tBB Documentation

Release 1.0.0

Daniele Parmeggiani

Sep 27, 2017

Contents

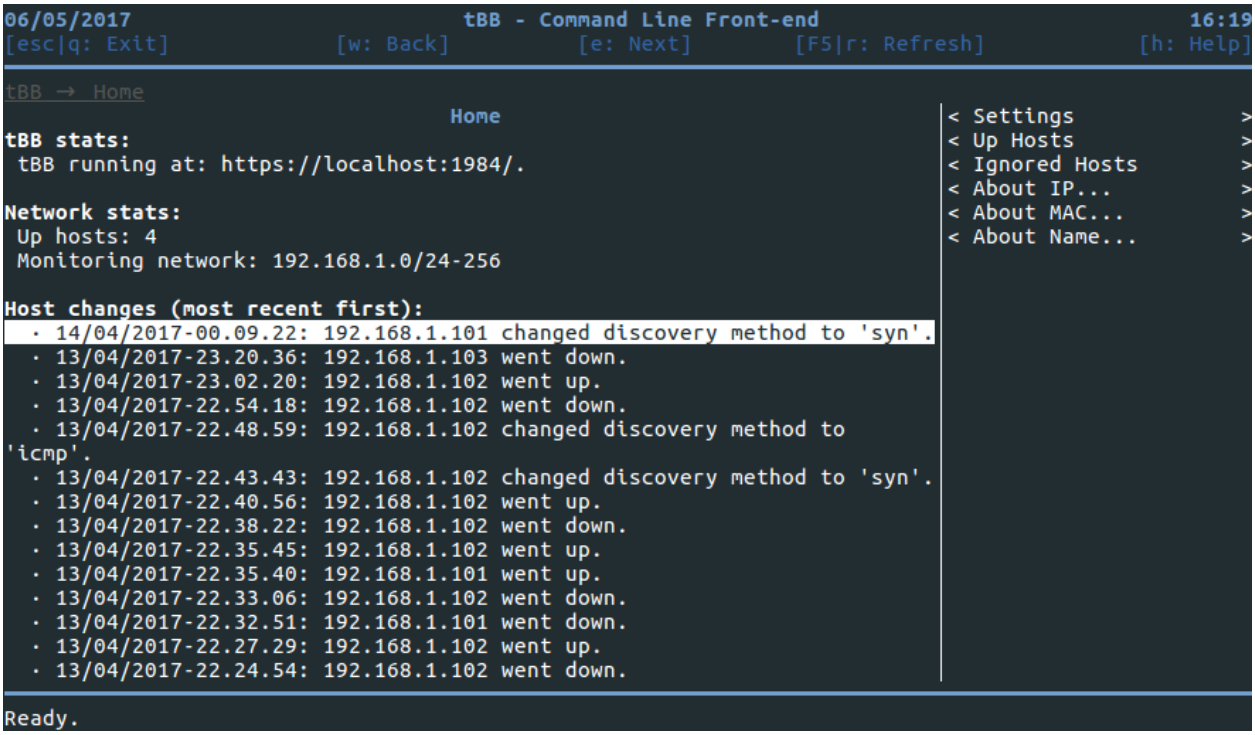
1 Getting Started **3**
- 1.1 0. Are you allowed to do this? . . . 3
- 1.2 1. Check Python is installed . . . 3
- 1.3 2. Download tBB . . . 3
- 1.4 3. Download dependencies . . . 4
- 1.5 4. Set your password . . . 4
- 1.6 You're done! . . . 4
- 1.7 Uh oh! . . . 4

2 Configuring tBB **5**
- 2.1 How configuration works in tBB . . . 5
- 2.2 Network-specific configuration files naming conventions . . . 6
- 2.3 Examples . . . 6
- 2.4 Configuration fields . . . 7

3 Full tBB reference **13**
- 3.1 tBB.async_stdio module . . . 13
- 3.2 tBB.builtin_configuration module . . . 13
- 3.3 tBB.discoveries module . . . 13
- 3.4 tBB.frontends module . . . 14
- 3.5 tBB.main module . . . 16
- 3.6 tBB.net_elements module . . . 16
- 3.7 tBB.paths module . . . 18
- 3.8 tBB.serialization module . . . 18
- 3.9 tBB.settings module . . . 19
- 3.10 tBB.tracker module . . . 19
- 3.11 Module contents . . . 23

Python Module Index **25**

tBB is an open-source Intrusion Detection System written in Python. It is able of keeping track of connections, disconnections and other changes in the network it monitors.



CHAPTER 1

Getting Started

Go through these 4 simple steps to install and configure tBB.

Up to this moment the only tested OS for tBB is Ubuntu, nevertheless you should be able to run tBB in many other POSIX environments.

0. Are you allowed to do this?

Before starting at all you should consider that tBB installation *and execution* require root privileges. If you're not able to supply such privileges for your environment, please contact your system administrator.

1. Check Python is installed

tBB requires Python 3 to run. Check that it is installed before proceeding:

```
sudo apt install python3 python3-pip git
python3
```

If a shell like this >>> pops up, Python had been correctly installed. Close the shell and proceed with the installation.

2. Download tBB

Fetch the latest version of tBB from GitHub:

```
git clone https://github.com/dpdani/tBB.git
cd tBB
```

3. Download dependencies

These dependencies are required to run tBB:

```
sudo python3 -m pip install -r requirements.txt
sudo apt install nmap iputils-arping
```

4. Set your password

First let tBB create the folders it needs to get working:

```
sudo ./run 192.168.0.0/24
```

Note: The network you specify with this command is not relevant. We just need to launch tBB so that it installs its required configuration folders.

Now tBB should complain that it didn't find a password file, let's go and create it.

Front-ends will ask for this password when you connect to a running tBB server:

```
cd ~/.tBB
sudo nano tBB_access_password
*enter password & save*
sudo chmod 600 tBB_access_password
```

You're done!

Now you can start using tBB. Go back to the installation folder and start tBB:

```
sudo ./run ...  # whatever network you mean to monitor
```

You can also configure tBB to fit your specific needs. Please refer to the configuration section of this manual.

Uh oh!

Did something go wrong? You might find the answer to your questions in this section, if not, feel free to open an issue on GitHub.

CHAPTER 2

Configuring tBB

If you wish to configure tBB settings, you can use the configuration files tBB is instructed to look for. These files are always located in the `~/.tBB/configs/` folder. Please check where this folder is located before continuing reading.

Configuration files are in the standard JSON format. If you're unfamiliar with such format, please refer to RFC 7159, or other documentation on the Internet.

Note: You're free to change settings in these files at any time, but if you want your changes to take effect you'll to have to restart tBB.

How configuration works in tBB

Configuration in tBB works like the way cascade sheets work. There are 3 levels of configuration:

```
BUILTIN configuration
        ^
        |
        |
DEFAULT configuration
        ^
        |
        |
SPECIFIC configuration
```

The first level is the *built-in* configuration that comes with tBB: this cannot be changed and acts as a fallback for tBB.

The second level is the *default* configuration which is stored in `~/.tBB/configs/config_default.json`: you can set this and will be applied to every network your tBB installation will monitor.

The third level is the *network-specific* configuration which is stored in `~/.tBB/configs/config_{NETWORK}.json`: this configuration will only be applied if tBB is asked to monitor the `{NETWORK}` network. On network-specific filename syntax please refer to the below section.

Each new level overrides the configurations of the previous one.

Note: There are many configuration files you may want to add to your tBB installation, but tBB is capable of running without any configuration file, falling back to the built-in configuration.

Network-specific configuration files naming conventions

Specific configuration files naming conventions follow the scans naming conventions, and is as follows:

```
~/.tBB/configs/config_{NETWORK IP}\{NETMASK (cidr)}-{NETWORK LENGTH}.json
```

This rigid naming conventions allow tBB to use the correct configuration file for every network you may want to monitor.

For instance, if you want to create a configuration file for network `192.168.100.0/24` you're going to need to create a file named `config_192.168.100.0\24-256.json`.

Note: Please note the backslash `\` replacing the forward slash `/` (forward slash is invalid for the Unix file name conventions). Also note the given network length in the filename after the dash `-` sign.

Examples

These examples may be applied to both a default or a network-specific configuration file.

Do not sleep between checks:

```
{
    "monitoring": {
        "time_between_checks": "00:00",
        "maximum_seconds_randomly_added": 0
    }
}
```

Change time format and enable syslog:

```
{
    "logging": {
        "default_time_format": "%m-%d-%Y, %H.%M.%S"  # <-- time format
        "handlers": {
            "syslog": {                  # <-- configure syslog
                "address": {             #
                    "ip": "192.168.0.1",  #
                    "port": 666          #
                },
                "socktype": "DATAGRAM"
            },
            "enable": ["console", "file", "syslog"]  # <-- enable logging handlers␣
↪here
    }
}
```

Bind tBB server to a non-loopback interface and disable SSL:

```
{
    "frontends": {
        "host": "192.168.100.15",
        "ssl": {
            "enable": false
        }
    }
}
```

Configuration fields

The various fields configurable in tBB are divided in logical sections, so that they can be easier to understand and recognize.

What follows are tables of the available configurable fields.

Default values are the values specified in the built-in configuration.

Root-level

Field name	Description	Example values	Default value
`monitoring`	Section dedicated to the monitoring machinery	*is section, see below*	...
`frontends`	Section dedicated to frontends communication	*is section, see below*	...
`serialization`	Section dedicated to scans storage handling	*is section, see below*	...
`logging`	Section dedicated to the logging facilities	*is section, see below*	...

monitoring

Field name	Description	Example values	Default value
discoveries	Section dedicated to the discovery methods	*is section, see below*	...
least_record_update	Maximum amount of time for which tBB will not re-perform a complete scan on startup (format: minutes:seconds).	'00:00'	'30:00'
enable_notifiers	Tell notifiers about detected changes.	false	true
time_between_checks	Amount of time to wait before proceeding to check the next host[1] (format: minutes:seconds).	'00:00'	'00:02'
maximum_seconds_randomly_added	Maximum amount of time to add randomly[2] to time_between_checks (in seconds). Must be a positive integer.	10	2
auto_ignore_broadcasts	Enable/disable automatic broadcasts ignore. If enabled, when a broadcast is detected during a scan, it will be ignored in the next ones.	false	true
hosts	Number of hosts sub-networks will be divided into. Must be a valid network length (aka, power of 2).	64	16
ignore	List of IPs to ignore.	['192.168.100.1']	[]
ignore_mac	List of MACs to ignore.	['00:...:00'] []	
ignore_name	List of host names to ignore.	['donald.duck']	[]

monitoring → discoveries

Field name	Description	Example values	Default value
arp	Section dedicated to the ARP discovery method	*is section, see below*	...
icmp	Section dedicated to the ICMP discovery method	*is section, see below*	...
syn	Section dedicated to the SYN discovery method	*is section, see below*	...

monitoring → discoveries → arp

Field name	Description	Example values	Default value
count	Number of ARP broadcasts to emit.	1	3
timeout	Maximum amount of time in which to wait for a response (in seconds). Must be a positive integer. A higher value in this field represent a more reliable check, but also a slower one.		
quit_on_first	Stop listening for responses at first response.	false	true

monitoring → discoveries → icmp

[1] Determined by Tracker.highest_priority_host.
[2] See Tracker.keep_network_tracked for further details.

Field name	Description	Example values	Default value
`count`	Number of requests to send. If `flood` is enabled, it represents the number of responses to receive before returning.	`4`	`1`
`timeout`	Maximum amount of time in which to wait for a response (in seconds). Must be a positive integer. A higher value in this field represent a more reliable check, but also a slower one.	`1`	`4`
`flood`	Enable/disable flood ping mode.	`false`	`true`
`enable`	Enable/disable discovery method.	`false`	`true`

monitoring → discoveries → syn

Field name	Description	Example values	Default value
`ports`	Ports to check. Must be of string type.	`'80'`	`'2'`
`timeout`	Maximum amount of time in which to wait for a response (in seconds). Must be a positive integer. A higher value in this field represent a more reliable check, but also a slower one.	`1`	`4`
`enable`	Enable/disable discovery method.	`false`	`true`

frontends

Field name	Description	Example values	Default value
`host`	IP address for the frontends socket.	`192.168.1.10`	`localhost`
`port`	Port number for the frontends socket.	`2000`	`1984`
`maximum_port_lookup`	Maximum number of times tBB will look for the next available port if the previous one is busy.	`1`	`20`
`ssl`	Section dedicated to securing communications with SSL/TLS.	*is section, see below*	...

frontends → ssl

Field name	Description	Example values	Default value
`enable`	Enable/disable SSL encryption. tBB will fall back to HTTP communication.	`false`	`true`
`check_hostname`	Enable/disable certificate checking, must agree with frontends on this field for correct SSL handshake.	`true`	`false`

serialization

Field name	Description	Example values	Default value
`indent`	Number of spaces with which indent the scan storages (json.dump(indent) ext. docs).	`0`	`4`
`do_sort`	Enable/disable sorting of scan storages (json.dump(sort_keys) ext. docs).	`false`	`true`

logging

Field name	Description	Example values	Default value
`default_time_format`	Default format for datetimes in log files.[3]	`'%d-%m-%Y %H.%M.%S'`	`'%Y-%m-%d %H.%M.%S'`
`level`	Minimum logging level. One of `DEBUG`, `INFO`, `WARNING`, `ERROR`, `CRITICAL`.	`DEBUG`	`INFO`
`formatters`	Section dedicated to loggers formatters.	*is section, see below*	...
`handlers`	Section dedicated to loggers handlers.	*is section, see below*	...

logging → formatters

Field name	Description	Example values	Default value
`complete`	Section dedicated to the complete formatter.	*is section, see below*	...
`brief`	Section dedicated to the brief formatter.	*is section, see below*	...
`syslog`	Section dedicated to the syslog formatter.	*is section, see below*	...
`custom_1`	Section dedicated to the custom_1 formatter.	*is section, see below*	...
`custom_2`	Section dedicated to the custom_2 formatter.	*is section, see below*	...
`custom_3`	Section dedicated to the custom_3 formatter.	*is section, see below*	...

logging → formatters → complete/brief/syslog/custom_1/custom_2/custom_3

All formatters share the same configuration skeleton.

Field name	Description	Example values	Default value
`format`	String to format logging upon.[4]	...	...
`datefmt`	String to format datetimes upon.[3] Macro `{default_time_format}` points to `logging → default_time_format`.	`'%d-%m-%Y %H.%M.%S'`	`{default_time_format}`

logging → handlers

Field name	Description	Example values	Default value
`console`	Section dedicated to the complete handler.	*is section, see below*	...
`syslog`	Section dedicated to the syslog handler.	*is section, see below*	...
`file`	Section dedicated to the file handler.	*is section, see below*	...
`enable`	List of enabled logging handlers: handlers found in this list will be triggered when logging.	`[]`	`['console', 'file']`

logging → handlers → console

Field name	Description	Example values	Default value
`level`	Minimum logging level for this handler. One of `DEBUG`, `INFO`, `WARNING`, `ERROR`, `CRITICAL`.	`DEBUG`	`INFO`
`formatter`	Formatter chosen for this handler, as defined in `logging → formatters`.	`custom_1`	`brief`

logging → handlers → syslog

[3] Python logging library date format documentation https://docs.python.org/3/library/logging.html#logging.Formatter.formatTime
[4] Python logging library log records documentation https://docs.python.org/3/library/logging.html#logrecord-attributes

Field name	Description	Example values	Default value
`level`	Minimum logging level for this handler. One of `DEBUG`, `INFO`, `WARNING`, `ERROR`, `CRITICAL`.	`DEBUG`	`INFO`
`formatter`	Formatter chosen for this handler, as defined in `logging` → `formatters`.	`custom_2`	`syslog`
`address`	Section dedicated to the syslog host address.	*is section, see below*	*...*
`socktype`	ISO/OSI level 4 protocol chosen by the syslog server. One of UDP: `DATAGRAM`, TCP: `STREAM`.	`STREAM`	`DATAGRAM`

logging → handlers → syslog → address

Field name	Description	Example values	Default value
`ip`	Syslog host IP.	`'192.168.100.20'`	`''`
`port`	Syslog server port.	`514`	`''`

logging → handlers → file

Field name	Description	Example values	Default value
`level`	Minimum logging level for this handler. One of `DEBUG`, `INFO`, `WARNING`, `ERROR`, `CRITICAL`.	`WARNING`	`DEBUG`
`formatter`	Formatter chosen for this handler, as defined in `logging` → `formatters`.	`syslog`	`complete`
`max_bytes`	Maximum size for log file (in bytes).	`1000`	`10000000` (`10 MB`)
`backup_count`	Maximum number of log files (of at most `max_bytes` size) to keep.	`1`	`4`
`filename`	Log file name.	`definetelynotalogfile.log`	`tBB.log`

CHAPTER 3

Full tBB reference

This section of tBB documentation is dedicated to documenting the internals of tBB.

tBB.async_stdio module

tBB.async_stdio.**async_input**(*message*)

tBB.async_stdio.**stdio**(*loop=None*)

tBB.builtin_configuration module

tBB's builtin configuration.

This is what tBB will fall back to if the user doesn't specify differently.

tBB.discoveries module

Discovery methods implementations.

Library `asyncio.subprocess` is used to implement these methods.

class tBB.discoveries.**ARPDiscovery**(*count*, *interface*, *timeout*, *quit_on_first=True*, *enabled=True*)
Bases: *tBB.discoveries.DiscoveryMethod*

ARP discovery method. Uses system's `arping` to perform requests.

exception tBB.discoveries.**ARPParsingException**(*ip*, *string*)
Bases: *tBB.discoveries.ParsingException*

class tBB.discoveries.**DiscoveryMethod**(*short_name*, *enabled=True*)
Bases: object

Base-abstract class for all discovery methods.

run(*ip*)
Wrapper for `DiscoveryMethod._run`. Does type checking. If `ip` is a string this function will create an IPElement and pass it to `self._run`.

Parameters **ip** – ip to run for.

Returns whatever is returned by `self._run`

run_multiple(*ips*)
Runs discovery for many ips.

Parameters **ips** – ips to run for.

Returns dict[ip, result]

class tBB.discoveries.**HostNameDiscovery**
Bases: *tBB.discoveries.DiscoveryMethod*

exception tBB.discoveries.**HostNameParsingException**(*ip*, *string*)
Bases: *tBB.discoveries.ParsingException*

class tBB.discoveries.**ICMPDiscovery**(*count*, *timeout*, *flood=False*, *enabled=True*)
Bases: *tBB.discoveries.DiscoveryMethod*

ICMP discovery method. Uses system's `ping` to perform requests.

exception tBB.discoveries.**ICMPParsingException**(*ip*, *string*)
Bases: *tBB.discoveries.ParsingException*

exception tBB.discoveries.**ParsingException**(*ip*, *method*, *string*)
Bases: Exception

exception tBB.discoveries.**PingedBroadcast**(*ip*)
Bases: Exception

class tBB.discoveries.**SYNDiscovery**(*ports*, *timeout*, *enabled=True*)
Bases: *tBB.discoveries.DiscoveryMethod*

SYN discovery method. Uses system's `nc` to perform requests.

exception tBB.discoveries.**SYNParsingException**(*ip*, *string*)
Bases: *tBB.discoveries.ParsingException*

tBB.discoveries.**shell**(*command*)
Uses `asyncio` 's subprocess internally.

Parameters **command** (*str*) – command to execute

Returns stdout of command

Return type *list*

tBB.frontends module

This module helps handling tBB front-ends.

class tBB.frontends.**FrontendsHandler**(*tracker*, *password*, *config*, *loop=None*)
Bases: object

bind_requests()

check_datetime(*input_*, *accept_now=True*)

check_ip(*ip*, *check_in_tracker=True*)

check_mac(*mac*, *check_in_tracker=True*)

check_name(*name*, *check_in_tracker=True*)

check_request_input(*input_*, *expected*, *password=True*)

close()

static determine_port(*host*, *starting_port*, *maximum_port_lookup*)
This method searches the first port available after (and including) starting_port. To limit this method from looking up to port 65535, use the maximum_port_lookup argument.

get_priority(*request*)

ignore(*request*)

ignore_mac(*request*)

ignore_name(*request*)

ignored_ips(*request*)

ignored_macs(*request*)

ignored_names(*request*)

ip_host_changes(*request*)

ip_info(*request*)

is_ignored(*request*)

is_mac_ignored(*request*)

is_name_ignored(*request*)

mac_host_changes(*request*)

mac_info(*request*)

name_host_changes(*request*)

name_info(*request*)

set_priority(*request*)

settings_get(*request*)

settings_set(*request*)

start()

stats(*request*)

status(*request*)

test(*request*)

up_ip_hosts(*request*)

up_mac_hosts(*request*)

up_name_hosts(*request*)

tBB.main module

tBB main entry point.

tBB.main.**configure_logging**(*settings*)

tBB.main.**developer_cli**(*globals_*, *locals_*)

tBB.main.**main**(*args*)

tBB.main.**original_stdout**()

tBB.main.**user_quit**(*tasks*)

tBB.net_elements module

Network elements representations for Python.

class tBB.net_elements.**IPElement**(**args*, ***kwargs*)
Bases: object

IP node object representation for Python. This object responds to the following interfaces:

- addition (and subtraction)
- equality (and inequality)
- hashable

as_int()

as_string()

ip

is_broadcast()
Returns True if self.ip is a broadcast IP in accordance to self.mask.

Returns bool

is_network()
Returns True if self.ip is a network IP in accordance to self.mask.

Returns bool

mask

static parse_ip(*string*)
Parses a string and checks if it is a valid IP.

Returns string, list

static parse_ip_with_mask(*string*)
Parses a string and returns the network IP and mask.

Returns (ip: str, mask: int)

static parse_mask(*string*)
Parses a string and checks if it is a valid mask.

Returns int

class tBB.net_elements.**IPHost**(*ip*)
Bases: object

add_to_discovery_history(*entry*)

add_to_is_up_history(*entry*)

add_to_mac_history(*entry*)

add_to_name_history(*entry*)

ago

ip

is_up

last_discovery_method

mac

name

print_histories()

second_last_mac

second_last_name

update(*mac*, *method*, *is_up*, *name*)

class tBB.net_elements.**MACElement**(*mac*)
Bases: object

class tBB.net_elements.**MACHost**(*mac*)
Bases: object

add_to_history(*entry*)

ago

ip

print_histories()

update(*ip*)

update_ip_disconnected(*ip*)

class tBB.net_elements.**NameHost**(*name*)
Bases: object

add_to_history(*entry*)

ago

ip

print_histories()

update(*ip*)

update_ip_disconnected(*ip*)

class tBB.net_elements.**Network**(**args*, ***kwargs*)
Bases: *tBB.net_elements.IPElement*

IP network object representation for Python. This object responds to the following interfaces (inherited from IPElement):

- addition (and subtraction)
- equality (and inequality)

•hashable

In addition, also responds to the following:

- iterable
- sliceable

as_string()

broadcast()

ip

last_host()

next()

tBB.net_elements.**netmask_from_netlength**(*hosts*)

tBB.paths module

This module contains various utilities for finding correct paths for tBB.

tBB.paths.**check_required_paths**()

This function checks for paths to be present on the filesystem. It checks for:

•~/.tBB/

•~/.tBB/scans/

•~/.tBB/certs/

•~/.tBB/configs/

tBB.paths.**update_paths**()

Updates the following global variables to paths:

- executable
- root
- configs
- scans
- certs

tBB.serialization module

Serialization module for tBB.

class tBB.serialization.**Serializer**(*network=None*, *path=None*, *track=None*, *config=None*, *sessions=[]*)

Bases: object

keep_saving(*frequency*)

load()

save()

tBB.serialization.**path_for_network**(*network*, *saving_path='/home/docs/.tBB/scans'*, *suffix='.tbbscan'*)

tBB.settings module

This module takes care of representing and handling settings throughout tBB.

exception tBB.settings.**ConversionException**(*value*, *value_type*)
Bases: Exception

exception tBB.settings.**InconsistentSettingTypeException**(*setting_path*, *should_be*, *got*)
Bases: Exception

class tBB.settings.**Settings**(*tree*)
Bases: object

static parse(*json_data*, *name='toplevel'*)

update(*new_tree*, *scope=''*)

class tBB.settings.**SettingsItem**(*name*, *value_type*)
Bases: object

convert()

static convert_to_settings_item(*value*)

static convert_to_timedelta(*value*)

class tBB.settings.**SettingsTypes**
Bases: enum.Enum

An enumeration.

boolean = 2

integer = 1

list = 5

settings_item = 4

string = 0

timedelta = 3

unknown = -1

exception tBB.settings.**UndefinedValueException**
Bases: Exception

exception tBB.settings.**UnknownSettingException**(*setting_path*)
Bases: Exception

tBB.tracker module

Tracker implementation.

class tBB.tracker.**Tracker**(*network*)
Bases: object

changes (*hosts*, *from_*, *to*, *json_compatible=False*)

This function returns changes occurred to given hosts within the given time period. If argument json_compatible evaluates to True, in the returned dict there will be no objects as defined in net_elements. Instead they will be converted into builtin types as follows:

- `IPElement("192.168.0.0/24") -> "192.168.0.0" # str`
- `MACElement("a0:ff:e4:bc:66:70") -> "a0:ff:e4:bc:66:70" # str`
- `datetime() -> datetime().timestamp() # float`

The returned dict will be in the following form:

```
{
    IPElement("...") : {
        'discovery_history': {
            datetime(...): 'icmp',  # or 'syn'
            ...
        },
        'is_up_history': {
            datetime(...): True,  # or False
            ...
        },
        'mac_history': {
            datetime(...): MACElement("..."),
            ...
        }
    },
    MACElement("..."): {
        'history': {
            datetime(...): [IPElement("..."), ...],
            ...
        }
        'is_up_history': {
          datetime(...): True,  # or False
          ...
        }
    }
    IPElement("..."): {
        ...
    },
    ...
}
```

Since this function may do some heavy calculations and therefore block, it had been designed to be a coroutine, in order to prevent blocking. For filtering results to `IPHosts` only or `MACHosts` only, see `Tracker.ip_changes` and `Tracker.mac_changes`.

Parameters

- **hosts** – IPHost,MACHost[]
- **from** – datetime.datetime
- **to** – datetime.datetime
- **json_compatible** – bool

Returns dict

configure (*config*)

do_complete_network_scan()
Runs complete network scan. Similarly to `Track.do_partial_scan`, this does not use `self.highest_priority_host` internally; iterates over `self.network` instead.

do_partial_scan(*start*, *hosts*)
Runs partial scan of the network. Starting from argument start for so many hosts as defined in argument `hosts`. Similarly to `Track.do_complete_network_scan`, this does not use self.highest_priority_host internally; iterates over `self.network` instead.

Parameters

- **start** (*int*) – integer to add to `self.network` to get first ip to scan.
- **hosts** (*int*) – number of ips to scan.

Returns number of up hosts

Return type int

do_single_scan(*ip*)
Runs a scan to the specified ip. Uses discovery methods found in `self.discoveries`. You can enable/disable each one of them by setting `self.discoveries[x].enable` to whatever suits you. This function takes care of detecting whether the host changed its status and if so it calls `self.fire_notifiers`. If one discovery method results positive others won't be run. Returns whether or not the host was found to be up.

Note: in order to provide the mac address of the scanning host, ARP will be run even if it had been disabled, but it won't be tracked as the discovery method used when executed for this purpose.

Parameters **ip** (`IPElement()`) – ip to scan.

Return type bool

fire_notifiers(*ip*, *mac*, *name*, *method*, *is_up*, *ip_what*, *mac_what*, *name_what*)

highest_priority_host()
Returns the host that has the highest priority in this moment. The calculation is made so that there can be no hosts with the same priority. It takes in account per-host set priorities in self.priorities. The calculation is done as follows:

```
priority = host_priority + time_since_last_check|IP
```

As shown, the IP added at the end prevents two hosts from having the same priority. Seen how the calculation is performed, priorities set in self.priorities should consider that if, for instance, the priority for host A is set to 10, every call within 10 seconds since last scan will return host A.

Return type *IPElement*

ip_changes(*hosts*, *from_*, *to*, *json_compatible=False*)
Similar to `Tracker.changes`, but only iterates over `IPHosts`.

keep_network_tracked(*initial_sleep=0*)
Keeps the given network (`self.network`) tracked. Differently from `Tracker.do_complete_network_scan` and `Tracker.do_partial_scan`, this function doesn't iterate over `self.network` to keep it tracked. Instead it calls `self.highest_priority_host` each time it has to scan a new host. Again, differently from `Tracker.do_complete_network_scan` and `Tracker.do_partial_scan`, this function implements a sleeping mechanisms between scans

in order to reduce its weight on the network. The time it takes for sleeping can be set using `Track.time_between_scans` and `Track.maximum_seconds_randomly_added`, calculated as follows:

```
sleep = time_between_scans + randint(0, maximum_seconds_randomly_added)
```

randint being the `random.randint` function included in the Python's standard library.

mac_changes(*hosts*, *from_*, *to*, *json_compatible=False*)
Similar to `Tracker.changes`, but only iterates over `MACHosts`.

name_changes(*hosts*, *from_*, *to*, *json_compatible=False*)
Similar to `Tracker.changes`, but only iterates over `NameHosts`.

outer_status
Used to supply information to front-ends.

status
Used to supply information to front-ends.

up_hosts
Number of IP hosts currently up.

up_ip_hosts
IPHosts currently up. Result is a dictionary `{IPElement: IPHost}`.

Return type dict[*IPElement*, *IPHost*]

up_mac_hosts
`MACHost``s currently up. Result is a dictionary ``{MACElement: MACHost}`. Determining how a MACHost is up is a little bit different from an IPHost. Since a MACHost doesn't hold any up state, a MACHost is considered up when any of the IPHosts related to it (found in `MACHost.ip`) is up. Therefore even if only one of the (possibly) many IPHosts is up, the MACHost is considered up.

Return type dict[MACElement: MACHost]

up_name_hosts
Similar to `Tracker.up_mac_hosts`.

class `tBB.tracker.`**TrackersHandler**(*network*, *hosts=16*)
Bases: `object`

This is capable of handling different `Tracker` instances at the same time. For methods and attributes documentation you may refer to `Tracker`'s documentation, since this class mimics most of its behaviour. Please, note that this is not a subclass of `Tracker`, though. In most cases `Tracker`'s attributes are mapped to properties in order to provide the attributes of all `Tracker` s this object is currently handling. Usually, setting one of these properties reflects the change to all `Tracker` s objects currently handled.

arp

auto_ignore_broadcasts

changes(*hosts*, *from_*, *to*, *json_compatible=False*)

configure(*config*)

discoveries

do_complete_network_scan()

force_notify

ignore

ignore_mac

ignore_name

ip_changes(*hosts*, *from_*, *to*, *json_compatible=False*)

ip_hosts

keep_network_tracked(*initial_sleep=False*)

mac_changes(*hosts*, *from_*, *to*, *json_compatible=False*)

maximum_seconds_randomly_added

name_discovery

outer_status

priorities

serializer

status

time_between_checks

up_hosts

up_ip_hosts

up_mac_hosts

up_name_hosts

warn_parsing_exception

Module contents

tBB - The Big Brother.

An open-source Intrusion Detection System written in Python: keeps track of connections, disconnections and changes in the specified network.

For further information open `tBB/docs/`.

Python Module Index

t

tBB, 23
tBB.async_stdio, 13
tBB.builtin_configuration, 13
tBB.discoveries, 13
tBB.frontends, 14
tBB.main, 16
tBB.net_elements, 16
tBB.paths, 18
tBB.serialization, 18
tBB.settings, 19
tBB.tracker, 19

Index

A

add_to_discovery_history() (tBB.net_elements.IPHost method), 16
add_to_history() (tBB.net_elements.MACHost method), 17
add_to_history() (tBB.net_elements.NameHost method), 17
add_to_is_up_history() (tBB.net_elements.IPHost method), 17
add_to_mac_history() (tBB.net_elements.IPHost method), 17
add_to_name_history() (tBB.net_elements.IPHost method), 17
ago (tBB.net_elements.IPHost attribute), 17
ago (tBB.net_elements.MACHost attribute), 17
ago (tBB.net_elements.NameHost attribute), 17
arp (tBB.tracker.TrackersHandler attribute), 22
ARPDiscovery (class in tBB.discoveries), 13
ARPParsingException, 13
as_int() (tBB.net_elements.IPElement method), 16
as_string() (tBB.net_elements.IPElement method), 16
as_string() (tBB.net_elements.Network method), 18
async_input() (in module tBB.async_stdio), 13
auto_ignore_broadcasts (tBB.tracker.TrackersHandler attribute), 22

B

bind_requests() (tBB.frontends.FrontendsHandler method), 14
boolean (tBB.settings.SettingsTypes attribute), 19
broadcast() (tBB.net_elements.Network method), 18

C

changes() (tBB.tracker.Tracker method), 19
changes() (tBB.tracker.TrackersHandler method), 22
check_datetime() (tBB.frontends.FrontendsHandler method), 14
check_ip() (tBB.frontends.FrontendsHandler method), 14
check_mac() (tBB.frontends.FrontendsHandler method), 15
check_name() (tBB.frontends.FrontendsHandler method), 15
check_request_input() (tBB.frontends.FrontendsHandler method), 15
check_required_paths() (in module tBB.paths), 18
close() (tBB.frontends.FrontendsHandler method), 15
configure() (tBB.tracker.Tracker method), 20
configure() (tBB.tracker.TrackersHandler method), 22
configure_logging() (in module tBB.main), 16
ConversionException, 19
convert() (tBB.settings.SettingsItem method), 19
convert_to_settings_item() (tBB.settings.SettingsItem static method), 19
convert_to_timedelta() (tBB.settings.SettingsItem static method), 19

D

determine_port() (tBB.frontends.FrontendsHandler static method), 15
developer_cli() (in module tBB.main), 16
discoveries (tBB.tracker.TrackersHandler attribute), 22
DiscoveryMethod (class in tBB.discoveries), 13
do_complete_network_scan() (tBB.tracker.Tracker method), 20
do_complete_network_scan() (tBB.tracker.TrackersHandler method), 22
do_partial_scan() (tBB.tracker.Tracker method), 21
do_single_scan() (tBB.tracker.Tracker method), 21

F

fire_notifiers() (tBB.tracker.Tracker method), 21
force_notify (tBB.tracker.TrackersHandler attribute), 22
FrontendsHandler (class in tBB.frontends), 14

G

get_priority() (tBB.frontends.FrontendsHandler method), 15

H

highest_priority_host() (tBB.tracker.Tracker method), 21
HostNameDiscovery (class in tBB.discoveries), 14
HostNameParsingException, 14

I

ICMPDiscovery (class in tBB.discoveries), 14
ICMPParsingException, 14
ignore (tBB.tracker.TrackersHandler attribute), 22
ignore() (tBB.frontends.FrontendsHandler method), 15
ignore_mac (tBB.tracker.TrackersHandler attribute), 22
ignore_mac() (tBB.frontends.FrontendsHandler method), 15
ignore_name (tBB.tracker.TrackersHandler attribute), 23
ignore_name() (tBB.frontends.FrontendsHandler method), 15
ignored_ips() (tBB.frontends.FrontendsHandler method), 15
ignored_macs() (tBB.frontends.FrontendsHandler method), 15
ignored_names() (tBB.frontends.FrontendsHandler method), 15
InconsistentSettingTypeException, 19
integer (tBB.settings.SettingsTypes attribute), 19
ip (tBB.net_elements.IPElement attribute), 16
ip (tBB.net_elements.IPHost attribute), 17
ip (tBB.net_elements.MACHost attribute), 17
ip (tBB.net_elements.NameHost attribute), 17
ip (tBB.net_elements.Network attribute), 18
ip_changes() (tBB.tracker.Tracker method), 21
ip_changes() (tBB.tracker.TrackersHandler method), 23
ip_host_changes() (tBB.frontends.FrontendsHandler method), 15
ip_hosts (tBB.tracker.TrackersHandler attribute), 23
ip_info() (tBB.frontends.FrontendsHandler method), 15
IPElement (class in tBB.net_elements), 16
IPHost (class in tBB.net_elements), 16
is_broadcast() (tBB.net_elements.IPElement method), 16
is_ignored() (tBB.frontends.FrontendsHandler method), 15
is_mac_ignored() (tBB.frontends.FrontendsHandler method), 15
is_name_ignored() (tBB.frontends.FrontendsHandler method), 15
is_network() (tBB.net_elements.IPElement method), 16
is_up (tBB.net_elements.IPHost attribute), 17

K

keep_network_tracked() (tBB.tracker.Tracker method), 21
keep_network_tracked() (tBB.tracker.TrackersHandler method), 23
keep_saving() (tBB.serialization.Serializer method), 18

L

last_discovery_method (tBB.net_elements.IPHost attribute), 17
last_host() (tBB.net_elements.Network method), 18
list (tBB.settings.SettingsTypes attribute), 19
load() (tBB.serialization.Serializer method), 18

M

mac (tBB.net_elements.IPHost attribute), 17
mac_changes() (tBB.tracker.Tracker method), 22
mac_changes() (tBB.tracker.TrackersHandler method), 23
mac_host_changes() (tBB.frontends.FrontendsHandler method), 15
mac_info() (tBB.frontends.FrontendsHandler method), 15
MACElement (class in tBB.net_elements), 17
MACHost (class in tBB.net_elements), 17
main() (in module tBB.main), 16
mask (tBB.net_elements.IPElement attribute), 16
maximum_seconds_randomly_added (tBB.tracker.TrackersHandler attribute), 23

N

name (tBB.net_elements.IPHost attribute), 17
name_changes() (tBB.tracker.Tracker method), 22
name_discovery (tBB.tracker.TrackersHandler attribute), 23
name_host_changes() (tBB.frontends.FrontendsHandler method), 15
name_info() (tBB.frontends.FrontendsHandler method), 15
NameHost (class in tBB.net_elements), 17
netmask_from_netlength() (in module tBB.net_elements), 18
Network (class in tBB.net_elements), 17
next() (tBB.net_elements.Network method), 18

O

original_stdout() (in module tBB.main), 16
outer_status (tBB.tracker.Tracker attribute), 22
outer_status (tBB.tracker.TrackersHandler attribute), 23

P

parse() (tBB.settings.Settings static method), 19
parse_ip() (tBB.net_elements.IPElement static method), 16
parse_ip_with_mask() (tBB.net_elements.IPElement static method), 16
parse_mask() (tBB.net_elements.IPElement static method), 16
ParsingException, 14

path_for_network() (in module tBB.serialization), 18
PingedBroadcast, 14
print_histories() (tBB.net_elements.IPHost method), 17
print_histories() (tBB.net_elements.MACHost method), 17
print_histories() (tBB.net_elements.NameHost method), 17
priorities (tBB.tracker.TrackersHandler attribute), 23

R

run() (tBB.discoveries.DiscoveryMethod method), 14
run_multiple() (tBB.discoveries.DiscoveryMethod method), 14

S

save() (tBB.serialization.Serializer method), 18
second_last_mac (tBB.net_elements.IPHost attribute), 17
second_last_name (tBB.net_elements.IPHost attribute), 17
Serializer (class in tBB.serialization), 18
serializer (tBB.tracker.TrackersHandler attribute), 23
set_priority() (tBB.frontends.FrontendsHandler method), 15
Settings (class in tBB.settings), 19
settings_get() (tBB.frontends.FrontendsHandler method), 15
settings_item (tBB.settings.SettingsTypes attribute), 19
settings_set() (tBB.frontends.FrontendsHandler method), 15
SettingsItem (class in tBB.settings), 19
SettingsTypes (class in tBB.settings), 19
shell() (in module tBB.discoveries), 14
start() (tBB.frontends.FrontendsHandler method), 15
stats() (tBB.frontends.FrontendsHandler method), 15
status (tBB.tracker.Tracker attribute), 22
status (tBB.tracker.TrackersHandler attribute), 23
status() (tBB.frontends.FrontendsHandler method), 15
stdio() (in module tBB.async_stdio), 13
string (tBB.settings.SettingsTypes attribute), 19
SYNDiscovery (class in tBB.discoveries), 14
SYNParsingException, 14

T

tBB (module), 23
tBB.async_stdio (module), 13
tBB.builtin_configuration (module), 13
tBB.discoveries (module), 13
tBB.frontends (module), 14
tBB.main (module), 16
tBB.net_elements (module), 16
tBB.paths (module), 18
tBB.serialization (module), 18
tBB.settings (module), 19
tBB.tracker (module), 19
test() (tBB.frontends.FrontendsHandler method), 15
time_between_checks (tBB.tracker.TrackersHandler attribute), 23
timedelta (tBB.settings.SettingsTypes attribute), 19
Tracker (class in tBB.tracker), 19
TrackersHandler (class in tBB.tracker), 22

U

UndefinedValueException, 19
unknown (tBB.settings.SettingsTypes attribute), 19
UnknownSettingException, 19
up_hosts (tBB.tracker.Tracker attribute), 22
up_hosts (tBB.tracker.TrackersHandler attribute), 23
up_ip_hosts (tBB.tracker.Tracker attribute), 22
up_ip_hosts (tBB.tracker.TrackersHandler attribute), 23
up_ip_hosts() (tBB.frontends.FrontendsHandler method), 15
up_mac_hosts (tBB.tracker.Tracker attribute), 22
up_mac_hosts (tBB.tracker.TrackersHandler attribute), 23
up_mac_hosts() (tBB.frontends.FrontendsHandler method), 15
up_name_hosts (tBB.tracker.Tracker attribute), 22
up_name_hosts (tBB.tracker.TrackersHandler attribute), 23
up_name_hosts() (tBB.frontends.FrontendsHandler method), 15
update() (tBB.net_elements.IPHost method), 17
update() (tBB.net_elements.MACHost method), 17
update() (tBB.net_elements.NameHost method), 17
update() (tBB.settings.Settings method), 19
update_ip_disconnected() (tBB.net_elements.MACHost method), 17
update_ip_disconnected() (tBB.net_elements.NameHost method), 17
update_paths() (in module tBB.paths), 18
user_quit() (in module tBB.main), 16

W

warn_parsing_exception (tBB.tracker.TrackersHandler attribute), 23