

# Advanced Wazuh Decoder Builder – Complete User Guide

---

Author: Mariskarthick M

Website: <https://decoder.mariskarthick.in>

Date: April 04, 2025

---

## What This Tool Does

The Advanced Wazuh Decoder Builder is an interactive tool that generates multi-layered XML decoders for Wazuh based on your log samples. It supports:

- Manual regex decoders
- Pure GROK pattern decoders
- Hybrid mode: GROK base + child decoders
- Smart spacing detection, timestamp handling, and PCRE2 support

## Sample Log to Work With

```
Jan 24 11:01:28 5.195.107.202 0|Fortinet|FortiGate-300E|7.2.4,build1396  
(GA)|0000000013|forward traffic close|5|start=Jan 18 2024 21:33:33 logid=0102201  
deviceExternalId=FG2K0E12345W2924823
```

## Mode 1: GROK Mode (ON)

How to Use:

1. Enable GROK Mode in the UI (checkbox).

2. Paste the following GROK pattern:

```
%{SYSLOGTIMESTAMP:timestamp} %{IP:src_ip}  
%{NUMBER:session_id}\|%{WORD:vendor}\|%{DATA:firewall_model}\|%{DATA:ver  
sion}\|%{NUMBER:event_id}\|%{DATA:message}\|%{NUMBER:severity}\|start=%{D  
ATE_DMY:start_date} %{TIME:start_time} logid=%{NUMBER:logid}  
deviceExternalId=%{WORD:deviceExternalId}
```

3. Paste the sample log.

4. (Optional) Add a row:

- decoderName: grok\_decoder
- tag: prematch

- sampleValue: Jan 24 11:01:28

5. Click Generate Decoder XML.

## Output for GROK Mode

```
<decoder name="grok_decoder">
<prematch>Jan 24 11:01:28</prematch>
<regex pcre2="yes">...converted pattern...</regex>
<order>timestamp,src_ip,session_id,vendor,...</order>
</decoder>
```

The screenshot shows the Mariskarthick's Advanced Powerful Wazuh Decoder Builder interface. It has three main sections:

- 1. Paste Your Log Sample**: A text area containing a log entry: "Jan 24 11:01:28 5.195.107.202 0|Fortinet|FortiGate-300E|7.2.4.build1396 (GA)|0000000013|forward traffic close|5|start=Jan 18 2024 21:33:33 logid=0102201 deviceExternalId=FG2k0E12345W2924823". Below it is a checkbox labeled "Enable GROK Mode" which is checked.
- 2. Review & Edit Field Mappings**: A table with columns: Decoder Name, Tag, Field Name, Sample Value, Custom Mapping, Offset, and Actions. There is a "Add Row" button at the top left of the table.
- 3. Generate Decoder XML**: A text area containing the generated XML code for the decoder.

```
<?xml version="1.0" ?>
<decoders>
  <decoder name="grok_decoder">
    <prematch>Jan 24 11:01:28</prematch>
    <regex pcre2="yes">(?P<timestamp>[A-Za-z]{3} \d{1,2} \d{2}:\d{2}:\d{2}) (?P<src_ip>.+?) (?P<session_id>\d+)|(?P<vendor>\w+)|(?P<firewall_model>.*?)|(?P<version>.*?)|(?P<event_id>\d+)|(?P<message>.*?)|(?P<severity>\d+)|(?P<start_date>\d{1,2} [A-Za-z]{3} \d{4}) (?P<start_time>\d{2}:\d{2}:\d{2}) logid=(?P<logid>\d+) deviceExternalId=(?P<deviceExternalId>\w+)</regex>
    <order>timestamp,src_ip,session_id,vendor,firewall_model,version,event_id,message,severity,start_date,start_time,logid,deviceExternalId</order>
  </decoder>
</decoders>
```

## ⌚ Mode 2: Hybrid Mode (GROK + Regex Rows)

Steps:

1. Enable GROK Mode.

2. Add GROK pattern and sample log.

3. Add rows:

- Row 1: decoderName=grok\_decoder, tag=prematch, sampleValue=Jan 24 11:01:28

- Row 2: decoderName=child\_decoder1, tag=regex, fieldName=deviceExternalId=, sampleValue=FG2K0E12345W2924823, customMapping=device\_id, offset=after\_parent

**1. Paste Your Log Sample**

```
Jan 24 11:01:28 5.195.107.202 0|Fortinet|FortiGate-300E|7.2.4,build1396 (GA)|00000000013|forward traffic close|5|start=Jan 18 2024 21:33:33 logid=0102201
deviceExternalId=FG2K0E12345W2924823
```

Enable GROK Mode

```
%{SYSLOGTIMESTAMP:timestamp} %{IP:src_ip} %{NUMBER:session_id}\| %{WORD:vendor}\| %{DATA:firewall_model}\| %{DATA:version}\| %{NUMBER:event_id}\| %{DATA:message}\|%
{NUMBER:severity}\| start=%{DATE_DMY:start_date} %{TIME:start_time} logid=%{NUMBER:logid}
```

Analyze Log

**2. Review & Edit Field Mappings**

Decoder Name	Tag	Field Name	Sample Value	Custom Mapping	Offset	Actions
grok_decoder	prematch		Jan 24 11:01:28			<button>Remove</button>
child_decoder1	regex	deviceExternalId=	FG2K0E12345W2924823	device_id	after_parent	<button>Remove</button>

Add Row

**3. Generate Decoder XML**

Generate Decoder XML

```
<?xml version="1.0" ?>
<decoders>
  <decoder name="grok_decoder">
    <prematch>Jan 24 11:01:28</prematch>
    <regex pcre2="yes">(?P<timestamp>[A-Za-z]{3} +\d{1,2} \d{2}:\d{2}) (?P<src_ip>.+) (?P<session_id>\d+)\\|(?P<vendor>\w+)\\|(?P<firewall_model>.*?)\\|(?P<version>.*?)\\|(?P<event_id>\d+)\\|(?P<message>.*?)\\|(?P<severity>\d+)\\|start=(?P<start_date>\d{1,2} [A-Za-z]{3} \d{4}) (?P<start_time>\d{2}:\d{2}:\d{2}) logid=(?P<logid>\d+)</regex>
    <order>timestamp,src_ip,session_id,vendor,firewall_model,version,event_id,message,severity,start_date,start_time,logid</order>
  </decoder>
</decoders>
```

## Output for Hybrid Mode

```
<?xml version="1.0" ?>
<decoders>
  <decoder name="grok_decoder">
    <prematch>Jan 24 11:01:28</prematch>
    <regex pcre2="yes">(?P<timestamp>[A-Za-z]{3} +\d{1,2} \d{2}:\d{2}:\d{2}) (?P<src_ip>.+) (?P<session_id>\d+)\\|(?P<vendor>\w+)\\|(?P<firewall_model>.*?)\\|(?P<version>.*?)\\|(?P<event_id>\d+)\\|(?P<message>.*?)\\|(?P<severity>\d+)\\|start=(?P<start_date>\d{1,2} [A-Za-z]{3} \d{4}) (?P<start_time>\d{2}:\d{2}:\d{2}) logid=(?P<logid>\d+)</regex>
    <order>timestamp,src_ip,session_id,vendor,firewall_model,version,event_id,message,severity,start_date,start_time,logid</order>
  </decoder>
</decoders>
```

## Mode 3: Manual Regex Mode (GROK Mode OFF)

Steps:

1. Turn GROK Mode OFF.
2. Add sample log.
3. Add rows:
  - Row 1: decoderName=fortinet\_root, tag=prematch, sampleValue=Jan 24 11:01:28
  - Row 2: decoderName=fortinet\_child, tag=parent, sampleValue=fortinet\_root
  - Row 3: decoderName=fortinet\_child, tag=regex, fieldName=logid=, sampleValue=0102201, customMapping=logid, offset=after\_parent
  - Row 4: decoderName=fortinet\_child, tag=regex, fieldName=deviceExternalId=, sampleValue=FG2K0E12345W2924823, customMapping=device\_id, offset=after\_regex

## Output for Manual Regex

```
<decoder name="fortinet_root">
  <prematch>Jan 24 11:01:28</prematch>
</decoder>
<decoder name="fortinet_child">
  <parent>fortinet_root</parent>
  <regex offset="after_parent">logid=(?P<logid>\d+)</regex>
  <regex offset="after_regex">deviceExternalId=(?P<device_id>\S+)</regex>
  <order>logid,device_id</order>
</decoder>
```

## Advanced Features Summary

Feature	How It Works
<b>Spacing Detection</b>	Field name like logid= auto-detects space gap: adds \s+ or \s{N}
<b>PCRE2 Regex</b>	All regexes use <regex pcre2="yes"> when GROK or dynamic
<b>Offset Logic</b>	Use after_parent after <prematch>, and after_regex for chaining
<b>Timestamp Detection</b>	Automatically matches known formats, e.g., SYSLOGTIMESTAMP, DATE_DMY + TIME
<b>Multi-Decoder Support</b>	Decoders are split into root + children based on <parent>
<b>Field Mapping</b>	customMapping → Wazuh field name (srcip, msg, logid, etc.)

## Deployment

1. Save decoder XML as .xml file in /var/ossec/decoders.d/
2. Restart Wazuh: systemctl restart wazuh-manager
3. Test: /var/ossec/bin/wazuh-logtest

## Bonus Tips

- For complex nested logs, break into root decoder + child decoders.
- Always test your GROK patterns separately if unsure <https://grokdebugger.com/>
- Use <type> and <description> tags if needed.
- Avoid overlapping decoder names in multi-decoder builds.