Internet Draft

Document: draft-moskowitz-RADIUS-Client-Kickstart-01.txt

Expires: May 2004

R. Moskowitz
A.DeKok
ICSAlabs
IDT

November 2003

RADIUS Client Kickstart

Status of this Memo

Abstract

RADIUS servers [2] require foreknowledge of the IP address of the RADIUS clients, as the shared secret is bound to the address. This has been a manageable situation when the RADIUS Clients were just NASs (Network Access Servers). With the advent of IEEE 802.1x [3], there is a significant increase in RADIUS clients in organizations not prepared to have the RADIUS Clients use fixed IP addresses and manage the shared secret. To address the concerns of the IEEE 802.1 and 802.11 Task Groups a level of indirection is added; a Master secret bound to the name of the RADIUS client. This Master secret is created by the Shared Secret Provisioning Protocol [4]. For RADIUS Client Kickstart, SSPP is run over SNMP [5]. The Master Secret is used in an initial RADIUS exchange to create a session

secret that is used as the normal RADIUS client shared secret. SSPP can be used to change the Master Secret whenever required.

Conventions used in this document

In examples, "C:" and "S:" indicate lines sent by the RADIUS client and server respectively.

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 [6].

Table of Contents

1. Introduction

The IEEE 802.1x Port-Based Network Access Control standard recommends the use of RADIUS for the Authentication Server, making the 802.1x Authenticators RADIUS Clients. RADIUS Clients have some well-defined security configuration requirements that will present challenges to effective 802.1x deployments as is anticipated for 802.1 compliant switches and 802.11i [7] Access Points.

The RADIUS server MUST have the RADIUS Client Shared Secret bound to the RADIUS ClientÆs IP Address. This normally requires the RADIUS Client to have a fixed IP address, which is not a DHCP dynamically assigned address. The RADIUS Client needs a method for entering the secret into the RADIUS Client securely and changing it. This

document discusses the RADIUS Client environment, the constraints it places on 802.1x deployments and a method to 'plumb' a RADIUS Client. The information imparted here is not expected to be included within any IEEE document, even though it impacts 802.1x, 802.11i and 802.11f implementations. If this information and methodology were included anywhere within IEEE, it would be 802.1x Annex D.

1.1 Problem Statement

802.1x states in Annex D, (informative text)

> IEEE 802.1X RADIUS Usage Guidelines
>
> D.1 Introduction
>
> IEEE Std 802.1X-2001 enables authenticated access to IEEE 802 media, including Ethernet, Token Ring, and IEEE 802.11 wireless LANs. Although RADIUS support is optional within IEEE Std 802.1X-2001, it is expected that most IEEE Std 802.1X-2001 Authenticators will function as RADIUS clients.

This text has been replaced in the 802.1x update, 802.1aa with:

> In situations where it is desirable to centrally manage authentication, authorization and accounting (AAA) for IEEE 802 networks, deployment of a backend authentication and accounting server is desirable. In such situations, it is expected that IEEE 802.1X Authenticators will function as AAA clients.

This only generalizes Annex D for any AAA client/server, including the aforementioned RADIUS usage.

This functioning as RADIUS clients results in adhering to the following from RADIUS RFC (2865), Section 3.

> Administrative Note
>
> The secret (password shared between the RADIUS Client and the RADIUS server) SHOULD be at least as large and unguessable as a well-chosen password. It is preferred that the secret be at least 16 octets. This is to ensure a sufficiently large range for the secret to provide protection against exhaustive search attacks. The secret MUST NOT be empty (length 0) since this would allow packets to be trivially forged.

A RADIUS server MUST use the source IP address of the RADIUS UDP packet to decide which shared secret to use, so that RADIUS requests can be proxied.

Simply put, every AP implementing RADIUS support MUST:

- Provide a secure method for entering the RADIUS Client secret that SHOULD be at least 16 bytes.

- Either provide the RADIUS Server with the RADIUS ClientÆs IP address, or provide a name that can be readily resolved to its IP address, for example the RADIUS ClientÆs DNS name.

The first casualty is the use by the switch or AP of DHCP for its IP address assignment. DHCP can only be used if the DHCP server always leases the same address to the switch or AP, or the RADIUS Server can query the DHCP Server for the switch's or AP's IP address by a DNS-styled name.

The second casualty is switch or AP auto configuration for small offices. You MUST enter the shared secret into the switch or AP in addition to setting the switch's or AP's IP address information (that most small office support people are not trained for).

The final casualty is network security. Secret changes after personnel changes are cumbersome and thus may not be done.

2. Adding a level of indirection to the RADIUS Client Secret

This conundrum can be dealt with by adding a level of indirection:

- Creation of a new RADIUS Client table in the Server, keyed by a RADIUS Client 'name' and a master secret.

- A RADIUS Client boot or registration protocol to update the traditional RADIUS Client Database of IPaddress and shared secret. This registration protocol will use the RADIUS Client name and master secret.

This new RADIUS Client name/secret database needs a method to effectively populate it and maintain it. This can be achieved by:

- A master secret bootstrap process based on SSPP over SNMP.

- A master secret change process, also based on SSPP over SNMP.

2.1 Basic components and protocols

When SSPP is used for maintaining the RADIUS Client name/secret database, the RADIUS client is the SSPP server and the RADIUS server is the SSPP client.

The SSPP domain parameters will be the Diffie-Hellman fixed field, g and p. Q is not needed in the protocol as it is only used in the calculation of p, and all values of p used are set below, provided from work in IPsec [8]. The cyrptographic community has produced many estimates of Diffie-Hellman key size to provide 128 bits of symmetric key strength. The generally agreed range is from a Diffie-Hellman size of 2048 to 3072. The 3072 length SHOULD be used unless the RADIUS Client is so constrained, as this is not practical, then the 2048 length MAY be used. It should be noted that Kickstart is a very infrequently used protocol and that in many cases, a long computational time will not be an impediment to its use.

```
Parameter   Value


Key length  2048

P        is: 2^2048 - 2^1984 - 1 + 2^64 * { [2^1918 pi] + 124476 }
         Its hexadecimal value is:

         FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1
         29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD
         EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
         E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED
         EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D
         C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F
         83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D
         670C354E 4ABC9804 F1746C08 CA18217C 32905E46 2E36CE3B
         E39E772C 180E8603 9B2783A2 EC07A28F B5C55DF0 6F4C52C9
         DE2BCBF6 95581718 3995497C EA956AE5 15D22618 98FA0510
         15728E5A 8AACAA68 FFFFFFFF FFFFFFFF

g        2 (decimal)

Key length  3072

P        is: 2^3072 - 2^3008 - 1 + 2^64 * { [2^2942 pi] + 1690314 }
         Its hexadecimal value is:

         FFFFFFFF FFFFFFFF C90FDAA2 2168C234 C4C6628B 80DC1CD1
         29024E08 8A67CC74 020BBEA6 3B139B22 514A0879 8E3404DD
         EF9519B3 CD3A431B 302B0A6D F25F1437 4FE1356D 6D51C245
```

```
         E485B576 625E7EC6 F44C42E9 A637ED6B 0BFF5CB6 F406B7ED
         EE386BFB 5A899FA5 AE9F2411 7C4B1FE6 49286651 ECE45B3D
         C2007CB8 A163BF05 98DA4836 1C55D39A 69163FA8 FD24CF5F
         83655D23 DCA3AD96 1C62F356 208552BB 9ED52907 7096966D
         670C354E 4ABC9804 F1746C08 CA18217C 32905E46 2E36CE3B
         E39E772C 180E8603 9B2783A2 EC07A28F B5C55DF0 6F4C52C9
         DE2BCBF6 95581718 3995497C EA956AE5 15D22618 98FA0510
         15728E5A 8AAAC42D AD33170D 04507A33 A85521AB DF1CBA64
         ECFB8504 58DBEF0A 8AEA7157 5D060C7D B3970F85 A6E1E4C7
         ABF5AE8C DB0933D7 1E8C94E0 4A25619D CEE3D226 1AD2EE6B
         F12FFA06 D98A0864 D8760273 3EC86A64 521F2B18 177B200C
         BBE11757 7A615D6C 770988C0 BAD946E2 08E24FA0 74E5AB31
         43DB5BFC E0FD108E 4B82D120 A93AD2CA FFFFFFFF FFFFFFFF

g        2 (decimal)
```

These are the same values as IKE Oakley Groups 14 and 15.

RADIUS Client Master database:

This table consists of the following entries:

  RADIUS Client Name
  RADIUS Client Public Diffie-Hellman value
  Current Master secret
  Date of last Master secret change
  Date of last Session secret

Function 1 û RADIUS Client Master Secret Bootstrap

This is a protocol that will facilitate the RADIUS Server's discovery of a new Access Point or NAS and provide for an over-the-wire establishment of a Master secret. This protocol MUST support multiple RADIUS Servers each having a unique Master secret with the RADIUS Client.

Function 2 û RADIUS Client Master Secret Change

This protocol will allow the RADIUS Server to trigger a change of the Master Secret. This is provided by the SSPP over SNMP changing a secret exchange. After the Master Secret is changed, the RADIUS Client MUST establish a new RADIUS Client Secret via the RADIUS Client Registration Session Secret.

Function 3 û RADIUS Client Boot Session Secret

When the RADIUS Client boots, it MUST use its Name and Master Secret to authenticate its IPaddress to the Server and establish a Randomly

selected Session Secret as the RADIUS Client shared secret in RFC 2865. If the Master Secret is changed via Function 2, a new boot Session Secret will also be created.

3. The MIB Objects

The SSPP Server has some specific MIB objects and two tables of objects for each SSPP Client. The SSPP Proxy also has a table of objects. The SSPP Server specific objects are:

Domain Parameters: g and p
Diffie-Hellman key pair (the private key MUST be protected from reading)
SSPP Server Address
Nonce

The SSPP Server's SSPP Client table objects are:

SSPP Client Address
SSPP Client Diffie-Hellman public key
SSPP Client Nonce
SSPP Client Signature
SSPP Server Signature
Change Flag
Shared Secret (MUST be protected from reading)

The SSPP Server's Proxied SSPP Client table is a temporary table with objects:

SSPP Proxy Address
SSPP Proxy Signature
SSPP Client Address
SSPP Client Diffie-Hellman public key
SSPP Client Nonce
SSPP Client Signature
SSPP Server Signature
Shared Secret (MUST be protected from reading)

The SSPP Proxy's SSPP Client table is a temporary table with objects:

SSPP Client Address
SSPP Client Diffie-Hellman public key
SSPP Client Nonce
SSPP Client Signature
ForwardFlag
Proxy Signature

A SSPP Client SHOULD keep the following objects in the MIB:

Domain Parameters
Diffie-Hellman key pair (the private key MUST be protected from reading)
SSPP Client Address
Nonce
SSPP Server Address
SSPP Server Signature
SSPP Proxy Address
Shared Secret (MUST be protected from reading)

4. RADIUS Client Master Secret Bootstrap

The RADIUS Client Master secret will be the result of an SSPP basic exchange over SNMP between the SSPP Client and the SSPP Server. The RADIUS Server is the SSPP Client and the RADIUS Client is the SSPP Server. Once a Master Secret is set, the RADIUS Client MUST not perform another bootstrap until it is reset to its initial status.

The RADIUS Server SHOULD have an SNMP based discovery process, identifying potential clients by the presence of the RADIUS Client MIB objects. The RADIUS Server MUST support the SSPP client function, both of the basic and the proxy exchanges. The RADIUS Server SHOULD support the SSPP proxy function.

The RADIUS Client MUST restrict the SSPP basic exchange to one RADIUS Server. After the first server performs the basic exchange, the RADIUS Client MUST reject any other RADIUS server performing the basic exchange, and MUST only accept the change secret and proxy exchanges. The RADIUS Client MUST have a 'local' function, like a reset button, to remove all RADIUS Server associations.

5. RADIUS Client Master Secret Change

A Master Secret change uses the SSPP over SNMP changing a secret exchange. The RADIUS Server initiates it.

6. RADIUS Client Registration

The RADIUS Client Registration process uses new RADIUS packets: Access-Boot and Access-Booted. These RADIUS packets are NEVER proxied across RADIUS servers. The Access-Boot is similar to the Access-Request. The Access-Booted is similar to the Access-Accept.

6.1 RADIUS Client Request

For this process, the RADIUS client has a Master shared secret, an IP address, and that it knows the IP address of the RADIUS server it will use.   The purpose of RADIUS client registration is to allow the client to inform the RADIUS server of its new IP address.

Once it has received an IP address (via some method outside of this specification), the RADIUS client "registers" with the RADIUS server. It does this by sending a self-signed RADIUS packet, of type Access-Boot, to the RADIUS server. The packet contains the IPv4 (or IPv6) address of the NAS, along with the UDP source port, an identifier for the NAS (usually MAC address, or name), and a timestamp. This packet is signed with a Message-Authenticator attribute.

The RADIUS server uses the Calling-Station-Id (usually containing the MAC address of the client), or the NAS-Identifier, to look up the shared secret in its database. The Master shared secret is then used to validate the Message-Authenticator attribute in the request. This process ensures that the RADIUS server can verify that the client possesses the same correct secret.

If the source IP (or IPv6) address of the Access-Boot packet does not match the contents of the NAS-IP-Address (or NAS-IPv6-Attribute) packet, then the request MUST be silently discarded, and a response MUST NOT be sent back to the NAS. Access-Boot packets MAY NOT be proxied.

If the source UDP port of the Access-Boot packet does not match the contents of the NAS-Port attribute, then the request MUST be silently discarded, and a response MUST NOT be sent back to the NAS. Access-Boot packets MAY NOT be sent from being a NAT gateway.

If the client is unknown (contents of Calling-Station-ID or NAS-Identifier are unknown), then the request MUST be silently discarded, and a response MUST NOT be sent back to the NAS. Request from unknown clients may be attacks, and must not compromise the server.

If the Access-Boot packet is badly formed, or does not contain a Message-Authenticator attribute, or the Message-Authenticator attribute fails validation, then the request MUST be silently discarded, and a response MUST NOT be sent back to the NAS.

The contents of the Called-Station-Id attribute SHOULD be an identifier of the RADIUS server to which the client is attempting to register. This attribute helps verify that the request is being

sent to the correct RADIUS server. The RADIUS server MAY ignore the contents of this attribute.

The Event-Timestamp attribute should be used to help prevent replays. We may want a separate "boot number" attribute, instead...

The Access-Boot packet MUST also contain a State attribute. The State attribute is sent by the client to the server, and is used by the client to associate Access-Booted packets with Access-Boot requests. If the server replies with an Access-Booted packet, then the State attribute MUST be copied unmodified from the Access-Boot packet to the Access-Booted packet. The client SHOULD create a State attribute at least 16 octets in length, using a CSPRNG.

6.2 RADIUS Server Response

If the RADIUS server validates the Access-Boot request, then it MUST add the IP (or IPv6) address of the client, from the NAS-IP-Address (or NAS-IPv6-Address) attribute to its list of addresses for allowed RADIUS clients. From this point on, the RADIUS server may treat the RADIUS client in the same manner as a client with a static IP, configured on the RADIUS server.

The RADIUS server responds to the client with an Access-Booted packet. This packet MUST echo back to the client the State attribute from the Access-Boot request. The server MAY also include a Session-Timeout attribute. After this timeout, the client MUST re-authenticate itself to the server, and the server MUST NOT process any more RADIUS requests from the client. The client SHOULD re-authenticate itself prior to this timeout, to ensure that it does not lose access to the RADIUS server, while it has outstanding Access-Requests to that server.

The Access-Booted packet SHOULD also contain the RADIUS Client Session Shared Secret. This Secret will be at least 16 octets in length, using a CSPRNG. It will be used as the regular RADIUS Client secret for this session. The secret will be transmitted in a new encrypted-data attribute.

The Access-Booted packet MUST contain a Message-Authenticator attribute, to sign the packet contents. The Access-Booted packet also SHOULD contain a "boot number" attribute, which is echoed from the Access-Boot packet.

6.3 Encrypted-Data Attribute

The Encrypted-Data Attribute is an AES Key-wrap Attribute containing the encrypted data, in this case just the RADIUS Client Session Secret encrypted by the Master Secret:

```
0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |     Type      |    Length     |             Reserved          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          Authenticator                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          Authenticator                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          Authenticator                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Client Session Secret                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Client Session Secret                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Client Session Secret                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Client Session Secret                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Author's Note: Need AES Key Wrap format here.

The Authenticator is LTRUNC-96(HMAC-SHA1(Master-Secret,(encrypted-data))).

6.4 Table of Attributes

The following table provides a guide to which attributes may be found in which kinds of packets, and in what quantity.

Boot	Booted	#	Attribute
0-1	0	4	NAS-IP-Address [Note 1]
1	0	5	NAS-Port
1	1	24	State [Note 2]
0	0-1	27	Session-Timeout
0-1	0	30	Called-Station-Id
1	0	31	Calling-Station-Id
0-1	0	32	NAS-Identifier
1	1	55	Event-Timestamp
1	1	80	Message-Authenticator
0-1	0	96	NAS-IPv6-Address [Note 1]
0	0-1	TBD	Encrypted-Data

Note 1: An Access-Boot packet MUST contain either a NAS-IP-Address or a NAS-IPv6-Address attribute. An Access-Boot packet MUST NOT contain both attributes.

Note 2: The client sends The State attribute in an Access-Boot packet to the server. If the server replies with an Access-Booted packet, then the State attribute MUST be copied unmodified from the Access-Boot packet to the Access-Booted packet.

The following table defines the meaning of the above table entries.

0	This attribute MUST NOT be present in packet.
0+	Zero or more instances of this attribute MAY be present in packet.
0-1	Zero or one instance of this attribute MAY be present in packet.
1	Exactly one instance of this attribute MUST be present in packet.

IANA Considerations

Allocate RADIUS codes for Access-Boot and Access-Booted. Allocate RADIUS Attribute types for Encrypted-Data.

Security Considerations

This protocol uses an un-authenticated Diffie-Hellman exchange. This is open to a Man-in-the-Middle attack. This requires either the operator of the RADIUS server to know that there is no possibility for a system between the RADIUS server and the client (e.g. operator can see the cross-over cable between the two devices), or the operator validates the fingerprint of the Client's public Diffie-Hellman value, as discussed in [5]. If the server operator detects a middleman, it can back off of the exchange.

There is a potential replay DOS attack against the Client Session Secret boot protocol. The inclusion of a Boot Number in the Client-Boot and Client-Booted attributes effectively blocks a replayed Access-Boot packet.

Acknowledgments

This document is the result of discussions at IEEE 802.11i and 802.1 meetings. John Vollbrecht was of invaluable assistance in focusing the problem statement and the solution methodology. At the October 2002 meeting of IEEE 802.1, a straw vote passed unanimously to back addressing the RADIUS Client deployment problem as covered here.

Author's Addresses

Robert Moskowitz
TruSecure/ICSAlabs
1000 Bent Creek Blvd, Suite 200
Mechanicsburg, PA
Email: rgm@trusecure.com

Alan DeKok
Email: aland@ox.org

References

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

2 Rigney, C., etal, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

3 IEEE Std 802.1X-2001, "Port-Based Network Access Control", June 2001.

4 Moskowitz, R., "The Shared Secret Provisioning Protocol", Internet Draft draft-moskowitz-shared-secret-provprotocol-01.txt, November 2003.

5 Moskowitz, R., "SSPP over SNMP", Internet Draft draft-moskowitz-sspp-snmp-01.txt, November 2003.

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

7 IEEE DRAFT Std 802.11i/D7, "Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications -- Specification for Enhanced Security", October 2003.

8 Kivinen, T., and Kopo, M., "More MODP Diffie-Hellman groups for IKE", Internet Draft draft-ietf-ipsec-ike-modp-groups-05.txt, January 2003.