



SPARK – Manual

Version 1.15, September 2018
Nextron Systems GmbH

| | | |
|-------------|---|-----------|
| 1 | WHAT IS SPARK? | 4 |
| 2 | REQUIREMENTS | 4 |
| 3 | PACKAGE | 5 |
| 4 | SCAN PARAMETERS | 5 |
| 5 | SINGLE SCAN | 5 |
| 6 | DEPLOYMENT | 5 |
| 6.1 | ASGARD | 6 |
| 7 | SYSLOG | 6 |
| 7.1 | TARGET DEFINITION | 6 |
| 7.2 | KEY-VALUE PAIRS | 7 |
| 8 | OFTEN USED PARAMETERS | 7 |
| 9 | UPDATE | 7 |
| 10 | SPECIAL SCAN MODES | 8 |
| 10.1 | SOFT SCAN | 8 |
| 10.2 | QUICK SCAN | 9 |
| 10.3 | INTENSE SCAN | 9 |
| 10.4 | LAB SCANNING | 9 |
| 10.5 | EVENTLOG SCAN (WINDOWS ONLY) | 9 |
| 11 | RESOURCE CONTROL | 9 |
| 11.1 | AUTOMATIC SOFT MODE | 10 |
| 11.2 | CONTROL CPU LOAD | 10 |
| 11.3 | PROCESS PRIORITY | 10 |
| 11.4 | MEMORY EXHAUSTION | 10 |
| 11.5 | SYSLOG VOLUME LIMIT | 10 |
| 11.6 | DISABLE YARA | 10 |
| 11.7 | DISABLE RESOURCE CONTROL | 11 |
| 12 | EXCLUSIONS | 11 |
| 13 | CUSTOM SIGNATURES | 11 |
| 14 | HOW TO WRITE YOUR OWN SIGNATURES | 12 |
| 14.1 | SIMPLE IOCs | 12 |
| 14.1.1 | Hashes | 12 |
| 14.1.2 | File Name Characteristics | 12 |
| 14.1.3 | Keyword IOCs | 14 |
| 14.1.4 | Initialization Based on Strings in File Names | 14 |
| 14.2 | YARA | 15 |
| 14.2.1 | Generic YARA Rules | 15 |
| 14.2.2 | Specific YARA Rules | 15 |

| | | |
|-------------|---|-----------|
| 14.2.3 | YARA Rule Application..... | 16 |
| 14.2.4 | Create YARA Rules | 16 |
| 14.2.5 | Typical Pitfalls..... | 17 |
| 14.2.6 | YARA Rule Performance | 18 |
| 14.3 | ADJUST YARA RULES TO SCANNER FEATURES..... | 18 |
| 14.3.1 | Score..... | 19 |
| 14.3.2 | Additional Attributes | 20 |
| 14.3.3 | YARA Rules for Registry Detection | 21 |
| 14.3.4 | YARA Rules for Eventlog and Log Detection | 22 |
| 14.3.5 | False Positive Yara Rules | 23 |
| 14.4 | ENCRYPT CUSTOM SIGNATURES | 23 |
| 15 | ENCRYPT OUTPUT FILES | 24 |
| 16 | LINKS AND REFERENCES..... | 25 |

1 What is SPARK?

SPARK 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 local log files and file system. SPARK aims to be a sensitive auditor noticing files and behavior traces a common Antivirus may have missed. An integrated "Scoring System" enables SPARK to rate elements based on numerous characteristics to give hints on unknown malware.

SPARK 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 agent-less "APT Scanner" that has numerous detection mechanisms for hacking activity.

The key features are:

- Scans for hack tools and attacker activity traces
- Portable – no installation required (agent-less)
- Runs on many 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, text file)
- Throttling of the scan process to reduce the system load to a minimum

2 Requirements

THOR runs in any Windows environment without any further requirements. Everything needed is already included in the program package. For best results it is necessary to run SPARK as Administrator or LOCAL_SYSTEM.

It has been successfully tested on:

- Windows XP SP2 x86 (unsupported)
- Windows XP SP2 x64 (unsupported)
- Windows 2003 x86 (unsupported)
- Windows 2003 x64 (unsupported)
- Windows 7 x86
- Windows 7 x64
- Windows 2008 x86
- Windows 2008 R2 x64
- Windows 8.1
- Windows 2012
- Windows 2012 R2
- Windows 10
- Windows 2016
- RHEL/CentOS 6
- RHEL/CentOS 7
- SLES 11
- SLES 12
- Ubuntu 14 LTS
- Ubuntu 16 LTS
- Debian 8 (jessie)
- Debian 9 (stretch)
- OS X 10.8 (Mountain Lion) and higher

3 Package

The Windows SPARK Package includes the following files and directories:

- Spark Binary for 32-bit Systems (thor-spark-win-x86.exe)
- Spark Binary for 64-bit Systems (thor-spark-win-x64.exe)
- THOR Util (thor-util.exe) used for updates, license file generation, signature verification
- SPARK Core Util (spark-core-util.exe) in case of SPARK Core

The Linux SPARK Package includes the following files and directories:

- Spark Binary for 32-bit Systems (thor-spark-linux-x86)
- Spark Binary for 64-bit Systems (thor-spark-linux-x64)
- THOR Util (thor-util) used for updates, license file generation, signature verification
- SPARK Core Util (spark-core-util) in case of SPARK Core

The OSX SPARK Package includes the following files and directories:

- Spark Binary for 64-bit Systems (spark-macosx-x64)
- THOR Util (thor-util) used for updates, license file generation, signature verification
- SPARK Core Util (spark-core-util) in case of SPARK Core

All packages contain:

- Change Log File (changes.log)
- Documents like this manual, licenses and the EULA (./docs)
- Signature Files (./signatures)
- Folder for your custom signatures, YARA, Sigma and IOCs (./custom-signatures)

Files that you have to place in the program folder:

- THOR / SPARK / SPARK Core License File (*.lic)

4 Scan Parameters

The following overview lists all available scan parameters.

5 Single Scan

1. Start a Terminal (cmd.exe) with administrative rights (Right Click > Run as Administrator)
2. Change directory to the program directory of SPARK
3. Run "thor-spark-win-x64.exe" and select the appropriate parameters (the default is fine in most use cases) i.e. "thor-spark-win-x64.exe -s 10.1.52.1 -quick"

6 Deployment

SPARK 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 best practice deployment provides a SPARK 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 SPARK program package to the system and run it from the command line. The locally written log files are fully compatible with 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 SPARK into memory and start the scan process. You can either mount the network share and run SPARK from there or access it directly via its UNC path (e.g. \\server\share\thor-spark-win-x64.exe).

6.1 ASGARD

The ASGARD Management Center provides flows that run SPARK a selected group of end systems, collects the syslog messages in a central location or forwards them to your SIEM system.

7 Syslog

Select a syslog server to which the messages should be forwarded by using -s, e.g.

```
thor-spark-win-x64.exe -s syslog1.intranet.de
```

By default, SPARK does not truncate the Syslog messages. If you don't know what maximum length your target systems supports and want to make sure that no message gets lost, use --rfc5424.

```
--rfc3164
    Truncate long Syslog messages to 1024 bytes
--rfc5424
    Truncate long Syslog messages to 2048 bytes
```

In the current version, you can only set a single syslog target.

7.1 Target Definition

SPARK version 1.8 comes with a flexible Syslog target definition.

The definition consists of 4 elements:

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

Screenshot 1 - THOR version 8 Syslog target definition format

The available options for each element are:

```
(dst ip):(dst port):(DEFAULT/CEF/JSON/SYSLOGJSON):(UDP/TCP/TCPSSL)
```

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

```
(dst 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
```

When changing the type from standard syslog to CEF:

```
-s 10.0.0.4:514:CEF
```

7.2 Key-Value Pairs

Some special SIEM systems need the key and value pairs in a more robust form. You can set the "--keyval" Parameter to get the following output format.

```
KEY1="VALUE 1" KEY2="VALUE 2"
```

Listing 1 - Output formatting when using --keyval parameter

8 Often Used Parameters

Deactivate all file output - syslog only

```
thor-spark-win-x64.exe -s 10.1.5.14 --nolog
```

Save the result files to a different directory

```
thor-spark-win-x64.exe -s 10.1.5.14 -l Z:\%COMPUTERNAME%_spark.txt
```

Do a quick scan (selects the most relevant file paths only)

```
thor-spark-win-x64.exe --quick
```

Scan Multiple Paths

```
thor-spark-win-x64.exe -p C:\ -p D:\webapps -p E:\inetpub
```

9 Update

Updates for SPARK can be downloaded with 'thor-util' ('thor-util.exe' on Windows systems). This utility has different functions including the 'update' function.

THOR and SPARK update and license utility

Usage:

```
thor-util [command]
```

Available Commands:

| | |
|----------|---|
| download | Download program files and signature data (full packs) |
| help | Help about any command |
| install | Install from local download package (e.g. downloaded via web browser) |
| license | Get a license from a license server (ASGARD) |
| upgrade | Upgrade program files and signature data (diffs only, no config files) |
| verify | Verify the signature of an executable file |
| version | Print the version number of THOR Util |

Use "thor-util [command] --help" for more information about a command.

The different function are:

1. download - Allows you to download full versions of scanner packages that you've licensed
2. install - Mainly used in deployment scenarios as in several ASGARD flows
3. license - Generate a client or server type license using a central ASGARD license server

4. upgrade - Download new versions of present scanner packages

5. verify - verify the signature of executables

By adding the the feature name and --help to the thor-util command line you get the options for the selected feature. (e.g. thor-util upgrade --help)

The 'upgrade' function has the following options:

Usage:

thor-util upgrade [flags]

| | |
|----------------------------------|---|
| --debug | debug mode |
| -h, --help | help for upgrade |
| --insecure | do not check signatures |
| --license-path string | Path with valid locating license file(s) (optional) |
| --path string | Application path (default ".") |
| -t, --product-type string | product type (thor, spark-win, spark-linux, spark-osx) |
| -a, --proxy string | proxy address (e.g. http://proxy.int:8080) |
| -p, --proxy-pass string | proxy password |
| -n, --proxy-user string | proxy user |
| -u, --url URL | Download URL |

The 'thor-util' will look for valid license files in the current folder automatically and checks all files with a *.lic extension. However you can set the location of a valid license file using the --license-path option.

The utility tries to detect the type and version of the local package automatically. You can set a package to download using the --product-type option.

A valid command line using a proxy server would look like:

```
./thor-util upgrade -t spark-linux -a http://proxy.company.net:8080 -n user1 -p password1
```

10 Special Scan Modes

SPARK feature different scan modes that can be activated via command line parameters.

Only the "soft mode" is activated automatically on systems with low hardware resources (see chapter 11 "Resource Control" for more details). All other scan modes have to be activated manually.

10.1 Soft Scan

The "soft mode" can be manually activated by using the "--soft" parameter.

In soft mode,

- resource control functions will reduce the CPU load,
- process priority will be set below normal and
- process memory consuming check will be skipped (e.g. process memory scanning)

10.2 Quick Scan

The “quick mode” is a perfect option to run the most important modules and scan the most important locations on the hard drive.

The Quick scan mode has the same intensity as default mode. The only two differences are:

- Eventlog module is disabled (Windows)
- File system scan is limited to a fixed list of directories

The list of the directories is undocumented. We provide the list of directories on request.

10.3 Intense Scan

By using the “-intense” parameter you activate “intense mode”. This mode is NOT recommended for everyday scanning or single compromise assessments.

The “intense mode” is recommended in situations in which:

- System stability does not have priority
- A “default” scan didn’t produce the expected results
- Scan run time doesn’t matter and runtimes >24h are acceptable

10.4 Lab Scanning

The lab scanning mode selects the right options for most lab scanning purposes.

```
thor-spark-linux-x64 --fsonly -p /mnt/image1/
```

By activating the “fsonly” mode automatically enables:

- Intense scan mode
 - DeepDive on certain file types (memory dumps)
 - YARA scan on every file (regardless of magic header and file extension)

Activating “fsonly” mode automatically disables:

- Check for another scan process (you can run multiple instances at the same time)
- Quick mode
- Soft mode (automatic adjustments made during initialization)

Active Modules: FileScan, LogScan

Active Feature: Sigma, ArchiveCheck

10.5 Eventlog Scan (Windows Only)

SPARK features a special scan mode on the Windows platform that allows to scan certain Windows Eventlog channels.

```
thor-spark-win-x64.exe -n <Eventlog Channel> -n <Eventlog Channel 2> ...
```

For example:

```
thor-spark-win-x64.exe -n Security
```

```
thor-spark-win-x64.exe -n Microsoft-Windows-Sysmon/Operational
```

11 Resource Control

SPARK uses different approaches to reduce and adjust system load.

11.1 Automatic Soft Mode

Under normal circumstances adjustments via command line parameters are not necessary. SPARK automatically evaluates the system's resources during the initialization phase and automatically switches into "soft mode" (`--soft`) under the following conditions:

- System has only a single CPU core (often used in virtual machines)
- System has less than 1024MB of main memory (RAM)

"Soft mode" automatically

- applies CPU throttling to 80%
- sets the process priority (NICE) to lower than normal and
- skips memory intense checks (like process memory scanning).

It is not recommended to disable automatic "soft mode" manually by using "`--nosoft`".

11.2 Control CPU Load

You can use the '`-c`' option to set a maximum CPU load for the scan. SPARK will then throttle its load to that maximum.

The following example will limit the maximum CPU load of the scan to 80%. (single core)

```
./thor-spark-linux-x86 -c 80
```

11.3 Process Priority

You can run SPARK with a process priority (NICE level) lower than normal by using the following parameter.

```
--lowprio
```

11.4 Memory Exhaustion

By default SPARK exits its scan in situation in which the level of available free main memory drops below 30 Megabytes. You can adjust this level by using the following parameter.

```
-minmem int
```

There is no way to disable this protection except for disabling resource control in general.

11.5 Syslog Volume Limit

By default resource control applies a limit to the data transmitted via SYSLOG. This protection tries to prevent situations in which a certain signature creates numerous false positives and transmits a high number of events via SYSLOG to a central server.

The default limit is 7MB. If the limit is reached, SPARK stops sending messages via SYSLOG but still keeps writing log lines to the log file.

```
--syslog-max-bytes int
```

11.6 Disable YARA

The YARA feature consumes a lot of memory. If you have file or hash IOCs only, you can deactivate the YARA module with the following parameter.

```
--noyara
```

This will reduce the memory usage of SPARK from 170MB+ to less than 40MB.

11.7 Disable Resource Control

All resource control protections can be disabled with the following parameter.

--noresourcecontrol

Warning: Use this option ONLY in cases in which system stability has low priority.

12 Exclusions

You may create the file "**./config/directory-excludes.cfg**" to exclude directories and files from the analysis.

The exclusions 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 of the partition | \\hpsm\\log\\ |
| Every file with the extension .nsf | \\.nsf\$ |
| THOR's custom signatures | \\custom\\-signatures\\ |

Table 1 - File and Directory Excludes

Important: Note that you have to use slashes on Linux/OSX and backslashes on the Windows platform.

13 Custom signatures

You can add custom signatures and IOC list to certain directories in order to make them visible in the initialization phase. SPARK checks the following folders for signatures.

- ./signatures (all kind of signatures)
- ./custom-signatures (all kind of signatures, available since version 1.5)
- ./signature-base/yara (YARA signatures with .yar extension)
- ./signature-base/iocs (IOC files with .txt extension)

The files have to meet certain characteristics to get initialized in the correct category:

- YARA rules - all YARA rule files have to use the *.yar file extension (e.g. 'case77.yar')
- Filename IOCs - all filename IOC files have to contain the keyword 'filename' (e.g. 'filename-iocs-case77.txt')

- Hash IOCs - all hash IOC files have to contain the keyword 'hash' (e.g. 'hashes-case77.txt')
- Keyword IOCs - all keyword IOC files have to contain the keyword 'keyword' (e.g. 'case77-keywords.txt')

The application of these IOCs follow the standards from THOR, which means:

- YARA rules: Files, Process Memory, Log Lines
- Filename IOCs: Files, SHIM Cache, Scheduled Tasks, Log Lines
- Hash IOCs: Files
- Keyword IOCs: Log Lines, Process Command Lines, Scheduled Tasks

Note: The encrypted signature format (public key) used in newer THOR versions is not yet supported.

14 How to Write Your Own Signatures

14.1 Simple IOCs

The custom simple IOCs format is basically CSV with comment lines. Currently there are three template files that can be filled with your own IOC data. The simple IOC config file templates can be found in the folder `./custom-signatures`.

14.1.1 Hashes

The file `"custom-hashes.txt"` allows you to define hashes of malware or hack tools that the scanner should report at least as a "Warning".

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

```
#
# THOR CUSTOM EVIL HASHES
# This file contains MD5, SHA1 and SHA256 hashes and a short info like file name
# or hash origin
#
# APPLICATION -----
#
# Every line is treated as STRING.
# Every detection with one of these hashes increments the score by 70.
#
# FORMAT -----
#
# MD5;COMMENT
# SHA1;COMMENT
# SHA256;COMMENT
#
# EXAMPLES -----
#
# 0c2674c3a97c53082187d930efb645c2;DEEP PANDA Sakula Malware - http://goo.gl/R3e
# 000c907d39924de62b5891f8d0e03116;The Darkhotel APT http://goo.gl/DuS7wS
# c03318cb12b827c03d556c8747b1e323225df97bdc4258c2756b0d6a4fd52b47;Operation SMN
```

Figure 1 - File `"custom-hashes.txt"`

14.1.2 File Name Characteristics

In the `"filename-characteristics.txt"` you are able 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.

Figure 2 - File "filename-characteristics.txt"

Version A

Version B

Version C

It contains three fields:

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

which matches on a file named "svchost.exe" in a short path (max. 40 characters) that does not contain one of the many keywords in the regex list.

Reduce Score for Whole Directories

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

```
\\directory_with_many_false_positives\\;-30
```

14.1.3 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. The comment above each block is used as reference in the THOR log entries.

```
#
# 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
```

Figure 3 - Keyword IOC Example

14.1.4 Initialization Based on Strings in File Names

The scanners check the "./custom-signatures" folder and processes every file in this folder based on string tags in the file names.

For example, every file that contains the string "c2" will be initialized as Simple IOC indicators file with C2 server information. Internally we use the regex '[_\\W]c2[_\\W]' to detect the tag, so "mysource-c2-iocs.txt", "mysource_c2_iocs.txt" and "dec15_batch1_c2_indicators.txt" would be detected correctly, whereas on the contrary "c2-iocs.txt" or "myc2iocs.txt" would not.

The following tags are currently supported:

- "c2" 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

See chapter **Error! Reference source not found.** for more information on files that can be used in the "./custom-signatures" folder. Files can have any extensions, but standard extensions are ".txt" and ".cfg". Only ".dat" extensions are treated differently as THOR expects ".dat" files to be encrypted.

| Keyword in File Name | C2 IOCs | File Name IOCs | Hash IOCs | Keyword IOC | Example |
|----------------------|---------|----------------|-----------|-------------|-------------------------------|
| c2 | Yes | | | | misp_c2_iocs.txt |
| filename | | Yes | | | Case_UX22_filename_iocs.txt |
| filenames | | Yes | | | Malicious_filenames_unitX.txt |
| hash | | | Yes | | op_aura_hash_iocs.cfg |
| hashes | | | Yes | | int_misp_hashes.txt |
| keyword | | | | Yes | keywords-incident-3389.txt |
| keywords | | | | Yes | Incident-22-keyword.cfg |

Table 2 - Keywords in Simple IOC file names and corresponding initialization

14.2 YARA

THOR offers an interface to include own rules based on the YARA format. Just place valid rule files with the Extension ".yar" in the custom signature folder ("/custom-signatures/yara").

Yara rules are widely used in THOR and SPARK.

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

1. Generic Rules
2. Specific Rules

14.2.1 Generic YARA Rules

The "Generic" rules are standard YARA rules that are applied to payloads of files and memory. Just place any file with "*.yar" extension in the "./custom-signatures/yara" folder.

Generic 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.cfg. It extends the standard conditions by THOR specific extensions (see below).
- **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. (Warning Level)

14.2.2 Specific YARA Rules

The specific YARA rules contain certain keywords in their filename in order to select them for application in certain modules only.

- **Registry Keys**
Keyword: 'registry'
Rules are applied to a whole key and all of its values. This means that you can combine several key values in a single YARA rule.

- **Log Files**
Keyword: 'log'
Rules are applied to each log line
- **Process Memory**
Keyword: 'process'
Rules are applied to process memory only
- **All String Checks**
Keyword: 'keyword'
Rules are applied to all string checks in many different modules

14.2.3 YARA Rule Application

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

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

Table 3 - Generic YARA Rule Application

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

| Keyword in File Name | Applied in Module | Examples |
|----------------------|---|-------------------------------|
| registry | Registry | incident-feb17-registry.yar |
| log | Eventlog, Logscan | general-log-strings.yar |
| process | ProcessCheck (on process memory only) | 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 |

Table 4 - Specific YARA Rule Application

14.2.4 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 our scanners because don't provide provide a useful debugging mechanism for Yara rules. The

Yara binary can be downloaded from the developer's website (<https://code.google.com/p/yara-project/>).

The options for the YARA tool are listed below. The most useful options are "-r" to recursively scan a path and "-s" to show all matching strings.

```
C:\yara-3.4.0-win32>yara32.exe --help
YARA 3.4.0, the pattern matching swiss army knife.
Usage: yara [OPTION]... RULES_FILE FILE | DIR | PID

Mandatory arguments to long options are mandatory for short options too.

-t, --tag=TAG           print only rules tagged as TAG
-i, --identifier=IDENTIFIER  print only rules named IDENTIFIER
-n, --negate            print only not satisfied rules (negate)
-D, --print-module-data  print module data
-g, --print-tags        print tags
-m, --print-meta        print metadata
-s, --print-strings      print matching strings
-e, --print-namespace   print rules' namespace
-p, --threads=NUMBER    use the specified NUMBER of threads to scan a directory
-l, --max-rules=NUMBER  abort scanning after matching a NUMBER of rules
-d VAR=VALUE           define external variable
-x MODULE=FILE          pass FILE's content as extra data to MODULE
-a, --timeout=SECONDS   abort scanning after the given number of SECONDS
-r, --recursive         recursively search directories
-f, --fast-scan         fast matching mode
-w, --no-warnings       disable warnings
-v, --version           show version information
-h, --help             show this help and exit

Send bug reports and suggestions to: vmalvarez@virustotal.com.
```

Screenshot 2 - Test Yara Rules with the Yara Binary

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 severe 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 50)
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 specific add-ons you may use to enhance your rules. (external variables)

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/>

14.2.5 Typical Pitfalls

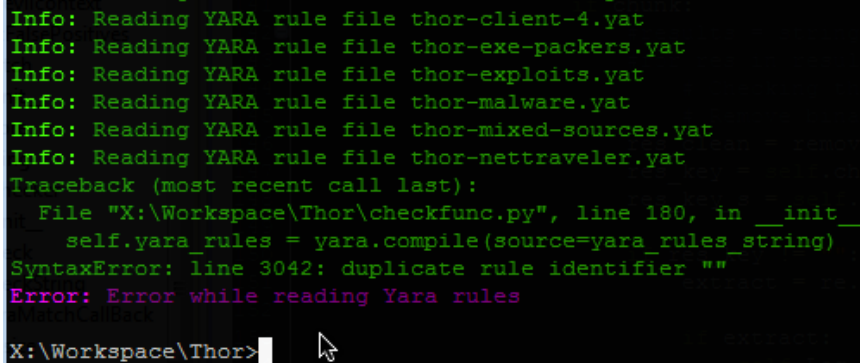
Some signatures - even the ones published by well-known vendors - cause problems on certain files. The most common source of trouble is the use of regular expressions with a variable length as shown in the following example. This APT1 rule published by the AlienVault team caused the YARA Binary as well as the THOR / SPARK 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.

```
rule APT1_WEBC2_TABLE {
  meta:
    author = "AlienVault Labs"
    info = "CommentCrew-threat-apt1"
  strings:
    $msg1 = "Fail To Execute The Command" wide ascii
    $msg2 = "Execute The Command Successfully" wide ascii
    $gif1 = /\w+\.gif/
    $gif2 = "GIF89" wide ascii
  condition:
    3 of them
}
```

Listing 2 – AlienVault APT1 Rule

Copying your rule to the signatures directory may cause the scanners to fail during rule initialization as shown in the following screenshot. In the current state of development, the error trace back is not as verbose as desired but gives the reason why the rule compiler failed. 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.



```
Info: Reading YARA rule file thor-client-4.yat
Info: Reading YARA rule file thor-exe-packers.yat
Info: Reading YARA rule file thor-exploits.yat
Info: Reading YARA rule file thor-malware.yat
Info: Reading YARA rule file thor-mixed-sources.yat
Info: Reading YARA rule file thor-nettraveler.yat
Traceback (most recent call last):
  File "X:\Workspace\Thor\checkfunc.py", line 180, in __init__
    self.yara_rules = yara.compile(source=yara_rules_string)
SyntaxError: line 3042: duplicate rule identifier ""
Error: Error while reading Yara rules

X:\Workspace\Thor>
```

Screenshot 3 - Error in THOR Yara rule set

14.2.6 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>

14.3 Adjust YARA Rules to Scanner Features

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 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".

```
rule simple_demo_rule_1 {
  meta:
    description = "Demo Rule"
```

```

strings:
    $a1 = "EICAR-STANDARD-ANTIVIRUS-TEST-FILE"
condition:
    $a1
}

```

Listing 3 - Simple Yara Rule

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

```

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
}

```

Listing 4 - Complex Yara Rule

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, md5 hashes, file sizes, path name and so on.

Our scanners add functionality to overcome these limitations.

14.3.1 Score

We make 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 our scanners 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.

```

rule demo_rule_score {
    meta:
        description = "Demo Rule"
        score = 35
    strings:
        $a1 = "EICAR-STANDARD-ANTIVIRUS-TEST-FILE"
        $a2 = "honkers" fullword
    condition:
        1 of them
}

```

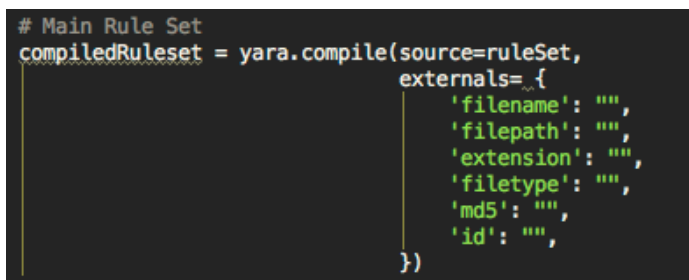
Listing 5 – YARA Rule with specific attribute "score"

Feel free to set your own "score" values in rules that 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 too 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 14.3.2.

14.3.2 Additional Attributes

Our scanners allow using certain external variables in your rules. They are passed to the "match" function and evaluated during matching.



```
# Main Rule Set
compiledRuleset = yara.compile(source=ruleSet,
                                externals={
                                    'filename': "",
                                    'filepath': "",
                                    'extension': "",
                                    'filetype': "",
                                    'md5': "",
                                    'id': "",
                                })
```

Screenshot 4 – Usable external variables

The 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/file-type-signatures.cfg") like "EXE" or "ZIP"
- 'md5' - the md5 hash of the file
- 'id' - only used in Windows Eventlog scanning as the Event ID / Event Code

The "filesize" value is a build-in variable and therefore not mentioned as THOR specific external variables.

```
rule demo_rule_enhanced_attribute_1 {
    meta:
        description = "Demo Rule - Eicar"
        score = 50
    strings:
        $a1 = "EICAR-STANDARD-ANTIVIRUS-TEST-FILE"
    condition:
        $a1 and filename matches /eicar.com/
}
```

Listing 6 – YARA Rule with external variable

The following listing shows a more complex rule using several of the external variables.

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.

```

rule demo_rule_enhanced_attribute_2 {
    meta:
        description = "Demo Rule - Eicar - exact"
        score = 50
    strings:
        $a1 = "EICAR-STANDARD-ANTIVIRUS-TEST-FILE"
    condition:
        $a1 and filename matches /eicar\.(com|dll|exe)/ and filesize < 100
}

```

Listing 7 – Yara Rule with more complex THOR Enhanced Attributes

The following YARA rule show a typical combination used in one of the client specific rule sets, which are integrated in our scanners. 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 files with a specific extension.

```

rule HvS_Client_2_APT_Java_IDX_Content_hard {
    meta:
        description = "VNCViewer.jar Entry in Java IDX file"
        score = 50
    strings:
        $a1 = "vncviewer.jar"
        $a2 = "vncviewer/VNCViewer.class"
    condition:
        1 of ($a*) and extension matches /\.idx/
}

```

Listing 8 – Real Life Yara Rule

14.3.3 YARA Rules for Registry Detection

The Registry module allows checking a complete registry path key/value pairs with a YARA rules. To accomplish this, the scanner composes a string from the key/value pairs of a registry key path and formats them as shown in the following screen shot.

```

-----
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;PMDisplayName;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#0005426
-----
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:\Us
ers\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

```

Figure 4 - Composed strings from registry key/value pairs

The composed format is:

```

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

```

This means that you can write a YARA rule that looks like this (remember to escape all back slashes):

```
rule Registry_DarkComet {
    meta:
        description = "DarkComet Registry Keys"
        author = "F.Roth "
        score = 50
    strings:
        $a1 = "LEGACY_MY_DRIVERLINKNAME_TEST;NextInstance"
        $a2 = "CurrentVersion\\Run;MicroUpdate"
    condition:
        1 of them
}
```

Listing 9 – Registry Yara Rule Example

The scanners are able to apply these rules to non-DWORD registry values. This means that e.g. a REG_BINARY value contains an executable as in the following example, you can write a YARA rule to detect this binary value in Registry as shown below.

REG_BINARY = 4d 5a 00 00 00 01

Corresponding YARA rule:

```
rule registry_binary_exe {
    meta:
        description = "Detects executables in Registry values"
        score = 70
    strings:
        $a1 = "Path;Value;4D5A00000001"
    condition:
        1 of them
}
```

Listing 10 - YARA rule to detect executables in Registry values

The letters in the expression have to be all uppercase. Remember that you have to use the keyword 'registry' in the file name that is placed in the ".custom-signatures/yara" folder in order to initialize the YARA rule file as registry rule set. (e.g. "registry_exe_in_value.yar")

Important:

Please notice that strings like HKEY_LOCAL_MACHINE, HKLM, HKCU, HKEY_CURRENT_CONFIG are not used in the strings 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. The strings should look like:

\\SOFTWARE\\Microsoft\\GPUPipeline;InstallLocation;Test

14.3.4 YARA Rules for Eventlog and Log Detection

Our scanners allow checking each Eventlog and log file entry with a YARA rules.

```
10/25 => Scanning Eventlog for malicious activity ... (1 - 60 mins)
This could take a while - depending on the number of entries
Info: Checking System Eventlog ...
Info: Eventlog lookback reached - skipping the rest of this log
Info: Eventlog System Entry Count: 267 in 1 days
Info: Checking Application Eventlog ...
Warning: YARA Rule Match NAME: Eventlog Test_Rule SCORE: 70 DESCRIPTION: Detects a cert
ain eventlog entry - EventID can be used as external variable MATCHED_STRINGS: Str1: gs
ecdump ENTRY: The program or feature "\\??\\M:\\real_life\\gsecdump.exe" cannot start or ru
n due to incompatibility with 64-bit versions of Windows. Please contact the software ven
dor to ask if a 64-bit Windows compatible version is available.
```

Figure 5 - Applied YARA rules to Eventlog entries

Strings defined in your rules will match on the full message texts as string. (not the XML format) The only external variable available is the ID of the event (EventCode) usable as "id".

```
rule Eventlog_Test_Rule
{
    meta:
        description = "Detects a certain eventlog entry - id can be used as
external variable"
        score = 70
    strings:
        $s1 = "gsecdump" ascii
    condition:
        $s1 and id == "1109"
}
```

Listing 11 – Eventlog YARA Rule Example

These rules are also applied to any type of log file scanning. (".log" file analysis)

14.3.5 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.

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

Listing 12 – False Positive Rule

14.4 Encrypt Custom Signatures

You can encrypt the YARA signatures and IOC files with the help of THOR-util's "encrypt" feature. In SPARK Core use the included "spark-core-util" instead of THOR-util.

```
thor-util.exe encrypt --help
```

Listing 13 – Show the help of the "encrypt" feature

| Flags for Output: | |
|--|---|
| --allreasons | Show all reasons and not only the top 2 |
| --brd | Suppress all PI in log outputs to comply with German data |
| -i, --case-id string | Define a unique case identifier (useful to filter on the |
| --cmdlog | Use logfile output's format for commandline output |
| -o, --csv string | MD5 List CSV of all relevant files (default "PROMETHEUS_ |
| sv") | |
| --debug | Debugging Output |
| --encrypt | Encrypt the logfile and csvfile |
| --json-file | Use json format for file-output |
| --keyval | Format output (logfile) with key value pairs to simplify |
| ion in SIEM systems (key='value') | |
| -l, --log string | Log file (default "PROMETHEUS_spark_2018-09-03.log") |
| -x, --min int | Minimum score to report (default 40) |
| --nocsv | Do not write a CSV of all mentioned files with MD5 hash |
| --nolog | Do not generate log file |
| -j, --overwrite-hostname string | Override the local hostname value with a static value (us |
| g mounted images in the Lab (default "PROMETHEUS") | |
| --printall | Show every single file that gets scanned (requires --debu |
| --printshim | Include all SHIM cache entries in the output as 'info' le |
| --pubkey string | Use a RSA public key to encrypt the logfile and csvfile. |
| or --pubkey="<file>") | |
| --silent | Do not print anything to command line |

Screenshot 6 - Encrypt output option in help

16 Links and References

Splunk App and Addon

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

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

Sigma

<https://github.com/Neo23x0/sigma>