
THOR Manual Documentation

Nextron Systems GmbH

Nov 18, 2021

CONTENTS:

1	What is THOR?	1
1.1	Package	2
2	Requirements	3
2.1	Supported	3
2.2	Limited Support	4
2.3	Unsupported	4
3	Before You Begin	5
3.1	Add License File	5
3.2	Upgrade THOR and Update The Signatures	6
3.3	Define an Antivirus / EDR Exclusion	6
3.4	Choose The Right THOR Variant	7
3.5	Choose The Right Architecture	9
3.6	Choose The Right Command Line Flags	9
3.7	Verify Public Key Signatures (optional)	9
4	Deployment	11
4.1	Licensing	11
4.2	Network Share (Windows)	17
4.3	ASGARD Management Center (Windows, Linux, macOS)	19
4.4	Ansible (Linux)	20
4.5	THOR Thunderstorm Service	22
4.6	THOR Remote	28
4.7	Distribute to Offline Networks / Field Offices	30
4.8	System Load Considerations	30
5	Scan	31
5.1	Quick Start	31
5.2	Often Used Parameters	31
5.3	Parameters Possibly Relevant for Your User Case	32
5.4	Parameters Better Avoided	32
5.5	Lesser Known Parameters the Analysts Often Use	33
5.6	Help and Debugging	33
5.7	Examples	34
5.8	Run a Scan with Specific Modules	36
5.9	PE-Sieve Integration	36
5.10	Multi-Threading	37
6	Scan Modes	39
6.1	Modules	40

6.2	Features	42
7	Special Scan Modes	45
7.1	Lab Scanning (--lab)	45
7.2	Lookback Mode (--lookback --all-module-lookback)	47
7.3	Drop Zone Mode (--dropzone)	47
7.4	Image File Scan Mode (-m)	47
7.5	DeepDive (--deepdive)	47
7.6	Eventlog Analysis (-n)	48
7.7	MFT Analysis (--mft)	49
8	Analysis	51
8.1	ASGARD Analysis Cockpit	51
8.2	Splunk	51
8.3	THOR Util Report Feature	53
8.4	Log Analysis Manual	53
9	Configuration	55
9.1	Scan Templates	55
9.2	Maximum File Size	56
9.3	Exclude Elements	57
10	Output Options	61
10.1	Scan Output	61
10.2	Syslog or TCP/UDP Output	65
10.3	Encrypted Output Files	66
11	Update	67
11.1	Update Locations	67
11.2	Update Server Information	67
12	Custom Signatures	69
12.1	Simple IOCs	69
12.2	Sigma Rules	74
12.3	STIX IOCs	74
12.4	YARA Rules	75
12.5	Enhance YARA Rules with THOR Specific Attributes	80
13	Other Topics	87
13.1	Evidence Collection	87
13.2	Resource Control	90
13.3	Scoring System	90
13.4	Action on Match	93
13.5	THOR DB	95
14	Command Line Options	99
14.1	Scan Options	99
14.2	Scan Modes	100
14.3	Resource Options	101
14.4	Special Scan Modes	102
14.5	Thor Thunderstorm Service	102
14.6	License Retrieval	103
14.7	Active Modules	103
14.8	Module Extras	104
14.9	Active Features	105

14.10 Feature Extras	106
14.11 Output Options	106
14.12 ThorDB	108
14.13 Syslog	108
14.14 Reporting and Actions	108
14.15 THOR Remote	109
14.16 Automatic Collection of Suspicious Files (Bifrost)	109
14.17 Debugging and Info	109
15 Debugging	111
15.1 Debugging Examples	111
15.2 Finding Bottlenecks	112
15.3 Most Frequent Causes of Missing Alerts	112
15.4 Most Frequent Causes of Failed Scans	113
15.5 Help Us With The Debugging	114
16 Analysis and Info	117
16.1 Log Analysis Manual	117
16.2 VALHALLA Rule Lookup	117
16.3 Rule List Output	117
17 Use Cases	121
17.1 Disk Image Analysis with FTKImager	121
17.2 Memory Image Analysis with Volatility	121
18 Links and References	125
18.1 Open Source License Acknowledgements	125
19 Indices and tables	139

WHAT IS THOR?

THOR is a portable scanner for attacker tools and activity on suspicious or compromised server systems.

It covers a big set of basic checks and in deep analysis of the local event log, registry and file system. THOR aims to be a sensitive auditor noticing files and behavior traces a common Antivirus may have missed. An integrated "Scoring System" enables THOR to rate elements based on numerous characteristics to give hints on unknown malware.

THOR can be easily expanded to handle individual, client-specific attack patterns (e.g. the detection of specific malware files or certain log entries on the basis of a forensic analysis).

It is a portable and agentless "APT Scanner".

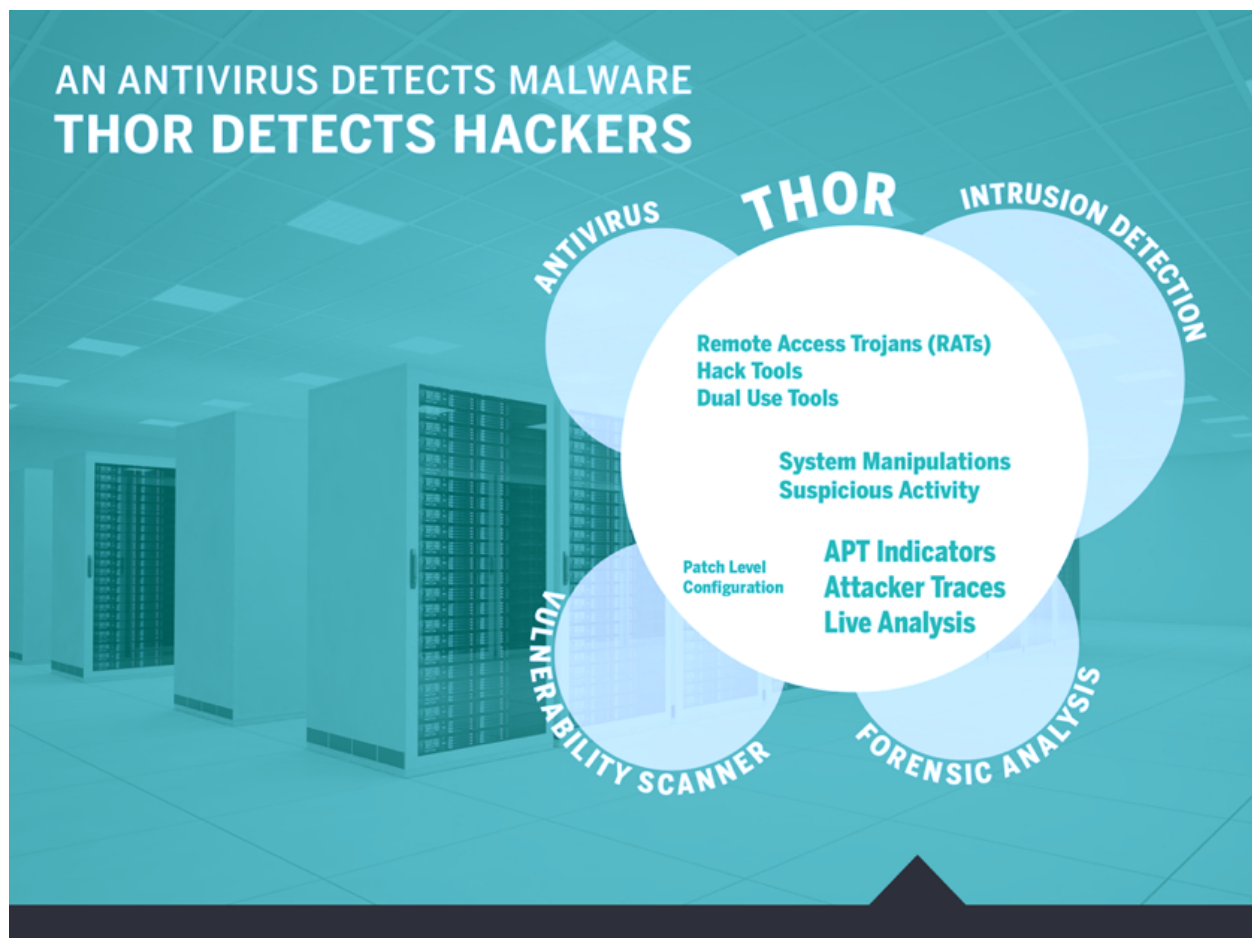


Fig. 1: THOR Coverage and Comparison to Antivirus and Intrusion Detection

The key features are:

- Scans for hack tools and attacker activity (with multiple detection mechanisms)
- Portable – no installation required
- Runs on Windows, Linux and macOS platforms without any prerequisites
- Adaptable to the specific tools and activity of new APT cases
- Scoring System – providing a way to detect previously unknown software
- Several Export Formats – Syslog (JSON/KV/CEF), HTML, TXT, JSON, CSV
- Throttling of the scan process to reduce the system load to a minimum

1.1 Package

The THOR Package includes the following files and directories:

- THOR Binaries - thor.exe for 32-bit and thor64.exe for 64-bit systems
- THOR Utility (thor-util.exe) (Helper tool for update, encryption, report generation, signature verification and other tasks – see the separate manual)
- Configuration Files in "./config" subfolder (directory-excludes.cfg, sigma.yml, false_positive_filters.cfg)
- Main Signature Database of THOR (./signatures/)
- Custom Signatures and Threat Intel IOCs (./custom-signatures/)
- THOR change log (changes.log)
- Additional tools like EXE packers and the Bifrost server script (./tools/)
- THOR manuals (./docs/)

REQUIREMENTS

THOR runs in any Windows, Linux and macOS environment without any further requirements. Everything needed is already included in the program package.

For best results, it is necessary to run THOR with administrative privileges or LOCAL_SYSTEM on Windows and “root” on Linux/macOS systems.

2.1 Supported

- Windows 7 x86 / x64
- Windows Server 2008 R2 x64
- Windows 8.1
- Windows Server 2012
- Windows Server 2012 R2
- Windows 10
- Windows Server 2016
- Windows Server 2019
- Windows Server 2022
- RHEL/CentOS 6
- RHEL/CentOS 7
- RHEL/CentOS 8
- SuSE SLES 11
- SuSE SLES 12
- SuSE SLES 15
- Ubuntu 16 LTS
- Ubuntu 18 LTS
- Ubuntu 20 LTS
- Debian 9
- Debian 10
- Debian 11

- macOS 10.14
- macOS 10.15
- macOS 11 (Intel)
- macOS 11 (ARM, Apple M1)

2.2 Limited Support

These versions are scannable with THOR Legacy. Contact us for details on how to download and use THOR Legacy.

- Windows XP SP2 x86 (unsupported)
- Windows XP SP2 x64 (unsupported)
- Windows Server 2003 x86 (unsupported)
- Windows Server 2003 x64 (unsupported)
- Windows Server 2008 x86
- Windows Server 2008 x64

2.3 Unsupported

- VMWare ESX - <https://kb.vmware.com/s/article/1036544>
- many others

If you need to perform an analysis on unsupported operating systems or architectures, contact us for a solution using [THOR Thunderstorm](#) and [Thunderstorm collectors](#).

We have productive setups with our customers involving the file collection from:

- SPARC Solaris
- RHEL Linux 4
- Citrix Netscaler
- ICS environments with Windows XP embedded systems (planned)
- VMWare ESX (see this [blog post](#))

BEFORE YOU BEGIN

3.1 Add License File

Place a valid license file into the program folder:

- THOR License File (*.lic)
THOR checks the program folder and all sub folder for valid license files. You can specify a certain path with **--licensepath path**.

3.1.1 Generate a License

You can generate a valid license in our [customer portal](#).

On Windows system you should use the computer name as hostname during license creation.

```
echo %COMPUTERNAME%
```

On Linux use the hostname command:

```
hostname
```

On macOS use the following command:

```
sysctl kern.hostname
```

Some more remarks regarding the hostname values:

- Use only the hostname of a FQDN (**master1** of **master1.internal.net**)
- The casing of the letters doesn't matter (case-insensitive)
- We do not store the hostnames anywhere in our portal

3.1.2 About License Files

THOR processes the program folder and all sub folders in search for a valid license file with a **"*.lic"** extension and picks the first valid license he can find.

This change has been made to facilitate the rollout using the new host based license model.

You can now generate licenses for a big set of systems, store all the licenses as **"thor-system1.lic"**, **"this-system2.lic"** and so on in a sub folder **"/licenses"** and transfer the THOR program folder with the "licenses" sub folder to all the different system for which you have generated licenses and just run the **"thor.exe"** executable.

3.2 Upgrade THOR and Update The Signatures

Run the following command to update THOR and its signatures:

```
thor-util upgrade
```

It is **important** that you update THOR after a download from the customer portal since the packages do not contain the newest signature files. (caused by internal integrity check processes)

Note: The upgrade requires a valid license for the host that performs the update. If you don't want to use a license for that host, ask us for a **silent license** that can be used for all kinds of testing purposes and also allows to update THOR and its signatures.

3.3 Define an Antivirus / EDR Exclusion

Since THOR accesses different process memories and probes for malicious Mutex, Named Pipe and Event values, it is recommended to exclude THOR from Antivirus / EDR scanning.

The Antivirus exclusion could also lead to a significant runtime reduction, since access to processes memory and files does not get intercepted anymore.

Note: We see massive runtime changes with Windows Defender since April 2021 (+50-100%). It is highly recommended to exclude THOR from scanning when using Windows Defender.

The quickest way to add an exclusion on a single system is:

Windows command line:

```
powershell -ep bypass -Command "Add-MpPreference -ExclusionProcess 'c:\temp\thor\thor64.exe'"
```

PowerShell:

```
Add-MpPreference -ExclusionProcess 'c:\temp\thor\thor64.exe'
```

For more information visit [this link](#).

3.3.1 A Note on SentinelOne

The process memory of systems running SentinelOne is polluted with suspicious strings. The most prevalent false positive is related to the keyword "ReflectiveLoader", but any other rule can match as well.

It is unclear what SentinelOne does to the process memory of many system processes. We cannot exclude these signatures from the scan. Be aware that the results from the "ProcessCheck" module on a system running SentinelOne can contain many false positives.

3.3.2 A Note on McAfee

It is not an easy task to define exclusions for THOR in all the different services when running McAfee products. You have to exclude the process in different sections (AV, EDR, On-Access). We've compiled a list of exclusions for our ASGARD customers, which you can find [here](#).

3.4 Choose The Right THOR Variant

We offer THOR in different variants.

- THOR
- THOR TechPreview
- THOR Legacy (limited support, availability)

3.4.1 THOR

The default version of THOR is the most stable version, intensively tested and without any broadly tested performance and detection tweaks.

The default version should be used for:

- Scan sweeps on many hundreds or thousands of systems
- Continuous compromise assessments on hundreds or thousands of systems
- Systems with high requirements on stability

3.4.2 THOR TechPreview

The TechPreview version is focussed on detection and speed. This [blog post](#) contains more information on the differences.

The TechPreview version should be used for:

- Digital forensic lab scanning
- Dropzone mode scanning
- Image scanning
- THOR Thunderstorm setups
- Single system live forensics on systems that don't have highest priority on stability

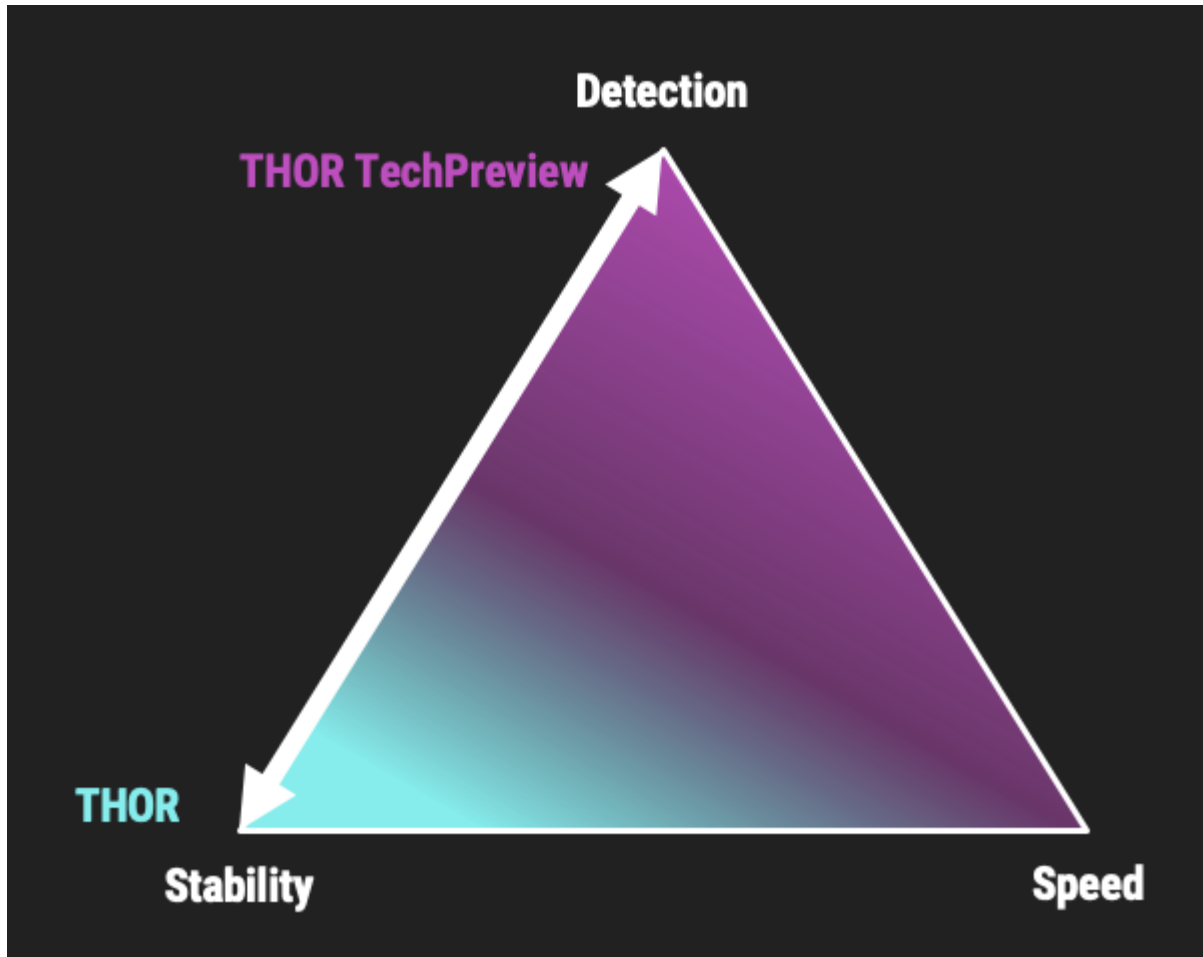


Fig. 1: THOR Default and TechPreview Focus

3.4.3 THOR Legacy

THOR Legacy is a stripped down version that includes all modules that can be used on outdated operating systems. This [blog post](#) contains more information on the legacy version.

The legacy version lacks:

- Module: Eventlog scanning
- Feature: Deeper process inspection (process hollowing, doppelganging, etc.)

The legacy version is not offered in our customer portal. We share it with our customers on request.

We only offer limited support for this version, since we cannot guarantee a successful stable scan on platforms that have already been deprecated.

3.5 Choose The Right Architecture

You will find a 32 and 64 bit version of the executable in the program folder. Never run the 32bit version of THOR named **thor.exe** on 64bit system. The 32bit version has some limitations that the 64bit version doesn't have. (memory usage, sees different folders on disk and registry versions)

Make sure to run the correct binary for your target architecture.

3.6 Choose The Right Command Line Flags

The recommended way to run THOR has already been put into the default. So, the recommended way to start a THOR is without any command line flags.

However, special circumstances can lead to different requirements and thus a different set of command line flags. See chapter '[Scan](#)' for often used flags.

3.7 Verify Public Key Signatures (optional)

You can verify the executable files in the THOR package with

- their digital signature (PE signature) issued by "Nextron Systems GmbH"
- thor-util's "verify" feature
- openssl verifying the integrity of executables manually

Find more information on THOR Util in its dedicated [online manual](#).

Note: THOR Util automatically verifies the signatures of the contained binaries in an update package and exits if one or more signatures cannot be verified. You don't have to check them manually unless you distrust the THOR Util itself. In this case, you can use the public key published on [our web page](#).

After downloading the public key the signatures can be manually verified with the following command:

```
openssl dgst -sha256 -verify <Path to public key .pem> -signature <Path to signature .
↳sig> <Path to the executable>
```

#Example:

```
openssl dgst -sha256 -verify nextronCode.pem -signature thor-linux.sig thor-linux
```


DEPLOYMENT

This chapter lists different ways to deploy THOR in an environment. Most of these methods are OS specific.

4.1 Licensing

In almost any method of deployment, the provision of valid licenses for the scanners on the endpoints is a core issue. Every license is limited to a certain host name. The only exception are the rare and relatively expensive "Incident Response" licenses.

In all other cases, a valid license has to be generated before a scan run.

There are numerous options to retrieve a valid license for a host.

With ASGARD:

- use an ASGARD Agent
- download THOR package with license from ASGARD's Downloads section
- generate licenses in ASGARD's web GUI under Licensing > Generate Licenses
- use THOR's `--asgard` and `--asgard-token` parameters to retrieve a license
- use ASGARD's API to retrieve a license manually

Without ASGARD:

- generate a license in the web GUI of the [customer portal](#)
- use THOR's `--portal-key` and `--portal-contracts` parameters to retrieve a license from the customer portal
- use the Customer Portal's API to retrieve a license manually

Some of the options are described in more detail in the following two chapters.

4.1.1 Retrieve Valid License From ASGARD

Use THOR's `--asgard` and `--asgard-token` parameters

In ASGARD 2.5+ you're able to configure a download token to limit the download of THOR packages and licenses to clients with knowledge of this token. The token is a protection that no one without knowledge of that token can intentionally exceed your license quota limit or retrieve a THOR package without authorization.

The download token can be configured in the Downloads section of you ASGARD server.

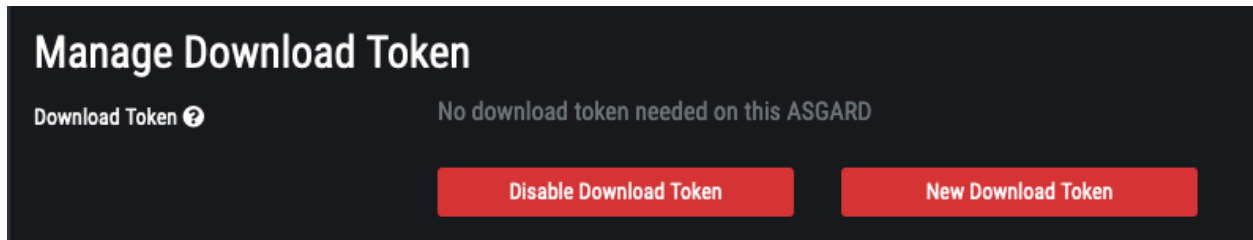


Fig. 1: Downloads > Download Token Configuration

You can retrieve an appropriate THOR license at the scan start using the builtin `--asgard` and `--asgard-token` parameters.

```
thor64.exe --asgard my-asgard.internal
```

```
thor64.exe --asgard my-asgard.internal --asgard-token OCU92GW1Cy0JLzaHkGrim1v200_ZkHPu0A
```

If everything works as expected, you'll see an INFO level message in the output that looks like:

```
Info: Init License file found LICENSE: my-asgard.internal OWNER: Master ASGARD: ACME Inc.
↪TYPE: Workstation STARTS: 2021/06/18 EXPIRES: 2022/06/18 SCANNER: All Scanners VALID:
↪true REASON:
```

Use ASGARD's API to retrieve a license manually

You can also script the license retrieval from a local ASGARD server by using the API. The help box in ASGARD's Licensing > Generate License section shows curl requests that can be used to retrieve licenses from your ASGARD server.

All you need are:

- Hostname
- System Type (server, workstation)

If there is uncertainty it's recommended to generate `server` type licenses which are more expensive but run on both system types.

For example: To retrieve a valid license for the servers named `SRV001` and `SRV002` you can use the following command:

```
curl -XPOST "https://my-asgard.internal:8443/api/v0/licensing/issue?
↪token=OJCBA7q4VGLjrCes2k4ACCQ0zg0AxAoz01" -o licenses.zip -d "type=server" -d
↪"hostnames=SRV001" -d "hostnames=SRV002" ... -d "hostnames=hostnameN"
```

If you can't use curl and want to retrieve a license as part of a bigger PowerShell script, you can use the following code snippet to help you with the retrieval.

```
# License retrieval script
# Florian Roth, June 2021

# ASGARD URL
$AsgardURL = "https://asgard.nextron-systems.com:8443/api/v0/licensing/issue"
$Token = "OJCBE7q7VGLjrCes4k4ACCQ0zg0AeAoz9Q"
```

(continues on next page)

Generate Licenses via API

Licenses can be generated and downloaded via License API.

Run THOR with License from ASGARD:

```
thor64.exe --asgard [redacted] nexttron-systems.com" --asgard-token  
[redacted] ...
```

Generate Server License(s) with **curl**:

```
curl -XPOST "https://[redacted]nexttron-systems.com:8443/api/v0/licensing/issue?  
token=[redacted] -o licenses.zip -d "type=server" -d  
"hostnames=hostname1" -d "hostnames=hostname2" ... -d "hostnames=hostnameN"
```

Generate Workstation License(s) with **curl**:

```
curl -XPOST "https://[redacted]nexttron-systems.com:8443/api/v0/licensing/issue?  
token=[redacted] -o licenses.zip -d "type=workstation" -d  
"hostnames=hostname1" -d "hostnames=hostname2" ... -d "hostnames=hostnameN"
```

Download all License(s) with **curl**:

```
curl -XGET "https://[redacted]nexttron-systems.com:8443/api/v0/licensing/download-all?  
token=[redacted] -o licenses.zip
```

i The license generation via API is restricted by a Download Token that can be configured or disabled [here](#).

Fig. 2: Licensing > Generate Licenses

(continued from previous page)

```

$LicenseFile = "licenses.zip"
$OutputPath = ".\"
$ExtractLicenses = $True

# Config
# Ignore Self-signed certificates
[System.Net.ServicePointManager]::ServerCertificateValidationCallback = {$true}
# Set current working directory for .NET as well
[Environment]::CurrentDirectory = (Get-Location -PSProvider FileSystem).ProviderPath

# Web Client
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12
$WebClient = New-Object System.Net.WebClient
if ( $Token ) {
    $AsgardURL = [string]::Format("{0}?token={1}", $AsgardURL, $Token)
}
Write-Host "Using URL: $AsgardURL"

# Hostname
$Hostname = $env:COMPUTERNAME

# License Type
$LicenseType = "server"
$OsInfo = Get-CimInstance -ClassName Win32_OperatingSystem
if ( $OsInfo.ProductType -eq 1 ) {
    $LicenseType = "workstation"
}

# Proxy Support
$WebClient.Proxy = [System.Net.WebRequest]::DefaultWebProxy
$WebClient.Proxy.Credentials = [System.Net.CredentialCache]::DefaultNetworkCredentials

# Prepare request
$postData=New-Object System.Collections.Specialized.NameValueCollection
$postData.Add('hostnames',$Hostname)
$postData.Add('type',$LicenseType)
Write-Host "Requesting license for HOST: $Hostname TYPE: $LicenseType"

# Request license
try {
    $Response = $WebClient.UploadValues($AsgardURL, $postData)
# HTTP Errors
} catch [System.Net.WebException] {
    Write-Host "The following error occurred: $_"
    $Response = $_.Exception.Response
    # 403
    if ( [int]$Response.StatusCode -eq 403 ) {
        Write-Host "This can be caused by a missing download token."
    }
    break
}
[System.IO.File]::WriteAllBytes($LicenseFile, $Response);

```

(continues on next page)

(continued from previous page)

```
# Extract licenses
if ( $ExtractLicenses ) {
    Add-Type -AssemblyName System.IO.Compression.FileSystem
    try {
        [System.IO.Compression.ZipFile]::ExtractToDirectory($LicenseFile, $OutputPath)
    } catch {
        Write-Host "The following error occurred: $_"
    }
    Remove-Item -Path $LicenseFile
}
```

Check the ASGARD helper scripts section in [our Github repo](#) for more scripts and snippets.

4.1.2 Retrieve Valid License From Customer Portal

Use THOR's `--portal-key` and `--portal-contracts` parameters to retrieve a license

To retrieve a licenses from the customer portal, you need an portal key. The portal key (API key) can be configured in the User Settings section of the [customer portal](#).

API Key

The API Key can be used to fetch and generate licenses via API. With **THOR 10.5+** you can pass your api key with `--portal-key YOUR-KEY` to automatically fetch a license for this host.

Fig. 3: User Settings > API Key

You can retrieve an appropriate THOR license at the scan start using the builtin `--portal-key` and `--portal-contracts` parameters. The `--portal-contracts` parameter is optional. It can be used to take licenses from a specific contract in case you have more than one and want to use a specific one. If none is set, THOR will automatically retrieve licenses from a contract of the right type. (e.g. retrieve workstation license from the first still valid contract that has workstation licenses available)

You can then use the parameters as shown in the following examples:

```
thor64.exe --portal-key IY5Y36thrt7h1775tt1ygfuyIadmGzZJmVk32lXcud4
```

```
thor64.exe --portal-key IY5Y36thrt7h1775tt1ygfuyIadmGzZJmVk32lXcud4 --portal-contracts_
↪ 13,14
```

If everything works as expected, you'll see an INFO level message in the output that looks like:

Contract ID ↑↓	Contract Type ↑↓	Description ↑↓
Search	All ▼	Search
13	Workstation	Workstation License Pool (400)
14	Server & Workstation	Server License Pool (400)
15	Service (Thunderstorm)	10 Service Licenses

Fig. 4: Contract IDs in Customer Portal

```
Info License file found LICENSE: portal.nextron-systems.com OWNER: ACME Inc TYPE:
↳ Workstation STARTS: 2021/06/23 EXPIRES: 2021/06/30 SCANNER: All Scanners VALID: true
↳ REASON:
```

Use the Customer Portal's API to retrieve a license manually

This is a bit more complicated as we've decided long ago that our customer portal will never contain personal or otherwise relatable information and this includes any kind of hostnames - not even in memory. Therefore it's necessary to generate a HMAC SHA1 hash of the lowercased hostname on the client side and include only the hash in the request to our customer portal.

This command generates a HMAC SHA1 of the current host you're working on. If you'd like to generate a license for a different host, simply replace the first part of the command with `echo -n "mycustomname"`.

```
echo -n "$(hostname -s)" | tr '[:upper:]' '[:lower:]' | openssl dgst -binary -sha1 -mac
↳ hmac -macopt
↳ hexkey:b190dd4a98456999b6d9c7e4e1ac1f231b978c3e7652898d7db2fcdede34613dbc7909c9fc8b3177bb904871b8b7fc
↳ | base64 | tr '/+' '_-' | tr -d '='
```

The values needed for a successful request are:

- \$CONTRACT = contract id (set to 0 for automatic selection)
- \$TYPE = [server/client]
- \$HASH = the hash generated from the hostname in the previous step
- \$APIKEY = the API from the User Settings section in the customer portal

```
curl -XPOST https://portal.nextron-systems.com/api/public/contracts/issue/$CONTRACT/
↳ $TYPE/$HASH?download=1 -H "Authorization: $APIKEY" -o license.lic
```

A valid license is an encrypted blob of at least 800 bytes. You can check the content of the license for possible error message that came back from the server using `xxd`.

```
xxd license.lic
```

If you find a **Error:** `HTTP-401` in the file, than you've most likely used an invalid API key.

4.2 Network Share (Windows)

THOR is a lightweight tool that can be deployed in many different ways. It does not require installation and leaves only a few temporary files on the target system.

A lightweight deployment option provides the THOR program folder on a read-only network share and makes it accessible from all systems within the network. Systems in DMZ networks can be scanned manually by transferring a THOR program package to the system and run it from the command line. The locally written log files have the same format as the Syslog messages sent to remote SIEM systems and can be mixed without any problem.

We often recommend triggering the scan via "Scheduled Task" distributed to the systems via GPO or PsExec. The servers access the file share at a given time, pull THOR into memory and start the scan process. You can either mount the network share and run THOR from there or access it directly via its UNC path (e.g. `\\server\share\thor.exe` or `\\server\share\thor64.exe`).

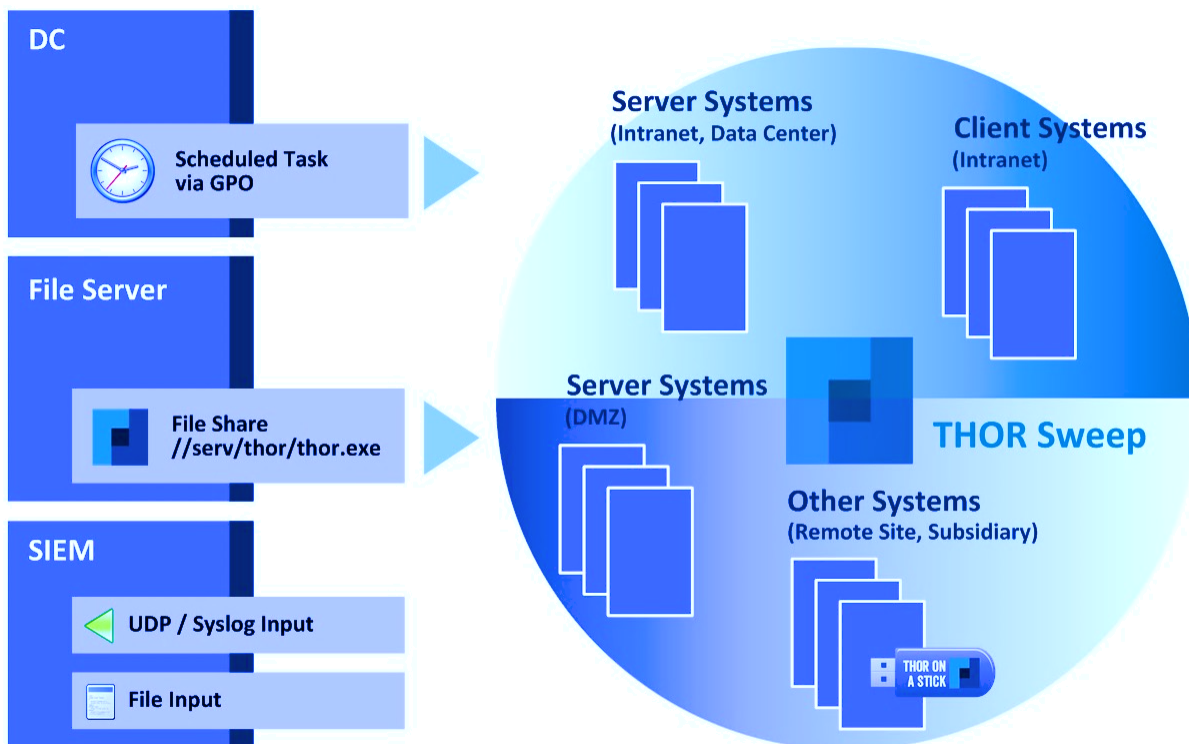


Fig. 5: Deployment via Network Share

4.2.1 Place THOR on a Network Share

A good way to run THOR on multiple systems is by defining a "Scheduled Task" using your Windows domain's group policy functionality.

The preferred way to run THOR on a remote system is by providing a network share on which the extracted THOR package resides. You can use this directory as the output directory but it is recommended to create another share with write permissions especially for the HTML and TXT result files. The share that holds the THOR program folder should be read-only. The various output files must be disabled or defined in different locations in order to avoid write-access errors.

The necessary steps are:

1. Create a network share and extract the THOR package into the root of the share, i.e. `\\fileserver\thor\`
2. Find the "thor_remote.bat" batch file, which can be found in the "tools" sub folder, place it directly in the root of the program folder and adjust it to your needs.
 - set the network share UNC path
 - set the parameters for the THOR run (see *chapter Scan*)

You should then test the setting like this:

1. Connect to a remote system (Remote Desktop), which you would like to scan
2. Start a command line "as Administrator"
(right click > Run as Administrator)
3. Run the following command, which is going to mount a network drive, run THOR and disconnect the previously mounted drive:
`\\fileserver\thor\thor_remote.bat`

After a successful test run, you decide on how to invoke the script on the network drive. The following chapters list different options.

4.2.2 Create a Scheduled Task via GPO

In a Windows Domain environment, you can create a Scheduled Task and distribute this Scheduled Task via GPO. This Scheduled Task would invoke the batch file on the network share and runs THOR. Make sure that the respective user account has the rights to mount the configured network share.

You can find more information here:

<https://technet.microsoft.com/en-us/library/cc725745.aspx>

4.2.3 Create a Scheduled Task via PsExec

This method uses Sysinternals PsExec and a list of target systems to connect and create a Scheduled Task via the command line. This could look like the following example:

```
psexec \\\\server1 -u domain/admin -p pass schtasks /create /tn "THOR
Run" /tr "\\server\share\thor\_remote.bat" /sc ONCE /st 08:00:00 /ru
DOMAIN\FUadmin /rp password
```

4.2.4 Start THOR on the Remote System via WMIC

THOR can be started on a remote system via "wmic" using a file share that serves the THOR package and is readable by the user that executes the scan.

```
wmic /node:10.0.2.10 /user:MYDOM\\scanadmin process call create "cmd.exe
/c \\\server\\thor10\\thor.exe"
```

4.3 ASGARD Management Center (Windows, Linux, macOS)

ASGARD is the central management platform for THOR scans. It manages distributed THOR scans on thousands of systems, collects, forwards and analyses logs. Furthermore, ASGARD can control and execute complex response tasks if needed.

ASGARD comes in two variations: While ASGARD Management Center features scan control and response functions, ASGARD Analysis Cockpit can be used to analyse large amounts of scan logs through an integrated base-lining and case management.

The hardened, Linux-based ASGARD appliance is a powerful, solid and scalable response platform with agents for Windows, Linux and MacOS. It provides essential response features like the collection of files, directories and main memory, remote file system browsing and other counteractive measures.

It features templates for scan runs and lets you plan and schedule distributed sweeps with the lowest impact on system resources. Other services are:

- **Quarantine Service** - file quarantine via Bifrost protocol
- **Update Service** - automatic updates for THOR scanners
- **License Service** - central registration and sub license generation
- **Asset Management Service** - central inventory and status dashboard
- **IOC Management** – manage and scan with custom IOC and YARA rule sets
- **Evidence Collection** – collect evidences (files and memory) from asset

Asset	Addresses	First Seen	Last Seen	Last Scan	System	Labels	Actions
ciWin2016 C:\Program Files\ASGARD\	172.28.31.125 fe80::697c:4f92:4219:a03b	2018-10-04	5m ago	11h ago	Windows	Windows	[Icons]
cimentos7 C:\Program Files\ASGARD\	172.28.31.127 192.168.122.1 fe80::754a:5a2:c20f:ab1b	2018-10-04	3m ago	12h ago	Linux	Linux	[Icons]
ciWin10 C:\Program Files\ASGARD\	172.28.31.123 fe80::f06c:cfbd:24e9:ae2e	2018-10-04	6m ago	12h ago	Windows	Windows	[Icons]
ciDebian9 C:\Program Files\ASGARD\	172.28.31.126 fe80::20c:29ff:fe80:6929	2018-10-04	4m ago	13h ago	Linux	Linux	[Icons]
ciWin7-1 C:\Program Files\ASGARD\	172.28.31.128 fe80::8c47:ea19:3446:a041	2018-10-04	11m ago	13h ago	Windows	Windows	[Icons]
ciWin2008R2 C:\Program Files\ASGARD\	172.28.31.128 fe80::ccc7:c7f5:465e:af94	2018-10-04	3m ago	13h ago	Windows	Windows	[Icons]
ciWin7-1x64 C:\Program Files\ASGARD\	172.28.31.133 fe80::10e:9365:f865:9ee7	2018-10-04	5m ago	13h ago	Windows	Windows	[Icons]

Fig. 6: ASGARD Management Center

4.4.3 Execute Thor using Ansible

The following section will show how to use an Ansible playbook to execute Thor on multiple Linux systems.

It will perform following steps on each system:

- Create a temporary folder
- Mount a RAM drive using the folder as mountpoint
- Copy Thor to this RAM drive
- Execute Thor
- Unmount the RAM drive
- Delete the temporary folder

4.4.4 Inventory File

First it is needed to define a list of hosts to execute Thor on. This is done by setting up a YAML file with the hostnames or IP addresses of the hosts. This file is later used with the `-i` parameter in the `ansible-playbook` command. A simple version of this could look like following:

```
---
host1.com
host2.com
132.123.213.111
```

To learn more about Ansible inventory files and how to use them, please refer to the official documentation:

https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html

4.4.5 Ansible Playbook Template

```
---
- hosts: all
#remote_user: root become: true tasks:
- name: Create folder for temporary RAM drive command: mkdir /mnt/temp_ram creates=/mnt/
temp_ram
- name: Create Thor RAM drive on target
command: mount -t ramfs -o size=60M ramfs /mnt/temp_ram/ ignore_warnings: true
- name: Copy Thor to RAM drive
copy: src=../thor-linux-pack/ dest=/mnt/temp_ram/ ignore_warnings: true
- name: Make Thor Executable
file: path=/mnt/temp_ram/thor-x64 state=touch
mode="0555"
- name: Execute Thor
command: /mnt/temp_ram/thor64 -l /mnt/temp_ram/thor.txt creates=/mnt/temp_ram/thor.
html
- name: Fetch Log file
fetch: src=/mnt/temp_ram/thor.txt dest=../thoransible-
output/{{inventory_hostname}}/thor.txt flat=true
- name: Unmount temporary RAM drive mount:
path: /mnt/temp_ram
```

(continues on next page)

(continued from previous page)

```
state: unmounted
- name: check Mount
command: mount
- name: Delete folder for temporary RAM drive
command: rmdir /mnt/temp_ram/
```

4.4.6 Usage of Thor's Ansible playbook

Copy the playbook in the main directory of Thor. After this is done it can be started as follows:

```
ansible-playbook -f <number\_of\_parallel\_executions> -i <inventory\_file> thorplaybook.
↪ yml
```

After the playbook finished running the scans, the output of each system can be found in the **thoransible**-output directory located at the parent directory of thor. Therefore it is important that the user starting ansible-playbook has the required rights to write in this directory.

4.4.7 Adjust Thor's Command Line Parameters

Per default this playbook will only start Thor with the parameter that defines the output log file. This can be changed in the playbook in the „Execute Thor“-Task. However, it should be kept in mind, that changing the output log file is not recommended, since the later tasks of the playbook depend on this.

4.5 THOR Thunderstorm Service

The command line flag "**--thunderstorm**" starts THOR as a RESTful web service on a given network interface and port. This service receives samples and returns a scan result.

The service can be started in two scan modes:

- Pure YARA
- Full-Featured

In the pure YARA mode (**--pure-yara**) THOR Thunderstorm only applies the 13,000 internal and all custom YARA rules to the submitted samples. It's lightweight and fast.

The full-featured mode is the default. In this mode Thunderstorm also parses and analyses Windows Eventlogs (EVTX), registry hives, memory dumps, Windows error reports (WER) and more. It's not just a YARA scan, but a full forensic processing.

Under normal circumstances, we recommend using the full-featured mode, since most files are not of a type that triggers an intense parsing function, the processing speed should be similar to the "pure-yara" mode.

It is recommended to use "pure-yara" mode in cases in which:

- huge forensic artefacts (EVTX or memory dump files) appear on the source systems and overload the Thunderstorm service
- deeper forensic parsing, IOC matching or other internal THOR checks aren't needed or wanted

The following table contains all THOR Thunderstorm related command line flags:

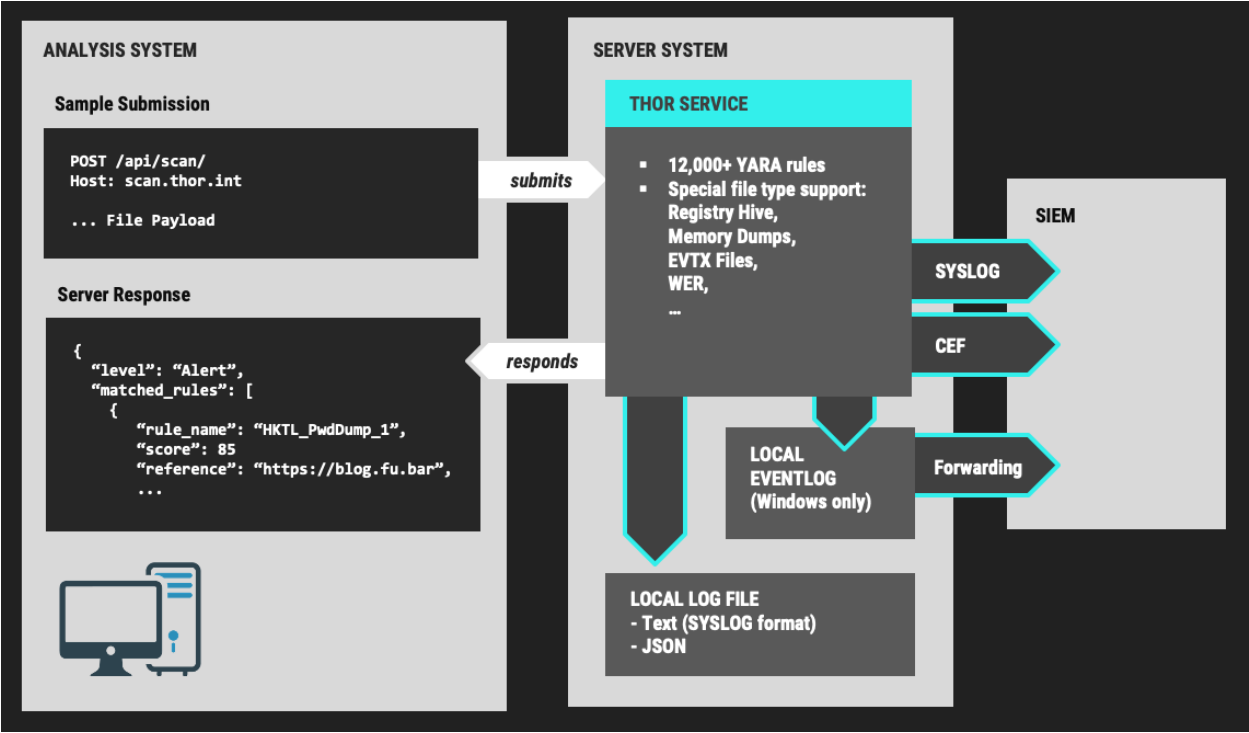


Fig. 8: THOR Thunderstorm Overview

Parameter	Values	Function
--thunderstorm		Watch and scan all files sent to a specific port (see --server-port). Disables resource checks and quick mode, activate intense mode, disable ThorDB and apply IOCs platform independently
--server-host	ip-address	IP address that THOR's server should bind to (default "127.0.0.1")
--server-port	port number	TCP port that THOR's server should bind to (default 8080)
--server-cert	.crt location	TLS certificate that THOR's server should use. If left empty, TLS is not used
--server-key	.key location	Private key for the TLS certificate that THOR's server should use. Required if

4.5. THOR Thunderstorm Service

<https://t.me/learningnets>

4.5.1 Service License Type

To run THOR in Thunderstorm service mode, you need a special license type named „Service License” that allows this mode of operation.

After the launch of THOR Thunderstorm, we may allow other license types to run THOR in service mode for a limited period of time, so that customers can test the service and its integration into other solutions.

4.5.2 Thunderstorm Collectors

4.5.3 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: **pip install thunderstormAPI**

4.5.4 Thunderstorm API Documentation

An API documentation is integrated into the web service.

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

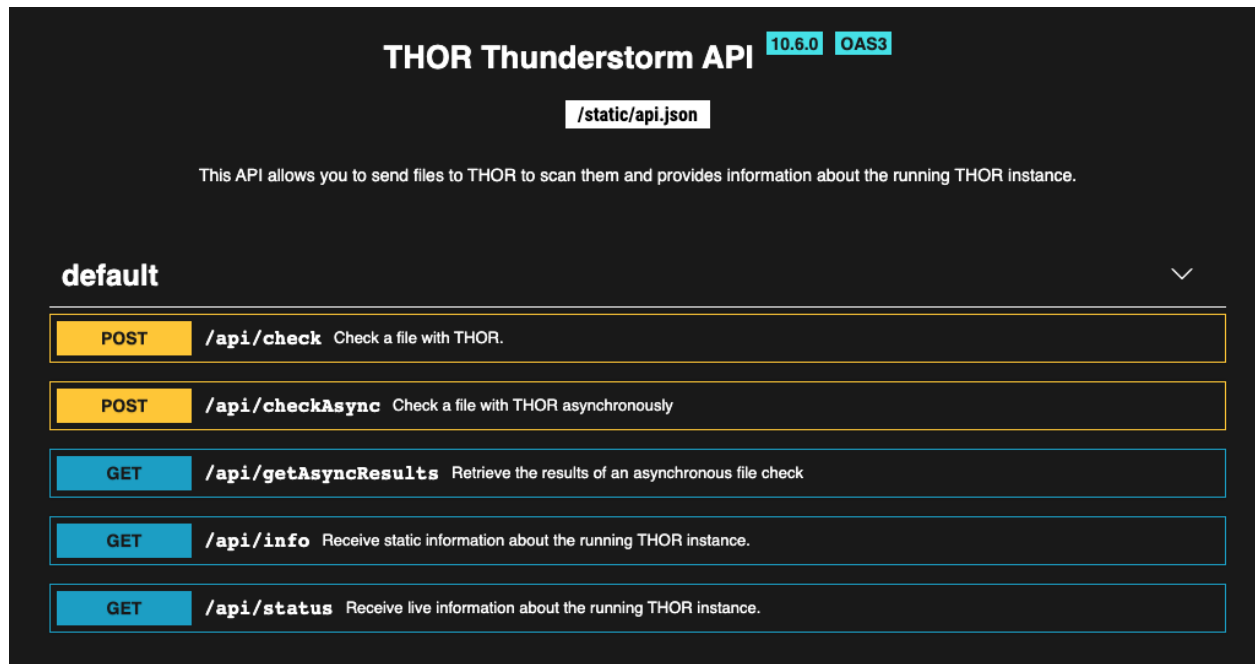


Fig. 9: Thunderstorm API documentation

4.5.5 Server Installer Script for Linux

A script that facilitates the installation on Linux systems can be found in our github repository.

<https://github.com/NextronSystems/nexttron-helper-scripts/blob/master/thunderstorm/thunderstorm-installer.sh>

The installation of a full THOR Thunderstorm server requires only two steps.

1. Download and place a THOR Service license file in the current working directory
2. Run the following command

```
wget -O - https://raw.githubusercontent.com/NextronSystems/nextron-helper-scripts/master/  
↳ thunderstorm/thunderstorm-installer.sh \& bash
```

Everything else will automatically be handled by the installer script. It even supports an “uninstall” flag to remove all files and folders from the system to get the system clean again after a successful proof-of-concept.

```
root@ygdrasil:/mnt/workspace/thunderstorm-installer# ./thunderstorm-setup.sh
=====
      _/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_\
     /_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_\_
    /_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_/_\_
   v0.1.2

THOR Thunderstorm Service Installer
Florian Roth, September 2020
=====

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
Started Thunderstorm Installer - version 0.1.2
Writing logfile to ./Thunderstorm_Installer_ygdrasil_20200904.log
HOSTNAME: ygdrasil
IP: 172.17.0.1 10.0.2.27
OS: BUG_REPORT_URL="https://bugs.debian.org/";HOME_URL="https://www.debian.org/";ID=debian
ME="Debian GNU/Linux 10 (buster)";SUPPORT_URL="https://www.debian.org/support";VERSION="10";
VERSION_ID="10";
ISSUE: Debian GNU/Linux 10 \n \l
KERNEL: Linux ygdrasil 4.19.0-8-amd64 #1 SMP Debian 4.19.98-1+deb10u1 (2020-04-27) x86_64
Checking the required utilities ...
All required utilities found.
Searching for license file in current folder ...
Found license file 9-20200409-020000-1.lic
Evaluated license hash: 8b7e4e6a10368d55a1720e54650b455f
Creating new directory '/opt/nextron/thunderstorm' ...
Creating new user 'thunderstorm' ...
Adding system user `thunderstorm' (UID 117) ...
```

Fig. 10: Thunderstorm Service Installer

After the installation, the configuration file is located in `/etc/thunderstorm`.

The log file of the service can be found in `/var/log/thunderstorm`.

4.5.6 Thunderstorm Update

The Thunderstorm service gets updated just as THOR does. Use “thor-util update” to update signatures or “thor-util upgrade” to update binaries and signatures. The service has to be stopped during the updates.

Update signatures:

```
thor-util update
```

Upgrade signatures:

```
thor-util upgrade
```

See the “thor-util” manual for details on how to use these functions.

Thunderstorm Update Script

The Thunderstorm installer script for Linux automatically places an updater script in the PATH of the server system.

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

Update binaries and signatures:

```
thunderstorm-update
```

Stop service, update binaries and signatures, restart service:

```
thunderstorm-update full
```

4.5.7 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.

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

4.5.8 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**)

4.5.9 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	40min	18min
Server Processing		46min
Total Time	40min	46min

4.5.10 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.

4.6 THOR Remote

THOR Remote is a quick method to distribute THOR in a Windows environment. It has been developed during an incident response and can be considered as a clever hack that makes use of PsExec to push and execute THOR with certain parameters on remote systems.

Requirements:

- Administrative Domain Windows user account with access rights on the target systems
- Reachability of the target systems (Windows Ports)
 - 135/tcp für SCM (Service Management)
 - 445/tcp für SMB (Mounting)
- A list of target systems

Advantages:

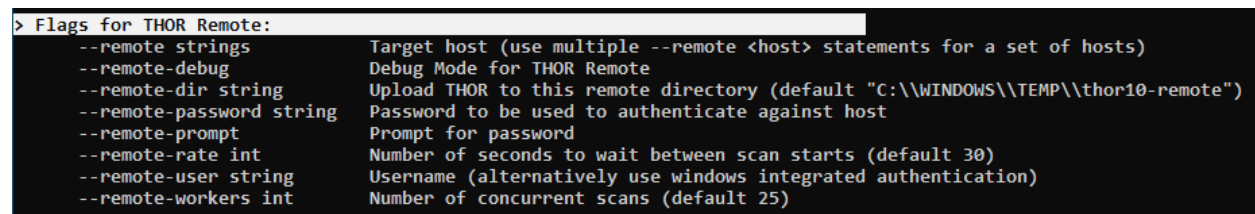
- Agent-less
- Comfortable scanning without scripting
- Quick results (useful in incident response scenarios)

Disadvantages:

- Requires reachability of Windows ports
- User credentials remain on the target system if it is used with explicit credentials (NTLM Auth) and the users doesn't already use an account that has access rights on target systems (Kerberos Auth)

4.6.1 Usage

A list of parameters used with the remote scanning function can be found in the help screen.



```
> Flags for THOR Remote:
--remote strings      Target host (use multiple --remote <host> statements for a set of hosts)
--remote-debug        Debug Mode for THOR Remote
--remote-dir string    Upload THOR to this remote directory (default "C:\\WINDOWS\\TEMP\\thor10-remote")
--remote-password string Password to be used to authenticate against host
--remote-prompt        Prompt for password
--remote-rate int       Number of seconds to wait between scan starts (default 30)
--remote-user string    Username (alternatively use windows integrated authentication)
--remote-workers int    Number of concurrent scans (default 25)
```

Fig. 11: THOR Remote Usage

As you can see, a list of target hosts can be provided with the help of the new YAML config files. See [chapter Configuration](#) for more details.

A YAML file with a list of hosts looks like this:

```
remote: - winatl001.dom.int - winatl002.dom.int - winnyk001.dom2.int
```

We recommend using a text editor that supports multi-line editing like Atom or Sublime.

<https://atom.io/>

<https://stackoverflow.com/questions/39391688/multi-line-editing-on-atom>

You can then use that file with:

```
thor64.exe -t targets.yml
```

4.6.2 THOR Remote Licensing

Valid licenses for all target systems are required. Place them in the program folder or any sub folder within the program directory (e.g. `./licenses`). In case of incident response licenses, just place that single license in the program folder.

You don't need a valid license for the system that runs THOR's remote scanning feature (the source system of the scans, e.g. admin workstation).

4.6.3 Output

The generated log files are collected and written to the folder `./remote-logs`

The "THOR Remote" function has its own interface, which allows you to view the progress of the scans, view and scroll through the log files of the different remote systems.

```

+-----+-----+
| Hosts | | THOR Events of dc-2016.testing |
+-----+-----+
| client-win10.testing | | Info: Report End Time: Thu Aug 8 12:55:02 2019 |
| dc-2016.testing | | Info: Report Scan took 0 hours 0 mins 4 secs |
| unknown.testing | | Notice: Report Thor Scan finished TIME: Thu Aug 8 12:55:02 2019 ALERTS: 0 WARNINGS: 0 NOTICES: 0... |
| | | Info: ThorDB Successfully closed ThorDB |
| | | ok |
+-----+-----+
| Console |
+-----+-----+
| Info: Remote The following remotes will be scanned HOSTS: client-win10.testing dc-2016.testing |
| unknown.testing |
| Info: Remote Mounting share HOST: client-win10.testing USER: MOUNT: \\client-win10.testing\c$ |
+-----+-----+

+-----+-----+
| Status | | Hosts |
+-----+-----+
| Pending 0 | | Info: Remote Starting THOR service HOST: dc-2016.testing |
| Running 0 | | Info: Remote Deleting THOR service HOST: dc-2016.testing |
| Completed 2 | | Info: Remote Deleting THOR pioneer service HOST: dc-2016.testing |
| Failed 1 | | Info: Remote Deleting THOR directory HOST: dc-2016.testing PATH: |
| | | \\dc-2016.testing\c$\WINDOWS\TEMP\thor10-remote |
+-----+-----+
| Keybindings | | Info: Remote Unmounting share HOST: dc-2016.testing MOUNT: \\dc-2016.testing\IPC$ |
| C-c, q Exit | | Info: Remote Unmounting share HOST: dc-2016.testing MOUNT: \\dc-2016.testing\c$ |
| Right Next Host | | Info: Remote Successfully scanned remote HOST: dc-2016.testing |
| Left Previous... | | Info: Remote Mounting share HOST: unknown.testing USER: MOUNT: \\unknown.testing\c$ |
+-----+-----+

```

Fig. 12: THOR Remote Interface

4.6.4 Issues

System Error 5 occurred – Access Denied

See: <https://helgeklein.com/blog/2011/08/access-denied-trying-to-connect-to-administrative-shares-on-windows-7/>

Running THOR from a Network Share

THOR must reside on the local filesystem of the source system. Don't run it from a mounted network share. This could lead to the following error:

```
CreateFile .: The system cannot find the path specified.
```

Using Templates and Other Absolute Paths

4.7 Distribute to Offline Networks / Field Offices

The quickest and most simple way to run THOR is by providing the ZIP archive to the colleagues in the remote location, letting them run the THOR executable and collect the report files afterwards.

The most usable format in this use case is the HTML report if only a few reports have to be analyzed. If the number of collected reports is high, we recommend using ASGARD Analysis Cockpit or Splunk with the free App and Add-on.

ASGARD Analysis Cockpit

<https://portal.nexttron-systems.com/webshop/downloads>

THOR APT Scanner App

<https://splunkbase.splunk.com/app/3717/>

THOR Add-On

<https://splunkbase.splunk.com/app/3718/>

4.8 System Load Considerations

We recommend staging the THOR Run in order to avoid resource bottlenecks (network or on VMware host systems). Especially during the THOR start, program files and signatures get pulled over the network, which is about 30 MB per system. Additionally, the modules, which take only a few seconds or minutes to complete, run first so that the load is higher during the first 10 to 15 minutes of the scan.

It is therefore recommended to define sets of systems that will run at the same time and let other systems start at intervals of an hour.

It is typically no problem to start a big set of physical machines at the same time. But if you start a scan on numerous virtual guests or on remote locations connected through slow WAN lines, you should define smaller scan groups.

SCAN

First of all, THOR runs fine with the default settings. The recommended scan options are already active in the default scan.

5.1 Quick Start

Follow these steps to complete your first THOR scan

1. Make sure you've read the *Before You Begin* guide
2. Open a command line (cmd.exe) as Administrator
3. Navigate to the folder in which you've extracted the THOR package and placed the license file(s)
4. Start THOR with **thor64.exe** (macOS: **thor-macos**, Linux: **thor-linux-64**)
5. Wait until the scan has completed (this can take between 20 and 180 minutes)
6. When the scan is complete, check the text log and HTML report in the THOR program directory

5.2 Often Used Parameters

Parameter	Description
--soft	Reduce CPU usage, skip all checks that can consume a lot of memory (even if only for a few seconds)
--quick	Perform a <i>quick scan</i> (skips Eventlog and checks only the most relevant folders)
-e target-folder	Write all output files to the given folder

5.3 Parameters Possibly Relevant for Your User Case

Parameter	Description
-c percentage	Reduce the average CPU load to the given percentage value note THOR already sets the process priority to the lowest possible value). This can be helpful to reduce the load e.g. on server systems with real-time services or reduce the noise produced by fans on user laptops.
--allhds	By default THOR scans only the C: partition on Windows machines and other files/folders only in cases in which some reference points to a different partition (e.g. configured web root of IIS is on D:\inetpub, registered service runs from D:\vondorservice)
--lookback days --globallookback	Only check the elements changed or created during the last X days in all available modules (reduces the scan duration significantly)

5.4 Parameters Better Avoided

This list contains flags often selected and used that should better be avoided unless you know exactly what you're doing.

Parameter	Description
--intense	long runtime, stability issues due to disabled resource control
--c2-in-memory	many false positives on user workstations (especially browser memory)
--alldrives	long runtime, stability issues due to scan on network drives or other remote filesystems
--mft	stability issues due to high memory usage
--dump-procs	stability issues, possibly high disk space usage (free disk space checks are implemented but may fail)
--full-registry	longer runtime, low positive impact

5.5 Lesser Known Parameters the Analysts Often Use

This list contains flags that are often used by analysts to tweak the scan in useful ways.

Parameter	Description
--allreasons	Show all reasons that led to a certain score
--printshim	Print all available SHIM cache entries into the log
--utc	Print all timestamps in UTC (helpful when creating timelines)
--string-context num-chars	Number of characters preceeding and following the string match to show in the output

5.6 Help and Debugging

You can use the following parameters to

Parameter	Description
--debug	Get debug information if errors occur
--help	Get a help with the most important scan options
--fullhelp	Get a help with all scan options

5.7 Examples

5.7.1 Logging to a Network Share

The following command creates a plaintext log file on a share called "rep" on system "sys" if the user running the command has the respective access rights on the share.

```
thor64.exe --nohtml --nocsv -l \\sys\rep\%COMPUTERNAME%\_thor.txt
```

5.7.2 Logging to Syslog Server

The following command instructs THOR to log to a remote syslog server only.

```
thor64.exe --nohtml --nocsv --nolog -s syslog.server.net
```

5.7.3 Scan Run on a Single Directory

```
thor64.exe --lab -p C:\ProgramData  
thor64.exe --lab -p I:\mounted\_image\disk1
```

IMPORTANT: This feature requires a [forensic lab license](#) type which is meant to be used in forensic labs.

You can imitate a lab scan without a lab license with these command line flags:

```
thor64.exe -a Filescan --intense --norescontrol --nosoft --cross-platform --alldrives -p  
↪C:\ProgramData
```

5.7.4 Deactivate all file output - Syslog only

```
thor64.exe -s 10.1.5.14 --nohtml --nolog --nocsv
```

5.7.5 Save the result files to a different directory

```
thor64.exe -s 10.1.5.14 -e Z:\
```

5.7.6 Only scan the last 7 days of the Windows Eventlog and log files on disk

```
thor64.exe --lookback 7
```

5.7.7 Scan System with Defaults and Make a Surface Scan

By default, the surface scan (DeepDive) applies all YARA rules in `./custom-signatures` folder. In this example all output files are written to a network share.

```
thor64.exe --deepdivecustom -e \\server\share\thor_output\
```

5.7.8 Intense Scan and DeepDive on a Mounted Image as Drive Z

```
thor64.exe --lab --deepdive -p Z:\
```

IMPORTANT: Lab scanning mode requires a [forensic lab license](#) type which is meant to be used in forensic labs.

You can achieve a similar (but not equal) scan using:

```
thor64.exe -a FileScan --intense -p C:\path-to-scan
```

5.7.9 Throttled THOR Run (static throttling value)

Will restrict THOR's CPU usage in the long running modules "FileScan", "Eventlog", "LogScan" and "Registry" to 60%. Note that THOR automatically applies certain restrictions in automatic soft mode.

```
thor64.exe -c 60
```

5.7.10 Scan Multiple Paths

```
thor64.exe --lab -p C:\\ D:\\webapps E:\\inetpub
```

(non-existent directories will be automatically skipped)

5.7.11 Scan All Hard Drives (Windows Only)

```
thor64.exe --allhds
```

5.7.12 Don't Scan Recursively

To instruct THOR to scan a folder non-recursively use the **:NOWALK** suffix.

```
thor64.exe -a FileScan -p C:\Windows\System32:NOWALK
```

5.8 Run a Scan with Specific Modules

With the parameter **-a** you can run a single module or select a set of modules that you'd like to run.

Valid modules are:

Autoruns, DeepDive, Dropzone, EnvCheck, Filescan, Firewall, Hosts, LoggedIn, OpenFiles, ProcessCheck, UserDir, ServiceCheck, Users, AtJobs, DNSCache, Eventlog, HotfixCheck, LSASessions, MFT, Mutex, NetworkSessions, NetworkShares, RegistryChecks, Rootkit, SHIMCache, ScheduledTasks, WMIShutdown

Run a Rootkit check only:

```
thor64.exe -a Rootkit
```

Run the Eventlog and file system scan:

```
thor64.exe -a Eventlog -a Filescan
```

5.9 PE-Sieve Integration

THOR integrates [PE-Sieve](#), an open-source tool by @hasherezade to check for malware masquerading as benevolent processes.

PE-Sieve can be activated by using the **--processintegrity** flag. It runs on Windows as part of the ProcessCheck module and is capable of detecting advanced techniques such as Process Doppelganging.

When investigating likely infections, you can also raise the sensitivity of the integrated PE-Sieve beyond the default with **--full-proc-integrity** (at the cost of likely false positives).

THOR reports PE-Sieve results as follows:

Findings	THOR's Reporting Level
Replaced PE File Implanted PE File	Warning
Unreachable File Patched IAT Hooked	Notice
Others	No Output in THOR

See the [PE-Sieve documentation](#) for more details on these values.

5.10 Multi-Threading

THOR 10.6 supports scanning a system with multiple threads in parallel, allowing for a significant increase in speed in exchange for a higher CPU usage.

To use this feature, use the **--threads** flag which allows you to specify THORs number of parallel threads.

When using the **--lab** (Lab Scanning), **--dropzone** (sample drop zone) or **--thunderstorm** (Thunderstorm) command line flags, THOR will default to using as many threads as the system has CPU cores; otherwise, THOR will still default to running with a single thread.

5.10.1 Enabled Modules

Not all modules support multi-threading. It is currently enabled in: File, Registry, Eventlog scanning and Thunderstorm and Dropzone service mode.

SCAN MODES

You can select between 6 different scan modes in THOR:

- **Default**

We recommend using the default scan mode for all sweeping activity. Scans take from 1 to 6 hours depending on the partition size and number of interesting files.

In default mode, THOR automatically chooses "Soft" mode if the system has only limited CPU and RAM resources.

There's a special "Lab Scanning" (**--lab**) method described in section 9.1 that disables many limitations and allows to scan mounted images in a Lab scenario, even with multiple THOR instances on a single Workstation.

- **Quick (--quick)**

This mode is the fastest one and oriented on the "Pareto Principe", covering 80% of the modules and check in 20% of the normal scan time. In "quick" mode, THOR skips elements that have not been created or modified within the last 2 days in the "Eventlog", "Registry" and "Filescan" modules. A set of 40+ directories will still be checked completely (e.g. AppData, Recycler, System32). "Quick" mode is known to be the "preventive" scan mode – less intense and very fast.

Themed scan modes:

- **Soft (--soft / force disable with --nosoft)**

This mode disables all modules and checks that could be risky for system stability.

It is automatically activated on (more details in *chapter Automatic Soft Mode*):

- Systems with only a single CPU core
- Systems with less than 1024 MB of RAM

- **Lab Scan (--lab)**

This mode scan only the filesystem and disable all other modules. (see *chapter Special Scan Modes* for more details and flags used in this scan mode)

Example: `./thor64 --lab -p /mnt/image_c/`

- **Intense (--intense)**

This mode is meant for system scanning in a non-productive or lab environment. It disables several speed optimizations and enables time-consuming extra checks for best detection results.

- **Difference (--diff)**

The Diff Mode looks for a last scan and last finished modules in the local THOR DB and scans only elements on disk that have been changed or created since the last scan start. This mode applies short-cuts to the "Filesystem", "Eventlog" and "Registry" modules. Diff scans are typically the shortest scans but require a completed previous scan. This scan mode is also susceptible to the so-called "Timestomping".

The following tables give an overview on the active modules and features in the different scan modes. The 'modules' section lists all available modules, whereas the 'features' section lists only features that are handled differently in the different scan modes.

6.1 Modules

6.1.1 OS Module Overview

Module	Windows	Linux	MacOS
File System Scan	Supported	Supported	Supported
Registry Scan	Supported	Not Supported	Not Supported
SHIM Cache Scan	Supported	Not Supported	Not Supported
Mutex Check	Supported	Not Supported	Not Supported
Named Pipes Check	Supported	Not Supported	Not Supported
DNS Cache Check	Supported	Supported	Supported
Hotfix Check	Supported	Not Supported	Not Supported
Hosts File Check	Supported	Supported	Supported
Firewall Config Check	Supported	Supported	Not Supported
Network Share Check	Supported	Not Supported	Not Supported
Logged In Check	Supported	Supported	Supported
Process Check	Supported	Supported*	Supported*
Service Check	Supported	Supported	Not Supported
Autoruns Check	Supported	Supported	Supported
Rootkit Check	Supported	Supported	Not Supported
LSA Sessions Analysis	Supported	Not Supported	Not Supported
User Account Check	Supported	Supported	Supported
User Profile Check	Supported	Supported	Supported
Network Sessions Check	Supported	Not Supported	Not Supported
Scheduled Tasks Analysis	Supported	Not Supported	Not Supported
WMI Startup Check	Supported	Not Supported	Not Supported
At Entries Check	Supported	Not Supported	Not Supported
MFT Analysis	Supported	Not Supported	Not Supported
Eventlog Analysis	Supported	Not Supported	Not Supported
KnowledgeDB Check	Not Supported	Not Supported	Supported
Environment Variables Check	Supported	Supported	Supported
Crontab Check	Not Supported	Supported	Not Supported
Integrity Check	Not Supported	Supported	Not Supported
Event Check	Supported	Not Supported	Not Supported

* = No process memory scan with YARA rules

6.1.2 Scan Mode Overview

Module	Normal	Quick	Soft	Intense
File System Scan		Reduced		
Registry Scan				
SHIM Cache Scan				
Mutex Check			Disabled	
Named Pipes Check				
DNS Cache Check				
Hotfix Check		Disabled		
Hosts File Check			Disabled	
Firewall Config Check		Disabled	Disabled	
Network Share Check			Disabled	
Logged In Check	Enabled*		Disabled	
Process Check			Reduced**	
Service Check				
Autoruns Check				
Rootkit Check				
LSA Sessions Analysis			Disabled	
User Account Check	Enabled*			
User Profile Check	Enabled*	Disabled		
Network Sessions Check	Enabled*		Disabled	
Scheduled Tasks Analysis				
WMI Startup Check				
At Entries Check				
MFT Analysis	Disabled	Disabled	Disabled	Enabled
Eventlog Analysis		Disabled		
KnowledgeDB Check				
Environment Variables Check				
Crontab Check				
Integrity Check				
Event Check				

* = Disabled on Domain Controllers

** = No process memory scan with YARA rules

6.2 Features

6.2.1 Feature Scan Mode Overview

Feature	Normal	Quick	Soft	Intense
Sigma Scan	Disabled	Disabled	Disabled	Enabled
EXE Decompression	Enabled**	Enabled	Disabled	Enabled
Archive Scan	Enabled	Enabled	Enabled	Enabled
Double Pulsar Check	Enabled**	Enabled	Disabled	Enabled
Groups XML Analysis	Enabled	Enabled	Enabled	Enabled
Vulnerability Check	Enabled	Enabled	Enabled	Enabled
Web Server Dir Scan	Enabled	Disabled	Enabled	Enabled
WMI Persistence	Enabled	Enabled	Enabled	Enabled
Registry Hive Scan	Enabled*	Enabled	Enabled	Enabled
AmCache Analysis	Enabled	Enabled	Enabled	Enabled
Process Handle Check	Enabled	Enabled	Enabled	Enabled
Process Connections Check	Enabled	Enabled	Enabled	Enabled
Windows Error Report (WER)	Enabled	Enabled	Enabled	Enabled
Windows At Job File Analysis	Enabled	Enabled	Enabled	Enabled
EVTX File Scanning	Enabled	Disabled	Enabled	Enabled
Prefetch Library Scanning	Enabled	Enabled	Enabled	Enabled
Memory Dump DeepDive	Disabled	Disabled	Disabled	Enabled
Text Log File Scanning	Enabled	Disabled	Enabled	Enabled
Shellbag Entry Analysis	Enabled	Enabled	Enabled	Enabled
Authorized Key File Analysis	Enabled	Enabled	Enabled	Enabled
Bifrost File Upload	Enabled	Enabled	Enabled	Enabled
Malicious Domain Check	Enabled	Enabled	Enabled	Enabled
File Scan	Enabled	Enabled	Enabled	Enabled
Cobalt Strike Beacon Parsing	Enabled	Enabled	Enabled	Enabled
Process Integrity Check	Disabled	Disabled	Disabled	Enabled**
SHIM Cache Analysis	Enabled	Enabled	Enabled	Enabled

* = Disabled on Domain Controllers

** = Only supported on Windows

6.2.2 Feature caller list

The following table gives an overview of THORs features and how they are called by the different modules and other features.

Feature	Callers
Sigma Scan	Eventlog, Log file scanning
EXE Decompression	File Scan
Archive Scan	File Scan
Double Pulsar Check	Rootkit Check
Groups XML Analysis	File Scan
Vulnerability Check	File Scan
Web Server Dir Scan	Process Check
WMI Persistence	File Scan
Registry Hive Scan	File Scan
AmCache Analysis	File Scan
Process Handle Check	Process Check
Process Memory Check	Process Check
Process Connections Check	Process Check
Windows Error Report (WER)	File Scan
Windows At Job File Analysis	File Scan
EVTX File Scanning	File Scan
Prefetch Library Scanning	File Scan
Memory Dump DeepDive	File Scan
Text Log File Scanning	File Scan
Shellbag Entry Analysis	Registry Hive Scan
Authorized Key File Analysis	File Scan
Bifrost File Upload	File Scan
Malicious Domain Check	File Scan
File Scan	Most modules and features
Cobalt Strike Beacon Parsing	File Scan, Process Check
Process Integrity Check	Process Check
SHIM Cache Analysis	SHIM Cache Scan, Registry Hive

SPECIAL SCAN MODES

7.1 Lab Scanning (--lab)

Lab scanning mode that is activated with **--lab** (formerly **--fsonly**). It is used to scan mounted forensic images or a single directory on a forensic workstation. All resource control functions are disabled and intense mode is activated by default.

The **--lab** parameter automatically activates the following other options:

- intense (scan every file intensively regardless of its extension or magic header)
- norescontrol (do not limit system resources or interrupt scan on low memory)
- nosoft (do not automatically activate soft mode on systems with single core CPUs or low memory)
- nodoublecheck (do not check for other THOR instances on the same system and do not interrupt scan if another instance has been found)
- multi-threading (it automatically sets the number of threads to use to the number of CPU cores found on the workstation)

The chapter *Use Cases* contains some use cases in which this scan mode is used. You may find the guides useful.

7.1.1 Forensic Lab License

Lab scanning mode and its features are usually required in a non-private context. It requires a [forensic lab license](#) which is meant to be used in corporate forensic labs.

You can achieve a similar (but not equal) scan using the following command line flags

```
thor64.exe -a Filescan --intense --norescontrol --nosoft --cross-platform --alldrives -p ↵  
↵path-to-scan
```

Without a valid lab license, you cannot use multiple instances of THOR on a single system or switch into multi-threaded scanning. The features mentioned in the following sub chapters are also limited.

7.1.2 Virtual Drive Mapping

Since THOR enriches messages with more details, it could be problematic to scan a mounted drive “s:”, which has originally been a partition “c:” on the source system of the image.

E.g. The analyst has mounted a partition “C:” from a source system to drive “F:” on the forensic lab workstation. A SHIMCache entry points to “C:\temp\mk.exe”. THOR would look at location “C:\temp\mk.exe” for that file and couldn’t find anything, since that file doesn’t exist on the forensic lab workstation.

Virtual drive mapping allows you to virtually map that drive to its original name. The syntax is as follows:

```
--virtual-map current-location:original-location
```

Some examples:

A original partition “C:” from the source system has been mounted to drive “F:” on the forensic lab workstation:

```
--virtual-map F:C
```

A original mount point “/” has been mounted to “/mnt/image1” on a Linux forensic lab workstation:

```
--virtual-map /mnt/image1:/
```

A Windows image of drive “C:” mounted to “/mnt/image1” on a Linux forensic lab workstation:

```
--virtual-map /mnt/image1:C
```

IMPORTANT: This feature requires a [forensic lab license](#) type which is meant to be used in forensic labs.

7.1.3 Hostname Replacement in Logs

The parameter **-j** can be used to set the hostname used in the log files to a given identifier instead of using the current workstation's name in all output files. If you don’t use this flag, all log files generated on that forensic lab workstation would contain the name of the forensic lab workstation as the source.

You should use the name of the host from which the image has been retrieved as the value for that parameter.

```
-j orig-hostname
```

7.1.4 Examples

A full command line of a THOR scan started in a lab environment would look like this:

```
thor64.exe --lab -p S:\\ --virtual-map S:C -j WKS001 -e C:\\reports
```

It instructs THOR to scan the mounted partition S: in lab scanning mode, maps the current partition “S:” to a virtual drive “C:”, replaces the hostname with “WKS001” in the outputs and saves every output file (text log, HTML, CSV) to a reports folder named “C:\reports”.

IMPORTANT: This feature requires a [forensic lab license](#) type which is meant to be used in forensic labs.

7.2 Lookback Mode (--lookback --all-module-lookback)

The **--lookback** option allows you to restrict the Eventlog and log file scan to a given amount of days. E.g. by using **--lookback 3** you instruct THOR to check only the log entries that have been created in the last 3 days.

In THOR v10.5 we've extended this feature to include all applicable modules, including "FileScan", "Registry", "Services", "Registry Hives" and "EVTX Scan".

By setting the flags **--all-module-lookback --lookback 2** you instruct THOR to scan only elements that have been created or modified during the last 2 days. This reduces the scan duration significantly.

This scan mode is perfect for quick scans to verify SIEM related events and is used by default in THOR Cloud's settings for executions via Microsoft Defender ATP.

7.3 Drop Zone Mode (--dropzone)

The drop zone mode allows you to define a folder on your local hard drive that is monitored for changes. If a new file is created in that folder, THOR scans this file and writes a log message if suspicious indicators have been found. The optional parameter **--dropdelete** can be used to remove the dropped file once it has been scanned. Example:

```
thor.exe --dropzone -p C:\\dropzone
```

IMPORTANT: This feature requires a [forensic lab license](#) type which is meant to be used in forensic labs.

7.4 Image File Scan Mode (-m)

The image file scan mode has a misleading name. It isn't meant to be used for forensic image scanning but for the scan of un-mountable images or memory dumps only. If you have a forensic image of a remote system, it is always recommended to mount the image as a Windows drive and scan it using the Lab Scanning (--lab) mode.

The Image File Scan mode performs a deep dive on a given data file. Therefore, the file type, structure or size of that file is not relevant. The DeepDive module processes the file in overlapping 3 Megabyte chunks and checks these chunks using the given YARA rule base only (including custom YARA signatures).

The only suitable use case is the scan of a memory dump using your own YARA signatures placed in the "./custom-signatures/yara" sub folder.

```
thor.exe -m systemX123.mem -j systemX123 -e C:\\reports
```

IMPORTANT: This feature requires a [forensic lab license](#) type which is meant to be used in forensic labs.

7.5 DeepDive (--deepdive)

The DeepDive module allows a surface scan of a given drive.

This check processes every byte of the whole hard drive including the free space. This enables THOR to detect deleted files that have not been wiped by the attackers.

DeepDive is not recommended for triage sweeps in a whole network as it generates more false positives than a normal file system scan. This is mainly caused by the fact that chunks of data read from the disk are processed regardless of their corresponding file's type, name or extension. It processes Antivirus signatures, pagefile contents and other data that may trigger an alert.

In the current stage of development, the DeepDive check parses out every executable file and applies all included Yara signatures. A positive match is reported according to the score as "Notice", "Warning" or "Alert".

There are some disadvantages linked with the DeepDive detection engine:

- The file name cannot be extracted from the raw executable code
- The file path of the reported sample is unknown

THOR uses other attributes to report these findings:

- Offsets
THOR reports the location on the disk, so that forensic investigators are able to check and extract the file from an image of the hard drive.
- Restore
THOR is able to restore the whole file to a given directory. It uses the system's NetBIOS name, rule name, the score and the offset to create a file name for the extracted file.

As a side effect of this dissection all the embedded executables in other file formats like RTF or PDF are detected regardless of their way of concealment.

To perform a surface scan, use the "**-a deepdive**" option. To restore all detected files to a restore directory additionally use the "**-r directory**" option.

Option	Description
-a deepdive	Activate DeepDive for the File System Scan. Only applicable if scan target is a drive – default or with selected drive root, i.e. "-p D:\\"
-r directory	Recovery directory for files found by DeepDive

While the DeepDive detects suspicious files regardless of their master file table reference the default file system scan that is executed afterwards may detect the same file twice.

The following example for the use of the DeepDive shows how to scan a mounted file system image as drive "X:".

```
thor --lab --deepdive -rd D:\\restore -p X:\\
```

7.6 Eventlog Analysis (-n)

The Eventlog scan mode allows scanning certain Windows Eventlogs.

The parameter **-n** works like the **-p** parameter in the Filesystem module. It takes the target Eventlog as parameter, which is the Windows Eventlog's full name.

```
thor.exe -a Eventlog -n "Microsoft-Windows-Sysmon/Operational"
```

You can get the full name of a Windows Eventlog by right clicking the Eventlog in Windows Event Viewer and selecting "Properties".

The **-n** parameter can also be used to restrict the Eventlog scanning to certain Eventlogs. The following command will start a default THOR scan and instructs the Eventlog module to scan only the "Security" and "System" Eventlog.

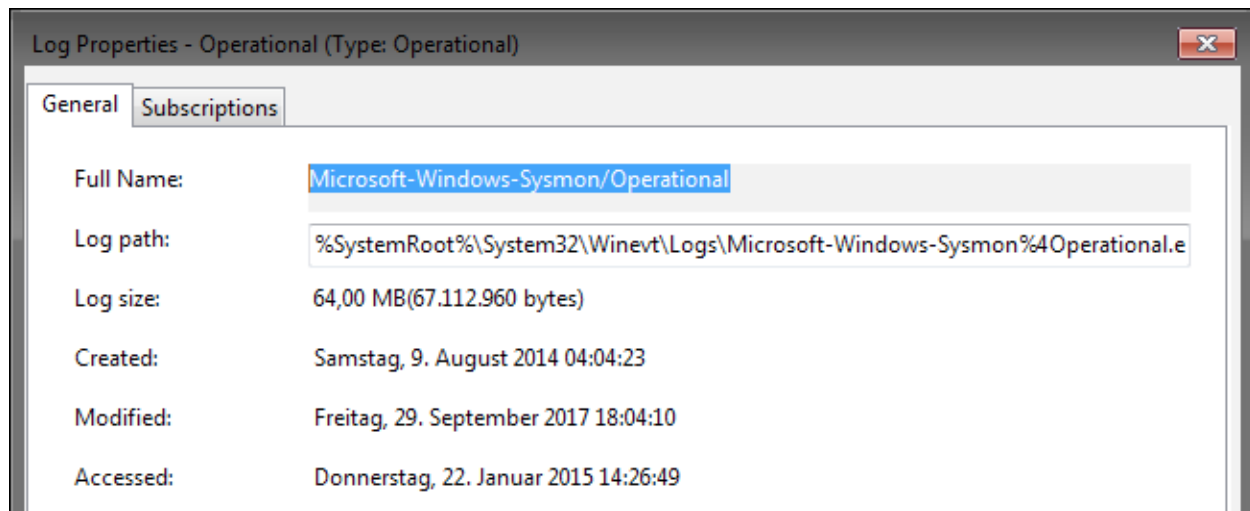


Fig. 1: Windows Eventlog Properties

```
thor.exe -n Security -n System
```

7.7 MFT Analysis (--mft)

The MFT analysis module reads the "Master File Table" (MFT) of a partition and parses its contents. The MFT analysis takes a significant amount of time and is only active in "intense" scan mode by default.

You can activate MFT analysis in any mode by using **--mft**.

The way THOR handles the MFT Analysis can be influenced by the following parameters:

Option	Description
--mft	Activate MFT analysis
--nomft	Do not perform any MFT analysis whatsoever (only useful in combination with --intense)
--maxmftsize MB	The maximum MFT size in Megabytes to process (default: 200 MB)

ANALYSIS

This chapter explains the possibilities for collecting and analyzing THOR logs.

8.1 ASGARD Analysis Cockpit

The ANALYSIS COCKPIT is the central platform for analyzing THOR logs. It can be used in an environment where scans are controlled by ASGARD Management Center and can also be used where THOR is executed manually or controlled by third party solutions. It is available as a virtual appliance on VMWare and also as a dedicated hardware appliance.

THOR can also be seen or used as hunting solution, it is optimized to avoid false negatives – meaning optimized to not miss an indicator of compromise. On the other side this clearly leads to more anomalies and false positives being reported.

In a scenario where you scan your infrastructure frequently you would either be seeing the same anomalies again and again or you would need to create many rules to filter out these anomalies in order to save analysis time.

The ANALYSIS COCKPIT is designed to facilitate this process and help you generate these rules automatically, so that you can set your baseline-filters after the first scan. After setting the first baseline it is now easy to focus on relevant Alerts and Warnings as only differences between the first and second scans are shown.

The ANALYSIS COCKPIT comes with an integrated and highly configurable ticketing system that helps organizing your analysis workflow. Furthermore, the ANALYSIS COCKPIT comes with a rule based alert forwarding and SIEM integration that makes it easy for your organization to react quickly on new incidents.

8.2 Splunk

We offer a THOR Splunk App and Add-on via the official Splunk App Store. This App helps you to extract the event fields and provides dashboards to get a better overview on distributed runs on multiple systems.

THOR APT Scanner App

<https://splunkbase.splunk.com/app/3717/>

THOR Add-On

<https://splunkbase.splunk.com/app/3718/>

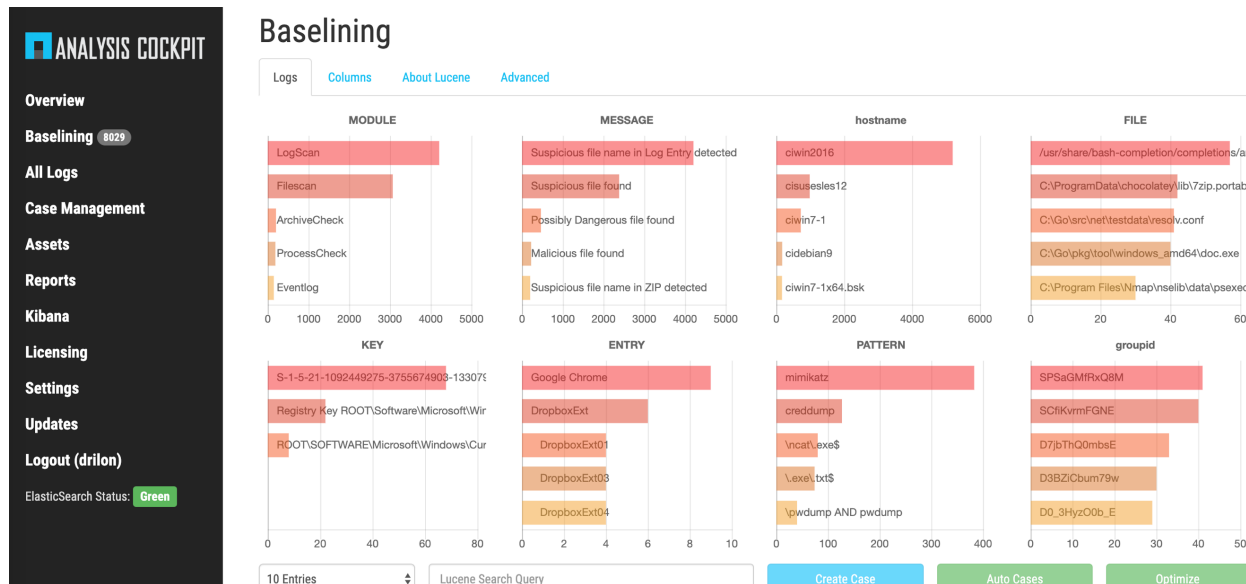


Fig. 1: Analysis Cockpit View

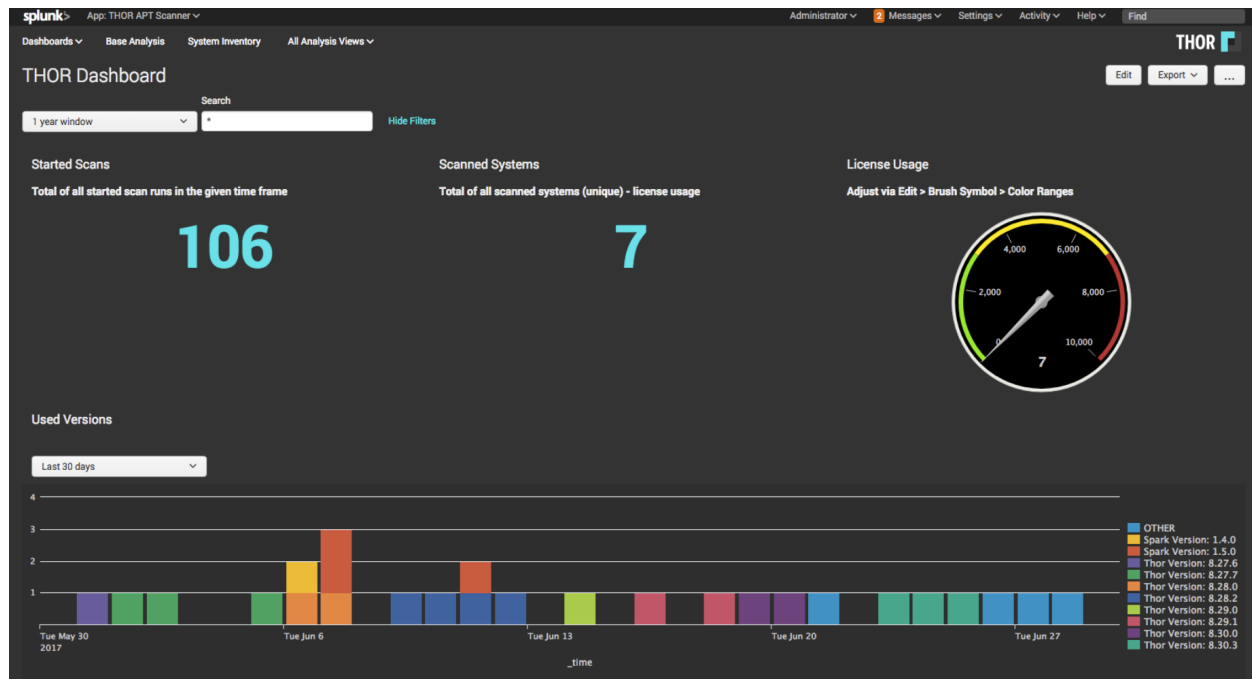


Fig. 2: THOR Splunk App (free)

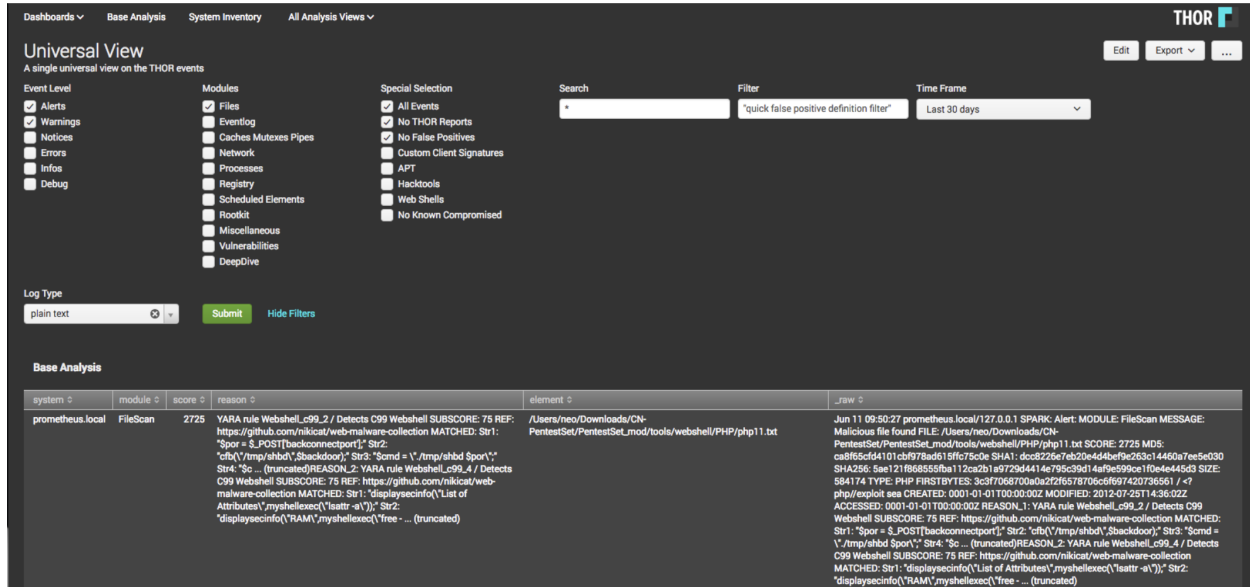


Fig. 3: Splunk THOR App Universal View

8.3 THOR Util Report Feature

THOR Util provides a feature called “report” that creates HTML reports from text logs of one or more scanned systems. Find more information about this feature on our website or the separate THOR Util manual.

<https://www.nextron-systems.com/2018/06/20/thor-util-with-html-report-generation/>

8.4 Log Analysis Manual

The Customer Portal contains a log analysis manual in the “Downloads” section.

<https://portal.nextron-systems.com/webshop/downloads>

Scan Information						Modules		Statistics	
PROMETHEUS	476	119	26	553	1	ArchiveCheck	24	Alerts	522
TRINITY	14	40	21	127	1	Log	1	Warnings	183
ALPHA	0	15	19	148	1	PESieve	1	Notice	113
PROMETHEUS	0	1	18	140	1	DeepDive	61	Info	1048
PROMETHEUS	1	0	6	13	0	Autoruns	427	Errors	4
prometheus.local	15	4	15	30	0	LogScan	491	Help Shortcuts Use Ctrl+↑ (Windows/Linux) or ⌘+↑ (macOS) to return to the top of the page Filters You can provide a file (--filter file) with regular expressions to suppress false positives Hint 1 Some header elements contain links to the sections below Hint 2 Values contain links to search engines	
metusalem	16	4	8	37	0	VulnerabilityCheck	5		
						Registry	455		
						Filescan	185		
						Amcache	4		
						SHIMCache	18		
						FileScan	2		
						RegistryChecks	18		
						ProcessCheck	52		
Alerts									
Alert 1	PROMETHEUS	2018-06-12T13:00:15Z PROMETHEUS/10.0.2.4 MODULE: RegistryChecks MESSAGE: Malicious filename found SCORE: 80 DESC: FEIB Heist - BAE Report https://goo.gl/8LbqZ9 ELEMENT: USBSERV C:\Windows\Temp\bitsran.exe \C:\Windows\Temp\bitsran.exe OBJECT: RUN Key							
Alert 2	PROMETHEUS	2018-06-12T08:15:32Z PROMETHEUS/10.0.2.4 MODULE: Filescan MESSAGE: Malicious file found FILE: C:\hp\Mailing_Oxford_Label.exe SCORE: 80							

Fig. 4: THOR Util's Report Output

THOR

Our APT Scanner THOR is the only flexible tool on the market that is able to evaluate the full extend of security incidents within your corporate networks in order to treat them appropriately. In contrast to common Antivirus solutions THOR focuses on the detection of attacker activity. While well-known Antivirus solutions are configured to detect malware like trojans, worms and some types of exploit code, THOR performs a deep system analysis using more than 25 modules to reveal hidden attacker activity in log files, typical attacker tools, anomalies within the user accounts, sessions, error reports, dump files, network connections and many other check items.

[Download for Windows](#)

[Download Manual](#)

[Download Log-Analysis Manual](#)

Fig. 5: Log Analysis Manual Download Link

CONFIGURATION

9.1 Scan Templates

THOR 10 accepts config files (called “templates”) in YAML format. They reflect all command options to make them flexible and their use as comfortable as possible.

This means that every parameter set via command line can be provided in the form of a config file. You can even combine several of these config files in a single scan run.

9.1.1 Default Template

By default, THOR only applies the file named **thor.yml** in the **./config** sub folder. Other config files can be applied using the **-t** command line parameter.

9.1.2 Apply Custom Scan Templates

The following command line provides a custom scan template named **mythor.yml**.

```
thor.exe -t mythor.yml
```

9.1.3 Example Templates

The default config **thor.yml** in the **./config** folder has the following content.

Content of THOR's Default Config 'thor.yml':

```
# This is the default config for THOR
# Terminate THOR if he runs longer than 72 hours
max_runtime: 72
# Minimum score to report is 40
min: 40
# Skip files bigger than 120000000 bytes
max_file_size: 120000000
# Skip files bigger than 300000000 bytes in intense mode (--fsonly, --intense)
max_file_size_intense: 300000000
# Limit THOR's CPU usage to 95%
cpulimit: 95
# The minimum amount of free physical memory to proceed (in MB)
minmem: 50
```

(continues on next page)

(continued from previous page)

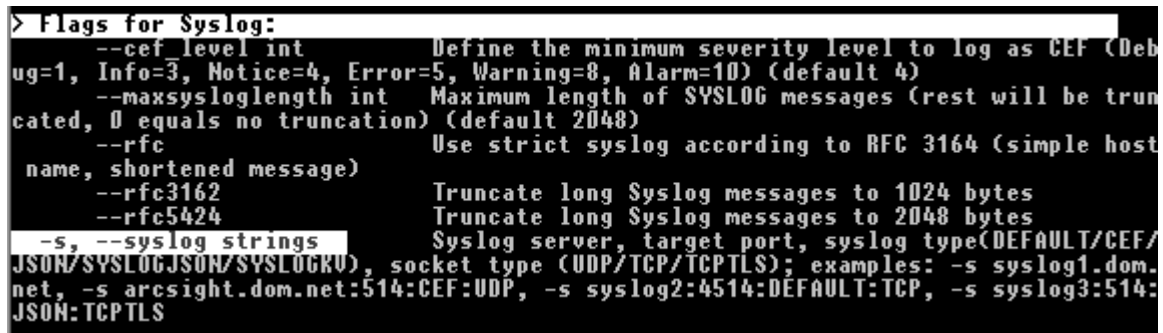
```
# Truncate THOR's field values after 2048 characters
truncate: 2048
```

Content of Config File 'mythor.yml':

```
resume: true
cpulimit: 40
intense: true
max_file_size: 7500000
syslog:
  - foo.nextron
  - bar.nextron:514:TCP
```

The default scan template is always applied first. Custom templates can then overwrite settings in the default template. In the example above, the **cpulimit** and **max_file_size** parameters are overwritten by the custom template.

As you can see in the example file, you have to use the long form of the command line parameter (e.g. **syslog**) and not the short form (e.g. **s**) in the template files. The long forms can be looked up in the command line help using **--help**.



```
> Flags for Syslog:
--cef_level int          Define the minimum severity level to log as CEF (Debug=1, Info=3, Notice=4, Error=5, Warning=8, Alarm=10) (default 4)
--maxsysloglength int    Maximum length of SYSLOG messages (rest will be truncated, 0 equals no truncation) (default 2048)
--rfc                    Use strict syslog according to RFC 3164 (simple host name, shortened message)
--rfc3162                Truncate long Syslog messages to 1024 bytes
--rfc5424                Truncate long Syslog messages to 2048 bytes
-s, --syslog strings     Syslog server, target port, syslog type(DEFAULT/CEF/JSON/SYSLOGJSON/SYSLOGKRV), socket type(UDP/TCP/TCP/TLS); examples: -s syslog1.dom.net, -s arcsight.dom.net:514:CEF:UDP, -s syslog2:4514:DEFAULT:TCP, -s syslog3:514:JSON:TCP/TLS
```

Fig. 1: Lookup command line parameter long forms using **--help**

9.2 Maximum File Size

The default maximum file size for deeper investigations (hash calculation and YARA scanning) is 20 MB. The maximum file size for the **"intense"** scan mode is 100 MB.

You can adjust the values in **./config/thor.yml**. This file does not get overwritten by an update or upgrade.

Special scan features like the EVTX or Memory Dump scan ignore these limits.

Features that obey the file size limit:

- YARA Matching
- Hash calculation
- STIX IOC application
- ArchiveScan

Features that ignore the file size limit:

- LogScan
- RegistryHive scanning

- EVTX scanning
- DeepDive on memory dumps (selected by .dmp and magic headers)
- Filename IOCs
- YARA meta rules (only check the first 100 bytes of a file and all meta data)

If the `--intense` flag is used, a different file size limit is applied.

The only exception is ArchiveScan (e.g. ZIP file analysis) that has no file size limit in intense scan.

9.2.1 Chunk Size in DeepDive

The chunk size in DeepDive module is set to the value defined as **maximum file size**. DeepDive uses overlapping chunks of this size for YARA rule scanning.

Example: If the maximum file size is set to a default of 12 MB, DeepDive use the following chunks in its scan to apply the YARA rule set:

```
Chunk 1: Offset 0 - 12
Chunk 2: Offset 6 - 18
Chunk 3: Offset 12 - 24
Chunk 4: Offset 18 - 30
```

9.3 Exclude Elements

9.3.1 Files and Directories

You may use the file "**directory-excludes.cfg**" to exclude directories and files(! The name of the config file is misleading) from the scan.

THOR will not scan the contents of these directories but it will still perform some basic checks on file names in these directories. This "**directory-excludes.cfg**" config is meant to avoid scanning sensitive files like databases or directories with a lot of content. If you want to suppress false positives that are generated in these directories, please see the following chapter and how to suppress them by using "**false_positive_filters.cfg**".

The exclusion file contains regular expressions that are applied to each scanned element. Each element consists of the file path and file name (e.g. C:\IBM\temp_tools\custom.exe). If one of the defined expressions matches, the element is excluded. Exclusions can be defined for a full element name, at the beginning at the end or somewhere in the element name.

As the configured exclusions are treated as regular expressions, special characters must be masqueraded by backslash. This applies at least for: `[ \ ^ $ . | ? * + () -`

Element to exclude	Possible solution
C:\IBM\temp_tools\custom.exe	C:\IBM\temp_tools\
Log folder of the tool "hpsm" regardless on the partition	\\HPSM\log\
Every file with the extension .nsf	\.nsf\$
THOR custom signatures	\\THOR\custom\signatures\

9.3.2 Eventlogs

Eventlog sources can be excluded as whole in "**eventlog-excludes.cfg**". The file holds one expression per line and applies them as regular expression on the name of the Eventlog. (e.g. "Microsoft-Windows-Windows Defender/Operational")

Element to exclude	Possible solution
Windows PowerShell	Windows PowerShell
Microsoft-Windows-Windows Defender/Operational	Windows Defender

9.3.3 Registry

Registry paths/keys can be excluded in "**registry-excludes.cfg**". The file holds one expression per line and applies them as regular expression on each registry key. (e.g. "Software\WOW6432Node"). Don't include the root of the key, e.g. HKLM.

Element to Exclude	Exclude Definition
HKEY_LOCAL_MACHINE\Software\Wow6432Node\Symantec\Symantec Endpoint Protection\AV\Exclusions	Symantec Endpoint Protection\AV\Exclusions

9.3.4 False Positives

The false positive filters work like the directory/file excludes. A regular expression is applied to the full content of the "MESSAGE:" value.

E.g. if you want to Exclude all messages that contain the string "**Trojan_Buzus_dev**" you just add this string to the "**false_positive_filters.cfg**" file. The file works with regular expressions so you could also define something like "**chinese_(charcode|keyboard)**".

9.3.5 Filter Verification

If you are unsure about the filters you just set, we recommend a test run on a certain directory that matches the criteria.

You can start a short test run on a certain directory with:

```
thor.exe -a FileScan --intense -p C:\\TestDir
```

9.3.6 Personal Information

THOR features an option named **--brd** that allows to filter the output messages and replace all known locations and fields that can contain user names or user ids with the value **"ANONYMIZED_BY_THOR"**.

What it does is:

- Replace all "USER" and "OWNER" field values of all modules with the anonymized string value
- Replaced the subfolder names of "C:\Users" and "C:\Documents and Settings" with the anonymized string value

There is no guarantee that all user IDs will be removed by the filter, as they may appear in the most unexpected locations, but in most cases this approach is sufficient to comply with data protection requirements.

OUTPUT OPTIONS

10.1 Scan Output

THOR creates several files during and at the end of the scan.

Real Time - the text log file is written during the scan process. Also the SYSLOG output is sent in real-time to one or more remote systems.

End of Scan - the full HTML report and CSV file with all file scan elements reported as suspicious are written at the end of the scan.

You can define different formatting options for each the FILE and the SYSLOG output.

10.1.1 Placeholders

Two placeholders can be used in command line parameters to facilitate the use of parameter on different operating systems.

- :hostname:
- :time:

These can be used in command line parameters and scan templates across all platforms.

```
thor64.exe -a FileScan -p S:\\ -o :hostname:\\_:time:.csv
```

10.1.2 Log File Output (.txt)

The standard log file is written by default.

- **--nolog**
Don't create a log file
- **-l / --logfile filename**
Set a filename for the log file

The log file's format aligns with the format of SYSLOG messages. This way it can easily be imported to most SIEM or log analysis systems.

10.1.3 CSV Output (.csv)

The CSV output is an optional legacy output file without much details. It contains only “Filescan” module findings and consist of 3 columns, file hash, file path and score.

CSV File Output:

```
c926bf384319e40506e3d6e409dc856e,C:\PowerZure.ps1,140
62160f1a71507e35ebf104a660d92794,C:\f.bat,180
c926bf384319e40506e3d6e409dc856e,C:\ntds.dit,50
c926bf384319e40506e3d6e409dc856e,C:\temp\ntds.zip\ntds.dit,140
36a93511fc0e2e967bc5ced6a5bc36a6,C:\temp\ntds.zip,50
44b34aac3135dcb03ababac5f7767a55,C:\temp\windows-hardening.bat,60
```

Be aware that archives with matches show up as “archive.zip|file-with-finding.js” (pipe separator) in the second column. If you need more columns in that CSV, consider processing the JSON output instead.

Note: our Github repository contains scripts to convert THOR’s JSON output into a CSV with any given field values, see:

<https://github.com/NextronSystems/nextron-helper-scripts/tree/master/thor-log-processors>

10.1.4 CSV Stats

The CSV stats file is an optional output file that contains only the scan statistics. It contains a single line with:

Hostname, scan start, scan end, THOR version, used command line flags, number of alerts, number of warnings, number of notices and number of errors

CSV Stats Output:

```
HYPERION,2021-02-17 17:01:25,2021-02-17 17:01:28,10.6.2,--lab -p C:\temp -o HYPERION_test_:time:.csv --
csvstats,5,2,3,0
```

10.1.5 JSON Output (.json)

The JSON output file can be configured with these options:

- **--json**
Create a JSON output file
- **--jsonfile filename**
Set a filename for the JSON log file
- **--cmdjson**
Print JSON format into the command line (e.g. used with Splunk scripted input)
- **-s [syslogtarget]:[port]: SYSLOGJSON**
Send syslog messages with JSON formatting

10.1.6 Key Value Output

THOR provides the option to create a "Key/Value" pair output that simplifies the SIEM integration.

By using the "**--keyval**" option you get the text and syslog output transformed as shown in the following example. The command line output stays untouched by this setting.

There are three different Key Value Pair Formatting flags:

- **--keyval**
Write key/value pairs to the log file
- **--cmdkeyval**
Print key/value pairs in the command line (e.g. used with Splunk scripted input)
- **-s [syslogtarget]:[port]:SYSLOGKV**
Send syslog messages with proper key/value formatting

Default - Without "--keyval" parameter

```
Jul 10 09:08:47 PROMETHEUS/10.0.2.15 THOR: Alert: MODULE: SHIMCache MESSAGE: Malware
name found in Shim Cache Entry ENTRY: C:\Users\neo\Desktop\ncat.exe KEYWORD: \\ncat.exe
DATE: 07/29/13 05:16:04 TYPE: system HIVEFILE: None EXTRAS: N/A N/A True
```

Key/Value Pairs - With "--keyval" parameter

```
Jul 10 09:07:59 PROMETHEUS/10.0.2.15 THOR : Alert: MODULE="SHIMCache" MESSAGE="Malware
name found in Shim Cache Entry" ENTRY="C:\Users\neo\Desktop\ncat.exe" KEYWORD="\\ncat.exe"
DATE="07/29/13 05:16:04" TYPE="system" HIVEFILE="None" EXTRAS="N/A N/A True"
```

10.1.7 Timestamps

Timestamp in all modules use the ANSIC standard, which looks like:

```
Mon Jan 2 15:04:05 2006
Mon Mar 19 09:04:05 2018
```

<https://flaviocopes.com/go-date-time-format>

UTC

The `--utc` parameter allows to use UTC in all timestamps.

RFC3339 Time Stamps

The parameter `--rfc3339` generates time stamps for UTC time in the format described in RFC 3339. In contrast to the default time stamps RFC 3339 timestamps include a year and look like this:

2017-02-31T23:59:60Z

10.1.8 SCAN ID

The former parameter “-i”, which has been used for so-called case IDs (CID) has been repurposed to allow users to set a certain scan ID (SCANID) that appears in every log line.

The scan ID helps SIEM and analysis systems to correlate the scan lines from multiple scans on a single host. Otherwise it would be very difficult to answer the following questions:

- How many scans completed successfully on a certain end system?
- Which scan on a certain end system terminated during the scan run?

If no parameter is set, THOR will automatically generate a random scan ID, which starts with an “S-” and contains the following characters: **a-zA-Z0-9_-**

Users can overwrite the scan ID with “-i myscanid” to assign the logs of multiple scan runs to a single logical scan, e.g. if multiple partitions of a system get scanned in the lab in different scan runs, but should be shown as a single scan in Analysis Cockpit or your SIEM of choice.

Examples:

S-Rooa61RfuM S-0vRKu-1_p7A

In a log line, it looks like:

Jul 10 09:08:47 PROMETHEUS/10.0.2.15 THOR: Alert: MODULE: SHIMCache
SCANID: S-r4GhEhEiRg MESSAGE: Malware name found in Shim Cache Entry ENTRY:
C:\Users\neo\Desktop\ncat.exe KEYWORD: \ncat.exe DATE: 07/29/13 05:16:04 TYPE: system
HIVEFILE: None EXTRAS: N/A N/A True

Custom Scan ID Prefix

Since version 10.5 you are able to set your custom prefix by using **--scanid-prefix**. The fixed character “S” can be replaced with any custom string. This allows users to set an identifier for a group of scans that can be grouped together in a SIEM or Analysis Cockpit.

10.2 Syslog or TCP/UDP Output

10.2.1 Target Definition

THOR version 10 comes with a very flexible Syslog target definition. You can define as many targets as you like and give them different ports, protocols and formats.

For example, if you want to send the THOR log entries to a Syslog server and an ArcSight SIEM at the same time, you just have to define two log targets with different formats.

```
thor.exe -s syslog1.server.net -s arcsight.server.net:514:CEF
```

The definition consists of 4 elements:

System	:	Port	:	Type	:	Protocol
--------	---	------	---	------	---	----------

The available options for each element are:

```
(target ip):(target port):(DEFAULT/CEF/JSON/SYSLOGJSON/SYSLOGKV):(UDP/TCP/TCPTLS)
```

The available type field values require an explication:

- DEFAULT: standard THOR log format
- CEF: Common Event Format (ArcSight)
- JSON: Raw JSON
- SYSLOGJSON: encoded and escaped single line JSON
- SYSLOGKV: syslog messages that contain strict key/value pairs

There are default values, which do not have to be defined explicitly:

```
(your target system ip):514:DEFAULT:UDP
```

Sending Syslog to a target on a port that differs from the default port 514/udp looks like this:

```
-s 10.0.0.4:2514
```

Sending Syslog to a receiving server using an SSL/TLS encrypted TCP connection:

```
-s 10.0.0.4:6514:DEFAULT:TCPTLS
```

You can define as many targets as you like.

An often used combination sends JSON formatted messages to a certain UDP port:

```
-s 10.0.0.4:5444:JSON:UDP
```

10.2.2 Common Event Format (CEF)

THOR supports the CEF format for easy integration into ArcSight SIEM systems. The CEF mapping is applied to a log line if the syslog target has the CEF format set, e.g.:

```
thor.exe -s syslog1.server.local:514:CEF
```

10.3 Encrypted Output Files

THOR allows to encrypt the output files of each scan using the **--encrypt** parameter. A second parameter **--pubkey** can be used to specify a public key to use. The public key must be a RSA key of 1024, 2048 or 4096 bit size in PEM format.

```
thor64.exe --encrypt --pubkey mykey.pub
```

If you don't specify a public key, THOR uses a default key. The private key for this default key is stored in "thor-util", which can be used to decrypt output files encrypted with the default key.

```
thor-util decrypt file.txt
```

For more information on "thor-util" see the separate [THOR Util manual](#).

You can download updates for THOR with “thor-util”.

Running "**thor-util --help**" shows two options that look very similar:

- **upgrade** : program and signature updates
- **update** : signature updates only

For more information on "thor-util" see the separate [THOR Util manual](#).

11.1 Update Locations

The following servers are used as update mirrors and should be accessible via HTTPS (443/tcp):

- update1.nexttron-systems.com
- update2.nexttron-systems.com

11.2 Update Server Information

You can get information on the available update packages on this site:

<https://update1.nexttron-systems.com/info.php>

CUSTOM SIGNATURES

THOR checks the contents of the **./custom-signatures** folder and processes every file in this folder. The extension determines the type of signature (e.g. a simple IOC file, a YARA rule, a Sigma rule, ...). For some signature types, string tags in the file names are used to further distinguish the signatures.

For example, a **my-c2-iocs.txt** file will be initialized as a file containing simple IOC indicators with C2 server information.

Internally the regex `\Wc2\W` is used to detect the tag, so **"mysource-c2-iocs.txt"** and **"dec15-batch1-c2-indicators.txt"** would be detected correctly, whereas on the contrary **"filenameiocs.txt"** or **"myc2iocs.txt"** would not.

If you do not wish to place your custom IOCs on potentially compromised systems during an engagements, you can use `thor-util` to encrypting custom signatures. This is described in detail in the [THOR Util manual](#)

12.1 Simple IOCs

Simple IOC files are basically CSV files that include the IOC and comments. The folder **./custom-signatures/iocs/templates** contains template files from which you can create your own IOC files.

Simple IOC files must have the extension **".txt"**. Encrypted simple IOC files must have the extension **".dat"**.

The following tags for simple IOCs are currently supported:

- **"c2"** or **"domains"** for C2 server IOCs like IPs and host names
- **"filename"** or **"filenames"** for file name IOCs
- **"hash"** or **"hashes"** for MD5, SHA1, SHA256 hash IOCs
- **"keyword"** or **"keywords"** for string based keywords
- **"trusted-hash"** or **"trusted-hashes"** or **"falsepositive-hash"** or **"falsepositive-hashes"** for hashes that you trust (also expects CSV format in the form **"hash;comment"** like the hash IOCs)
- **"handles"** for malicious Mutex / Event IOCs
- **"pipes"** or **"pipe"** for Named Pipe IOCs

Tag in File Name	Example
c2	misp-c2-domains-iocs.txt
filename	Case-UX22-filename-iocs.txt
filenames	Malicious-filenames-unitX.txt
hash	op-aura-hash-iocs.txt
hashes	int-misp-hashes.txt
keyword	keywords-incident-3389.txt
keywords	Incident-22-keyword.txt
trusted-hash	my-trusted-hashes.dat (encrypted)
handles	Operation-fallout-handles.txt
pipes	incident-22-named-pipes.txt

12.1.1 Hashes

Files with the tag "hash" or "hashes" in their name get initialized as hash IOC sets. Every match with one of these hashes receives a sub score of 100.

You can add MD5, SHA1 or SHA256 hashes and add a comment in a second column, which is separated by a semicolon.

```
1  #
2  # THOR CUSTOM HASH IOCs
3  # This file contains MD5, SHA1 and SHA256 hashes and a short info like file name
4  # or hash origin
5  #
6  # Important: Rename this template file from .txt.template to .txt
7  #
8  # FORMAT -----
9  #
10 # MD5;COMMENT
11 # SHA1;COMMENT
12 # SHA256;COMMENT
13 #
14 # EXAMPLES -----
15 #
16
17 0c2674c3a97c53082187d930efb645c2;DEEP PANDA Sakura Malware - http://goo.gl/R3e6eG
18 000c907d39924de62b5891f8d0e03116;The Darkhotel APT http://goo.gl/DuS7WS
19
```

Fig. 1: Example hash IOC file

12.1.2 File Name IOCs

Filename IOC files allow you to define IOCs based on file name and path using regular expressions. You can add or reduce the total score of a file element during the scan with a positive (e.g. "40") or negative score (e.g. "-30").

While this can also be used to define false positives, or reduce the score of well-known files and locations, it gives you all the flexibility to add scores according to your needs.

Filename IOCs are case insensitive if they don't use any special regex characters (such as *, ., [, ...). Otherwise, they are case sensitive by default, but can be set as case insensitive by using (?i) anywhere in the regex.

For example, if you know that administrators in your organization use "PsExec.exe" in a folder "Sysinternals" and any other location should be reported as suspicious you could define the following statements:

```

1  #
2  # THOR File Name Characteristics
3  # This file contains regex definitions and a score
4  #
5  # APPLICATION -----
6  #
7  # Every line is treated as REGEX case sensitive.
8  # Every line includes a score that adds up to a total score for this module.
9  #
10 # FORMAT -----
11 #
12 # # COMMENT
13 # REGEX;SCORE to add
14 #
15 # EXAMPLES -----
16 #
17 # # Various examples from APT case X
18 # \\svcsstat\.exe;75
19 # \\(server|services|smrr|srrm|svchost|svhost|svshost|taskmgr)\.exe$;25
20 # ProgramData\\Mail\\MailAg\\;40
21 # (Anwendungsdaten|Application Data|APPDATA)\\sydmain\\.dll;55
22 # (TEMP|Temp)\\[^\\]+\\. (xmd|yls)$;45
23 # (LOCAL SETTINGS\\Temp|Local Settings\\Temp|Local\\Temp)\\(word\\.exe|winword\\.ex
24 #
25 # NOTICE -----
26 # Notice that you can overwrite an exact regex definition within the THOR
27 # signatures with your own definition. That means that if there is a definition
28 # for "\\ncat\\.exe", you can overwrite the score given by THOR by defining
29 # exactly the same regex.
30 # You are able to the reduce the score given by THOR by negative score
31 # definitions like "\\bin\\ncat\\.exe;-40" to exclude ncats that are located in
32 # a typical directory of e.g. a Cygwin installation. If "ncat" is detected in a
33 # different directory, the reduction will not take place.
34 #
35 # Ncat Example
36 bin\\nc\\.exe;-20

```

Fig. 2: File "filename-iocs.txt"

```
\\\\\\PsExec\\.exe;60
\\\\\\SysInternals\\\\\\PsExec\\.exe;-60
```

This following example represents the 3rd generation filename IOC format introduced with THOR version 8.30 and SPARK version 1.5, which is now the recommended form to define such signatures.

It contains three fields:

- Column 1: Regex
- Column 2: Score
- Column 3: False Positive Regex

The False Positive Regex statement is only evaluated if the Regex statement in column 1 matched.

```
\\\\PsExec\\.exe;60;\\\\SysInternals\\\\
```

We use this new format internally to describe abnormal locations of system files like

```
[C-Zc-z:\\|\\\\\\\\\\\\\\\\|.}{1,40}\\\\svchost\\.exe;65;(?(i)(HKCR\\\\\\\\Applications\\System32\\  
↪|system32\\SYSTEM32\\winsxs\\WinSxS\\SysWow64\\SysWow64\\syswow64\\SYSNATIVE\\  
↪|Sysnative\\dllcache\\WINXP\\WINDOWS\\i386\\%system32%)\\\\\\\\
```

You could also score down directories with many false positives reported as "Notices" or "Warnings" like this:

```
\\\\\\directory\\_with\\_many\\_false\\_positives\\\\\\;-30
```

12.1.3 Keyword IOCs

The keyword-based IOC files contain plain strings that are matched against the output lines of almost every module (e.g. Eventlog entries, log lines, Autoruns elements, local user names, at jobs etc.)

Every line is treated as case-sensitive string. A comment can be specified with a line starting with a # and applies to all following IOCs until another comment is encountered.

Keyword IOCs are case sensitive.

```
#
# THOR Keyword Signatures
# This file contains string definitions
#
# Important: Rename this template file from .cfg.template to .cfg or .txt
#
# APPLICATION -----
#
# Every line is treated as plain string case sensitive
#
# FORMAT -----
#
# # COMMENT
# STRING
#
# EXAMPLES -----
#
# # Evil strings from our case
# sekurlsa::logonpasswords
# failed to create Service 'GAMEOVER'
# kiwi.eo.oe
#

# My Keyword Test
sekurlsa::logonpasswords
sekurlsa::LogonPasswords
```

Fig. 3: Keyword IOC Example

12.1.4 C2 IOCs

C2 IOC files specify remote servers which are known to be malicious. This can include:

- Domain names
- FQDNs
- Single IPs
- IP address ranges in CIDR notation

These IOCs are applied to the connections of examined processes and can optionally be used to search process memory.

Each IOC must be placed on a single line. A comment can be specified with a line starting with a # and applies to all following IOCs until another comment is encountered. A score for the IOC can optionally be specified after the IOC, separated by a ;, it defaults to 100 if none is specified.

```
# OpMuhadib C2 servers
182.34.23.10;90
update1.usul.ru
usul-updates.info
182.34.23.0/24
```

Example for custom C2 IOCs

12.1.5 Mutex or Event Values

Custom mutex or event values can be provided in a file that contains the “handles” keyword in its filename. The entries can be string or regular expression values. The entries are applied to the processes handles as “equals” if no unescaped special regex characters are used, otherwise they are applied as “contains” (though a regex can, of course, specify its match position by using ^ and/or \$).

You can decide if you want to set a scope by using Global\\ or BaseNamedObjects\\ as a prefix. If you decide to use none, your expression will be applied to any scope.

Mutex and event IOCs are case sensitive.

```
Global\\mymaliciousmutex;Operation Fallout - RAT Mutex
Global\\WMI\\_CONNECTION\\_RECV;Flame Event https://bit.ly/2KjUTuP
Dwm-[a-f0-9]{4}-ApiPort-[a-f0-9]{4};Chinese campaign malware June 19
```

Example for custom Mutex IOCs

12.1.6 Named Pipes

Custom named pipe values can be provided in a file that contains the “pipes” keyword in its filename. The entries should be regular expressions that match the malicious named pipes. The \\\\.\\pipe\\ prefix should not be part of the entry. The IOCs are applied to the pipes as “equals” if no unescaped special regex characters are used, otherwise they are applied as “contains” (though a regex can, of course, specify its match position by using ^ and/or \$).

Optionally, a score can be added as 2nd field. If none is present, it defaults to 100.

Named Pipe IOCs are case insensitive.

```
MyMaliciousNamedPipe;Malicious pipe used by known RAT
MyInteresting[a-z]+Pipe;50;Interesting pipe we have seen in new malware
```

Example for custom Named Pipe IOCs

12.2 Sigma Rules

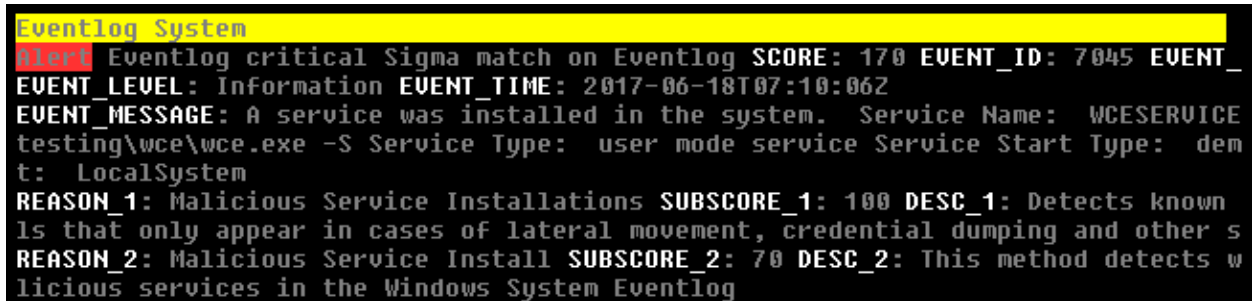
Sigma is a generic rule format for detections on log data. Sigma is for log data, as Snort is for network packets and YARA is for files.

THOR applies Sigma rules to Windows Eventlogs and log files on disk (*.log). By default, THOR ships with the public Sigma rule set, which is maintained by the community at <https://github.com/SigmaHQ/sigma>.

To activate Sigma scanning, you have to use the **--sigma** command line option or perform an **--intense** scan. Sigma scanning is not activated by default. This behavior may change in the future.

By default only the results of Sigma rules of level critical and high are shown. If called with the **--intense** flag, medium level rules are applied as well.

Custom Sigma rules must have the **.yaml** extension for unencrypted sigma rules and the **.ymls** extension for encrypted sigma rules.



```
Eventlog System
Alert Eventlog critical Sigma match on Eventlog SCORE: 170 EVENT_ID: 7045 EVENT_
EVENT_LEVEL: Information EVENT_TIME: 2017-06-18T07:10:06Z
EVENT_MESSAGE: A service was installed in the system. Service Name: WCESERVICE
testing\wce\wce.exe -S Service Type: user mode service Service Start Type: dem
t: LocalSystem
REASON_1: Malicious Service Installations SUBSCORE_1: 100 DESC_1: Detects known
ls that only appear in cases of lateral movement, credential dumping and other s
REASON_2: Malicious Service Install SUBSCORE_2: 70 DESC_2: This method detects w
licious services in the Windows System Eventlog
```

Fig. 4: Example Sigma match on Windows Eventlog

12.2.1 Sigma Examples

Perform a scan with the Sigma rules on the different local Windows Eventlogs (-a Eventlog)

```
thor64.exe -a Eventlog --sigma
```

Perform a scan with the Sigma rules on logs of Linux systems (-a LogScan) only

```
thor64 -a Filesystem -p /var/log -sigma
```

12.3 STIX IOCs

THOR can read and apply IOCs provided in STIXv2 JSON files. They must have the **.json** extension for unencrypted STIXv2 files and the **.jsos** extension for encrypted STIXv2 files.

The following observables are supported.

- **file:name** with = != LIKE and MATCHES
- **file:parent_directory_ref.path** with = != LIKE and MATCHES
- **file:hashes.sha-256 / file:hashes.sha256** with = and !=

```

Info Init Adding rule set from thor-all.yas as 'default' type
Info Init Adding rule set from thor-keywords.yas as 'keyword', 'log' and 'registr
Info Init Adding rule set from thor-log-sigs.yas as 'log' type
Info Init Adding rule set from thor-registry.yas as 'registry' type
Info Init Rule Set Sizes default: 1, log: 2, reg: 2, key: 1, prc: 1
Info Init Reading STIXv2 indicators ...
Info Init Adding STIXv2 indicators from misp.stix2.event11964.json
Info Init Adding STIXv2 indicators from misp.stix2.event12096.json
Info Init Adding STIXv2 indicators from misp.stix2.event12117.json
Info Init Successfully compiled 6 / 48 STIXv2 indicators

Initializing Active Modules
Info Init Active modules: FileScan, LogScan
Info Init Active features: DumpScan, STIX, Bifrost, RegistryWalk, YARA

```

Fig. 5: STIXv2 Initialization during startup

- file:hashes.sha-1 / file:hashes.sha1 with = and !=
- file:hashes.md-5 / file:hashes.md5 with = and !=
- file:size with < <= > >= = !=
- file:created with < <= > >= = !=
- file:modified with < <= > >= = !=
- file:accessed with < <= > >= = !=
- win-registry-key:key with = != LIKE and MATCHES
- win-registry-key:values.name with = != LIKE and MATCHES
- win-registry-key:values.data with = != LIKE and MATCHES
- win-registry-key:values.modified_time with < <= > >= = !=

12.3.1 STIX v1

STIX version 1 is not supported.

12.4 YARA Rules

THOR offers an interface to include own rules based on the YARA format. YARA rules must have the **.yar** extension for unencrypted YARA rules and the **.yas** extension for encrypted YARA rules.

There are two custom YARA rule types that you can define in THOR:

- Generic Rules
- Specific Rules

12.4.1 Generic YARA Rules

All YARA rules which do not contain any specific tag (see *Specific YARA Rules*) are considered generic YARA rules.

The generic YARA rules are applied to the following elements:

- Files
THOR applies the Yara rules to all files that are smaller than the size limit set in the **thor.yml** and matches specific rules. *Additional Attributes* are available.
- Process Memory
THOR scans the process memory of all processes with a working set memory size up to a certain limit. This limit can be altered by the "**--max_process_size**" parameter.
- Data Chunks
The rules are applied to the data chunks read during the DeepDive scan. DeepDive only reports and restores chunks if the score level of the rule is high enough to cause at least a warning.

The following table shows in which modules the Generic YARA rules are applied to content.

Applied in Module	Examples
Filescan, ProcessCheck, DeepDive	incident-feb17.yar misp-3345-samples.yar

12.4.2 Specific YARA Rules

The specific YARA rules contain certain tags in their filename to differentiate them further:

- Registry Keys
Tag: **'registry'**
Rules are applied to a whole key with all of its values. See *THOR YARA Rules for Registry Detection* for more details.
- Log Files
Tag: **'log'**
Rules are applied to each log entry. See *THOR YARA Rules for Log Detection* for more details.
- Process Memory
Tag: **'process'** or **'memory'**
Rules are applied to process memory only.
- All String Checks
Tag: **'keyword'**
Rules are applied to all string checks in many different modules.
- Metadata Checks (since THOR 10.6)
Tag: **'meta'**
Rules are applied to all files without exception, including directories, symlinks and the like, but can only access the THOR specific external variables (see *Additional Attributes*) and the first 100 bytes of the file.
Since THOR 10.6.8: If a metadata rule has the special tag DEEPSCAN, THOR will apply the generic YARA rules (see *Generic YARA Rules*) to this file.

The following table shows in which modules the specific YARA rules are applied to content.

Tag in File Name	Applied in Module	Examples
registry	RegistryChecks, RegistryHive	incident-feb17-registry.yar
log	Eventlog, Logscan, EVTX	general-log-strings.yar
process	ProcessCheck (only on process memory)	case-a23-process-rules.yar
keyword	Mutex, Named Pipes, Eventlog, MFT, ProcessCheck (on all process handles), ProcessHandles, ServiceCheck, AtJobs, LogScan, AmCache, SHIMCache, Registry	misp-3345-keyword-extract.yar
meta	Filescan	meta-rules.yar

THOR YARA Rules for Registry Detection

THOR allows checking a complete registry path key/value pairs with Yara rules. To accomplish this, he composes a string from the key/value pairs of a registry key path and formats them as shown in the following screenshot.

```
-----
YARA CHECK ON REGISTRY KEY CONTENT:
-----
CMI-CreateHive{D43B12B8-09B5-40DB-B4F6-F6DFEB78DAEC}\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\4;(default);
CMI-CreateHive{D43B12B8-09B5-40DB-B4F6-F6DFEB78DAEC}\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\4;DisplayName;Restricted sites
CMI-CreateHive{D43B12B8-09B5-40DB-B4F6-F6DFEB78DAEC}\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\4;PMDDisplayName;Restricted sites [Protected Mode]
CMI-CreateHive{D43B12B8-09B5-40DB-B4F6-F6DFEB78DAEC}\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\4;Description;This zone contains Web sites that could potentially damage your computer or data.
CMI-CreateHive{D43B12B8-09B5-40DB-B4F6-F6DFEB78DAEC}\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\4;Icon;inetcpl.cpl#00004481
CMI-CreateHive{D43B12B8-09B5-40DB-B4F6-F6DFEB78DAEC}\Software\Microsoft\Windows\CurrentVersion\Internet Settings\Zones\4;LowIcon;inetcpl.cpl#005426
-----

Warning: Suspicious file name in Registry Value detected STRING: C:\TEMP\gsecdump.exe PATTERN: gsecdump\.exe KEY : CMI-CreateHive{D43B12B8-09B5-40DB-B4F6-F6DFEB78DAEC}\Software\Microsoft\Windows\CurrentVersion\Run HIVE: C:\Users\trinity\NTUSER.DAT
Alarm: Malicious registry key found KEY: CMI-CreateHive{D43B12B8-09B5-40DB-B4F6-F6DFEB78DAEC}\Software\Microsoft\Windows\CurrentVersion\Run NAME: xyz VALUE: evil.exe HIVE: C:\Users\trinity\NTUSER.DAT DESC: Test
YARA CHECK ON REGISTRY KEY CONTENT:
-----
CMI-CreateHive{D43B12B8-09B5-40DB-B4F6-F6DFEB78DAEC}\Software\Microsoft\Windows\CurrentVersion\Run;THOR Test;C:\TEMP\gsecdump.exe
CMI-CreateHive{D43B12B8-09B5-40DB-B4F6-F6DFEB78DAEC}\Software\Microsoft\Windows\CurrentVersion\Run;xyz;evil.exe
```

Fig. 6: Composed strings from registry key/value pairs

The composed format is:

```
KEYPATH;KEY;VALUE\n
KEYPATH;KEY;VALUE\n
KEYPATH;KEY;VALUE\n
```

Registry Base Names

Please notice that strings like HKEY_LOCAL_MACHINE, HKLM, HKCU, HKEY_CURRENT_CONFIG are **not** part of the key path that your YARA rules are applied to. They depend on the analyzed hive and should not be in the strings that you define in your rules.

Values are formatted as follows:

- REG_BINARY values are hex encoded with upper case.
- REG_MULTI_SZ values are printed with `\\0` separating the multiple strings.
- Numeric values are printed normally (with base 10).
- String values are printed normally.

This means that you can write a Yara rule that looks like this (remember to escape all backslashes):

```
rule Registry_DarkComet {
    meta:
        description = "DarkComet Registry Keys"
    strings:
        $a1 = "LEGACY_MY_DRIVERLINKNAME_TEST;NextInstance"
        $a2 = "\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run;MicroUpdate"
        $a3 = "Path;Value;4D5A00000001" # REG_BINARY value
        $a4 = "Shell\\Open;Command;explorer.exe\\0comet.exe" # REG_MULTI_SZ value
    condition:
        1 of them
}
```

Remember that you have to use the keyword **'registry'** in the file name in order to initialize the YARA rule file as registry rule set (e.g. "**registry_exe_in_value.yar**").

Registry scanning uses bulk scanning. See *Bulk Scanning* for more details.

THOR YARA Rules for Log Detection

YARA Rules for logs are applied as follows:

- For text logs, each line is passed to the YARA rules.
- For Windows Event Logs, each event is serialized as follows for the YARA rules: Key1: Value1 Key2: Value2 ... where each key / value pair is an entry in EventData or UserData in the XML representation of the event.

Log (both text log and event log) scanning uses bulk scanning. See *Bulk Scanning* for more details.

12.4.3 How to Create YARA Rules

Using the UNIX "string" command on Linux systems or in a CYGWIN environment enables you to extract specific strings from your sample base and write your own rules within minutes. Use "**string -el**" to also extract the UNICODE strings from the executable.

A useful Yara Rule Generator called "yarGen" provided by our developers can be downloaded from Github. It takes a target directory as input and generates rules for all files in this directory and so called "super rules" if characteristics from different files can be used to generate a single rule to match them all. (<https://github.com/Neo23x0/yarGen>)

Another project to mention is the "Yara Generator", which creates a single Yara rule from one or multiple malware samples. Placing several malware files of the same family in the directory that gets analyzed by the generator will lead to a signature that matches all descendants of that family. (<https://github.com/Xen0ph0n/YaraGenerator>)

We recommend testing the Yara rule with the "yara" binary before including it into THOR because THOR does not provide a useful debugging mechanism for Yara rules. The Yara binary can be downloaded from the developers' website (<https://github.com/VirusTotal/yara>).

The best practice steps to generate a custom rule are:

1. Extract information from the malware sample (Strings, Byte Code, MD5 ...)
2. Create a new Yara rule file. It is important to:
 - a. Define a unique rule name – duplicates lead to errors
 - b. Give a description that you want to see when the signature matches
 - c. Define an appropriate score (optional but useful in THOR, default is 75)
3. Check your rule by scanning the malware with the Yara binary from the project's website to verify a positive match
4. Check your rule by scanning the "Windows" or "Program Files" directory with the Yara binary from the project's website to detect possible false positives
5. Copy the file to the "/custom-signatures/yara" folder of THOR and start THOR to check if the rule integrates well and no error is thrown

There are some THOR specific add-ons you may use to enhance your rules.

Also see these articles on how to write "simple but sound" YARA rules:

<https://www.bsk-consulting.de/2015/02/16/write-simple-sound-yara-rules/>

<https://www.bsk-consulting.de/2015/10/17/how-to-write-simple-but-sound-yara-rules-part-2/>

12.4.4 Typical Pitfalls

Some signatures - even the ones published by well-known vendors - cause problems on certain files. The most common source of trouble is the use of regular expressions with a variable length as shown in the following example. This APT1 rule published by the AlienVault team caused the Yara Binary as well as the THOR binary to run into a loop while checking certain malicious files. The reason why this happened is the string expression "\$gif1" which causes Yara to check for a "word character" of undefined length. Try to avoid regular expressions of undefined length and everything works fine.

AlienVault APT1 Rule:

```
rule APT1_WEBC2_TABLE {
  meta:
    author = "AlienVault Labs"
  strings:
    $msg1 = "Fail To Execute The Command" wide ascii
    $msg2 = "Execute The Command Successfully" wide
    $gif1 = /\w+\.gif/
    $gif2 = "GIF89" wide ascii
  condition:
    3 of them
}
```

Copying your rule to the signatures directory may cause THOR to fail during rule initialization. If this happens you should check your rule again with the Yara binary. Usually this is caused by a duplicate rule name or syntactical errors.

12.4.5 YARA Rule Performance

We compiled a set of guidelines to improve the performance of YARA rules. By following these guidelines you avoid rules that cause many CPU cycles and hamper the scan process.

<https://gist.github.com/Neo23x0/e3d4e316d7441d9143c7>

12.5 Enhance YARA Rules with THOR Specific Attributes

The following listing shows a typical YARA rule with the three main sections "meta", "strings" and "condition". The YARA Rule Manual which can be downloaded as PDF from the developer's website and is bundled with the THOR binary is a very useful guide and reference to get a function and keyword overview and build your own rules based on the YARA standard.

The "meta" section contains all types of meta information and can be extended freely to include own attributes. The "strings" section lists strings, regular expressions or hex string to identify the malware or hack tool. The condition section defines the condition on which the rule generates a "match". It can combine various strings and handles keywords like "not" or "all of them".

Simple Yara Rule:

```
rule simple_demo_rule_1 {
    meta:
        description = "Demo Rule"
    strings:
        $a1 = "EICAR-STANDARD-ANTIVIRUS-TEST-FILE"
    condition:
        $a1
}
```

The following listing shows a more complex rule that includes a lot of keywords used in typical rules included in the rule set.

Complex Yara Rule:

```
rule complex_demo_rule_1 {
    meta:
        description = "Demo Rule"
    strings:
        $a1 = "EICAR-STANDARD-ANTIVIRUS-TEST-FILE"
        $a2 = "li0n" fullword
        $a3 = /msupdate\\.(exe\\|dll)/ nocase
        $a4 = { 00 45 9A ?? 00 00 00 AA }
        $fp = "MSWORD"
    condition:
        1 of ($a*) and not $fp
}
```

The example above shows the most common keywords used in our THOR rule set. These keywords are included in the YARA standard. The rule does not contain any THOR specific expressions.

Yara provides a lot of functionality but lacks some mayor attributes that are required to describe an indicator of compromise (IOC) defined in other standards as i.e. OpenIOC entirely. Yara's signature description aims to detect any kind of string or byte code within a file but is not able to match on meta data attributes like file names, file path, extensions and so on.

THOR adds functionality to overcome these limitations.

12.5.1 Score

THOR makes use of the possibility to extend the Meta information section by adding a new parameter called "score".

This parameter is the essential value of the scoring system, which enables THOR to increment a total score for an object and generate a message of the appropriate level according to the final score.

Every time a signature matches the value of the score attribute is added to the total score of an object.

Yara Rule with THOR specific attribute "score":

```
rule demo_rule_score {
    meta:
        description = "Demo Rule"
        score = 80
    strings:
        $a1 = "EICAR-STANDARD-ANTIVIRUS-TEST-FILE"
        $a2 = "honkers" fullword
    condition:
        1 of them
}
```

Feel free to set your own "score" values in rules you create. If you don't define a "score" the rule gets a default score of 75.

The scoring system allows you to include ambiguous, low scoring rules that can't be used with other scanners, as they would generate to many false positives. If you noticed a string that is used in malware as well as legitimate files, just assign a low score or combine it with other attributes, which are used by THOR to enhance the functionality and are described in [chapter 12.5.2 Additional Attributes](#).

12.5.2 Additional Attributes

THOR allows using certain external variables in your generic and meta YARA rules.

These external variables are:

- **"filename"** - single file name like **"cmd.exe"**
- **"filepath"** - file path without file name like **"C:\temp"**
- **"extension"** - file extension with a leading ".", lower case like **".exe"**
- **"filetype"** - type of the file based on the magic header signatures (for a list of valid file types see: ["./signatures/misc/file-type-signatures.cfg"](#)) like **"EXE"** or **"ZIP"**
- **"timezone"** – the system's time zone (see https://golang.org/src/time/zoneinfo_abbrr_windows.go for valid values)
- **"language"** – the systems language settings (see <https://docs.microsoft.com/en-us/windows/win32/intl/sort-order-identifiers>)
- **"owner"** - The file owner, e.g. **"NT-AUTHORITY\SYSTEM"** on Windows or **"root"** on Linux

- **"group"** (available since THOR 10.6.8) - The file group, e.g. **"root"** on Linux. This variable is empty on Windows.
- **"filemode"** (available since THOR 10.6) - file mode for this file (see <https://man7.org/linux/man-pages/man7/inode.7.html>, "The file type and mode"). On Windows, this variable will be an artificial approximation of a file mode since Windows is not POSIX compliant.

The **"filesize"** value contains the file size in bytes. It is provided directly by YARA and is not specific to THOR.

Yara Rule with THOR External Variable:

```
rule demo_rule_enhanced_attribute_1 {
    meta:
        description = "Demo Rule - Eicar"
    strings:
        $a1 = "EICAR-STANDARD-ANTIVIRUS-TEST-FILE"
    condition:
        $a1 and filename matches /eicar.com/
}
```

A more complex rule using several of the THOR external variables would look like the one in the following listing.

This rule matches to all files containing the EICAR string, having the name **"eicar.com"**, **"eicar.dll"** or **"eicar.exe"** and a file size smaller 100byte.

Yara Rule with more complex THOR Enhanced Attributes.

```
rule demo_rule_enhanced_attribute_2 {
    meta:
        author = "F.Roth"
    strings:
        $a1 = "EICAR-STANDARD-ANTIVIRUS-TEST-FILE"
    condition:
        $a1 and filename matches /eicar\.(com|dll|exe)/ and filesize < 100
}
```

The following YARA rule shows a typical combination used in one of the client specific rule sets, which are integrated in THOR. The rule matches on **".idx"** files that contain strings used in the Java Version of the VNC remote access tool. Without the enhancements made this wouldn't be possible as there would be no way to apply the rule only to a special type of extension.

Real Life Yara Rule:

```
rule HvS_Client_2_APT_Java_IDX_Content_hard {
    meta:
        description = "VNCViewer.jar Entry in Java IDX file"
    strings:
        $a1 = "vncviewer.jar"
        $a2 = "vncviewer/VNCViewer.class"
    condition:
        1 of ($a*) and extension matches /\.idx/
}
```

12.5.3 Bulk Scanning

THOR scans registry and log entries in bulks since each YARA invocation has a relatively high overhead. This means that during the scan, the following happens:

- THOR gathers entries that need to be scanned.
- When sufficiently many entries are gathered, all of them are combined (separated by line breaks) and passed to YARA.
- If any YARA rule matches, each entry is scanned separately with YARA to determine whether any YARA rule matches for this specific entry.

One potential caveat of this is that false positive strings may prevent a rule from ever applying.

For example, consider this rule:

```
rule FakeMicrosoftStartupEntry {
  strings:
    $s1 = "Microsoft\\SomeRegistryKey;ShouldBeUsedOnlyByMicrosoft;"
    $fp = "Windows\\System32"
  condition:
    $s1 and not $fp
}
```

This rule is meant to match if the specified registry key contains some DLL that is not in C:\Windows\System32. However, the false positive string may inadvertently match on other entries in the bulk, like here:

```
Path\to\Microsoft\SomeRegistryKey;ShouldBeUsedOnlyByMicrosoft;C:\evil.exe
...
Path\to\SomeOtherRegistryKey;Entry;C:\Windows\System32\explorer.exe
...
```

Because the rule does not apply to the bulk, THOR never scans the single elements and does not report any match. Therefore, be very careful with false positive strings with log or registry YARA rules.

A possible workaround for this issue is to define the false positive strings in ways that they can't match anywhere else, e.g. like this:

```
rule FakeMicrosoftStartupEntry {
  strings:
    $s1 = "Microsoft\\SomeRegistryKey;ShouldBeUsedOnlyByMicrosoft;"
    $fp = /Microsoft\\SomeRegistryKey;ShouldBeUsedOnlyByMicrosoft;[^\\n]{0,40}
    ↪Windows\\System32/
  condition:
    $s1 and not $fp
}
```

12.5.4 Restrict Yara Rule Matches

On top of the keyword based initialization you can restrict Yara rules to match on certain objects only. It is sometimes necessary to restrict rules that e.g. cause many false positives on process memory to file object detection only. Use the meta attribute "type" to define if the rule should apply to file objects or process memory only.

Apply rule in-memory only: .. code-block:

```
rule Malware_in_memory {
  meta:
    author = "Florian Roth"
    description = "Think Tank Campaign"
    type = "memory"
  strings:
    $s1 = "evilstring-inmemory-only"
  condition:
    1 of them
}
```

Apply rule on file objects only:

```
rule Malware_in_fileobject {
  meta:
    description = "Think Tank Campaign"
    type = "file"
  strings:
    $s1 = "evilstring-infile-only"
  condition:
    1 of them
}
```

You can also decide if a rule should not match in "DeepDive" module by setting the "nodeepdive" attribute to "1".

Avoid DeepDive application:

```
rule Malware_avoid_DeepDive {
  meta:
    description = "Think Tank Campaign"
    nodeepdive = 1
  strings:
    $s1 = "evilstring-not-deepdive"
  condition:
    1 of them
}
```

If you have problems with false positives caused by the specific YARA rules, try using the "limit" modifier in the meta data section of your YARA rule. Using the "limit" attribute, you can limit the scope of your rules to a certain module. (Important: Use the module name as stated in the log messages of the module, e.g. "ServiceCheck" and not "services")

E.g. if you have defined a malicious 'Mutex' named '_evt_x_' in a rule and saved it to a file named "mutex-keyword.yar", the string "_evt_x_" will be reported in all other modules to which the keyword rules are applied – e.g. during the Eventlog scan.

You can limit the scope of your rule by setting 'limit = "Mutex"' in the meta data section of the YARA rule.

Limits detection to the "Mutex" module:

```
rule Malicious_Mutex_EvtX {
    meta:
        description = "Detects malicious mutex EVT_X"
        limit = "Mutex"
    strings:
        $s1 = "_evtX_"
    condition:
        1 of them
}
```

Notes:

- the internal check in THOR against the module name is case-insensitive
- this "limit" parameter only applies to specific YARA rules (legacy reasons – will be normalized in a future THOR version)

12.5.5 False Positive Yara Rules

Yara rules that have the "falsepositive" flag set will cause a score reduction on the respective element by the value defined in the "score" attribute. Do not use a negative score value in YARA rules.

False Positive Rule:

```
rule FalsePositive_AVSig1 {
    meta:
        description = "Match on McAfee Signature Files"
        falsepositive = 1
        score = 50
    strings:
        $s1 = "%%McAfee-Signature%%"
    condition:
        1 of them
}
```


OTHER TOPICS

13.1 Evidence Collection

13.1.1 Process Memory Dumps (--dump-procs)

Since THOR version 10.5 it supports process dumping to backup volatile malware information.

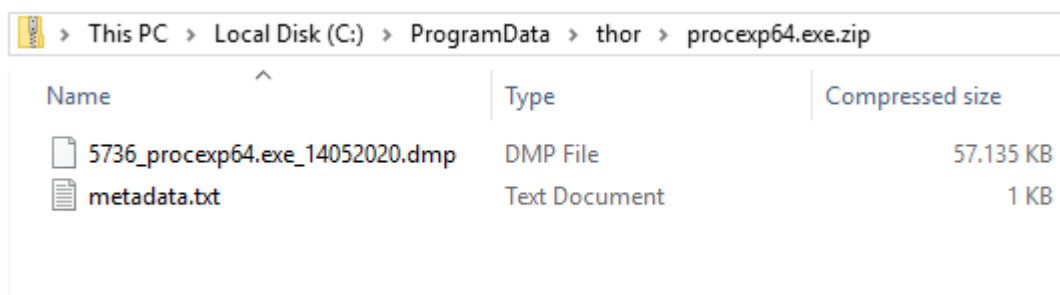
THOR on Windows creates a process dump of any process that is considered malicious. Maliciousness is determined as anything that triggers a warning or an alert.

Activate process memory dumping with "--dump-procs".

This process dump can then be analyzed with standard tools later on to examine the found malware.

```
Warning: ProcessCheck YARA rule match on process memory RULE: Example_Rule TAGS: DESC: Example r
ule to match a process SCORE: 75 REFERENCE: not set RULEDATE: 2019-05-14 PID: 5736 NAME: procexp6
4.exe CMD: "C:\Users\Max\Downloads\PROCEXP64.EXE" USER: DESKTOP-EEM5B52\Max MATCH_STRINGS: Str1:
"procexp"
Info: ProcessCheck Successfully dumped process PID: 5736 PPID: 3320 PARENT: NAME: procexp64.exe
OWNER: DESKTOP-EEM5B52\Max COMMAND: "C:\Users\Max\Downloads\PROCEXP64.EXE" PATH: C:\Users\Max\Do
wnloads\PROCEXP64.EXE
CREATED: Mon May 11 14:27:58 2020
MD5: 7e7eaa8aebc4026be3b56b965b0d8947 CONNECTION_COUNT: 0 LISTEN_PORTS: FILE_1: C:\Users\Max\Dow
nloads\PROCEXP64.EXE EXISTS_1: yes MD5_1: 7e7eaa8aebc4026be3b56b965b0d8947 SHA1_1: 57fe177df7e94b
a8495e1885c9b5946fa4312df3 SHA256_1: aac11d3ff8661e14a6d7073e44f0d6ccabc436856af5faf10e761c57e8b4
2f71 FIRSTBYTES_1: 4d5a90000300000004000000ffff0000b8000000 DUMP_FILE: C:\ProgramData\thor\procex
p64.exe.zip
Info: ProcessCheck Finished module TOOK: 0 hours 0 mins 9 secs
```

Fig. 1: Process dumping



Name	Type	Compressed size
5736_procexp64.exe_14052020.dmp	DMP File	57.135 KB
metadata.txt	Text Document	1 KB

Fig. 2: Process dumps on disk

To prevent flooding the disk fully in case many dumps are created, old dumps of a process are overwritten if a new dump is generated. Also, THOR will not generate dumps by default if less than 5 GB disk space is available. This can be overwritten to always or never dump malicious processes.

Also note that THOR will never dump lsass.exe to prevent these dumps from potentially being used to extract passwords by any attackers.

13.1.2 File Collection (Bifrost)

Bifrost v1 with Script-Based Server

The `./tools` folder in the program directory contains a simple Python based file collection server named Bifrost. The script is named **bifrost-server.py**.

You can run that script on any internal system with a Python script interpreter installed. By default, it uses port 1400/tcp for incoming connections but you can use any port you like.

Usage is:

usage: bifrost-server.py [-h] [-d out-dir] [-i ip] [-p port]

Bifrost

optional arguments:

- -h, --help show this help message and exit
- -d out-dir Quarantine directory
- -i ip IP address to bind to
- -p port Port to bind to (tcp, default 1400)

You can run the server script with:

```
python ./bifrost-server.py
```

In order to send suspicious file to that server, you have to set some command line flags when running THOR, e.g.

```
thor64.exe --bifrostServer myserver
```

A more complex statement setting a minimum score and custom port would look like this:

```
thor64.exe --bifrostServer myserver --bifrost-port 8080 --bifrostLevel 80
```

THOR will then try to submit all samples with score equal or higher than 80 to a Bifrost service running on myserver port 8080/tcp.

Bifrost v2 with ASGARD

Bifrost v2 cannot be used standalone yet. The required API Key is set by ASGARD v2 during initialization and is unknown to a THOR user.

You can activate the quarantine function via Bifrost v2 when creating a single or group scan via the ASGARD management interface.

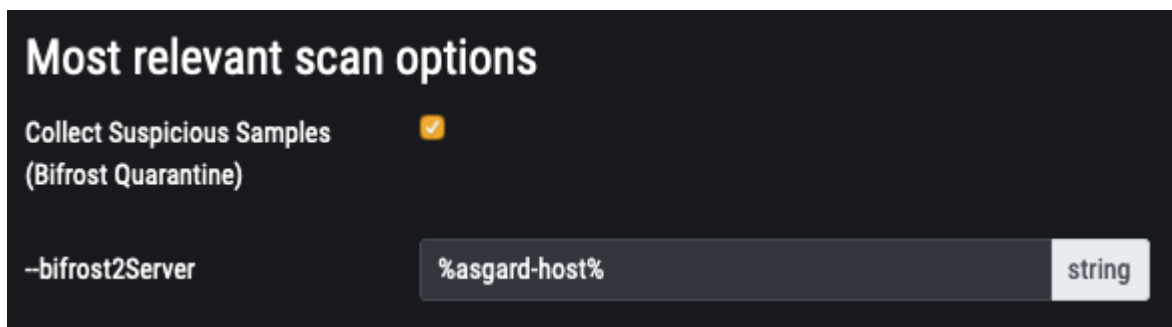


Fig. 3: Configure Quarantine via Bifrost in New Scan Dialogue

SHA256	Type	Affected Hosts	Filenames
b57bf397984545f419045391b56dcdf7b0bed8b6ee331b5c46cee35c92ffa13d	JSP	server-001.de.sales.nexttron-systems.com	zehir4.asp.php.txt
7d72ed0ef1b497619f12bc962512061d131c96cf9bccdd4a9a4345490e0a088c	JSP	server-001.de.sales.nexttron-systems.com	zehir4.php
ee256d7cc3ceb2bf3a1934d553cdd36e3fbde62a02b20a1b748a74e85d4dbd33	PHP	server-001.de.sales.nexttron-systems.com	safe0ver.php

Fig. 4: Collected File Evidence in ASGARD v2

13.2 Resource Control

THOR's internal resource control feature puts the system's stability and the responsiveness of running services first.

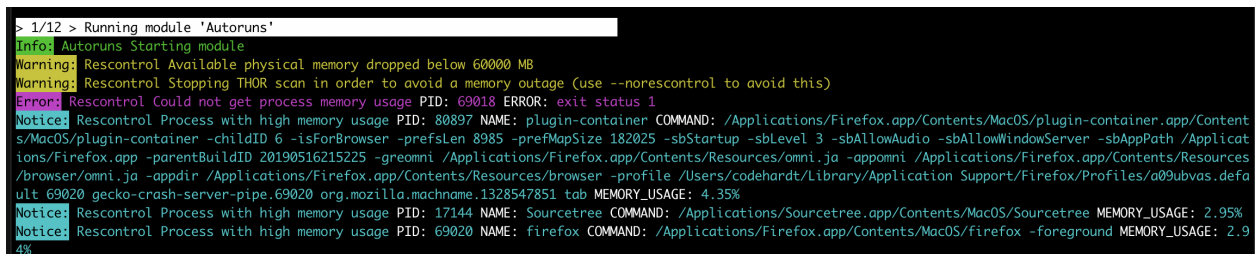
Resource control is active by default. You can deactivate it using **--norescontrol**.

Be advised that due to Resource Control, the THOR scan may terminate its completion. The scan gets terminated under the following conditions:

1. If the available physical memory drops below 60MB
2. If more than 60 MB of log data have been written (disk / syslog)

In this case, THOR switches in the "reduced-logging" mode in which it only transmits "Notices, Warnings and Alerts" and after another 4 MB of log data THOR terminates itself in order to prevent log flooding due to a high number of false positives

If the scan constantly terminates you should check what causes the performance issues or choose times with less workload (e.g. weekends, night). To debug such states, you can check the last warning that THOR generates before exiting the scan. It includes the top memory consumers that could have caused the memory exhaustion.



```
> 1/12 > Running module 'Autoruns'
Info: Autoruns Starting module
Warning: Rescontrol Available physical memory dropped below 60000 MB
Warning: Rescontrol Stopping THOR scan in order to avoid a memory outage (use --norescontrol to avoid this)
Error: Rescontrol Could not get process memory usage PID: 69018 ERROR: exit status 1
Notice: Rescontrol Process with high memory usage PID: 80897 NAME: plugin-container COMMAND: /Applications/Firefox.app/Contents/MacOS/plugin-container.app/Content
s/MacOS/plugin-container -childID 6 -isForBrowser -prefsLen 8985 -prefMapSize 182025 -sbStartup -sbLevel 3 -sbAllowAudio -sbAllowWindowServer -sbAppPath /Applicat
ions/Firefox.app -parentBuildID 20190516215225 -greomni /Applications/Firefox.app/Contents/Resources/omni.ja -appomni /Applications/Firefox.app/Contents/Resources
/browser/omni.ja -appdir /Applications/Firefox.app/Contents/Resources/browser -profile /Users/codehardt/Library/Application Support/Firefox/Profiles/a09ubvas.defa
ult 69020 gecko-crash-server-pipe.69020 org.mozilla.machname.1328547851 tab MEMORY_USAGE: 4.35%
Notice: Rescontrol Process with high memory usage PID: 17144 NAME: Sourcetree COMMAND: /Applications/Sourcetree.app/Contents/MacOS/Sourcetree MEMORY_USAGE: 2.95%
Notice: Rescontrol Process with high memory usage PID: 69020 NAME: firefox COMMAND: /Applications/Firefox.app/Contents/MacOS/firefox -foreground MEMORY_USAGE: 2.9
4%
```

Fig. 5: Resource Control Scan Termination

Warning: Deactivating Resource Control on systems with exhausted resources can put the system's stability at risk.

13.2.1 Automatic Soft Mode

Soft mode is automatically activated on systems with low hardware resources.

One of the following conditions activates soft mode:

- Less than 2 CPU cores
- Less than 1024 MB of RAM

In Soft mode several checks and features that could risk system's stability or could provoke an Antivirus or HIDS to intervene with the scanner are disabled. See [chapter Scan Modes](#) for a complete overview.

13.3 Scoring System

The scoring system is one of THOR's most prominent features. Both YARA signatures and filename IOCs contain a score field. The score is an integer value that can be negative to reduce the score on elements that are prone to false positives.

Only YARA rules and Filename IOCs support a user defined score. But since you are able to write YARA rules for almost every module, the scoring system is very flexible.

The total score of an element determines the level/severity of the resulting log message.

Score	Level	Condition
40	Notice	
60	Warning	
100	Alert	At least 1 sub score of 75 or higher

13.3.1 Scoring per Signature Type Match

Type	Score
YARA match	Defined in the meta data of the YARA rule as integer value (e.g. "score = 50")
Filename IOC match	Defined in the 2 nd field of the CSV (e.g. "\\evil.exe;80")
Keyword IOC match	"warning" level messages, see 14.3 "Default Scores"
C2 IOC match	"warning" and "alert" level messages, see 14.3 "Default Scores"

13.3.2 Accumulated Score by Module

Module	Cumulated Score	Scoring
Filescan Archive Scan DeepDive Prefetch WER	Yes	Score is a sum of the scores of all "REASON"s (YARA matches, filename IOCs, other anomalies) Note 1: Only positive scores are shown by default Note 2: Only the top 2 reasons are shown by default (use <code>-allreasons</code> to show all positive scores)
All Other Modules	No	Individual score of each signature match (YARA, filename IOC, keywords, C2) Note 1: This means that multiple matches for a single element are possible

13.3.3 Default Scores

If no score is set in an "alert" or "warning" message, THOR automatically appends a score that corresponds to the message level: Warning = 70, Alert = 100.

13.3.4 Exception: High total score with low sub scores

"Alerts" on file system elements are only generated if one of the sub scores is at least 75.

Before that change, multiple low scoring reasons had led to a score higher 100 and caused an "Alert" level message although not a single hard match was included in the "Reasons". A wrong extension, e.g. ".txt" for an executable, which is often used by employees to hand executables through tight mail filters, and a suspicious location, e.g. "C:\Temp\funprog.txt" caused an "Alert" level message.

Since version 8.27.2, one of the sub scores that pushes the total score over 100 has to be 75 or higher. (internally calculated as "alert_level - 25" because the user can adjust the alert level via the "--alert" parameter)

13.3.5 Exception: Filename IOC Matches

The "Filename IOC Check" is a sub check of the "String Check", which is applied to many elements, like Eventlog messages or Registry keys.

The function "checkString()" receives a string as input and returns possible matches.

The string is checked in multiple sub-checks against different signature lists. The most important sub-checks are "checkKeyword()" and "checkFilename()".

While the "checkKeyword()" sub-check returns each individual match, the "checkFilename()" sub check accumulates the score of all matches and returns a single total score. It is possible that many different filename signatures have matched on that string but only one match with a total score is reported. This is an exception to the usual behavior where only the "FileScan" module accumulates scores.

Filename IOC Matching in String Check Example

Imagine the following filename IOC signatures:

```
\\nmap.exe;70
\\bin\\nmap.exe;-30
```

and the following Keyword signature:

```
nmap.exe
```

The "checkString()" function receives the following string from the Eventlog scan module (here: a Sysmon Eventlog entry):

```
Process Create:
UtcTime: 2018-01-10 10:22:25.277
ProcessGuid: {c1b49677-e961-5a55-0000-0010bbc80702}
ProcessId: 3912
Image: C:\Program Files\Nmap\bin\nmap.exe
CommandLine: nmap.exe
CurrentDirectory: C:\Windows\system32\
User: PROMETHEUS\user1
LogonGuid: {c1b49677-1d72-5a53-0000-0020d4232500}
LogonId: 0x2523d4
TerminalSessionId: 1
IntegrityLevel: High
Hashes: SHA1=F5DC12D658402900A2B01AF2F018D113619B96B8,
        MD5=9FEA051A9585F2A303D55745B4BF63AA
ParentProcessGuid: {c1b49677-1d74-5a53-0000-001057452500}
ParentProcessId: 1036
ParentImage: C:\Windows\explorer.exe
ParentCommandLine: C:\Windows\Explorer.EXE
```

The "checkString()" function would create two messages: 1 "warning" for the keyword signature and 1 "notice" of the filename IOC signatures.

The keyword IOC matches in the "checkKeyword()" sub-check and "checkString()" returns a match, that generates a "Warning" level message that automatically receives a score of 75 (see *chapter 13.3.3 Default Scores*).

The filename IOCs would both match on the string in the "checkFilename()" sub-check and both scores would be summed up to a total score of 40 ($70 + (-30) = 40$), which would generate a "Notice".

13.4 Action on Match

The action command allows you define a command that runs whenever THOR encounters a file during "Filescan" that has a certain total score or higher. The default score that triggers the action command (if set) is 40.

The most popular use case for the action command is sample collection.

13.4.1 Action Flags

Parameter	Description
<code>--action_command</code> string	Run this command for each file that has a score greater than the score from <code>--action_level</code>
<code>---action_args</code> strings	Arguments to pass to the command specified via <code>--action_command</code> . The placeholders <code>%filename%</code> , <code>%filepath%</code> , <code>%file%</code> , <code>%ext%</code> , <code>%md5%</code> , <code>%score%</code> and <code>%date%</code> are replaced at execution time
<code>--action_level</code> int	Only run the command from <code>--action_command</code> for files with at least this score (default 40)

13.4.2 Command Line Use

A typical use would be e.g. to copy a sample to a network share:

```
copy %filepath% \\server\share1
```

To instruct THOR to run this command, you need

```
thor64.exe --action\_command copy --action\_args %filepath% --action\_args \\server\share1
```

13.4.3 Use in a Config File

The `./config` folder contains a template for a config file that uses the action commands.

Content of 'tmpl-action.yml':

```
# Action to perform if file has been detected with a score more than the defined 'action_level'
# You may use all environment variables that are available on the system, i.e. %COMPUTERNAME%.
# Further available meta vars are:
# %score% = Score
# %file% = Filename without extension
# %filename% = Basename
# %filepath% = Full path
# %ext% = Extension without dot
# %md5% = MD5 value
# %date% = Detection time stamp

action_level: 35
action_command: "copy"
action_args:
- "%filepath%"
- "\\VBOXSVR\Downloads\restore_files\%COMPUTERNAME%\_%md5%\_%file%\_%ext%\_%date%"
```

13.5 THOR DB

This simple SQLite database is created by default in the "`%ProgramData%\thor`" (Linux, macOS: `/var/lib/thor/`) directory as "`thor.db`". You can deactivate THOR DB and all its features by using the "`--nothordb`" flag.

It stores persistent information over several scan runs:

- Scan State Information
This information is used to resume scan runs where they were stopped
- Delta Comparison
This detection feature allows to compare the result of a former module check with the current results and indicate suspicious changes between scan runs

The THOR DB related command line options are:

Parameter	Description
<code>--nothordb</code>	Disables THOR DB completely. All related features will be disabled as well.
<code>--dbfile [string]</code>	Allows to define a location of the THOR database file. File names or path names are allowed. If a path is given, the database file 'thor.db' will be created in the directory. Environment variables are expanded.
<code>--resume</code>	Resumes a previous scan (if scan state information is still available and the exact same command line arguments are used)
<code>--resumeonly</code>	Only resume a scan if a scan state is available. Do not run a full scan if no scan state can be found.

13.5.1 Scan Resume

THOR tries to resume a scan when you set the **--resume** parameter. Since THOR version 10.5 the resume state doesn't get tracked by default due to its significant performance implications. If you want to be able to resume a scan, you have to start scans with the **--resume** flag. If you start a scan and a previous resume state is present, then THOR is going to resume the interrupted scan.

It will only resume the previous scan if

1. you have started the scan with **--resume**
2. the argument list is exactly the same as in the first scan attempt
3. you haven't used the flag **--nothordb**
4. scan state information is still available
(could have been cleared by running THOR a second time without the **--resume** parameter)

You can always clear the resume state and discard an old state by running `thor.exe` once without using the **--resume** parameter.

13.5.2 Delta Comparison

The delta comparison feature allows comparing former scan results on a system with the current results, indicating changes in system configurations and system components.

Currently, the following scan modules feature the delta comparison check:

- **Autoruns**
THOR compares the output of the Autoruns module with the output of the last scan run. The Autoruns does not only check "Autorun" locations but also elements like browser plugins, drivers, LSA providers, WMI objects and scheduled tasks.
- **Services**
The comparison detects new service entries and reports them.
- **Hosts**
New or changed entries in the "hosts" file could indicate system manipulations by attackers to block certain security functions or intercept connections.

13.5.3 Archive Scan

The ArchiveScan feature supports the following archive types:

- ZIP
- RAR
- TAR
- TAR + GZIP (.tar.gz)
- TAR + BZIP2 (.tar.bz2)

COMMAND LINE OPTIONS

This section lists all options that THOR TechPreview currently offers.

14.1 Scan Options

-t, --template string	Process default scan parameters from this YAML file
--generate-config	Print a YAML config from the given parameters and exit
-p, --path strings	Scan a specific file path. Define multiple paths by specifying this option multiple times. Append ':NOWALK' to the path for non-recursive scanning (default: only the system drive)
--allhds	(Windows Only) Scan all local hard drives (default: only the system drive)
--alldrives	(Windows Only) Scan all local drives, including network drives and ROM drives (default: only the system drive)
--max_file_size uint	Max. file size to check (larger files are ignored). Increasing this limit will also increase memory usage of THOR.
--max_file_size_intense uint	Max. file size to check in intense mode. See --intense and --max_file_size.
--max_log_lines int	Maximum amount of lines to check in a log file before skipping the remaining lines
--max_process_size uint	Max process size to check (larger processes won't be scanned). Increasing this limit will also increase possible memory usage of THOR.
--max_runtime int	Maximum runtime in hours. THOR will stop once this time has run out. 0 means no maximum runtime.
--nodoublecheck	Don't check whether another THOR instance is running (e.g. in Lab use cases when several mounted images are scanned simultaneously on a single system) (requires a Forensic Lab license)
-f, --epoch strings	Specify a range of days with attacker activity as start and end date pairs. Files created/modified between these days (including the specified start, excluding the specified end) will receive an extra score. Example: -f 2009-10-09 -f 2009-10-10 marks the 09.10.2009 as relevant.

--epochscore int	Score to add for files that were created/modified on days with attacker activity (see --epoch parameter)
--insecure	Skip TLS host verification (insecure)
--ca strings	Root CA for host certificate verification during TLS handshakes
--cross-platform	Apply IOCs with path separators platform independently.
--require-admin	Terminate immediately if THOR is executed without administrator rights.

14.2 Scan Modes

--quick	Activate a number of flags to speed up the scan at cost of some detection. This is equivalent to: --noeventlog --nofirewall --nopfiles --nowebdirscan --nologscan --noevtx --nohotfixes --nomft --lookback 3 --lookback-modules filescan
--soft	Skip CPU and RAM intensive modules (Mutexes, Firewall, Logons, Network sessions and shares, LSA sessions, open files, hosts file), don't decompress executables and doesn't perform a DoublePulsar backdoor check, lower max CPU usage to 70% and set low priority for THOR. This mode activates automatically on systems with 1 CPU core or less than 1024 MB RAM.
--intense	Disable soft mode, activate dump file analysis, MFT analysis and sigma rules, don't skip registry keys with less relevance, and scan processes regardless of size. WARNING: This scan mode performs tasks that affect systems stability. Do not use this mode in live scanning unless you accept the risk.
--diff	Set lookback time (see --lookback) for each module to the last time the module ran successfully and activates --global-lookback. Effectively, this means that only elements that changed since the last scan are examined. (only works if ThorDB has been active)
--lookback int	Specify how many past days shall be analyzed. Event log entries from before this point will be ignored. 0 means no limit
--global-lookback	Apply Lookback to all modules that support it (not only Eventlog). See also --lookback and --lookback-modules. Warning: Timestamping or similar methods of antivirus evasion may result in elements not being examined.
--force-aptdir-lookback	Enforce lookback application on all files in the FileScan module. By default, especially endangered directories ignore the lookback value.
--lookback-modules strings	Apply Lookback to the given modules. See also --lookback and --modules.

Warning: Timestomping or similar methods of antivirus evasion may result in elements not being examined.

--lab

Lab scan mode - scan only the file system, disable resource checks and quick mode, activate intense mode, disable ThorDB, apply IOCs platform independently and use all CPU cores.

This option scans all drives by default, but is often used with -p to scan only a single path. Requires a Forensic Lab license.

--virtual-map strings Rewrite found file paths to use a different prefix.

This can be useful for mounted images, where the current location of files does not match the original location and therefore references might be out of date.

Specify the original and current path as --virtual-map path/to/current/location:path/to/original/location.

On Windows, drive names are also supported, e.g. specify --virtual-map F:C if the drive on F: was originally used as C:.

Requires a Forensic Lab license.

14.3 Resource Options

-c, --cpulimit float	Limit CPU usage to this level (in percent). Minimum is 15%
--nocpulimit	Disable cpulimit check
--nosoft	Disable automatic activation of soft mode (see --soft)
--norescontrol	Do not check whether the system is running out of resources. Use this option to enforce scans that have been canceled due to resource scarcity. (use with care!)
--minmem uint	Cancel the running scan if the amount of free physical memory drops below this value (in MB)
--lowprio	Reduce the priority of the THOR process to a lower level
--verylowprio	Reduce the priority of the THOR process to a very low level
--nolowprio	Do not reduce the priority of the THOR process to a lower level due to soft mode (see --soft)
--nolockthread	Do not lock calls to C libraries to main thread (this may increase performance at the cost of memory usage)
--yara-stack-size int	Allocate this number of slots for the YARA stack. Default: 16384. Increasing this limit will allow you to use larger rules, albeit with more memory overhead.
--yara-max-strings-per-rule int	Allow this amount of strings per YARA rule.
--yara-timeout int	Cancel any YARA checks that take longer this amount of time (in seconds)
--threads uint16	Run this amount of THOR threads in parallel. Requires a Forensic Lab license.

--bulk-size uint Check this amount of elements together, e.g. log lines or registry entries

14.4 Special Scan Modes

-m, --image_file string Scan only the given single memory image / dump file (don't use for disk images, scan them mounted with --lab). Requires a Forensic Lab license.

--image-chunk-size uint Scan image / dump files in chunks of this size

-r, --restore_directory string Restore PE files with YARA rule matches during the Deep-Dive into the given folder

--restore_score int Restore only chunks with a total match score higher than the given value

--dropzone Watch and scan all files dropped to a certain directory (which must be passed with -p). Disable resource checks and quick mode, activate intense mode, disable ThorDB and apply IOCs platform independently. Requires a Forensic Lab license.

--dropdelete Delete all files dropped to the drop zone after the scan.

14.5 Thor Thunderstorm Service

--thunderstorm Watch and scan all files sent to a specific port (see --server-port). Disable resource checks and quick mode, activate intense mode, disable ThorDB and apply IOCs platform independently.

--server-upload-dir string Path to a directory where THOR drops uploaded files.

If this path does not exist, THOR tries to create it.

--server-host string IP address that THOR's server should bind to.

--server-port uint16 TCP port that THOR's server should bind to.

--server-cert string TLS certificate that THOR's server should use. If left empty, TLS is not used.

--server-key string Private key for the TLS certificate that THOR's server should use. Required if --server-cert is specified.

--server-store-samples string Sets whether samples should be stored permanently in the folder specified with --server-upload-dir.

Specify "all" to store all samples, or "malicious" to store only samples that generated a warning or an alert.

--server-result-cache-size uint32 Size of the cache that is used to store results of asynchronous requests temporarily.

If set to 0, the cache is disabled and asynchronous results are not stored.

--pure-yara Only scan files using YARA signatures (disables all programmatic checks, STIX, Sigma, IOCs, as well as most features and modules)

- sync-only-threads uint16** Reserve this amount of THOR threads for synchronous requests
- force-max-file-size** Enforce the maximum file size even on files like registry hives or log files which are usually scanned despite size.

14.6 License Retrieval

- asgard string** Request a license from the given asgard server
- asgard-token string** Use this token to authenticate with the License API of the asgard server. The token can be found in the 'Downloads' or 'Licensing' section in the ASGARD. This requires ASGARD 2.5+.
- q, --license-path string** Path containing the THOR license
- portal-key string** Get a license for this host from portal.nexttron-systems.com using this API Key.

This feature is only supported for host-based server / workstation contracts.
- portal-contracts ints** Use these contracts for license generation. If no contract is specified, the portal selects a contract by itself. See --portal-key.
- portal-nonewlic** Only use an existing license from the portal. If none exists, exit. See --portal-key.

14.7 Active Modules

- a, --module strings** Activate the following modules only (Specify multiple modules with -a Module1 -a Module2 ... -a ModuleN).
- noproc** Do not analyze Processes
- nofilesystem** Do not scan the file system
- noreg** Do not analyze the registry
- nouser** Do not analyze user accounts
- nologons** Do not show currently logged in users
- noautoruns** Do not analyse autorun elements
- noeventlog** Do not analyse the eventlog
- norootkits** Do not check for rootkits
- noevents** Do not check for malicious events
- nodnscache** Do not analyze the local DNS cache
- noenv** Do not analyze environment variables
- nohosts** Do not analyze the hosts file
- nomutex** Do not check for malicious mutexes
- notasks** Do not analyse scheduled tasks
- noservices** Do not analyze services

--noprofiles	Do not analyze profile directories
--noatjobs	Do not analyze jobs scheduled with the 'at' tool
--nonetworksessions	Do not analyze network sessions
--nonetworkshares	Do not analyze network shares
--noshimcache	Do not analyze SHIM Cache entries
--nohotfixes	Do not analyze Hotfixes
--nowmistartup	Do not analyze startup elements using WMI
--nofirewall	Do not analyze the local Firewall
--nowmi	Disable all checks with WMI functions
--nolsasessions	Do not analyze lsa sessions
--nomft	Do not analyze the drive's MFT (default, unless in intense mode)
--mft	Analyze the drive's MFT
--nopipes	Do not analyze named pipes
--noetwwatcher	Do not analyze ETW logs during THOR runtime
--nointegritycheck	Do not check with the package manager for package integrity on Linux

14.8 Module Extras

--process ints	Process IDs to be scanned. Define multiple processes by specifying this option multiple times (default: all processes) (Module: ProcessCheck)
--dump-procs	Generate process dumps for suspicious or malicious processes (Module: ProcessCheck)
--max-procdumps uint	Create at most this many process dumps (Module: ProcessCheck)
--procdump-dir string	Store process dumps of suspicious processes in this directory (Module: ProcessCheck)
-n, --eventlog-target strings	Scan specific Eventlogs (e.g. 'Security' or 'Microsoft-Windows-Sysmon/Operational') (Module: Eventlog)
--nodoublepulsar	Do not check for DoublePulsar Backdoor (Module: Rootkit)
--full-registry	Do not skip registry hives keys with less relevance (Module: Registry)
--showdeleted	Show deleted files found in the MFT as 'info' messages.
--allfiles	Scan all files, even ones that are usually not interesting. Uses the file size set by --max_file_size_intense.

14.9 Active Features

--nothordb	Do not use or create ThorDB database for holding scan information
--sigma	Scan with Sigma signatures
--dumpscan	Scan memory dumps
--nologscan	Do not scan log files (identified by .log extension or location)
--noregwalk	Do not scan the whole registry during the registry scan
--noyara	Disable checks with YARA
--nostix	Disable checks with STIX
--noarchive	Do not scan contents of archives
--noc2	Disable checks for known C2 Domains
--noprochandles	Do not analyze process handles
--noproconnections	Do not analyze process connections
--noamcache	Do not analyze Amcache files
--noregistryhive	Do not analyze Registry Hive files
--noexedecompress	Do not decompress and scan portable executables
--nowebdirscan	Do not analyze web directories that were found in process handles
--novulnerabilitycheck	Do not analyze system for vulnerabilities
--noprefetch	Do not analyze prefetch directory
--nogroupxml	Do not analyze groups.xml
--nowmipersistence	Do not check WMI Persistence
--nolnk	Do not analyze LNK files
--noknowledgedb	Do not check Knowledge DB on Mac OS
--nower	Do not analyze .wer files
--noevtx	Do not analyze EVTX files
--noauthorizedkeys	Do not analyze authorized_keys files
--timestomp	Activate timestomping detection (experimental, may take some time)
--noimphash	Do not calculate imphash for suspicious EXE files (Windows only)
--c2-in-memory	Apply C2 IOCs on process memory (not recommended unless you are willing to accept many false positives on browser and other process memories)

14.10 Feature Extras

--customonly	Use custom signatures only (disables all internal THOR signatures and detections)
--full-proc-integrity	Increase sensitivity for process impersonation detection. Likely to cause false positives, but also better at detecting real threats.
--processintegrity	Run PE-Sieve to check for process integrity (Windows only)

14.11 Output Options

-l, --logfile string	Log file for text output
--jsonfile string	Log file for JSON output, see --json
--htmlfile string	Log file for HTML output
--appendlog	Append text log to existing log instead of overwriting
-o, --csvfile string	Generate a CSV containing MD5,Filepath,Score for all files with at least the minimum score
--csvstats	Generate a CSV file containing the scan summary in a single line
--stats-file string	Set the name of the CSV stats file, see --csvstats
-e, --rebase-dir string	Specify the output directory where all output files will be written. Defaults to the current working directory.
--nocsv	Do not write a CSV of all mentioned files with MD5 hash (see --csvstats)
--silent	Do not print anything to command line
--brd	Suppress all personal information in log outputs to comply with local data protection policies
--nolog	Do not generate text or HTML log files
--eventlog	Log to windows application eventlog
-x, --min int	Only report files with at least this score
--json	Create a json report file
--nohtml	Do not create an HTML report file
--allreasons	Show all reasons why a match is considered dangerous (default: only the top 2 reasons are displayed)
--printshim	Include all SHIM cache entries in the output as 'info' level messages
--printamcache	Include all AmCache entries in the output as 'info' level messages
-j, --overwrite-hostname string	Override the local hostname value with a static value (useful when scanning mounted images in the lab. Requires a Forensic Lab license.
-i, --scanid string	Specify a scan identifier (useful to filter on the scan ID, should be unique)

--scanid-prefix string	Specify a prefix for the scan ID that is concatenated with a random ID if neither --scanid nor --noscanid are specified
--noscanid	Do not automatically generate a scan identifier if none is specified
--keyval	Format text and HTML log files with key value pairs to simplify the field extraction in SIEM systems (key='value')
--cmdjson	Format command line output as JSON
--cmdkeyval	Use key-value pairs for command line output, see --keyval
--encrypt	Encrypt the generated log files and the MD5 csv file
--pubkey string	Use this RSA public key to encrypt the logfile and csvfile (see --encrypt). Both --pubkey="<key>" and --pubkey="<file>" are supported.
--nocolor	Do not use ANSI escape sequences for colorized command line output
--genid	Print a unique ID for each log message. Identical log messages will have the same ID.
--print-rescontrol	Print THOR's resource threshold and usage when it is checked
--truncate int	Max. length per THOR value (0 = no truncate)
--registry_depth_print int	Don't print info messages when traversing registry keys at a higher depth than this
--utc	Print timestamps in UTC instead of local time zone
--rfc3339	Print timestamps in RFC3339 (YYYY-MM-DD'T'HH:mm:ss'Z') format
--reduced	Reduced output mode - only warnings, alerts and errors will be printed
--printlicenses	Print all licenses to command line (default: only 10 licenses will be printed)
--local-syslog	Print THOR events to local syslog
--showall	Print rule matches even if that rule already matched more than 10 times.
--ascii	Don't print non-ASCII characters to command line and log files
--string-context uint	When printing strings from YARA matches, include this many bytes surrounding the match
--include-info-in-html	Include info messages in the HTML report

14.12 ThorDB

--dbfile string	Location of the thor.db file
--resumeonly	Don't start a new scan, only finish an interrupted one. If no interrupted scan exists, nothing is done.
--resume	Store information while running that allows to resume an interrupted scan later. If a previous scan was interrupted, resume it instead of starting a new one.

14.13 Syslog

-s, --syslog strings	Write output to the specified syslog server, format: server[:port[:syslogtype[:sockettype]]]. Supported syslog types: DEFAULT/CEF/JSON/SYSLOGJSON/SYSLOGKV Supported socket types: UDP/TCP/TCPTLS Examples: -s syslog1.dom.net, -s arcsight.dom.net:514:CEF:UDP, -s syslog2:4514:DEFAULT:TCP, -s syslog3:514:JSON:TCPTLS
--rfc3164	Truncate long Syslog messages to 1024 bytes
--rfc5424	Truncate long Syslog messages to 2048 bytes
--rfc	Use strict syslog according to RFC 3164 (simple host name, shortened message)
--maxsysloglength int	Truncate Syslog messages to the given length (0 means no truncation)
--cef_level int	Define the minimum severity level to log to CEF syslogs (Debug=1, Info=3, Notice=4, Error=5, Warning=8, Alarm=10)

14.14 Reporting and Actions

--notice int	Minimum score on which a notice is generated
--warning int	Minimum score on which a warning is generated
--alert int	Minimum score on which an alert is generated
--action_command string	Run this command for each file that has a score greater than the score from --action_level.
--action_args strings	Arguments to pass to the command specified via --action_command. The placeholders %filename%, %filepath%, %file%, %ext%, %md5%, %score% and %date% are replaced at execution time.
--action_level int	Only run the command from --action_command for files with at least this score.
--nofserrors	Silently ignore filesystem errors

14.15 THOR Remote

--remote strings	Target host (use multiple --remote <host> statements for a set of hosts)
--remote-user string	Username (if not specified, windows integrated authentication is used)
--remote-password string	Password to be used to authenticate against remote hosts
--remote-prompt	Prompt for password for remote hosts
--remote-debug	Debug Mode for THOR Remote
--remote-dir string	Upload THOR to this remote directory
--remote-workers int	Number of concurrent scans
--remote-rate int	Number of seconds to wait between scan starts

14.16 Automatic Collection of Suspicious Files (Bifrost)

--bifrost2Server string	Server running the Bifrost 2 quarantine service. THOR will upload all suspicious files to this server. This flag is only usable when invoking THOR from ASGARD 2.
--bifrost2Score int	Send all files with at least this score to the Bifrost 2 quarantine service. This flag is only usable when invoking THOR from ASGARD 2.
-b, --bifrostServer string	Server running the Bifrost quarantine service. THOR will upload all suspicious files to this server.
--bifrostPort int	Port where the Bifrost quarantine service is running
--bifrostScore int	Minimum score to send file to the Bifrost quarantine service

14.17 Debugging and Info

--debug	Show Debugging Output
--trace	Show Tracing Output
--printall	Print all files that are checked (noisy)
--print-signatures	Show THOR Signatures and exit
--version	Show THOR, signature and software versions and exit
-h, --help	Show help for most important options and exit
--fullhelp	Show help for all options and exit

DEBUGGING

Most unexpected behavior can be debugged by using the parameter **--debug**.

If you ever encounter a situation in which:

- THOR doesn't produce an alert on a known malicious element
- THOR exits with an error
- THOR takes a long time or unexpected short time on elements

15.1 Debugging Examples

Then try scanning that specific element with the **--debug** parameter set.

To run only a certain module use:

```
thor64.exe -a Mutex
thor64.exe -a FileScan
thor64.exe -a Eventlog
```

You can try to reduce the scope of a module even further by using lookbacks

```
thor64.exe -a Eventlog --lookback 3
thor64.exe -a FileScan -p C:\Windows\System32 --globallookback --lookback 1
```

To find out why a certain file couldn't be detected, use **--debug** with **--printall** and try to switch into **intense mode**.

```
thor64.exe -a Filescan -p C:\testfolder --debug --printall
thor64.exe -a Filescan -p C:\testfolder --debug --printall --intense
```

If it has been detected in **intense mode** but not in default mode, the file extension or the magic header is most likely the problem. You can adjust **max_file_size** in **./config/thor.yml** or add a magic header in **./signatures/misc/file-type-signatures.cfg**.

15.2 Finding Bottlenecks

You may get the error message "MODULE: RuntimeWatcher MESSAGE: Maximum runtime has exceeded, killing THOR" or encounter very slow or never-ending scans.

You can check the statistics table in "thor.db" on that end system after a scan to determine the last element or elements that took a long time to process.

We recommend using: <https://sqlitebrowser.org/>

The THOR DB is located at: **C:\ProgramData\thor\thor.db**

DB Browser for SQLite - C:\ProgramData\thor\thor10.db

File Edit View Tools Help

New Database Open Database Write Changes Revert Changes Open Project Save Project Attach Database Close Database

Database Structure Browse Data Edit Pragma Execute SQL

Table: stats

	module	element	started	duration
	Filter	Filter	Filter	Filter
1	RegistryChecks	SOFTWARE\Classes\WorkspaceRuntime.Workspace.1	2020-04-09 11:46:32.732851+00:00	0
2	RegistryChecks	SOFTWARE\Classes\WorkspaceRuntime.Workspace	2020-04-09 11:46:32.7284332+00:00	0
3	RegistryChecks	SOFTWARE\Classes\WorkspaceBrokerAx.Workspac...	2020-04-09 11:46:32.7231963+00:00	0
4	RegistryChecks	SOFTWARE\Classes\WorkspaceBrokerAx.Workspac...	2020-04-09 11:46:32.7184094+00:00	0
5	RegistryChecks	SOFTWARE\Classes\WorkspaceBroker.WorkspaceB...	2020-04-09 11:46:32.7129112+00:00	0
6	RegistryChecks	SOFTWARE\Classes\WorkspaceBroker.WorkspaceB...	2020-04-09 11:46:32.7078139+00:00	0
7	RegistryChecks	SOFTWARE\Classes\Workspace.ResTypeRegistry.1	2020-04-09 11:46:32.7035492+00:00	0
8	RegistryChecks	SOFTWARE\Classes\Workspace.ResTypeRegistry	2020-04-09 11:46:32.6995246+00:00	0
9	RegistryChecks	SOFTWARE\Classes\Workspace.PolicyProcessor.1	2020-04-09 11:46:32.6948404+00:00	0
10	RegistryChecks	SOFTWARE\Classes\Workspace.PolicyProcessor	2020-04-09 11:46:32.6902687+00:00	0
11	RegistryChecks	SOFTWARE\Classes\Workspace.Installer.1	2020-04-09 11:46:32.6847406+00:00	0
12	RegistryChecks	SOFTWARE\Classes\Workspace.Installer	2020-04-09 11:46:32.6804795+00:00	0
13	RegistryChecks	SOFTWARE\Classes\Wordpad.Document.1	2020-04-09 11:46:32.6754806+00:00	0
14	RegistryChecks	SOFTWARE\Classes\WordAddin_RD.WordAddinRD.1	2020-04-09 11:46:32.6603448+00:00	0
15	RegistryChecks	SOFTWARE\Classes\WordAddin_RD.WordAddinRD	2020-04-09 11:46:32.3211086+00:00	0
16	RegistryChecks	SOFTWARE\Classes\WMVFile	2020-04-09 11:46:32.3182644+00:00	0
17	RegistryChecks	SOFTWARE\Classes\WMSDKNamespace.Namespac...	2020-04-09 11:46:32.3120948+00:00	0
18	RegistryChecks	SOFTWARE\Classes\WMSDKNamespace.Namespac...	2020-04-09 11:46:32.3079582+00:00	0
19	RegistryChecks	SOFTWARE\Classes\WMSDKMSBSourcePlugin.MSB...	2020-04-09 11:46:32.302025+00:00	0
20	RegistryChecks	SOFTWARE\Classes\WMSDKMSBSourcePlugin.MSB...	2020-04-09 11:46:32.2981605+00:00	0
21	RegistryChecks	SOFTWARE\Classes\WMSDKHTTPSourcePlugin.HT...	2020-04-09 11:46:32.2941111+00:00	0
22	RegistryChecks	SOFTWARE\Classes\WMSDKHTTPSourcePlugin.HT...	2020-04-09 11:46:32.2892924+00:00	0
23	RegistryChecks	SOFTWARE\Classes\WMSClientNetManager.ClientN...	2020-04-09 11:46:32.2843999+00:00	0
24	RegistryChecks	SOFTWARE\Classes\WMSClientNetManager.ClientN...	2020-04-09 11:46:32.2793848+00:00	0

Edit Database Cell

Mode: Text

3712

Type of data currently in cell: Text / Numeric
4 char(s)

Remote

Identity

Name Commit L

15.3 Most Frequent Causes of Missing Alerts

15.3.1 THOR didn't scan file due to file size restrictions

Solution: Use `--max_file_size` parameter or set permanently in config file `./config/thor.yml`. Also note that in lab scanning mode the default value is much bigger (`--max_file_size_intense`)

15.3.2 THOR didn't scan the file due to a skipped deeper inspection

This can be caused by two reasons:

the magic header of that file is not in the list of interesting magic headers (see **./signatures/misc/file-type-signatures.cfg**) AND file doesn't have a relevant file extension (.asp, .vbs, .ps, .ps1, .rar, .tmp, .bas, .bat, .chm, .cmd, .com, .cpl, .crt, .dll, .exe, .hta, .js, .lnk, .msc, .ocx, .pcd, .pif, .pot, .pdf, .reg, .scr, .set, .sys, .url, .vb, .vbe, .vbs, .wsc, .wsf, .wsh, .ct, .t, .input, .war, .jsp, .php, .asp, .aspx, .doc, .docx, .pdf, .xls, .xlsx, .ppt, .pptx, .tmp, .log, .dump, .pwd, .w, .txt, .conf, .cfg, .conf, .config, .psd1, .psm1, .ps1xml, .clixml, .pscl, .pssc, .pl, .www, .rdp, .jar, .docm, .ace, .job, .temp, .plg, .asm)

Solution: Use an intense scanning mode for that folder (**--intense**) or add the magic header to **file-type-signatures.cfg** (Warning: this file gets overwritten with an update; Warning: intense scanning mode threatens the scan's and system's stability).

15.3.3 THOR fails to initialize custom rules with the correct type

It happens very often that users that prepare custom IOCs or YARA rules forget to include the correct keyword in the filename of the IOC or YARA rule file.

The correct use of keywords is described in [this chapter](#) for IOCs and [this chapter](#) for YARA rules.

A wrong or missing keyword leads to situations in which a file that contains YARA rules that are meant to be applied to log files, doesn't contain a "log" keyword in its name.

You can review a correct initialization in the command line output or log file.

```
Info Adding rule set from my-log-rules.yar as 'log' type
```

Using the keyword **c2** for C2 IOCs in a filename should result in a line like the following:

```
Info Reading iocs from /tmp/thor10/custom-signatures/iocs/my-c2-iocs.txt as 'domains'
→ type
```

15.4 Most Frequent Causes of Failed Scans

15.4.1 External Processes Terminating THOR

Whenever THOR dies without any traceback or panic message and no error message in the log file, an external process terminated the THOR process.

Usually the four following sources are responsible (descending order, by frequency):

1. Antivirus or EDR killed the THOR process
2. A user killed the THOR process
3. A management solution that noticed a high CPU load caused by the THOR process killed it
4. Attackers killed the THOR process

Note: A process termination that always happens at the same element is a sign for an Antivirus or EDR detection.

15.4.2 Insufficient Free Memory

`fatal error: out of memory`

Probable causes:

1. Other processes consume a lot of memory
2. THOR's scanning of certain elements requires a lot of memory
3. You are using the wrong THOR version for your architecture
4. You've activated a feature that consumes a lot of memory (e.g. `--mft` or `--intense`)

Whenever THOR recognizes a low amount of free memory, THOR checks the top 3 memory consumers on the system and includes them in the log message that it writes before exiting.

You could try running THOR in Soft Mode (`--soft`) in which modules and features that require a lot of memory are deactivated.

Using the 32bit binary of thor named **thor.exe** on a 64bit system can lead to interrupted scans with this error message. The 32bit binary isn't able to address as much memory as the 64bit version. Always make sure to use the correct THOR version for the respective architecture.

15.5 Help Us With The Debugging

If you cannot find the source of a problem, please contact us using the support@nexttron-systems.com email address.

You can help us find and debug the problem as quickly as possible by providing the following information.

15.5.1 Which THOR version do you use?

Tell us which THOR version you are using:

1. for which operating system (Windows, Linux, macOS, AIX)
2. for which architecture (32bit, 64bit)

Run **thor --version** and copy the resulting text into the email.

On Windows:

`thor64.exe --version`

On Linux:

`./thor-linux-64 --version`

On Mac OS:

`./thor-macosx --version`

This should produce a message like this:

```
THOR 10.6.6
Build bea8066 (2021-04-27 14:32:40)
YARA 4.0.5
PE-Sieve 0.2.8.5
OpenSSL 1.1.1j
Signature Database 2021/05/03-150936
Sigma Database 0.19.1-1749-g2f12c5c5
```

15.5.2 What is the target platform that THOR fails on?

Please provide the output of the following commands.

On Windows:

```
systeminfo > systeminfo.txt
```

On Linux:

```
uname -a
```

On Mac OS:

```
system_profiler -detailLevel mini > system_profile.txt
```

15.5.3 Which command line arguments do you use?

Please provide a complete list of command line arguments that you've used when the error occurred.

```
thor64.exe --quick -e D:\logs -p C:\Windows\System32
```

15.5.4 Provide the log of a scan with --debug flag

The most important element is a scan log of a scan with set **--debug** flag.

The easiest way is to run the scan exactly as you've run it when the problem occurred adding the **--debug** command line flag.

e.g.

```
thor64.exe --quick -e D:\logs -p C:\Windows\System32 --debug
```

If you're able to pinpoint the error to a certain module, you could limit the scan to that module to get to the problematic element more quickly, e.g.

```
thor64.exe -a Rootkit --debug
```

After the scan you will find the normal text log (*.txt) in the program folder. It is okay to replace confidential information like the hostname or IP addresses.

Note: The debug log files can be pretty big, so please compress the file before submitting it to us. Normal log files have a size between 1 and 4 MB. Scans started with the **--debug** flag typically have sizes of 30-200 MB. The compression ratio is typically between 2-4%, so a compressed file shouldn't be larger than 10 MB.

15.5.5 Provide a Screenshot (Optional)

Sometimes errors lead to panics of the executables, which causes the situation that the relevant log lines don't appear in the log file. In these cases, please also create a screenshot of a panic shown in the command line window.

15.5.6 Provide the THOR database (Optional)

The *THOR DB* helps us debug situations in which the THOR scan timed out or didn't complete at all. It contains statistics on the run time of all used modules and the durations of all folders up to the second folder level from the root of a partition. (e.g. C:\Windows\SysWow64).

The default location of that file is:

- Windows: **C:\ProgramData\thor\thor.db**
- Linux/macOS: **/var/lib/thor/thor.db**

Please provide that file in situations in which:

- THOR exceeded its maximum run time
- THOR froze and didn't complete a scan for days
- THOR scans take too long for the selected scan targets

15.5.7 Further Notes

- If the files are too big to send, even after compression, please contact us and you'll receive a file upload link that you can use
- If a certain file or element (eventlog, registry hive) caused the issue, please check if you can provide that file or element for our analysis

ANALYSIS AND INFO

16.1 Log Analysis Manual

The `/docs` folder of the THOR program directory contains a manual named **THOR_LogAnalysis.pdf** on how to process the events generated by THOR.

16.2 VALHALLA Rule Lookup

The new rule info pages allow you to get more information on a certain rule. You can find all the meta data, as well as past rule matches and previous antivirus verdicts. A second tab contains statistics. You can also report false positives that you've encountered with that rule using the button in the tab bar.

Note that the rule info lookups in the web GUI are rate limited. If you query rule infos too often, you get blocked.

The rule info pages can be access using this URL scheme:

https://valhalla.nextron-systems.com/info/rule/RULE_NAME

For example:

https://valhalla.nextron-systems.com/info/rule/HKTL_Empire_ShellCodeRDI_Dec19_1

16.3 Rule List Output

By using the `--print-signatures` flag, you can get a list of all initialized YARA and Sigma rules.


VALHALLA
SUPERCHARGE YOUR DETECTION

HKTL_Empire_ShellCodeRDI_Dec19_1

Info
Statistics
Report False Positive

Rule Info

Name	HKTL_Empire_ShellCodeRDI_Dec19_1
Author	Florian Roth
Description	Detects Empire shellcode RDI
Score	75
Reference	https://github.com/BC-SECURITY/Empire/
Date	2019-12-09
Minimum Yara	1.7
Rule Hash	da9a57d689dbb4ea1d7bf1ae9229025f
Tags	[HKTL]
Required Modules	[]
Av Ratio	44.59
Virustotal Matches	https://www.virustotal.com/gui/search/hktl_empire_shellcoderdi_dec19_1/comments

Antivirus Verdicts

Rating	Number of Samples
Malicious (>= 10 engines)	66
Suspicious (< 10 engines)	15
Clean (0 engines)	0

Rule Matches

Timestamp	Positives	Total	Hash	VT Link
2020-05-10 13:45:29	10	59	497ff4d9f4b8b24d7aaf37b06b53ea7efe753b6581c5e9b0d982e63ec7bca72f	>
2020-05-09 23:05:27	11	59	7d19e0e8e3920366a22460a92245b8d3983be1d361a8248ab2c8c5f7f53c7fb4	>
2020-05-02 02:42:59	3	73	485fcc6c4a64bc63a7a9312ae92f69c4123aec81fb9d637f320111798f92dab	>
2020-04-24 01:59:27	10	58	5b07014032ff99616f49b08af009ba5a74d7abe2cc49903b5dbd80fff5271678	>
2020-04-23 20:50:34	28	67	bff921b96bd5541e5cd393684f4f2626b6b0f6a606ff7c3a2cdef878b2c16eda	>
2020-04-19 01:15:18	29	70	4d120187bb417b65c553c181fc17520ac7b1912ef9f458efb5b5e9cb682cf79c	>
2020-04-17 17:56:11	9	55	5efae2d2cb84adbc68ef14dc326735d5a357eefb4be6009f468066a4645f2b38	>
2020-04-17 14:58:28	49	72	6b5a437399030a758f4fbac6d20eaa76ca917c11737b30fd527c53491d9ac579	>
2020-04-17 14:11:25	21	72	6c80375edb554b570e2b4a5bc5e3236d2eeb0a9bf7319b7cd976aeeba2a3077e	>

Fig. 1: Rule Info Page

```
TYPE: YARA RULE: APT_Sakabota_Malware_Sep19_1 DESCRIPTION: Detects Sakabota malware RULE_DATE: 2019-09-26 REFERENCE: https://unit42.paloaltonetworks.com/xhunt-campaign-attacks-on-kuwait-shipping-and-transportation-organizations/ AUTHOR: Florian Roth
TYPE: YARA RULE: APT_KEYBOY_Malware_Sep19_1 DESCRIPTION: Detects KEYBOY malware RULE_DATE: 2019-09-27 REFERENCE: https://twitter.com/stvemillertime/status/1177623076151779329 AUTHOR: Florian Roth
TYPE: YARA RULE: APT_MAL_NK_Kimsuky_Sep19_1 DESCRIPTION: Detects Kimsuky malware RULE_DATE: 2019-09-30 REFERENCE: https://twitter.com/cyberwar_15/status/1178529821573279744 AUTHOR: Florian Roth
TYPE: YARA RULE: APT_MAL_NK_Kimsuky_PS1_Sep19_2 DESCRIPTION: Detects Kimsuky PowerShell malware helper RULE_DATE: 2019-09-30 REFERENCE: https://twitter.com/cyberwar_15/status/1178529821573279744 AUTHOR: Florian Roth
TYPE: YARA RULE: APT_MAL_Lazarus_Sep19_1 DESCRIPTION: Detects Lazarus malware RULE_DATE: 2019-09-30 REFERENCE: https://twitter.com/KseProso/status/1178580006047539200 AUTHOR: Florian Roth
TYPE: YARA RULE: APT_NATO_Lamberts_Strings DESCRIPTION: Detects strings from Lamberts malware samples RULE_DATE: 2019-09-30 REFERENCE: https://ti.qianxin.com/blog/articles/network-weapons-of-cia/ AUTHOR: Florian Roth
```

Fig. 2: Rule List Output

USE CASES

This chapter contains use cases that users often asked for.

17.1 Disk Image Analysis with FTKImager

THOR, as a scanner, does not mount disk images to a certain driver on your forensic workstation. You have to use 3rd party tools for that task. We recommend using [FTKImager](#).

In case you plan to use an automated setup in which you use scripts to automatically process images, you could try to use the old [command line versions of FTKImager](#).

So, first you mount the image to a certain drive and then use THOR in lab scanning mode to analyze the mounted image. To benefit from all features of this mode, you have to acquire and use a so-called **forensic lab** or **lab** license.

The following example shows a recommended set of parameters, scanning a mounted image of a host named **WKS0001** on drive **S:** of your forensic Windows workstation.

```
thor64.exe --lab --virtual-map S:C -j WKS0001 -p S:\
```

The `--lab` parameter will apply several internal flags (e.g. enables intense mode scanning every file, enables multi-threading, disables resource control, removes all limitations). The `--virtual-map` parameter maps every file found in elements of that image to the original drive letter and allows the message enrichment to work correctly. The `-j HOSTNAME` parameter can be used to write every log line with the hostname of the original system and not with that of the forensic workstation.

You find more information on the scan parameters in the chapter *Special Scan Modes*.

This [blog post](#) mentions different ways to use commercial or built-in tools to mount and scan VMDK images.

17.2 Memory Image Analysis with Volatility

In this use case, we show a way to run a THOR scan on a full memory image of a target system.

In volatility, we first evaluate the right profile for a memory image. You can use the `imageinfo` command or select one manually from the list that is shown when you run `vol.py --info`.

```
vol.py -f win10-lab1.mem imageinfo

Volatility Foundation Volatility Framework 2.6.1
INFO      : volatility.debug      : Determining profile based on KDBG search...
           Suggested Profile(s) : Win10x64_19041
```

(continues on next page)

(continued from previous page)

```

AS Layer1 : SkipDuplicatesAMD64PagedMemory (Kernel AS)
AS Layer2 : FileAddressSpace (/mnt/downloads/mem-dumps/win10-lab1.
↪mem)

PAE type : No PAE
DTB : 0x1aa002L
KDBG : 0xf8005aa00b20L
Number of Processors : 2
Image Type (Service Pack) : 0
KPCR for CPU 0 : 0xffffffff80055ec00000L
KPCR for CPU 1 : 0xffffffff8500313c000000L
KUSER_SHARED_DATA : 0xffffffff78000000000000L
Image date and time : 2021-06-15 08:25:08 UTC+0000
Image local date and time : 2021-06-15 10:25:08 +0200

```

We then create a directory that will store all our process memory images.

```
mkdir procs
```

Now we can extract all process memory images and save them to the new directory.

```

vol.py -f win10-lab1.mem --profile=Win10x64_19041 memdump -D procs/

Volatility Foundation Volatility Framework 2.6.1
*****
Writing System [ 4 ] to 4.dmp
*****
Writing Registry [ 92 ] to 92.dmp
*****
Writing smss.exe [ 348 ] to 348.dmp
*****
Writing csrss.exe [ 440 ] to 440.dmp
*****
Writing wininit.exe [ 512 ] to 512.dmp
*****
Writing csrss.exe [ 520 ] to 520.dmp
*****
Writing winlogon.exe [ 608 ] to 608.dmp
*****
Writing services.exe [ 624 ] to 624.dmp
*****
Writing lsass.exe [ 656 ] to 656.dmp
*****
Writing fontdrvhost.ex [ 748 ] to 748.dmp

```

We recommend saving that output for mapping purposes, since THOR will only report the file names upon a YARA rule match, e.g. 748.dmp, and not the name of the executable fontdrvhost.exe.

Using THOR we can now scan the extracted process memory images.

```
./thor-linux-64 ---lab -p /mnt/mem-dumps/procs/
```

Without a valid lab license, we can simulate that behaviour using the following command (see *chapter Special Scan Modes* for more details and flags used in lab scan mode):

```
./thor-linux-64 -a Filescan --intense -p /mnt/mem-dumps/procs/
```

The output of such a scan will look like this

```
[?%] Worker 01: /mnt/mem-dumps/procs/3812.dmp [_____
↪]Progress: 286 MB
[?%] Worker 01: /mnt/mem-dumps/procs/3812.dmp [_____
↪]Progress: 343 MB
Alert YARA Score Rule Match TARGET: /mnt/mem-dumps/procs/3812.dmp TYPE: file NAME: SUSP_
↪Encoded_UA_Mozilla SCORE: 50 DESCRIPTION: Detects encoded keyword - User-Agent:
↪Mozilla/ SIGTYPE: internal CHUNK_OFFSET: 366000000 TAGS: SUSP, T1027 MATCHING_STRINGS:
↪Str1: "VzZXItQWdlbnQ6IE1vemlsbGEv" in "dBRMD0NC1VzZXItQWdlbnQ6IE1vemlsbGEvNS4wIChjb2"
↪at 0x1672eacc MODIFIED: Tue Jun 15 11:38:13 2021 CHANGED: Tue Jun 15 11:38:13 2021
↪TARGET_SIZE: 610324480
[?%] Worker 01: /mnt/mem-dumps/procs/3812.dmp [_____
↪]Progress: 400 MB
[?%] Worker 01: /mnt/mem-dumps/procs/3812.dmp [_____
↪]Progress: 457 MB
```

The match includes an offset, e.g. `CHUNK_OFFSET: 366000000`, and a matching string, e.g. `Str1: "VzZXItQWdlbnQ6IE1vemlsbGEv"` which help you to locate the correct section in the dump file using a hex editor for further analysis.

LINKS AND REFERENCES

THOR Website

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

Nextron Customer Portal

<https://portal.nextron-systems.com>

Nextron Software Update Status

<https://update1.nextron-systems.com/info.php>

YARA Documentation

<https://yara.readthedocs.io/>

yarGen - YARA Rule Generator

<https://github.com/Neo23x0/yarGen/>

THOR APT Scanner App and Add-on v2

<https://splunkbase.splunk.com/app/3717/>

<https://splunkbase.splunk.com/app/3718/>

Sigma Project

<https://github.com/Neo23x0/sigma>

18.1 Open Source License Acknowledgements

List of third-party software components used by THOR 10 with open source licensing requirements.

18.1.1 golang.org

Copyright (c) 2009 The Go Authors

Licensed under BSD-3 License

18.1.2 OpenSSL

Copyright (c) OpenSSL

Licensed under OpenSSL license (<https://www.openssl.org/source/license.html>)

18.1.3 YARA

Copyright (c) 2007-2016 The YARA Authors

Licensed under BSD-3 License

18.1.4 github.com/Azure/go-ntlmssp

Copyright (c) 2016 Microsoft

Licensed under MIT License

18.1.5 github.com/botherder/go-autoruns

modified as github.com/Codehardt/go-autoruns

Copyright (c) 2018 Claudio Guarnieri

Licensed under MIT License

18.1.6 github.com/omarghader/pefile-go

modified as github.com/Codehardt/go-pefile

Copyright (c) 2004-2015 Ero Carrera

Licensed under MIT License

18.1.7 Copyright (c) 2018 Marcel Gebhardt

github.com/Codehardt/go-cpulimit

github.com/Codehardt/go-handle

github.com/Codehardt/go-ntlm-proxy-auth

github.com/Codehardt/go-osversion

github.com/Codehardt/go-priority

github.com/Codehardt/go-taskscheduler

Licensed under MIT License

18.1.8 Copyright (c) 2018 Samuel Melrose

github.com/Codehardt/go-win64api
github.com/iamacarpet/go-win64api
Licensed under MIT License

18.1.9 Copyright (c) 2011, Evan Shaw <edsrzf@gmail.com>

github.com/Codehardt/mmap-go
github.com/ncw/mmap-go
Licensed under BSD-3 License

18.1.10 github.com/StackExchange/wmi

Copyright (c) 2013 Stack Exchange
Licensed under MIT License

18.1.11 Copyright (c) 2018-2020 velocidex

github.com/Velocidex/ordereddict
github.com/Velocidex/regparser
github.com/secDre4mer/regparser
www.velocidex.com/golang/go-ntfs
www.velocidex.com/golang/evtx
github.com/Velocidex/evtx
Licensed under Apache License 2.0

18.1.12 github.com/andrewkroh/sys

Copyright (c) 2009 The Go Authors

18.1.13 github.com/botherder/go-files

Copyright (c) 2018 Nex
Licensed under MIT License

18.1.14 github.com/coreos/go-systemd/v22

(no copyright notes found)

Licensed under Apache License 2.0

18.1.15 github.com/dsnet/compress

Copyright (c) 2015, Joe Tsai and The Go Authors

Licensed under BSD-3 License

18.1.16 github.com/dustin/go-humanize

Copyright (c) 2005-2008 Dustin Sallings <dustin@spy.net>

Licensed under MIT License

18.1.17 Copyright (c) 2014-2020 Elasticsearch BV

github.com/elastic/beats

github.com/elastic/go-ucfg

github.com/elastic/go-sysinfo

Licensed under Apache License 2.0

18.1.18 github.com/fatih/color

Copyright (c) 2013 Fatih Arslan

Licensed under MIT License

18.1.19 github.com/fsnotify/fsnotify

Copyright (c) 2012 The Go Authors

Copyright (c) 2012 fsnotify Authors

Licensed under BSD-3 License

18.1.20 Copyright (c) 2015 Zack Guo

github.com/gizak/termui/v3

github.com/gizak/termui/v3/widgets

github.com/gizak/termui/v3/drawille

Licensed under MIT License

18.1.21 github.com/go-ole/go-ole

Copyright (c) 2013-2017 Yasuhiro Matsumoto, <mattn.jp@gmail.com>
Licensed under MIT License

18.1.22 github.com/godbus/dbus

Copyright (c) 2013, Georg Reinke (<guelfey at gmail dot com>), Google
Licensed under BSD-2 License

18.1.23 github.com/gofrs/uuid

Copyright (C) 2013-2018 by Maxim Bublis <b@codemonkey.ru>
Licensed under MIT License

18.1.24 github.com/google/pprof

(no copyright notes found)
Licensed under Apache License 2.0

18.1.25 github.com/golang/snappy

Copyright (c) 2011 The Snappy-Go Authors
Licensed under BSD-3 License

18.1.26 Copyright (c) 2010-2012 The w32 Authors

github.com/gonutz/w32
github.com/shirou/w32
github.com/AllenDang/w32
Licensed under MIT License

18.1.27 Licensed under Mozilla Public License 2.0

github.com/hashicorp/go-multierror
github.com/hashicorp/golang-lru
github.com/hashicorp/errwrap
(no copyright notes found)

18.1.28 github.com/hillu/go-yara/v4

Copyright (c) 2015-2020 Hilko Bengen <bengen@hilluzination.de>
Licensed under BSD-2 License

18.1.29 github.com/inconshreveable/mousetrap

Copyright (c) 2014 Alan Shreve
Licensed under Apache License 2.0

18.1.30 github.com/joeshaw/multierror

Copyright (c) 2014 Joe Shaw
Licensed under MIT License

18.1.31 github.com/kardianos/service

Copyright (c) 2015 Daniel Theophanes
Licensed under zlib License

18.1.32 github.com/marcsauter/single

Copyright (c) 2018 Marc Sauter
Licensed under MIT License

18.1.33 Copyright (c) Yasuhiro MATSUMOTO <mattn.jp@gmail.com>

github.com/mattn/go-colorable
github.com/mattn/go-isatty
github.com/mattn/go-runewidth
github.com/mattn/go-shellwords
github.com/mattn/go-sqlite3
Licensed under MIT License

18.1.34 github.com/mitchellh/go-wordwrap

Copyright (c) 2014 Mitchell Hashimoto
Licensed under MIT License

18.1.35 github.com/mholt/archiver

Copyright (c) 2016 Matthew Holt
Licensed under MIT License

18.1.36 github.com/nsf/termbox-go

Copyright (C) 2012 termbox-go authors
Licensed under MIT License

18.1.37 github.com/nwaples/rardecode

Copyright (c) 2015 Nicholas Waples
Licensed under BSD-2 License

18.1.38 github.com/pierrec/lz4

Copyright (c) 2015 Pierre Curto
Licensed under BSD-3 License

18.1.39 Copyright (c) Dave Cheney <dave@cheney.net>

github.com/pkg/errors
github.com/pkg/profile
Licensed under BSD-2 License

18.1.40 github.com/pytimer/win-netstat

Copyright (c) 2018 pytimer
Licensed under MIT License

18.1.41 github.com/sebdah/goldie

Copyright 2016 Sebastian Dahlgren <sebastian.dahlgren@gmail.com>
Licensed under MIT License

18.1.42 github.com/shirou/gopsutil

Copyright (c) 2014 WAKAYAMA Shirou
Copyright (c) 2009 The Go Authors
Licensed under BSD License

18.1.43 Copyright (c) 2016 SmartyStreets, LLC

github.com/smartystreets/goconvey
github.com/smartystreets/assertions
Licensed under MIT License

18.1.44 github.com/spf13/cobra

(no copyright notes found)
Licensed under Apache License 2.0

18.1.45 github.com/spf13/pflag

Copyright (c) 2012 Alex Ogier
Copyright (c) 2012 The Go Authors
Licensed under BSD-3 License

18.1.46 github.com/stretchr/testify

Copyright (c) 2012-2018 Mat Ryer and Tyler Bunnell
Licensed under MIT License

18.1.47 github.com/xi2/xz

(no license and copyright notes found)

18.1.48 Copyright (c) 2016-2017 Uber Technologies, Inc.

go.uber.org/atomic
go.uber.org/multierr
Licensed under MIT License

18.1.49 go.uber.org/zap

Copyright (c) 2016-2017 Uber Technologies, Inc.
Licensed under MIT License

18.1.50 Copyright (c) 2009 The Go Authors

golang.org/x/arch
golang.org/x/crypto
golang.org/x/sys
golang.org/x/exp
golang.org/x/net
golang.org/x/oauth2
golang.org/x/term
golang.org/x/time
golang.org/x/tools
golang.org/x/sync

Licensed under BSD-3 License

18.1.51 gopkg.in/ini.v1

Copyright (c) 2014 Unknwon
Licensed under Apache License 2.0

18.1.52 gopkg.in/natefinch/npipes.v2

Copyright (c) 2013 npipes authors
Licensed under MIT License

18.1.53 Copyright 2011-2016 Canonical Ltd.

gopkg.in/yaml.v2
gopkg.in/yaml.v3
Licensed under Apache License 2.0

18.1.54 howett.net/plist

Copyright (c) 2013, Dustin L. Howett
Copyright (c) 2012 The Go Authors
Licensed under BSD-3 License

18.1.55 github.com/williballenthin/shellbags

(no copyright notes found)

Licensed under Apache License 2.0

18.1.56 go.opencensus.io

(no copyright notes found)

Licensed under Apache License 2.0

18.1.57 cloud.google.com/go

(no copyright notes found)

Licensed under Apache License 2.0

18.1.58 Copyright (c) 2015 Chzyer

github.com/chzyer/logex

github.com/chzyer/readline

github.com/chzyer/test

Licensed under MIT License

18.1.59 github.com/ianlancetaylor/demangle

Copyright (c) 2015 The Go Authors

Licensed under BSD-3 License

18.1.60 github.com/jstemmer/go-junit-report

Copyright (c) 2012 Joel Stemmer

Licensed under MIT License

18.1.61 Google Go modules

google.golang.org/api

google.golang.org/appengine

google.golang.org/genproto

google.golang.org/grpc

(no copyright notes found)

Licensed under Apache License 2.0

18.1.62 Copyright (c) 2018 The Go Authors

google.golang.org/protobuf
github.com/golang/protobuf
Licensed under BSD-3 License

18.1.63 github.com/golang/grouper

(no copyright notes found)
Licensed under Apache License 2.0

18.1.64 github.com/google/go-cmp

Copyright (c) 2017 The Go Authors
Licensed under BSD-3 License

18.1.65 https://github.com/hasherezade/pe-sieve

Copyright (c) 2017-2020, @hasherezade
Licensed under BSD 2-Clause License

18.1.66 https://github.com/hasherezade/libpeconv

Copyright (c) 2017-2019, hasherezade (@hasherezade)
Licensed under BSD 2-Clause License

18.1.67 https://github.com/parsiya/golink

(no copyright notes found)
Licensed under Apache License 2.0

18.1.68 https://github.com/olekukonko/tablewriter

Copyright (C) 2014 by Oleku Konko
Licensed under MIT License

18.1.69 github.com/frankban/quicktest

Copyright (c) 2017 Canonical Ltd.
Licensed under MIT License

18.1.70 github.com/niemeyer/pretty

Copyright 2012 Keith Rarick
Licensed under MIT License

18.1.71 github.com/prometheus/procfs

(no copyright notes found)
Licensed under Apache License 2.0

18.1.72 Beats

<https://github.com/secDre4mer/beats/v7>
<https://github.com/elastic/beats/v7>
Licensed by Elasticsearch B.V.
Licensed under Apache License 2.0

18.1.73 <https://github.com/ulikunitz/xz>

Copyright (c) 2014-2020 Ulrich Kunitz
Licensed under BSD 2-Clause "Simplified" License

18.1.74 go.elastic.co/ecszap

Copyright 2020 Elastic and contributors
Licensed under Apache License 2.0

18.1.75 gopkg.in/check.v1

Copyright (c) 2010-2013 Gustavo Niemeyer <gustavo@niemeyer.net>
Licensed under BSD 2-Clause "Simplified" License

18.1.76 github.com/gopherjs/gopherjs

Copyright (c) 2013 Richard Musiol
Licensed under BSD 2-Clause "Simplified" License

18.1.77 github.com/kr/text

Copyright 2012 Keith Rarick
Licensed under MIT License

18.1.78 Copyright (c) 2016 Mark Bates

github.com/gobuffalo/envy
github.com/gobuffalo/packr
github.com/gobuffalo/packr/v2
Licensed under MIT License

18.1.79 github.com/karrick/godirwalk

Copyright (c) 2017, Karrick McDermott
Licensed under BSD 2-Clause License

18.1.80 github.com/rogppe/go-internal

Copyright (c) 2018 The Go Authors
Licensed under BSD 3-Clause "New" License

18.1.81 github.com/sirupsen/logrus

Copyright (c) 2014 Simon Eskildsen
Licensed under MIT License

18.1.82 github.com/Workiva/go-datastructures

(no copyright notes found)
Licensed under Apache License 2.0

18.1.83 github.com/swagger-api/swagger-ui

Copyright 2020 SmartBear Software Inc.
Licensed under Apache License 2.0

18.1.84 github.com/cheggaaa/pb/v3

Copyright (c) 2012-2015, Sergey Cherepanov
Licensed under BSD 3-Clause "New" License

18.1.85 github.com/magefile/mage

Copyright 2017 the Mage authors
Licensed under Apache License 2.0

18.1.86 github.com/secDre4mer/go-parseflags

(no copyright notes found)
Licensed under BSD 3-Clause "New" License

18.1.87 github.com/secDre4mer/go-inject

Copyright (c) 2021 secDre4mer
Licensed under MIT License

INDICES AND TABLES

- search