Aztarna Documentation

Release 1.0

Alias Robotics

Aug 24, 2020

CONTENTS:

Alias Robotics supports original robot manufacturers assessing their security and improving their quality of software. By no means we encourage or promote the unauthorized tampering with running robotic systems. This can cause serious human harm and material damages.

- A list of the ROS nodes present in the system (Publishers and Subscribers)
- For each node, the published and subscribed topis including the topic type
- For each node, the ROS services each of the nodes offer
- A list of all ROS parameters present in the Parameter Server
- A list of the active communications running in the system. A single communication includes the involved publiser/subscriber nodes and the topics

CHAPTER

ONE

FOR SROS

- Determining if the system is a SROS master.
- Detecting if demo configuration is in use.
- A list of the nodes found in the system. (Extended mode)
- A list of allow/deny policies for each node.
  - Publishable topics.
  - Subscriptable topics.
  - Executable services.
  - Readable parameters.

CHAPTER

TWO

FOR ROS2

- Detection of ROS2 nodes in all possible ROS2 domain IDs. Local network.
- Listing of all available topics and their relationship to nodes.
- Listing of all available services and their relationship to nodes.

CHAPTER

THREE

FOR INDUSTRIAL ROUTERS

- Detecting eWON, Moxa, Sierra Wireless and Westermo industrial routers.
- Default credential checking for found routers.

3.1 Installing

3.1.1 For production

```
pip3 install .
```

or

```
python3 setup.py install
```

3.1.2 For development

```
pip3 install -e .
```

or

```
python3 setup.py develop
```

The only requirement is setuptools package, which is usually a defacto standard in a python3 installation.

3.1.3 Install with docker

```
docker build -t aztarna_docker .
```

3.2 Code usage

```
usage: aztarna [-h] -t TYPE [-a ADDRESS] [-p PORTS] [-i INPUT_FILE]
               [-o OUT_FILE] [-e] [-r RATE] [--shodan] [--api-key API_KEY]

Aztarna
```

(continues on next page)

(continued from previous page)

```
optional arguments:
  -h, --help            show this help message and exit
  -t TYPE, --type TYPE  <ROS/ros/SROS/sros/ROS2/ros2/IROUTERS/irouters> Scan ROS,␣
↪SROS, ROS2
                        hosts or Industrial routers
  -a ADDRESS, --address ADDRESS
                        Single address or network range to scan.
  -p PORTS, --ports PORTS
                        Ports to scan (format: 13311 or 11111-11155 or
                        1,2,3,4)
  -i INPUT_FILE, --input_file INPUT_FILE
                        Input file of addresses to use for scanning
  -o OUT_FILE, --out_file OUT_FILE
                        Output file for the results
  -e, --extended        Extended scan of the hosts
  -r RATE, --rate RATE  Maximum simultaneous network connections
  --shodan              Use shodan for the scan types that support it.
  --api-key API_KEY     Shodan API Key
```

- Run the code (example input file):

```
aztarna -t ROS -p 11311 -i ros_scan_s20.csv
```

- Run the code with Docker (example input file):

```
docker run -v <host_path>:/root -it aztarna_docker -t ROS -p 11311 -i <input_file>
```

- Run the code (example single ip address):

```
aztarna -t ROS -p 11311 -a 115.129.241.241
```

- Run the code (example subnet):

```
aztarna -t ROS -p 11311 -a 115.129.241.0/24
```

- Run the code (example single ip address, port range):

```
aztarna -t ROS -p 11311-11500 -a 115.129.241.241
```

- Run the code (example single ip address, port list):

```
aztarna -t ROS -p 11311,11312,11313 -a 115.129.241.241
```

- Run the code (example piping directly from zmap):

```
zmap -p 11311 0.0.0.0/0 -q | aztarna -t SROS -p 11311
```

- Run the code (example search for industrial routers in shodan)

```
aztarna -t IROUTERS --shodan --api-key <yourshodanapikey>
```

- Run the code (example search for industrial routers in shodan, piping to file)

```
aztarna -t IROUTERS --shodan --api-key <yourshodanapikey> -o routers.csv
```

- Run the code (example search against ROS2 nodes)

```
aztarna -t ROS2
```

- Run the code (example search against ROS2 nodes in extended mode)

```
aztarna -t ROS2 -e
```

- Run the code (example search against ROS2 nodes in extended mode with file output)

```
aztarna -t ROS2 -e -o output.csv
```

3.3 Aztarna modules

3.3.1 aztarna package

3.3.1.1 Subpackages

aztarna.ros.ros package

Submodules

aztarna.ros.ros.helpers module

ROS Scanner helper module.

:author Alias Robotics SL (https://aliasrobotics.com)

class aztarna.ros.ros.helpers.**Node**(*name*)
Bases: aztarna.ros.commons.BaseNodeROS

Node class, an extension of the BaseNodeROS

class aztarna.ros.ros.helpers.**ROSHost**(*address*, *port*)
Bases: aztarna.ros.commons.BaseHostROS

Class for keeping all the attributes of a ROS Node.Extends:class:*aztarna.commons.BaseHostROS*

class aztarna.ros.ros.helpers.**Service**(*name*)
Bases: aztarna.ros.commons.BaseServiceROS

Service class, an extension of BaseServiceROS

class aztarna.ros.ros.helpers.**Topic**(*name*, *topic_type*)
Bases: aztarna.ros.commons.BaseNodeROS

Topic class, an extension of BaseNodeROS

aztarna.ros.ros.scanner module

ROS Scanner module.

:author Alias Robotics SL (https://aliasrobotics.com)

class aztarna.ros.ros.scanner.**ROSScanner**
Bases: *aztarna.commons.RobotAdapter*

ROSScanner class, an extension of BaseScanner for ROS.

analyze_nodes(*address*, *port*)
Scan a node and gather all its data including topics, services and Communications.

Parameters

- **address** – address of the ROS master
- **port** – port of the ROS master

static analyze_topic_types(*ros_master_client*)
Extract topic from ROS Master and disassemble them into topic name and topic type.

Parameters **ros_master_client** – xml-rpc object for the ROS Master Client

Returns A dictionary of topics. Key is the topic name and value the topic type

extract_nodes(*source_array*, *topics*, *pub_or_sub*, *host*)
From all the data ROS Master returns, extract just the node info.

Parameters

- **source_array** – A multiple level array containing data from the the ROS system state
- **topics** – A list of all topics found in the ROS system
- **pub_or_sub** – A boolean to separate publisher and subscriber nodes
- **host** – Current ROS host

extract_services(*source_array*, *host*)
Extract the services from the ROS system state.

Parameters

- **source_array** – A multiple level array containing data from the the ROS system state
- **host** – Current ROS host

static get_create_node(*node_name*, *host*)
Generate new `aztarna.ros.helpers.Node` objects, and if they exist just return them.

Parameters

- **node_name** – The name of the node to create or return
- **host** – Current ROS host

Returns The newly created node or an existing that matches `node_name`

print_results()
Print the information of a ROS system.

scan()
Call to `aztarna.ros.scanner.scan_network()` asynchronously

scan_network()
Scan the provided network (from args) searching for ROS nodes.

scan_pipe()

scan_pipe_main()

set_xmlrpcuri_node(*ros_master_client*, *host*)
Once all node data is collected, set the xml.

Parameters **ros_master_client** – xml-rpc object for the ROS Master Client

write_to_file(*out_file*)
Write the information of a ROS system into the provided file.

Parameters **out_file** – The file where to write the results

Module contents

aztarna.ros.ros2 package

Submodules

aztarna.ros.ros2.helpers module

class aztarna.ros.ros2.helpers.**ROS2Host**
Bases: object

class aztarna.ros.ros2.helpers.**ROS2Node**
Bases: object

class aztarna.ros.ros2.helpers.**ROS2Service**
Bases: object

class aztarna.ros.ros2.helpers.**ROS2Topic**
Bases: object

aztarna.ros.ros2.helpers.**raw_services_to_pyobj_list**(*services*) → List[aztarna.ros.ros2.helpers.ROS2Service]
Utility function for converting the raw data returned by the ROS2 API into a list of python service objects.

Parameters **services** – Raw services list.

Returns A list containing all parsed *aztarna.ros.ros2.helpers.ROS2Service* objects.

aztarna.ros.ros2.helpers.**raw_topics_to_pyobj_list**(*topics*, *include_default=False*) → List[aztarna.ros.ros2.helpers.ROS2Topic]
Utility function for converting the raw data returned by the ROS2 API into a list of python topic objects.

Parameters

- **topics** – Raw topics list.
- **include_default** – If to include the default topic names on the returned list or not.

Returns A list containing all parsed *aztarna.ros.ros2.helpers.ROS2Topic* objects.

aztarna.ros.ros2.scanner module

Module contents

aztarna.ros.sros package

Submodules

aztarna.ros.sros.helpers module

SROS Scanner helpers classes module. :author Alias Robotics SL (https://aliasrobotics.com)

class aztarna.ros.sros.helpers.**SROSHost**
Bases: aztarna.ros.commons.BaseHostROS

Class for keeping all the attributes of a SROS Node.Extends:class:*aztarna.commons.BaseHostROS*

class aztarna.ros.sros.helpers.**SROSNode**
Bases: aztarna.ros.commons.BaseNodeROS

Class for keeping all the attributes of a SROS Node. Extends aztarna.commons.BaseNodeROS

class aztarna.ros.sros.helpers.**SROSPolicy**
Bases: object

Class for representing a SROS Policy, containing the possible types for it, the value and its permissions.

POLICY_ALLOWED = True

POLICY_DENIED = False

TYPE_EXECUTABLE_SVCS = 'Executable services'

TYPE_PUBLISHABLE_TOPICS = 'Publishable topics'

TYPE_READABLE_PARAMS = 'Readable parameters'

TYPE_SUBSCRIPTABLE_TOPICS = 'Subscriptable topics'

TYPE_UNKNOWN = 'Unknown'

aztarna.ros.sros.helpers.**check_port**(*ip*, *port*)
Check if a port is open. :param ip: Address of the host :param port: Port to check :return: Returns the port if it is open. None otherwise.

aztarna.ros.sros.helpers.**check_port_sem**(*sem*, *ip*, *port*)
Check ports from a host, limiting concurrency with a semaphore.

Parameters

- **sem** – Asyncio semaphore.
- **ip** – Address of the host.
- **port** – Port to be checked.

Returns The result of aztarna.sros.helpers.check_port().

aztarna.ros.sros.helpers.**find_node_ports**(*address*, *ports*)
Find all the open ports for a host.

Parameters

- **address** – IP address of the host.
- **ports** – Port to check.

Returns A list of the found open ports.

aztarna.ros.sros.helpers.**get_node_info**(*cert*)
Extract all the information for a node, based on it's certificate. :param cert: The input certificate in X509 format. :return: aztarna.sros.helpers.SROSNode The extracted node info from the certificate.

aztarna.ros.sros.helpers.**get_policies**(*cert*)
Get the related policies from an input SROS Node certificate. :param cert: Certificate in X509 format. :return: List containing all policies as instances of aztarna.sros.helpers.SROSPolicy

aztarna.ros.sros.helpers.**get_sros_certificate**(*address*, *port*, *timeout=3*)
Function to connect to a SROS Node, simulate the TLS handshake, and get it's server certificate on the process. :param address: Address of the node. :param port: Port of the node. :param timeout: Timeout for the connection. :return: A tuple containing the address, port and certificate if found, otherwise, a tuple containing address, port and None.

aztarna.ros.sros.scanner module

SROS Scanner module.

:author Alias Robotics SL (https://aliasrobotics.com)

class aztarna.ros.sros.scanner.**SROSScanner**
Bases: *aztarna.commons.RobotAdapter*

SROS Scanner class, extending aztarna.commons.BaseScanner.

print_results()
Print the results of the scan into console. Extended from aztarna.commons.BaseScanner.

scan()
Run the scan for SROS hosts. Extended from aztarna.commons.BaseScanner. This function is the one to be called externally in order to run the scans. Internally those scans are run with the help of asyncio.

scan_host(*address: str*, *master_port: int*, *timeout=1*)
Scan a single SROS host and return a aztarna.sros.helpers.SROSHost instance with all the data if found.

Parameters

- **address** – Host IP address.
- **master_port** – Master node port.
- **timeout** – Timeout for the connection.

Returns aztarna.sros.helpers.SROSHost instance.

scan_network()
Scan all the hosts specified in the internal hosts list self.hosts.

Returns A list of aztarna.sros.helpers.SROSHost containing all the found hosts.

scan_pipe()

scan_pipe_main()

write_to_file(*out_file: str*)
Write the results to the specified output file. Extended from aztarna.commons.BaseScanner.

Parameters **out_file** – Output file name in which the results will be writen.

Module contents

aztarna.industrialrouters package

Submodules

aztarna.industrialrouters.scanner module

Industrial routers scanner module.

:author Alias Robotics SL (https://aliasrobotics.com)

class aztarna.industrialrouters.scanner.**BaseIndustrialRouter**
Bases: object

Base class for holding industrial routers.

class aztarna.industrialrouters.scanner.**BaseIndustrialRouterScanner**
Bases: object

Base class fo the different manufacturer router scanners. Includes default methods for Basic Authentication password checking and scanning.

classmethod check_default_password(*router*, *semaphore=<asyncio.locks.Semaphore object at 0x7f51195a5da0 [unlocked, value:1]>*)
Base method to check fo default credentials by using basic HTTP authentication schemes.

This method can be overwritten in order to support different authentication schemes. Valid credentials are appended in the valid credentials attribute of each router object.

Parameters

- **router** – The router for which to check the credentials.
- **semaphore** – Asyncio semaphore for limiting the concurrency level.

classmethod check_is_router(*address: str*, *port: int*, *semaphore=<asyncio.locks.Semaphore object at 0x7f51195a59b0 [unlocked, value:1]>*) → aztarna.industrialrouters.scanner.BaseIndustrialRouter
Check if a certain router is an industrial router, given the headers defined at class level.

Parameters

- **address** – IP address of the router to check.
- **port** – Port of the web interface of the device to check.
- **semaphore** – Asyncio semaphore to be used for concurrency limitation.

Returns A *aztarna.industrialrouters.scanner.BaseIndustrialRouter* object if the checked device is a router. None otherwise.

check_router_credentials(*routers: List[aztarna.industrialrouters.scanner.BaseIndustrialRouter]*)
Check default credentials for a list of routers.

Parameters **routers** – List of routers to be checked.

check_routers(*addresses: List[str], ports: List[int]*) → List[aztarna.industrialrouters.scanner.BaseIndustrialRouter]
Check for routers in a range of addressess and ports.

Parameters

- **addresses** – List of addressess to be checked.
- **ports** – List of ports to be checked for each address.

Returns A list of found routers.

classmethod check_routers_shodan(*shodan: shodan.client.Shodan*) → List[aztarna.industrialrouters.scanner.BaseIndustrialRouter]
Method to search for industrial routers in Shodan, given the headers defined at class level.

Parameters **shodan** – Shodan API object to be used.

Returns List of found routers.

default_credentials = []

get_address_info(*routers*)
Get country code and ASN description based on the routers IP address. :param routers: :return:

possible_headers = {}

router_cls = None

url_path = ''

class aztarna.industrialrouters.scanner.**EWonRouter**
Bases: *aztarna.industrialrouters.scanner.BaseIndustrialRouter*

Class for holding EWON Manufacturer routers.

class aztarna.industrialrouters.scanner.**EWonScanner**
Bases: *aztarna.industrialrouters.scanner.BaseIndustrialRouterScanner*

Scanner class for EWon routers.

default_credentials = [('adm', 'adm')]

possible_headers = {'Server': ['eWON']}

router_cls
alias of *EWonRouter*

url_path = 'Ast/MainAst.shtm'

class aztarna.industrialrouters.scanner.**IndustrialRouterAdapter**
Bases: *aztarna.commons.RobotAdapter*

Adapter for searching, analyzing and footprinting Industrial Routers.

initialize_shodan()
Intialize API connection to Shodan.

print_results()
Method for printing the scan results in stdout.

router_scanner_types = [<class 'aztarna.industrialrouters.scanner.SierraWirelessScanne

scan()
Method to be called in order to start the full scan procedure, based on Shodan, or locally via network scan.

scan_network()
Scan a network in search for industrial routers.

scan_pipe_main()

write_to_file(*out_file*)
Method for writing the scan results to a CSV file. :param out_file: Filename for the output

class aztarna.industrialrouters.scanner.**MoxaRouter**
Bases: *aztarna.industrialrouters.scanner.BaseIndustrialRouter*

Class for holding Moxa Manufacturer routers.

class aztarna.industrialrouters.scanner.**MoxaScanner**
Bases: *aztarna.industrialrouters.scanner.BaseIndustrialRouterScanner*

Scanner class for Moxa routers.

Due to the different authentication schema used by Moxa routers, methods for checking passwords have been extended.

classmethod check_default_password(*router: aztarna.industrialrouters.scanner.BaseIndustrialRouter*, *semaphore=<asyncio.locks.Semaphore object at 0x7f51195a5e10 [unlocked, value:1]>*)

Method for checking for default passwords on Moxa Routers.

Parameters

- **router** – Input router object to check the credentials for.
- **semaphore** – Asyncio semaphore for limiting the concurrency leve.

classmethod check_password_moxahttp_1_0(*client: aiohttp.client.ClientSession*, *context: ssl.SSLContext*, *content: str*, *router: aztarna.industrialrouters.scanner.BaseIndustrialRouter*)

Method for checking the passwords in MoxaHttp/1.0 router authentication schemas.

Parameters

- **client** – ClientSession for the connection to the router.
- **context** – SSLContext of the connection.
- **content** – Content of the response of the router.
- **router** – *aztarna.industrialrouters.scanner.BaseIndustrialRouter* router to check.

classmethod check_password_moxahttp_2_2(*client*, *context*, *content*, *router*)

Method for checking the passwords in MoxaHttp/2.2 router authentication schemas.

Parameters

- **client** – ClientSession for the connection to the router.
- **context** – SSLContext of the connection.
- **content** – Content of the response of the router.
- **router** – *aztarna.industrialrouters.scanner.BaseIndustrialRouter* router to check.

default_credentials_http1 = [('root', 'efa59ad49b7bc93a9a7bb1004f24b1cc'), ('', 'd41d8

default_credentials_http2 = [('admin', 'root', '63a9f0ea7bb98050796b649e85481845'), ('

classmethod get_challenge_moxahttp_1_0(*text: str*) → Optional[str]

Get authentication challenge from MoxaHTTP/1.0 routers.

Parameters **text** – HTML response provided by the router.

Returns Authentication challenge if found

classmethod get_challenge_moxahttp_2_2(*text: str*) → Optional[str]

Get authentication challenge from MoxaHTTP/2.2 routers.

Parameters **text** – HTML response provided by the router.

Returns Authentication challenge if found

possible_headers = {'Server': ['MoxaHttp', 'MoxaHttp/1.0', 'MoxaHttp/2.2']}

router_cls

alias of *MoxaRouter*

valid_login_text_moxahttp_1_0 = 'FRAME name=main src=main.htm'

valid_login_text_moxahttp_2_2 = 'FRAME name="main" src="main.htm"'

class aztarna.industrialrouters.scanner.**SierraRouter**
Bases: *aztarna.industrialrouters.scanner.BaseIndustrialRouter*

Class for holding Sierra Wireless manufactuer routers.

class aztarna.industrialrouters.scanner.**SierraWirelessScanner**
Bases: *aztarna.industrialrouters.scanner.BaseIndustrialRouterScanner*

Scanner class for Sierra Wireless routers.

classmethod check_default_password(*router: aztarna.industrialrouters.scanner.BaseIndustrialRouter*, *semaphore=<asyncio.locks.Semaphore object at 0x7f51195a5f60 [unlocked, value:1]>*)
Method for checking credentials on Sierra Wireless Routers.

Parameters

- **router** – *aztarna.industrialrouters.scanner.BaseIndustrialRouter* router to check.
- **semaphore** – Asyncio semaphore for limiting concurrency level.

default_credentials = [('sconsole', '12345'), ('', 'admin'), ('', 'swiadmin'), ('scons

failed_message = 'Invalid UserName / Password'

possible_headers = {'Server': ['Sierra Wireless Inc, Embedded Server']}

router_cls
alias of *SierraRouter*

class aztarna.industrialrouters.scanner.**WestermoRouter**
Bases: *aztarna.industrialrouters.scanner.BaseIndustrialRouter*

Class for holding Westermo Manufacturer routers.

class aztarna.industrialrouters.scanner.**WestermoScanner**
Bases: *aztarna.industrialrouters.scanner.BaseIndustrialRouterScanner*

Scanner class for Westermo routers.

default_credentials = [('admin', 'westermo')]

possible_headers = {'Server': ['Westermo', 'EDW']}

router_cls
alias of *WestermoRouter*

Module contents

3.3.1.2 Submodules

3.3.1.3 aztarna.cmd module

aztarna.cmd.**main**()
Main method

3.3.1.4 aztarna.commons module

class aztarna.commons.**BaseRobotHost**
Bases: object

A base class for different type of Robot hosts

class aztarna.commons.**RobotAdapter**(*ports=[80], extended=False*)
Bases: object

BaseScanner class, an abstraction for different type scans

load_from_file(*filename*)
Load a range of ipv4 addresses to scan and add them The :class:BaseScanner host_list attribute :param filename: name of the input file

load_range(*net_range*)
Transform ipv4 address strings to pythons ipaddress library type objects for scanning purposes :param net_range: A range of string type IPv4 addresses

print_results()

rate

scan()

scan_pipe_main()

static stream_as_generator(*loop*, *stream*)

write_to_file(*out_file*)

3.3.1.5 aztarna.helpers module

class aztarna.helpers.**HelpersLINQ**
Bases: object

A helper class for emulating .NET useful methods.

static distinct(*sequence*)

class aztarna.helpers.**HelpersNetWorking**
Bases: object

A helper class that checks networking related data

static ping(*host*)
A method that replicates the command line ping utility.

Parameters **host** – Host to ping to

Returns A boolean type that means if the ping reaches the destination or not

class aztarna.helpers.**PortScanner**
Bases: object

A base class that provides methods to check correct por scans.

static check_port(*ip*, *port*)
Checks if a certain port is open :param ip: The host's IP address :param port: The host's port :return: The scanned port if open, else None

static check_port_sem(*sem*, *ip*, *port*)
Calls to :method:check_port with a Semaphore to avoid too many open connections.

Parameters

- **sem** –
- **ip** –
- **port** –

Returns

static scan_host(*address*, *start_port*, *end_port*, *max_conns=400*)

Parameters

- **address** – IPv4 address to scan
- **start_port** – First port value to scan
- **end_port** – Last port value to scan
- **max_conns** – Maximum simultaneous number of connections

Returns

3.3.1.6 Module contents

3.4 Indices and tables

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

INDEX

R

S

T

U

V

W