TLS_RSA_WITH_AES_128_CBC_SHA) are widely implemented in TLS toolkits and are considered good practice to implement.

TLS also supports compression. Compression is an optional feature. During the RadSec conversation the client and server may negotiate compression, but must continue the conversation even if the other peer rejects compression.

3.3. RADIUS Datagram Considerations

(1) After the TLS session is established, RADIUS packet payloads are exchanged over the encrypted TLS tunnel. In plain RADIUS, the packet size can be determined by evaluating the size of the datagram that arrived. Due to the stream nature of TCP and TLS, this does not hold true for RadSec packet exchange. Instead, packet boundaries of RADIUS packets that arrive in the stream are calculated by evaluating the packet's Length field. Special care needs to be taken on the packet sender side that the value of the Length field is indeed correct before sending it over the TLS tunnel, because incorrect packet lengths can no longer be detected by a differing datagram boundary.

(2) Within RADIUS [RFC2865], a shared secret is used for hiding of attributes such as User-Password, as well as in computation of the Response Authenticator. In RADIUS accounting [RFC2866], the shared secret is used in computation of both the Request Authenticator and the Response Authenticator. Since TLS provides integrity protection and encryption sufficient to substitute for RADIUS application-layer security, it is not necessary to configure a RADIUS shared secret. The use of a fixed string for the obsolete shared secret eliminates possible node misconfigurations.

(3) RADIUS [RFC2865] uses different UDP ports for authentication,

accounting and dynamic authorisation changes. RadSec allocates a single port for all RADIUS packet types. Nevertheless, in RadSec the notion of a client which sends authentication requests and processes replies associated with it's users' sessions and the notion of a server which receives requests, processes them and sends the appropriate replies is to be preserved. The normative rules about acceptable packet types for clients and servers mirror the packet flow behaviour from RADIUS.

(4) RADIUS [RFC2865] used negative ICMP responses to a newly allocated UDP port to signal that a peer RADIUS server does not support reception and processing of the packet types in [RFC5176]. These packet types are listed as to be received in RadSec implementations. Note well: it is not required for an implementation to actually process these packet types. It is sufficient that upon receiving such a packet, an unconditional NAK is sent back to indicate that the action is not supported.

4. Diameter Compatibility

Since RadSec is only a new transport profile for RADIUS, compatibility of RadSec - Diameter [RFC3588] vs. RADIUS [RFC2865] - Diameter [RFC3588] is identical. The considerations regarding payload size in [I-D.dekok-radext-tcp-transport] apply.

5. Security Considerations

The computational resources to establish a TLS tunnel are significantly higher than simply sending mostly unencrypted UDP datagrams. Therefore, clients connecting to a RadSec node will more easily create high load conditions and a malicious client might create a Denial-of-Service attack more easily.

In the case of dynamic peer discovery, a RadSec node needs to be able to accept connections from a large, not previously known, group of hosts, possibly the whole internet. In this case, the server's RadSec port can not be protected from unauthorised connection attempts with measures on the network layer, i.e. access lists and firewalls. This opens more attack vectors for Distributed Denial of Service attacks, just like any other service that is supposed to serve arbitrary clients (like for example web servers).

In the case of dynamic peer discovery, X.509 certificates are the only proof of authorisation for a connecting RadSec nodes. Special care needs to be taken that certificates get verified properly according to the chosen trust model (particularly: consulting CRL lists, checking critical extensions, checking subjectAltNames etc.) to prevent unauthorised connections.

Some TLS ciphersuites only provide integrity validation of their payload, and provide no encryption. This specification forbids the use of such ciphersuites. Since the RADIUS payload's shared secret is fixed and well-known, failure to comply with this requirement will expose the entire datagram payload in plain text, including User-Password, to intermediate IP nodes.

6. IANA Considerations

This document has no actions for IANA. The TCP port 2083 was already previously assigned by IANA for RadSec. No new RADIUS attributes or packet codes are defined.

7. Acknowledgements

RadSec version 1 was first implemented by Open Systems Consultants, Currumbin Waters, Australia, for their "Radiator" RADIUS server product (see [radsec-whitepaper]).

Funding and input for the development of this Internet Draft was provided by the European Commission co-funded project "GEANT2" [geant2] and further feedback was provided by the TERENA Task Force Mobility [terena].

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC2866] Rigney, C., "RADIUS Accounting", RFC 2866, June 2000.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC5176] Chiba, M., Dommety, G., Eklund, M., Mitton, D., and B. Aboba, "Dynamic Authorization Extensions to Remote Authentication Dial In User Service (RADIUS)", RFC 5176, January 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[I-D.dekok-radext-tcp-transport] DeKok, A., "RADIUS Over TCP", draft-dekok-radext-tcp-transport-00 (work in progress), July 2008.

8.2. Informative References

[RFC3588] Calhoun, P., Loughney, J., Guttman, E., Zorn, G., and J. Arkko, "Diameter Base Protocol", RFC 3588, September 2003.

[radsec-whitepaper] Open System Consultants, "RadSec - a secure, reliable RADIUS Protocol", May 2005, <http://www.open.com.au/radiator/radsec-whitepaper.pdf>.

[radiator-manual] Open System Consultants, "Radiator Radius Server - Installation and Reference Manual", 2006, <http://www.open.com.au/radiator/ref.html>.

[radsecproxy-impl] Venaas, S., "radsecproxy Project Homepage", 2007, <http://software.uninett.no/radsecproxy/>.

[eduroam] Trans-European Research and Education Networking Association, "eduroam Homepage", 2007, <http://www.eduroam.org/>.

[geant2] Delivery of Advanced Network Technology to Europe, "European Commission Information Society and Media: GEANT2", 2008, <http://www.geant2.net/>.

[terena] TERENA, "Trans-European Research and Education Networking Association", 2008, <http://www.terena.org/>.

Appendix A. DNS NAPTR Peer Discovery

The following text is paraphrased from the file goodies/dnsroam.cfg in the Radiator distribution; further documentation of the <AuthBy DNSROAM> feature in Radiator can be found at [radiator-manual]. It describes an algorithm to retrieve the RadSec route information from the global DNS using NAPTR and SRV records. The input of the algorithm is the realm part of the user name.

The following algorithm is used to discover a target server from a Realm using DNS:

1. Look for NAPTR records for the Realm. If found, continue at step 2, otherwise continue at step 4.

2. For each NAPTR found record, examine the Service field and use that to determine the transport protocol and TLS requirements for the server. The Service field starts with 'AAA' for insecure and 'AAAS' for TLS secured. The Service field contains '+RADSECS' for RadSec over SCTP, '+RADSECT' for RadSec over TCP or '+RADIUS' for RADIUS protocol over UDP. The most common Service field you will see will be 'AAAS+RADSECT' for TLS secured RadSec over TCP.

3.

    A. If the NAPTR has the 'S' flag, look for SRV records for the name. For each SRV record found, note the Port number and then look for A and AAAA records corresponding to the name in the SRV record.

    B. If the NAPTR has the 'A' flag, look for a A and AAAA records for the name.

4. All A and AAAA records found are ordered according to their Order and Preference fields. The most preferable server address is used as the target server address, along with any other server attributes discovered from DNS. If no SRV record was found for the address, the DNSROAM configured Port is used. Algorithm terminates.

5. Look for A and AAAA records on the literal realm name, preceded by "_radsec._tcp.". For example, if the realm is 'example.com', it looks for the record '_radsec._tcp.example.com'. If more than

one result is returned, no ordering is assumed. Algorithm terminates.

For example, if the User-Name realm was 'example.com', and DNS contained the following records:

```
example.com.  IN NAPTR 50 50 "s" "AAAS+RADSECT" ""
_radsec._tcp.example.com.

_radsec._tcp.example.com.  IN SRV 0 10 2083 radsec.example.com.

radsec.example.com.  IN AAAA 2001:0DB8::202:44ff:fe0a:f704
```

Then the target selected would be a RadSec server on port 2083 at IPv6 address 2001:0DB8::202:44ff:fe0a:f704. The connection would be made over TCP/IP, and TLS encryption would be used. This complete specification of the realm is the most flexible and is recommended.

Appendix B. Implementation Overview: Radiator

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 listen 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 is compliant to version 2 of RadSec if the following options are used:

<AuthBy RADSEC>

* Protocol tcp

* UseTLS

* TLS_CertificateFile

* Secret radsec

<ServerRADSEC>

* Protocol tcp

* UseTLS

* TLS_RequireClientCert

* Secret radsec

As of Radiator 3.15, the default shared secret for RadSec connections is configurable and defaults to "mysecret" (without quotes). For compliance with this document, this setting needs to be configured for the shared secret "radsec". The implementation uses TCP keepalive socket options, but does not send Status-Server packets. Once established, TLS connections are kept open throughout the server instance lifetime.

Appendix C. Implementation Overview: radsecproxy

The RADIUS proxy named radsecproxy was written in order to allow use of RadSec in current RADIUS deployments. This is a generic proxy that supports any number and combination of clients and servers, supporting RADIUS over UDP and RadSec. The main idea is that it can be used on the same host as a non-RadSec client or server to ensure RadSec is used on the wire, however as a generic proxy it can be used in other circumstances as well.

The configuration file consists of client and server clauses, where there is one such clause for each client or server. In such a clause one specifies either "type tls" or "type udp" for RadSec or UDP transport. For RadSec the default shared secret "mysecret" (without quotes), the same as Radiator, is used. A secret may be specified by putting say "secret somesharedsecret" inside a client or server clause.

In order to use TLS for clients and/or servers, one must also specify where to locate CA certificates, as well as certificate and key for the client or server. This is done in a TLS clause. There may be one or several TLS clauses. A client or server clause may reference a particular TLS clause, or just use a default one. One use for multiple TLS clauses may be to present one certificate to clients and another to servers.

If any RadSec (TLS) clients are configured, the proxy will at startup listen on port 2083, as assigned by IANA for the OSC RadSec

implementation. An alternative port may be specified. When a client connects, the client certificate will be verified, including checking that the configured FQDN or IP address matches what is in the certificate. Requests coming from a RadSec client are treated exactly like requests from UDP clients.

The proxy will at startup try to establish a TLS connection to each (if any) of the configured RadSec (TLS) servers. If it fails to connect to a server, it will retry regularly. There is some back-off where it will retry quickly at first, and with longer intervals later. If a connection to a server goes down it will also start retrying regularly. When setting up the TLS connection, the server certificate will be verified, including checking that the configured FQDN or IP address matches what is in the certificate. Requests are sent to a RadSec server just like they would to a UDP server.

The proxy supports Status-Server messages. They are only sent to a server if enabled for that particular server. Status-Server requests are always responded to.

This RadSec implementation has been successfully tested together with Radiator. It is a freely available open-source implementation. For source code and documentation, see [radsecproxy-impl].

Authors' Addresses

Stefan Winter
Fondation RESTENA
6, rue Richard Coudenhove-Kalergi
Luxembourg 1359
LUXEMBOURG

Phone: +352 424409 1
Fax: +352 422473
EMail: stefan.winter@restena.lu
URI: http://www.restena.lu.

Mike McCauley
Open Systems Consultants
9 Bulbul Place
Currumbin Waters QLD 4223
AUSTRALIA

Phone: +61 7 5598 7474
Fax: +61 7 5598 7070
EMail: mikem@open.com.au
URI: http://www.open.com.au.

Stig Venaas
UNINETT
Abels gate 5 - Teknobyen
Trondheim 7465
NORWAY

Phone: +47 73 55 79 00
Fax: +47 73 55 79 01
EMail: stig.venaas@uninett.no
URI: http://www.uninett.no.

Klaas Wierenga
Cisco Systems International BV
Haarlerbergweg 13-19
Amsterdam 1101 CH
The Netherlands

Phone: +31 (0)20 3571752
Fax:
EMail: kwiereng@cisco.com
URI: http://www.cisco.com.

Acknowledgement

This document was produced using xml2rfc v1.33 (of http://xml.resource.org/) from a source in RFC-2629 XML format.