vmam

Release 1.4.4

Apr 05, 2021

Contents:

vmam is a Unix server-side application, which allows you to manage LDAP users representing the physical cards present on a given network. Network cards are represented and interpreted by *vmam* as LDAP user, with specific attributes. *vmam* also manages computer accounts, in case your radius policies provide for authentication from computer accounts. All this is possible thanks to a simple configuration file.

vmam can be used in two ways. One, using its command line CLI, or as a python module. Using it as a python module, you can create your own environment without using the configuration file. See the next section for more information.

CHAPTER 1

RFC

vmam is based on various RFC. See this: IEEE 802.1X, RFC 3580, RFC 4014, RFC 2865, RFC 3579.

1.1 Prerequisites

1.1.1 How LDAP RADIUS MAC Authentication Works

1. MAC authentication is initiated based on the security settings configured for the switch or WiFi.
2. Authenticates the MAC address of the connecting client with a RADIUS server which in turn authenticates itself with a configured LDAP server.
3. If the MAC authentication is successful, the client device is allowed to access the VLAN.
4. If the MAC authentication fails, you can configure the switch or WiFi to take one of these actions:
    - Connect the client even though it not authorized. You can optionally assign a role to the client from your defined role profiles. This role can assign the client to a specific VLAN ID or have other restrictions based on the role configuration. You can also redirect the user to web site or portal that provides information about why access was denied or displays instructions for self-registration.
    - Disconnect the client device because it is not authorized.

1.1.2 vmam Server

The server hosting *vmam* must be unix-like and with systemd installed.

Python

vmam is written in python3 (3.3 and higher). It also uses four third-party python libraries, necessary for the correct functioning of the tool:

- pywinrm

- ldap3
- daemon
- pyyaml

1.1.3 Client on the network

If you use an automatic mode, the clients on the network managed by *vmam* must be Windows machines, with WINRM enabled. To enable it, run `winrm quickconfig`. On the other hand, if you use the manual process, the clients can be anything (linux, MacOSX, BSD, printers, router, etc.)

1.1.4 Directory Server

vmam allows the management of mac-addresses thanks to operations on a directory server through the LDAP protocol. The directory server (Active Directory or FreeIPA) must be installed before configuring *vmam*.

LDAP Protocol

Through the LDAP Protocol, *vmam* creates, searches, deletes, disables and authenticates all the mac-addresses to manage.

LDAP Users

vmam, creates mac-addresses in the directory server that represent the physical cards of the machines that access the network. These mac-addresses are for all intents and purposes of LDAP users.

LDAP Computers

vmam uses computer accounts linked to the domain for remote contact (via WINRM), in order to take the necessary information to understand which mac-address is used by which user.

LDAP Groups

vmam uses LDAP security groups to directly represent VLAN-IDs. These groups will be configured in the radius server policies. Then create the LDAP groups based on the VLAN ids you need to manage.

LDAP Organizational Unit

vmam uses organizational units (OU) as a search base for the three types of LDAP objects: users, computers and groups. In the *vmam* configuration file, you will find three LDAP object search bases. Nobody forbids all three to coincide, but it's best to keep them separate in different OUs for proper functionality.

LDAP Group Configuration

This is an example of an LDAP group creation that represents a VLAN-ID on FreeIPA server:

```
$> ipa group-add vlan_100 --desc="VLAN group corrisponding to VLAN 100" --nonposix

----------------------
Added group "vlan_100"
----------------------
  Group name: vlan_100
  Description: VLAN group corrisponding to VLAN 100
  GID: 855800010
```

Note: To create LDAP groups on other LDAP servers, search for the documentation in your LDAP server. For Active Directory follow the link here.

1.1.5 Radius Server

To accept the authentication of the various mac-addresses and *"release"* a VLAN, a Radius Server is required. If you have an Active Directory server, it is better to install NPS. Otherwise you can choose to install Free Radius. Below is an example of a Radius configuration with LDAP authentication.

Radius Configuration

Radius has its own database of users, anyway, since this information is already contained in LDAP, it will be more convenient to use it! Once you have installed the server you have to configure it using the configuration files, that are located under `/etc/raddb` In the `radiusd.conf` file edit :

```
[...omissis]
# Uncomment this if you want to use ldap (Auth-Type = LDAP)
# Also uncomment it in the authenticate{} block below
        ldap {
                server   = ldap.yourorg.com
                #login    = "cn=admin,o=My Org,c=US"
                #password = mypass
                basedn   = "ou=users,dc=yourorg,dc=com"
                filter   = "(posixAccount)(uid=%u))"
        }

[...omissis]

# Authentication types, Auth-Type = System and PAM for now.
authenticate {
        pam
        unix
#       sql
#       sql2
# Uncomment this if you want to use ldap (Auth-Type = LDAP)
        ldap
}
[...omissis]
```

Also edit the `dictionary` file:

```
[...omissis]
#
#       Non-Protocol Integer Translations
#

VALUE           Auth-Type               Local                   0
VALUE           Auth-Type               System                  1
VALUE           Auth-Type               SecurID                 2
VALUE           Auth-Type               Crypt-Local             3
VALUE           Auth-Type               Reject                  4
VALUE           Auth-Type               ActivCard               4
VALUE           Auth-Type               LDAP                    5
[...omissis]
```

And the `users` file to have a default authorization entry:

```
[...omissis]
DEFAULT         Auth-Type := LDAP
                Fall-Through = 1
[...omissis]
```

On the LDAP server ensure also that the radius server can read the all the posixAccount attributes (expecially `uid` and `userpassword`).

Note: To configure Microsoft Radius, see the following link.

Radius Policy

The radius policies must be configured so that if the mac-address users belongs to a specific LDAP group representing the VLAN-ID, that VLAN is released on the client port.

1.1.6 Network Appliance

Based on your network devices, you will need to configure *"ldap mac-address authentication"* (IEEE 802.1x).

Note: BEST PRACTICE: MAC-based authentication is not as secure as agent access or agentless access authentication. MAC addresses are not generally guarded as secrets, so an attacker can spoof a MAC address and impersonate a device to gain network access. To reduce risk of an exploit, create a special VLAN for each device type.

Configure Network Device

To configure your network devices, you need to follow and search the manuals for the following steps:

1. Create VLANs and configure the VLANs allowed by interfaces so that packets can be forwarded.
2. Create and configure a RADIUS server template, an AAA authentication scheme, and an authentication domain.
3. Enable MAC authentication.
4. Configure the post-authentication domain.

This is an example on Huawei Switch:

```
$> # 1.
$>vlan 100 # users VLAN
$>vlan 200 # guest VLAN
$> # 2.
$>aaa
$>authentication-scheme Test
$>authentication-mode radius
$>authorization-scheme Test
$>authorization-mode if-authenticated
$>accounting-scheme default
$>service-scheme Guest
$>user-vlan 200
$>domain test
$>authentication-scheme Test
$>authorization-scheme Test
$>radius-server Test
$>radius-server template Test
$>radius-server shared-key cipher xxxxxxxxxxxxxxxxxx
$>radius-server authentication 192.168.42.1 1812 weight 81 # radius server
$> # 3.
$>authentication-profile name mac_authen_profile
$>mac-access-profile mac_access_profile
$>authentication timer handshake-period 120
$>authentication mode single-voice-with-data
$>authentication event authen-fail action authorize service-scheme Guest
$>authentication device-type voice authorize
$> # 4.
$>mac-access-profile name mac_access_profile
$>mac-authen reauthenticate
$>authentication trigger-condition dhcp arp
```

1.2 Installation

Now that we have configured everything correctly, we can proceed with the installation of *vmam*. The installation of *vmam* is very simple. With *pip*:

```
pip install vmam
```

Or just run these commands:

```
git clone https://github.com/MatteoGuadrini/vmam.git
cd vmam
sudo python3 setup.py install
```

1.3 Configuration File

1.3.1 Config Mode

To generate an empty configuration file, type at the command line (or through its function, see the *vmam* module):

```
#> vmam config --new <path-to-configuration>.yml
```

If you don't specify the file path, it will create a configuration file in a default path: `/etc/vmam/vmam.yml`

The configuration file is in YAML format.

```
LDAP:                                       # LDAP section
  add_group_type:                           # LDAP objects that will receive the VLAN␣
↪groups (user - computer)
  - user
  - computer
  bind_pwd: password                        # LDAP password of "bind_user"
  bind_user: test\administrator             # LDAP user with write privileges (admin)
  computer_base_dn: OU=PCs,DC=test,DC=local # LDAP base search of computer object
  domain: test.local                        # LDAP domain in dot format
  mac_user_base_dn: OU=mac,DC=test,DC=local # LDAP base search of mac-address user␣
↪object
  match: like                               # Matching operator used for "verify_
↪attrib" (like - match)
  mac_user_ttl: 30d                         # LDAP mac-address user time-to-live
  other_group:                              # Additional LDAP groups, in addition to␣
↪those representing VLAN-IDs
  - all_vlans
  servers:                                  # LDAP servers (ip-address, hostname or␣
↪FQDN)
  - dc1
  - dc2
  ssl: false                                # LDAP ssl connection (TCP port 636)
  time_computer_sync: 1m                    # Computers that have logged on to the␣
↪domain N time ago
  tls: true                                 # LDAP start-tls
  user_base_dn: OU=user,DC=test,DC=bol      # LDAP base search of user object
  verify_attrib:                            # Verification attributes for considering␣
↪a user of a certain VLAN
  - memberof
  write_attrib:                             # Vmam attribute used to write internal␣
↪value
VMAM:                                       # VMAM section
  filter_exclude:                           # Mac-address filters to be excluded
  - TAP
  - disconnect
  log: /tmp/vmam.log                        # Path of vmam log
  remove_process: true                      # Enable vmam remove or disabling process;
↪ disabling depend of "soft_deletion"
  automatic_process_wait: 3                 # Integer represent seconds of wait for␣
↪automatic process
  mac_format: none                          # Mac-address format (none - dot - hypens␣
↪- colon)
  black_list: /etc/vmam/black.list          # File containing blacklisted mac-
↪addresses
  soft_deletion: true                       # If this is true, the mac-addresses are␣
↪disabled and not deleted
  user_match_id:                            # Based on the attribute specified in
↪"verify_attrib". The key is the value to be matched while the value is the VLAN id
    OU=IT: 100
    OU=Sales: 101
    OU=HR: 102
  vlan_group_id:                            # The key is the group VLAN id. The value␣
↪is the name of the LDAP group
    100: it_vlan
```

(continues on next page)

(continued from previous page)

```
    101: sales_vlan
    102: hr_vlan
  winrm_pwd: password                          # WINRM password of "winrm_user"
  winrm_user: test\remoteadmin                 # WINRM user with admin privileges
```

1.3.2 Keys and Values

Below are the key-value references for each reference and section of the configuration file.

LDAP section

This is the LDAP section

Key	Value
add_group_type	"user" or "computer" [list]
bind_user	LDAP user with write privileges [string]
bind_pwd	Password of "bind_user" [string]
computer_base_dn	LDAP base search of computer object [string]
domain	LDAP domain in dot format [string]
mac_user_base_dn	LDAP base search of mac-address user object [string]
match	"like" or "match" [string]
mac_user_ttl	NumberString - Ns - 1s, 2m, 3h, 4d, 5w [string]
other_group	Additional LDAP groups [list]
servers	LDAP Server list [list]
ssl	If "true", protocol is ldaps:// and port is 636 [boolean]
time_computer_sync	NumberString - Ns - 1s, 2m, 3h, 4d, 5w [string]
tls	If "true", starttls (if you have set "ssl", tls will not be considered) [boolean]
user_base_dn	LDAP base search of user object [string]
verify_attrib	Verification attributes for considering a user of a certain VLAN [string]
write_attrib	Vmam attribute used to write internal value (if empty, employeeType is set) [string]

VMAM section

This is the VMAM section

Key	Value
filter_exclude	Mac-address filters to be excluded (See output of command `getmac /fo csv /v`) [list]
log	Path of vmam log [string]
remove_process	Enable vmam remove or disabling process; disabling depend of "soft_deletion" [boolean]
mac_format	"none", "dot", "hypens" or "colon" [string]
black_list	File containing blacklisted mac-addresses. The file can contains mac in any format and comment ("#comment") [string]
automatic_process_wait	Integer represent seconds of wait for automatic process [int]
soft_deletion	If this is "true", the mac-addresses are disabled and not deleted [boolean]
user_match_id	Based on the attribute specified in "verify_attrib". The key is the value to be matched while the value is the VLAN id [dictionary]
vlan_group_id	The key is the group VLAN id. The value is the name of the LDAP group [dictionary]
winrm_user	WINRM user with admin privileges [string]
winrm_pwd	WINRM password of "winrm_user" [string]

Get prerequisites configuration

Once you have compiled the configuration file with your values, to get the prerequisites scheme, just run this command:

```
vmam config --get-cmd <path-to-configuration>.yml
```

If you don't specify the file path, it will create a configuration file in a default path: `/etc/vmam/vmam.yml`

1.4 Command Line

vmam can be run in manual or automatic mode.

```
$> vmam [action] [parameter] [options]
```

1.4.1 Manual Mode

`mac {action}`: Manual action for adding, modifying, deleting and disabling of the mac-address users

In manual mode, you can do these operations:

- Creation
- Disabling
- Deletion

Creation

The process of creating a mac-address user involves these steps:

1. Creation of an LDAP user representing the mac-address
2. Insertion in the VLAN group according to the parameters of the configuration file
3. Insertion in the custom group based on the parameters of the configuration file
4. Check if other VLAN groups are assigned to the user

5. Set password equals as a mac-address

Parameter

`--add/-a {parameter}`: Add a specific mac-address on LDAP with specific VLAN. See also –vlan-id/-i

`--vlan-id/-i {parameter}`: Specify a specific VLAN-id

`--description/-D {parameter}`: Description field

```
$> vmam mac --add <mac-address> --vlan-id <vlan-id> --description <description>
```

Disabling

The disabling process involves only one step; disabling the user.

Parameter

`--disable/-d {parameter}`: Disable a mac-address user on LDAP, without removing

`--force/-f {parameter}`: Force remove/disable action, without prompt confirmation (optional)

`--description/-D {parameter}`: Description field

```
$> vmam mac --disable <mac-address> --description <description>
```

Deletion

The deletion process involves only one step; delete the user.

Parameter

`--remove/-r {parameter}`: Remove a mac-address user on LDAP

`--force/-f {parameter}`: Force remove/disable action, without prompt confirmation (optional)

```
$> vmam mac --remove <mac-address>
```

Common Parameter

`--config-file/-c {parameter}`: Specify a configuration file in a custom path (optional)

Note: If you don't specify the file path, it will create a configuration file in a default path: `/etc/vmam/vmam.yml`

1.4.2 Automatic Mode

`start {action}`: Automatic action for vmam environment

The automatic process can be launched in two ways: *finite* or *system daemon*.

Both have the same process:

1. Check if there are updated computers
2. Connection to the client via WINRM protocol
3. Run the commands: `getmac /FO csv /v` and `quser`
4. Search the last user on LDAP server

5. Check the match of the attributes for the creation of the mac-address
6. Creation of an LDAP user representing the mac-address
7. Insertion in the VLAN group according to the parameters of the configuration file
8. Insertion in the custom group based on the parameters of the configuration file
9. Check if other VLAN groups are assigned to the user
10. Set password equals as a mac-address
11. Assign computer to VLAN groups
12. Add VLAN LDAP group to computer account
13. Add description to computer account
14. Get old mac-address user based on *"mac_user_ttl"*
15. Disable/Remove mac-address based on *"soft_deletion"*

Finite

The process in finished mode, involves above steps, after which, exit with code 0.

```
$> vmam start
```

Daemon

The process in daemon mode, involves the same previous steps, with the only difference that the process is launched in the background as a systemd daemon (see here). If something goes wrong, the process does not exit but writes error lines to the log file and will proceed in its course.

```
$> vmam start --daemon
```

Parameter

`--daemon/-d {parameter}`: If specified, the automatic process run in background

Common Parameter

`--config-file/-c {parameter}`: Specify a configuration file in a custom path (optional)

Note: If you don't specify the file path, it will create a configuration file in a default path: `/etc/vmam/vmam.yml`

Service

You can create a systemd service to make vmam operational by booting the operating system. In addition, this allows you to do all systemd operations (stop, start, restart etc.). Let's create this file `/etc/systemd/system/vmamd.service` with this content:

```
# systemd unit file for the vmam daemon

[Unit]
# Human readable name of the unit
```

(continues on next page)

(continued from previous page)

```
Description=vmam Service

[Service]
# Command to execute when the service is started (add -v for debug)
ExecStart=vmam start -d
# Disable Python's buffering of STDOUT and STDERR, so that output from the
# service shows up immediately in systemd's logs
Environment=PYTHONUNBUFFERED=1
# Automatically restart the service if it crashes
Restart=on-failure

[Install]
# Tell systemd to automatically start this service when the system boots
# (assuming the service is enabled)
WantedBy=default.target
```

Now we can enable and start it:

```
$> systemctl enable vmamd
$> systemctl start vmamd
```

1.4.3 Common Parameter

`--version/-V {parameter}`: Print version and exit

`--verbose/-v {parameter}`: Print and log verbose information, for debugging

1.5 Exit Code

This table contains exit codes and their descriptions:

Exit Code	Description
1	Print help
2	Read/write file error
3	File doesn't exists
4	VLAN-ID doesn't exists
8	Failed check of mac-address existence
9	Set user password error
10	User enable error
11	User disable error
12	Mac-address user removal error
13	Mac-address is blacklisted
16	LDAP group VLAN-ID does not exist
17	LDAP custom group not existing
18	Error removing LDAP VLAN group
32	Error connecting to client

1.6 vmam

1.6.1 vmam module

VLAN Mac-address Authentication Manager

vmam is a command line tool which allows the management and maintenance of the mac-addresses that access the network under a specific domain and a specific VLAN, through LDAP authentication. This is based on RFC-3579(https://tools.ietf.org/html/rfc3579#section-2.1).

Usage for command line:

SYNOPSYS

```
vmam [action] [parameter] [options]

config {action}: Configuration command for vmam environment

    --new/-n {parameter}: Instruction to create a new configuration file. By␣
↪specifying a path, it creates the
    file in the indicated path. The default path is /etc/vmam/vmam.cfg

    $> vmam config --new
    Create a new configuration in a standard path: /etc/vmam/vmam.cfg

    --get-cmd/-g {parameter}: Instruction to obtain the appropriate commands to␣
↪configure your network
    infrastructure and radius server around the created configuration file. By␣
↪specifying a path, get
    the file in the indicated path. The default path is /etc/vmam/vmam.cfg

    $> vmam config --get-cmd
    It takes instructions to configure its own network and radius server structure,
    from standard path: /etc/vmam/vmam.cfg

start {action}: Automatic action for vmam environment

    --config-file/-c {parameter}: Specify a configuration file in a custom path␣
↪(optional)

    $> vmam start --config-file /home/arthur/vmam.cfg
    Start automatic process based on custom path configuration file: /home/arthur/
↪vmam.cfg

    --daemon/-d {parameter}: If specified, the automatic process run in background

    $> vmam start --daemon
    Start automatic process in background based on standard path: /etc/vmam/vmam.cfg

mac {action}: Manual action for adding, modifying, deleting and disabling of the mac-
↪address users

    --add/-a {parameter}: Add a specific mac-address on LDAP with specific VLAN. See␣
↪also --vlan-id/-i

    $> vmam mac --add 000018ff12dd --vlan-id 110
    Add new mac-address user with VLAN 110, based on standard configuration file: /
↪etc/vmam/vmam.cfg
```

(continues on next page)

(continued from previous page)

```
    $> vmam mac --add 000018ff12dd --vlan-id 111
    Modify new or existing mac-address user with VLAN 111, based on standard␣
↪configuration
    file: /etc/vmam/vmam.cfg

    --description/-D {parameter}: Add description on created mac-address

    $> vmam mac --add 000018ff12dd --vlan-id 110 --description "My personal linux"
    Add new mac-address user with VLAN 110, based on standard configuration file: /
↪etc/vmam/vmam.cfg

    --remove/-r {parameter}: Remove a mac-address user on LDAP

    $> vmam mac --remove 000018ff12dd
    Remove mac-address user 000018ff12dd, based on standard configuration file: /etc/
↪vmam/vmam.cfg

    --disable/-d {parameter}: Disable a mac-address user on LDAP, without removing

    $> vmam mac --disable 000018ff12dd
    Disable mac-address user 000018ff12dd, based on standard configuration file: /etc/
↪vmam/vmam.cfg

    --force/-f {parameter}: Force remove/disable action

    $> vmam mac --remove 000018ff12dd --force
    Force remove mac-address user 000018ff12dd, based on standard configuration file:␣
↪/etc/vmam/vmam.cfg

    --vlan-id/-i {parameter}: Specify a specific VLAN-id

    $> vmam mac --add 000018ff12dd --vlan-id 100
    Add new mac-address user with VLAN 100, based on standard configuration file: /
↪etc/vmam/vmam.cfg

    --config-file/-c {parameter}: Specify a configuration file in a custom path␣
↪(optional)

    $> vmam mac --remove 000018ff12dd --config-file /opt/vlan-office/office.cfg
    Remove mac-address user 000018ff12dd, based on custom configuration file: /opt/
↪vlan-office/office.cfg

--version/-V {option}: Print version and exit

--verbose/-v {option}: Print and log verbose information, for debugging
```

Usage like a module:

```
#!/usr/bin/env python3
from vmam import *

# activate debug
debug = True

# define log writer
wt = logwriter('/tmp/log.log')
```

(continues on next page)

(continued from previous page)

```
# start script
debugger(debug, wt, 'Start...')

# connect to LDAP server
conn = connect_ldap(['dc1.foo.bar'])
bind = bind_ldap(conn, r'domain\admin', 'password', tls=True)
ldap_version = check_ldap_version(bind, 'dc=foo,dc=bar')

for mac in get_mac_from_file('/tmp/mac_list.txt'):
    debugger(debug, wt, 'create mac address {}'.format(mac))
    # create mac address
    dn = 'cn={},ou=mac,dc=foo,dc=bar'.format(mac)
    attrs = {'givenname': 'mac-address',
                'sn': mac,
                'samaccountname': mac
            }
    # create mac-address user
    new_user(bind, dn, **attrs)
    # add mac user to vlan group
    add_to_group(bind, 'cn=vlan_group100,ou=groups,dc=foo,dc=bar', dn)
    # set password and password never expires
    set_user(bind, dn, pwdlastset=-1, useraccountcontrol=66048)
    set_user_password(bind, dn, mac, ldap_version=ldap_version)
```

AUTHOR

Matteo Guadrini <matteo.guadrini@hotmail.it>

COPYRIGHT

(c) Matteo Guadrini. All rights reserved.

vmam.**logwriter**(*logfile*)

Logger object than write line in a log file

Parameters **logfile** – Path of logfile(.log)

Returns Logger object

```
>>> wl = logwriter('test.log')
>>> wl.info('This is a test')
```

vmam.**debugger**(*verbose*, *writer*, *message*)

Debugger: write debug and print verbose message

Parameters

- **verbose** – verbose status; boolean
- **writer** – Log writer object
- **message** – String message

Returns String on stdout

```
>>> wl = logwriter('test.log')
>>> debugger(True, wl, 'Test debug')
```

vmam.**confirm**(*message*)

Confirm action

Parameters **message** – Question that expects a 'yes' or 'no' answer

Returns Boolean

```
>>> if confirm('Please, respond'):
...     print('yep!')
```

vmam.**read_config**(*path*)
Open YAML configuration file

Parameters **path** – Path of configuration file

Returns Python object

```
>>> cfg = read_config('/tmp/vmam.yml')
>>> print(cfg)
```

vmam.**get_platform**()
Get a platform (OS info)

Returns Platform info dictionary

```
>>> p = get_platform()
>>> print(p)
```

vmam.**new_config**(*path='/etc/vmam/vmam.yml'*)
Create a new vmam config file (YAML)

Parameters **path** – Path of config file

Returns None

```
>>> new_config('/tmp/vmam.yml')
```

vmam.**bind_ldap**(*server*, *user*, *password*, *, *tls=False*)
Bind with user a LDAP connection

Parameters

- **server** – LDAP connection object
- **user** – user used for bind
- **password** – password of user
- **tls** – if True, start tls connection

Returns LDAP bind object

```
>>> conn = connect_ldap(['dc1.foo.bar'])
>>> bind = bind_ldap(conn, r'domain\user', 'password', tls=True)
>>> print(bind)
```

vmam.**check_connection**(*ip*, *port*, *timeout=3*)
Test connection of remote (ip) machine on (port)

Parameters

- **ip** – ip address or hostname of machine
- **port** – tcp port
- **timeout** – set timeout of connection

Returns Boolean

```
>>> check_connection('localhost', 80)
```

vmam.**check_config**(*path*)
Check YAML configuration file

Parameters **path** – Path of configuration file

Returns Boolean

```
>>> cfg = check_config('/tmp/vmam.yml')
```

vmam.**connect_ldap**(*servers*, *, *ssl=False*)
Connect to LDAP server (SYNC mode)

Parameters

- **servers** – LDAP servers list
- **ssl** – If True, set port to 636 else 389

Returns LDAP connection object

```
>>> conn = connect_ldap(['dc1.foo.bar'], ssl=True)
>>> print(conn)
```

vmam.**unbind_ldap**(*bind_object*)
Unbind LDAP connection

Parameters **bind_object** – LDAP bind object

Returns None

```
>>> conn = connect_ldap(['dc1.foo.bar'])
>>> bind = bind_ldap(conn, r'domain\user', 'password', tls=True)
>>> bind.unbind()
```

vmam.**query_ldap**(*bind_object*, *base_search*, *attributes*, *comp='='*, **filters*)
Query LDAP

Parameters

- **bind_object** – LDAP bind object
- **base_search** – distinguishedName of LDAP base search
- **attributes** – list of returning LDAP attributes
- **comp** – comparison operator. Default is '='. Accepted:

  Equality (attribute=abc) =

  Negation (!attribute=abc) !

  Presence (attribute=*) =*

  Greater than (attribute>=abc) >=

  Less than (attribute<=abc) <=

  Proximity (attribute~=abc) ~=
- **filters** – dictionary of ldap query

Returns query result list

```
>>> conn = connect_ldap(['dc1.foo.bar'])
>>> bind = bind_ldap(conn, r'domain\user', 'password', tls=True)
>>> ret = query_ldap(bind, 'dc=foo,dc=bar', ['sn', 'givenName'], objectClass=
↪'person', samAccountName='person1')
>>> print(ret)
```

vmam.**check_ldap_version**(*bind_object*)
Determines the LDAP version

Parameters **bind_object** – LDAP bind object

Returns LDAP version code: MS-LDAP or N-LDAP or LDAP

```
>>> conn = connect_ldap(['dc1.foo.bar'])
>>> bind = bind_ldap(conn, r'domain\user', 'password', tls=True)
>>> ret = check_ldap_version(bind)
>>> print(ret)
```

vmam.**new_user**(*bind_object*, *username*, ***attributes*)
Create a new LDAP user

Parameters

- **bind_object** – LDAP bind object
- **username** – distinguishedName of user
- **attributes** – Dictionary attributes

Returns LDAP operation result

```
>>> conn = connect_ldap(['dc1.foo.bar'])
>>> bind = bind_ldap(conn, r'domain\user', 'password', tls=True)
>>> new_user(bind, 'CN=ex_user1,OU=User_ex,DC=foo,DC=bar', objectClass='user',␣
↪givenName='User 1', sn='Example')
```

vmam.**set_user**(*bind_object*, *username*, ***attributes*)
Modify an exists LDAP user

Parameters

- **bind_object** – LDAP bind object
- **username** – distinguishedName of user
- **attributes** – Dictionary attributes

Returns LDAP operation result

```
>>> conn = connect_ldap(['dc1.foo.bar'])
>>> bind = bind_ldap(conn, r'domain\user', 'password', tls=True)
>>> set_user(bind, 'CN=ex_user1,OU=User_ex,DC=foo,DC=bar', givenName='User 1', sn=
↪'Example')
```

vmam.**delete_user**(*bind_object*, *username*)
Modify an exists LDAP user

Parameters

- **bind_object** – LDAP bind object
- **username** – distinguishedName of user

Returns LDAP operation result

```
>>> conn = connect_ldap(['dc1.foo.bar'])
>>> bind = bind_ldap(conn, r'domain\user', 'password', tls=True)
>>> delete_user(bind, 'CN=ex_user1,OU=User_ex,DC=foo,DC=bar')
```

vmam.**set_user_password**(*bind_object*, *username*, *password*, *, *ldap_version='LDAP'*)
Set password to LDAP user

Parameters

- **bind_object** – LDAP bind object
- **username** – distinguishedName of user
- **password** – password to set of user
- **ldap_version** – LDAP version (LDAP or MS-LDAP)

Returns None

```
>>> conn = connect_ldap('dc1.foo.bar')
>>> bind = bind_ldap(conn, r'domain\user', 'password', tls=True)
>>> new_user(bind, 'CN=ex_user1,OU=User_ex,DC=foo,DC=bar', givenName='User 1', sn=
↪'Example')
>>> set_user_password(bind, 'CN=ex_user1,OU=User_ex,DC=foo,DC=bar', 'password',␣
↪ldap_version='MS-LDAP')
>>> set_user(bind, 'CN=ex_user1,CN=Users,DC=office,DC=bol', pwdLastSet=-1,␣
↪userAccountControl=66048)
```

vmam.**add_to_group**(*bind_object*, *groupname*, *members*)
Add a member of exists LDAP group

Parameters

- **bind_object** – LDAP bind object
- **groupname** – distinguishedName of group
- **members** – List of a new members

Returns LDAP operation result

```
>>> conn = connect_ldap('dc1.foo.bar')
>>> bind = bind_ldap(conn, r'domain\user', 'password', tls=True)
>>> add_to_group(bind, 'CN=ex_group1,OU=Groups,DC=foo,DC=bar', 'CN=ex_user1,
↪CN=Users,DC=office,DC=bol')
```

vmam.**remove_to_group**(*bind_object*, *groupname*, *members*)
Remove a member of exists LDAP group

Parameters

- **bind_object** – LDAP bind object
- **groupname** – distinguishedName of group
- **members** – List of a removed members

Returns LDAP operation result

```
>>> conn = connect_ldap('dc1.foo.bar')
>>> bind = bind_ldap(conn, r'domain\user', 'password', tls=True)
>>> remove_to_group(bind, 'CN=ex_group1,OU=Groups,DC=foo,DC=bar', 'CN=ex_user1,
↪CN=Users,DC=office,DC=bol')
```

vmam.**filetime_to_datetime**(*filetime*)
Convert MS filetime LDAP to datetime

Parameters **filetime** – filetime number (nanoseconds)

Returns datetime object

```
>>> dt = filetime_to_datetime(132130209369676516)
>>> print(dt)
```

vmam.**datetime_to_filetime**(*date_time*)
Convert datetime to LDAP MS filetime

Parameters **date_time** – datetime object

Returns filetime number

```
>>> ft = datetime_to_filetime(datetime.datetime(2001, 1, 1))
>>> print(ft)
```

vmam.**get_time_sync**(*timedelta*)
It takes the date for synchronization

Parameters **timedelta** – Time difference to subtract (string: 1s, 2m, 3h, 4d, 5w)

Returns datetime object

```
>>> td = get_time_sync('1d')
>>> print(td)
```

vmam.**string_to_datetime**(*string*)
Convert string date to datetime

Parameters **string** – Datetime in string format ('dd/mm/yyyy' or 'mm/dd/yyyy')

Returns Datetime object

```
>>> dt = string_to_datetime('28/2/2019')
>>> print(dt)
```

vmam.**format_mac**(*mac_address*, *mac_format='none'*)
Format mac-address with the specified format

Parameters

- **mac_address** – mac-address in any format
- **mac_format** – mac format are (default=none):

  none 112233445566

  hypen 11-22-33-44-55-66

  colon 11:22:33:44:55:66

  dot 1122.3344.5566

Returns mac-address with the specified format

```
>>> m = format_mac('1A2b3c4D5E6F', 'dot')
>>> print(m)
```

vmam.**connect_client**(*client*, *user*, *password*)
Connect to client with WINRM protocol

Parameters

- **client** – hostname or ip address
- **user** – username used for connection on client
- **password** – password of user

Returns WINRM protocol object

```
>>> cl = connect_client('host1', r'domain\user', 'password')
>>> print(cl)
```

vmam.**run_command**(*protocol*, *command*)
Run command to a WINRM client

Parameters

- **protocol** – WINRM protocol object
- **command** – command to run on client

Returns Output of command

```
>>> cl = connect_client('host1', r'domain\user', 'password')
>>> cmd = run_command(cl, 'ipconfig /all')
>>> print(cmd)
```

vmam.**get_mac_address**(*protocol*, **exclude*)
Get mac-addresses to remote client

Parameters **protocol** – WINRM protocol object

Returns list mac-address

```
>>> cl = connect_client('host1', r'domain\user', 'password')
>>> m = get_mac_address(cl)
>>> print(m)
```

vmam.**get_client_user**(*protocol*)
Get the last user who logged in to the machine

Parameters **protocol** – WINRM protocol object

Returns user string

```
>>> cl = connect_client('host1', r'domain\user', 'password')
>>> u = get_client_user(cl)
>>> print(u)
```

vmam.**check_vlan_attributes**(*value*, *method='like'*, **attributes*)
Check VLAN attributes with like or match method

Parameters

- **value** – value to check

- **method** – 'like' or 'match'
- **attributes** – list of attributes

Returns boolean

```
>>> conn = connect_ldap(['dc1.foo.bar'])
>>> bind = bind_ldap(conn, r'domain\user', 'password', tls=True)
>>> user = query_ldap(bind, 'dc=foo,dc=bar', ['memberof', 'description',
↪'department'],
                      objectClass='person', samAccountName='person1')
>>> ok = check_vlan_attributes('business', user[0].get('attributes').get(
↪'description'))
>>> print(ok)
```

vmam.**get_mac_from_file**(*path*, *mac_format='none'*)

Get mac-address from file list

Parameters

- **path** – Path of file list. Mac-address can write in any format.

  file example (/tmp/list.txt):

  112233445566

  # mac of my Linux

  11-22-33-44-55-66

  # this macs is

  # other pc of my office

  11:22:33:44:55:66

  1122.3344.5566

- **mac_format** – mac format are (default=none):

  none 112233445566

  hypen 11-22-33-44-55-66

  colon 11:22:33:44:55:66

  dot 1122.3344.5566

Returns list

```
>>> get_mac_from_file('/tmp/list')
```

vmam.**timestamp_to_datetime**(*timestamp*)

Convert LDAP Kerberos timestamp LDAP to datetime

Parameters **timestamp** – kerberos timestamp string

Returns datetime object

```
>>> dt = timestamp_to_datetime('20200903053604Z')
>>> print(dt)
```

vmam.**datetime_to_timestamp**(*date_time*)

Convert datetime to LDAP Kerberos timestamp

Parameters **date_time** – datetime object

Returns kerberos timestamp string

```
>>> ft = datetime_to_timestamp(datetime.datetime(1986, 1, 25))
>>> print(ft)
```

1.7 Support

This python module is open source, under GPLv3 license. Anyone who wants, passion, respect and love can contribute to this project, for himself and for all of humanity.

1.7.1 Contribute

If you wanted to contribute to this python module, fork.

1.7.2 Issue

If you have found a bug or have an improvement in mind, open an issue.

1.7.3 Donations

Donating is important. If you do not want to do it to me, do it to some companies that do not speculate. My main activity and the people of non-profit associations is to work for others, be they male or female, religious or non-religious, white or black or yellow or red, rich and poor. The only real purpose is to serve the humanity of one's experience. Below you will find some links to do it. **Thanks a lot**.

For me

For Telethon

The Telethon Foundation is a non-profit organization recognized by the Ministry of University and Scientific and Technological Research. They were born in 1990 to respond to the appeal of patients suffering from rare diseases. Come today, we are organized to dare to listen to them and answers, every day of the year.

Adopt the future

1.8 Indices and tables

- genindex
- modindex
- search

Python Module Index

v

Index