Omnissiah code

Architecture

The Omnissiah program code is a set of small programs and libraries, each of which performs a single action. One cycle of work is the sequential launch of these programs. A relational database is used as a repository for program results. Processing of the tables of this database is carried out by these programs using SQL queries.

Virtual environment and libraries

To run the Omnissiah code we use a virtual environment in the omnienv directory. When creating this virtual environment, the libraries installed on the system at that time are not copied here. The following libraries are installed for the programs to work (via PIP):

- wheel
- requests
- setproctitle
- munch
- pyparsing
- pyzabbix
- pynetbox
- easysnmp
- python-nmap
- mariadb
- psycopg2-binary
- torch

Common modules

The scripts use common python modules from the omnissiah directory

#	*Module*	*Description*
1	activaire.py	working with Activaire API
2	const.py	constants
3	db.py	working with the Omnissiah database and tables
4	enplug.py	working with Enplug API
5	mist.py	working with MIST wireless API
6	msg.py	messages
7	nnml.py	creation, training, saving and loading of neural networks
8	omnissiah.py	common code for all programs
9	ruckussz.py	working with Ruckus wireless controllers
10	util.py	various small functions
11	zbx.py	working with Zabbix web API

Common modules table

Programs

Programs are grouped into layers (except for several files used by different programs)

#	*Layer*	*Program*	*Description*
1		omni_config.py	parameters for programs
2		omni_const.py	common constants

3		omni_cycle.sh	shell script with Omnissiah cycle
4		omni_unpwd.py	usernames, passwords, keys, tokens and other sensitive information
5	raw	raw_activaire.py	getting data from the Activaire API and writing it to tables
6	raw	raw_enplug.py	getting data from Enplug API and writing it to tables
7	raw	raw_mac.py	obtaining MAC addresses from IEEE and writing them into tables
8	raw	raw_map.py	scanning hosts found by raw_scan.py
9	raw	raw_mist.py	receiving data from MIST wireless API and writing it to tables
10	raw	raw_netbox.py	obtaining network information from Netbox and recording it in tables
11	raw	raw_ruckussz.py	receiving data from Ruckus wireless controllers and writing it to tables
12	raw	raw_scan.py	network scanning and host search
13	raw	raw_snmp.py	SNMP polling of hosts found by raw_map.py and raw_scan.py
14	info	info_mac.py	processing data collected by raw_mac.py
15	info	info_netbox.py	processing data collected by raw_netbox.py
16	ref	ref_netbox.py	synchronization of ref tables with data from Netbox
17	src	src_activaire.py	processing data collected by raw_activaire.py
18	src	src_addr.py	extracting address information (IP, MAC, ports) from raw_scan.py and raw_map.py data, etc.
19	src	src_enplug.py	processing data collected by raw_enplug.py
20	src	src_mist.py	processing data collected by raw_mist.py
21	src	src_ruckussz.py	processing data collected by raw_ruckussz.py
22	src	src_scan.py	processing data collected by raw_scan.py and raw_map.py
23	src	src_snmp.py	processing data collected by raw_snmp.py
24	nnml	nnml_label.py	host labeling for neural network
25	nnml	nnml_predict.py	prediction of device type, manufacturer, etc. by neural network
26	nnml	nnml_prepare.py	preparing tables for the neural network
27	nnml	nnml_train.py	neural network training
28	shot	shot_activaire.py	Activaire devices in the current network snapshot
29	shot	shot_enplug.py	Enplug devices in the current network snapshot
30	shot	shot_host.py	hosts in the current network snapshot
31	shot	shot_mist.py	access points from MIST in the current network snapshot
32	shot	shot_nnml.py	neural network predictions in the current network snapshot
33	shot	shot_router.py	routers in the current network snapshot
34	shot	shot_ruckussz.py	access points from Ruckus controllers in the current network snapshot
35	shot	shot_wap.py	access points in the current network snapshot
36	main	main_addr.py	addressing (IP, MAC, ports) in accordance with the network snapshot in the shot layer
37	main	main_host.py	hosts according to the network snapshot in the shot layer
38	zbx	zbx_main2zbx.py	preparing a configuration that should be uploaded to Zabbix in accordance with the network image in the main layer
39	zbx	zbx_omni2zbx.py	managing the Zabbix config according to the results of zbx_main2zbx.py
40	zbx	zbx_zbx2omni.py	getting the current config from Zabbix
41	hist	hist_dump.py	creating a database dump in a file

Omnissiah programs table

Logs

Each program writes its log to a separate file with the name of this program and the extension log. The directory with logs is located in /var/log/omnissiah Which messages are recorded is determined by the program itself. In any case, a program crash message will be recorded with diagnostic information.