

# Alert Generation

Suricata Core Functions



# Generating Alerts

- To ensure Suricata is up and running, you can add rules to generate alerts which you know you can trigger.
- An alert will fire when a rule is triggered, that is, when network traffic falls into certain patterns described in one of the Suricata rules.
- The image shows what an alert looks like when pretty printed from `eve.json`:

- `cat eve.json | jq`

```
{
  "timestamp": "2021-05-17T16:46:19.224736+0100",
  "flow_id": 491376735907296,
  "ip_iface": "eth0",
  "event_type": "alert",
  "src_ip": "fe02:0000:0000:0000:95c4:f11a:0559:1b78",
  "src_port": 0,
  "dest_ip": "ff02:0000:0000:0000:0000:0000:0000:0016",
  "dest_port": 0,
  "proto": "IPv6-ICMP",
  "icmp_type": 143,
  "icmp_code": 0,
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 1,
    "rev": 1,
    "signature": "ICMP Packet found",
    "category": "",
    "severity": 3
  },
  "flow": {
    "pkts_to_server": 1,
    "pkts_to_client": 0,
    "bytes_to_server": 150,
    "bytes_to_client": 0,
    "start": "2021-05-17T16:46:19.224736+0100"
  }
}
```

# Generating Alerts Con't

- This is a very basic rule that will alert when an ICMP Echo (ping) is detected:
  - `alert icmp any any -> any any (msg: "ICMP Packet found"; sid: 1; rev: 1;)`
- You can add personal rules to `suricata.rules` or create a `local.rules` file and add the path to it to your `suricata.yaml` file.

```
## Configure Suricata to load Suricata-Update managed rules.
##

default-rule-path: /usr/local/var/lib/suricata/rules

rule-files:
- suricata.rules
- local.rules
```

# Checking Logs

---

- To confirm that alerts were generated, check Suricata logs after running the engine.
- `Fast.log` is an alternative that allows you to specifically check the generated alerts.
- For a quick check on the latest alerts:
  - `tail /path/to/log/suricata/fast.log`
  - `05/11/2021-12:34:41.554752 [**] [1:1:1] ICMP Packet found`  
`[**] [Classification: (null)] [Priority: 3] {IPv6-ICMP}`  
`fe80:0000:0000:0000:0000:0000:0000:0001:134 →`  
`ff02:0000:0000:0000:0000:0000:0000:0001:0`



# Checking Logs Con't

- For a more detailed and useful log, we can and should use `eve.json`.
- This will also log flow id:

```
cat /path/to/log/suricata/eve.json | jq -c 'select(.event_type ==  
"alert")'
```

```
{  
  "timestamp": "2021-05-  
11T12:35:29.044173+0100",  
  "flow_id": 1379446523276429,  
  "in_iface": "enp3s0",  
  "event_type": "alert",  
  "src_ip": "fe80:0000:0000:0000:0000:0000:0000:0001",  
  "src_port": 0,  
  "dest_ip": "ff02:0000:0000:0000:0000:0000:0000:0001",  
  "dest_port": 0,  
  "proto": "IPv6-  
ICMP",  
  "icmp_type": 134,  
  "icmp_code": 0,  
  "alert": {  
    "action": "allowed",  
    "gid": 1,  
    "signature_id": 1,  
    "rev": 1,  
    "signature": "ICMP Packet  
found",  
    "category": "",  
    "severity": 3  
  },  
  "flow": {  
    "pkts_toserver": 1,  
    "pkts_toclient": 0,  
    "bytes_toserver": 118,  
    "bytes_toclient": 0,  
    "start": "2021-05-  
11T12:35:29.044173+0100"  
  }  
}
```

# Eve.json Log Fields

- All the JSON log types share a common structure, with data for timestamp, flow tracking, log type and more in depth data based on event type.
  - **timestamp**: event timestamp (if reading from pcap, this is from the packet).
  - **pcap\_cnt**: contains the packet number of the pcap (when in pcap mode), this matches what shows in Wireshark
  - **app\_proto**: application layer protocol inspected (optional)

```
{
  "timestamp": "2017-10-09T17:10:39.136184+0100",
  "flow_id": 874599751228065,
  "pcap_cnt": 154,
  "event_type": "alert",
  "src_ip": "192.168.10.31",
  "src_port": 49238,
  "dest_ip": "192.168.10.30",
  "dest_port": 445,
  "proto": "TCP",
  "metadata": {
    "flowbits": [
      "ET.smb.binary"
    ]
  },
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 2025701,
    "rev": 2,
    "signature": "ET POLICY SMB2 NT Create AndX Request For an Executable File",
    "category": "Potentially Bad Traffic",
    "severity": 2
  },
  "app_proto": "smb"
}
```

For more: <https://suricata.readthedocs.io/en/latest/output/eve/eve-json-format.html>

# Eve.json Log Fields Con't

- **src\_ip**: ip address of the source
- **src\_port**: port used by the source
- **dest\_ip**: ip address of the destination
- **dest\_port**: port used to destination
- **flow\_id**: uniquely identifies the traffic exchange based on IPs, ports and protocol (5-tuple)

```
{
  "timestamp": "2021-05-17T16:45:26.214863+0100",
  "flow_id": 2070850955528015,
  "in_iface": "enp3s0",
  "event_type": "alert",
  "src_ip": "fe80:0000:0000:0000:0000:0000:0000:0001",
  "src_port": 0,
  "dest_ip": "ff02:0000:0000:0000:0000:0000:0000:0001",
  "dest_port": 0,
  "proto": "IPv6-ICMP",
  "icmp_type": 134,
  "icmp_code": 0,
  "alert": {
    "action": "allowed",
    "gid": 1,
    "signature_id": 1,
    "rev": 1,
    "signature": "ICMP Packet found",
    "category": "",
    "severity": 3
  },
  "flow": {
    "pkts_toserver": 1,
    "pkts_toclient": 0,
    "bytes_toserver": 118,
    "bytes_toclient": 0,
    "start": "2021-05-17T16:45:26.214863+0100"
  }
}
```

For more: <https://suricata.readthedocs.io/en/latest/output/eve/eve-json-format.html>

# Eve.json Log Fields Con't

- **in\_iface**: network interface name (in live mode)
- **event\_type**: indicates the log type. For alerts, that will be “alert”
- **proto**: network protocol
- **alert**: this field is based on event\_type. If the record is for an alert, for instance, it will have a field “alert”, containing data specific to alert records
- **flow**: (optional) flow related stats, like number of packets and bytes exchanged

# Eve.json log fields - Alert Type

- Alert type specific fields:
  - **action**: can be “allowed” or “blocked”. Set to “allowed” unless a rule uses the “drop” action and Suricata is in IPS mode, or when the rule uses the “reject” action
  - **signature**: textual information about the rule and the alert, usually rule source and alert name
  - **signature\_id**: the rule signature identifier (sid)

```
"alert": {  
  "action": "allowed",  
  "gid": 1,  
  "signature_id": 2018216,  
  "rev": 5,  
  "signature": "ET INFO HTTP Connection To DDNS Domain Hopto.org",  
  "category": "Potentially Bad Traffic",  
  "severity": 2,  
  "metadata": {  
    "created_at": [  
      "2014_03_04"  
    ],  
    "updated_at": [  
      "2020_09_15"  
    ]  
  }  
},
```

# Eve.json log fields - Alert Type

- Alert type specific fields:
  - **category**: same as class type in the rule, condenses classification, priority and short/long name for rules
  - **severity**: same as priority in the rule, ranges from 1-255, higher priority (1) rules are checked first
  - **metadata**: optional tag with extra information (like malware family, affected products, creation dates, etc.)

```
"alert": {  
  "action": "allowed",  
  "gid": 1,  
  "signature_id": 2018216,  
  "rev": 5,  
  "signature": "ET INFO HTTP Connection To DDNS Domain Hopto.org",  
  "category": "Potentially Bad Traffic",  
  "severity": 2,  
  "metadata": {  
    "created_at": [  
      "2014_03_04"  
    ],  
    "updated_at": [  
      "2020_09_15"  
    ]  
  }  
},
```

For more: <https://suricata.readthedocs.io/en/latest/rules/meta.html>  
<https://suricata.readthedocs.io/en/latest/rules/meta.html#classtype>

# Suricata's *flow id*

---

- An important and powerful feature of Suricata is its ability to identify and log traffic by a flow:
  - **Flow**: a bidirectional flow of packets with the same 5-tuple elements: protocol, source ip, destination ip, source port, destination port.
- Suricata also allows alerts on fields via dedicated keywords and produces detailed application layer transaction records.
- We can then use Suricata's flow id to relate all logs to an alert and vice versa, including different event types. This gives us a powerful tool when analyzing traffic.

# Understanding and Writing Suricata Rules



# What is a rule?

---

- A rule is an action and pattern mapping.
- Suricata can process thousands of rules against high speed network traffic.
- They can be used to alert or take action on a specific flow of traffic.

```
<Info> -- Loaded 30268 rules.  
<Info> -- Disabled 14 rules.  
<Info> -- Enabled 0 rules.  
<Info> -- Modified 0 rules.  
<Info> -- Dropped 0 rules.  
<Info> -- Enabled 130 rules for flowbit dependencies.
```

# What do rules look like?

---

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ET POLICY
External IP Lookup api.ipify.org"; flow:established,to_server;
urilen:1; http.method; content:"GET"; http.host;
content:"api.ipify.org"; fast_pattern;
reference:md5,79809fd3e05a852581b897cc4b06aa32; classtype:external-
ip-check; sid:2021997; rev:4; metadata:created_at 2015_10_23,
former_category POLICY, updated_at 2020_08_18;)
```

# Rule Headers

---

- Rules have multiple fields that make up the rule, starting with 3 main pieces of core information:
  - `alert http $HOME_NET any -> $EXTERNAL_NET any ...`
- **Action:** drop, alert, pass, reject, rejectsrc, rejectdst, rejectboth
  - This is the action to take on a matched rule.
- **Protocol:** tcp, udp, icmp, ip or application layer specific protocols like http
  - In this example http will be on if your suricata.yaml has the http protocol enabled.
- **Source:** the source IP(s) or network(s) and port(s)
- **Destination:** the destination IP(s) or network(s) and port(s)
- **Direction:** usually a source '>' destination but sometimes both directions '<>'

# Common Rule Fields

---

- **msg:** usually the source and name of the alert. In this example “ET” for Emerging Threats source, “POLICY” for the type of alert, and “External IP Lookup api.ipify.org” as a description.
- **sid:** “signature ID” helps keep rules identifiable
- **rev:** for modifications or improvements on the same SID, rev keeps track of the version.
- **classtype:** with a classification configuration this can assign a short name, long name and priority all in one.

# Common Rule Fields Con't

---

- **priority**: a number 1-255. Rules with a priority of 1 will be checked before rules with a priority of 4.
- **content**: a simple pattern to match. This is followed with a "content modifier" or preceded by a "sticky buffer", telling Suricata where to match the content for this rule.
- **flow**: signifies if flow should be in a certain state
- **reference/metadata**: additional info about this rule. Links, comments and creation dates.

# Common Rule Fields - Example

---

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg:"ET POLICY  
External IP Lookup api.ipify.org"; flow:established,to_server;  
urilen:1; http.method; content:"GET"; http.host;  
content:"api.ipify.org"; fast_pattern;  
reference:md5,79809fd3e05a852581b897cc4b06aa32; classtype:external-  
ip-check; sid:2021997; rev:4; metadata:created_at 2015_10_23,  
former_category POLICY, updated_at 2020_08_18;)
```

# Tuning Out the Noise

# What makes noise in Suricata deployments?

- Noise means too much of the following that makes identifying actionable issues more difficult:
  - Logging
  - Alerts
  - Too wide a net (not limiting network traffic inspection ranges)
  - Rules
  - Configuration



# Suricata Logging - Anomaly Logging

---

- Noise based on the amount of problems with monitored network traffic.
- Establish anomaly logging baseline, then
  - Fix issues
  - Verify that the issues have been addressed
- Disable anomaly logging and periodically review and correct issue(s).

# Anomaly Logging Con't

---

```
# Configure the type of alert (and other) logging you would like.  
outputs:  
  # Extensible Event Format (nicknamed EVE) event log in JSON format  
  - eve-log:  
    enabled: yes  
    filetype: regular #regular|syslog|unix_dgram|unix_stream|redis  
    filename: eve.json
```

```
- anomaly:  
  enabled: yes  
  types:  
    # decode: no  
    # stream: no
```

# Suricata Alerts

---

- Identify low-value alerts (high rate and false positives or irrelevant for the deployment/usage scenario).
- Use suricata-update to disable rules that don't apply -- to not even load those.
- Suppress alerts from hosts and/or networks:  
<https://suricata.readthedocs.io/en/latest/performance/ignoring-traffic.html#suppress>.
- Threshold keywords:  
<https://suricata.readthedocs.io/en/latest/rules/thresholding.html>.

**Note:** You can reduce alert content through filtering.

# Suricata Rules

---

- **Noisy rules:** rules that trigger frequently but provide little value forensically and/or diagnostically.
- **Pass rules:** use to designate network traffic that is not of interest to Suricata:
  - See: <https://suricata.readthedocs.io/en/latest/performance/ignoring-traffic.html#pass-rules>.
- **Thresholds:** limit the amount of alert logging for a rule.
- **Bypass keyword:** exclude traffic from further evaluation:
  - See: <https://suricata.readthedocs.io/en/latest/rules/bypass-keyword.html#bypass-keyword>.

# Suricata Logging - Protocol Logging

---

- Consider disabling protocol logging for certain application layers or all protocols.
- Not all will apply to your deployment environment and/or may be supplemented with an existing Network Security Monitor (NSM).
- Refine: Logging content for some protocols can be reduced (e.g. HTTP, TLS, DNS).

# Protocol Logging Con't

---

- **Hint:**

- use `suricata --list-app-layer-protos` to display the application layer protocols Suricata understands.

```
# Extensible Event Format (nicknamed EVE) event log in JSON format
- eve-log:
  enabled: yes
  filetype: regular #regular|syslog|unix_dgram|unix_stream|redis
  filename: eve.json
  .
  types:
    - http:
      extended: yes      # enable this for extended logging information
    - dns:
      #types: [a, aaaa, cname, mx, ns, ptr, txt]
    - tls:
      extended: yes      # enable this for extended logging information
```

# Suricata Logging

---

- EVE JSON Log
  - JSON format
  - Each entry identified with an event\_type, e.g., alert, flow, stats, that categorizes the content.
- Can disable/enable different log categories; the output section contains the following configuration settings:
  - Protocols
  - Alerts
  - Files
  - Stats
  - Flows

# Suricata Logging Con't

- Application Layer parsing
- Stats logging
- File Extraction

```
outputs:  
- eve-log:  
  types:  
    - alert:  
    - dns:  
      enabled: yes  
    - stats:  
      enabled: yes  
    - flow:  
      enabled: yes  
- file-store:  
  version: 2  
  enabled: no
```

```
app-layer:  
  protocols:  
    dns:  
      tcp:  
        enabled: yes  
      udp:  
        enabled: yes
```

~



# Suricata Configuration

---

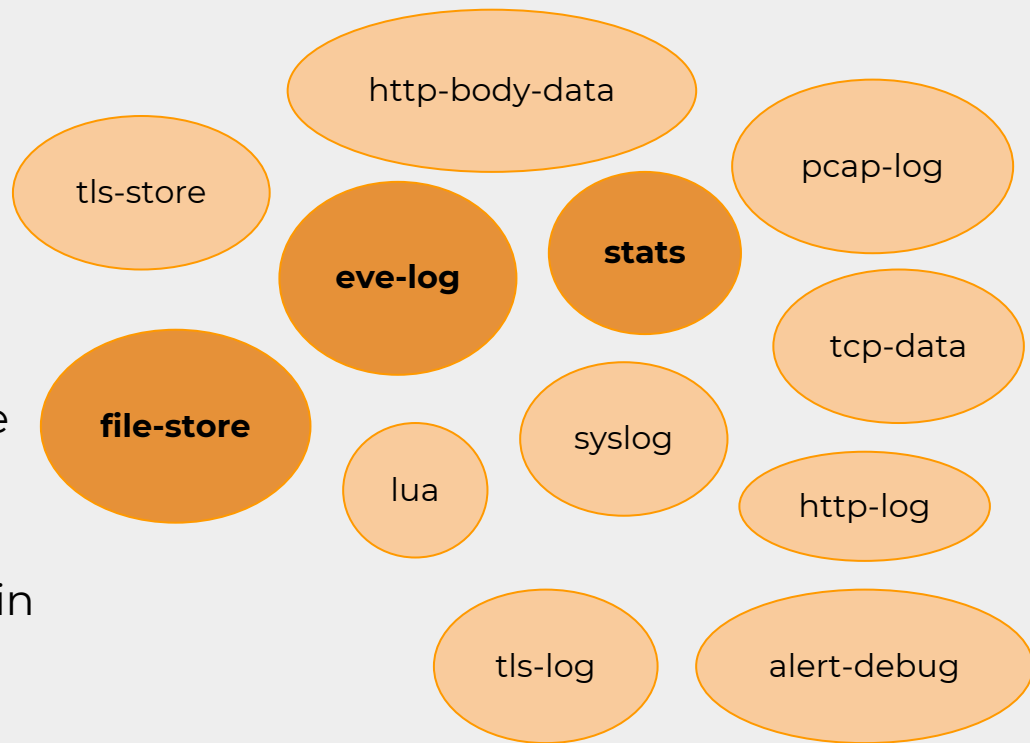
- Global threshold configuration:
  - <https://suricata.readthedocs.io/en/latest/configuration/global-thresholds.html>.
- Host/Network suppression:
  - <https://suricata.readthedocs.io/en/latest/configuration/global-thresholds.html#suppress>.
- Restrict how much of each flow is inspected by establishing a maximum depth:
  - `stream.reassembly.depth` indicates how much of the stream will be reassembled and hence, limit the amount of inspection
  - `file-store.stream-depth` limits how much of a file is stored
  - Protocol specific depth (e.g, `smb.stream-depth`, `modbus.stream-depth`).

# Suricata's Investigative Data Output

# Configuring Data Outputs

---

- There are several output formats made available by Suricata which can all be configured in the **outputs** section of [suricata.yaml](#).
- The output that suits your use case should be enabled.
- Every output has different settings which are described in the configuration file in their appropriate sub-sections.

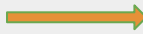


# Flows

---

- Generally (and in this course), a flow refers to a bi-directional flow of packets made up of 5-tuple (**protocol**, **source IP**, **destination IP**, **source port**, **destination port**) or 7-tuple if VLAN flags are counted (5-tuple + **vlan IDs**).
- Two types of flows: flow and netflow

**outputs->eve-log->types->flow**  
enabled by **default**.



```
# bi-directional flows
- flow
# uni-directional flows
#- netflow
```

# Flows Con't

---

- Example of a flow:

```
{
  "timestamp": "2014-02-26T09:19:59.239753+0000",
  "flow_id": 1902325601904777,
  "event_type": "flow",
  "src_ip": "10.30.28.90",
  "src_port": 43246,
  "dest_ip": "10.30.28.94",
  "dest_port": 53,
  "proto": "UDP",
  "app_proto": "dns",
  "flow": {
    "pkts_toserver": 81,
    "pkts_toclient": 80,
    "bytes_toserver": 10078,
    "bytes_toclient": 18389,
    "start": "2014-02-26T09:19:59.239753+0000",
    "end": "2014-02-26T09:20:08.648061+0000",
    "age": 9,
    "state": "established",
    "reason": "shutdown",
    "alerted": false
  }
}
```

# Per Protocol Specific Logs

---

- Loggers specific to a protocol can be enabled. This helps get more metadata particular to that protocol. Some protocols may offer some options other than the default logging.
- Enabled **outputs->eve-log->types** by default:

```
# Extensible Event Format (nicknamed EVE) event log in JSON format
- eve-log:
  enabled: yes
  filetype: regular #regular|syslog|unix_dgram|unix_stream|redis
  filename: eve.json
  types:
    - alert:
    - http:
      extended: yes      # enable this for extended logging information
    - dns:
    - tls:
      extended: yes      # enable this for extended logging information
    - files:
      force-magic: no    # force logging magic on all logged files
```

# Per Protocol Specific Logs Con't

---

- Example protocol: **TLS**
  - Configuration parameters to take care of:
    - Protocol detection must be enabled:
      - `app-layer->protocols->tls->enabled == yes`
    - Any protocol specific settings for particular log fields.
- Examples:
  - JA3 fingerprinting with TLS:
    - `app-layer->protocols->tls->ja3-fingerprints == auto`
  - Store TLS certificates chain to disk:
    - `outputs->eve-log->types->tls-store->enabled == yes`

# Per Protocol Specific Logs Con't

- Example of a TLS event

```
{
  "timestamp": "2016-09-19T15:13:55.657295+0000",
  "flow_id": 1576568689503897,
  "pcap_cnt": 13,
  "event_type": "tls",
  "src_ip": "10.16.1.11",
  "src_port": 54684,
  "dest_ip": "24.244.4.23",
  "dest_port": 443,
  "proto": "TCP",
  "tls": {
    "subject": "C=US, ST=California, L=Mountain View, O=Google Inc, CN=www.google.com",
    "issuerdn": "C=US, O=Google Inc, CN=Google Internet Authority G2",
    "serial": "2B:A2:F5:3B:B5:85:83:75",
    "fingerprint": "90:86:a4:3b:f5:cf:1b:2e:4e:f7:97:96:f9:de:ba:b9:66:35:86:3f",
    "sni": "www.google.com",
    "version": "TLS 1.2",
    "notbefore": "2016-09-14T08:20:40",
    "notafter": "2016-12-07T08:19:00"
  }
}
```



# Per Protocol Specific Logs Con't

- Example: **SMB**

**command:** SMB command for this transaction

**tree\_id:** Represents an open connection to a share, otherwise known as a tree connect. An open tree id MUST be unique within an SMB connection.

```
{
  "timestamp": "2017-04-07T06:17:36.564212+0530",
  "flow_id": 1270583960429429,
  "pcap_cnt": 8426,
  "event_type": "smb",
  "src_ip": "192.168.2.14",
  "src_port": 51839,
  "dest_ip": "192.168.2.101",
  "dest_port": 139,
  "proto": "TCP",
  "metadata": {
    "flowbits": [
      "smb.tree.connect.ipc"
    ],
    "flowints": {
      "applayer.anomaly.count": 1
    }
  },
  "smb": {
    "id": 6,
    "dialect": "NT LM 0.12",
    "command": "SMB1_COMMAND_LOGOFF_ANDX",
    "status": "STATUS_SUCCESS",
    "status_code": "0x0",
    "session_id": 2049,
    "tree_id": 2,
    "request_done": true,
    "response_done": true
  }
}
```

**dialect:** A version of the SMB Protocol that is generally defined in terms of additions and changes relative to a previous version.

**session\_id:** An SMB session ID identifies an authenticated connection to the server.

# Parser Anomalies

- Anomalies are event records created when packets with unexpected or anomalous values are handled.
- Anomalies are also helpful for encrypted traffic hunting.
- Set **outputs->eve-log->anomaly->enabled** to yes.

Types of anomalies that can be logged

```
- anomaly:
# Anomaly log records describe unexpected conditions such
# as truncated packets, packets with invalid IP/UDP/TCP
# length values, and other events that render the packet
# invalid for further processing or describe unexpected
# behavior on an established stream. Networks which
# experience high occurrences of anomalies may experience
# packet processing degradation.
#
# Anomalies are reported for the following:
# 1. Decode: Values and conditions that are detected while
# decoding individual packets. This includes invalid or
# unexpected values for low-level protocol lengths as well
# as stream related events (TCP 3-way handshake issues,
# unexpected sequence number, etc).
# 2. Stream: This includes stream related events (TCP
# 3-way handshake issues, unexpected sequence number,
# etc).
# 3. Application layer: These denote application layer
# specific conditions that are unexpected, invalid or are
# unexpected given the application monitoring state.
#
# By default, anomaly logging is enabled. When anomaly
# logging is enabled, applayer anomaly reporting is
# also enabled.
enabled: yes
#
# Choose one or more types of anomaly logging and whether to enable
# logging of the packet header for packet anomalies.
types:
# decode: no
# stream: no
# applayer: yes
#packethdr: no
```

Reported  
and  
configurable  
types of  
anomalies