pyrad Documentation

Release 2.1

Wichert Akkerman

February 02, 2017

Contents

1 Introduction **3**

2 Requirements & Installation **5**

3 API Documentation **7**

3.1 `pyrad.client` – basic client . . . 7

3.2 `pyrad.dictionary` – RADIUS dictionary . . . 8

3.3 `pyrad.host` – RADIUS host definition . . . 9

3.4 `pyrad.packet` – packet encoding and decoding . . . 10

3.5 `pyrad.proxy` – basic proxy . . . 13

3.6 `pyrad.server` – basic server . . . 13

4 Indices and tables **15**

Python Module Index **17**

Author Wichert Akkerman

Version 2.1

CHAPTER 1

Introduction

pyrad is an implementation of a RADIUS client/server as described in RFC2865. It takes care of all the details like building RADIUS packets, sending them and decoding responses.

Here is an example of doing a authentication request:

```
from __future__ import print_function
from pyrad.client import Client
from pyrad.dictionary import Dictionary
import pyrad.packet

srv = Client(server="localhost", secret=b"Kah3choteereethiejeimaeziecumi",
             dict=Dictionary("dictionary"))

# create request
req = srv.CreateAuthPacket(code=pyrad.packet.AccessRequest,
                           User_Name="wichert", NAS_Identifier="localhost")
req["User-Password"] = req.PwCrypt("password")

# send request
reply = srv.SendPacket(req)

if reply.code == pyrad.packet.AccessAccept:
    print("access accepted")
else:
    print("access denied")

print("Attributes returned by server:")
for i in reply.keys():
    print("%s: %s" % (i, reply[i]))
```

CHAPTER 2

Requirements & Installation

pyrad requires Python 2.6 or later, or Python 3.2 or later

Installing is simple; pyrad uses the standard distutils system for installing Python modules:

```
python setup.py install
```

CHAPTER 3

API Documentation

Per-module `pyrad` API documentation.

3.1 `pyrad.client` – basic client

class `pyrad.client.`**`Timeout`**

Simple exception class which is raised when a timeout occurs while waiting for a RADIUS server to respond.

class `pyrad.client.`**`Client`**(*server*, *authport=1812*, *acctport=1813*, *coaport=3799*, *secret=''*, *dict=None*)

Basic RADIUS client. This class implements a basic RADIUS client. It can send requests to a RADIUS server, taking care of timeouts and retries, and validate its replies.

Variables

- **`retries`** – number of times to retry sending a RADIUS request
- **`timeout`** – number of seconds to wait for an answer

`CreateAcctPacket`(***args*)

Create a new RADIUS packet. This utility function creates a new RADIUS packet which can be used to communicate with the RADIUS server this client talks to. This is initializing the new packet with the dictionary and secret used for the client.

Returns a new empty packet instance

Return type *pyrad.packet.Packet*

`CreateAuthPacket`(***args*)

Create a new RADIUS packet. This utility function creates a new RADIUS packet which can be used to communicate with the RADIUS server this client talks to. This is initializing the new packet with the dictionary and secret used for the client.

Returns a new empty packet instance

Return type *pyrad.packet.Packet*

`CreateCoAPacket`(***args*)

Create a new RADIUS packet. This utility function creates a new RADIUS packet which can be used to communicate with the RADIUS server this client talks to. This is initializing the new packet with the dictionary and secret used for the client.

Returns a new empty packet instance

Return type *pyrad.packet.Packet*

SendPacket(*pkt*)
Send a packet to a RADIUS server.

Parameters **pkt** (pyrad.packet.Packet) – the packet to send

Returns the reply packet received

Return type *pyrad.packet.Packet*

Raises **Timeout** – RADIUS server does not reply

bind(*addr*)
Bind socket to an address. Binding the socket used for communicating to an address can be usefull when working on a machine with multiple addresses.

Parameters **addr** (*host,port tuple*) – network address (hostname or IP) and port to bind to

3.2 pyrad.dictionary – RADIUS dictionary

RADIUS uses dictionaries to define the attributes that can be used in packets. The Dictionary class stores the attribute definitions from one or more dictionary files.

Dictionary files are textfiles with one command per line. Comments are specified by starting with a # character, and empty lines are ignored.

The commands supported are:

```
ATTRIBUTE <attribute> <code> <type> [<vendor>]
specify an attribute and its type

VALUE <attribute> <valuename> <value>
specify a value attribute

VENDOR <name> <id>
specify a vendor ID

BEGIN-VENDOR <vendorname>
begin definition of vendor attributes

END-VENDOR <vendorname>
end definition of vendor attributes
```

The datatypes currently supported are:

type	description
string	ASCII string
ipaddr	IPv4 address
date	32 bits UNIX
octets	arbitrary binary data
abinary	ascend binary data
ipv6addr	16 octets in network byte order
ipv6prefix	18 octets in network byte order
integer	32 bits unsigned number
signed	32 bits signed number
short	16 bits unsigned number
byte	8 bits unsigned number

These datatypes are parsed but not supported:

type	description
ifid	8 octets in network byte order
ether	6 octets of hh:hh:hh:hh:hh:hh where 'h' is hex digits, upper or lowercase.

class `pyrad.dictionary.`**`ParseError`**(*msg=None*, ***data*)

Dictionary parser exceptions.

Variables

- **`msg`** – Error message
- **`linenumber`** – Line number on which the error occured

class `pyrad.dictionary.`**`Dictionary`**(*dict=None*, **dicts*)

RADIUS dictionary class. This class stores all information about vendors, attributes and their values as defined in RADIUS dictionary files.

Variables

- **`vendors`** – bidict mapping vendor name to vendor code
- **`attrindex`** – bidict mapping
- **`attributes`** – bidict mapping attribute name to attribute class

`ReadDictionary`(*file*)

Parse a dictionary file. Reads a RADIUS dictionary file and merges its contents into the class instance.

Parameters **`file`** (*`string or file-like object`*) – Name of dictionary file to parse or a file-like object

3.3 `pyrad.host` – RADIUS host definition

class `pyrad.host.`**`Host`**(*authport=1812*, *acctport=1813*, *coaport=3799*, *dict=None*)

Generic RADIUS capable host.

Variables

- **`dict`** – RADIUS dictionary
- **`authport`** – port to listen on for authentication packets
- **`acctport`** – port to listen on for accounting packets

`CreateAcctPacket`(***args*)

Create a new accounting RADIUS packet. This utility function creates a new accouting RADIUS packet which can be used to communicate with the RADIUS server this client talks to. This is initializing the new packet with the dictionary and secret used for the client.

Returns a new empty packet instance

Return type *pyrad.packet.AcctPacket*

`CreateAuthPacket`(***args*)

Create a new authentication RADIUS packet. This utility function creates a new RADIUS authentication packet which can be used to communicate with the RADIUS server this client talks to. This is initializing the new packet with the dictionary and secret used for the client.

Returns a new empty packet instance

Return type *pyrad.packet.AuthPacket*

CreateCoAPacket(***args*)

Create a new CoA RADIUS packet. This utility function creates a new CoA RADIUS packet which can be used to communicate with the RADIUS server this client talks to. This is initializing the new packet with the dictionary and secret used for the client.

Returns a new empty packet instance

Return type *pyrad.packet.CoAPacket*

CreatePacket(***args*)

Create a new RADIUS packet. This utility function creates a new RADIUS authentication packet which can be used to communicate with the RADIUS server this client talks to. This is initializing the new packet with the dictionary and secret used for the client.

Returns a new empty packet instance

Return type *pyrad.packet.Packet*

SendPacket(*fd*, *pkt*)

Send a packet.

Parameters

- **fd** (*socket class instance*) – socket to send packet with
- **pkt** (*Packet class instance*) – packet to send

SendReplyPacket(*fd*, *pkt*)

Send a packet.

Parameters

- **fd** (*socket class instance*) – socket to send packet with
- **pkt** (*Packet class instance*) – packet to send

3.4 pyrad.packet – packet encoding and decoding

class pyrad.packet.**Packet**(*code=0*, *id=None*, *secret=''*, *authenticator=None*, ***attributes*)

Packet acts like a standard python map to provide simple access to the RADIUS attributes. Since RADIUS allows for repeated attributes the value will always be a sequence. pyrad makes sure to preserve the ordering when encoding and decoding packets.

There are two ways to use the map intereface: if attribute names are used pyrad take care of en-/decoding data. If the attribute type number (or a vendor ID/attribute type tuple for vendor attributes) is used you work with the raw data.

Normally you will not use this class directly, but one of the *AuthPacket* or *AcctPacket* classes.

AddAttribute(*key*, *value*)

Add an attribute to the packet.

Parameters

- **key** (*string, attribute code or (vendor code, attribute code) tuple*) – attribute name or identification
- **value** (*depends on type of attribute*) – value

static CreateAuthenticator()

Create a packet autenticator. All RADIUS packets contain a sixteen byte authenticator which is used to

authenticate replies from the RADIUS server and in the password hiding algorithm. This function returns a suitable random string that can be used as an authenticator.

Returns valid packet authenticator

Return type binary string

CreateID()

Create a packet ID. All RADIUS requests have a ID which is used to identify a request. This is used to detect retries and replay attacks. This function returns a suitable random number that can be used as ID.

Returns ID number

Return type integer

CreateReply(***attributes*)

Create a new packet as a reply to this one. This method makes sure the authenticator and secret are copied over to the new instance.

DecodePacket(*packet*)

Initialize the object from raw packet data. Decode a packet as received from the network and decode it.

Parameters **packet** (*string*) – raw packet

ReplyPacket()

Create a ready-to-transmit authentication reply packet. Returns a RADIUS packet which can be directly transmitted to a RADIUS server. This differs with Packet() in how the authenticator is calculated.

Returns raw packet

Return type string

SaltCrypt(*value*)

Salt Encryption

Parameters **value** – plaintext value

Returns obfuscated version of the value

Return type binary string

class pyrad.packet.**AuthPacket**(*code=1*, *id=None*, *secret=''*, *authenticator=None*, ***attributes*)

CreateReply(***attributes*)

Create a new packet as a reply to this one. This method makes sure the authenticator and secret are copied over to the new instance.

PwCrypt(*password*)

Obfuscate password. RADIUS hides passwords in packets by using an algorithm based on the MD5 hash of the packet authenticator and RADIUS secret. If no authenticator has been set before calling PwCrypt one is created automatically. Changing the authenticator after setting a password that has been encrypted using this function will not work.

Parameters **password** (*unicode stringn*) – plaintext password

Returns obfuscated version of the password

Return type binary string

PwDecrypt(*password*)

Unobfuscate a RADIUS password. RADIUS hides passwords in packets by using an algorithm based on the MD5 hash of the packet authenticator and RADIUS secret. This function reverses the obfuscation process.

Parameters **password** (*binary string*) – obfuscated form of password

Returns plaintext password

Return type unicode string

RequestPacket()

Create a ready-to-transmit authentication request packet. Return a RADIUS packet which can be directly transmitted to a RADIUS server.

Returns raw packet

Return type string

VerifyChapPasswd(*userpwd*)

Verify RADIUS ChapPasswd

Parameters **userpwd** (*str*) – plaintext password

Returns is verify ok

Return type bool

class pyrad.packet.**AcctPacket**(*code=4*, *id=None*, *secret=''*, *authenticator=None*, ***attributes*)

RADIUS accounting packets. This class is a specialization of the generic *Packet* class for accounting packets.

CreateReply(***attributes*)

Create a new packet as a reply to this one. This method makes sure the authenticator and secret are copied over to the new instance.

RequestPacket()

Create a ready-to-transmit authentication request packet. Return a RADIUS packet which can be directly transmitted to a RADIUS server.

Returns raw packet

Return type string

VerifyAcctRequest()

Verify request authenticator.

Returns True if verification failed else False

Return type boolean

class pyrad.packet.**CoAPacket**(*code=43*, *id=None*, *secret=''*, *authenticator=None*, ***attributes*)

RADIUS CoA packets. This class is a specialization of the generic *Packet* class for CoA packets.

CreateReply(***attributes*)

Create a new packet as a reply to this one. This method makes sure the authenticator and secret are copied over to the new instance.

RequestPacket()

Create a ready-to-transmit CoA request packet. Return a RADIUS packet which can be directly transmitted to a RADIUS server.

Returns raw packet

Return type string

VerifyCoARequest()

Verify request authenticator.

Returns True if verification failed else False

Return type boolean

class `pyrad.packet.`**`PacketError`**

3.4.1 Constants

The *`pyrad.packet`* module defines several common constants that are useful when dealing with RADIUS packets.

The following packet codes are defined:

Constant name	Value
AccessRequest	1
AccessAccept	2
AccessReject	3
AccountingRequest	4
AccountingResponse	5
AccessChallenge	11
StatusServer	12
StatusClient	13
DisconnectRequest	40
DisconnectACK	41
DisconnectNAK	42
CoARequest	43
CoAACK	44
CoANAK	45

3.5 `pyrad.proxy` – basic proxy

class `pyrad.proxy.`**`Proxy`**(*addresses=[], authport=1812, acctport=1813, coaport=3799, hosts=None, dict=None, auth_enabled=True, acct_enabled=True, coa_enabled=False*)

Base class for RADIUS proxies. This class extends tha RADIUS server class with the capability to handle communication with other RADIUS servers as well.

Variables **`_proxyfd`** – network socket used to communicate with other servers

3.6 `pyrad.server` – basic server

class `pyrad.server.`**`RemoteHost`**(*address, secret, name, authport=1812, acctport=1813, coaport=3799*)

Remote RADIUS capable host we can talk to.

class `pyrad.server.`**`ServerPacketError`**

Exception class for bogus packets. ServerPacketError exceptions are only used inside the Server class to abort processing of a packet.

class `pyrad.server.`**`Server`**(*addresses=[], authport=1812, acctport=1813, coaport=3799, hosts=None, dict=None, auth_enabled=True, acct_enabled=True, coa_enabled=False*)

Basic RADIUS server. This class implements the basics of a RADIUS server. It takes care of the details of receiving and decoding requests; processing of the requests should be done by overloading the appropriate methods in derived classes.

Variables

- **`hosts`** – hosts who are allowed to talk to us

- **_poll** – poll object for network sockets
- **_fdmap** – map of filedescriptors to network sockets
- **MaxPacketSize** – maximum size of a RADIUS packet

BindToAddress(*addr*)

Add an address to listen to. An empty string indicated you want to listen on all addresses.

Parameters **addr** (*string*) – IP address to listen on

CreateReplyPacket(*pkt*, ***attributes*)

Create a reply packet. Create a new packet which can be returned as a reply to a received packet.

Parameters **pkt** (*Packet instance*) – original packet

HandleAcctPacket(*pkt*)

Accounting packet handler. This is an empty function that is called when a valid accounting packet has been received. It can be overriden in derived classes to add custom behaviour.

Parameters **pkt** (*Packet class instance*) – packet to process

HandleAuthPacket(*pkt*)

Authentication packet handler. This is an empty function that is called when a valid authentication packet has been received. It can be overriden in derived classes to add custom behaviour.

Parameters **pkt** (*Packet class instance*) – packet to process

HandleCoaPacket(*pkt*)

CoA packet handler. This is an empty function that is called when a valid accounting packet has been received. It can be overriden in derived classes to add custom behaviour.

Parameters **pkt** (*Packet class instance*) – packet to process

HandleDisconnectPacket(*pkt*)

CoA packet handler. This is an empty function that is called when a valid accounting packet has been received. It can be overriden in derived classes to add custom behaviour.

Parameters **pkt** (*Packet class instance*) – packet to process

Run()

Main loop. This method is the main loop for a RADIUS server. It waits for packets to arrive via the network and calls other methods to process them.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

p

pyrad.client, 7
pyrad.dictionary, 8
pyrad.host, 9
pyrad.packet, 10
pyrad.proxy, 13
pyrad.server, 13

A

AcctPacket (class in pyrad.packet), 12
AddAttribute() (pyrad.packet.Packet method), 10
AuthPacket (class in pyrad.packet), 11

B

bind() (pyrad.client.Client method), 8
BindToAddress() (pyrad.server.Server method), 14

C

Client (class in pyrad.client), 7
CoAPacket (class in pyrad.packet), 12
CreateAcctPacket() (pyrad.client.Client method), 7
CreateAcctPacket() (pyrad.host.Host method), 9
CreateAuthenticator() (pyrad.packet.Packet static method), 10
CreateAuthPacket() (pyrad.client.Client method), 7
CreateAuthPacket() (pyrad.host.Host method), 9
CreateCoAPacket() (pyrad.client.Client method), 7
CreateCoAPacket() (pyrad.host.Host method), 9
CreateID() (pyrad.packet.Packet method), 11
CreatePacket() (pyrad.host.Host method), 10
CreateReply() (pyrad.packet.AcctPacket method), 12
CreateReply() (pyrad.packet.AuthPacket method), 11
CreateReply() (pyrad.packet.CoAPacket method), 12
CreateReply() (pyrad.packet.Packet method), 11
CreateReplyPacket() (pyrad.server.Server method), 14

D

DecodePacket() (pyrad.packet.Packet method), 11
Dictionary (class in pyrad.dictionary), 9

H

HandleAcctPacket() (pyrad.server.Server method), 14
HandleAuthPacket() (pyrad.server.Server method), 14
HandleCoaPacket() (pyrad.server.Server method), 14
HandleDisconnectPacket() (pyrad.server.Server method), 14
Host (class in pyrad.host), 9

P

Packet (class in pyrad.packet), 10
PacketError (class in pyrad.packet), 13
ParseError (class in pyrad.dictionary), 9
Proxy (class in pyrad.proxy), 13
PwCrypt() (pyrad.packet.AuthPacket method), 11
PwDecrypt() (pyrad.packet.AuthPacket method), 11
pyrad.client (module), 7
pyrad.dictionary (module), 8
pyrad.host (module), 9
pyrad.packet (module), 10
pyrad.proxy (module), 13
pyrad.server (module), 13

R

ReadDictionary() (pyrad.dictionary.Dictionary method), 9
RemoteHost (class in pyrad.server), 13
ReplyPacket() (pyrad.packet.Packet method), 11
RequestPacket() (pyrad.packet.AcctPacket method), 12
RequestPacket() (pyrad.packet.AuthPacket method), 12
RequestPacket() (pyrad.packet.CoAPacket method), 12
Run() (pyrad.server.Server method), 14

S

SaltCrypt() (pyrad.packet.Packet method), 11
SendPacket() (pyrad.client.Client method), 7
SendPacket() (pyrad.host.Host method), 10
SendReplyPacket() (pyrad.host.Host method), 10
Server (class in pyrad.server), 13
ServerPacketError (class in pyrad.server), 13

T

Timeout (class in pyrad.client), 7

V

VerifyAcctRequest() (pyrad.packet.AcctPacket method), 12
VerifyChapPasswd() (pyrad.packet.AuthPacket method), 12

VerifyCoARequest() (pyrad.packet.CoAPacket method), 12