



MOBIA is a business technology integrator with over thirty years of experience and 500+ employees across Canada and USA. Our talented bench of technical engineers and trusted advisors deliver process improvements and business transformations within our core pillars of **Cloud, Infrastructure, Software Development, Cybersecurity** and **Broadband & Wireless Services**.

Our inside-out approach allows us to understand business challenges from deep within the organization, mapping the impact as it ripples outward. This insight enables us to create future-proof solutions that maximize results and repeatedly exceed our client's expectations.

Adversary Simulation Workshop

Lab Guide: Sigma Rules

Objective:

Learn how the generic event detection language sigma is used to define event alerting rules.

Instructions:

These instruction below are taken from the write-up here:

<https://www.nextron-systems.com/2018/02/10/write-sigma-rules/>

Sigma is an open standard for rules that allow you to describe searches on log data in generic form. These rules can be converted and applied to many log management or SIEM systems and can even be used with grep on the command line.

In this article I'd like to give you a brief practical introduction into the rule creation process. I'll recommend some tools and draft a guide that helps you to write Sigma rules as quick and sound as possible.

Get the Repository

First download or clone our [Sigma repository](#) from Github.

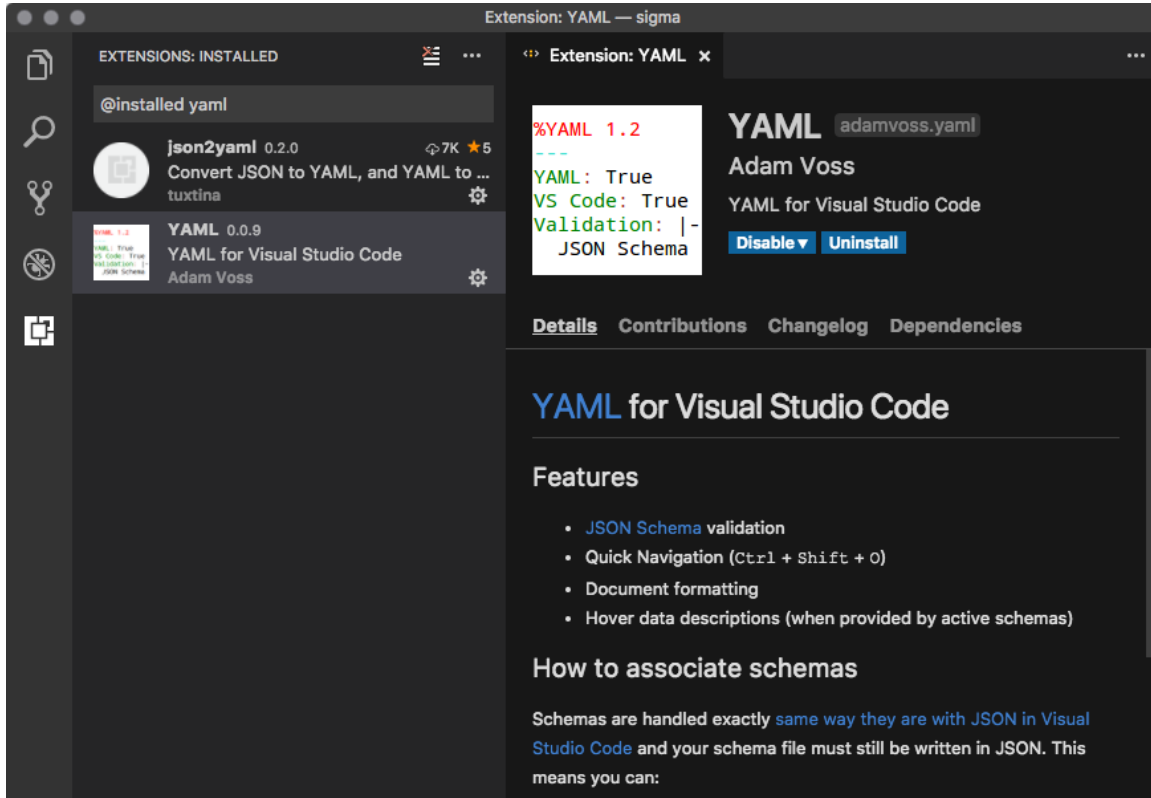
It contains the rule base in the folder “./rules” and the Sigma rule compiler “./tools/sigmac”. We will use the existing rules as examples and create a new rule based on a similar existing one. We will then test that rule by using “sigmac”.

The screenshot shows the GitHub repository for 'Neo23x0 / sigma'. At the top, it displays repository statistics: 112 Unwatch, 526 Stars, and 102 Forks. Below this is a navigation bar with links for Code, Issues (10), Pull requests (1), Projects (0), Wiki, Insights, and Settings. The repository description is 'Generic Signature Format for SIEM Systems', with tags for security, monitoring, siem, logging, signatures, elasticsearch, splunk, ids, sysmon, and a 'Manage topics' link. A progress bar shows 573 commits, 4 branches, 4 releases, and 13 contributors. Below the progress bar are buttons for 'Branch: master', 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. A table lists the repository's files and folders, including 'images', 'rules', 'tests', 'tools', '.gitignore', '.travis.yml', '.yamllint', and 'LICENSE.GPL.txt', each with a brief description and the time since the last update. A 'Clone with HTTPS' modal is open, showing the URL 'https://github.com/Neo23x0/sigma.git' and buttons for 'Open in Desktop' and 'Download ZIP'.

File/Folder	Description	Last Update
images	Added sigmac Screenshot	
rules	Rule: Msiexec web install	
tests	Finalizing PyPI release	
tools	Fixed before/after logic	8 days ago
.gitignore	IDE settings file	11 months ago
.travis.yml	Fixed Travis config	2 months ago
.yamllint	yamllint starter configuration, bad path for sigmac	7 months ago
LICENSE.GPL.txt	Re-licensing toolchain under LGPLv3	2 months ago

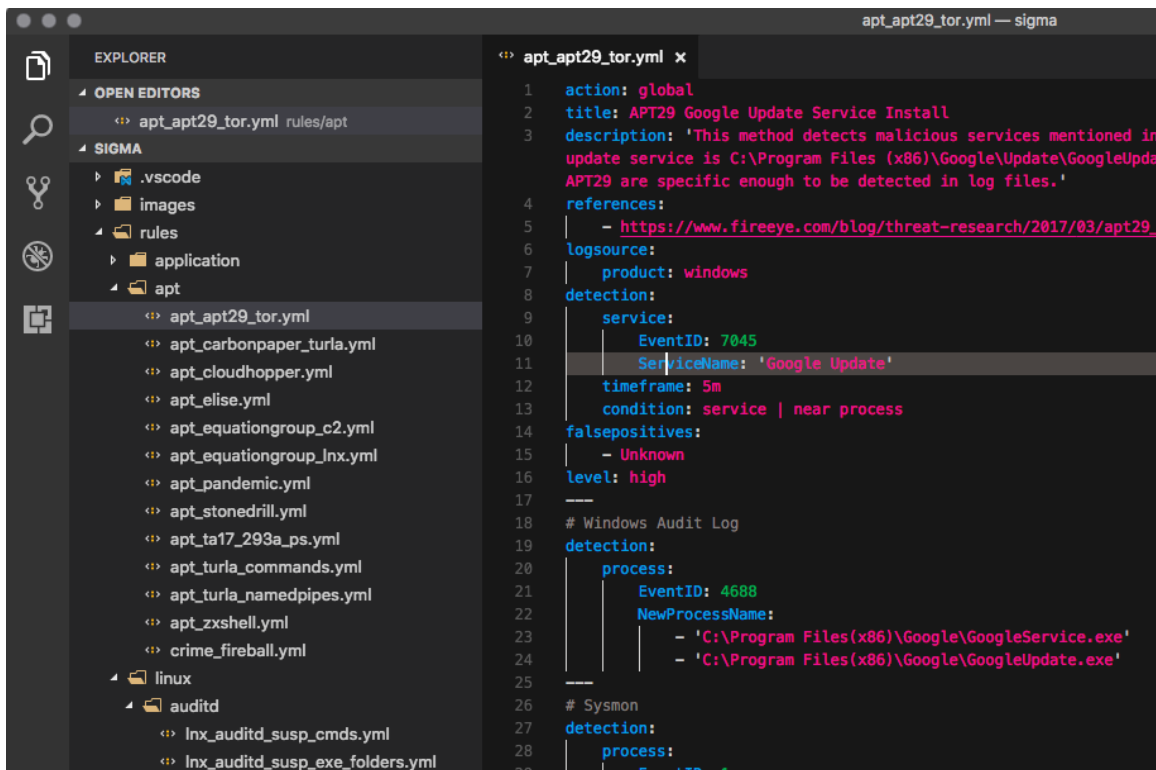
Sigma Github Repository

Copy and Edit YAML Files



YAML Extensions for VSCode

We open the Sigma repository folder with “Open ...” and see all existing rules.



Create a Sigma Rule

I selected an example in which we will create a Sigma rule from one of [@JPCERT](#)'s findings in their awesome "[Tool Analysis Result Sheet](#)".

We open the results for "[Quarks PWDump](#)", a password dumper often used by Chinese threat groups. It creates temporary files that we want to detect in our [SysInternals Sysmon](#) log data. To collect the needed events we use Sysmon with [@SwiftOnSecurity](#)'s [Sysmon config file](#), [Windows Event Forwarding](#) or [NXlog](#).

Quarks PWDump

- ☒ **Table of Contents**
- ☒ **Tool Overview**
- ☒ **Tool Operation Overview**
- ☒ **Information Acquired from Log**
- ☐ **Evidence That Can Be Confirmed When Execution is Successful**

- A temporary file ("SAM-[RANDOM].dmp") was created and deleted.

- **Main Information Recorded at Execution**

- + Host**

- Details: Host

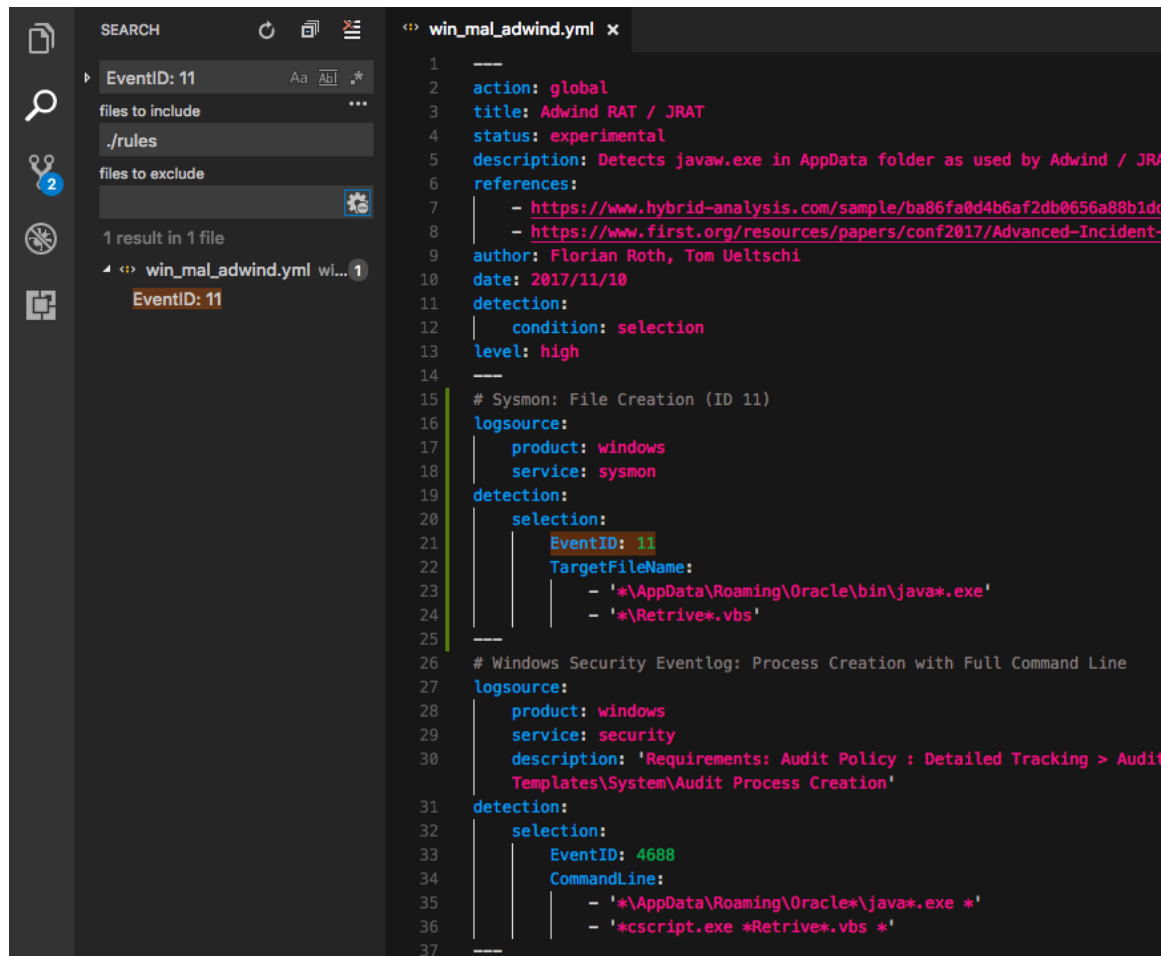
- #### **+ Event Log**

- USN Journal

#	File Name	Process
1	SAM-[RANDOM].dmp	FILE_CREATE
	SAM-[RANDOM].dmp	DATA_EXTEND+FILE_CREATE
	SAM-[RANDOM].dmp	DATA_EXTEND+DATA_OVERWRITE+FILE_CREATE
	SAM-[RANDOM].dmp	CLOSE+DATA_EXTEND+DATA_OVERWRITE+FILE_CREATE
	SAM-[RANDOM].dmp.LOG[NUM]	FILE_CREATE

Quarks PWDump Analysis Results

So, what we do is to find a Sigma rule in the repository that we can use as a template for our new rule. We use the 'search' function to find a rule that looks for "File Creation" events (EventID 11) in Sysmon log data.



```
1 ---
2 action: global
3 title: Adwind RAT / JRAT
4 status: experimental
5 description: Detects javaw.exe in AppData folder as used by Adwind / JRAT
6 references:
7   - https://www.hybrid-analysis.com/sample/ba86fa0d4b6af2db0656a88b1d...
8   - https://www.first.org/resources/papers/conf2017/Advanced-Incident-...
9 author: Florian Roth, Tom Ueltschi
10 date: 2017/11/10
11 detection:
12   condition: selection
13   level: high
14 ---
15 # Sysmon: File Creation (ID 11)
16 logsource:
17   product: windows
18   service: sysmon
19 detection:
20   selection:
21     EventID: 11
22     TargetFileName:
23       - '*\AppData\Roaming\Oracle\bin\java*.exe'
24       - '*\Retrive*.vbs'
25 ---
26 # Windows Security Eventlog: Process Creation with Full Command Line
27 logsource:
28   product: windows
29   service: security
30   description: 'Requirements: Audit Policy : Detailed Tracking > Audit
31     Templates\System\Audit Process Creation'
32 detection:
33   selection:
34     EventID: 4688
35     CommandLine:
36       - '*\AppData\Roaming\Oracle*\java*.exe *'
37       - '*cscript.exe *Retrive*.vbs *'
```

Sigma Example Rule

We find a rule that has a special format. It is a so-called “[rule-collection](#)“, which allows us to define a global section in the YAML file marked with “action: global” that will be applied to all other sections in that file during the search query generation process. This way you can define and create multiple search queries from a single YAML file.

Note:

If you are having difficulty identifying a sample rule to work with this one is a good place to start:

https://github.com/SigmaHQ/sigma/blob/master/rules/windows/process_creation/win_1sass_dump.yml

In the case of our QuarksPwDump example we don't need a rule collection, so we reduce the rule to a standard rule that contains a detection expression looking Sysmon Events with

Event ID 11 and save it as “sysmon_quarkspw_filedump.yml” to a new file in the folder “./rules/windows/sysmon/”.

```
sysmon_quarkspw_filedump.yml x
1  title: Adwind RAT / JRAT
2  status: experimental
3  description: Detects javaw.exe in AppData folder as used by Adwind / JRAT
4  references:
5    - https://www.hybrid-analysis.com/sample/ba86fa0d4b6af2db0656a88b1dd29f36fe
6    - https://www.first.org/resources/papers/conf2017/Advanced-Incident-Detecti
7  author: Florian Roth, Tom Ueltschi
8  date: 2017/11/10
9  level: high
10 logsource:
11   product: windows
12   service: sysmon
13 detection:
14   selection:
15     # Sysmon: File Creation (ID 11)
16     EventID: 11
17     TargetFileName:
18       - '*\AppData\Roaming\Oracle\bin\java*.exe'
19       - '*\Retrive*.vbs'
20
```

Simple Sysmon Sigma Rule

After that, we modify several fields of that rule:

- We give the rule a correct “title” and “description”
- We leave the status “experimental” to inform everyone that this is a new and untested rule
- We add the correct reference to the source from which we derived that rule
- We change the author of the rule
- We set the level of that rule to one of “low”, “medium”, “high” or “critical”
- We adjust the date (of last modification) and use the format %Y/%m%d (strftime)
- We check if the log source is correct, which is important for the [field mappings](#) used by “sigmac”


```

<> sysmon_quarkspw_filedump.yml x
1  title: QuarksPwDump Dump File
2  status: experimental
3  description: Detects a dump file written by QuarksPwDump
4  references:
5  |   - https://jpcertcc.github.io/ToolAnalysisResultSheet/details/QuarksPwDump.htm
6  author: Florian Roth
7  date: 2018/02/10
8  level: critical
9  logsource:
10 |   product: windows
11 |   service: sysmon
12 detection:

```

New Sigma Rule Header

Before we create the new “detection” section, we review the analysis report in detail.

USN journal

#	File Name	
1	SAM-[RANDOM].dmp	CLOSE+FILE_DELETE
2	SAM-[RANDOM].dmp.LOG[NUM]	CLOSE+DATA_EXTEND+
3	SAM-[RANDOM].dmp{[GUID]}.TM.blf	DATA_EXTEND+DATA_O
4	SAM-[RANDOM].dmp{[GUID]}.TMContainer[NUM].regtrans-ms	CLOSE+DATA_EXTEND+

MFT

#	Path
1	[Drive Name]:\Users\[User Name]\AppData\Local\Temp\SAM-[NUM].dmp.LOG[NUM]
2	[Drive Name]:\Users\[User Name]\AppData\Local\Temp\SAM-[NUM].dmp{[GUID]}.TM.blf
3	[Drive Name]:\Users\[User Name]\AppData\Local\Temp\SAM-[NUM].dmp{[GUID]}.TMContainer[NUM].regtrans-ms

Details: QuarksPwDump Temporary Files

We add a string with wildcards that matches on the ‘TargetFilename’ field in the Sysmon events of type 11.

That’s what the new rule looks like:

```

1  title: QuarksPwDump Dump File
2  status: experimental
3  description: Detects a dump file written by QuarksPwDump password dumper
4  references:
5  |   - https://jpcertcc.github.io/ToolAnalysisResultSheet/details/QuarksPwDump.htm
6  author: Florian Roth
7  date: 2018/02/10
8  level: critical
9  logsource:
10 |   product: windows
11 |   service: sysmon
12 detection:
13 |   selection:
14 |     # Sysmon: File Creation (ID 11)
15 |     EventID: 11
16 |     TargetFileName: '*\AppData\Local\Temp\SAM-*.dmp*'
17 |   condition: selection
18 falsepositives:
19 |   - Unknown
20

```

QuarksPwDump Sigma Rule

Test the Rules

We test our newly created rule with “sigmac”, which requires python3. It is located in the “./tools” folder. It features several targets for which we can create searches/configurations from our rules.

Currently supported targets (10.02.2018):

- es-q3 (Elastic Search Query Language)
- kibana
- xpack-watcher
- logpoint
- splunk
- grep
- fieldlist (only used to show all fields that require mapping in a config file)

Running “python3 sigmac -h” shows a help:

```

$ python3 sigmac -h
usage: sigmac [-h] [--recurse] [--filter FILTER]
              [--target {es-q3,kibana,xpack-watcher,logpoint,splunk,grep,fieldlist}]
              [--target-list] [--config CONFIG] [--output OUTPUT]

```

```
[--backend-option BACKEND_OPTION] [--defer-abort]
[--ignore-not-implemented] [--verbose] [--debug]
[inputs [inputs ...]]
```

Convert Sigma rules into SIEM signatures.

positional arguments:

inputs Sigma input files

optional arguments:

```
-h, --help                show this help message and exit
--recurse, -r             Recurse into subdirectories (not yet implemented)
--filter FILTER, -f FILTER
                        Define comma-separated filters that must match (AND-
                        linked) to rule to be processed. Valid filters:
                        level<=x, level>=x, level=x, status=y, logsource=z. x
                        is one of: low, medium, high, critical. y is one of:
                        experimental, testing, stable. z is a word appearing
                        in an arbitrary log source attribute. Multiple log
                        source specifications are AND linked.
--target {es-qs,kibana,xpack-watcher,logpoint,splunk,grep,fieldlist}, -t {es-
qs,kibana,xpack-watcher,logpoint,splunk,grep,fieldlist}
                        Output target format
--target-list, -l        List available output target formats
--config CONFIG, -c CONFIG
                        Configuration with field name and index mapping for
                        target environment (not yet implemented)
--output OUTPUT, -o OUTPUT
                        Output file or filename prefix if multiple files are
                        generated (not yet implemented)
--backend-option BACKEND_OPTION, -O BACKEND_OPTION
                        Options and switches that are passed to the backend
--defer-abort, -d        Don't abort on parse or conversion errors, proceed
                        with next rule. The exit code from the last error is
                        returned
--ignore-not-implemented, -I
                        Only return error codes for parse errors and ignore
                        errors for rules with not implemented features
--verbose, -v            Be verbose
--debug, -D             Debugging output
```

We test our new rule with “sigmac” and the target “splunk”.

```
$ python3 sigmac -t splunk ../rules/windows/sysmon/sysmon_quarkspw_dump.yml
(EventID="11" TargetFilename="*\\AppData\\Local\\Temp\\SAM-*.dmp*")
```

Now the rule is ready for a pull request.

End of Lab