Hogeschool van Amsterdam
Amsterdam University of Applied Sciences

Project From Moonshot to Mars

Technical Document FreeRadius Module

*Remy Bien, Ruben Bras, Marvin Hiemstra,
Sebastiaan Groot, Wouter Miltenburg, Koen Veelenturf*

29 May 2013,
Version 0.1

NIKHEF

Table of contents

Introduction

This Technical document is written in order to elaborate on the technical aspects of the FreeRadius Module.

This document only specifies the FreeRadius Module which makes other technical aspects like the port to CentOS from JaNET fall out of scope for this document.

In chapter “Protocol Specifications” the used protocols, algorithms, message formats and message sequence will be described. Chapter Written Modules will describe the use of SMIME and the management and builds of the certificates.

Protocol Specification

This chapter describes the message sequence, used message formats and used algorithms.

Environment

Client
The “user with end system” who wants to obtain extra services from the IdP. It sends data to the Radius proxy which forwards it to the IdP.

Radius proxy
The “middleman” who obtains data from the client and forwards it to the IdP and vice versa. The Radius proxy is able to “inject” additional information when requested by the IdP to the data which have to be forwarded.

IdP
The IdentityProvider or “end point” which receives authentication requests from the client encrypted and forwarded by the Radius proxy. It validates and approves the request from the client and request if necessary additional information from the Radius proxy to be sends to the client.

Message formats

Client to IdP request:
The RADIUS client adds an inner-tunnel AVP to his Access-Request indicating that an additional attribute is required. The value segment of the AVP will have the following format:

```
#Mime-Version: 1.0
#Content-Type: text/plain
#Content-Transfer-Encoding: base64
#
#[Base64]VOMS attributes request[/BASE64]
```

Radius proxy to IdP certificate:
Radius proxy server using this module will add his certificate to the outer tunnel. The value segment of the AVP will have the following format:

```
#Mime-Version: 1.0
#Content-Type: application/pkcs7-mime; smime-type=certs-only
#Content-Transfer-Encoding: base64
#
#[PKCS#7-format Certificate Chain here]
```

IdP to Radius proxy attribute request:
The IdP server will sign using his own private key and encrypt this request using the Radius Proxy servers' certificate. The value segment of the AVP will have the following format:

```
#Mime-Version: 1.0
#Content-Type: application/pkcs7-mime; smime-type: enveloped-data
#Content-Transfer-Encoding: base64
#
#[Base64 and encrypted data]
#Content-Type: multipart/signed; protocol="application/pkcs7-signature"; micalg=sha1;
#boundary=boundary42
#
#--boundary42
#Content-Type: text/plain
#
#MessageType=VOMS-GroupInfo-Request
#Attribute-Username=<username>
#
#--boundary42
#Content-Type: application/pkcs7-signature
#Content-Transfer-Encoding: base64
#
#[Base64-encoded p7s containing pkcs#7 certificate and sha-1 hash of message above]
#
#--boundary42--
#[/Base64 and encrypted data]
```

Radius proxy to Client attribute information:
Upon receiving a valid request for extra information in an Access-Accept message, the AAA-proxy server will insert additional attributes to the Access-Accept messages' outer-tunnel. The value segment of the AVP will have the following format:

```
#Mime-Version: 1.0
#Content-Type: application/pkcs7-mime; smime-type: enveloped-data
#Content-Transfer-Encoding: base64
#
#[Base64 and encrypted data]
#Content-Type: multipart/signed; protocol="application/pkcs7-signature"; micalg=sha1;
#boundary=boundary42
#
#--boundary42
#Content-Type: text/plain
#
#MessageType=VOMS-GroupInfo
#AttributeGroupInfo=<GroupInfo>
#
#--boundary42
#Content-Type: application/pkcs7-signature
#Content-Transfer-Encoding: base64
#
#[Base64-encoded p7s containing pkcs#7 certificate and sha-1 hash of message above]
#
#--boundary42--
```

#[/Base64 and encrypted data]

Carrier Protocol

The MIME entities will be wrapped in a vendor-specific AVP in the RADIUS Access-Request of Access-Accept packets. The vendor-specific AVPs are added according to RFC2865.

Supported Algorithms

Encryption

The IdP will send messages encrypted with RSA to the Radius proxy. The Radius proxy will respond by signing the message with RSA and send the message to the client.

Hashing

The Radius proxy will hash the message received from the IdP with RSA and send the message encrypted to the client.

Encoding

The mime body’s will be encoded Base64, the headers will be constructed with US ASCI.

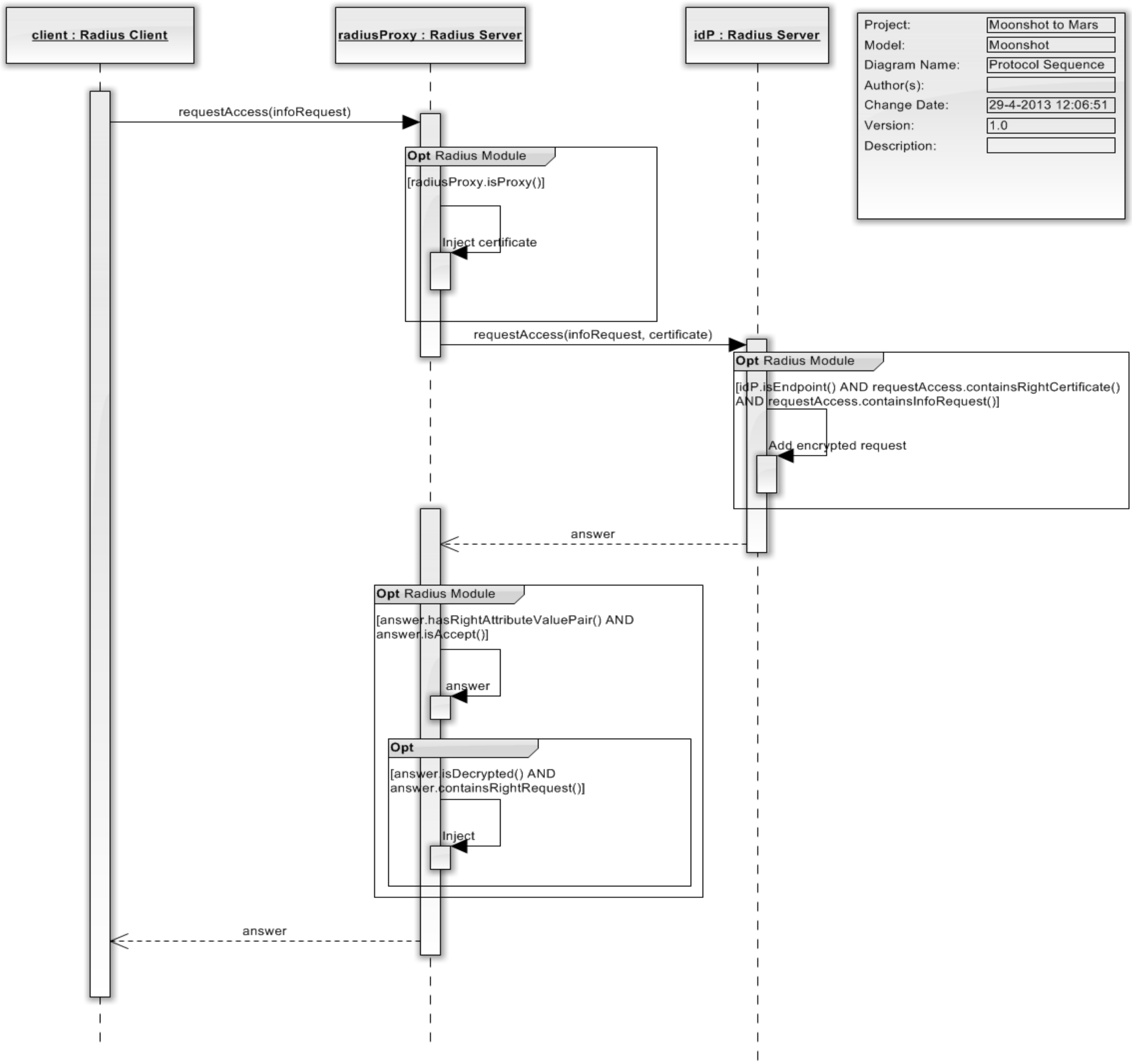
Certificate format

For the certificate format, the PKCS#7 format will be used according to the SMIME standard.

Communication specification

A rough outline of the communication specification of the protocol

Sequence Diagram


client : Radius Client
radiusProxy : Radius Server
idP : Radius Server
requestAccess(infoRequest)
Opt Radius Module
[radiusProxy.isProxy()]
Inject certificate
requestAccess(infoRequest, certificate)
Opt Radius Module
[idP.isEndpoint() AND requestAccess.containsRightCertificate() AND requestAccess.containsInfoRequest()]
Add encrypted request
answer
Opt Radius Module
[answer.hasRightAttributeValuePair() AND answer.isAccept()]
answer
Opt
[answer.isDecrypted() AND answer.containsRightRequest()]
Inject
answer

Project:	Moonshot to Mars
Model:	Moonshot
Diagram Name:	Protocol Sequence
Author(s):	
Change Date:	29-4-2013 12:06:51
Version:	1.0
Description:	

Figuur 1: Sequence

Data processing

The described sequence will run from Client to Radius Proxy to IdP and back. The Client wants to connect to the IdP, and this will happen via the Radius Proxy. The Client will send an AccessRequest, coupled with a request for additional information inside the inner-tunnel. The Radius proxy will inject its certificate in the AccessRequest's outer-tunnel. The IdP will perform a check to see if the received certificate is legit, and if the inner-tunnel contains a request for additional information. If this is the case, an encrypted request will be added to the regular answer as an Attribute Value Pair back to the Radius proxy. The Radius proxy will in turn perform a check to see if this is the right Attribute Value Pair, and if the IdP accepted the credentials (if the answer is Reject or Challenge, the sequence will continue without any more intervention from our Radius Module, until the returned answer is Accept). If this is the case, the Attribute Value Pair will be decrypted, and the requested attributes injected in the answer. This answer, plus the encrypted requested attributes, are then sent back to the Client.

Written modules

SMIME

Certificate Management