



Log Hunting with Sigma

A hands-on introduction to Sigma rules and the conversion tool

Thomas Patzke, 17. October 2018

Prerequisites

Requirements:

- Python 3.5 or 3.6
<https://www.python.org/downloads/release/python-365/>
- Docker CE (current version)
- Clone of the Sigma workshop repository: <https://github.com/thomaspatzke/sigma-workshop>

```
git clone --recursive \  
https://github.com/thomaspatzke/sigma-workshop.git
```
- **Sigma dependencies:**

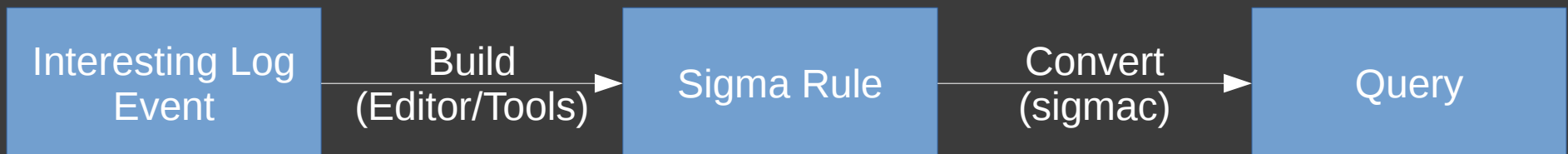
```
pip3 install -r sigma/tools/requirements.txt  
or apt-get install python3-yaml
```
- **Elasticsearch and Kibana with log data:**
 - `docker load -i sigma-workshop-docker.tar`
 - `docker-compose -f es_kibana.docker-compose.yml up`
 - `./sigma_workshop_prepare_es.sh`

Overview

- A short (!) introduction to Sigma
- Writing a log Signature for:
 - Execution of a Mimikatz release binary (process execution by hash)
 - Common parameter usage of NirSoft's NetPass tool (process execution by command line)
 - WCE LSASS injection behaviour
- Building a Sigma Converter configuration
- Convert to Elasticsearch query and search ELK instance
- Generic log sources

Sigma Introduction

- Generic signature format for description of log events
 - YAML-based
 - Indicators: Key-Value, Key-List and Value-only
 - Conditions and aggregations
 - Meta-data: Title, description, authors, tags (ATT&CK), severity, ...
- Conversion tool sigmac
- Workflow:



Sigma Rule – Example 1

```
title: Shells Spawned by Web Servers
status: experimental
description: Web servers that spawn shell processes could be the result of
author: Thomas Patzke
```

```
logsource:
  product: windows
  service: sysmon
```

```
detection:
  selection:
    EventID: 1
    ParentImage:
      - '*\w3wp.exe'
      - '*\httpd.exe'
      - '*\nginx.exe'
      - '*\php-cgi.exe'
    Image:
      - '*\cmd.exe'
      - '*\sh.exe'
      - '*\bash.exe'
      - '*\powershell.exe'
  condition: selection
```

```
fields:
  - CommandLine
  - ParentCommandLine
```

```
falsepositives:
  - Particular web applications may spawn a shell process legitimately
```

```
level: high
```

Log source definition
Scope generated query
by

- restriction to indices
- addition of conditions

Rule metadata
(purely descriptive)

Values to search in specific fields of log data,
grouped in selections

Selections are linked in a condition

Fields that are particularly interesting and
should be displayed in search results

Severity of matches, may be used for filtering rules

Sigma Rule – Example 2

```
title: Rundll32 Internet Connection
status: experimental
description: Detects a rundll32 that com
references:
- https://www.hybrid-analysis.com/sa
author: Florian Roth
date: 2017/11/04
tags:
- attack.t1085
- attack.defense_evasion
- attack.execution
logsource:
  product: windows
  service: sysmon
detection:
  selection:
    EventID: 3
    Image: '*\rundll32.exe'
  filter:
    DestinationIp:
      - '10.*'
      - '192.168.*'
      - '172.*'
condition: selection and not filter
falsepositives:
- Communication to other corporate s
level: medium
```

Tagging of rules with ATT&CK tactics, techniques and software tags. Can be used for filtering of rules.

Usual condition: search for selection and filter uninteresting events.

Advantages

- Reduced vendor lock-in
- Distribution of log signatures in heterogeneous environments or in blog posts/threat intel products
- Build one rule and use it in your SIEM, alerting, endpoint security solution or even for grepping in log files and querying from PowerShell
- 220+ Sigma rules in GitHub repository
- Evolving tool/services support: MISP conversion extension, online editor, ...
- Intermediate language for generation of queries from IOCs in other formats
- Increasing community contribution

Sigma Goals and Scope

- Being human-writable and readable
 - No XML or JSON, no deeply nested structures
- Machine-readable and writable
 - YAML, no ambiguities
- Simpleness
 - Expressiveness for 95% of log signatures
 - NOT: description of every imaginable SIEM use case or threat hunting technique
 - It should be relatively easy to build an own Sigma parser
- Tooling: it should be practicably usable, not just theory

Enough Theory!

Let's get our hands dirty!

Exercise 1

Mimikatz Release Binary

- Let's assume we're targeted by an attacker who is known to use the Mimikatz 2.1.1 release
- SHA256 hashes (see challenges/1-Mimikatz_2.1.1_Hashes.txt):
 - 97f93fe103c5a3618b505e82a1ce2e0e90a481aae102e52814742badd
d9fed41 ./Win32/mimilove.exe
 - 6bfc1ec16f3bd497613f57a278188ff7529e94eb48dcabf81587f7c275
b3e86d ./Win32/mimikatz.exe
 - e46ba4bdd4168a399ee5bc2161a8c918095fa30eb20ac88cac6ab1d6
dbea2b4a ./x64/mimikatz.exe
- Write a rule for Sysmon events that detects execution of the above binaries (EventID 1) by utilization of the *Hashes* field

Exercise 1

Possible Solution

```
title: Mimikatz detection
status: stable
description: Detects Mimikatz 2.1.1 release by recognition of executable hashes
tags:
  - attack.s0002
  - attack.t1003
  - attack.lateral_movement
  - attack.credential_access
logsource:
  product: windows
  service: sysmon
detection:
  selection:
    EventID: 1
    Hashes:
      - 97f93fe103c5a3618b505e82a1ce2e0e90a481aae102e52814742badd9fed41
      - 6bfc1ec16f3bd497613f57a278188ff7529e94eb48dcabf81587f7c275b3e86d
      - e46ba4bdd4168a399ee5bc2161a8c918095fa30eb20ac88cac6ab1d6dbea2b4a
    condition: selection
level: high
```

Rule Conversion with Sigma Converter

- The Sigma Converter (sigmac) is located in tools/sigmac in the Sigma repository
- Run it with `--help` to get an overview
- Convert into target query language (-t) *es-qs* (Elasticsearch Query String)
- No matches! Why?
 - Sigma rules are (or should be) generic, so some further work is required
 - Mapping of field names:
 - EventID → event_id
 - Hashes → event_data.Hashes
 - EventID 1 may also appear from other sources, search needs to be restricted to log source by addition of further conditions
- Sigma conversion configuration defines the transformation

Sigma Converter Configuration

```
Logsources:  
  windows:  
    product: windows  
    index: winlogbeat-*  
  windows-sysmon:  
    product: windows  
    service: sysmon  
    conditions:  
      log_name: 'Microsoft-Windows-Sysmon/Operational'
```

This section describes log sources and is matched against the logsource definition from the Sigma rule

Matches to all rules where product is windows

All Windows logs are in indices matching the Pattern winlogbeat-*

All queries of Windows Sysmon logs should be restricted to events where the field log_name is set to *Microsoft-Windows-Sysmon/Operational*

```
defaultindex: winlogbeat-*
```

Fallback index if no logsource matches

```
fieldmappings:
```

Multiple matching log source definitions are accumulated

```
  TargetImage: event_data.TargetImage
```

Mapping from fieldnames in Sigma rule to these in target system.

Try Again – with Configuration!

- Try to write your own configuration
- Configurations can be passed to Sigma converter with parameter -c

Further Exercises

- Exercise 2: NirSoft NetPass
 - NetPass has some very characteristic parameter names: /stext, /stab, /scomma, /stabular, /shtml, /sverhtml, /sxml
 - Write a rule for Sysmon process creation events and utilize the *CommandLine* field for identification of parameter usage, don't:
 - Try to match hashes of any releases
 - Match the file name
- Exercise 3: WCE LSASS Injection
 - WCE causes a burst of Sysmon CreateRemoteThread (EventID 8) events into lsass.exe (TargetImage)
 - Further, some security products also inject into LSASS, but only WCE does without a *StartModule*. Filter these out.

Exercise 2

Possible Solution

```
title: Detection of Nirsoft NetPass parameter usage
status: stable
description: NetPass supported some characteristic parameters
tags:
  - attack.credential_access
  - attack.t1003
logsource:
  product: windows
  service: sysmon
detection:
  selection:
    EventID: 1
    CommandLine:
      - "* /stext"
      - "* /stab"
      - "* /scomma"
      - "* /stabular"
      - "* /shtml"
      - "* /sverhtml"
      - "* /sxml"
    condition: selection
level: high
```


Exercise 3

Possible Solution

```
title: WCE Remote Thread Injection
status: stable
description: Detection of remote thread creation in LSASS by Windows Credential Editor
tags:
  - attack.credential_access
  - attack.t1003
  - attack.s0005
logsource:
  product: windows
  service: sysmon
detection:
  selection:
    EventID: 8
    TargetImage: 'C:\Windows\System32\lsass.exe'
  filter:
    StartModule: '*'
  condition: selection and not filter
level: high
```

Handling many Sigma Rules

- Copy and pasting rules between terminal and browser is not very convenient.
- Build a Kibana import file from all previous solutions with the *kibana* backend
- Import the generated file into Kibana

Generic Log Sources: Introduction

- There are different EventIDs for the same events
 - Process execution: Sysmon/1 and Security/4688
- Products that recognize such events, but don't know about these EventIDs
 - Windows Defender ATP and various other EDR products
- Causes
 - Redundancy: multiple rules for the same event
 - Inconsistency: one rule for a event id that may be recognized by another
 - Complex conversion (matching all EventIDs to target query language objects)

Generic Log Sources: Example

```
title: Bitsadmin Download
status: experimental
description: Detects usage of bits
references:
  - https://blog.netSPI.com/15-w
  - https://isc.sans.edu/diary/2
tags:
  - attack.defense_evasion
  - attack.persistence
  - attack.t1197
  - attack.s0190
author: Michael Haag
logsource:
  product: windows
  service: sysmon
detection:
  selection:
    EventID: 1
    Image:
      - '*\bitsadmin.exe'
    CommandLine:
      - '/transfer'
  condition: selection
fields:
  - CommandLine
  - ParentCommandLine
falsepositives:
  - Some legitimate apps use thi
level: medium
```

```
title: Bitsadmin Download
status: experimental
description: Detects usage of bitsa
references:
  - https://blog.netSPI.com/15-wa
  - https://isc.sans.edu/diary/22
tags:
  - attack.defense_evasion
  - attack.persistence
  - attack.t1197
  - attack.s0190
author: Michael Haag
logsource:
  category: process_creation
  product: windows
detection:
  selection:
    Image:
      - '*\bitsadmin.exe'
    CommandLine:
      - '/transfer'
  condition: selection
fields:
  - CommandLine
  - ParentCommandLine
falsepositives:
  - Some legitimate apps use this
level: medium
```



Generic Log Sources: State & Usage

- Current state: open for testing
 - Many rules have to be converted
 - Project branch: project-1
(<https://github.com/Neo23x0/sigma/tree/project-1>)
- Usage: chained configurations
 1. Generic rule → specific
Process creation → Sysmon/1
 2. Specific rule → Environment-specific rule
Sysmon/1 → Sysmon/1 with field mappings and additional conditions
- Configuration for process creation to Sysmon already exists
- Let's try it!
 - Sigma Converter with generic log source support in directory `sigma_with_generic_logsources/`

Questions?

- E-Mail: thomas@patzke.org
- Twitter: [@blubbfiction](https://twitter.com/blubbfiction)