THOR Thunderstorm Manual

Nextron Systems GmbH

Apr 02, 2024

CONTENTS:

CHAPTER

ONE

REQUIREMENTS

1.1 Supported Operating Systems

- RHEL/CentOS 7 / 8
- SuSE SLES 15
- Ubuntu 18 / 20 LTS
- Debian 9 / 10

Since THOR also runs on Windows and macOS operating systems, THOR Thunderstorm can also be used on one of these platforms but without any support.

1.2 License Requirements

A valid service license is needed to use THOR Thunderstorm. Please see *Get a Service License* for more information.

1.3 Hardware Requirements

The hardware requirements highly depend on the number of samples per minute that the service has to process.

In cases in which only a few samples per minute have to be processed, even a dual core barebone system could be enough. However, in cases in which thousands of samples per minute should be processed, we recommend having a **high amount of CPU cores**, a decent amount of RAM and an SSD as disk for a faster processing of queued samples.

Component	Minimum	Recommended
CPU	2 CPU Cores	12+ CPU Cores
Memory	2 GB of RAM	16 GB of RAM
Disk Space (sample queue)	5 GB Hard Disk Drive	1 TB Solid State Drive

1.4 Network Connections

For a detailed and up to date list of our update and licensing servers, please visit https://www.nextron-systems.com/hosts/.

1.4.1 Web UI and Sample Submission

The THOR Thunderstorm service is listening on port 8080/tcp. This can be changed in the configuration to any other port. Additionally, you can use HTTPs for the service. Please see the chapter *Configuration*.

1.4.2 Update Server

Remote Server	Port
update1.nextron-systems.com	443/tcp
update2.nextron-systems.com	443/tcp

1.4.3 Installer

Remote Server	Port
portal.nextron-systems.com	443/tcp
cloud.nextron-systems.com	443/tcp

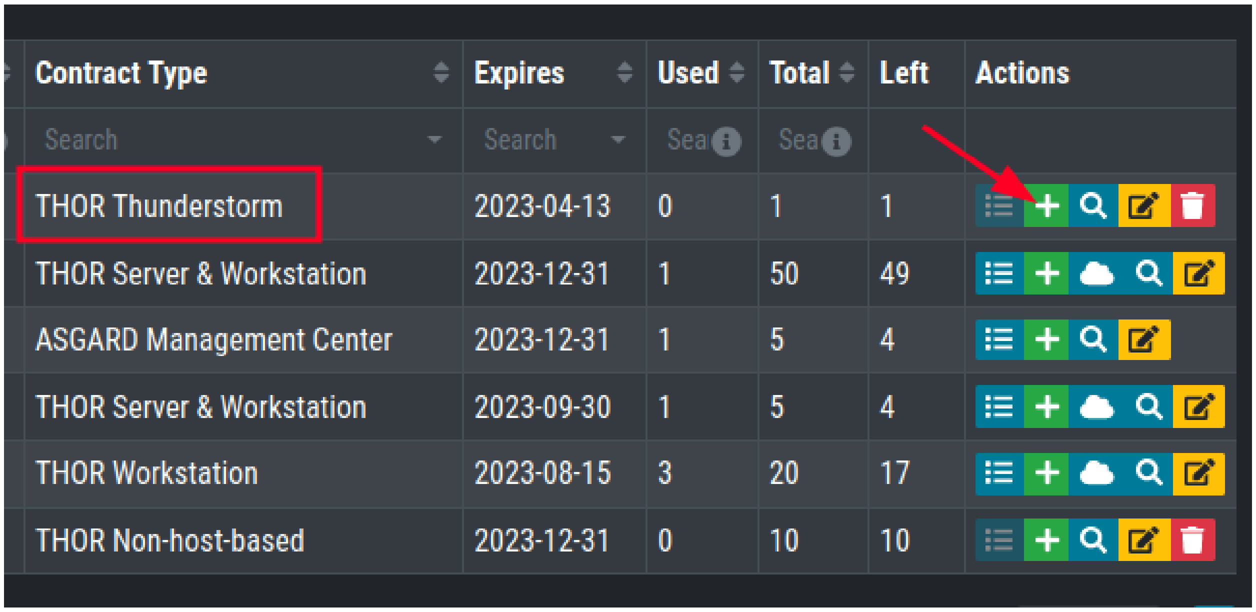
CHAPTER

TWO

INSTALL THUNDERSTORM SERVICE

2.1 Get a Service License

To run THOR in Thunderstorm service mode, you need a license of a special type named `THOR Thunderstorm` which allows this mode of operation.



Contract Type	Expires	Used	Total	Left	Actions
Search	Search	Sea	Sea		
THOR Thunderstorm	2023-04-13	0	1	1	
THOR Server & Workstation	2023-12-31	1	50	49	
ASGARD Management Center	2023-12-31	1	5	4	
THOR Server & Workstation	2023-09-30	1	5	4	
THOR Workstation	2023-08-15	3	20	17	
THOR Non-host-based	2023-12-31	0	10	10	

Fig. 1: Thunderstorm License Type in Customer Portal

2.2 Download Thunderstorm Installer Script

Use the Thunderstorm installer script `thunderstorm-installer.sh` for Linux systems published in our Github repository:

https://github.com/NextronSystems/nextron-helper-scripts/tree/master/thunderstorm

2.3 Install Required Packages

The Installer script requires the tools `wget` and `unzip`. To see if those tools are installed, run the following command:

```
user@unix:~$ which wget unzip
/usr/bin/wget
/usr/bin/unzip
```

If the output is empty or missing one of the tools, you can install the missing tools on your Linux system with one of the following commands:

```
user@unix:~$ sudo apt install wget unzip
```

```
user@unix:~$ sudo yum install wget unzip
```

```
user@unix:~$ sudo zypper install wget unzip
```

2.4 Run Thunderstorm Installer Script

Make sure that the license file is in the current working directory together with the thunderstorm-installer.sh and run the following commands:

```
user@unix:~$ chmod +x thunderstorm-installer.sh
```

The script will show you the changes that it's going to make and asks for a confirmation.

```
user@unix:~$ sudo ./thunderstorm-installer.sh
[sudo] password for user:
==============================================================
   ________                __            __
   /_  __/ /  __ _____  ___/ /__ _______ / /____  ______ _
   / / / _ \/ // / _ \/ _  / -_) __(_-</ __/ _ \/ __/  ' \
  /_/ /_//_/\_,_/_//_/\_,_/\__/_/ /___/\__/\___/_/ /_/_/_/
  v0.4.1

  THOR Thunderstorm Service Installer
  Florian Roth, August 2022
==============================================================

The script will make the following changes to your system:
1. Install THOR into /opt/nextron/thunderstorm
2. Drops a base configuration into /etc/thunderstorm
3. Create a log directory /var/log/thunderstorm for log files of the service
4. Create a user named 'thunderstorm' for the new service
5. Create a new service named 'thor-thunderstorm'

You can uninstall THOR Thunderstorm with './thunderstorm-installer uninstall'

Are you ready to install THOR Thunderstorm? (y/N)y
Started Thunderstorm Installer - version 0.4.1
Writing logfile to ./Thunderstorm_Installer_unix_20230105.log
```

(continues on next page)

(continued from previous page)

```
HOSTNAME: unix
IP: 192.168.0.110
OS: BUG_REPORT_URL="https://bugs.debian.org/";HOME_URL="https://www.debian.org/";
↪ID=debian;NAME="Debian GNU/Linux";PRETTY_NAME="Debian GNU/Linux 10 (buster)";SUPPORT_
↪URL="https://www.debian.org/support";VERSION="10 (buster)";VERSION_CODENAME=buster;
↪VERSION_ID="10";
ISSUE: Nextron Systems - ASGARD Management Center - \l
KERNEL: Linux unix 4.19.0-21-amd64 #1 SMP Debian 4.19.249-2 (2022-06-30) x86_64 GNU/Linux
Checking the required utilities ...
All required utilities found.
Searching for license file in current folder ...
```

2.5 Debugging

2.5.1 Most Common Problems

- Wrong or expired license
- Port 8080 is already in use

2.5.2 Access the Web GUI

Check the Web GUI on: `http://127.0.0.1:8080/`

2.5.3 Check the Log File

```
user@unix:~$ sudo tail -100 /var/log/thunderstorm/thunderstorm.log
```

2.5.4 Start Service Manually

```
user@unix:~$ sudo /opt/nextron/thunderstorm/thor-linux-64 --thunderstorm -t /etc/
↪thunderstorm/thunderstorm.yml
```

Warning: in case of a successful service start, the log file will be created readable for root user only, make sure to delete if afterwards. An unwritable log file causes the service to fail.

```
user@unix:~$ sudo rm /var/log/thunderstorm/thunderstorm.log
```

2.6 Silent Installation

In cases in which you do not want the installer to prompt you for a confirmation (e.g. Docker installation), use the `auto` parameter.

```
user@unix:~$ sudo ./thunderstorm-installer.sh auto
```

2.7 Uninstall Thunderstorm

You can always uninstall THOR Thunderstorm with

```
user@unix:~$ sudo ./thunderstorm-installer.sh uninstall
```

The only files that are left on a system are the log files in `/var/log/thunderstorm`.

CHAPTER

THREE

NEXT STEPS

After the installation you can test the service using one of the Thunderstorm collectors or the Python API client.

3.1 Configuration

The installation script for Linux system installs a service that passes the parameter `-t /etc/thunderstorm/thunderstorm.yml` to initialize the default config stored at that location.

The default configuration file on Linux looks like this:

```
# License path
license-path: /etc/thunderstorm
# Write all outputs to the following directory
logfile: /var/log/thunderstorm/thunderstorm.log
appendlog: True
# Listen on all possible network interfaces
server-host: 0.0.0.0
server-port: 8080
# Pure YARA scanning
pure-yara: False
# SSL/TLS
# SSL/TLS Server Certificate
#server-cert: /path/to/file
# SSL/TLS Server Certificate Private Key
#server-key: /path/to/file
# File Submissions
# Directory to which the samples get stored in asynchronous mode
server-upload-dir: /tmp/thunderstorm
# Permanently store the submitted samples (valied values: none/all/malicious)
server-store-samples: none
# Tuning
# Server Result Cache
# This is the number of cached results from asynchronous submission
# available for remote queries (default: 10000)
#server-result-cache-size: 10000
```

You can use all of THOR's flags in that configuration. Be advised that you always have to use their long form.

This page lists all of THOR command line flags:

https://github.com/NextronSystems/nextron-helper-scripts/tree/master/thor-help

The following chapters list some of the most useful command line flags when using THOR Thunderstorm.

3.1.1 Forward Logs to SIEM or Analysis Cockpit

```
syslog: mysiem.local
```

Config entry to forward logs to a SIEM

We recommend reading chapter *Syslog or TCP/UDP Output <https://thor-manual.nextron-systems.com/en/latest/usage/output-options.html#syslog-or-tcp-udp-output>* in the THOR User Manual for details on the SYSLOG forwarding flags. You can find it in the folder `/opt/nextron/thunderstorm/docs` after a successful Thunderstorm installation on Linux or in the "Downloads" section in the customer portal.

Keep Samples on the Thunderstorm Server

Keep samples with findings

```
server-store-samples: malicious
```

Keep all samples

```
server-store-samples: all
```

3.2 Custom Signatures

Since most of the functionalities of THOR are included in Thunderstorm, you can also include your own custom signatures. The process is identical to that of a normal THOR installation. Please see the Custom Signatures chapter in the THOR Manual.

Note: Don't forget to *Restart the Service* after placing your custom signatures in the dedicated directory.

3.3 Log Output

The scan results and startup messages can be found in:

```
user@unix:~$ sudo less /var/log/thunderstorm/thunderstorm.log
```

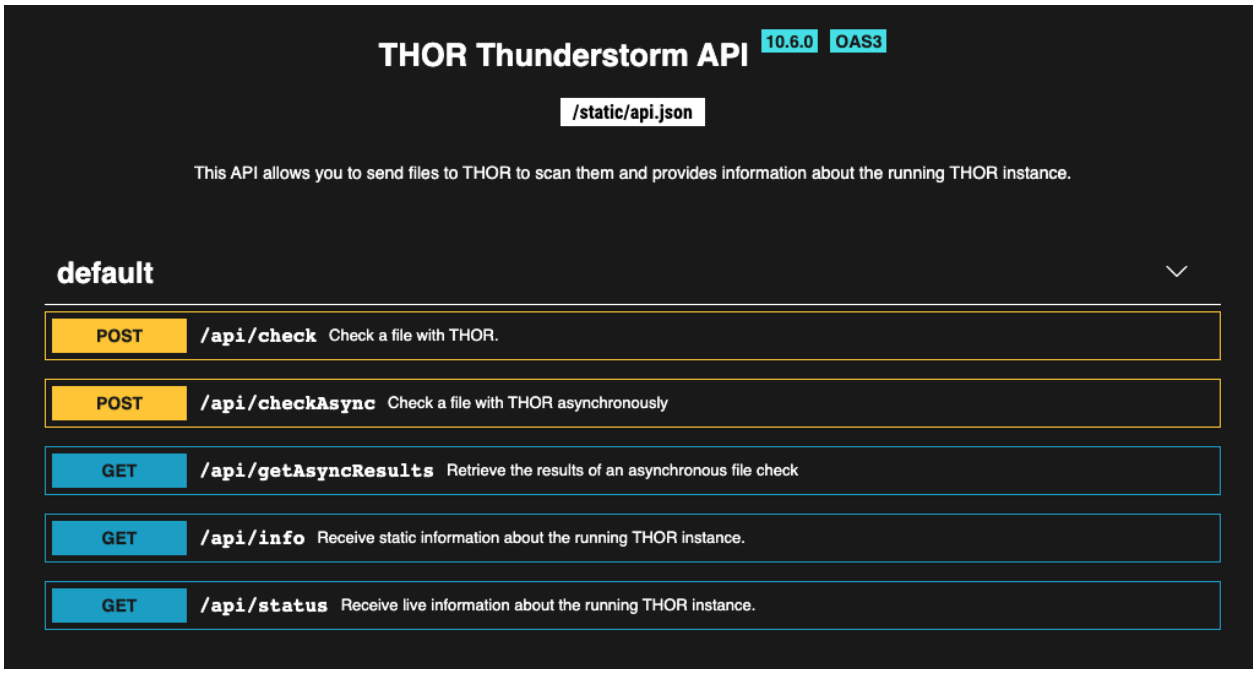
You could open another command line window and monitor new messages with:

```
user@unix:~$ sudo tail -f /var/log/thunderstorm/thunderstorm.log
```

3.4 Thunderstorm API Documentation

An API documentation is integrated into the web service.

Simply visit the service URL, e.g.: `http://my-server:8080/`



THOR Thunderstorm API 10.6.0 OAS3

/static/api.json

This API allows you to send files to THOR to scan them and provides information about the running THOR instance.

default

POST /api/check Check a file with THOR.

POST /api/checkAsync Check a file with THOR asynchronously

GET /api/getAsyncResults Retrieve the results of an asynchronous file check

GET /api/info Receive static information about the running THOR instance.

GET /api/status Receive live information about the running THOR instance.

Fig. 1: Thunderstorm API Documentation

3.5 Test Submission

To test the Thunderstorm service, you can create a tiny webshell sample and submit it to the service using the following commands.

```bash
#!/bin/bash
echo "<%eval request(" > test.txt
curl -X POST "http://my-server:8080/api/check?pretty=true" -F "file=@test.txt"
```

This should produce the following output in the current command line.

```
[
   {
      "level": "Alert",
      "module": "Filescan",
      "message": "Malware file found",
      "score": 350,
      "context": {
         "ext": ".txt",
         "file": "test.txt",
         "firstBytes": "3c256576616c207265717565737428a / \\u003c%eval request(\\n",
         "md5": "2481bc6bb2d063522ef8b5d579fd97d7",
```

(continues on next page)

(continued from previous page)

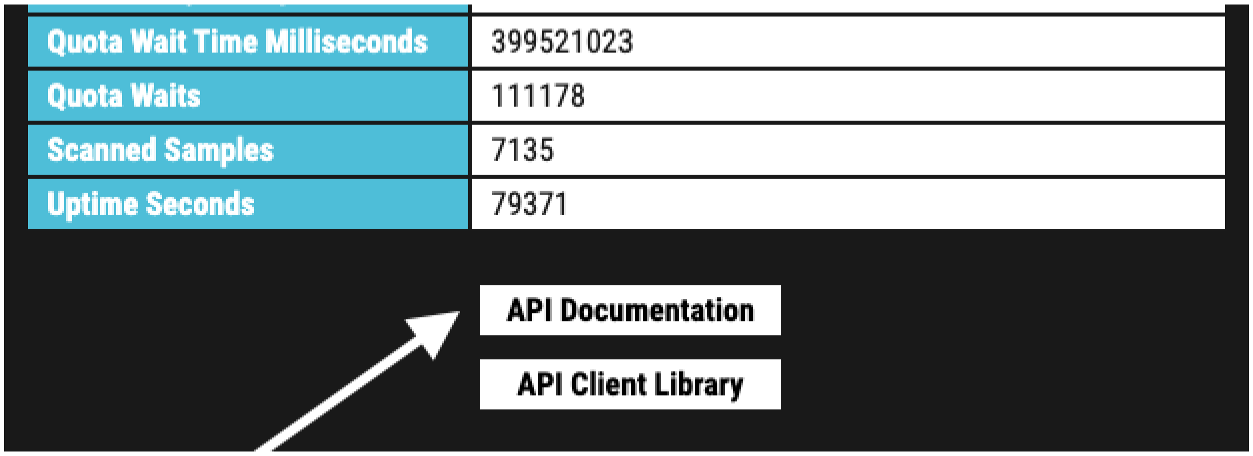
```
            "sha1": "4d40de75d7c8591d2ea59d3a000fb6cf58d97896",
            "sha256": "3b435df5076f6b1df7f2bc97cd86fbf7b479352e8c33960dfc4f1cbbe9b14fa7",
            "size": 16,
            "type": "JSP"
        }
    }
]
```

Output of test sample submission

Be aware that this has been a "synchronous" submission to the API endpoint "check". The collection of high amounts of samples in collector scripts and tools uses the endpoint "checkAsync", which doesn't return a result to the submitting source.

3.5.1 Test Submission Using the API Documentation

The web GUI running on Port 8080 contains an interactive API documentation, which you can use to submit a first test sample.



Quota Wait Time Milliseconds	399521023
Quota Waits	111178
Scanned Samples	7135
Uptime Seconds	79371

Fig. 2: Link to API Documentation on Start Page

Select the API function `/api/check`, then click "Try it out" and then select and submit a sample using the enabled form.

The result appears in a separate text field. Use the "pretty" flag to get a prettified JSON response.

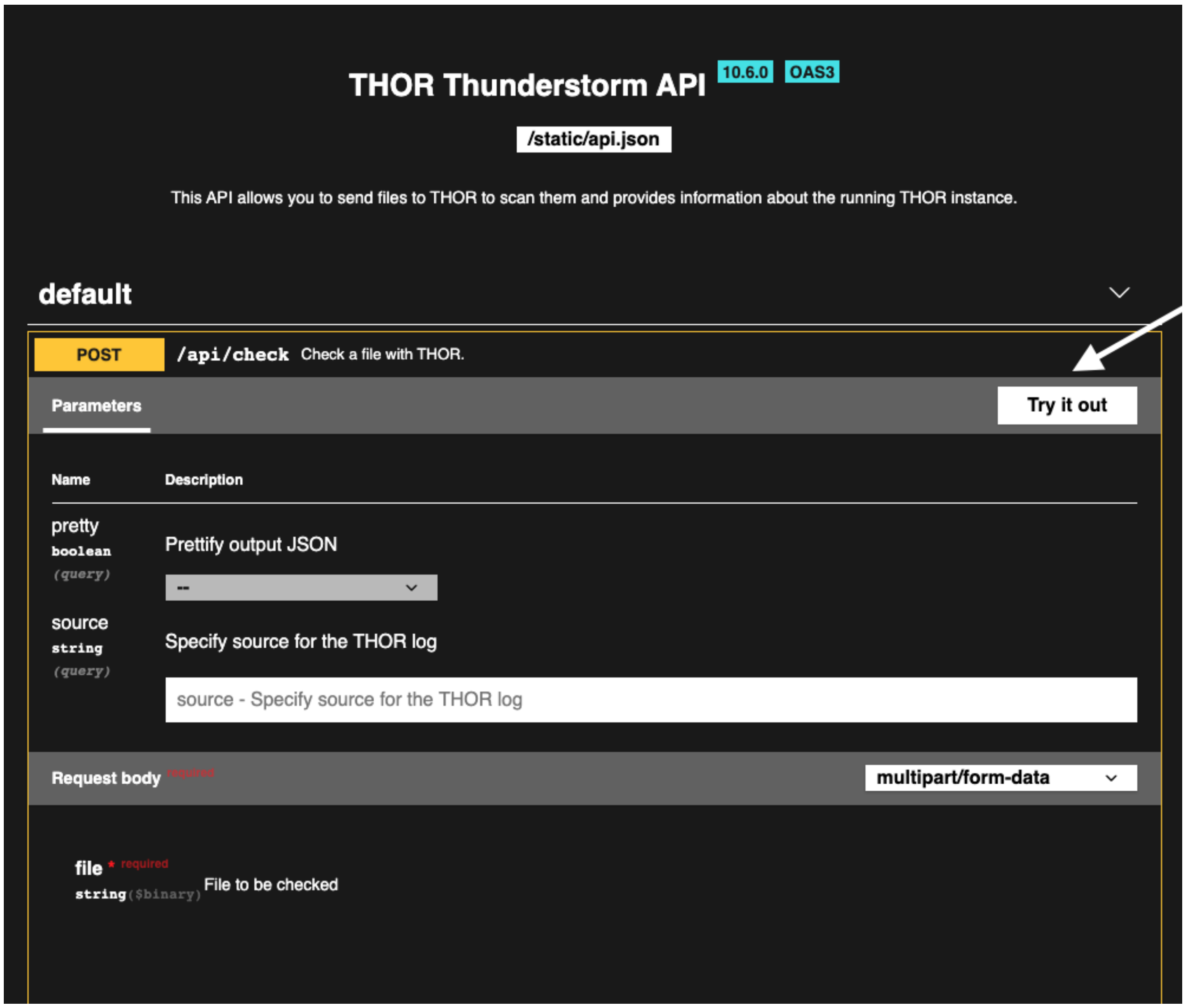
3.6 Thunderstorm Collectors

You can find a Thunderstorm collector for numerous different operating systems and architecture in our Github repository. We recommend using the collectors written in Go.

https://github.com/NextronSystems/thunderstorm-collector

You find pre-compiled collector binaries in the release section of the repository.

See the README on Github for more information.



This API allows you to send files to THOR to scan them and provides information about the running THOR instance.

pretty boolean (query) — Prettify output JSON: --

source string (query) — Specify source for the THOR log: source - Specify source for the THOR log

file * required string($binary) — File to be checked

Fig. 3: Test Sample Submission via API Documentation

3.6.1 Run the Collectors

We highly recommend using the config.yml as a configuration during the collection. It limits the samples the collector selects for a submission to relevant file types and sizes. Otherwise the collector would transmit every possible file, which is usually not recommended.

To retrieve the latest `config.yml` file, you can use the URL in the following listing or download it using `wget`.

```
user@unix:~$ wget https://github.com/NextronSystems/thunderstorm-collector/releases/
↪latest/download/config.yml
```

You would then start a collector run using the following command line:

Windows (64 bit):

```
C:\nextron\thunderstorm>amd64-windows-thunderstorm-collector.exe -t config.yml
```

Linux (64 bit):

```
user@unix:~$ ./amd64-linux-thunderstorm-collector -t config.yml
```

(Replace the collector binary name with the one you plan to use)

3.6.2 Performance Considerations for the Collection

In a THOR Thunderstorm setup, the system load moves from the end systems to the Thunderstorm server.

In cases in which you don't use the default configuration file provided with the collectors (`config.yml`) and collect all files from an end system, the Thunderstorm server requires a much higher amount of time to process the samples.

E.g. A Thunderstorm server with 40 CPU Cores (40 threads) needs 1 hour to process all 400,000 files sent from a Windows 10 end system. Sending all files from 200 Windows 10 end systems to a Thunderstorm server with that specs would take up to 10 days to process all the samples.

As a rule of thumb, when using the hardware recommended in *Hardware Requirements*, calculate with a processing speed of `250 samples per core per minute`.

We highly recommend using the default configuration file named `config.yml` provided with the collectors. See the README on Github for more information.

3.7 Thunderstorm API Client

We provide a free and open source command line client written in Python to communicate with the Thunderstorm service.

https://github.com/NextronSystems/thunderstormAPI

It can be installed with:

```
user@unix:~$ pip install thunderstormAPI
```

3.8 Source Identification

The log file generated by THOR Thunderstorm doesn't contain the current host as hostname in each line. By default, it contains the sending source's FQDN or IP address if a name cannot be resolved using the locally configured DNS server.

However, every source can set a "source" value in the request and overwrite the automatically evaluated hostname. This way users can use custom values that are evaluated or set on the sending on the end system.

```
user@unix:~$ curl -X POST "http://myserver:8080/api/check?source=test" -F "file=@sample.
↪exe"
```

3.9 Synchronous and Asynchronous Mode

It is also important to mention that THOR Thunderstorm supports two ways to submit samples, a synchronous and an asynchronous mode.

The default is synchronous submission. In this mode, the sender waits for the scan result, which can be empty in case of no detection or contains match elements in cases in which a threat could be identified.

In asynchronous mode, the submitter doesn't wait for the scan result but always gets a send receipt with an id, which can just be discarded or used to query the service at a later point in time. This mode is best for use cases in which the submitter doesn't need to know the scan results and batch submission should be as fast as possible.

	Synchronous	Asynchronous
Server API Endpoint	/api/check	/api/checkAsync
ThunderstormAPI Client Parameter		--asyn
Advantage	Returns Scan Result	Faster Submission
Disadvantage	Client waits for result of each sample	No immediate scan result on the client side

In asynchronous mode, the Thunderstorm service keeps the samples in a queue on disk and processes them one by one as soon as a thread has time to scan them. The number of files in this queue can be queried at the status endpoint `/api/status` and checked on the landing page of the web GUI.

In environments in which the Thunderstorm service is used to handle synchronous and asynchronous requests at the same time, it is possible that all threads are busy processing cached asynchronous samples and not more synchronous requests are possible.

In this case use the `--sync-only-threads` flag to reserve a number of threads for synchronous requests. (e.g. `--threads 40 --sync-only-threads 10`)

3.10 Performance Tests

Performance tests showed the differences between the two submission modes.

In Synchronous mode, sample transmission and server processing take exactly the same time since the client always waits for the scan result. In asynchronous mode, the sample transmission takes much less time, but the processing on the server takes a bit longer, since the sever caches the samples on disk.

	Synchronous	Asynchronous
Client Transmission	40 minutes	18 minutes
Server Processing		46 minutes
Total Time	40 minutes	46 minutes

3.11 SSL/TLS

We do not recommend the use of SSL/TLS since it impacts the submission performance. In cases in which you transfer files through networks with IDS/IPS appliances, the submission in an SSL/TLS protected tunnel prevents IDS alerts and connection resets by the IPS.

Depending on the average size of the samples, the submission frequency and the number of different sources that submit samples, the transmission could take up to twice as much time.

Note: The thunderstormAPI client doesn't verify the server's certificate by default as in this special case, secrecy isn't important. The main goal of the SSL/TLS encryption is an obscured method to transport potentially malicious samples over network segments that could be monitored by IDS/IPS systems. You can activate certificate checks with the `--verify` command line flag or `verify` parameter in API library's method respectively.

CHAPTER

FOUR

MAINTENANCE

4.1 Location of the Components

THOR Thunderstorm uses several components and paths for the configuration. Please see the table below for more information:

Component	Path
Config	/etc/thunderstorm/thunderstorm.yml
Binaries & Signatures	/opt/nextron/thunderstorm
Logs	/var/log/thunderstorm (you can change that in the configuration)
Sample Files	/tmp/thunderstorm (you can change that in the configuration)

4.2 Restart the Service

The following command will restart the THOR Thunderstorm service:

```
nextron@thunder:~$ sudo systemctl restart thor-thunderstorm.service
```

Warning: Only restart the service if you are sure that no samples are in the queue. The queue will get deleted once the service restarts, so only use this if you are sure that no samples get lost.

4.3 Review the Service Status

To debug potential problems, you can run the following commands and see what might be the problem.

Check if the service is still running:

```
nextron@thunder:~$ sudo systemctl status thor-thunderstorm.service
[sudo] password for nextron:
 thor-thunderstorm.service - THOR Thunderstorm Server
   Loaded: loaded (/etc/systemd/system/thor-thunderstorm.service; enabled; vendor␣
↪preset: enabled)
   Active: active (running) since Wed 2023-01-18 15:28:17 CET; 17h ago
Main PID: 39960 (thor-linux-64)
```

Check the last entries in the service log for errors

```
nextron@thunder:~$ tail /var/log/thunderstorm/thunderstorm.log
```

Check if the service is listening on a port:

```
nextron@thunder:~$ sudo netstat -anpt | grep thor
```

4.4 Update

There are two methods on how to update THOR Thunderstorm.

The first method will only update the signatures. This is the safer option, since the service will not be restarted automatically.

```
user@unix:~$ sudo thunderstorm-update
[...]
Mar 21 15:54:21 unix THOR_UTIL: Info: Starting Upgrade Process
Mar 21 15:54:21 unix THOR_UTIL: Info: License file found OWNER: user TYPE: thunderstorm␣
↪STARTS: 2023/03/21 EXPIRES: 2023/03/24
Mar 21 15:54:21 unix THOR_UTIL: Info: Downloading 'signatures'
Mar 21 15:54:21 unix THOR_UTIL: Info: Downloading from: https://update1.nextron-systems.
↪com/[...]
Mar 21 15:54:21 unix THOR_UTIL: Info: already up-to-date
Successfully updated signatures
Now restart the Thunderstorm service at the next opportunity with: sudo systemctl␣
↪restart thor-thunderstorm
Note: Use './thunderstorm-update full' to upgrade the binaries and signatures (warning:␣
↪it will also restart the service automatically)
```

The second method will also update the signatures and the THOR Thunderstorm binary. This should only be used when you are sure that the sample queue is empty and no samples are being scanned at the moment!

```
user@unix:~$ sudo thunderstorm-update full
[...]
Mar 21 15:58:47 unix THOR_UTIL: Info: Starting Upgrade Process
Mar 21 15:58:47 unix THOR_UTIL: Info: License file found OWNER: user TYPE: thunderstorm␣
↪STARTS: 2023/03/21 EXPIRES: 2023/03/24
Mar 21 15:58:47 unix THOR_UTIL: Info: Downloading 'thor-linux'
Mar 21 15:58:47 unix THOR_UTIL: Info: Downloading from: https://update1.nextron-systems.
↪com/[...]
Mar 21 15:58:48 unix THOR_UTIL: Info: already up-to-date
Mar 21 15:58:48 unix THOR_UTIL: Info: THOR 10 detected, also updating signatures ...
Mar 21 15:58:48 unix THOR_UTIL: Info: Starting Upgrade Process
Mar 21 15:58:48 unix THOR_UTIL: Info: License file found OWNER: user TYPE: thunderstorm␣
↪STARTS: 2023/03/21 EXPIRES: 2023/03/24
Mar 21 15:58:48 unix THOR_UTIL: Info: Downloading 'signatures'
Mar 21 15:58:48 unix THOR_UTIL: Info: Downloading from: https://update1.nextron-systems.
↪com/[...]
Mar 21 15:58:48 unix THOR_UTIL: Info: already up-to-date
Restarting Thunderstorm service ...
Successfully updated THOR and signatures
```

4.5 Replace the License

In order to add a new license, copy it to the `/etc/thunderstorm/` directory.

The THOR Thunderstorm service will automatically pick the first valid license and use it.

Note: If you've added a license with a higher quota limit (samples per hour) and the old one has not expired, you have to remove the old license, so that the Thunderstorm service cannot select and use it.

CHAPTER

FIVE

THUNDERSTORM API

In this chapter we will describe the API endpoints of the THOR Thunderstorm service.

`POST /api/check`

Check a file with THOR.

Query Parameters

- **`pretty`** (*`boolean`*) -- Prettify output JSON
- **`source`** (*`string`*) -- Specify source for the THOR log

Status Codes

- 200 OK -- Returns a Set with messages
- 400 Bad Request -- Invalid parameters given - no file, or multiple files, ...
- 500 Internal Server Error -- Internal server error
- 503 Service Unavailable -- Too many simultaneous requests, retry later

`POST /api/checkAsync`

Check a file with THOR asynchronously

Query Parameters

- **`source`** (*`string`*) -- Specify source for the THOR log

Status Codes

- 200 OK -- Returns a map containing the sample id
- 400 Bad Request -- Invalid parameters given - no file, or multiple files, ...
- 500 Internal Server Error -- Internal server error

`GET /api/getAsyncResults`

Retrieve the results of an asynchronous file check

Query Parameters

- **`pretty`** (*`boolean`*) -- Prettify output JSON
- **`id`** (*`integer`*) -- Sample ID

Status Codes

- 200 OK -- Returns a JSON with the current status and, if applicable, the results
- 400 Bad Request -- Invalid parameters - no ID specified, or ID invalid or not present in cache
- 500 Internal Server Error -- Internal server error

`GET /api/queueHistory`

Retrieve a history of how many asynchronous requests were queued

Query Parameters

- `pretty` (*boolean*) -- Prettify output JSON
- `aggregate` (*integer*) -- Aggregate this many minutes per value (default 1)
- `limit` (*integer*) -- Give a history for the last this many minutes (default infinite)

Status Codes

- 200 OK -- Returns a JSON Map with each time mapped to the queue length (estimated) from the last time mentioned to that time
- 400 Bad Request -- Invalid parameters - aggregate specified is no integer

`GET /api/sampleHistory`

Retrieve a history of how many samples were scanned

Query Parameters

- `pretty` (*boolean*) -- Prettify output JSON
- `aggregate` (*integer*) -- Aggregate this many minutes per value (default 1)
- `limit` (*integer*) -- Give a history for the last this many minutes (default infinite)

Status Codes

- 200 OK -- Returns a JSON Map with each time mapped to the samples scanned from the last time mentioned to that time
- 400 Bad Request -- Invalid parameters - aggregate specified is no integer

`GET /api/deniedHistory`

Retrieve a history of how many requests were denied

Query Parameters

- `pretty` (*boolean*) -- Prettify output JSON
- `aggregate` (*integer*) -- Aggregate this many minutes per value (default 1)
- `limit` (*integer*) -- Give a history for the last this many minutes (default infinite)

Status Codes

- 200 OK -- Returns a JSON Map with each time mapped to the requests denied from the last time mentioned to that time
- 400 Bad Request -- Invalid parameters - aggregate specified is no integer

`GET /api/info`

Receive static information about the running THOR instance.

Query Parameters

- `pretty` (*boolean*) -- Prettify output JSON

Status Codes

- 200 OK -- map with values to running version, rate limitation, ...

`GET /api/status`

Receive live information about the running THOR instance.

Query Parameters

- **`pretty`** (*`boolean`*) -- Prettify output JSON

Status Codes

- 200 OK -- map with values to scan times, scanned samples, wait times, ...

5.1 openAPI Specs

Since the plugin used for rendering the API schema does not consider the request body, please have a look down below what the request body needs to contains:

```
{
    "openapi": "3.0.1",
    "info": {
        "description": "This API allows you to send files to THOR to scan them and␣
↪provides information about the running THOR instance.",
        "title": "THOR Thunderstorm API",
        "version": "10.6.0"
    },
    "paths": {
        "/api/check": {
            "post": {
                "produces": [
                    "application/json"
                ],
                "summary": "Check a file with THOR.",
                "operationId": "check",
                "parameters": [
                    {
                        "description": "Prettify output JSON",
                        "name": "pretty",
                        "in": "query",
                        "schema": {
                            "type": "boolean"
                        }
                    },
                    {
                        "description": "Specify source for the THOR log",
                        "name": "source",
                        "in": "query",
                        "schema": {
                            "type": "string"
                        }
                    }
                ],
                "requestBody": {
                    "required": true,
                    "content": {
```

(continues on next page)

(continued from previous page)

```
                    "multipart/form-data": {
                        "schema": {
                            "properties": {
                                "file": {
                                    "description": "File to be checked",
                                    "type": "string",
                                    "format": "binary"
                                }
                            },
                            "required": [
                                "file"
                            ],
                            "type": "object"
                        }
                    }
                }
            },
            "responses": {
                "200": {
                    "description": "Returns a Set with messages"
                },
                "400": {
                    "description": "Invalid parameters given - no file, or multiple␣
↪files, ..."
                },
                "500": {
                    "description": "Internal server error"
                },
                "503": {
                    "description": "Too many simultaneous requests, retry later"
                }
            }
        }
    },
    "/api/checkAsync": {
        "post": {
            "produces": [
                "application/json"
            ],
            "summary": "Check a file with THOR asynchronously",
            "operationId": "checkAsync",
            "parameters": [
                {
                    "description": "Specify source for the THOR log",
                    "name": "source",
                    "in": "query",
                    "schema": {
                        "type": "string"
                    }
                }
            ],
            "requestBody": {
```

(continues on next page)

(continued from previous page)

```
                    "required": true,
                    "content": {
                        "multipart/form-data": {
                            "schema": {
                                "properties": {
                                    "file": {
                                        "description": "File to be checked",
                                        "type": "string",
                                        "format": "binary"
                                    }
                                },
                                "required": [
                                    "file"
                                ],
                                "type": "object"
                            }
                        }
                    }
                },
                "responses": {
                    "200": {
                        "description": "Returns a map containing the sample id"
                    },
                    "400": {
                        "description": "Invalid parameters given - no file, or multiple␣
↪files, ..."
                    },
                    "500": {
                        "description": "Internal server error"
                    }
                }
            }
        },
        "/api/getAsyncResults": {
            "get": {
                "produces": [
                    "application/json"
                ],
                "summary": "Retrieve the results of an asynchronous file check",
                "operationId": "getAsyncResults",
                "parameters": [
                    {
                        "description": "Prettify output JSON",
                        "name": "pretty",
                        "in": "query",
                        "schema": {
                            "type": "boolean"
                        }
                    },
                    {
                        "description": "Sample ID",
                        "name": "id",
```

(continues on next page)

(continued from previous page)

```
                        "in": "query",
                        "schema": {
                            "type": "integer"
                        }
                    }
                ],
                "responses": {
                    "200": {
                        "description": "Returns a JSON with the current status and, if␣
↪applicable, the results"
                    },
                    "400": {
                        "description": "Invalid parameters - no ID specified, or ID␣
↪invalid or not present in cache"
                    },
                    "500": {
                        "description": "Internal server error"
                    }
                }
            }
        },
        "/api/queueHistory": {
            "get": {
                "produces": [
                    "application/json"
                ],
                "summary": "Retrieve a history of how many asynchronous requests were␣
↪queued",
                "operationId": "queueHistory",
                "parameters": [
                    {
                        "description": "Prettify output JSON",
                        "name": "pretty",
                        "in": "query",
                        "schema": {
                            "type": "boolean"
                        }
                    },
                    {
                        "description": "Aggregate this many minutes per value (default 1)
↪",
                        "name": "aggregate",
                        "in": "query",
                        "schema": {
                            "type": "integer"
                        }
                    },
                    {
                        "description": "Give a history for the last this many minutes␣
↪(default infinite)",
                        "name": "limit",
                        "in": "query",
```

(continues on next page)

(continued from previous page)

```
                        "schema": {
                            "type": "integer"
                        }
                    }
                ],
                "responses": {
                    "200": {
                        "description": "Returns a JSON Map with each time mapped to the␣
↪queue length (estimated) from the last time mentioned to that time"
                    },
                    "400": {
                        "description": "Invalid parameters - aggregate specified is no␣
↪integer"
                    }
                }
            }
        },
        "/api/sampleHistory": {
            "get": {
                "produces": [
                    "application/json"
                ],
                "summary": "Retrieve a history of how many samples were scanned",
                "operationId": "sampleHistory",
                "parameters": [
                    {
                        "description": "Prettify output JSON",
                        "name": "pretty",
                        "in": "query",
                        "schema": {
                            "type": "boolean"
                        }
                    },
                    {
                        "description": "Aggregate this many minutes per value (default 1)
↪",
                        "name": "aggregate",
                        "in": "query",
                        "schema": {
                            "type": "integer"
                        }
                    },
                    {
                        "description": "Give a history for the last this many minutes␣
↪(default infinite)",
                        "name": "limit",
                        "in": "query",
                        "schema": {
                            "type": "integer"
                        }
                    }
                ],
```

(continues on next page)

(continued from previous page)

```
                "responses": {
                    "200": {
                        "description": "Returns a JSON Map with each time mapped to the␣
↪samples scanned from the last time mentioned to that time"
                    },
                    "400": {
                        "description": "Invalid parameters - aggregate specified is no␣
↪integer"
                    }
                }
            }
        },
        "/api/deniedHistory": {
            "get": {
                "produces": [
                    "application/json"
                ],
                "summary": "Retrieve a history of how many requests were denied",
                "operationId": "deniedHistory",
                "parameters": [
                    {
                        "description": "Prettify output JSON",
                        "name": "pretty",
                        "in": "query",
                        "schema": {
                            "type": "boolean"
                        }
                    },
                    {
                        "description": "Aggregate this many minutes per value (default 1)
↪",
                        "name": "aggregate",
                        "in": "query",
                        "schema": {
                            "type": "integer"
                        }
                    },
                    {
                        "description": "Give a history for the last this many minutes␣
↪(default infinite)",
                        "name": "limit",
                        "in": "query",
                        "schema": {
                            "type": "integer"
                        }
                    }
                ],
                "responses": {
                    "200": {
                        "description": "Returns a JSON Map with each time mapped to the␣
↪requests denied from the last time mentioned to that time"
                    },
```

(continues on next page)

(continued from previous page)

```
                    "400": {
                        "description": "Invalid parameters - aggregate specified is no␣
↪integer"
                    }
                }
            }
        },
        "/api/info": {
            "get": {
                "produces": [
                    "application/json"
                ],
                "summary": "Receive static information about the running THOR instance.",
                "operationId": "info",
                "responses": {
                    "200": {
                        "description": "map with values to running version, rate␣
↪limitation, ..."
                    }
                },
                "parameters": [
                    {
                        "description": "Prettify output JSON",
                        "name": "pretty",
                        "in": "query",
                        "schema": {
                            "type": "boolean"
                        }
                    }
                ]
            }
        },
        "/api/status": {
            "get": {
                "produces": [
                    "application/json"
                ],
                "summary": "Receive live information about the running THOR instance.",
                "operationId": "status",
                "responses": {
                    "200": {
                        "description": "map with values to scan times, scanned samples,␣
↪wait times, ..."
                    }
                },
                "parameters": [
                    {
                        "description": "Prettify output JSON",
                        "name": "pretty",
                        "in": "query",
                        "schema": {
                            "type": "boolean"
```

(continues on next page)

(continued from previous page)

```
                        }
                    }
                ]
            }
        }
    }
}
```

CHAPTER

SIX

LINKS AND REFERENCES

THOR Website

https://www.nextron-systems.com/thor/

Thunderstorm Collectors

https://github.com/NextronSystems/thunderstorm-collector

Thunderstorm Helper Scripts

https://github.com/NextronSystems/nextron-helper-scripts/tree/master/thunderstorm

Python based thunderstormAPI client module

https://github.com/NextronSystems/thunderstormAPI

https://pypi.org/project/thunderstormAPI

CHAPTER

SEVEN

INDICES AND TABLES

- search

HTTP ROUTING TABLE

/api