MerLink Documentation

Release 0.8.5

Ross Jacobs

Sep 05, 2018

Contents:

Use this is some capacity: https://www.divio.com/blog/documentation/

CHAPTER 1

Merlink API

1.1 CLI modules

1.1.1 cli_modal_prompts

CLI modal prompts to send information to console.

class src.cli.cli_modal_prompts.**CliModalPrompts**
CliModalPrompts class constructor, inits with no params.

static show_error_dialog(*message*)
Prints an error message

Parameters **message** – An error message to show to the user

static show_feature_in_development_dialog()
Prints a message if user encounters a work in progress feature.

1.1.2 merlink_cli

Provides a CLI for MerLink

class src.cli.merlink_cli.**MainCli**
MerLink CLI Based on ncurses

Command line options spec based on expected use cases:

1. To enter all information manually Usage: merlink –name <vpn name> –address <server IP/FQDN> –psk <PSK> –username <username> –password <password>

ex. merlink –name 'ACME VPN' –address 'acme.corp' –psk '@AnyCost' –username 'wile.e.coyote@acme.corp' –password 'SuperGenius!'

2. Interactive mode: If dashboard username/password are known Usage: merlink –username <username> –password <password> << Organizations and networks are shown to user >> User selects the network they would like to connect to << program creates and connects vpn

3. If username + password of dashboard admin, organization name, and network name are already known Usage: merlink –username <username> –password <password> –organization <organization> –network <network> –options <options>

ex. merlink –username 'wile.e.coyote@acme.corp' –password 'SuperGenius!' –organization 'ACME Corp' –network 'Wild West'

NOTE: If there are conflicts in organization or network names, there will be an error and it may be wise to use organization_id/network_id instead.

add_vpn()
Adds a vpn by name

Currently not implemented but mirrored in main_window

connect_vpn(**vpn_vars*)
Connects using the specified vpn connection

Currently not implemented but mirrored in main_window

get_user_action()
Gets the user's next action

Currently not implemented but mirrored in main_window

static parse_options()
Parses argparse options and then calls other parts of program.

show_main_menu()
Shows the main cli menu

Currently not implemented but mirrored in main_window

show_result(*result*)
Shows the result of the vpn connection to the console

Currently not implemented but mirrored in main_window

troubleshoot_vpn()
Provides an interface to interact with troubleshooting functionality.

Currently not implemented but mirrored in main_window

1.2 GUI modules

1.2.1 login_dialog

Login dialog GUI elements.

class src.gui.login_dialog.**LoginDialog**
Create UI vars necessary for login window to be shown.

check_login_attempt()
Verifies whether entered username/password combination is correct

NOTE: Keeping this code in here even though it is interface-independent. If we don't keep this here, then the login button will need to connect to self.close. It may look weird if the login window closes due to the user incorrectly entering user/pass and then reopens

get_login_info()
Returns the values currently in the user/pass text fields

show_login()
Shows the login window and records if the login button is pressed.

Uses methods stored in gui_setup to decorate the dialog object. If the login button is pressed, check whether the credentials are valid by sending them to the virtual browser.

tfa_dialog_setup()
Create and execute the UI for the TFA dialog

tfa_verify()
Submit the tfa code and communicate success/failure to user.

This fn is partially required because we need a function to connect to the button click signal.

1.2.2 main_window

Main Window is the controlling class for the GUI.

class src.gui.main_window.**MainWindow**
Main Window is the controlling class for the GUI.

browser
DashboardBrowser – Browser used to store user credentials

menu_widget
MenuBars – Used to tie the menu bars to the MainWindow

tray_icon
SystrayIcon – Used to tie the tray icon to the MainWindow

change_network()
Change the network to new value for both model and view

This will have been triggered by a network dropdown change. Get network info for this network and let user know.

change_organization()
Change the org by getting required data and showing it to user

- If we don't have data for this org, get info; otherwise don't.
- If the user has not selected an organization, this fn will do nothing.

communicate_vpn_failure()
Let the user know that the VPN connection failed.

communicate_vpn_success()
Let the user know that they are connected.

is_vpn_connected()
Determines whether the VPN is connected. TODO: Implement this function :returns: Whether or not there is an active VPN connection :rtype: vpn_status (bool)

refresh_network_dropdown()
Remove old values of the network dropdown and add new ones.

Remove previous contents of Networks QComboBox and add new ones according to chosen organization

set_admin_layout()
Set the dashboard admin layout.

set_guest_layout()
Set the guest user layout.

setup_vpn()
Setup VPN vars and start OS-dependent connection scripts

Passes vpn vars that are required for an L2TP connection as list vpn_data. Passes OS-specific parameters as list <OS>_options.

show_main_menu()
Show the main menu GUI

Is called on main_window instantiation. Creates the scaffolding for the main menu GUI and generates all GUI elements that will be later used by other class methods.

Has 2 radio option Qt signals/slots:

- Admin user radio option toggled -> Set admin user/pass view
- Guest user radio option toggled -> Set guest user/pass view

Has 3 program-driving Qt signals/slots:

- Organization dropdown changed -> Update model and view
- Network dropdown changed -> Update model and view
- "Connect" button clicked -> Initiate VPN connection

tshoot_vpn_fail_gui()
Troubleshoot VPN failure and then show the user the results.

1.2.3 main_window_ui

1.2.4 modal_dialogs

https://ux.stackexchange.com/questions/12045/what-is-a-modal-dialog-window

"A modal dialog is a window that forces the user to interact with it before they can go back to using the parent application."

This script contains multiple GUI modal dialogs.

src.gui.modal_dialogs.**show_error_dialog**(*message*)
Show an error dialog with a message.

Parameters **message** (*string*) – A message telling the user what is wrong.

src.gui.modal_dialogs.**show_feature_in_development_dialog**()
Informs the user that something is a feature in development.

src.gui.modal_dialogs.**show_question_dialog**(*message*)
Send the user a question and record their decision.

Parameters **message** (*string*) – A question asking the user what they want to do.

Returns Returns a QDialog code of Rejected (no) | Accepted (yes) depending on user input.

Return type result (QDialog.DialogCode)

src.gui.modal_dialogs.**vpn_status_dialog**(*title*, *message*)
Tells the user the status of the VPN connection.

Parameters

- **title** (*string*) – A window title to summarize the message.
- **message** (*string*) – A message to give to the user.

1.2.5 systray

This class manages the system tray icon.

class src.gui.systray.**SystrayIcon**(*app*)
This class manages the system tray icon of the main program, post-login.

app
QMainWindow – Set to MainWindow object (required binding for Qt)

tray_icon
QSystemTrayIcon – System Tray object that has all of the functionality that this class requires.

application_minimized()
Minimize the window.

icon_activated(*reason*)
The user has clicked on the systray icon, so respond

If single or double click, show the application If middle click, go to meraki.cisco.com Override closeEvent, to intercept the window closing event

Parameters **reason** (*QSystemTrayIcon.ActivationReason*) – An enum of [0,4] of how the user interacted with the system tray ~ More information on ActivationReasons can be found here: http://doc.qt.io/qt-5/qsystemtrayicon.html#ActivationReason-enum

set_vpn_failure()
Tell user that VPN connection was unsuccessful.

Show an icon of Miles with a red interdictory circle and let the user know the connection failed.

set_vpn_success()
Tell user that VPN connection was successful.

NOTE: There's no such thing as "minimize to system tray". What we're doing is hiding the window and then adding an icon to the system tray

This function will set the icon to Miles with 3D glasses and show a message that the connection was successful.

1.2.6 tshoot_failed_vpn_dialog

After a failure to connect, this function will GUIfy the cause.

src.gui.tshoot_failed_vpn_dialog.**tshoot_failed_vpn_dialog**(*has_passed_validation*)
After a failure to connect, this function will GUIfy the cause.

Parameters

- **has_passed_validation** (*list(bool)*) – A list of bools corresponding
- **the success of 6 tests validating against common misconfigurations.** (*to*) –

Returns A QListWidget that has checkmark/X icons and accompanying text according to whether that text's test passed.

Return type (QListWidget)

1.3 Utilities

1.3.1 controller

This is the MVC controller for both gui and cli interfaces

class src.modules.controller.**Controller**
This is the MVC controller for both gui and cli interfaces.

This class will be called by /merlink.py and will control the application. Controller uses an interface object whose methods are implemented by both MainWindow and MainCli.

interface
MainWindow | *MainCli* – Calls interface-dependent functions (this is a primitive form of overloading).

app
QApplication – Required for the Qt program flow, not used for CLI

program_structure()
Contains the interface-independent structure of the program.

Currently, it implements a couple functions that start the GUI application while the CLI is still unwritten.

Main menu should show the following across interfaces: 1. Existing VPN connections that we can connect to 2. After list of VPN connections, have a "+ Add a connection" option 3. Indicate which VPN connections that are currently active (if any) 4. Route table (should change when a VPN connection is made) 5. Latency/loss/bandwidth graph used for a connected VPN

It should return the next user action

1.3.2 dashboard_browser

1.3.3 os_utils

src.modules.os_utils.**is_duplicate_application**(*program_name*)
Detect whether there are multiple processes with the same name.

src.modules.os_utils.**is_online**()
Detects whether the device is connected to the internet

src.modules.os_utils.**list_vpns**()
This script will get the existing VPN connections from the OS.

src.modules.os_utils.**open_vpnsettings**()
Opens OS-specific VPN settings.

src.modules.os_utils.**pyinstaller_path**(*relative_path*)
When using the –onefile flag, PyInstaller will by default extract necessary files into a temporary folder named '_MEIPASS2'. In order for the executable to access them, file paths must be modified to include this folder name.

Executables using –onedir are not affected as the files are where they are expected to be in the original or installation folder

Modified form source: https://stackoverflow.com/questions/7674790

1.3.4 vpn_tests

This should run after a connection has resulted in failure.

class src.modules.vpn_tests.**TroubleshootVpn**(*fw_status_text*, *client_vpn_text*, *ddns*, *firewall_ip*)

Checks for things that could prevent the VPN connection.

This class will perform the (currently 6) tests to see what might be causing VPN issues. Once all tests have been done, all results can be retrieved with the get_test_results() method.

pagetext
dict – Contain these input variables: * HTML text of appliance status page * HTML text of client vpn page

addr
dict – Contain these input variables: * ddns address of the current firewall * ip address of the current firewall

test_results
list(bool) – A bool list of tests passed/failed

get_test_results()
Returns the test results after tests have been run.

run_tests()
Iterate through all of the tests.

test0_is_mx_online()
Tests whether there is an MX in the appliance network.

No 'status#' in HTML means there is no firewall in that network

Raises In progress...

test1_is_fw_reachable()
Verify whether firewall is reachable by pinging 4 times

If at least one ping that made it, mark this test as successful.

test2_is_user_behind_fw()
Tests whether the user is behind the firewall.

If the user is behind their firewall, they will not be able to connect (and a VPN connection would be pointless.) request_ip is the IP is the source public IP that the user is connecting with.

test3_is_client_vpn_enabled()
Tests whether client VPN is enabled.

If client VPN is not enabled, the firewall will not respond to client VPN requests.

test4_is_auth_type_meraki()
Is the authentication type is Meraki Auth?

For the time being, flag use of RADIUS or Active Directory as an error as those auth types aren't being tested against.

When an auth type is selected, we get one of these in the client VPN HTML depending on user's auth choice:

Meraki cloud</option></select> Active Directory</option></select> RADIUS</option></select>

test5_incompatible_port_forwards()
Test for port forwards that break an IPSEC VPN.

An IPSEC connection uses UDP port 500 and UDP port 4500 if there is NAT. If the following text exists, they're port forwarding ports 500 or 4500: "public_port":"500" "public_port":"4500"

1.3.5 vpn_connection

Given VPN vars, uses OS built-ins to create/connect a L2TP/IPSEC VPN.

class src.modules.vpn_connection.**VpnConnection**(*vpn_data*)

VpnConnection list arguments

vpn_data vpn_name psk ip ddns username password

windows_options dns_suffix idle_disconnect_seconds split_tunneled remember_credentials use_winlogon DEBUG

VpnConnection takes 2 lists as args: vpn_data and vpn_options Required VPN parameters will arrive in vpn_data Any OS-specific VPN parameters will go into vpn_options

attempt_linux_vpn(*vpn_options*)
Attempt to connect on linux.

- sudo required to create a connection with nmcli
- pkexec is built into latest Fedora, Debian, Ubuntu.
- 'pkexec <cmd>' correctly asks in GUI on Debian, Ubuntu but in terminal on Fedora
- pkexec is PolicyKit, which is the preferred means of asking for permission on LSB

attempt_macos_vpn(*vpn_options*)
Attempt to connect over VPN on macOS.

scutil is required to add the VPN to the active set. Without this, it is not possible to connect, even if a VPN is listed in Network Services

scutil –nc select <connection> throws '0:227: execution error: No service (1)'. if it's a part of the build script instead of here. This is why it's added directly to the osacript request. Connection name with forced quotes in case it has spaces.

attempt_windows_vpn(*vpn_options*)
Attempt to connect to Windows VPN.

- Arguments sent to powershell MUST BE STRINGS
- Each argument cannot be the empty string or null or PS will think there's no param there!!!
- Last 3 ps params are bools converted to ints (0/1) converted to strings. It's easy to force convert '0' and '1' to ints on powershell side.
- Setting execution policy to unrestricted is necessary so that we can access VPN functions
- Email CANNOT have spaces, but password can.

disconnect()
Disconnect any connected VPN

static is_vpn_connected()
Detect whether VPN is connected or not.

sanitize_variables()
Sanitize variables for powershell/bash input.

CHAPTER 2

Development resources

2.1 Documentation

2.1.1 Google Python Style Guide

- Google's Python Style Guide
- Docstring examples

2.1.2 reStructuredText and Sphinx

- Brandon Rhodes' readthedocs.io Sphinx Tutorial
- A primer on reStructuredText

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

S

Index

R

S

T

V