## Step 1: Inform Suricata about your network ##

vars:

# more specific is better for alert accuracy and performance address-groups:

HOME\_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"

#HOME\_NET: "[192.168.0.0/16]"

#HOME\_NET: "[10.0.0.0/8]"

#HOME\_NET: "[172.16.0.0/12]"

#HOME\_NET: "any"

EXTERNAL\_NET: "!$HOME\_NET"

#EXTERNAL\_NET: "any"

HTTP\_SERVERS: "$HOME\_NET"

SMTP\_SERVERS: "$HOME\_NET"

SQL\_SERVERS: "$HOME\_NET"

DNS\_SERVERS: "$HOME\_NET"

TELNET\_SERVERS: "$HOME\_NET"

AIM\_SERVERS: "$EXTERNAL\_NET"

DC\_SERVERS: "$HOME\_NET"

DNP3\_SERVER: "$HOME\_NET"

DNP3\_CLIENT: "$HOME\_NET"

MODBUS\_CLIENT: "$HOME\_NET"

MODBUS\_SERVER: "$HOME\_NET"

ENIP\_CLIENT: "$HOME\_NET"

ENIP\_SERVER: "$HOME\_NET"

port-groups: HTTP\_PORTS: "80"

SHELLCODE\_PORTS: "!80"

ORACLE\_PORTS: 1521

SSH\_PORTS: 22

DNP3\_PORTS: 20000

MODBUS\_PORTS: 502

FILE\_DATA\_PORTS: "[$HTTP\_PORTS,110,143]"

FTP\_PORTS: 21

GENEVE\_PORTS: 6081

VXLAN\_PORTS: 4789

TEREDO\_PORTS: 3544

이 파일은 Suricata라는 네트워크 침입 탐지 시스템에서 중요한 설정 파일이다.

‘vars’ 섹션으로, Suricata가 네트워크를 어떻게 인식하고 분석할지에 대한 정보를 설정하는 부분. 구체적으로는 네트워크 주소와 포트를 정의하여 Suricata의 규칙이 이들에 맞춰 적용될 수 있도록 한다.

이 파일을 통해 네트워크 안에서 어떤 IP 대역이 집(home) 네트워크인지, 외부 네트워크는 어디인지, 그리고 서버나 포트에 대한 정보를 설정할 수 있다.

### **1. 네트워크 주소 그룹 (address-groups)**

vars:

# more specific is better for alert accuracy and performance address-groups:

HOME\_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"

#HOME\_NET: "[192.168.0.0/16]"

#HOME\_NET: "[10.0.0.0/8]"

#HOME\_NET: "[172.16.0.0/12]"

#HOME\_NET: "any"

이 부분은 네트워크 주소를 그룹으로 묶어서, Suricata가 어떤 네트워크를 '우리 집'으로 생각해야 하는지, 그리고 외부 네트워크는 어디인지 설정하는 부분이다.

* **HOME\_NET**: 자신의 네트워크를 정의합니다. 이 네트워크를 공격받는 대상이라고 생각하면 된다. 여러 IP 대역을 쉼표로 구분하여 설정할 수 있습니다. 예를 들어, ‘[192.168.0.0/16, 10.0.0.0/8, 172.16.0.0/12]’는 세 개의 서브넷을 포함하는 설정입니다. 이 주소 그룹은 Suricata가 분석할 주요 내부 네트워크를 정의합니다. 주석처리 된 다른 예제들은 특정 서브넷만 포함하거나, ‘any’를 사용하여 모든 IP 주소를 허용 할 수 있습니다.

이렇게 여러 개의 네트워크 범위를 HOME\_NET으로 지정할 수 있다.

* + 예를 들어 192.168.0.0/16은 192.168.0.0부터 192.168.255.255까지의 IP 주소 대역.
  + 여러 네트워크 범위를 HOME\_NET으로 설정해서, Suricata가 이 범위 안에 있는 IP를 '우리 집'으로 인식하도록 할 수 있음.
  + any라고 설정하면, 모든 IP를 집 네트워크로 인식하게 됨. 이건 권장하지 않는다, 정확도가 떨어지기 때문.

EXTERNAL\_NET: "!$HOME\_NET"

#EXTERNAL\_NET: "any"

* **EXTERNAL\_NET**: ‘HOME NET’과 반대되는 외부 네트워크를 정의합니다. 보통 !$HOME\_NET으로 설정해서, 집 네트워크가 아닌 모든 IP를 외부 네트워크로 인식하게 한다. 여기서 !를 "제외하다"는 의미.

HTTP\_SERVERS: "$HOME\_NET"

SMTP\_SERVERS: "$HOME\_NET"

SQL\_SERVERS: "$HOME\_NET"

DNS\_SERVERS: "$HOME\_NET"

TELNET\_SERVERS: "$HOME\_NET"

AIM\_SERVERS: "$EXTERNAL\_NET"

DC\_SERVERS: "$HOME\_NET"

DNP3\_SERVER: "$HOME\_NET"

DNP3\_CLIENT: "$HOME\_NET"

MODBUS\_CLIENT: "$HOME\_NET"

MODBUS\_SERVER: "$HOME\_NET"

ENIP\_CLIENT: "$HOME\_NET"

ENIP\_SERVER: "$HOME\_NET"

특정 서비스나 프로토콜에 대한 서버와 클라이언트를 정의합니다.

* **서버 그룹들**: HTTP\_SERVERS, SMTP\_SERVERS, SQL\_SERVERS 등등은 각각 HTTP, 이메일, 데이터베이스 같은 서버들이 위치한 네트워크를 설정하는 것. 여기서는 보통 집 네트워크에 속해 있으니까 $HOME\_NET으로 설정해놨음.
  + **예시**:
    - HTTP\_SERVERS: "$HOME\_NET": HTTP 서버들이 집 네트워크에 있다고 지정해주는 것.
* **‘DNP3\_SERVER’, “MODBUS\_SERVER’** 등은 각각 DNP3 및 MODBUS 프로토콜을 사용하는 서버의 IP 주소를 ‘HOME\_NET’으로 설정합니다.
* **특정 서버들 (EXTERNAL\_NET)**: AIM, Telnet 같은 경우는 외부 서버로 설정할 수도 있다. 예를 들어 AIM\_SERVERS: "$EXTERNAL\_NET"는 AIM 서버들이 외부 네트워크에 있다고 지정한 것.

### **2. 포트 그룹 (port-groups)**

port-groups: HTTP\_PORTS: "80"

SHELLCODE\_PORTS: "!80"

ORACLE\_PORTS: 1521

SSH\_PORTS: 22

DNP3\_PORTS: 20000

MODBUS\_PORTS: 502

FILE\_DATA\_PORTS: "[$HTTP\_PORTS,110,143]"

FTP\_PORTS: 21

GENEVE\_PORTS: 6081

VXLAN\_PORTS: 4789

TEREDO\_PORTS: 3544

Suricata가 특정 포트를 모니터링할 때 사용할 포트 그룹을 정의합니다.

* **HTTP\_PORTS**: 트래픽을 모니터링할 포트를 설정. 기본적으로 80번 포트는 웹 트래픽(HTTP)에 사용됩니다. 예를 들어, 우리가 웹사이트를 열 때 기본적으로 80번 포트를 통해 접속한다.
* **SHELLCODE\_PORTS**: 쉘코드 공격을 탐지할 포트를 설정. 여기서는 80번 포트를 제외하고 다른 포트들을 지정. !80은 "80번 포트는 제외"라는 의미.
* **특정 서비스 포트들**:
  + **ORACLE\_PORTS**: Oracle 데이터베이스의 기본 포트 1521을 설정.
  + **SSH\_PORTS**: SSH 프로토콜의 기본 포트 22를 설정.
  + **DNP3\_PORT**: DNP3 프로토콜에 사용되는 포트 2000을 설정.
  + **MODBUS\_PORTS**: MODBUS 프로토콜의 기본 포트 502를 설정.
  + **FILE\_DATA\_PORTS**: 파일 데이터를 주고받는 포트들을 묶어서 지정해둔 것. 여기서는 HTTP 포트, 110번 포트(POP3 이메일), 143번 포트(IMAP 이메일)을 묶어놓았음.
  + 파일 전송을 다루는 포트들을 설정. HTTP 포트 80과 추가로 110, 143 을 포함.
  + **FTP\_PORTS**: FTP 프로토콜의 기본 포트 21을 설정.
  + **GENEVE\_PORTS**: GENEVE 프로토콜의 포트 6081을 설정.
  + **VXLAN\_PORTS**: VXLAN 프로토콜의 포트 4789를 설정.
  + **TEREDO\_PORTS**: TEREDO 프로토콜의 포트 3544를 설정.
* 이 외에도 여러 가지 포트를 지정할 수 있음. 포트는 서비스를 구분하는 중요한 정보라서, 올바르게 설정해야 함.

### **요약**

* **네트워크 주소 그룹**: HOME\_NET은 내 네트워크, EXTERNAL\_NET은 외부 네트워크.
* **포트 그룹**: 서비스마다 사용하는 포트를 지정.

이 파일을 수정하면서, Suricata에게 "이건 내 네트워크야!", "이 포트는 이 서비스가 사용해!"라고 알려주는 거지. 그렇게 해서 네트워크를 더 안전하게 지킬 수 있는 거야.

##

## Step 2: Select outputs to enable

##

# The default logging directory. Any log or output file will be

# placed here if it's not specified with a full path name. This can be

# overridden with the -l command line parameter.

default-log-dir: /var/log/suricata/

# Global stats configuration

stats:

enabled: yes

# The interval field (in seconds) controls the interval at

# which stats are updated in the log.

interval: 8

# Add decode events to stats.

#decoder-events: true

# Decoder event prefix in stats. Has been 'decoder' before, but that leads

# to missing events in the eve.stats records. See issue #2225.

#decoder-events-prefix: "decoder.event"

# Add stream events as stats.

#stream-events: false

# Plugins -- Experimental -- specify the filename for each plugin shared object

plugins:

# - /path/to/plugin.so

# Configure the type of alert (and other) logging you would like.

outputs:

# a line based alerts log similar to Snort's fast.log

- fast:

enabled: yes

filename: fast.log

append: yes

#filetype: regular # 'regular', 'unix\_stream' or 'unix\_dgram'

# 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

# Enable for multi-threaded eve.json output; output files are amended with

# an identifier, e.g., eve.9.json

#threaded: false

#prefix: "@cee: " # prefix to prepend to each log entry

# the following are valid when type: syslog above

#identity: "suricata"

#facility: local5

#level: Info ## possible levels: Emergency, Alert, Critical,

## Error, Warning, Notice, Info, Debug

#ethernet: no # log ethernet header in events when available

#redis:

# server: 127.0.0.1

# port: 6379

# async: true ## if redis replies are read asynchronously

# mode: list ## possible values: list|lpush (default), rpush, channel|publish

# ## lpush and rpush are using a Redis list. "list" is an alias for lpush

# ## publish is using a Redis channel. "channel" is an alias for publish

# key: suricata ## key or channel to use (default to suricata)

# Redis pipelining set up. This will enable to only do a query every

# 'batch-size' events. This should lower the latency induced by network

# connection at the cost of some memory. There is no flushing implemented

# so this setting should be reserved to high traffic Suricata deployments.

# pipelining:

# enabled: yes ## set enable to yes to enable query pipelining

# batch-size: 10 ## number of entries to keep in buffer

# Include top level metadata. Default yes.

#metadata: no

# include the name of the input pcap file in pcap file processing mode

pcap-file: false

# Community Flow ID

# Adds a 'community\_id' field to EVE records. These are meant to give

# records a predictable flow ID that can be used to match records to

# output of other tools such as Zeek (Bro).

#

# Takes a 'seed' that needs to be same across sensors and tools

# to make the id less predictable.

# enable/disable the community id feature.

community-id: false

# Seed value for the ID output. Valid values are 0-65535.

community-id-seed: 0

# HTTP X-Forwarded-For support by adding an extra field or overwriting

# the source or destination IP address (depending on flow direction)

# with the one reported in the X-Forwarded-For HTTP header. This is

# helpful when reviewing alerts for traffic that is being reverse

# or forward proxied.

xff:

enabled: no

# Two operation modes are available: "extra-data" and "overwrite".

mode: extra-data

# Two proxy deployments are supported: "reverse" and "forward". In

# a "reverse" deployment the IP address used is the last one, in a

# "forward" deployment the first IP address is used.

deployment: reverse

# Header name where the actual IP address will be reported. If more

# than one IP address is present, the last IP address will be the

# one taken into consideration.

header: X-Forwarded-For

types:

- alert:

# payload: yes # enable dumping payload in Base64

# payload-buffer-size: 4kb # max size of payload buffer to output in eve-log

# payload-printable: yes # enable dumping payload in printable (lossy) format

# packet: yes # enable dumping of packet (without stream segments)

# metadata: no # enable inclusion of app layer metadata with alert. Default yes

# http-body: yes # Requires metadata; enable dumping of HTTP body in Base64

# http-body-printable: yes # Requires metadata; enable dumping of HTTP body in printable format

# Enable the logging of tagged packets for rules using the

# "tag" keyword.

tagged-packets: yes

# Enable logging the final action taken on a packet by the engine

# (e.g: the alert may have action 'allowed' but the verdict be

# 'drop' due to another alert. That's the engine's verdict)

# verdict: yes

# app layer frames

- frame:

# disabled by default as this is very verbose.

enabled: no

- 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

- http:

extended: yes # enable this for extended logging information

# custom allows additional HTTP fields to be included in eve-log.

# the example below adds three additional fields when uncommented

#custom: [Accept-Encoding, Accept-Language, Authorization]

# set this value to one and only one from {both, request, response}

# to dump all HTTP headers for every HTTP request and/or response

# dump-all-headers: none

- dns:

# This configuration uses the new DNS logging format,

# the old configuration is still available:

# https://docs.suricata.io/en/latest/output/eve/eve-json-output.html#dns-v1-format

# As of Suricata 5.0, version 2 of the eve dns output

# format is the default.

#version: 2

# Enable/disable this logger. Default: enabled.

#enabled: yes

# Control logging of requests and responses:

# - requests: enable logging of DNS queries

# - responses: enable logging of DNS answers

# By default both requests and responses are logged.

#requests: no

#responses: no

# Format of answer logging:

# - detailed: array item per answer

# - grouped: answers aggregated by type

# Default: all

#formats: [detailed, grouped]

# DNS record types to log, based on the query type.

# Default: all.

#types: [a, aaaa, cname, mx, ns, ptr, txt]

- tls:

extended: yes # enable this for extended logging information

# output TLS transaction where the session is resumed using a

# session id

#session-resumption: no

# ja4 hashes in tls records will never be logged unless

# the following is set to on. (Default off)

# ja4: off

# custom controls which TLS fields that are included in eve-log

#custom: [subject, issuer, session\_resumed, serial, fingerprint, sni, version, not\_before, not\_after, certificate, chain, ja3, ja3s, ja4]

- files:

force-magic: no # force logging magic on all logged files

# force logging of checksums, available hash functions are md5,

# sha1 and sha256

#force-hash: [md5]

#- drop:

# alerts: yes # log alerts that caused drops

# flows: all # start or all: 'start' logs only a single drop

# # per flow direction. All logs each dropped pkt.

# Enable logging the final action taken on a packet by the engine

# (will show more information in case of a drop caused by 'reject')

# verdict: yes

- smtp:

#extended: yes # enable this for extended logging information

# this includes: bcc, message-id, subject, x\_mailer, user-agent

# custom fields logging from the list:

# reply-to, bcc, message-id, subject, x-mailer, user-agent, received,

# x-originating-ip, in-reply-to, references, importance, priority,

# sensitivity, organization, content-md5, date

#custom: [received, x-mailer, x-originating-ip, relays, reply-to, bcc]

# output md5 of fields: body, subject

# for the body you need to set app-layer.protocols.smtp.mime.body-md5

# to yes

#md5: [body, subject]

#- dnp3

- ftp

- rdp

- nfs

- smb

- tftp

- ike

- dcerpc

- krb5

- bittorrent-dht

- snmp

- rfb

- sip

- quic:

# ja4 hashes in quic records will never be logged unless

# the following is set to on. (Default off)

# ja4: off

- dhcp:

enabled: yes

# When extended mode is on, all DHCP messages are logged

# with full detail. When extended mode is off (the

# default), just enough information to map a MAC address

# to an IP address is logged.

extended: no

- ssh

- mqtt:

# passwords: yes # enable output of passwords

- http2

- pgsql:

enabled: no

# passwords: yes # enable output of passwords. Disabled by default

- stats:

totals: yes # stats for all threads merged together

threads: no # per thread stats

deltas: no # include delta values

# bi-directional flows

- flow

# uni-directional flows

#- netflow

# Metadata event type. Triggered whenever a pktvar is saved

# and will include the pktvars, flowvars, flowbits and

# flowints.

#- metadata

# EXPERIMENTAL per packet output giving TCP state tracking details

# including internal state, flags, etc.

# This output is experimental, meant for debugging and subject to

# change in both config and output without any notice.

#- stream:

# all: false # log all TCP packets

# event-set: false # log packets that have a decoder/stream event

# state-update: false # log packets triggering a TCP state update

# spurious-retransmission: false # log spurious retransmission packets

# a line based log of HTTP requests (no alerts)

- http-log:

enabled: no

filename: http.log

append: yes

#extended: yes # enable this for extended logging information

#custom: yes # enable the custom logging format (defined by customformat)

#customformat: "%{%D-%H:%M:%S}t.%z %{X-Forwarded-For}i %H %m %h %u %s %B %a:%p -> %A:%P"

#filetype: regular # 'regular', 'unix\_stream' or 'unix\_dgram'

# a line based log of TLS handshake parameters (no alerts)

- tls-log:

enabled: no # Log TLS connections.

filename: tls.log # File to store TLS logs.

append: yes

#extended: yes # Log extended information like fingerprint

#custom: yes # enabled the custom logging format (defined by customformat)

#customformat: "%{%D-%H:%M:%S}t.%z %a:%p -> %A:%P %v %n %d %D"

#filetype: regular # 'regular', 'unix\_stream' or 'unix\_dgram'

# output TLS transaction where the session is resumed using a

# session id

#session-resumption: no

# output module to store certificates chain to disk

- tls-store:

enabled: no

#certs-log-dir: certs # directory to store the certificates files

# Packet log... log packets in pcap format. 3 modes of operation: "normal"

# "multi" and "sguil".

#

# In normal mode a pcap file "filename" is created in the default-log-dir,

# or as specified by "dir".

# In multi mode, a file is created per thread. This will perform much

# better, but will create multiple files where 'normal' would create one.

# In multi mode the filename takes a few special variables:

# - %n -- thread number

# - %i -- thread id

# - %t -- timestamp (secs or secs.usecs based on 'ts-format'

# E.g. filename: pcap.%n.%t

#

# Note that it's possible to use directories, but the directories are not

# created by Suricata. E.g. filename: pcaps/%n/log.%s will log into the

# per thread directory.

#

# Also note that the limit and max-files settings are enforced per thread.

# So the size limit when using 8 threads with 1000mb files and 2000 files

# is: 8\*1000\*2000 ~ 16TiB.

#

# In Sguil mode "dir" indicates the base directory. In this base dir the

# pcaps are created in the directory structure Sguil expects:

#

# $sguil-base-dir/YYYY-MM-DD/$filename.<timestamp>

#

# By default all packets are logged except:

# - TCP streams beyond stream.reassembly.depth

# - encrypted streams after the key exchange

#

- pcap-log:

enabled: no

filename: log.pcap

# File size limit. Can be specified in kb, mb, gb. Just a number

# is parsed as bytes.

limit: 1000mb

# If set to a value, ring buffer mode is enabled. Will keep maximum of

# "max-files" of size "limit"

max-files: 2000

# Compression algorithm for pcap files. Possible values: none, lz4.

# Enabling compression is incompatible with the sguil mode. Note also

# that on Windows, enabling compression will \*increase\* disk I/O.

compression: none

# Further options for lz4 compression. The compression level can be set

# to a value between 0 and 16, where higher values result in higher

# compression.

#lz4-checksum: no

#lz4-level: 0

mode: normal # normal, multi or sguil.

# Directory to place pcap files. If not provided the default log

# directory will be used. Required for "sguil" mode.

#dir: /nsm\_data/

#ts-format: usec # sec or usec second format (default) is filename.sec usec is filename.sec.usec

use-stream-depth: no #If set to "yes" packets seen after reaching stream inspection depth are ignored. "no" logs all packets

honor-pass-rules: no # If set to "yes", flows in which a pass rule matched will stop being logged.

# Use "all" to log all packets or use "alerts" to log only alerted packets and flows or "tag"

# to log only flow tagged via the "tag" keyword

#conditional: all

# a full alert log containing much information for signature writers

# or for investigating suspected false positives.

- alert-debug:

enabled: no

filename: alert-debug.log

append: yes

#filetype: regular # 'regular', 'unix\_stream' or 'unix\_dgram'

# Stats.log contains data from various counters of the Suricata engine.

- stats:

enabled: yes

filename: stats.log

append: yes # append to file (yes) or overwrite it (no)

totals: yes # stats for all threads merged together

threads: no # per thread stats

#null-values: yes # print counters that have value 0. Default: no

# a line based alerts log similar to fast.log into syslog

- syslog:

enabled: no

# reported identity to syslog. If omitted the program name (usually

# suricata) will be used.

#identity: "suricata"

facility: local5

#level: Info ## possible levels: Emergency, Alert, Critical,

## Error, Warning, Notice, Info, Debug

# Output module for storing files on disk. Files are stored in

# directory names consisting of the first 2 characters of the

# SHA256 of the file. Each file is given its SHA256 as a filename.

#

# When a duplicate file is found, the timestamps on the existing file

# are updated.

#

# Unlike the older filestore, metadata is not written by default

# as each file should already have a "fileinfo" record in the

# eve-log. If write-fileinfo is set to yes, then each file will have

# one more associated .json files that consist of the fileinfo

# record. A fileinfo file will be written for each occurrence of the

# file seen using a filename suffix to ensure uniqueness.

#

# To prune the filestore directory see the "suricatactl filestore

# prune" command which can delete files over a certain age.

- file-store:

version: 2

enabled: no

# Set the directory for the filestore. Relative pathnames

# are contained within the "default-log-dir".

#dir: filestore

# Write out a fileinfo record for each occurrence of a file.

# Disabled by default as each occurrence is already logged

# as a fileinfo record to the main eve-log.

#write-fileinfo: yes

# Force storing of all files. Default: no.

#force-filestore: yes

# Override the global stream-depth for sessions in which we want

# to perform file extraction. Set to 0 for unlimited; otherwise,

# must be greater than the global stream-depth value to be used.

#stream-depth: 0

# Uncomment the following variable to define how many files can

# remain open for filestore by Suricata. Default value is 0 which

# means files get closed after each write to the file.

#max-open-files: 1000

# Force logging of checksums: available hash functions are md5,

# sha1 and sha256. Note that SHA256 is automatically forced by

# the use of this output module as it uses the SHA256 as the

# file naming scheme.

#force-hash: [sha1, md5]

# NOTE: X-Forwarded configuration is ignored if write-fileinfo is disabled

# HTTP X-Forwarded-For support by adding an extra field or overwriting

# the source or destination IP address (depending on flow direction)

# with the one reported in the X-Forwarded-For HTTP header. This is

# helpful when reviewing alerts for traffic that is being reverse

# or forward proxied.

xff:

enabled: no

# Two operation modes are available, "extra-data" and "overwrite".

mode: extra-data

# Two proxy deployments are supported, "reverse" and "forward". In

# a "reverse" deployment the IP address used is the last one, in a

# "forward" deployment the first IP address is used.

deployment: reverse

# Header name where the actual IP address will be reported. If more

# than one IP address is present, the last IP address will be the

# one taken into consideration.

header: X-Forwarded-For

# Log TCP data after stream normalization

# Two types: file or dir:

# - file logs into a single logfile.

# - dir creates 2 files per TCP session and stores the raw TCP

# data into them.

# Use 'both' to enable both file and dir modes.

#

# Note: limited by "stream.reassembly.depth"

- tcp-data:

enabled: no

type: file

filename: tcp-data.log

# Log HTTP body data after normalization, de-chunking and unzipping.

# Two types: file or dir.

# - file logs into a single logfile.

# - dir creates 2 files per HTTP session and stores the

# normalized data into them.

# Use 'both' to enable both file and dir modes.

#

# Note: limited by the body limit settings

- http-body-data:

enabled: no

type: file

filename: http-data.log

# Lua Output Support - execute lua script to generate alert and event

# output.

# Documented at:

# https://docs.suricata.io/en/latest/output/lua-output.html

- lua:

enabled: no

#scripts-dir: /etc/suricata/lua-output/

scripts:

# - script1.lua

# Logging configuration. This is not about logging IDS alerts/events, but

# output about what Suricata is doing, like startup messages, errors, etc.

logging:

# The default log level: can be overridden in an output section.

# Note that debug level logging will only be emitted if Suricata was

# compiled with the --enable-debug configure option.

#

# This value is overridden by the SC\_LOG\_LEVEL env var.

default-log-level: notice

# The default output format. Optional parameter, should default to

# something reasonable if not provided. Can be overridden in an

# output section. You can leave this out to get the default.

#

# This console log format value can be overridden by the SC\_LOG\_FORMAT env var.

#default-log-format: "%D: %S: %M"

#

# For the pre-7.0 log format use:

#default-log-format: "[%i] %t [%S] - (%f:%l) <%d> (%n) -- "

# A regex to filter output. Can be overridden in an output section.

# Defaults to empty (no filter).

#

# This value is overridden by the SC\_LOG\_OP\_FILTER env var.

default-output-filter:

# Requires libunwind to be available when Suricata is configured and built.

# If a signal unexpectedly terminates Suricata, displays a brief diagnostic

# message with the offending stacktrace if enabled.

#stacktrace-on-signal: on

# Define your logging outputs. If none are defined, or they are all

# disabled you will get the default: console output.

outputs:

- console:

enabled: yes

# type: json

- file:

enabled: yes

level: info

filename: suricata.log

# format: "[%i - %m] %z %d: %S: %M"

# type: json

- syslog:

enabled: no

facility: local5

format: "[%i] <%d> -- "

# type: json

Suricata가 어떤 로그 파일을 만들고, 어떤 형식으로 데이터를 기록할지 결정하는 부분

### **기본 로그 디렉토리**

default-log-dir: /var/log/suricata/

default-log-dir: /var/log/suricata/

* **설명**: 로그 파일이 저장될 기본 디렉토리를 지정하는 부분. 특별한 설정이 없으면 이 경로에 로그 파일들이 저장.
* 디렉토리 변경: 기본적으로 설정된 경로를 변경하려면 이 값을 수정하면 된다. 예를 들어, 로그를 ‘home/user/suricata\_logs/’에 저장하고 싶다면 ‘default-log-dir’을 ‘/home/user/suricata\_logs/’로 설정하면 된다.
* 명령줄 파라미터: ‘default-log-dir’으로 설정된 경로는 Suricata의 실행 시 명령줄의 ‘-l’ 파라미터를 사용하여 다른 경로로 오버라이드할 수 있다. 예를 들어 Suricata를 실행할 때 ‘-l /path/to/other/log/dir’를 추가하면, 지정된 다른 디렉토리에 로그를 저장한다.

이 설정은 로그 파일이 적절한 위치에 저장되고, 파일 시스템의 경로를 일관되게 관리할 수 있도록 도와준다. Suricata의 로그는 네트워크 보안 모니터링 및 분석에 중요한 정보를 담고 있기 때문에, 로그 파일의 위치를 잘 관리하는 것이 중요하다.

### **통계 설정**

# Global stats configuration

stats:

enabled: yes

# The interval field (in seconds) controls the interval at

# which stats are updated in the log.

interval: 8

# Add decode events to stats.

#decoder-events: true

# Decoder event prefix in stats. Has been 'decoder' before, but that leads

# to missing events in the eve.stats records. See issue #2225.

#decoder-events-prefix: "decoder.event"

# Add stream events as stats.

#stream-events: false

stats:

* **설명**: Suricata에서 수집한 통계 데이터를 로그에 기록할지 여부를 설정.

enabled: yes로 설정하면 통계 로그가 활성화.

* **interval**: 통계를 갱신하는 주기를 초 단위로 설정. 여기서는 8초마다 통계가 업데이트돼.
* **decoder-events-prefix:** 디코더 이벤트의 접두사를 설정.
* **decoder-events**: 디코딩 중 발생한 이벤트를 통계에 추가할지 여부를 설정. 여기서는 주석 처리돼 있어 비활성 상태.
* **stream-events**: 스트림 이벤트를 통계에 포함할지 여부를 설정. 여기서는 주석 처리돼 있어 비활성 상태.

### **플러그인 설정**

# Plugins -- Experimental -- specify the filename for each plugin shared object

plugins:

# - /path/to/plugin.so

plugins:

* **설명**: Suricata에서 사용할 플러그인 파일의 경로를 설정. 여기에 특정 플러그인 파일 경로를 추가하면 Suricata에서 해당 플러그인을 사용할 수 있다. 기본적으로 주석 처리되어 있어서 플러그인은 사용되지 않는다. 이 기능은 실험적이며 추가적인 기능을 제공할 수 있다.

### **로그 출력 설정**

outputs:

# Configure the type of alert (and other) logging you would like.

outputs:

# a line based alerts log similar to Snort's fast.log

- fast:

enabled: yes

filename: fast.log

append: yes

#filetype: regular # 'regular', 'unix\_stream' or 'unix\_dgram'

* **설명**: 이 부분에서 Suricata가 생성하는 다양한 종류의 로그 파일과 그 포맷을 설정할 수 있어.

1. **Fast 로그 (fast.log)**
   * fast:: 경량화된 경보 로그 파일로 Suricata의 경고를 라인 기반 로그로 기록, Snort의 fast.log와 유사한 형식. 알림을 빠르게 확인할 수 있는 형식으로 저장.
   * enabled: yes: Fast 로그를 활성화.
   * filename: fast.log: 이 로그 파일의 이름을 지정. 기본적으로 fast.log"로 저장.
   * append: yes: 기존 파일에 로그를 추가.
   * filetype: 파일 형식을 설정할 수 있다. 주석 처리된 부분에서는 ‘regular’,’unix\_stream’,’unix\_dgram’ 중 하나를 선택할 수 있다.
2. **EVE 로그 (eve.json)**

# 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

# Enable for multi-threaded eve.json output; output files are amended with

# an identifier, e.g., eve.9.json

#threaded: false

#prefix: "@cee: " # prefix to prepend to each log entry

# the following are valid when type: syslog above

#identity: "suricata"

#facility: local5

#level: Info ## possible levels: Emergency, Alert, Critical,

## Error, Warning, Notice, Info, Debug

#ethernet: no # log ethernet header in events when available

#redis:

# server: 127.0.0.1

# port: 6379

# async: true ## if redis replies are read asynchronously

# mode: list ## possible values: list|lpush (default), rpush, channel|publish

# ## lpush and rpush are using a Redis list. "list" is an alias for lpush

# ## publish is using a Redis channel. "channel" is an alias for publish

# key: suricata ## key or channel to use (default to suricata)

# Redis pipelining set up. This will enable to only do a query every

# 'batch-size' events. This should lower the latency induced by network

# connection at the cost of some memory. There is no flushing implemented

# so this setting should be reserved to high traffic Suricata deployments.

# pipelining:

# enabled: yes ## set enable to yes to enable query pipelining

# batch-size: 10 ## number of entries to keep in buffer

# Include top level metadata. Default yes.

#metadata: no

# include the name of the input pcap file in pcap file processing mode

pcap-file: false

# Community Flow ID

# Adds a 'community\_id' field to EVE records. These are meant to give

# records a predictable flow ID that can be used to match records to

# output of other tools such as Zeek (Bro).

#

# Takes a 'seed' that needs to be same across sensors and tools

# to make the id less predictable.

# enable/disable the community id feature.

community-id: false

# Seed value for the ID output. Valid values are 0-65535.

community-id-seed: 0

# HTTP X-Forwarded-For support by adding an extra field or overwriting

# the source or destination IP address (depending on flow direction)

# with the one reported in the X-Forwarded-For HTTP header. This is

# helpful when reviewing alerts for traffic that is being reverse

# or forward proxied.

xff:

enabled: no

# Two operation modes are available: "extra-data" and "overwrite".

mode: extra-data

# Two proxy deployments are supported: "reverse" and "forward". In

# a "reverse" deployment the IP address used is the last one, in a

# "forward" deployment the first IP address is used.

deployment: reverse

# Header name where the actual IP address will be reported. If more

# than one IP address is present, the last IP address will be the

# one taken into consideration.

header: X-Forwarded-For

types:

- alert:

# payload: yes # enable dumping payload in Base64

# payload-buffer-size: 4kb # max size of payload buffer to output in eve-log

# payload-printable: yes # enable dumping payload in printable (lossy) format

# packet: yes # enable dumping of packet (without stream segments)

# metadata: no # enable inclusion of app layer metadata with alert. Default yes

# http-body: yes # Requires metadata; enable dumping of HTTP body in Base64

# http-body-printable: yes # Requires metadata; enable dumping of HTTP body in printable format

# Enable the logging of tagged packets for rules using the

# "tag" keyword.

tagged-packets: yes

# Enable logging the final action taken on a packet by the engine

# (e.g: the alert may have action 'allowed' but the verdict be

# 'drop' due to another alert. That's the engine's verdict)

# verdict: yes

# app layer frames

- frame:

# disabled by default as this is very verbose.

enabled: no

- 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

- http:

extended: yes # enable this for extended logging information

# custom allows additional HTTP fields to be included in eve-log.

# the example below adds three additional fields when uncommented

#custom: [Accept-Encoding, Accept-Language, Authorization]

# set this value to one and only one from {both, request, response}

# to dump all HTTP headers for every HTTP request and/or response

# dump-all-headers: none

- dns:

# This configuration uses the new DNS logging format,

# the old configuration is still available:

# https://docs.suricata.io/en/latest/output/eve/eve-json-output.html#dns-v1-format

# As of Suricata 5.0, version 2 of the eve dns output

# format is the default.

#version: 2

# Enable/disable this logger. Default: enabled.

#enabled: yes

# Control logging of requests and responses:

# - requests: enable logging of DNS queries

# - responses: enable logging of DNS answers

# By default both requests and responses are logged.

#requests: no

#responses: no

# Format of answer logging:

# - detailed: array item per answer

# - grouped: answers aggregated by type

# Default: all

#formats: [detailed, grouped]

# DNS record types to log, based on the query type.

# Default: all.

#types: [a, aaaa, cname, mx, ns, ptr, txt]

- tls:

extended: yes # enable this for extended logging information

# output TLS transaction where the session is resumed using a

# session id

#session-resumption: no

# ja4 hashes in tls records will never be logged unless

# the following is set to on. (Default off)

# ja4: off

# custom controls which TLS fields that are included in eve-log

#custom: [subject, issuer, session\_resumed, serial, fingerprint, sni, version, not\_before, not\_after, certificate, chain, ja3, ja3s, ja4]

- files:

force-magic: no # force logging magic on all logged files

# force logging of checksums, available hash functions are md5,

# sha1 and sha256

#force-hash: [md5]

#- drop:

# alerts: yes # log alerts that caused drops

# flows: all # start or all: 'start' logs only a single drop

# # per flow direction. All logs each dropped pkt.

# Enable logging the final action taken on a packet by the engine

# (will show more information in case of a drop caused by 'reject')

# verdict: yes

- smtp:

#extended: yes # enable this for extended logging information

# this includes: bcc, message-id, subject, x\_mailer, user-agent

# custom fields logging from the list:

# reply-to, bcc, message-id, subject, x-mailer, user-agent, received,

# x-originating-ip, in-reply-to, references, importance, priority,

# sensitivity, organization, content-md5, date

#custom: [received, x-mailer, x-originating-ip, relays, reply-to, bcc]

# output md5 of fields: body, subject

# for the body you need to set app-layer.protocols.smtp.mime.body-md5

# to yes

#md5: [body, subject]

* + eve-log:: 이 설정은 JSON 형식으로 다양한 이벤트를 기록하는 걸 담당. 로그 파일의 형식, 파일 이름, 기타 옵션들을 여기서 설정할 수 있다. Extensible Event Format (EVE) 이라고도 한다.
  + enbaled: yes: EVE 로그를 활성화.
  + filetype: 로그를 기록할 형식을 설정.(‘regular’, ’syslog’, ’unix\_dgram’, ’unix\_stream’, ’redis’)
  + filename: eve.json: JSON 로그 파일의 이름을 지정. 기본적으로 "eve.json"으로 저장.
  + threaded: 다중 스레드 모드에서 EVE 로그를 출력할지 여부를 설정.
  + prefix: 각 로그 항목에 추가할 접두사를 설정.
  + identity, facility, level: syslog 관련 설정. syslog를 사용할 경우 유용함.
  + ethernet: 이더넷 헤더를 로그에 포함할지 여부를 설정. 이 옵션을 yes로 설정하면 이더넷 헤더를 로그에 포함.
  + redis: Redis를 사용하는 설정. Redis 서버의 주소와 포트, 비동기 처리 여부, 데이터 전송 모드 등을 설정할 수 있음.
  + pipelining: Redis에 대한 쿼리 파이프라이닝 설정.
  + metadata: EVE 로그에 메타데이터를 포함할지 여부를 설정.
  + pcap-file: pcap 파일을 처리할 때 파일 이름을 포함할지 여부를 설정.
  + **Community ID**: Community ID를 사용하여 로그 기록을 일관되게 할지 여부를 설정. 네트워크 흐름을 추적할 수 있는 고유한 ID를 기록할 수 있는 옵션. 기본적으로 비활성화돼 있음.
  + community-id-seed: Community ID의 시디를 설정.
  + **X-Forwarded-For (XFF)**: HTTP 헤더를 지원하여 IP 주소를 기록하는 방식. 프록시 서버를 통해 전달된 원래의 IP 주소를 기록할 수 있는 설정. 여기도 기본적으로 비활성화돼 있음.

1. **Alert 로그 설정**
   * types: 아래에 다양한 로그 종류를 설정할 수 있음. 여기서는 alert, frame, anomaly, http, dns, tls 등 여러 가지 유형의 로그가 활성화돼 있거나 설정되어 있음.
   * **Alert 로그**: 경고와 관련된 로그 설정으로, HTTP 바디를 Base64 형식으로 기록하거나 패킷 데이터를 기록하는 등의 세부 설정이 가능. 태그가 붙은 패킷을 기록하며, 패킷의 최종 행동을 로그할 수도 있음.
   * **Anomaly 로그**: 비정상적인 네트워크 동작을 기록하는 로그. 비활성화된 디코드, 스트림 등과 같은 특정 유형의 비정상 행동을 선택적으로 기록할 수 있음.
2. **HTTP 로그**
   * http-log:: HTTP 요청 및 응답에 대한 정보를 기록하는 로그. 이 설정에서는 비활성화돼 있음.
3. **TLS 로그**
   * tls-log:: TLS 핸드셰이크와 관련된 정보를 기록하는 로그. TLS 트랜잭션 정보를 기록. 역시 여기서는 비활성화 상태.
4. **PCAP 로그**
   * pcap-log:: 네트워크 패킷을 PCAP 형식으로 기록하는 설정. 기본적으로 이 기능도 꺼져 있음.
5. **기타 로그들**
   * **Syslog**: 시스템 로그에 Suricata의 경고를 기록할 수 있는 설정이지만, 기본적으로 꺼져 있음.
   * **File-store**: 네트워크를 통해 전송된 파일을 디스크에 저장할 수 있는 설정. 여기도 비활성화돼 있음.

이 설정 파일을 통해 Suricata가 어떤 데이터를 기록하고, 어떤 형식으로 기록할지 세세하게 조정할 수 있어. 필요한 로그 종류에 따라 설정을 활성화하거나 비활성화하면 돼. 도움이 되었길 바라! 😊

#- dnp3

- ftp

- rdp

- nfs

- smb

- tftp

- ike

- dcerpc

- krb5

- bittorrent-dht

- snmp

- rfb

- sip

- quic:

# ja4 hashes in quic records will never be logged unless

# the following is set to on. (Default off)

# ja4: off

- dhcp:

enabled: yes

# When extended mode is on, all DHCP messages are logged

# with full detail. When extended mode is off (the

# default), just enough information to map a MAC address

# to an IP address is logged.

extended: no

- ssh

- mqtt:

# passwords: yes # enable output of passwords

- http2

- pgsql:

enabled: no

# passwords: yes # enable output of passwords. Disabled by default

- stats:

totals: yes # stats for all threads merged together

threads: no # per thread stats

deltas: no # include delta values

# bi-directional flows

- flow

# uni-directional flows

#- netflow

# Metadata event type. Triggered whenever a pktvar is saved

# and will include the pktvars, flowvars, flowbits and

# flowints.

#- metadata

# EXPERIMENTAL per packet output giving TCP state tracking details

# including internal state, flags, etc.

# This output is experimental, meant for debugging and subject to

# change in both config and output without any notice.

#- stream:

# all: false # log all TCP packets

# event-set: false # log packets that have a decoder/stream event

# state-update: false # log packets triggering a TCP state update

# spurious-retransmission: false # log spurious retransmission packets

네트워크 보안 시스템에서 사용하는 프로토콜들을 다루는 설정 부분. 보통 네트워크 보안 장비나 침입 탐지 시스템(IDS), 침입 방지 시스템(IPS)에서 사용하는 설정일 가능성이 높음. 각각의 항목들은 다양한 네트워크 프로토콜이나 기능들을 활성화하거나 비활성화하는 역할.

1. **DNP3** (Distributed Network Protocol 3):
   * 주로 전력 시스템에서 사용하는 통신 프로토콜. 스카다(SCADA) 시스템과 같은 산업 제어 시스템에서 많이 사용.
2. **FTP** (File Transfer Protocol):
   * 파일을 서버와 클라이언트 간에 전송하는 데 사용되는 기본적인 파일 전송 프로토콜. 하지만 보안이 취약해서 일반적으로 FTP보다는 SFTP(SSH를 사용하는 FTP)가 더 많이 사용.
3. **RDP** (Remote Desktop Protocol):
   * 원격 데스크탑 연결을 통해 다른 컴퓨터에 접속해서 그 컴퓨터를 제어할 수 있는 프로토콜. 주로 윈도우에서 사용.
4. **NFS** (Network File System):
   * 네트워크를 통해 파일을 공유하는 프로토콜. 주로 유닉스와 리눅스 시스템에서 사용.
5. **SMB** (Server Message Block):
   * 윈도우 시스템에서 네트워크 파일 공유에 사용되는 프로토콜. 윈도우 환경에서 프린터 공유나 파일 공유에 많이 사용.
6. **TFTP** (Trivial File Transfer Protocol):
   * FTP보다 더 단순한 파일 전송 프로토콜. 보안이나 인증이 필요 없는 단순한 파일 전송에 사용. 주로 네트워크 장비의 설정 파일을 전송할 때 많이 사용.
7. **IKE** (Internet Key Exchange):
   * VPN(가상 사설망)에서 사용되는 프로토콜로, 암호화 키 교환을 관리해주는 역할.
8. **DCERPC** (Distributed Computing Environment / Remote Procedure Calls):
   * 윈도우 환경에서 프로세스 간 통신을 위해 사용되는 프로토콜. 네트워크 상의 다른 장치나 프로그램과 통신하기 위해 사용.
9. **KRB5** (Kerberos version 5):
   * 네트워크 인증 프로토콜로, 특히 안전한 인증이 필요한 환경에서 많이 사용. 보안된 네트워크에서 사용자를 인증하기 위해 사용.
10. **BitTorrent-DHT**:
    * P2P(Peer-to-Peer) 네트워크에서 사용하는 프로토콜로, BitTorrent에서 사용. DHT(Distributed Hash Table)는 피어 간의 연결을 관리하고, 파일 조각을 찾는 데 사용.
11. **SNMP** (Simple Network Management Protocol):
    * 네트워크 장치들을 모니터링하고 관리하기 위해 사용되는 프로토콜. 주로 네트워크 관리자들이 장치 상태를 체크하고, 설정을 변경할 때 사용.
12. **RFB** (Remote Framebuffer Protocol):
    * VNC(Virtual Network Computing)에서 사용하는 프로토콜로, 원격 화면을 제어할 때 사용. 원격 데스크탑과 비슷한 역할.
13. **SIP** (Session Initiation Protocol):
    * VoIP(Voice over IP) 통신에서 사용되는 프로토콜로, 음성 및 영상 통화를 설정하고 종료하는 역할.
14. **QUIC** (Quick UDP Internet Connections):
    * 구글이 개발한 새로운 전송 프로토콜로, 기존 TCP보다 더 빠르고 효율적인 통신을 목표로 한다. 여기서 ja4 해시는 QUIC 프로토콜에 관련된 보안 해시인데, 이 기능은 기본적으로 꺼져 있음.
15. **DHCP** (Dynamic Host Configuration Protocol):
    * 네트워크에서 IP 주소를 자동으로 할당해주는 프로토콜. 이 설정에서는 extended 모드가 꺼져 있어서 기본적인 정보(IP와 MAC 주소 매핑)만 로그에 기록되게 되어 있음.
16. **SSH** (Secure Shell):
    * 원격 서버에 안전하게 접속할 때 사용하는 프로토콜. 터미널을 통해 서버 관리 작업을 할 때 주로 사용.
17. **MQTT** (Message Queuing Telemetry Transport):
    * 사물 인터넷(IoT) 환경에서 많이 사용되는 메시지 전송 프로토콜. 여기서는 passwords 설정이 주석 처리되어 있어서, 비밀번호가 로그에 기록되지 않게 돼 있음.
18. **HTTP2** (Hypertext Transfer Protocol/2):
    * 웹 브라우징에서 사용되는 최신 HTTP 프로토콜 버전. 기존 HTTP보다 더 빠르고 효율적인 데이터 전송을 가능하게 함.
19. **PGSQL** (PostgreSQL):
    * 오픈 소스 관계형 데이터베이스 시스템. 여기서는 enabled가 no로 설정되어 있어서, 이 프로토콜은 비활성화 상태.
20. **Stats**:
    * 시스템에서 통계 정보를 로그로 남기는 기능. totals가 yes로 설정되어 있어서 모든 스레드의 통계가 병합된 정보를 기록. 반면에, 각 스레드별 통계(threads)나 델타 값(deltas)은 기록되지 않음.

### **추가 기능들:**

1. **Flow**:
   * 양방향 흐름을 추적하고, 이를 로그로 남기는 기능.
2. **NetFlow**:
   * 단방향 흐름을 추적하는 기능인데, 여기서는 주석 처리되어 있어서 비활성화 상태.
3. **Metadata**:
   * 패킷 변수를 저장할 때 발생하는 메타데이터 이벤트를 로그로 남기는 기능. 주석 처리된 상태로 비활성화되어 있음.
4. **Stream**:
   * TCP 상태 추적 정보를 기록하는 실험적인 기능. 주석 처리되어 있어서 사용되지 않고 있음.

이 파일에서 각각의 프로토콜이나 기능이 어떻게 설정되고, 어떤 역할을 하는지 파악하는 게 중요한데, 네트워크 보안을 관리하는 시스템에서 각 프로토콜의 로그 기록 방법이나 설정을 조정하는 것에 대해 다루고 있다. 필요한 부분을 활성화하거나 세부적으로 조정할 수 있다.

# a line based log of HTTP requests (no alerts)

- http-log:

enabled: no

filename: http.log

append: yes

#extended: yes # enable this for extended logging information

#custom: yes # enable the custom logging format (defined by customformat)

#customformat: "%{%D-%H:%M:%S}t.%z %{X-Forwarded-For}i %H %m %h %u %s %B %a:%p -> %A:%P"

#filetype: regular # 'regular', 'unix\_stream' or 'unix\_dgram'

HTTP 요청을 기록하는 로그 기능에 대한 설정. 네트워크 보안 시스템이나 웹 서버에서 HTTP 요청에 대한 로그를 남길 때 사용될 수 있음. 이 설정은 기본적으로 로그 파일을 어떻게 생성하고, 어떤 정보를 기록할지 정의하는 부분.

1. **http-log**:
   * HTTP 요청을 기록하는 로그 기능을 설정. 이 로그는 보안 경고 없이 단순히 HTTP 요청의 기록을 남기는 데 사용.
2. **enabled: no**:
   * 이 설정은 기본적으로 no로 설정되어 있어 HTTP 로그 기능이 비활성화되어 있다는 뜻. 만약 HTTP 요청 로그를 기록하고 싶다면, 이 값을 yes로 변경해야 함.
3. **filename: http.log**:
   * 로그 파일의 이름을 지정하는 부분. 여기서는 로그 파일이 http.log라는 이름으로 저장되도록 설정되어 있음.
4. **append: yes**:
   * 기존 로그 파일이 있을 경우, 새로운 로그를 추가로 기록할 때 파일을 덮어쓰지 않고, 기존 파일의 끝에 이어서 기록하게 됨. 이 옵션이 yes로 설정되어 있어서 기존 로그에 계속해서 새로운 로그를 덧붙이는 방식으로 작동.
5. **extended: yes** (주석 처리됨):
   * HTTP 요청의 확장 정보를 기록할지 여부를 설정. 이 옵션은 주석 처리되어 있어서 기본적으로 비활성화 상태. 만약 활성화하면, 기본 로그보다 더 많은 정보가 포함된 확장된 로그를 기록.
6. **custom: yes** (주석 처리됨):
   * 사용자 정의 형식으로 로그를 기록할지 여부를 설정.이 설정도 주석 처리되어 있어서 비활성화 상태.
7. **customformat: "%{%D-%H:%M:%S}t.%z %{X-Forwarded-For}i %H %m %h %u %s %B %a:%p -> %A:%P"** (주석 처리됨):
   * 사용자 맞춤형 로그 형식을 정의하는 부분. 여기 정의된 형식은 시간, HTTP 메서드, 요청한 호스트, 상태 코드, 바이트 크기 등과 같은 다양한 정보를 포함. 이 부분이 주석 처리되어 있어서, 현재는 사용되지 않고 있음.
8. **filetype: regular** (주석 처리됨):
   * 로그 파일의 타입을 정의하는 설정. 여기서 regular는 일반적인 파일 형식으로 로그를 저장하는 걸 의미. 다른 옵션으로는 unix\_stream이나 unix\_dgram이 있는데, 이는 Unix 기반 시스템에서 스트림이나 데이터그램 소켓으로 로그를 전송하는 방법을 의미. 이 부분도 주석 처리되어 있어서 비활성화 상태.

### **요약:**

* 기본적으로 이 설정 파일에서는 HTTP 로그 기능이 비활성화(enabled: no)되어 있다.
* 로그 파일은 http.log라는 이름으로 저장되며, 기존 로그에 새로운 로그가 덧붙여져(append: yes) 기록된다.
* 확장된 로그 정보나 맞춤형 로그 형식은 주석 처리되어 있어서 현재는 사용되지 않지만, 필요에 따라 활성화할 수 있다.

이 설정을 통해 HTTP 요청에 대한 상세한 기록을 남길 수 있지만, 보안 경고는 포함되지 않아서 주로 트래픽 분석이나 문제 해결을 위한 용도로 사용될 수 있다.

# a line based log of TLS handshake parameters (no alerts)

- tls-log:

enabled: no # Log TLS connections.

filename: tls.log # File to store TLS logs.

append: yes

#extended: yes # Log extended information like fingerprint

#custom: yes # enabled the custom logging format (defined by customformat)

#customformat: "%{%D-%H:%M:%S}t.%z %a:%p -> %A:%P %v %n %d %D"

#filetype: regular # 'regular', 'unix\_stream' or 'unix\_dgram'

# output TLS transaction where the session is resumed using a

# session id

#session-resumption: no

TLS(Transport Layer Security) 핸드셰이크 과정에 대한 로그를 기록하는 기능. TLS는 네트워크에서 데이터를 암호화하고 안전하게 전송하기 위해 사용하는 프로토콜. TLS 연결의 세부 정보를 로그로 남길 때 어떻게 저장할지 정의.

1. **tls-log**:
   * TLS 핸드셰이크(클라이언트와 서버 간의 보안 연결 설정 과정)에 대한 로그를 관리하는 기능을 담당. 이 로그는 보안 경고 없이, TLS 연결에 대한 정보를 단순히 기록하는 데 사용.
2. **enabled: no**:
   * 기본적으로 no로 설정되어 있어서, TLS 로그 기능이 비활성화되어 있음. 만약 TLS 연결에 대한 로그를 남기고 싶다면, 이 값을 yes로 변경.
3. **filename: tls.log**:
   * 로그 파일의 이름을 지정하는 부분. 여기서는 tls.log라는 이름으로 로그 파일이 생성되도록 설정되어 있음.
4. **append: yes**:
   * 기존 로그 파일이 있을 경우, 새로운 로그를 추가로 기록할 때 파일을 덮어쓰지 않고, 기존 파일의 끝에 이어서 기록하게 됨. 이 설정이 yes로 설정되어 있어서, 로그가 추가 기록되는 방식으로 작동.
5. **extended: yes** (주석 처리됨):
   * TLS 핸드쉐이크의 확장 정보를 기록할지 여부를 설정. 이 옵션이 활성화되면, 기본 로그보다 더 많은 정보(예: TLS 핸드셰이크에서 사용하는 인증서의 지문 정보 등)가 포함된 확장된 로그를 기록. 하지만 주석 처리되어 있어서, 현재는 사용되지 않고 있음.
6. **custom: yes** (주석 처리됨):
   * 맞춤형 로그 형식을 사용하기 위한 옵션. 활성화하면, 사용자가 정의한 형식으로 로그를 기록. 현재는 주석 처리되어 있어서 기본 로그 형식을 사용하고 있음.
7. **customformat: "%{%D-%H:%M:%S}t.%z %a:%p -> %A:%P %v %n %d %D"** (주석 처리됨):
   * 이 부분은 맞춤형 로그 형식을 정의하는 설정. 시간, 클라이언트와 서버의 IP와 포트, 서버 이름, 인증서 정보 등 다양한 정보를 포함할 수 있음. 주석 처리되어 있어서 현재는 사용되지 않고 있음.
8. **filetype: regular** (주석 처리됨):
   * 로그 파일의 타입을 정의하는 설정. regular는 일반적인 파일 형식으로 로그를 저장하는 걸 의미. 다른 옵션으로는 unix\_stream이나 unix\_dgram이 있는데, 이는 Unix 기반 시스템에서 스트림이나 데이터그램 소켓으로 로그를 전송하는 방식. 주석 처리되어 있음.
9. **session-resumption: no** (주석 처리됨):
   * TLS 연결에서는 세션 재개 기능이 있는데, 이는 이전에 설정된 TLS 세션을 재활용해 새로운 핸드셰이크 과정을 줄이는 기능. 이 옵션을 yes로 설정하면, 세션 재개가 이루어진 경우에도 해당 세션 정보를 로그에 기록. 현재는 no로 설정되어 있어서 세션 재개 로그를 기록하지 않음.

### **요약:**

* 이 설정 파일은 TLS 핸드셰이크에 대한 로그를 기록하는 기능을 다루고 있지만, 기본적으로 비활성화(enabled: no) 상태이다.
* 로그 파일 이름은 tls.log로 설정되어 있으며, 기존 로그에 추가 기록(append: yes)하는 방식으로 작동.
* 확장된 정보나 맞춤형 로그 형식은 주석 처리되어 있어서 사용되지 않고 있다.
* TLS 세션 재개 정보는 현재 로그에 기록되지 않지만, 필요하면 session-resumption 설정을 변경해 활성화할 수 있다.

이 설정을 통해 TLS 연결에 대한 다양한 정보를 로그로 남길 수 있는데, 이를 통해 보안 관련 문제를 분석하거나 트래픽을 모니터링할 수 있다.

# output module to store certificates chain to disk

- tls-store:

enabled: no

#certs-log-dir: certs # directory to store the certificates files

# Packet log... log packets in pcap format. 3 modes of operation: "normal"

# "multi" and "sguil".

#

# In normal mode a pcap file "filename" is created in the default-log-dir,

# or as specified by "dir".

# In multi mode, a file is created per thread. This will perform much

# better, but will create multiple files where 'normal' would create one.

# In multi mode the filename takes a few special variables:

# - %n -- thread number

# - %i -- thread id

# - %t -- timestamp (secs or secs.usecs based on 'ts-format'

# E.g. filename: pcap.%n.%t

#

# Note that it's possible to use directories, but the directories are not

# created by Suricata. E.g. filename: pcaps/%n/log.%s will log into the

# per thread directory.

#

# Also note that the limit and max-files settings are enforced per thread.

# So the size limit when using 8 threads with 1000mb files and 2000 files

# is: 8\*1000\*2000 ~ 16TiB.

#

# In Sguil mode "dir" indicates the base directory. In this base dir the

# pcaps are created in the directory structure Sguil expects:

#

# $sguil-base-dir/YYYY-MM-DD/$filename.<timestamp>

#

# By default all packets are logged except:

# - TCP streams beyond stream.reassembly.depth

# - encrypted streams after the key exchange

#

- pcap-log:

enabled: no

filename: log.pcap

# File size limit. Can be specified in kb, mb, gb. Just a number

# is parsed as bytes.

limit: 1000mb

# If set to a value, ring buffer mode is enabled. Will keep maximum of

# "max-files" of size "limit"

max-files: 2000

# Compression algorithm for pcap files. Possible values: none, lz4.

# Enabling compression is incompatible with the sguil mode. Note also

# that on Windows, enabling compression will \*increase\* disk I/O.

compression: none

# Further options for lz4 compression. The compression level can be set

# to a value between 0 and 16, where higher values result in higher

# compression.

#lz4-checksum: no

#lz4-level: 0

mode: normal # normal, multi or sguil.

# Directory to place pcap files. If not provided the default log

# directory will be used. Required for "sguil" mode.

#dir: /nsm\_data/

#ts-format: usec # sec or usec second format (default) is filename.sec usec is filename.sec.usec

use-stream-depth: no #If set to "yes" packets seen after reaching stream inspection depth are ignored. "no" logs all packets

honor-pass-rules: no # If set to "yes", flows in which a pass rule matched will stop being logged.

# Use "all" to log all packets or use "alerts" to log only alerted packets and flows or "tag"

# to log only flow tagged via the "tag" keyword

#conditional: all

# a full alert log containing much information for signature writers

# or for investigating suspected false positives.

- alert-debug:

enabled: no

filename: alert-debug.log

append: yes

#filetype: regular # 'regular', 'unix\_stream' or 'unix\_dgram'

# Stats.log contains data from various counters of the Suricata engine.

- stats:

enabled: yes

filename: stats.log

append: yes # append to file (yes) or overwrite it (no)

totals: yes # stats for all threads merged together

threads: no # per thread stats

#null-values: yes # print counters that have value 0. Default: no

# a line based alerts log similar to fast.log into syslog

- syslog:

enabled: no

# reported identity to syslog. If omitted the program name (usually

# suricata) will be used.

#identity: "suricata"

facility: local5

#level: Info ## possible levels: Emergency, Alert, Critical,

## Error, Warning, Notice, Info, Debug

1. **tls-store**: TLS 인증서를 디스크에 저장하는 기능을 설정.
   * **enabled: no**: TLS(Transport Layer Security) 인증서 체인을 디스크에 저장하는 기능을 다루는 설정. enabled: no로 설정되어 있어 이 기능이 비활성화된 상태.
   * **certs-log-dir: certs** (주석 처리됨): 인증서 파일을 저장할 디렉터리를 설정. 주석 처리되어 있어서 현재는 이 디렉터리가 사용되지 않음.
2. **pcap-log**: 패킷 데이터를 PCAP(패킷 캡처 파일) 형식으로 저장하는 기능. PCAP 파일은 네트워크 트래픽 분석에 사용되는 표준 형식.
   * **enabled: no**: 패킷 로그 기능이 비활성화된 상태. 활성화하려면 yes로 변경.
   * **filename: log.pcap**: 패킷 로그 파일의 기본 이름을 지정하는 부분. 기본적으로 log.pcap라는 이름으로 저장되도록 설정되어 있음.
   * **limit: 1000mb**: 생성되는 PCAP 파일의 최대 크기를 1000MB로 제한하는 설정. 이 크기를 초과하면 새로운 파일이 생성됨. 단위는 kb, mb, gb로 지정 가능.
   * **max-files: 2000**: 최대 2000개의 파일까지만 저장되도록 설정하는 부분. 파일 개수가 이 제한을 넘으면, 가장 오래된 파일부터 삭제됨.
   * **compression: none**: pcap 파일의 압축 알고리즘을 설정. 가능한 값은 ‘none’(압축 없음), ‘lz4’(LZ4 압축).
   * **mode: normal**: 로그 저장 모드를 설정하는 부분. normal 모드에서는 단일 pcap 파일에 모든 패킷을 기록, multi 모드에서는 각 스레드마다 별도의 파일을 생성해 성능을 향상시킬 수 있음. sguil 모드는 Sguil이라는 특정 침입 탐지 시스템에서 사용하는 디렉터리 구조를 생성해.
   * **dir:** Sguil 시스템에 맞는 디렉토리 구조를 사용하여 pcap 파일을 기록
   * **ts-format: usec:** 타임스탬프 형식을 설정. sec(초 단위) 또는 usec(마이크로초 단위) 중 선택할 수 있음.
   * **use-stream-depth: no**: 스트림 검사 깊이를 초과한 패킷을 무시할지 여부를 설정. no로 설정되면, 스트림의 최대 깊이에 도달한 후에도 모든 패킷을 계속 기록.
   * **honor-pass-rules: no**: pass 규칙이 일치하는 흐름을 로그에서 제외할지 여부를 설정. yes로 설정되면, 패스 규칙(특정 트래픽을 무시하는 규칙)이 적용된 흐름에 대한 로그 기록이 중단.
   * **conditional: all** (주석 처리됨): 어떤 패킷을 기록할지 설정하는 부분. all은 모든 패킷을 기록하고, alerts는 경고가 발생한 패킷 및 흐름, tag는 태그가 붙은 흐름.
3. **alert-debug**:
   * 서명 작성자나 오탐지(잘못된 경고)를 조사할 때 사용할 수 있는 상세한 알림 로그를 기록하는 기능.(디버깅용 경고 로그를 설정)
   * **enabled: no**: 이 기능이 비활성화되어 있는 상태. 활성화하려면 yes로 변경.
   * **filename: alert-debug.log**: 디버깅 경고 로그 파일의 이름을 설정.
   * **append: yes**: 로그를 덮어쓰지 않고 기존 파일 끝에 추가로 기록.
   * **filetype:** 파일 형식을 설정(**‘regular’, ‘unix\_stream’, ‘unix\_dgram’**).
4. **stats**: Suricata 엔진의 통계 데이터를 기록하는 로그를 설정.
   * **enabled: yes**: Suricata 엔진의 다양한 카운터 데이터를 기록하는 기능 활성화.
   * **filename: stats.log**: 통계 데이터를 기록할 파일 이름을 지정.
   * **append: yes**: 기존 파일에 추가로 기록하도록 설정.
   * **totals: yes**: 모든 스레드의 통계 데이터를 병합하여 기록.
   * **threads: no**: 각 스레드별로 개별적인 통계를 기록할지 여부를 설정.
   * **null-values: yes** (주석 처리됨): 값이 0인 카운터도 기록할지 여부를 설정하는 부분. 현재는 비활성화 상태.
5. **syslog**:
   * syslog를 사용해 시스템 로그로 알림을 보내는 기능.
   * **enabled: no**: syslog를 통한 로그 전송 기능이 비활성화된 상태.
   * **identity: "suricata"** (주석 처리됨): syslog에서 로그를 전송할 때 사용될 프로그램 이름을 지정하는 부분. 이 부분이 주석 처리되어 있으면 기본적으로 suricata라는 이름을 사용.
   * **facility: local5**: syslog에 보고할 프로그램 이름을 설정. 로그의 분류를 지정. local5는 로컬 로그 서버로 로그를 보내는 데 사용.
   * **level: Info** (주석 처리됨): 로그 수준을 설정. 가능한 값은 ‘Emergency’, ‘Alert’, ‘Critical’, ‘Error’, ‘Warning’, ‘Notice’, ‘Info’, ‘Debug’. syslog에서 로그의 심각도 수준을 설정하는 부분. Info는 정보 수준의 로그만 전송되도록 설정.

### **요약:**

이 설정 파일은 TLS 인증서 체인 저장, 패킷 로그, 알림 디버그, 통계 로그, 그리고 syslog를 통한 로그 전송 기능을 다루고 있음. 기본적으로 대부분의 기능이 비활성화된 상태지만, 필요한 경우 활성화해 네트워크 트래픽이나 보안 경고를 더욱 자세히 분석할 수 있음. 각 기능의 활성화 여부와 설정을 통해 로그 파일의 형식, 저장 위치, 기록 조건 등을 세밀하게 조정할 수 있음.

# Output module for storing files on disk. Files are stored in

# directory names consisting of the first 2 characters of the

# SHA256 of the file. Each file is given its SHA256 as a filename.

#

# When a duplicate file is found, the timestamps on the existing file

# are updated.

#

# Unlike the older filestore, metadata is not written by default

# as each file should already have a "fileinfo" record in the

# eve-log. If write-fileinfo is set to yes, then each file will have

# one more associated .json files that consist of the fileinfo

# record. A fileinfo file will be written for each occurrence of the

# file seen using a filename suffix to ensure uniqueness.

#

# To prune the filestore directory see the "suricatactl filestore

# prune" command which can delete files over a certain age.

- file-store:

version: 2

enabled: no

# Set the directory for the filestore. Relative pathnames

# are contained within the "default-log-dir".

#dir: filestore

# Write out a fileinfo record for each occurrence of a file.

# Disabled by default as each occurrence is already logged

# as a fileinfo record to the main eve-log.

#write-fileinfo: yes

# Force storing of all files. Default: no.

#force-filestore: yes

# Override the global stream-depth for sessions in which we want

# to perform file extraction. Set to 0 for unlimited; otherwise,

# must be greater than the global stream-depth value to be used.

#stream-depth: 0

# Uncomment the following variable to define how many files can

# remain open for filestore by Suricata. Default value is 0 which

# means files get closed after each write to the file.

#max-open-files: 1000

# Force logging of checksums: available hash functions are md5,

# sha1 and sha256. Note that SHA256 is automatically forced by

# the use of this output module as it uses the SHA256 as the

# file naming scheme.

#force-hash: [sha1, md5]

# NOTE: X-Forwarded configuration is ignored if write-fileinfo is disabled

# HTTP X-Forwarded-For support by adding an extra field or overwriting

# the source or destination IP address (depending on flow direction)

# with the one reported in the X-Forwarded-For HTTP header. This is

# helpful when reviewing alerts for traffic that is being reverse

# or forward proxied.

xff:

enabled: no

# Two operation modes are available, "extra-data" and "overwrite".

mode: extra-data

# Two proxy deployments are supported, "reverse" and "forward". In

# a "reverse" deployment the IP address used is the last one, in a

# "forward" deployment the first IP address is used.

deployment: reverse

# Header name where the actual IP address will be reported. If more

# than one IP address is present, the last IP address will be the

# one taken into consideration.

header: X-Forwarded-For

file-store 모듈 설정과 관련된 세부 사항을 설명한 것입니다.

### **file-store 모듈 설정 세부 설명**

1. **모듈 개요:**
   * file-store 모듈은 네트워크 트래픽에서 추출된 파일을 저장하는 기능을 제공합니다. 각 파일은 SHA256 해시 값의 앞 두 글자를 사용하여 생성된 디렉터리에 저장되며, 파일 이름은 전체 SHA256 해시 값으로 지정됩니다.
2. **중복 파일 처리:**
   * 동일한 파일이 여러 번 발견될 경우, 새로운 파일을 생성하지 않고 기존 파일의 타임스탬프를 갱신하여 중복 처리를 수행합니다.
3. **메타데이터 저장:**
   * file-store 모듈은 기본적으로 파일 메타데이터를 저장하지 않습니다. 각 파일에 대한 메타데이터는 이미 eve-log의 "fileinfo" 레코드로 기록되기 때문입니다. write-fileinfo를 yes로 설정하면 각 파일에 대해 .json 확장자를 가진 메타데이터 파일이 생성되며, 해당 파일에는 "fileinfo" 레코드가 포함됩니다. 이 파일은 파일이 탐지될 때마다 고유한 파일명으로 생성됩니다.
4. **디렉터리 설정:**
   * dir: 파일을 저장할 디렉터리를 지정합니다. 상대 경로를 사용하는 경우, 기본 로그 디렉터리(default-log-dir) 내에 저장됩니다. 기본 설정에서는 주석 처리되어 있으며, 사용자가 필요에 따라 경로를 설정해야 합니다.
5. **파일정보 기록:**
   * write-fileinfo: 각 파일에 대해 "fileinfo" 레코드를 생성하고 .json 파일로 저장할지를 결정합니다. 기본값은 no이며, yes로 설정하면 파일의 발생 횟수마다 개별 파일이 생성됩니다.
6. **파일 저장 강제화:**
   * force-filestore: 모든 파일을 강제로 저장할지 여부를 설정합니다. 기본값은 no이며, yes로 설정할 경우 모든 파일이 저장됩니다.
7. **스트림 깊이 제한:**
   * stream-depth: 파일 추출 시 사용할 세션의 스트림 깊이를 설정합니다. 기본값은 0으로, 이 경우 무제한입니다. 특정 값으로 설정하면 전역 스트림 깊이 설정을 무시하고 해당 값 이상에서만 파일을 추출합니다.
8. **최대 파일 열기 수:**
   * max-open-files: file-store 모듈이 동시에 열 수 있는 파일의 최대 개수를 설정합니다. 기본값은 0이며, 이 경우 파일에 쓸 때마다 파일을 닫습니다. 예를 들어, 1000으로 설정하면 최대 1000개의 파일을 동시에 열어둘 수 있습니다.
9. **체크섬 강제화:**
   * force-hash: 파일 저장 시 체크섬을 강제로 기록합니다. 사용 가능한 해시 함수는 md5, sha1, sha256입니다. 기본적으로 file-store 모듈은 파일 이름으로 SHA256을 사용하기 때문에 해당 해시는 자동으로 기록됩니다. 추가적으로 md5와 sha1을 함께 사용하려면 해당 값을 추가할 수 있습니다.
10. **HTTP X-Forwarded-For 설정:**
    * xff 설정은 HTTP 트래픽의 X-Forwarded-For 헤더를 이용하여 실제 클라이언트 IP를 추가로 기록하거나, 기존 IP 주소를 대체합니다.
    * enabled: X-Forwarded-For 기능을 활성화(yes) 또는 비활성화(no)할 수 있습니다.
    * mode: 두 가지 모드가 있습니다.
      + extra-data: X-Forwarded-For 헤더 정보를 추가로 기록합니다.
      + overwrite: 소스 또는 목적지 IP 주소를 X-Forwarded-For 헤더 값으로 대체합니다.
    * deployment: 프록시 배포 유형을 지정합니다.
      + reverse: 마지막 IP 주소를 사용합니다.
      + forward: 첫 번째 IP 주소를 사용합니다.
    * header: 실제 IP 주소가 포함된 HTTP 헤더 이름을 지정합니다. 여러 IP 주소가 있을 경우 마지막 IP 주소가 사용됩니다.

# Log TCP data after stream normalization

# Two types: file or dir:

# - file logs into a single logfile.

# - dir creates 2 files per TCP session and stores the raw TCP

# data into them.

# Use 'both' to enable both file and dir modes.

#

# Note: limited by "stream.reassembly.depth"

- tcp-data:

enabled: no

type: file

filename: tcp-data.log

# Log HTTP body data after normalization, de-chunking and unzipping.

# Two types: file or dir.

# - file logs into a single logfile.

# - dir creates 2 files per HTTP session and stores the

# normalized data into them.

# Use 'both' to enable both file and dir modes.

#

# Note: limited by the body limit settings

- http-body-data:

enabled: no

type: file

filename: http-data.log

# Lua Output Support - execute lua script to generate alert and event

# output.

# Documented at:

# https://docs.suricata.io/en/latest/output/lua-output.html

- lua:

enabled: no

#scripts-dir: /etc/suricata/lua-output/

scripts:

# - script1.lua

tcp-data, http-body-data, lua 모듈 설정에 대한 설명입니다.

### **tcp-data 모듈 설정**

1. **모듈 개요:**
   * tcp-data 모듈은 스트림 재조립 후의 TCP 데이터를 로그로 저장하는 기능을 제공합니다. 이 기능을 통해 TCP 세션의 데이터를 파일로 저장할 수 있습니다.
2. **저장 방식:**
   * type 설정에 따라 로그 저장 방식이 달라집니다.
     + file: 모든 TCP 데이터를 하나의 로그 파일(tcp-data.log)에 저장합니다.
     + dir: 각 TCP 세션에 대해 2개의 파일을 생성하고, 해당 파일에 원시 TCP 데이터를 저장합니다.
   * both로 설정할 경우, file 및 dir 방식 모두 활성화됩니다.
3. **제한 사항:**
   * stream.reassembly.depth 설정에 의해 저장되는 데이터의 양이 제한될 수 있습니다. 이 설정은 스트림 재조립의 깊이를 제한합니다.
4. **현재 설정:**
   * enabled: no로 설정되어 있어 해당 기능이 비활성화되어 있습니다.
   * type: file로 설정되어 있으며, 활성화될 경우 tcp-data.log 파일에 데이터를 저장합니다.

### **http-body-data 모듈 설정**

1. **모듈 개요:**
   * http-body-data 모듈은 HTTP 바디 데이터를 로그로 저장하는 기능을 제공합니다. 이 기능을 통해 HTTP 세션에서 추출된 바디 데이터를 파일로 저장할 수 있습니다. 데이터는 정규화(normalization), 디청크(de-chunking), 압축 해제(unzipping) 과정을 거친 후 저장됩니다.
2. **저장 방식:**
   * type 설정에 따라 로그 저장 방식이 달라집니다.
     + file: 모든 HTTP 바디 데이터를 하나의 로그 파일(http-data.log)에 저장합니다.
     + dir: 각 HTTP 세션에 대해 2개의 파일을 생성하고, 해당 파일에 정규화된 데이터를 저장합니다.
   * both로 설정할 경우, file 및 dir 방식 모두 활성화됩니다.
3. **제한 사항:**
   * 저장되는 데이터는 HTTP 바디의 크기 제한 설정에 의해 제한될 수 있습니다. 이는 바디의 최대 크기를 설정하는 body limit 옵션에 의존합니다.
4. **현재 설정:**
   * enabled: no로 설정되어 있어 해당 기능이 비활성화되어 있습니다.
   * type: file로 설정되어 있으며, 활성화될 경우 http-data.log 파일에 데이터를 저장합니다.

### **lua 모듈 설정**

1. **모듈 개요:**
   * lua 모듈은 Lua 스크립트를 이용하여 경고 및 이벤트 출력을 생성하는 기능을 제공합니다. 사용자 정의 스크립트를 작성하여 Suricata의 이벤트를 처리하거나 새로운 로깅 형식을 정의할 수 있습니다.
2. **설정 항목:**
   * scripts-dir: Lua 스크립트가 위치한 디렉터리를 지정합니다. 기본값은 /etc/suricata/lua-output/입니다.
   * scripts: 실행할 Lua 스크립트 목록을 지정합니다. 스크립트는 배열 형식으로 나열되며, 활성화될 경우 지정된 스크립트들이 순차적으로 실행됩니다.
3. **현재 설정:**
   * enabled: no로 설정되어 있어 해당 기능이 비활성화되어 있습니다.
   * scripts-dir 및 scripts는 주석 처리되어 있으며, 활성화하려면 설정을 해제하고 스크립트 파일들을 명시해야 합니다.

# Logging configuration. This is not about logging IDS alerts/events, but

# output about what Suricata is doing, like startup messages, errors, etc.

logging:

# The default log level: can be overridden in an output section.

# Note that debug level logging will only be emitted if Suricata was

# compiled with the --enable-debug configure option.

#

# This value is overridden by the SC\_LOG\_LEVEL env var.

default-log-level: notice

# The default output format. Optional parameter, should default to

# something reasonable if not provided. Can be overridden in an

# output section. You can leave this out to get the default.

#

# This console log format value can be overridden by the SC\_LOG\_FORMAT env var.

#default-log-format: "%D: %S: %M"

#

# For the pre-7.0 log format use:

#default-log-format: "[%i] %t [%S] - (%f:%l) <%d> (%n) -- "

# A regex to filter output. Can be overridden in an output section.

# Defaults to empty (no filter).

#

# This value is overridden by the SC\_LOG\_OP\_FILTER env var.

default-output-filter:

# Requires libunwind to be available when Suricata is configured and built.

# If a signal unexpectedly terminates Suricata, displays a brief diagnostic

# message with the offending stacktrace if enabled.

#stacktrace-on-signal: on

# Define your logging outputs. If none are defined, or they are all

# disabled you will get the default: console output.

outputs:

- console:

enabled: yes

# type: json

- file:

enabled: yes

level: info

filename: suricata.log

# format: "[%i - %m] %z %d: %S: %M"

# type: json

- syslog:

enabled: no

facility: local5

format: "[%i] <%d> -- "

# type: json

logging 설정에 대한 세부 설명입니다.

### **logging 설정**

logging 설정은 IDS 경고나 이벤트가 아닌 Suricata 자체의 동작 상태, 시작 메시지, 오류 등을 기록하는 로그 설정입니다. Suricata의 일반적인 동작에 대한 정보를 다양한 형식으로 출력하며, 출력 대상과 형식은 여러 설정 옵션을 통해 조정할 수 있습니다.

#### **1. 기본 로그 레벨 (default-log-level):**

* default-log-level: Suricata의 기본 로그 레벨을 설정합니다. 이 값은 output 섹션에서 재정의할 수 있습니다.
  + 가능한 값: emergency, alert, critical, error, warning, notice, info, debug.
  + 현재 설정: notice로, 일반적인 정보와 경고, 오류 등을 기록합니다.
* SC\_LOG\_LEVEL 환경 변수를 통해 이 값을 덮어쓸 수 있습니다.

#### **2. 기본 로그 형식 (default-log-format):**

* default-log-format: 로그 메시지의 기본 형식을 설정합니다. 이 설정은 선택 사항이며, 설정하지 않으면 합리적인 기본값이 사용됩니다.
* output 섹션에서 재정의가 가능하며, SC\_LOG\_FORMAT 환경 변수로 덮어쓸 수 있습니다.
* 기본값: 주석 처리되어 있으며, 설정되지 않은 경우 기본 형식이 적용됩니다.
* 예시 형식:
  + %D: 날짜
  + %S: 심볼(메시지 수준)
  + %M: 메시지 내용

#### **3. 기본 출력 필터 (default-output-filter):**

* default-output-filter: 로그 출력 시 정규 표현식을 이용하여 특정 메시지만 필터링하여 기록합니다. 기본값은 빈 값으로, 필터링 없이 모든 메시지를 기록합니다.
* SC\_LOG\_OP\_FILTER 환경 변수를 통해 이 값을 덮어쓸 수 있습니다.

#### **4. 신호에 의한 스택 추적 (stacktrace-on-signal):**

* stacktrace-on-signal: Suricata가 예상치 못한 신호로 종료될 때, 오류 발생 지점의 스택 추적 정보를 출력하는 옵션입니다.
* 이 옵션은 libunwind 라이브러리가 필요하며, on 또는 off로 설정할 수 있습니다.
* 기본값: 주석 처리되어 있으며, 설정하지 않으면 비활성화됩니다.

#### **5. 로그 출력 대상 (outputs):**

로그를 출력할 대상을 정의합니다. 여러 개의 출력 대상(console, file, syslog)을 설정할 수 있으며, 각 대상에 대해 개별 설정이 가능합니다.

1. **콘솔 출력 (console):**
   * enabled: yes로 설정되어 있으며, 콘솔에 로그를 출력합니다.
   * type: 로그의 형식을 정의합니다. 예를 들어, json 형식으로 출력할 수 있습니다. 현재는 주석 처리되어 있어 기본 형식으로 출력됩니다.
2. **파일 출력 (file):**
   * enabled: yes로 설정되어 있으며, 로그를 파일로 기록합니다.
   * level: info로 설정되어 있으며, info 레벨 이상의 로그를 기록합니다.
   * filename: 로그를 저장할 파일 이름을 설정합니다. 현재 suricata.log로 설정되어 있습니다.
   * format: 로그 메시지의 형식을 정의합니다. 예시로 [%i - %m] %z %d: %S: %M 형식을 사용할 수 있습니다. 현재 주석 처리되어 있어 기본 형식이 사용됩니다.
   * type: 로그 파일의 형식을 정의합니다. 예를 들어, json 형식으로 기록할 수 있습니다. 현재 주석 처리되어 있습니다.
3. **시스템 로그 (syslog):**
   * enabled: no로 설정되어 있으며, 비활성화 상태입니다.
   * facility: 시스템 로그에서 사용할 시설을 지정합니다. 현재 local5로 설정되어 있습니다.
   * format: 로그 메시지의 형식을 정의합니다. 현재 [%i] <%d> -- 형식으로 설정되어 있습니다.
   * type: 로그 형식을 정의할 수 있으며, 예를 들어 json 형식을 사용할 수 있습니다. 현재 주석 처리되어 있습니다.

## Step 3: Configure common capture settings

##

## See "Advanced Capture Options" below for more options, including Netmap

## and PF\_RING.

##

# Linux high speed capture support

af-packet:

- interface: eth0

# Number of receive threads. "auto" uses the number of cores

#threads: auto

# Default clusterid. AF\_PACKET will load balance packets based on flow.

cluster-id: 99

# Default AF\_PACKET cluster type. AF\_PACKET can load balance per flow or per hash.

# This is only supported for Linux kernel > 3.1

# possible value are:

# \* cluster\_flow: all packets of a given flow are sent to the same socket

# \* cluster\_cpu: all packets treated in kernel by a CPU are sent to the same socket

# \* cluster\_qm: all packets linked by network card to a RSS queue are sent to the same

# socket. Requires at least Linux 3.14.

# \* cluster\_ebpf: eBPF file load balancing. See doc/userguide/capture-hardware/ebpf-xdp.rst for

# more info.

# Recommended modes are cluster\_flow on most boxes and cluster\_cpu or cluster\_qm on system

# with capture card using RSS (requires cpu affinity tuning and system IRQ tuning)

# cluster\_rollover has been deprecated; if used, it'll be replaced with cluster\_flow.

cluster-type: cluster\_flow

# In some fragmentation cases, the hash can not be computed. If "defrag" is set

# to yes, the kernel will do the needed defragmentation before sending the packets.

defrag: yes

# To use the ring feature of AF\_PACKET, set 'use-mmap' to yes

use-mmap: yes

# Lock memory map to avoid it being swapped. Be careful that over

# subscribing could lock your system

mmap-locked: no

# Use tpacket\_v3 capture mode, only active if use-mmap is true

# Don't use it in IPS or TAP mode as it causes severe latency

tpacket-v3: yes

# Ring size will be computed with respect to "max-pending-packets" and number

# of threads. You can set manually the ring size in number of packets by setting

# the following value. If you are using flow "cluster-type" and have really network

# intensive single-flow you may want to set the "ring-size" independently of the number

# of threads:

#ring-size: 2048

# Block size is used by tpacket\_v3 only. It should set to a value high enough to contain

# a decent number of packets. Size is in bytes so please consider your MTU. It should be

# a power of 2 and it must be multiple of page size (usually 4096).

#block-size: 32768

# tpacket\_v3 block timeout: an open block is passed to userspace if it is not

# filled after block-timeout milliseconds.

#block-timeout: 10

# On busy systems, set it to yes to help recover from a packet drop

# phase. This will result in some packets (at max a ring flush) not being inspected.

#use-emergency-flush: yes

# recv buffer size, increased value could improve performance

# buffer-size: 32768

# Set to yes to disable promiscuous mode

# disable-promisc: no

# Choose checksum verification mode for the interface. At the moment

# of the capture, some packets may have an invalid checksum due to

# the checksum computation being offloaded to the network card.

# Possible values are:

# - kernel: use indication sent by kernel for each packet (default)

# - yes: checksum validation is forced

# - no: checksum validation is disabled

# - auto: Suricata uses a statistical approach to detect when

# checksum off-loading is used.

# Warning: 'capture.checksum-validation' must be set to yes to have any validation

#checksum-checks: kernel

# BPF filter to apply to this interface. The pcap filter syntax applies here.

#bpf-filter: port 80 or udp

# You can use the following variables to activate AF\_PACKET tap or IPS mode.

# If copy-mode is set to ips or tap, the traffic coming to the current

# interface will be copied to the copy-iface interface. If 'tap' is set, the

# copy is complete. If 'ips' is set, the packet matching a 'drop' action

# will not be copied.

#copy-mode: ips

#copy-iface: eth1

# For eBPF and XDP setup including bypass, filter and load balancing, please

# see doc/userguide/capture-hardware/ebpf-xdp.rst for more info.

# Put default values here. These will be used for an interface that is not

# in the list above.

- interface: default

#threads: auto

#use-mmap: no

#tpacket-v3: yes

af-packet 설정에 대한 설명입니다.

### **af-packet 설정**

af-packet 설정은 Linux에서 고속 네트워크 패킷 캡처를 지원하기 위한 모듈입니다. 이를 통해 Suricata는 네트워크 인터페이스에서 수집된 패킷을 처리할 수 있으며, 여러 고급 설정을 통해 성능을 최적화할 수 있습니다.

#### **1. 인터페이스 (interface):**

* interface: 패킷을 캡처할 네트워크 인터페이스를 지정합니다. 예시에서는 eth0이 지정되어 있습니다.

#### **2. 수신 스레드 수 (threads):**

* threads: 수신 스레드의 수를 지정합니다. auto로 설정하면 시스템의 코어 수에 따라 자동으로 설정됩니다.
* 현재 주석 처리되어 있어 기본값이 사용됩니다.

#### **3. 클러스터 ID (cluster-id):**

* cluster-id: 패킷을 처리할 클러스터의 ID를 지정합니다. 동일한 클러스터 ID를 가진 인터페이스 간에 패킷을 로드 밸런싱합니다.
* 현재 99로 설정되어 있습니다.

#### **4. 클러스터 타입 (cluster-type):**

* cluster-type: 패킷을 클러스터링하는 방식입니다. 설정 가능한 값은 다음과 같습니다.
  + cluster\_flow: 특정 흐름(플로우)에 속한 모든 패킷을 동일한 소켓으로 보냅니다.
  + cluster\_cpu: 커널에서 특정 CPU가 처리하는 모든 패킷을 동일한 소켓으로 보냅니다.
  + cluster\_qm: 네트워크 카드의 RSS 큐와 연계된 모든 패킷을 동일한 소켓으로 보냅니다. (Linux 3.14 이상 필요)
  + cluster\_ebpf: eBPF 파일을 사용한 로드 밸런싱입니다. 자세한 내용은 ebpf-xdp.rst 문서를 참조하십시오.
* 현재 cluster\_flow로 설정되어 있습니다.

#### **5. 프래그먼트 재조립 (defrag):**

* defrag: 일부 프래그먼트된 패킷의 경우 해시 계산이 불가능할 수 있습니다. defrag를 yes로 설정하면 커널이 패킷을 Suricata에 전달하기 전에 재조립합니다.
* 현재 yes로 설정되어 있습니다.

#### **6. 메모리 매핑 사용 (use-mmap):**

* use-mmap: AF\_PACKET의 링 버퍼 기능을 사용하기 위해 메모리 매핑을 활성화합니다.
* 현재 yes로 설정되어 있습니다.

#### **7. 메모리 매핑 잠금 (mmap-locked):**

* mmap-locked: 메모리 매핑을 잠가 스왑되는 것을 방지합니다. 시스템 리소스를 초과할 경우 시스템이 멈출 수 있으므로 주의가 필요합니다.
* 현재 no로 설정되어 있으며, 이는 비활성화 상태를 의미합니다.

#### **8. tpacket\_v3 모드 사용 (tpacket-v3):**

* tpacket-v3: use-mmap이 true일 때만 활성화됩니다. IPS 또는 TAP 모드에서는 심각한 지연을 유발할 수 있어 사용하지 않는 것이 좋습니다.
* 현재 yes로 설정되어 있습니다.

#### **9. 링 크기 (ring-size):**

* ring-size: 링의 크기를 수동으로 설정합니다. 링의 크기는 패킷 수로 설정되며, max-pending-packets와 스레드 수를 기준으로 자동 계산됩니다.
* 대규모 네트워크 트래픽이 있는 경우 설정을 통해 성능을 최적화할 수 있습니다.
* 현재 주석 처리되어 있으며 기본값이 사용됩니다.

#### **10. 블록 크기 (block-size):**

* block-size: tpacket\_v3 모드에서 사용되는 블록의 크기를 바이트 단위로 설정합니다. 일반적으로 MTU 값을 고려하여 설정해야 하며, 값은 2의 거듭제곱이어야 하고 페이지 크기의 배수여야 합니다.
* 현재 주석 처리되어 있으며 기본값이 사용됩니다.

#### **11. 블록 타임아웃 (block-timeout):**

* block-timeout: 블록이 채워지지 않더라도 block-timeout 밀리초 후에 사용자 공간으로 전달됩니다.
* 현재 주석 처리되어 있으며 기본값이 사용됩니다.

#### **12. 긴급 플러시 사용 (use-emergency-flush):**

* use-emergency-flush: 패킷 손실이 발생했을 때 이를 복구하기 위해 링을 비워 패킷을 다시 수신할 수 있도록 합니다. 설정 시 일부 패킷은 검사되지 않을 수 있습니다.
* 현재 주석 처리되어 있으며 기본값이 사용됩니다.

#### **13. 수신 버퍼 크기 (buffer-size):**

* buffer-size: 수신 버퍼 크기를 설정합니다. 값을 증가시키면 성능이 향상될 수 있습니다.
* 현재 주석 처리되어 있으며 기본값이 사용됩니다.

#### **14. 프라미스큐어스 모드 비활성화 (disable-promisc):**

* disable-promisc: yes로 설정하면 프라미스큐어스 모드를 비활성화하여 수신 중인 모든 패킷을 수집하지 않습니다.
* 현재 주석 처리되어 있으며 기본값이 사용됩니다.

#### **15. 체크섬 검사 모드 (checksum-checks):**

* checksum-checks: 인터페이스의 체크섬 검사 모드를 설정합니다. 네트워크 카드에서 체크섬 계산이 오프로드될 때 유용합니다.
  + kernel: 커널이 패킷별로 보낸 체크섬 정보를 사용합니다. (기본값)
  + yes: 체크섬 검사를 강제로 수행합니다.
  + no: 체크섬 검사를 비활성화합니다.
  + auto: Suricata가 통계적 접근을 사용하여 체크섬 오프로드 사용 여부를 감지합니다.
* 현재 kernel로 설정되어 있습니다.

#### **16. BPF 필터 (bpf-filter):**

* bpf-filter: 해당 인터페이스에 적용할 BPF 필터를 설정합니다. pcap 필터 구문을 따릅니다.
* 현재 주석 처리되어 있으며 기본값이 사용됩니다.

#### **17. 복사 모드 및 인터페이스 (copy-mode 및 copy-iface):**

* copy-mode: ips 또는 tap 모드로 설정하여 현재 인터페이스로 들어오는 트래픽을 copy-iface 인터페이스로 복사합니다.
* copy-iface: copy-mode에 따라 트래픽을 복사할 인터페이스를 지정합니다.
* 현재 두 항목 모두 주석 처리되어 기본값이 사용됩니다.

#### **18. 기본 인터페이스 설정:**

* interface: default로 설정되어 있으며, 위에 명시된 인터페이스 외의 모든 인터페이스에 대해 기본값이 적용됩니다.
* 이 섹션에서 threads, use-mmap, tpacket-v3 등의 값을 설정하여 기본 인터페이스의 동작을 정의할 수 있습니다.

# Linux high speed af-xdp capture support

af-xdp:

- interface: default

# Number of receive threads. "auto" uses least between the number

# of cores and RX queues

#threads: auto

#disable-promisc: false

# XDP\_DRV mode can be chosen when the driver supports XDP

# XDP\_SKB mode can be chosen when the driver does not support XDP

# Possible values are:

# - drv: enable XDP\_DRV mode

# - skb: enable XDP\_SKB mode

# - none: disable (kernel in charge of applying mode)

#force-xdp-mode: none

# During socket binding the kernel will attempt zero-copy, if this

# fails it will fallback to copy. If this fails, the bind fails.

# The bind can be explicitly configured using the option below.

# If configured, the bind will fail if not successful (no fallback).

# Possible values are:

# - zero: enable zero-copy mode

# - copy: enable copy mode

# - none: disable (kernel in charge of applying mode)

#force-bind-mode: none

# Memory alignment mode can vary between two modes, aligned and

# unaligned chunk modes. By default, aligned chunk mode is selected.

# select 'yes' to enable unaligned chunk mode.

# Note: unaligned chunk mode uses hugepages, so the required number

# of pages must be available.

#mem-unaligned: no

# The following options configure the prefer-busy-polling socket

# options. The polling time and budget can be edited here.

# Possible values are:

# - yes: enable (default)

# - no: disable

#enable-busy-poll: yes

# busy-poll-time sets the approximate time in microseconds to busy

# poll on a blocking receive when there is no data.

#busy-poll-time: 20

# busy-poll-budget is the budget allowed for packet batches

#busy-poll-budget: 64

# These two tunables are used to configure the Linux OS's NAPI

# context. Their purpose is to defer enabling of interrupts and

# instead schedule the NAPI context from a watchdog timer.

# The softirq NAPI will exit early, allowing busy polling to be

# performed. Successfully setting these tunables alongside busy-polling

# should improve performance.

# Defaults are:

#gro-flush-timeout: 2000000

#napi-defer-hard-irq: 2

af-xdp 설정에 대한 설명입니다. af-xdp는 Linux에서 고속 패킷 캡처를 지원하는 기능 중 하나로, XDP(eXpress Data Path)를 활용하여 성능을 최적화합니다. 이를 통해 Suricata는 네트워크 인터페이스에서 더 빠르고 효율적으로 패킷을 처리할 수 있습니다.

### **af-xdp 설정**

af-xdp 설정은 고속 네트워크 패킷 캡처를 위해 XDP를 사용하는 구성 요소입니다. 이를 통해 패킷의 수집과 처리 성능을 개선할 수 있습니다.

#### **1. 인터페이스 (interface):**

* interface: 패킷을 캡처할 네트워크 인터페이스를 지정합니다. 여기서는 default로 설정되어 있으며, 명시적으로 지정되지 않은 모든 인터페이스에 대해 기본값이 사용됩니다.

#### **2. 수신 스레드 수 (threads):**

* threads: 수신 스레드 수를 지정합니다. auto로 설정하면 코어 수와 RX 큐 중 작은 값이 자동으로 선택됩니다.
* 현재 주석 처리되어 있으며, 기본값이 사용됩니다.

#### **3. 프라미스큐어스 모드 비활성화 (disable-promisc):**

* disable-promisc: true로 설정하면 프라미스큐어스 모드를 비활성화하여 모든 패킷을 수신하지 않도록 합니다. 기본값은 false이며, 프라미스큐어스 모드가 활성화된 상태입니다.
* 현재 주석 처리되어 있으며, 기본값이 사용됩니다.

#### **4. XDP 모드 강제 설정 (force-xdp-mode):**

* force-xdp-mode: XDP 모드를 강제 설정할 수 있습니다. 선택 가능한 값은 다음과 같습니다.
  + drv: XDP\_DRV 모드를 활성화합니다. 드라이버가 XDP를 지원할 때 사용합니다.
  + skb: XDP\_SKB 모드를 활성화합니다. 드라이버가 XDP를 지원하지 않을 때 사용합니다.
  + none: 비활성화하고 커널이 모드를 적용하도록 합니다.
* 현재 none으로 설정되어 있으며, 커널이 모드를 자동으로 적용합니다.

#### **5. 바인딩 모드 강제 설정 (force-bind-mode):**

* force-bind-mode: 소켓 바인딩 중 제로-카피(zero-copy) 모드를 사용합니다. 실패 시 기본적으로 복사(copy) 모드로 전환합니다.
  + zero: 제로-카피 모드를 활성화합니다. 실패 시 바인딩도 실패합니다.
  + copy: 복사 모드를 활성화합니다.
  + none: 비활성화하고 커널이 모드를 적용하도록 합니다.
* 현재 none으로 설정되어 있으며, 커널이 모드를 자동으로 적용합니다.

#### **6. 메모리 정렬 모드 (mem-unaligned):**

* mem-unaligned: 메모리 정렬 모드를 선택합니다. 기본적으로 정렬된 메모리(chunk) 모드가 선택됩니다.
  + yes: 정렬되지 않은 메모리 모드를 활성화합니다. 이 모드는 hugepages를 사용하므로 필요한 페이지 수가 충분해야 합니다.
* 현재 no로 설정되어 있으며, 기본값인 정렬된 메모리 모드가 사용됩니다.

#### **7. 바쁜 대기 (busy-poll) 사용 설정 (enable-busy-poll):**

* enable-busy-poll: 소켓이 바쁜 대기(polling)를 사용할지 설정합니다.
  + yes: 바쁜 대기(polling)를 활성화합니다.
  + no: 비활성화합니다.
* 현재 주석 처리되어 있으며 기본값인 yes가 사용됩니다.

#### **8. 바쁜 대기 시간 (busy-poll-time):**

* busy-poll-time: 수신이 없을 때 블로킹 수신 상태에서 바쁜 대기(polling)할 시간을 마이크로초 단위로 설정합니다.
* 현재 주석 처리되어 있으며 기본값이 사용됩니다.

#### **9. 바쁜 대기 예산 (busy-poll-budget):**

* busy-poll-budget: 패킷 배치(batch)를 위한 바쁜 대기의 예산을 설정합니다.
* 현재 주석 처리되어 있으며 기본값이 사용됩니다.

#### **10. NAPI 컨텍스트 설정 (NAPI context configuration):**

* gro-flush-timeout: NAPI 컨텍스트를 조정하기 위한 설정입니다. 기본값은 2000000으로, NAPI 컨텍스트의 소프트 IRQ가 빨리 종료되어 바쁜 대기를 수행할 수 있도록 합니다.
* napi-defer-hard-irq: NAPI 컨텍스트의 하드 IRQ를 지연시키는 설정입니다. 기본값은 2로 설정되어 있습니다.
* 주석이 해제되어 있지 않으므로 현재 기본값이 사용됩니다.

이 af-xdp 설정을 통해 Suricata는 XDP를 활용하여 고속 네트워크 캡처를 수행할 수 있습니다.

dpdk:

eal-params:

proc-type: primary

# DPDK capture support

# RX queues (and TX queues in IPS mode) are assigned to cores in 1:1 ratio

interfaces:

- interface: 0000:3b:00.0 # PCIe address of the NIC port

# Threading: possible values are either "auto" or number of threads

# - auto takes all cores

# in IPS mode it is required to specify the number of cores and the numbers on both interfaces must match

threads: auto

# interrupt-mode: false # true to switch to interrupt mode

promisc: true # promiscuous mode - capture all packets

multicast: true # enables also detection on multicast packets

checksum-checks: true # if Suricata should validate checksums

checksum-checks-offload: true # if possible offload checksum validation to the NIC (saves Suricata resources)

mtu: 1500 # Set MTU of the device in bytes

# rss-hash-functions: 0x0 # advanced configuration option, use only if you use untested NIC card and experience RSS warnings,

# For `rss-hash-functions` use hexadecimal 0x01ab format to specify RSS hash function flags - DumpRssFlags can help (you can see output if you use -vvv option during Suri startup)

# setting auto to rss\_hf sets the default RSS hash functions (based on IP addresses)

# To approximately calculate required amount of space (in bytes) for interface's mempool: mempool-size \* mtu

# Make sure you have enough allocated hugepages.

# The optimum size for the packet memory pool (in terms of memory usage) is power of two minus one: n = (2^q - 1)

mempool-size: 65535 # The number of elements in the mbuf pool

# Mempool cache size must be lower or equal to:

# - RTE\_MEMPOOL\_CACHE\_MAX\_SIZE (by default 512) and

# - "mempool-size / 1.5"

# It is advised to choose cache\_size to have "mempool-size modulo cache\_size == 0".

# If this is not the case, some elements will always stay in the pool and will never be used.

# The cache can be disabled if the cache\_size argument is set to 0, can be useful to avoid losing objects in cache

# If the value is empty or set to "auto", Suricata will attempt to set cache size of the mempool to a value

# that matches the previously mentioned recommendations

mempool-cache-size: 257

rx-descriptors: 1024

tx-descriptors: 1024

#

# IPS mode for Suricata works in 3 modes - none, tap, ips

# - none: IDS mode only - disables IPS functionality (does not further forward packets)

# - tap: forwards all packets and generates alerts (omits DROP action) This is not DPDK TAP

# - ips: the same as tap mode but it also drops packets that are flagged by rules to be dropped

copy-mode: none

copy-iface: none # or PCIe address of the second interface

- interface: default

threads: auto

promisc: true

multicast: true

checksum-checks: true

checksum-checks-offload: true

mtu: 1500

rss-hash-functions: auto

mempool-size: 65535

mempool-cache-size: 257

rx-descriptors: 1024

tx-descriptors: 1024

copy-mode: none

copy-iface: none

dpdk 설정에 대한 설명입니다. DPDK(Data Plane Development Kit)는 고성능 패킷 처리 성능을 제공하는 라이브러리로, Suricata와 같은 네트워크 보안 도구에서 사용될 때 네트워크 인터페이스 카드(NIC)와 효율적으로 상호작용할 수 있게 합니다.

### **dpdk 설정**

이 설정은 Suricata가 DPDK를 활용하여 네트워크 패킷을 고속으로 캡처하고 처리할 수 있도록 구성합니다. 이를 통해 IDS/IPS 성능을 크게 향상시킬 수 있습니다.

#### **1. EAL(Ethernet Abstraction Layer) 파라미터 (eal-params):**

* proc-type: DPDK 프로세스 유형을 지정합니다.
  + primary: 기본(primary) 프로세스로 설정하여, 네트워크 인터페이스와 직접 상호작용하도록 설정합니다.

#### **2. 인터페이스 설정 (interfaces):**

* 여러 네트워크 인터페이스에 대한 DPDK 설정을 정의합니다. 각 인터페이스는 PCIe 주소를 기반으로 지정됩니다.

##### **a. 인터페이스 1 (interface: 0000:3b:00.0):**

* interface: 캡처할 네트워크 인터페이스의 PCIe 주소를 지정합니다. 여기서는 0000:3b:00.0입니다.
* threads: 수신 스레드 수를 지정합니다. auto로 설정하면 사용할 수 있는 모든 코어를 사용합니다.
  + IPS 모드에서는 양쪽 인터페이스에 대해 동일한 스레드 수를 설정해야 합니다.
* promisc: 프라미스큐어스 모드를 설정합니다. true로 설정하면 모든 패킷을 캡처합니다.
* multicast: 멀티캐스트 패킷을 감지합니다. true로 설정하여 활성화합니다.
* checksum-checks: Suricata가 체크섬을 검증할지 여부를 설정합니다. true로 설정하여 체크섬 검증을 활성화합니다.
* checksum-checks-offload: 가능할 경우 NIC가 체크섬 검증을 수행하도록 설정합니다. Suricata의 리소스를 절약할 수 있습니다.
* mtu: 장치의 MTU(Maximum Transmission Unit)를 바이트 단위로 설정합니다. 여기서는 1500바이트로 설정되어 있습니다.
* mempool-size: mbuf(mempool buffer) 풀의 요소 수를 설정합니다. 이는 메모리 풀의 크기를 결정하며, 인터페이스의 패킷 처리 능력에 영향을 미칩니다.
  + 여기서는 65535로 설정되어 있으며, 풀의 크기는 메모리 사용량을 기준으로 설정됩니다.
* mempool-cache-size: mempool 캐시의 크기를 설정합니다. 캐시 크기는 mempool-size와의 관계를 고려하여 선택해야 하며, 일반적으로 mempool-size % cache\_size == 0이 되도록 설정합니다.
  + 기본값은 257로 설정되어 있습니다.
* rx-descriptors: 수신(RX) 디스크립터 수를 설정합니다. 기본값은 1024입니다.
* tx-descriptors: 송신(TX) 디스크립터 수를 설정합니다. 기본값은 1024입니다.
* copy-mode: Suricata의 IPS 모드를 설정합니다.
  + none: IDS 모드로 설정되어 IPS 기능이 비활성화됩니다.
  + tap: 모든 패킷을 전달하고 경고를 생성합니다. DROP 동작은 무시됩니다.
  + ips: tap 모드와 동일하지만, DROP 동작에 따라 패킷을 드롭합니다.
* 여기서는 none으로 설정되어 IDS 모드로 동작합니다.
* copy-iface: IPS 모드에서 패킷을 복사할 인터페이스의 PCIe 주소를 지정합니다. none으로 설정되어 있으며, 복사 대상이 없습니다.

##### **b. 기본 인터페이스 (interface: default):**

* interface: 기본 인터페이스에 대한 설정입니다. 명시적으로 설정되지 않은 모든 인터페이스에 대해 사용됩니다.
* threads, promisc, multicast, checksum-checks, checksum-checks-offload, mtu, mempool-size, mempool-cache-size, rx-descriptors, tx-descriptors, copy-mode, copy-iface 설정이 모두 인터페이스 1과 동일하게 적용됩니다.
* rss-hash-functions: 기본값은 auto로 설정되어 있으며, 네트워크 인터페이스 카드에서 사용 가능한 RSS(Redistributed Storage Service) 해시 함수를 자동으로 설정합니다.

이 dpdk 설정을 통해 Suricata는 DPDK 라이브러리를 사용하여 고성능 패킷 처리를 수행할 수 있습니다. 네트워크 인터페이스에 따라 성능을 최적화하기 위해 설정을 조정할 수 있으며, 필요한 경우 특정 NIC의 PCIe 주소를 명시적으로 지정하여 사용할 수 있습니다.

# Cross platform libpcap capture support

pcap:

- interface: eth0

# On Linux, pcap will try to use mmap'ed capture and will use "buffer-size"

# as total memory used by the ring. So set this to something bigger

# than 1% of your bandwidth.

#buffer-size: 16777216

#bpf-filter: "tcp and port 25"

# Choose checksum verification mode for the interface. At the moment

# of the capture, some packets may have an invalid checksum due to

# the checksum computation being offloaded to the network card.

# Possible values are:

# - yes: checksum validation is forced

# - no: checksum validation is disabled

# - auto: Suricata uses a statistical approach to detect when

# checksum off-loading is used. (default)

# Warning: 'capture.checksum-validation' must be set to yes to have any validation

#checksum-checks: auto

# With some accelerator cards using a modified libpcap (like Myricom), you

# may want to have the same number of capture threads as the number of capture

# rings. In this case, set up the threads variable to N to start N threads

# listening on the same interface.

#threads: 16

# set to no to disable promiscuous mode:

#promisc: no

# set snaplen, if not set it defaults to MTU if MTU can be known

# via ioctl call and to full capture if not.

#snaplen: 1518

# Put default values here

- interface: default

#checksum-checks: auto

# Settings for reading pcap files

pcap-file:

# Possible values are:

# - yes: checksum validation is forced

# - no: checksum validation is disabled

# - auto: Suricata uses a statistical approach to detect when

# checksum off-loading is used. (default)

# Warning: 'checksum-validation' must be set to yes to have checksum tested

checksum-checks: auto

# See "Advanced Capture Options" below for more options, including Netmap

# and PF\_RING.

pcap 및 pcap-file 설정에 대한 설명입니다. libpcap은 네트워크 패킷 캡처를 위한 라이브러리로, Suricata가 다양한 플랫폼에서 패킷을 캡처할 수 있도록 해줍니다. 여기에서는 libpcap을 사용하여 네트워크 인터페이스와 pcap 파일로부터 패킷을 읽는 방법을 설정할 수 있습니다.

### **pcap 설정**

이 섹션은 Suricata가 네트워크 인터페이스를 통해 libpcap을 사용하여 패킷을 캡처할 때의 설정을 정의합니다.

#### **1. 인터페이스별 설정 (pcap):**

* 여러 네트워크 인터페이스에 대한 pcap 설정을 정의합니다.

##### **a. 인터페이스 1 (interface: eth0):**

* interface: 캡처할 네트워크 인터페이스의 이름을 지정합니다. 여기서는 eth0입니다.
* buffer-size: 캡처 버퍼의 크기를 바이트 단위로 지정합니다.
  + mmap 기반의 캡처를 사용할 때 버퍼의 전체 크기로 사용되며, 네트워크 대역폭의 1% 이상으로 설정하는 것이 좋습니다.
  + 예를 들어, 16MB(16777216 bytes)로 설정 가능합니다.
* bpf-filter: BPF(Berkeley Packet Filter) 필터를 설정하여 캡처할 패킷을 필터링할 수 있습니다.
  + 예를 들어, "tcp and port 25"로 설정하면 TCP 프로토콜의 포트 25번 패킷만 캡처합니다.
* checksum-checks: 체크섬 검증 모드를 설정합니다.
  + yes: 체크섬 검증을 강제로 수행합니다.
  + no: 체크섬 검증을 비활성화합니다.
  + auto: Suricata가 체크섬 오프로딩이 사용되는지 여부를 통계적으로 감지하여 자동으로 설정합니다.
  + 주의: capture.checksum-validation이 yes로 설정되어 있어야 체크섬 검증이 활성화됩니다.
* threads: 인터페이스에서 패킷을 캡처하는 스레드 수를 지정합니다.
  + 예를 들어, threads: 16으로 설정하면 16개의 스레드가 인터페이스에서 패킷을 캡처합니다.
* promisc: 프라미스큐어스 모드를 설정합니다.
  + no로 설정하면 프라미스큐어스 모드를 비활성화하고, yes로 설정하면 모든 패킷을 캡처합니다.
* snaplen: 패킷 캡처 시의 스냅샷 길이를 지정합니다.
  + 설정하지 않으면 기본적으로 MTU에 맞춰지며, MTU 정보를 알 수 없는 경우에는 전체 패킷을 캡처합니다.
  + 예를 들어, snaplen: 1518으로 설정하면 최대 1518 바이트까지 캡처합니다.

##### **b. 기본 인터페이스 (interface: default):**

* interface: 기본 인터페이스에 대한 설정입니다. 명시적으로 설정되지 않은 모든 인터페이스에 대해 사용됩니다.
* checksum-checks: 체크섬 검증 모드를 auto로 설정하여 Suricata가 체크섬 검증을 자동으로 처리하도록 합니다.

### **pcap-file 설정**

이 섹션은 Suricata가 파일로부터 libpcap을 사용하여 패킷을 읽을 때의 설정을 정의합니다.

* checksum-checks: 체크섬 검증 모드를 설정합니다.
  + yes: 체크섬 검증을 강제로 수행합니다.
  + no: 체크섬 검증을 비활성화합니다.
  + auto: Suricata가 체크섬 오프로딩이 사용되는지 여부를 통계적으로 감지하여 자동으로 설정합니다.
  + 주의: checksum-validation이 yes로 설정되어 있어야 체크섬 검증이 활성화됩니다.

### **요약**

* pcap 섹션에서는 libpcap을 사용하여 네트워크 인터페이스로부터 패킷을 실시간으로 캡처할 때의 설정을 정의합니다. 각 인터페이스에 대해 개별적으로 버퍼 크기, 스레드 수, 필터, 체크섬 검증, 프라미스큐어스 모드 등을 설정할 수 있습니다.
* pcap-file 섹션에서는 Suricata가 pcap 파일을 읽을 때의 설정을 정의하며, 체크섬 검증 모드를 설정할 수 있습니다.

이러한 설정을 통해 Suricata가 다양한 플랫폼에서 네트워크 패킷을 효율적으로 캡처하고 분석할 수 있게 됩니다.

##

## Step 4: App Layer Protocol configuration

##

# Configure the app-layer parsers.

#

# The error-policy setting applies to all app-layer parsers. Values can be

# "drop-flow", "pass-flow", "bypass", "drop-packet", "pass-packet", "reject" or

# "ignore" (the default).

#

# The protocol's section details each protocol.

#

# The option "enabled" takes 3 values - "yes", "no", "detection-only".

# "yes" enables both detection and the parser, "no" disables both, and

# "detection-only" enables protocol detection only (parser disabled).

app-layer:

# error-policy: ignore

protocols:

telnet:

enabled: yes

rfb:

enabled: yes

detection-ports:

dp: 5900, 5901, 5902, 5903, 5904, 5905, 5906, 5907, 5908, 5909

mqtt:

enabled: yes

# max-msg-length: 1mb

# subscribe-topic-match-limit: 100

# unsubscribe-topic-match-limit: 100

# Maximum number of live MQTT transactions per flow

# max-tx: 4096

krb5:

enabled: yes

bittorrent-dht:

enabled: yes

snmp:

enabled: yes

ike:

enabled: yes

tls:

enabled: yes

detection-ports:

dp: 443

# Generate JA3/JA4 fingerprints from client hello. If not specified it

# will be disabled by default, but enabled if rules require it.

#ja3-fingerprints: auto

#ja4-fingerprints: auto

# What to do when the encrypted communications start:

# - default: keep tracking TLS session, check for protocol anomalies,

# inspect tls\_\* keywords. Disables inspection of unmodified

# 'content' signatures.

# - bypass: stop processing this flow as much as possible. No further

# TLS parsing and inspection. Offload flow bypass to kernel

# or hardware if possible.

# - full: keep tracking and inspection as normal. Unmodified content

# keyword signatures are inspected as well.

#

# For best performance, select 'bypass'.

#

#encryption-handling: default

pgsql:

enabled: no

# Stream reassembly size for PostgreSQL. By default, track it completely.

stream-depth: 0

# Maximum number of live PostgreSQL transactions per flow

# max-tx: 1024

dcerpc:

enabled: yes

# Maximum number of live DCERPC transactions per flow

# max-tx: 1024

ftp:

enabled: yes

# memcap: 64mb

rdp:

#enabled: yes

ssh:

enabled: yes

#hassh: yes

http2:

enabled: yes

# Maximum number of live HTTP2 streams in a flow

#max-streams: 4096

# Maximum headers table size

#max-table-size: 65536

# Maximum reassembly size for header + continuation frames

#max-reassembly-size: 102400

smtp:

enabled: yes

raw-extraction: no

# Maximum number of live SMTP transactions per flow

# max-tx: 256

# Configure SMTP-MIME Decoder

mime:

# Decode MIME messages from SMTP transactions

# (may be resource intensive)

# This field supersedes all others because it turns the entire

# process on or off

decode-mime: yes

# Decode MIME entity bodies (ie. Base64, quoted-printable, etc.)

decode-base64: yes

decode-quoted-printable: yes

# Maximum bytes per header data value stored in the data structure

# (default is 2000)

header-value-depth: 2000

# Extract URLs and save in state data structure

extract-urls: yes

Suricata의 애플리케이션 계층 프로토콜 설정 섹션(app-layer)은 네트워크 트래픽에서 다양한 애플리케이션 프로토콜을 분석하고 감지하는 기능을 제공합니다. 이 섹션에서는 각 프로토콜의 활성화 상태 및 특정 설정을 구성할 수 있습니다. 여기에는 모든 애플리케이션 프로토콜에 적용되는 error-policy 옵션과, 각각의 프로토콜에 대해 감지, 분석, 검사 등의 설정이 포함됩니다.

### **기본 설정**

#### **error-policy (기본값: ignore):**

* 애플리케이션 계층 프로토콜에서 오류가 발생했을 때 어떻게 처리할지 결정합니다.
* 가능한 값:
  + drop-flow: 오류가 발생한 흐름을 삭제.
  + pass-flow: 오류가 발생한 흐름을 통과시킴.
  + bypass: 오류가 발생한 흐름을 우회.
  + drop-packet: 오류가 발생한 패킷을 삭제.
  + pass-packet: 오류가 발생한 패킷을 통과시킴.
  + reject: 오류가 발생한 연결을 거부.
  + ignore: 오류를 무시하고 계속 처리 (기본값).

### **프로토콜별 설정 (protocols)**

#### **telnet:**

* enabled: yes로 설정되어 있어, Telnet 프로토콜 감지 및 분석이 활성화됨.

#### **rfb:**

* enabled: yes: RFB (Remote Frame Buffer) 프로토콜 감지 및 분석이 활성화됨.
* detection-ports: RFB 감지를 위해 사용할 포트 목록을 지정합니다. 여기서는 5900부터 5909까지 지정되어 있어, VNC와 관련된 모든 포트에서 RFB 트래픽을 감지합니다.

#### **mqtt:**

* enabled: yes: MQTT 프로토콜 감지 및 분석이 활성화됨.
* max-msg-length: MQTT 메시지의 최대 길이를 설정할 수 있습니다. (예: 1mb)
* subscribe-topic-match-limit: 구독 주제의 최대 일치 수를 설정할 수 있습니다. (기본값: 100)
* unsubscribe-topic-match-limit: 구독 취소 주제의 최대 일치 수를 설정할 수 있습니다. (기본값: 100)
* max-tx: 각 흐름에 대해 유지할 수 있는 최대 MQTT 트랜잭션 수를 설정합니다. (기본값: 4096)

#### **krb5:**

* enabled: yes: Kerberos V5 프로토콜 감지 및 분석이 활성화됨.

#### **bittorrent-dht:**

* enabled: yes: BitTorrent DHT(Distributed Hash Table) 프로토콜 감지 및 분석이 활성화됨.

#### **snmp:**

* enabled: yes: SNMP 프로토콜 감지 및 분석이 활성화됨.

#### **ike:**

* enabled: yes: IKE (Internet Key Exchange) 프로토콜 감지 및 분석이 활성화됨.

#### **tls:**

* enabled: yes: TLS(Transport Layer Security) 프로토콜 감지 및 분석이 활성화됨.
* detection-ports: TLS 감지를 위한 포트 목록. 여기서는 443이 기본적으로 설정되어 있습니다.
* ja3-fingerprints 및 ja4-fingerprints: JA3/JA4 핑거프린트 생성을 자동으로 설정할 수 있습니다.
* encryption-handling: 암호화된 통신이 시작될 때의 처리 방식을 설정합니다.
  + default: TLS 세션을 추적하고 프로토콜 이상 징후를 검사하며 tls\_\* 키워드를 사용한 검사를 수행함.
  + bypass: 가능한 한 이 흐름의 처리를 중단하고 TLS 파싱 및 검사를 중단함.
  + full: 일반적으로 추적하고 검사하며, 수정되지 않은 content 키워드를 포함한 시그니처도 검사함.

#### **pgsql:**

* enabled: no: PostgreSQL 프로토콜 감지 및 분석이 비활성화됨.
* stream-depth: PostgreSQL 스트림 재조립 크기. 기본값은 전체 추적입니다.
* max-tx: 각 흐름에 대해 유지할 수 있는 최대 PostgreSQL 트랜잭션 수를 설정합니다. (기본값: 1024)

#### **dcerpc:**

* enabled: yes: DCERPC 프로토콜 감지 및 분석이 활성화됨.
* max-tx: 각 흐름에 대해 유지할 수 있는 최대 DCERPC 트랜잭션 수를 설정합니다. (기본값: 1024)

#### **ftp:**

* enabled: yes: FTP 프로토콜 감지 및 분석이 활성화됨.
* memcap: FTP 분석에 사용할 최대 메모리 크기를 설정할 수 있습니다. (예: 64mb)

#### **rdp:**

* enabled: 기본적으로 주석 처리되어 있어 비활성화 상태입니다. 주석을 제거하고 enabled: yes로 설정하면 RDP 프로토콜 감지 및 분석이 활성화됩니다.

#### **ssh:**

* enabled: yes: SSH 프로토콜 감지 및 분석이 활성화됨.
* hassh: yes로 설정하면 hassh 기능이 활성화되어, SSH 핑거프린트 감지를 수행합니다.

#### **http2:**

* enabled: yes: HTTP/2 프로토콜 감지 및 분석이 활성화됨.
* max-streams: 각 흐름에 대해 유지할 수 있는 최대 HTTP/2 스트림 수를 설정할 수 있습니다. (기본값: 4096)
* max-table-size: 헤더 테이블의 최대 크기를 설정합니다. (기본값: 65536)
* max-reassembly-size: 헤더 및 연속 프레임의 최대 재조립 크기를 설정합니다. (기본값: 102400)

#### **smtp:**

* enabled: yes: SMTP 프로토콜 감지 및 분석이 활성화됨.
* raw-extraction: 기본값 no로 설정되어 있으며, yes로 설정하면 MIME 디코딩이 이루어지지 않은 상태의 데이터를 추출합니다.
* max-tx: 각 흐름에 대해 유지할 수 있는 최대 SMTP 트랜잭션 수를 설정할 수 있습니다. (기본값: 256)
* mime: MIME 디코더 설정.
  + decode-mime: yes로 설정하면 SMTP 트랜잭션에서 MIME 메시지를 디코드합니다.
  + decode-base64: yes로 설정하면 Base64 인코딩된 MIME 본문을 디코드합니다.
  + decode-quoted-printable: yes로 설정하면 Quoted-Printable 인코딩된 MIME 본문을 디코드합니다.
  + header-value-depth: 헤더 데이터 값 저장 시의 최대 바이트 수를 설정합니다. (기본값: 2000)
  + extract-urls: yes로 설정하면 MIME 본문에서 URL을 추출하여 상태 데이터 구조에 저장합니다.

위와 같은 설정을 통해 Suricata가 다양한 애플리케이션 계층 프로토콜을 효과적으로 감지하고 분석할 수 있도록 할 수 있습니다. 각 프로토콜의 설정은 필요에 따라 조정할 수 있으며, 이를 통해 네트워크 트래픽을 더 세밀하게 모니터링할 수 있습니다.

# Scheme of URLs to extract

# (default is [http])

#extract-urls-schemes: [http, https, ftp, mailto]

# Log the scheme of URLs that are extracted

# (default is no)

#log-url-scheme: yes

# Set to yes to compute the md5 of the mail body. You will then

# be able to journalize it.

body-md5: no

# Configure inspected-tracker for file\_data keyword

inspected-tracker:

content-limit: 100000

content-inspect-min-size: 32768

content-inspect-window: 4096

imap:

enabled: detection-only

smb:

enabled: yes

detection-ports:

dp: 139, 445

# Maximum number of live SMB transactions per flow

# max-tx: 1024

# Stream reassembly size for SMB streams. By default track it completely.

#stream-depth: 0

nfs:

enabled: yes

# max-tx: 1024

tftp:

enabled: yes

dns:

tcp:

enabled: yes

detection-ports:

dp: 53

udp:

enabled: yes

detection-ports:

dp: 53

http:

enabled: yes

# Byte Range Containers default settings

# byterange:

# memcap: 100mb

# timeout: 60

# memcap: Maximum memory capacity for HTTP

# Default is unlimited, values can be 64mb, e.g.

# default-config: Used when no server-config matches

# personality: List of personalities used by default

# request-body-limit: Limit reassembly of request body for inspection

# by http\_client\_body & pcre /P option.

# response-body-limit: Limit reassembly of response body for inspection

# by file\_data, http\_server\_body & pcre /Q option.

#

# For advanced options, see the user guide

# server-config: List of server configurations to use if address matches

# address: List of IP addresses or networks for this block

# personality: List of personalities used by this block

#

# Then, all the fields from default-config can be overloaded

#

# Currently Available Personalities:

# Minimal, Generic, IDS (default), IIS\_4\_0, IIS\_5\_0, IIS\_5\_1, IIS\_6\_0,

# IIS\_7\_0, IIS\_7\_5, Apache\_2

libhtp:

default-config:

personality: IDS

# Can be specified in kb, mb, gb. Just a number indicates

# it's in bytes.

request-body-limit: 100kb

response-body-limit: 100kb

# inspection limits

request-body-minimal-inspect-size: 32kb

request-body-inspect-window: 4kb

response-body-minimal-inspect-size: 40kb

response-body-inspect-window: 16kb

# response body decompression (0 disables)

response-body-decompress-layer-limit: 2

# auto will use http-body-inline mode in IPS mode, yes or no set it statically

http-body-inline: auto

# Decompress SWF files. Disabled by default.

# Two types: 'deflate', 'lzma', 'both' will decompress deflate and lzma

# compress-depth:

# Specifies the maximum amount of data to decompress,

# set 0 for unlimited.

# decompress-depth:

# Specifies the maximum amount of decompressed data to obtain,

# set 0 for unlimited.

swf-decompression:

enabled: no

type: both

compress-depth: 100kb

decompress-depth: 100kb

# Use a random value for inspection sizes around the specified value.

# This lowers the risk of some evasion techniques but could lead

# to detection change between runs. It is set to 'yes' by default.

#randomize-inspection-sizes: yes

# If "randomize-inspection-sizes" is active, the value of various

# inspection size will be chosen from the [1 - range%, 1 + range%]

# range

# Default value of "randomize-inspection-range" is 10.

#randomize-inspection-range: 10

# decoding

double-decode-path: no

double-decode-query: no

# Can enable LZMA decompression

#lzma-enabled: false

# Memory limit usage for LZMA decompression dictionary

# Data is decompressed until dictionary reaches this size

#lzma-memlimit: 1mb

# Maximum decompressed size with a compression ratio

# above 2048 (only LZMA can reach this ratio, deflate cannot)

#compression-bomb-limit: 1mb

# Maximum time spent decompressing a single transaction in usec

#decompression-time-limit: 100000

# Maximum number of live transactions per flow

#max-tx: 512

server-config:

#- apache:

# address: [192.168.1.0/24, 127.0.0.0/8, "::1"]

# personality: Apache\_2

# # Can be specified in kb, mb, gb. Just a number indicates

# # it's in bytes.

# request-body-limit: 4096

# response-body-limit: 4096

# double-decode-path: no

# double-decode-query: no

#- iis7:

# address:

# - 192.168.0.0/24

# - 192.168.10.0/24

# personality: IIS\_7\_0

# # Can be specified in kb, mb, gb. Just a number indicates

# # it's in bytes.

# request-body-limit: 4096

# response-body-limit: 4096

# double-decode-path: no

# double-decode-query: no

Suricata의 다양한 프로토콜에 대한 더 깊은 구성 요소들을 포함하고 있습니다. URL 추출, IMAP, SMB, NFS, TFTP, DNS, HTTP와 같은 프로토콜에 대한 세부 설정을 다루고 있습니다.

### **URL 추출 설정**

#### **URL 추출 관련 설정**

* extract-urls-schemes: URL을 추출할 때 사용할 스킴을 지정합니다. 기본값은 [http]이며, 주석 처리된 상태에서 [http, https, ftp, mailto]와 같이 여러 스킴을 추가할 수 있습니다.
* log-url-scheme: URL 스킴을 로깅할지 여부를 결정합니다. yes로 설정하면 추출된 URL의 스킴이 로그에 기록됩니다.
* body-md5: 메일 본문의 MD5 해시를 계산할지를 결정합니다. yes로 설정하면 저널링이 가능해집니다.

#### **inspected-tracker**

* content-limit: 추적할 수 있는 콘텐츠의 최대 크기를 설정합니다. 여기서는 100000 바이트로 설정되어 있습니다.
* content-inspect-min-size: 검사할 최소 콘텐츠 크기를 설정합니다. 기본값은 32768 바이트입니다.
* content-inspect-window: 콘텐츠 검사 시 사용할 윈도우 크기를 설정합니다. 여기서는 4096 바이트로 설정되어 있습니다.

### **프로토콜별 설정**

#### **imap**

* enabled: detection-only: IMAP 프로토콜 감지는 가능하지만, 분석은 비활성화 상태입니다.

#### **smb**

* enabled: yes: SMB 프로토콜 감지 및 분석이 활성화되어 있습니다.
* detection-ports: SMB 감지를 위한 포트로 139와 445가 설정되어 있습니다.
* max-tx: 각 흐름에 대해 유지할 수 있는 최대 SMB 트랜잭션 수를 설정합니다. 기본값은 1024입니다.
* stream-depth: SMB 스트림 재조립 크기. 기본적으로 전체 추적이 가능합니다.

#### **nfs**

* enabled: yes: NFS(Network File System) 프로토콜 감지 및 분석이 활성화됨.

#### **tftp**

* enabled: yes: TFTP(Trivial File Transfer Protocol) 감지 및 분석이 활성화됨.

#### **dns**

* **TCP 설정**
  + tcp.enabled: yes: DNS의 TCP 프로토콜 감지 및 분석이 활성화되어 있습니다.
  + tcp.detection-ports: TCP DNS 감지를 위한 포트는 53입니다.
* **UDP 설정**
  + udp.enabled: yes: DNS의 UDP 프로토콜 감지 및 분석이 활성화되어 있습니다.
  + udp.detection-ports: UDP DNS 감지를 위한 포트는 53입니다.

#### **http**

* enabled: yes: HTTP 프로토콜 감지 및 분석이 활성화되어 있습니다.
* **Byte Range Containers**: 기본 설정이 주석 처리되어 있으며, 필요 시 메모리 용량과 타임아웃을 설정할 수 있습니다.
* memcap: HTTP 분석에 사용할 최대 메모리 용량을 설정할 수 있으며, 기본값은 무제한입니다.

##### **libhtp 설정**

* default-config: HTTP 서버의 기본 설정을 정의합니다.
  + personality: 기본적으로 IDS로 설정됩니다.
  + request-body-limit: 요청 본문의 최대 크기를 설정합니다. 여기서는 100kb입니다.
  + response-body-limit: 응답 본문의 최대 크기를 설정합니다. 여기서는 100kb입니다.
* **검사 한계 설정**
  + request-body-minimal-inspect-size: 최소 검사 크기를 32kb로 설정합니다.
  + request-body-inspect-window: 검사 시 사용할 윈도우 크기를 4kb로 설정합니다.
  + response-body-minimal-inspect-size: 응답 본문의 최소 검사 크기를 40kb로 설정합니다.
  + response-body-inspect-window: 응답 본문 검사 시 사용할 윈도우 크기를 16kb로 설정합니다.
* **SWF 파일 압축 해제**: 기본적으로 비활성화되어 있으며, 압축 해제 깊이를 설정할 수 있습니다.
  + compress-depth: 최대 압축 해제 깊이를 설정합니다. 기본값은 100kb입니다.
  + decompress-depth: 최대 압축 해제된 데이터 크기를 설정합니다. 기본값은 100kb입니다.

이 설정들을 통해 Suricata가 다양한 프로토콜을 효과적으로 감지하고 검사할 수 있도록 하고, 필요에 따라 세부적인 조정을 할 수 있습니다.

# Note: Modbus probe parser is minimalist due to the limited usage in the field.

# Only Modbus message length (greater than Modbus header length)

# and protocol ID (equal to 0) are checked in probing parser

# It is important to enable detection port and define Modbus port

# to avoid false positives

modbus:

# How many unanswered Modbus requests are considered a flood.

# If the limit is reached, the app-layer-event:modbus.flooded; will match.

#request-flood: 500

enabled: no

detection-ports:

dp: 502

# According to MODBUS Messaging on TCP/IP Implementation Guide V1.0b, it

# is recommended to keep the TCP connection opened with a remote device

# and not to open and close it for each MODBUS/TCP transaction. In that

# case, it is important to set the depth of the stream reassembling as

# unlimited (stream.reassembly.depth: 0)

# Stream reassembly size for modbus. By default track it completely.

stream-depth: 0

# DNP3

dnp3:

enabled: no

detection-ports:

dp: 20000

# SCADA EtherNet/IP and CIP protocol support

enip:

enabled: no

detection-ports:

dp: 44818

sp: 44818

ntp:

enabled: yes

quic:

enabled: yes

dhcp:

enabled: yes

sip:

#enabled: yes

Suricata에서 다양한 산업용 프로토콜(예: Modbus, DNP3, EtherNet/IP) 및 기타 네트워크 프로토콜(NTP, QUIC, DHCP, SIP)에 대한 세부 구성을 포함하고 있어요.

### **프로토콜별 설정**

#### **modbus**

* **주석 내용**: Modbus 프로토콜은 필드에서의 사용이 제한적이기 때문에 최소한의 파서로 설정되어 있어요. Modbus 메시지의 길이(헤더 길이보다 큰 경우)와 프로토콜 ID(0이어야 함)만 검사합니다.
* **enabled**: no로 설정되어 있어 현재 Modbus 감지가 비활성화되어 있습니다.
* **detection-ports**: Modbus 감지를 위한 포트는 502로 설정되어 있습니다.
* **stream-depth**: 스트림 재조립 크기를 설정합니다. 기본적으로 완전 추적이 가능하며, 0으로 설정되어 있습니다.
* **request-flood**: 주석 처리된 상태로, 설정된 요청 수를 초과하면 플러드 공격으로 간주하는 기준을 설정합니다. 여기서는 500으로 설정되어 있습니다.

#### **dnp3**

* **enabled**: no로 설정되어 DNP3 프로토콜 감지가 비활성화되어 있습니다.
* **detection-ports**: DNP3 감지를 위한 포트는 20000으로 설정되어 있습니다.

#### **enip (EtherNet/IP 및 CIP 프로토콜)**

* **enabled**: no로 설정되어 EtherNet/IP 감지가 비활성화되어 있습니다.
* **detection-ports**: EtherNet/IP 감지를 위한 포트는 44818으로 설정되어 있으며, 소스 포트도 동일하게 설정되어 있습니다.

#### **ntp**

* **enabled**: yes로 설정되어 NTP(Network Time Protocol) 감지가 활성화되어 있습니다.

#### **quic**

* **enabled**: yes로 설정되어 QUIC(Quick UDP Internet Connections) 프로토콜 감지가 활성화되어 있습니다.

#### **dhcp**

* **enabled**: yes로 설정되어 DHCP(Dynamic Host Configuration Protocol) 감지가 활성화되어 있습니다.

#### **sip**

* **enabled**: 주석 처리되어 있어서 현재 SIP(Session Initiation Protocol) 감지가 비활성화되어 있습니다.

### **프로토콜 사용 시 유의사항**

* Modbus와 같은 산업 프로토콜은 올바른 포트와 깊이 설정을 통해 허위 양성(false positive)을 피하는 것이 중요합니다.
* 각 프로토콜에 대한 감지 설정을 필요에 따라 활성화하거나 비활성화하여 네트워크 환경에 맞게 조정할 수 있습니다.

이런 설정을 통해 Suricata는 다양한 프로토콜을 적절하게 감지하고 관리할 수 있습니다.

# Limit for the maximum number of asn1 frames to decode (default 256)

asn1-max-frames: 256

# Datasets default settings

datasets:

# Default fallback memcap and hashsize values for datasets in case these

# were not explicitly defined.

defaults:

#memcap: 100mb

#hashsize: 2048

rules:

# Set to true to allow absolute filenames and filenames that use

# ".." components to reference parent directories in rules that specify

# their filenames.

#allow-absolute-filenames: false

# Allow datasets in rules write access for "save" and

# "state". This is enabled by default, however write access is

# limited to the data directory.

#allow-write: true

Suricata의 asn1 프레임 제한 설정과 datasets 기본 설정에 대해 설명하고 있어요.

### **ASN1 프레임 제한 설정**

#### **asn1-max-frames**

* **설명**: ASN1 프레임은 주로 보안 관련 프로토콜(예: SSL/TLS, SNMP)에서 사용되는 구조로, 디코딩할 프레임의 최대 개수를 제한합니다.
* **기본값**: 256
  + ASN1 프레임을 디코딩할 때 최대 256개의 프레임만 처리합니다. 이를 통해 프로토콜 처리 시 과도한 리소스 사용을 방지할 수 있어요.
  + 이 값을 늘리면 더 많은 프레임을 처리할 수 있지만, 리소스 사용이 증가할 수 있으니 신중하게 설정하는 게 좋아요.

### **데이터셋 기본 설정**

datasets는 Suricata의 데이터셋 기능과 관련된 설정을 정의합니다. 데이터셋은 외부 데이터를 Suricata 규칙에서 사용할 수 있게 하는 기능으로, IP 블랙리스트나 파일 해시 리스트 같은 데이터를 규칙에서 활용할 수 있어요.

#### **defaults**

* **memcap**: 메모리 사용량 제한. 주석 처리되어 있어서 기본값은 설정되지 않았습니다. 예를 들어, memcap: 100mb로 설정하면 데이터셋에 할당되는 메모리를 100MB로 제한할 수 있어요.
* **hashsize**: 해시 테이블 크기. 주석 처리되어 있어 기본값이 설정되지 않았습니다. 예를 들어, hashsize: 2048로 설정하면 데이터셋의 해시 테이블 크기를 2048로 설정할 수 있어요.
  + 이 값들을 적절히 설정해 데이터셋의 메모리 사용량과 성능을 조절할 수 있습니다.

#### **rules**

* **allow-absolute-filenames**:
  + 주석 처리되어 있어 기본값인 false로 설정되어 있습니다.
  + true로 설정 시, 규칙 파일에서 절대 경로나 상위 디렉토리(..) 참조를 허용합니다.
  + 보안상의 이유로 기본값은 false로 설정되어 있으며, 필요한 경우에만 사용해야 합니다.
* **allow-write**:
  + 주석 처리되어 있어 기본값인 true로 설정되어 있습니다.
  + 규칙에서 데이터셋의 save와 state 옵션을 통해 데이터를 쓸 수 있도록 허용합니다.
  + 다만, 쓰기 작업은 데이터 디렉토리로 제한됩니다. 이 설정을 통해 데이터셋의 상태를 저장하고 관리할 수 있어요.

이 설정들을 통해 Suricata가 ASN1 디코딩을 효율적으로 처리하고, 데이터셋의 메모리 사용량을 조절하며, 규칙 파일의 보안성을 유지할 수 있습니다.

##

## Detection settings

##

# Set the order of alerts based on actions

# The default order is pass, drop, reject, alert

# action-order:

# - pass

# - drop

# - reject

# - alert

# Define maximum number of possible alerts that can be triggered for the same

# packet. Default is 15

#packet-alert-max: 15

# Exception Policies

#

# Define a common behavior for all exception policies.

# In IPS mode, the default is drop-flow. For cases when that's not possible, the

# engine will fall to drop-packet. To fallback to old behavior (setting each of

# them individually, or ignoring all), set this to ignore.

# All values available for exception policies can be used, and there is one

# extra option: auto - which means drop-flow or drop-packet (as explained above)

# in IPS mode, and ignore in IDS mode. Exception policy values are: drop-packet,

# drop-flow, reject, bypass, pass-packet, pass-flow, ignore (disable).

exception-policy: auto

# IP Reputation

#reputation-categories-file: /etc/suricata/iprep/categories.txt

#default-reputation-path: /etc/suricata/iprep

#reputation-files:

# - reputation.list

# When run with the option --engine-analysis, the engine will read each of

# the parameters below, and print reports for each of the enabled sections

# and exit. The reports are printed to a file in the default log dir

# given by the parameter "default-log-dir", with engine reporting

# subsection below printing reports in its own report file.

engine-analysis:

# enables printing reports for fast-pattern for every rule.

rules-fast-pattern: yes

# enables printing reports for each rule

rules: yes

#recursion and match limits for PCRE where supported

pcre:

match-limit: 3500

match-limit-recursion: 1500

Suricata의 탐지(detection) 설정에 대한 내용입니다. 탐지 설정은 Suricata가 네트워크 트래픽을 분석하고, 알림을 생성하며, 예외 상황을 처리하는 방식을 정의합니다.

### **탐지 설정 (Detection Settings)**

#### **1. Alert Action Order (action-order)**

* **설명**: 패킷 처리 시 적용할 동작(액션)의 우선순위를 정의합니다. 기본 순서는 pass, drop, reject, alert입니다.
  + pass: 패킷을 허용합니다. 이후 규칙 평가를 중단합니다.
  + drop: 패킷을 차단하고 네트워크에서 버립니다.
  + reject: 패킷을 차단하고 거부 응답을 보냅니다.
  + alert: 패킷에 대해 경고를 생성합니다.
* **기본값**: pass, drop, reject, alert
  + 주석이 해제되면 이 순서대로 경고가 처리됩니다.

#### **2. 최대 경고 수 (packet-alert-max)**

* **설명**: 하나의 패킷에 대해 발생할 수 있는 최대 경고 수를 정의합니다.
* **기본값**: 15
  + 하나의 패킷에 대해 15개의 경고가 발생할 수 있습니다. 이 수치를 넘어가는 경고는 무시됩니다.
  + 이를 통해 과도한 경고 발생을 방지하고 성능을 최적화할 수 있습니다.

#### **3. 예외 정책 (exception-policy)**

* **설명**: 예외 상황이 발생했을 때 적용할 기본 동작을 정의합니다.
* **값**:
  + auto: IPS 모드에서는 drop-flow 또는 drop-packet, IDS 모드에서는 ignore.
  + drop-packet: 문제가 있는 패킷만 차단합니다.
  + drop-flow: 문제가 있는 패킷이 포함된 전체 흐름을 차단합니다.
  + reject: 문제를 발견한 후 연결을 거부합니다.
  + bypass: 패킷을 더 이상 처리하지 않고 우회시킵니다.
  + pass-packet: 문제를 발견해도 패킷을 통과시킵니다.
  + pass-flow: 문제가 있는 흐름을 그대로 통과시킵니다.
  + ignore: 예외 상황을 무시합니다.
* **기본값**: auto
  + 기본적으로 IPS 모드에서는 drop-flow 또는 drop-packet, IDS 모드에서는 ignore로 설정됩니다.

#### **4. IP 평판 (IP Reputation)**

* **설명**: IP 평판 기능을 사용하여 악성 IP 주소를 감지하고 필터링합니다.
  + reputation-categories-file: IP 평판 카테고리 파일 경로를 정의합니다.
  + default-reputation-path: IP 평판 파일의 기본 경로를 설정합니다.
  + reputation-files: 사용할 IP 평판 파일 목록을 정의합니다.
* **기본값**: 주석 처리되어 있어, IP 평판 기능이 비활성화되어 있습니다.
  + 이 기능을 활성화하려면 주석을 해제하고 필요한 파일 경로와 목록을 설정해야 합니다.

#### **5. 엔진 분석 (engine-analysis)**

* **설명**: Suricata 엔진의 성능 및 규칙 분석 보고서를 생성합니다.
  + rules-fast-pattern: 규칙마다 fast-pattern 보고서를 생성합니다. (yes로 설정되어 있음)
  + rules: 각 규칙에 대한 보고서를 생성합니다. (yes로 설정되어 있음)
* **기본값**: 주석이 해제되어 있어, 엔진 분석을 수행할 때 해당 보고서들이 생성됩니다.

#### **6. PCRE 설정 (pcre)**

* **설명**: PCRE(Perl Compatible Regular Expressions) 매칭의 재귀와 매칭 제한을 설정합니다.
  + match-limit: 정규표현식 매칭을 시도하는 최대 횟수. 기본값은 3500회입니다.
  + match-limit-recursion: 정규표현식 재귀 매칭의 최대 깊이. 기본값은 1500입니다.
  + 이를 통해 무한 재귀 및 과도한 매칭 시도로 인한 성능 저하를 방지할 수 있습니다.

이 설정들은 Suricata가 네트워크 트래픽을 모니터링하고, 경고를 생성하며, 예외 상황을 처리하는 방식을 결정합니다. 각 설정을 필요에 따라 조정하면, 성능 최적화와 함께 원하는 보안 정책을 적용할 수 있어요.

##

## Advanced Traffic Tracking and Reconstruction Settings

##

# Host specific policies for defragmentation and TCP stream

# reassembly. The host OS lookup is done using a radix tree, just

# like a routing table so the most specific entry matches.

host-os-policy:

# Make the default policy windows.

windows: [0.0.0.0/0]

bsd: []

bsd-right: []

old-linux: []

linux: []

old-solaris: []

solaris: []

hpux10: []

hpux11: []

irix: []

macos: []

vista: []

windows2k3: []

# Defrag settings:

# The memcap-policy value can be "drop-packet", "pass-packet", "reject" or

# "ignore" (which is the default).

defrag:

memcap: 32mb

# memcap-policy: ignore

hash-size: 65536

trackers: 65535 # number of defragmented flows to follow

max-frags: 65535 # number of fragments to keep (higher than trackers)

prealloc: yes

timeout: 60

# Enable defrag per host settings

# host-config:

#

# - dmz:

# timeout: 30

# address: [192.168.1.0/24, 127.0.0.0/8, 1.1.1.0/24, 2.2.2.0/24, "1.1.1.1", "2.2.2.2", "::1"]

#

# - lan:

# timeout: 45

# address:

# - 192.168.0.0/24

# - 192.168.10.0/24

# - 172.16.14.0/24

Suricata의 고급 트래픽 추적 및 재구성 설정(Advanced Traffic Tracking and Reconstruction Settings)에 관한 내용입니다. 이 설정들은 호스트별로 운영체제 정책을 적용하고, 트래픽의 재조합과 관련된 기능들을 제어합니다.

### **고급 트래픽 추적 및 재구성 설정 (Advanced Traffic Tracking and Reconstruction Settings)**

#### **1. 호스트 운영체제 정책 (host-os-policy)**

* **설명**: 특정 호스트 또는 네트워크에 대해 운영체제(OS)별 패킷 조립 정책을 설정합니다.
  + **운영체제별 정책**: 각 운영체제마다 다른 TCP 스트림 재조립 및 패킷 조립 방식을 사용할 수 있습니다.
  + **windows: [0.0.0.0/0]**: 모든 네트워크(0.0.0.0/0)에 대해 기본적으로 Windows 운영체제 정책을 적용합니다.
  + 나머지 운영체제(bsd, linux, macos, 등)는 현재 설정이 비어 있어서 적용되지 않습니다.

#### **2. 패킷 조각화 재조립 설정 (defrag)**

* **설명**: 네트워크 트래픽에서 조각화된 패킷을 다시 조립하는 설정을 정의합니다.
  + **memcap**: 패킷 조각화 재조립에 사용할 최대 메모리 용량을 정의합니다.
    - 기본값: 32mb
  + **memcap-policy**: 메모리 한도를 초과했을 때 수행할 동작을 설정합니다.
    - 가능한 값: drop-packet, pass-packet, reject, ignore
    - 기본값: ignore (주석 처리됨)
  + **hash-size**: 패킷 조각화 정보를 저장할 해시 테이블 크기를 설정합니다.
    - 기본값: 65536
  + **trackers**: 동시에 추적할 수 있는 조각화된 플로우의 최대 개수를 설정합니다.
    - 기본값: 65535
  + **max-frags**: 동시에 추적할 수 있는 조각의 최대 개수를 설정합니다.
    - 기본값: 65535 (플로우보다 많아야 함)
  + **prealloc**: 재조립에 사용할 메모리를 사전에 할당할지 여부를 설정합니다.
    - 기본값: yes
  + **timeout**: 조각화된 패킷이 재조립될 때까지 기다리는 시간(초)을 설정합니다.
    - 기본값: 60초

#### **3. 호스트별 조각화 재조립 설정 (host-config)**

* **설명**: 특정 네트워크 또는 호스트에 대해 패킷 조각화 재조립 설정을 적용할 수 있습니다.
  + **dmz**: DMZ(비무장지대) 영역에 대한 설정 예시
    - timeout: 30: 조각화된 패킷이 30초 내에 재조립되지 않으면 버립니다.
    - address: DMZ 영역에 포함되는 IP 주소 범위 또는 개별 IP 주소를 정의합니다.
  + **lan**: 내부 네트워크(LAN)에 대한 설정 예시
    - timeout: 45: 조각화된 패킷이 45초 내에 재조립되지 않으면 버립니다.
    - address: LAN에 포함되는 IP 주소 범위를 정의합니다.
  + host-config가 주석 처리되어 있어, 현재는 호스트별 설정이 적용되지 않습니다. 이를 활성화하려면 주석을 해제하고, 각 호스트별 설정을 정의해야 합니다.

이 설정들은 트래픽을 보다 효율적으로 관리하고, 특정 네트워크 또는 운영체제에 대해 최적화된 탐지 정책을 적용하는 데 유용합니다. 설정을 조정하여 특정 요구사항이나 네트워크 환경에 맞는 정책을 적용할 수 있어요.

# Flow settings:

# By default, the reserved memory (memcap) for flows is 32MB. This is the limit

# for flow allocation inside the engine. You can change this value to allow

# more memory usage for flows.

# The hash-size determines the size of the hash used to identify flows inside

# the engine, and by default the value is 65536.

# At startup, the engine can preallocate a number of flows, to get better

# performance. The number of flows preallocated is 10000 by default.

# emergency-recovery is the percentage of flows that the engine needs to

# prune before clearing the emergency state. The emergency state is activated

# when the memcap limit is reached, allowing new flows to be created, but

# pruning them with the emergency timeouts (they are defined below).

# If the memcap is reached, the engine will try to prune flows

# with the default timeouts. If it doesn't find a flow to prune, it will set

# the emergency bit and it will try again with more aggressive timeouts.

# If that doesn't work, then it will try to kill the oldest flows using

# last time seen flows.

# The memcap can be specified in kb, mb, gb. Just a number indicates it's

# in bytes.

# The memcap-policy can be "drop-packet", "pass-packet", "reject" or "ignore"

# (which is the default).

flow:

memcap: 128mb

#memcap-policy: ignore

hash-size: 65536

prealloc: 10000

emergency-recovery: 30

#managers: 1 # default to one flow manager

#recyclers: 1 # default to one flow recycler thread

# This option controls the use of VLAN ids in the flow (and defrag)

# hashing. Normally this should be enabled, but in some (broken)

# setups where both sides of a flow are not tagged with the same VLAN

# tag, we can ignore the VLAN id's in the flow hashing.

vlan:

use-for-tracking: true

# This option controls the use of livedev ids in the flow (and defrag)

# hashing. This is enabled by default and should be disabled if

# multiple live devices are used to capture traffic from the same network

livedev:

use-for-tracking: true

# Specific timeouts for flows. Here you can specify the timeouts that the

# active flows will wait to transit from the current state to another, on each

# protocol. The value of "new" determines the seconds to wait after a handshake or

# stream startup before the engine frees the data of that flow it doesn't

# change the state to established (usually if we don't receive more packets

# of that flow). The value of "established" is the amount of

# seconds that the engine will wait to free the flow if that time elapses

# without receiving new packets or closing the connection. "closed" is the

# amount of time to wait after a flow is closed (usually zero). "bypassed"

# timeout controls locally bypassed flows. For these flows we don't do any other

# tracking. If no packets have been seen after this timeout, the flow is discarded.

#

# There's an emergency mode that will become active under attack circumstances,

# making the engine to check flow status faster. This configuration variables

# use the prefix "emergency-" and work similar as the normal ones.

# Some timeouts doesn't apply to all the protocols, like "closed", for udp and

# icmp.

flow-timeouts:

default:

new: 30

established: 300

closed: 0

bypassed: 100

emergency-new: 10

emergency-established: 100

emergency-closed: 0

emergency-bypassed: 50

tcp:

new: 60

established: 600

closed: 60

bypassed: 100

emergency-new: 5

emergency-established: 100

emergency-closed: 10

emergency-bypassed: 50

udp:

new: 30

established: 300

bypassed: 100

emergency-new: 10

emergency-established: 100

emergency-bypassed: 50

icmp:

new: 30

established: 300

bypassed: 100

emergency-new: 10

emergency-established: 100

emergency-bypassed: 50

Suricata의 플로우(Flow) 관리와 관련된 설정입니다. 플로우는 네트워크 트래픽에서 송신자와 수신자 간의 데이터 흐름을 나타내며, 이를 추적하고 관리하는 것이 네트워크 침입 탐지 및 방지 시스템의 중요한 기능 중 하나입니다.

### **플로우 설정 (Flow Settings)**

#### **1. 플로우 메모리 제한 설정 (flow)**

* **memcap: 128mb**
  + 설명: 플로우를 관리하기 위해 사용할 메모리의 최대 크기를 설정합니다. 기본값은 32MB지만, 여기서는 128MB로 설정되어 있습니다.
  + 값이 크면 더 많은 플로우를 추적할 수 있지만, 메모리 사용량도 증가합니다.
* **memcap-policy**
  + 설명: 메모리 한도를 초과했을 때 어떤 행동을 취할지 결정합니다. 주석 처리되어 있으므로 기본적으로 ignore로 설정됩니다.
  + 가능한 값: drop-packet, pass-packet, reject, ignore
* **hash-size: 65536**
  + 설명: 플로우를 식별하기 위해 사용하는 해시 테이블의 크기를 설정합니다. 큰 값일수록 더 많은 플로우를 효과적으로 추적할 수 있지만, 메모리 사용량도 증가합니다.
* **prealloc: 10000**
  + 설명: 성능 향상을 위해 초기화 시 사전 할당할 플로우의 수를 지정합니다. 기본값은 10,000입니다.
* **emergency-recovery: 30**
  + 설명: 메모리 한도 초과 시 긴급 상태로 진입하는데, 긴급 상태 해제 전에 제거해야 할 플로우의 비율을 설정합니다.
  + 기본값은 30%로, 메모리 사용량이 일정 수준 이하로 떨어지면 긴급 상태를 해제합니다.
* **managers와 recyclers**
  + 설명: 플로우 관리 및 재활용을 담당하는 스레드 수를 설정합니다. 기본값은 각각 1로 주석 처리되어 있습니다.

#### **2. VLAN 및 라이브 장치 설정**

* **vlan: use-for-tracking: true**
  + 설명: 플로우 및 조각화 해싱에 VLAN ID를 사용할지 여부를 설정합니다.
  + true로 설정하면 VLAN ID를 사용하여 네트워크 세그먼트를 더 정확하게 구분합니다.
* **livedev: use-for-tracking: true**
  + 설명: 라이브 장치 ID를 플로우 해싱에 사용할지 여부를 설정합니다.
  + true로 설정하면 여러 장치에서 동일 네트워크의 트래픽을 캡처할 때 충돌을 방지합니다.

#### **3. 플로우 타임아웃 설정 (flow-timeouts)**

플로우 상태에 따라 타임아웃 시간을 설정합니다. 각각의 프로토콜에 대해 별도의 타임아웃을 지정할 수 있습니다.

* **default** (기본 타임아웃)
  + new: 30: 새로운 플로우의 초기 상태에서 30초 동안 상태 변화가 없으면 삭제합니다.
  + established: 300: 확립된 플로우에서 300초 동안 트래픽이 없으면 삭제합니다.
  + closed: 0: 플로우가 닫히면 즉시 삭제합니다.
  + bypassed: 100: 로컬에서 우회된 플로우의 타임아웃 (100초 후 삭제).
  + **긴급 모드 타임아웃 (emergency- prefix)**: 긴급 상태에서 사용할 더 짧은 타임아웃 설정.
* **tcp, udp, icmp**
  + 각 프로토콜에 대해 개별 타임아웃 설정을 정의합니다.
  + 예를 들어, tcp의 경우 new는 60초, established는 600초, closed는 60초로 설정되어 있습니다.
  + 긴급 상태에서는 더 짧은 타임아웃 값을 사용하여 플로우를 빠르게 제거하고 메모리를 확보합니다.

이 설정들은 네트워크 환경에 맞게 조정되어야 시스템이 효율적으로 동작할 수 있습니다. 만약 네트워크에서 다루는 트래픽 양이 많거나 다양한 플로우를 다뤄야 한다면, 메모리 설정과 타임아웃 설정을 적절히 변경하는 것이 좋습니다.

# Stream engine settings. Here the TCP stream tracking and reassembly

# engine is configured.

#

# stream:

# memcap: 64mb # Can be specified in kb, mb, gb. Just a

# # number indicates it's in bytes.

# memcap-policy: ignore # Can be "drop-flow", "pass-flow", "bypass",

# # "drop-packet", "pass-packet", "reject" or

# # "ignore" default is "ignore"

# checksum-validation: yes # To validate the checksum of received

# # packet. If csum validation is specified as

# # "yes", then packets with invalid csum values will not

# # be processed by the engine stream/app layer.

# # Warning: locally generated traffic can be

# # generated without checksum due to hardware offload

# # of checksum. You can control the handling of checksum

# # on a per-interface basis via the 'checksum-checks'

# # option

# prealloc-sessions: 2048 # 2k sessions prealloc'd per stream thread

# midstream: false # don't allow midstream session pickups

# midstream-policy: ignore # Can be "drop-flow", "pass-flow", "bypass",

# # "drop-packet", "pass-packet", "reject" or

# # "ignore" default is "ignore"

# async-oneside: false # don't enable async stream handling

# inline: no # stream inline mode

# drop-invalid: yes # in inline mode, drop packets that are invalid with regards to streaming engine

# max-syn-queued: 10 # Max different SYNs to queue

# max-synack-queued: 5 # Max different SYN/ACKs to queue

# bypass: no # Bypass packets when stream.reassembly.depth is reached.

# # Warning: first side to reach this triggers

# # the bypass.

# liberal-timestamps: false # Treat all timestamps as if the Linux policy applies. This

# # means it's slightly more permissive. Enabled by default.

#

# reassembly:

# memcap: 256mb # Can be specified in kb, mb, gb. Just a number

# # indicates it's in bytes.

# memcap-policy: ignore # Can be "drop-flow", "pass-flow", "bypass",

# # "drop-packet", "pass-packet", "reject" or

# # "ignore" default is "ignore"

# depth: 1mb # Can be specified in kb, mb, gb. Just a number

# # indicates it's in bytes.

# toserver-chunk-size: 2560 # inspect raw stream in chunks of at least

# # this size. Can be specified in kb, mb,

# # gb. Just a number indicates it's in bytes.

# toclient-chunk-size: 2560 # inspect raw stream in chunks of at least

# # this size. Can be specified in kb, mb,

# # gb. Just a number indicates it's in bytes.

# randomize-chunk-size: yes # Take a random value for chunk size around the specified value.

# # This lowers the risk of some evasion techniques but could lead

# # to detection change between runs. It is set to 'yes' by default.

# randomize-chunk-range: 10 # If randomize-chunk-size is active, the value of chunk-size is

# # a random value between (1 - randomize-chunk-range/100)\*toserver-chunk-size

# # and (1 + randomize-chunk-range/100)\*toserver-chunk-size and the same

# # calculation for toclient-chunk-size.

# # Default value of randomize-chunk-range is 10.

#

# raw: yes # 'Raw' reassembly enabled or disabled.

# # raw is for content inspection by detection

# # engine.

#

# segment-prealloc: 2048 # number of segments preallocated per thread

#

# check-overlap-different-data: true|false

# # check if a segment contains different data

# # than what we've already seen for that

# # position in the stream.

# # This is enabled automatically if inline mode

# # is used or when stream-event:reassembly\_overlap\_different\_data;

# # is used in a rule.

##

stream:

memcap: 64mb

#memcap-policy: ignore

checksum-validation: yes # reject incorrect csums

#midstream: false

#midstream-policy: ignore

inline: auto # auto will use inline mode in IPS mode, yes or no set it statically

reassembly:

memcap: 256mb

#memcap-policy: ignore

depth: 1mb # reassemble 1mb into a stream

toserver-chunk-size: 2560

toclient-chunk-size: 2560

randomize-chunk-size: yes

#randomize-chunk-range: 10

#raw: yes

#segment-prealloc: 2048

#check-overlap-different-data: true

Suricata의 스트림 엔진과 TCP 스트림 재조립(Reassembly) 엔진에 관한 것으로, 네트워크 트래픽의 세부적인 추적 및 재구성을 담당합니다. 스트림 엔진은 TCP 세션을 추적하고, 이를 바탕으로 네트워크 침입 탐지/방지 시스템이 정상적인 트래픽과 비정상적인 트래픽을 구분할 수 있도록 돕습니다.

### **스트림 엔진 설정 (Stream Engine Settings)**

#### **1. 기본 스트림 엔진 설정 (stream)**

* **memcap: 64mb**
  + 설명: 스트림 추적에 사용할 메모리의 최대 크기를 지정합니다. 여기서는 64MB로 설정되어 있으며, 더 큰 값을 설정하면 더 많은 스트림 세션을 관리할 수 있지만, 메모리 사용량도 증가합니다.
* **memcap-policy**
  + 설명: 메모리 제한을 초과했을 때의 동작을 설정합니다. 주석 처리되어 있어서 기본값인 ignore가 적용됩니다.
  + 가능한 값: drop-flow, pass-flow, bypass, drop-packet, pass-packet, reject, ignore
* **checksum-validation: yes**
  + 설명: 수신된 패킷의 체크섬을 검증합니다. yes로 설정되어 있으면, 체크섬이 잘못된 패킷은 스트림/애플리케이션 계층에서 처리되지 않습니다.
  + 로컬에서 생성된 트래픽은 체크섬이 없을 수 있으므로, 인터페이스별로 체크섬 검사 제어가 필요할 수 있습니다.
* **midstream: false**
  + 설명: 세션 시작 중간에 스트림 세션을 수집하지 않도록 설정합니다. 기본값은 false입니다.
* **midstream-policy**
  + 설명: 스트림 중간 수집이 허용된 경우의 정책을 설정합니다. 주석 처리되어 있어서 기본값인 ignore가 적용됩니다.
  + 가능한 값: drop-flow, pass-flow, bypass, drop-packet, pass-packet, reject, ignore
* **inline: auto**
  + 설명: 인라인 모드를 자동으로 사용합니다. IPS(침입 방지 시스템) 모드에서는 자동으로 인라인 모드를 활성화하며, yes 또는 no로 설정하면 수동으로 활성화/비활성화할 수 있습니다.
* **drop-invalid: yes**
  + 설명: 인라인 모드에서 스트리밍 엔진에 부적합한 패킷을 삭제합니다.

#### **2. 스트림 재조립 설정 (reassembly)**

TCP 세션의 데이터를 재조립하여 완전한 데이터 스트림을 만듭니다. 이를 통해 Suricata는 데이터를 더욱 정확하게 분석할 수 있습니다.

* **memcap: 256mb**
  + 설명: 재조립 과정에서 사용할 메모리의 최대 크기를 지정합니다. 여기서는 256MB로 설정되어 있습니다.
* **memcap-policy**
  + 설명: 재조립 메모리 제한을 초과했을 때의 동작을 설정합니다. 주석 처리되어 있어서 기본값인 ignore가 적용됩니다.
* **depth: 1mb**
  + 설명: 스트림 재조립의 최대 깊이(바이트 수)를 지정합니다. 여기서는 1MB로 설정되어 있습니다. 더 큰 값을 설정하면 더 많은 데이터를 재조립할 수 있지만, 메모리 사용량도 증가합니다.
* **toserver-chunk-size: 2560**
  + 설명: 서버로 향하는 트래픽을 최소 2560바이트 크기로 재조립하여 검사합니다. 이 값을 조정하여 트래픽 처리 효율을 최적화할 수 있습니다.
* **toclient-chunk-size: 2560**
  + 설명: 클라이언트로 향하는 트래픽을 최소 2560바이트 크기로 재조립하여 검사합니다.
* **randomize-chunk-size: yes**
  + 설명: 청크 크기를 설정된 값 주위에서 랜덤하게 조정하여 패턴 탐지를 어렵게 만듭니다. 기본값은 yes입니다.
* **randomize-chunk-range: 10**
  + 설명: randomize-chunk-size가 활성화된 경우, 청크 크기를 설정된 값의 ±10% 범위 내에서 랜덤하게 선택합니다.
* **raw: yes**
  + 설명: "raw" 재조립을 활성화하여 검출 엔진이 콘텐츠를 검사할 수 있도록 합니다.
* **segment-prealloc: 2048**
  + 설명: 스레드당 사전 할당된 세그먼트 수를 설정합니다. 기본값은 2048개입니다.
* **check-overlap-different-data: true**
  + 설명: 스트림의 동일 위치에서 이미 본 데이터와 다른 데이터가 포함된 세그먼트를 확인합니다. 기본적으로 true로 설정되어 있으며, 인라인 모드나 특정 스트림 이벤트를 사용할 때 자동으로 활성화됩니다.

이러한 설정들은 네트워크 환경과 Suricata의 역할에 따라 최적화해야 합니다. 예를 들어, 고속 네트워크에서는 메모리 제한을 더 높게 설정하고, 인라인 모드에서는 drop-invalid와 같은 설정을 활성화하여 악성 트래픽을 더 효과적으로 차단할 수 있습니다.

# Host table:

#

# Host table is used by the tagging and per host thresholding subsystems.

#

host:

hash-size: 4096

prealloc: 1000

memcap: 32mb

# IP Pair table:

#

# Used by xbits 'ippair' tracking.

#

#ippair:

# hash-size: 4096

# prealloc: 1000

# memcap: 32mb

Suricata의 호스트 및 IP 페어 테이블에 관한 것입니다. 호스트 테이블과 IP 페어 테이블은 Suricata가 네트워크 상의 호스트와 IP 주소 쌍을 관리하고 추적하는 데 사용됩니다. 이러한 테이블은 태그 지정(tagging) 및 호스트별 임계값 조정(thresholding) 등의 기능에서 중요한 역할을 합니다.

### **호스트 테이블 설정 (host)**

호스트 테이블은 네트워크 상의 개별 호스트에 대한 정보를 저장하고 추적합니다. Suricata는 이를 통해 각 호스트의 행동 패턴을 분석하고, 특정 호스트에 대한 경고 임계값 등을 설정할 수 있습니다.

* **hash-size: 4096**
  + 설명: 호스트 테이블의 해시 테이블 크기를 지정합니다. 해시 테이블은 호스트 정보를 저장하는 데 사용되며, 크기가 클수록 더 많은 호스트 정보를 저장할 수 있습니다.
  + 예: 4096개의 슬롯을 사용하여 호스트 정보를 저장하고 빠르게 검색할 수 있습니다.
* **prealloc: 1000**
  + 설명: 시작 시 미리 할당할 호스트 엔트리의 수를 지정합니다. 이 값을 높이면 초기 메모리 사용량이 증가하지만, 새로운 호스트가 등장할 때 성능이 향상될 수 있습니다.
* **memcap: 32mb**
  + 설명: 호스트 테이블에 할당된 메모리의 최대 크기를 설정합니다. 여기서는 32MB로 설정되어 있으며, 네트워크에서 많은 수의 호스트를 추적할 경우 더 큰 값을 설정할 수 있습니다.

### **IP 페어 테이블 설정 (ippair)**

IP 페어 테이블은 네트워크에서 서로 통신하는 IP 주소 쌍(IP Pair)을 추적합니다. 이를 통해 특정 IP 쌍 간의 트래픽 패턴을 분석하거나, 특정 IP 쌍에 대한 xbits 트래킹 등을 수행할 수 있습니다. 이 설정은 주석 처리되어 있어서 기본값이 적용되며, 필요할 경우 활성화하여 사용할 수 있습니다.

* **hash-size: 4096**
  + 설명: IP 페어 테이블의 해시 테이블 크기를 지정합니다. 해시 테이블은 IP 쌍 정보를 저장하고 빠르게 검색하는 데 사용됩니다.
* **prealloc: 1000**
  + 설명: 시작 시 미리 할당할 IP 페어 엔트리의 수를 지정합니다. 이 값도 네트워크 환경에 따라 조정할 수 있습니다.
* **memcap: 32mb**
  + 설명: IP 페어 테이블에 할당된 메모리의 최대 크기를 설정합니다. 여기서는 32MB로 설정되어 있으며, 네트워크 상의 많은 IP 쌍을 추적할 경우 더 큰 값을 설정할 수 있습니다.

이러한 테이블 설정은 Suricata의 메모리 사용량과 성능에 큰 영향을 미칩니다. 예를 들어, 대규모 네트워크에서는 hash-size나 memcap 값을 더 크게 설정하여 더 많은 호스트와 IP 쌍을 추적할 수 있도록 해야 합니다. 반대로 소규모 네트워크에서는 불필요한 메모리 낭비를 줄이기 위해 값을 작게 설정하는 것이 좋습니다.

네트워크 환경에 맞게 이 설정을 조정하고 최적화하면 Suricata의 성능과 효율성이 크게 향상될 수 있습니다.

# Decoder settings

decoder:

# Teredo decoder is known to not be completely accurate

# as it will sometimes detect non-teredo as teredo.

teredo:

enabled: true

# ports to look for Teredo. Max 4 ports. If no ports are given, or

# the value is set to 'any', Teredo detection runs on \_all\_ UDP packets.

ports: $TEREDO\_PORTS # syntax: '[3544, 1234]' or '3533' or 'any'.

# VXLAN decoder is assigned to up to 4 UDP ports. By default only the

# IANA assigned port 4789 is enabled.

vxlan:

enabled: true

ports: $VXLAN\_PORTS # syntax: '[8472, 4789]' or '4789'.

# Geneve decoder is assigned to up to 4 UDP ports. By default only the

# IANA assigned port 6081 is enabled.

geneve:

enabled: true

ports: $GENEVE\_PORTS # syntax: '[6081, 1234]' or '6081'.

# maximum number of decoder layers for a packet

# max-layers: 16

Suricata의 디코더(Decoder) 설정을 정의하고 있으며, 네트워크 패킷에서 다양한 프로토콜을 인식하고 분석하기 위한 규칙들을 포함하고 있습니다. 디코더는 각 프로토콜의 특성을 바탕으로 패킷을 해석하여 보다 정확한 트래픽 분석을 가능하게 합니다. 본 보고서에서는 Teredo, VXLAN, Geneve 디코더 설정 및 최대 디코더 레이어에 대한 주요 설정 사항을 기술합니다.

#### **1. Teredo 디코더 설정**

Teredo는 IPv4 네트워크를 통해 IPv6 패킷을 전송하기 위한 터널링 프로토콜입니다. 다음과 같은 설정이 적용되었습니다:

* **활성화 여부**: enabled: true  
  Teredo 디코더가 활성화되어 있으며, 이를 통해 Teredo 패킷을 해석할 수 있도록 설정되어 있습니다.
* **분석 포트**: ports: $TEREDO\_PORTS  
  Teredo 패킷을 분석할 포트는 $TEREDO\_PORTS 환경 변수에 따라 지정됩니다. 이 변수는 사용자 정의 값을 포함할 수 있으며, 예를 들어 [3544, 1234]와 같이 포트를 지정할 수 있습니다. 만약 포트가 any로 설정될 경우, 모든 UDP 패킷이 Teredo 디코더에 의해 처리됩니다.

#### **2. VXLAN 디코더 설정**

VXLAN은 가상 네트워크 환경에서 트래픽 오버레이를 위해 사용되는 프로토콜입니다. 다음과 같은 설정이 적용되었습니다:

* **활성화 여부**: enabled: true  
  VXLAN 디코더가 활성화되어 있으며, VXLAN 패킷을 해석할 수 있도록 설정되어 있습니다.
* **분석 포트**: ports: $VXLAN\_PORTS  
  VXLAN 패킷을 분석할 포트는 $VXLAN\_PORTS 환경 변수에 따라 지정됩니다. 예를 들어, [8472, 4789]와 같이 포트를 지정할 수 있으며, 이 경우 VXLAN 디코더는 8472와 4789 포트에서 들어오는 패킷을 분석하게 됩니다.

#### **3. Geneve 디코더 설정**

Geneve는 VXLAN과 유사한 오버레이 네트워크 프로토콜로, 보다 유연하고 다양한 오버레이 네트워크 기능을 제공합니다. 다음과 같은 설정이 적용되었습니다:

* **활성화 여부**: enabled: true  
  Geneve 디코더가 활성화되어 있으며, Geneve 패킷을 해석할 수 있도록 설정되어 있습니다.
* **분석 포트**: ports: $GENEVE\_PORTS  
  Geneve 패킷을 분석할 포트는 $GENEVE\_PORTS 환경 변수에 따라 지정됩니다. 예를 들어, [6081, 1234]와 같이 포트를 지정할 수 있으며, 이 경우 Geneve 디코더는 6081과 1234 포트에서 들어오는 패킷을 분석하게 됩니다.

#### **4. 최대 디코더 레이어 설정**

* **최대 디코더 레이어 수**: max-layers: 16  
  본 설정에서는 패킷 분석 시 최대 16개의 디코더 레이어를 허용합니다. 이는 여러 터널링 프로토콜이 중첩된 경우에도 디코더가 패킷을 정확하게 분석할 수 있도록 합니다.

본 설정 파일은 Teredo, VXLAN, Geneve와 같은 다양한 프로토콜의 디코더를 활성화하여 네트워크 패킷을 효과적으로 분석할 수 있도록 구성되었습니다. 이를 통해 Suricata 엔진이 보다 다양한 네트워크 환경에서 정확한 트래픽 분석을 수행할 수 있을 것으로 기대됩니다.

##

## Performance tuning and profiling

##

# The detection engine builds internal groups of signatures. The engine

# allows us to specify the profile to use for them, to manage memory in an

# efficient way keeping good performance. For the profile keyword you

# can use the words "low", "medium", "high" or "custom". If you use custom,

# make sure to define the values in the "custom-values" section.

# Usually you would prefer medium/high/low.

#

# "sgh mpm-context", indicates how the staging should allot mpm contexts for

# the signature groups. "single" indicates the use of a single context for

# all the signature group heads. "full" indicates a mpm-context for each

# group head. "auto" lets the engine decide the distribution of contexts

# based on the information the engine gathers on the patterns from each

# group head.

#

# The option inspection-recursion-limit is used to limit the recursive calls

# in the content inspection code. For certain payload-sig combinations, we

# might end up taking too much time in the content inspection code.

# If the argument specified is 0, the engine uses an internally defined

# default limit. When a value is not specified, there are no limits on the recursion.

detect:

profile: medium

custom-values:

toclient-groups: 3

toserver-groups: 25

sgh-mpm-context: auto

inspection-recursion-limit: 3000

# If set to yes, the loading of signatures will be made after the capture

# is started. This will limit the downtime in IPS mode.

#delayed-detect: yes

prefilter:

# default prefiltering setting. "mpm" only creates MPM/fast\_pattern

# engines. "auto" also sets up prefilter engines for other keywords.

# Use --list-keywords=all to see which keywords support prefiltering.

default: mpm

# the grouping values above control how many groups are created per

# direction. Port whitelisting forces that port to get its own group.

# Very common ports will benefit, as well as ports with many expensive

# rules.

grouping:

#tcp-whitelist: 53, 80, 139, 443, 445, 1433, 3306, 3389, 6666, 6667, 8080

#udp-whitelist: 53, 135, 5060

profiling:

# Log the rules that made it past the prefilter stage, per packet

# default is off. The threshold setting determines how many rules

# must have made it past pre-filter for that rule to trigger the

# logging.

#inspect-logging-threshold: 200

grouping:

dump-to-disk: false

include-rules: false # very verbose

include-mpm-stats: false

Suricata의 탐지 엔진 성능 최적화와 프로파일링을 위한 설정을 포함하고 있으며, 내부적으로 시그니처 그룹을 효율적으로 관리하여 메모리 사용량을 줄이고 성능을 향상시키기 위한 다양한 옵션들을 제공합니다. 이 문서에서는 탐지 프로파일, 프리필터링, 시그니처 그룹화 및 프로파일링 설정에 대해 기술합니다.

#### **1. 탐지 엔진 프로파일 설정**

탐지 엔진은 시그니처 그룹의 메모리 관리와 성능을 최적화하기 위해 탐지 프로파일을 설정합니다.

* **탐지 프로파일**: profile: medium  
  탐지 엔진은 medium 프로파일을 사용하여 메모리와 성능 간의 균형을 유지하도록 설정되어 있습니다.
* **사용자 정의 값**:
  + toclient-groups: 3  
    클라이언트로의 트래픽에 대해 3개의 시그니처 그룹이 생성됩니다.
  + toserver-groups: 25  
    서버로의 트래픽에 대해 25개의 시그니처 그룹이 생성됩니다.
* **MPM 컨텍스트 설정**: sgh-mpm-context: auto  
  엔진이 수집한 시그니처 패턴 정보에 따라 자동으로 MPM(Multi-Pattern Matcher) 컨텍스트를 할당합니다. 이 설정은 성능 최적화에 유리한 값을 자동으로 선택하게 합니다.
* **검사 재귀 제한**: inspection-recursion-limit: 3000  
  탐지 엔진의 콘텐츠 검사 코드에서 발생할 수 있는 과도한 재귀 호출을 방지하기 위해 재귀 호출 한도를 3000으로 설정합니다. 이를 통해 특정 페이로드 및 시그니처 조합으로 인해 발생할 수 있는 성능 저하를 방지할 수 있습니다.

#### **2. 프리필터링 설정**

프리필터링은 탐지 엔진이 트래픽을 분석하기 전에 시그니처를 미리 필터링하여 성능을 최적화합니다.

* **기본 프리필터링 설정**: default: mpm  
  MPM/빠른 패턴 매칭 엔진만을 생성하여 프리필터링을 수행합니다.  
  auto로 설정 시, 특정 키워드에 대해서도 프리필터 엔진을 설정합니다.

#### **3. 시그니처 그룹화 설정**

시그니처 그룹화 설정은 탐지 엔진이 특정 트래픽에 대해 효율적으로 시그니처를 적용하도록 그룹을 생성하는 방식입니다.

* **그룹화 설정**:
  + tcp-whitelist 및 udp-whitelist를 통해 자주 사용되는 포트를 지정할 수 있습니다.  
    설정된 포트에 대해 개별 그룹을 생성하여 성능을 최적화합니다.
    - TCP 화이트리스트 예시: 53, 80, 139, 443, 445, 1433, 3306, 3389, 6666, 6667, 8080
    - UDP 화이트리스트 예시: 53, 135, 5060

#### **4. 프로파일링 설정**

탐지 엔진의 프로파일링 설정은 시그니처가 패킷에 대해 어떻게 동작하는지에 대한 성능 데이터를 수집할 수 있도록 합니다.

* **프로파일링 설정**:
  + grouping의 dump-to-disk: false  
    그룹화된 시그니처 데이터는 디스크로 덤프되지 않습니다.
  + include-rules: false  
    매우 상세한 정보를 포함하지 않습니다.
  + include-mpm-stats: false  
    MPM 관련 통계를 포함하지 않습니다.

본 설정 파일은 Suricata의 탐지 엔진 성능을 최적화하고, 필요 시 프로파일링을 통해 성능 데이터를 수집할 수 있도록 구성되었습니다. 이를 통해 메모리 관리 효율성을 극대화하고, 다양한 트래픽 환경에서도 안정적인 성능을 유지할 수 있을 것으로 기대됩니다.

# Select the multi pattern algorithm you want to run for scan/search the

# in the engine.

#

# The supported algorithms are:

# "ac" - Aho-Corasick, default implementation

# "ac-bs" - Aho-Corasick, reduced memory implementation

# "ac-ks" - Aho-Corasick, "Ken Steele" variant

# "hs" - Hyperscan, available when built with Hyperscan support

#

# The default mpm-algo value of "auto" will use "hs" if Hyperscan is

# available, "ac" otherwise.

#

# The mpm you choose also decides the distribution of mpm contexts for

# signature groups, specified by the conf - "detect.sgh-mpm-context".

# Selecting "ac" as the mpm would require "detect.sgh-mpm-context"

# to be set to "single", because of ac's memory requirements, unless the

# ruleset is small enough to fit in memory, in which case one can

# use "full" with "ac". The rest of the mpms can be run in "full" mode.

mpm-algo: auto

# Select the matching algorithm you want to use for single-pattern searches.

#

# Supported algorithms are "bm" (Boyer-Moore) and "hs" (Hyperscan, only

# available if Suricata has been built with Hyperscan support).

#

# The default of "auto" will use "hs" if available, otherwise "bm".

spm-algo: auto

Suricata 탐지 엔진에서 패턴 매칭을 최적화하기 위해 사용되는 멀티 패턴 매칭(MPM) 및 싱글 패턴 매칭(SPM) 알고리즘의 설정 사항을 설명합니다. 각 알고리즘의 특성과 사용 목적에 대해 기술하고, 최적의 설정값을 제안합니다.

#### **1. 멀티 패턴 매칭 알고리즘 (MPM)**

MPM은 여러 시그니처 패턴을 동시에 검색하여 탐지 엔진의 성능을 최적화하는 알고리즘입니다. Suricata에서 지원되는 MPM 알고리즘은 다음과 같습니다:

* **"ac"**: Aho-Corasick 알고리즘으로, 기본 구현 방식입니다.
* **"ac-bs"**: Aho-Corasick의 메모리 사용량을 줄인 버전입니다.
* **"ac-ks"**: Aho-Corasick의 "Ken Steele" 변형 버전으로, 성능 개선을 목표로 합니다.
* **"hs"**: Hyperscan 알고리즘으로, Hyperscan 라이브러리가 지원되는 경우 사용할 수 있습니다.

**기본 설정**: mpm-algo: auto  
기본적으로 "auto" 옵션은 Hyperscan이 사용 가능한 경우 "hs" 알고리즘을 사용하며, 그렇지 않은 경우 "ac" 알고리즘을 사용합니다. Hyperscan은 높은 성능을 제공하므로, 사용 가능한 경우 이를 선택하는 것이 바람직합니다.

**참고 사항**:  
detect.sgh-mpm-context 설정은 선택된 MPM 알고리즘에 따라 조정되어야 합니다. 예를 들어, "ac" 알고리즘을 사용할 경우 메모리 요구 사항으로 인해 single 모드로 설정하는 것이 일반적이며, 시그니처 그룹이 메모리에 모두 적재 가능한 경우 full 모드를 사용할 수 있습니다.

#### **2. 싱글 패턴 매칭 알고리즘 (SPM)**

SPM은 단일 패턴 검색 시 사용되는 알고리즘입니다. Suricata에서 지원되는 SPM 알고리즘은 다음과 같습니다:

* **"bm"**: Boyer-Moore 알고리즘으로, 단일 패턴 검색에 효율적입니다.
* **"hs"**: Hyperscan 알고리즘으로, Hyperscan 라이브러리가 지원되는 경우 사용할 수 있습니다.

**기본 설정**: spm-algo: auto  
기본적으로 "auto" 옵션은 Hyperscan이 사용 가능한 경우 "hs" 알고리즘을 사용하며, 그렇지 않은 경우 "bm" 알고리즘을 사용합니다. Boyer-Moore는 Hyperscan이 사용 불가능한 환경에서 효율적인 성능을 제공하지만, Hyperscan이 지원되는 경우 이를 선택하는 것이 더욱 빠르고 효율적입니다.

#### **3. 권장 설정**

현재 설정된 mpm-algo와 spm-algo 모두 auto로 설정되어 있어, Hyperscan 라이브러리가 사용 가능한 경우 이를 활용하도록 구성되어 있습니다. 이는 최적의 성능을 보장하는 설정으로 판단됩니다. 다만, 다음 사항들을 고려하여 필요시 추가적인 조정을 할 수 있습니다:

1. **메모리 사용량 관리**: Hyperscan의 메모리 사용량이 큰 경우, 메모리 제한 환경에서는 "ac-bs" 또는 "ac-ks"와 같은 메모리 절약형 알고리즘을 고려할 수 있습니다.
2. **시그니처 그룹 컨텍스트 설정**: Hyperscan 사용 시 detect.sgh-mpm-context를 full로 설정하여 모든 시그니처 그룹에 대해 최적화된 탐지를 제공할 수 있습니다.
3. **성능 테스트**: 실제 트래픽 환경에서 탐지 성능을 테스트하여 특정 알고리즘이 더 높은 성능을 제공하는지 확인 후 최종 설정을 조정하는 것이 좋습니다.

현재 설정된 mpm-algo: auto와 spm-algo: auto는 Hyperscan이 사용 가능한 환경에서 최상의 성능을 제공하도록 구성되어 있으며, 메모리 및 성능 간의 균형을 고려한 최적의 설정입니다. 탐지 환경에 따라 세부 설정을 조정하여 더 나은 성능을 확보할 수 있습니다.

# Suricata is multi-threaded. Here the threading can be influenced.

threading:

set-cpu-affinity: no

# Tune cpu affinity of threads. Each family of threads can be bound

# to specific CPUs.

#

# These 2 apply to the all runmodes:

# management-cpu-set is used for flow timeout handling, counters

# worker-cpu-set is used for 'worker' threads

#

# Additionally, for autofp these apply:

# receive-cpu-set is used for capture threads

# verdict-cpu-set is used for IPS verdict threads

#

cpu-affinity:

- management-cpu-set:

cpu: [ 0 ] # include only these CPUs in affinity settings

- receive-cpu-set:

cpu: [ 0 ] # include only these CPUs in affinity settings

- worker-cpu-set:

cpu: [ "all" ]

mode: "exclusive"

# Use explicitly 3 threads and don't compute number by using

# detect-thread-ratio variable:

# threads: 3

prio:

low: [ 0 ]

medium: [ "1-2" ]

high: [ 3 ]

default: "medium"

#- verdict-cpu-set:

# cpu: [ 0 ]

# prio:

# default: "high"

#

# By default Suricata creates one "detect" thread per available CPU/CPU core.

# This setting allows controlling this behaviour. A ratio setting of 2 will

# create 2 detect threads for each CPU/CPU core. So for a dual core CPU this

# will result in 4 detect threads. If values below 1 are used, less threads

# are created. So on a dual core CPU a setting of 0.5 results in 1 detect

# thread being created. Regardless of the setting at a minimum 1 detect

# thread will always be created.

#

detect-thread-ratio: 1.0

#

# By default, the per-thread stack size is left to its default setting. If

# the default thread stack size is too small, use the following configuration

# setting to change the size. Note that if any thread's stack size cannot be

# set to this value, a fatal error occurs.

#

# Generally, the per-thread stack-size should not exceed 8MB.

#stack-size: 8mb

Suricata 탐지 엔진의 멀티 스레딩 설정에 대해 설명합니다. 스레드 설정은 CPU 자원 활용도를 최적화하고 성능을 개선하기 위해 중요한 요소입니다. 아래에서는 각 설정 항목과 그 의미, 최적의 설정 방안을 제시합니다.

#### **1. CPU Affinity 설정**

Suricata는 스레드를 특정 CPU에 바인딩하여 성능을 최적화할 수 있습니다. CPU affinity 설정은 다음과 같은 구성 요소로 나뉩니다:

* **management-cpu-set**: 흐름 타임아웃 처리 및 카운터 관리를 위한 스레드의 CPU 설정입니다.
* **receive-cpu-set**: 캡처 스레드를 위한 CPU 설정입니다.
* **worker-cpu-set**: 작업자 스레드를 위한 CPU 설정으로, 기본적으로 모든 CPU를 사용할 수 있도록 설정되어 있습니다.

**현재 설정**:

set-cpu-affinity: no

cpu-affinity:

- management-cpu-set:

cpu: [ 0 ]

- receive-cpu-set:

cpu: [ 0 ]

- worker-cpu-set:

cpu: [ "all" ]

mode: "exclusive"

이 설정은 관리 스레드와 수신 스레드를 CPU 0에 바인딩하고, 작업자 스레드는 모든 CPU를 독점적으로 사용하도록 구성되어 있습니다. 이를 통해 관리와 수신 처리에 필요한 리소스를 제한하면서 작업자 스레드는 최대한의 성능을 발휘할 수 있습니다.

#### **2. 탐지 스레드 설정**

기본적으로 Suricata는 사용 가능한 CPU 코어마다 하나의 탐지 스레드를 생성합니다. detect-thread-ratio 설정을 통해 이 수를 조정할 수 있습니다. 예를 들어, 비율을 2로 설정하면 듀얼 코어 CPU에서 총 4개의 탐지 스레드를 생성합니다. 반대로, 1보다 작은 값을 설정하면 스레드 수가 감소합니다.

**현재 설정**:

detect-thread-ratio: 1.0

이 설정은 각 CPU 코어에 대해 하나의 탐지 스레드를 생성하는 기본값을 유지합니다. 이는 안정적인 성능을 보장하는데 적합합니다.

#### **3. 스레드 스택 크기 설정**

Suricata의 스레드 스택 크기는 기본값으로 설정되어 있습니다. 기본값이 너무 작으면 스레드가 비정상적으로 종료될 수 있습니다. 필요시 스택 크기를 조정할 수 있지만, 일반적으로 스택 크기는 8MB를 초과하지 않는 것이 권장됩니다.

**현재 설정**:

#stack-size: 8mb

현재 주석 처리되어 있어 기본값이 적용됩니다. 필요시 이 값을 조정하여 스레드의 안정성을 높일 수 있습니다.

#### 

Suricata의 스레딩 설정은 성능과 리소스 사용의 균형을 맞추는 데 중요한 역할을 합니다. 현재 설정은 CPU affinity를 통해 관리 및 수신 스레드에 필요한 리소스를 최적화하고, 탐지 스레드는 CPU 코어당 하나로 유지하여 안정적인 성능을 보장합니다. 필요시 스레드 스택 크기를 조정하여 안정성을 더욱 강화할 수 있습니다. 추가적인 성능 개선을 위해 스레드 수를 조정하거나 CPU affinity 설정을 세부적으로 조정할 수 있습니다.

# Luajit has a strange memory requirement, its 'states' need to be in the

# first 2G of the process' memory.

#

# 'luajit.states' is used to control how many states are preallocated.

# State use: per detect script: 1 per detect thread. Per output script: 1 per

# script.

luajit:

states: 128

# Profiling settings. Only effective if Suricata has been built with

# the --enable-profiling configure flag.

#

profiling:

# Run profiling for every X-th packet. The default is 1, which means we

# profile every packet. If set to 1024, one packet is profiled for every

# 1024 received. The sample rate must be a power of 2.

#sample-rate: 1024

# rule profiling

rules:

# Profiling can be disabled here, but it will still have a

# performance impact if compiled in.

enabled: yes

filename: rule\_perf.log

append: yes

# Set active to yes to enable rules profiling at start

# if set to no (default), the rules profiling will have to be started

# via unix socket commands.

#active:no

# Sort options: ticks, avgticks, checks, matches, maxticks

# If commented out all the sort options will be used.

#sort: avgticks

# Limit the number of sids for which stats are shown at exit (per sort).

limit: 10

# output to json

json: yes

# per keyword profiling

keywords:

enabled: yes

filename: keyword\_perf.log

append: yes

prefilter:

enabled: yes

filename: prefilter\_perf.log

append: yes

# per rulegroup profiling

rulegroups:

enabled: yes

filename: rule\_group\_perf.log

append: yes

# packet profiling

packets:

# Profiling can be disabled here, but it will still have a

# performance impact if compiled in.

enabled: yes

filename: packet\_stats.log

append: yes

# per packet csv output

csv:

# Output can be disabled here, but it will still have a

# performance impact if compiled in.

enabled: no

filename: packet\_stats.csv

# profiling of locking. Only available when Suricata was built with

# --enable-profiling-locks.

locks:

enabled: no

filename: lock\_stats.log

append: yes

pcap-log:

enabled: no

filename: pcaplog\_stats.log

append: yes

Suricata의 프로파일링 설정에 대한 개요와 각 구성 요소의 기능, 성능 최적화 방안에 대해 설명합니다. 프로파일링은 성능 분석 및 조정을 위한 필수 도구이며, 설정된 구성에 따라 패킷 처리의 효율성을 높일 수 있습니다.

#### **1. LuaJIT 상태 설정**

Suricata는 LuaJIT를 사용하여 스크립트 실행 성능을 향상시킵니다. LuaJIT의 'states'는 프로세스의 첫 2G 메모리 내에 위치해야 하며, 각 탐지 스레드 및 출력 스크립트에 대해 하나의 상태가 필요합니다.

**현재 설정**:

luajit:

states: 128

이 설정은 최대 128개의 LuaJIT 상태를 미리 할당합니다. 이는 여러 스크립트를 동시에 처리할 수 있도록 충분한 메모리를 확보하는 데 도움이 됩니다.

#### **2. 프로파일링 설정**

Suricata에서 프로파일링을 활성화하면, 패킷 처리와 관련된 다양한 지표를 수집할 수 있습니다. 프로파일링은 --enable-profiling 플래그로 빌드되어야 하며, 다음과 같은 구성 요소가 포함됩니다:

**샘플링 비율**: 기본적으로 모든 패킷을 프로파일링하지만, 샘플링 비율을 설정하여 특정 패킷만 프로파일링할 수 있습니다. 샘플링 비율은 2의 거듭제곱으로 설정해야 합니다.

#sample-rate: 1024

**규칙 프로파일링**: 규칙별 성능을 기록할 수 있으며, 결과는 rule\_perf.log에 저장됩니다. 활성화된 경우, 규칙 프로파일링이 시작 시 자동으로 실행됩니다.

rules:

enabled: yes

filename: rule\_perf.log

append: yes

active: no

limit: 10

json: yes

* **키워드 프로파일링**: 키워드 성능 통계를 keyword\_perf.log에 기록합니다.
* **프리필터 프로파일링**: 프리필터링 과정에서의 성능을 prefilter\_perf.log에 기록합니다.
* **규칙 그룹 프로파일링**: 각 규칙 그룹의 성능을 rule\_group\_perf.log에 기록합니다.

**패킷 프로파일링**: 패킷 처리 성능 통계를 packet\_stats.log에 기록하며, CSV 형식으로의 출력도 지원합니다.

packets:

enabled: yes

filename: packet\_stats.log

append: yes

csv:

enabled: no

filename: packet\_stats.csv

* **락 프로파일링**: 락 관련 통계를 수집할 수 있으나, 이 기능은 --enable-profiling-locks로 빌드되어야 합니다.
* **PCAP 로그**: PCAP 통계를 기록하는 옵션입니다.

#### 

Suricata의 프로파일링 설정은 패킷 처리의 성능을 분석하고 최적화하는 데 중요한 역할을 합니다. 현재 설정은 LuaJIT 상태 수를 충분히 할당하여 스크립트 실행 성능을 보장하고, 다양한 프로파일링 옵션을 통해 성능 통계를 수집할 수 있도록 구성되어 있습니다. 각 프로파일링 항목은 필요한 경우 활성화할 수 있으며, 최적의 성능 분석을 위해 JSON 출력 및 추가 로그 파일 생성을 고려할 수 있습니다. 이러한 설정을 통해 시스템의 성능을 지속적으로 모니터링하고 개선할 수 있습니다.

##

## Netfilter integration

##

# When running in NFQ inline mode, it is possible to use a simulated

# non-terminal NFQUEUE verdict.

# This permits sending all needed packet to Suricata via this rule:

# iptables -I FORWARD -m mark ! --mark $MARK/$MASK -j NFQUEUE

# And below, you can have your standard filtering ruleset. To activate

# this mode, you need to set mode to 'repeat'

# If you want a packet to be sent to another queue after an ACCEPT decision

# set the mode to 'route' and set next-queue value.

# On Linux >= 3.1, you can set batchcount to a value > 1 to improve performance

# by processing several packets before sending a verdict (worker runmode only).

# On Linux >= 3.6, you can set the fail-open option to yes to have the kernel

# accept the packet if Suricata is not able to keep pace.

# bypass mark and mask can be used to implement NFQ bypass. If bypass mark is

# set then the NFQ bypass is activated. Suricata will set the bypass mark/mask

# on packet of a flow that need to be bypassed. The Netfilter ruleset has to

# directly accept all packets of a flow once a packet has been marked.

nfq:

# mode: accept

# repeat-mark: 1

# repeat-mask: 1

# bypass-mark: 1

# bypass-mask: 1

# route-queue: 2

# batchcount: 20

# fail-open: yes

#nflog support

nflog:

# netlink multicast group

# (the same as the iptables --nflog-group param)

# Group 0 is used by the kernel, so you can't use it

- group: 2

# netlink buffer size

buffer-size: 18432

# put default value here

- group: default

# set number of packets to queue inside kernel

qthreshold: 1

# set the delay before flushing packet in the kernel's queue

qtimeout: 100

# netlink max buffer size

max-size: 20000

Suricata의 Netfilter 통합 기능에 대한 설정 내용을 설명합니다. Netfilter는 Linux 커널의 방화벽 프레임워크로, Suricata와 결합하여 네트워크 트래픽을 보다 효과적으로 모니터링하고 처리할 수 있습니다.

#### **1. NFQ (Netfilter Queue) 모드 설정**

NFQ 모드는 Suricata가 패킷을 처리하는 방식과 관련됩니다. NFQUEUE를 사용하여 패킷을 Suricata로 전달하고, 처리 후 다시 Netfilter 규칙으로 전달됩니다. 주요 옵션들은 다음과 같습니다:

* **mode**: 패킷을 처리하는 방식 결정. 예를 들어, accept 모드에서는 패킷을 허용하고, repeat 모드에서는 비단말 NFQUEUE 판정을 시뮬레이션합니다.
* **repeat-mark 및 repeat-mask**: 패킷의 반복 처리를 위한 마크와 마스크 설정.
* **bypass-mark 및 bypass-mask**: NFQ 우회 기능을 활성화하며, 해당 마크가 설정되면 해당 패킷 흐름은 우회처리됩니다.
* **batchcount**: 여러 패킷을 묶어서 처리하여 성능을 개선할 수 있는 옵션입니다. Linux 커널 3.1 이상에서 지원됩니다.
* **fail-open**: Suricata가 성능에 따라 패킷을 처리하지 못할 경우, 커널이 패킷을 허용하도록 하는 옵션입니다. Linux 3.6 이상에서 지원됩니다.

**현재 설정 예시**:

nfq:

# mode: accept

# repeat-mark: 1

# repeat-mask: 1

# bypass-mark: 1

# bypass-mask: 1

# route-queue: 2

# batchcount: 20

# fail-open: yes

NFQ 기능이 활성화되지 않은 상태로 주석 처리되어 있으며, 설정을 통해 다양한 네트워크 상황에 대응할 수 있습니다.

#### **2. NFLOG 설정**

NFLOG는 Netfilter의 로깅 인터페이스로, 네트워크 트래픽에 대한 정보를 캡처하고 Suricata에서 처리할 수 있게 합니다. 주요 옵션들은 다음과 같습니다:

* **group**: iptables에서 사용된 --nflog-group과 일치하는 멀티캐스트 그룹 번호입니다. 그룹 0은 커널에서 사용하므로 사용할 수 없습니다.
* **buffer-size**: Suricata로 전달되는 데이터의 버퍼 크기 설정.
* **qthreshold**: 커널 내에서 대기 중인 패킷 수의 임계값.
* **qtimeout**: 커널 큐에 대기 중인 패킷이 플러시되기 전의 지연 시간(밀리초 단위).
* **max-size**: Netlink 최대 버퍼 크기.

**현재 설정 예시**:

nflog:

- group: 2

buffer-size: 18432

- group: default

qthreshold: 1

qtimeout: 100

max-size: 20000

설정된 그룹은 2이며, 기본적인 버퍼 크기와 큐 타임아웃이 지정되었습니다.

Suricata와 Netfilter의 통합을 통해 트래픽 처리 및 모니터링의 효율성을 극대화할 수 있습니다. NFQ 모드는 고성능 네트워크 환경에서 패킷 처리를 최적화할 수 있도록 설계되었으며, NFLG는 네트워크 로깅을 통해 네트워크 상태를 추적하는 데 중요한 역할을 합니다.

##

## Advanced Capture Options

##

# General settings affecting packet capture

capture:

# disable NIC offloading. It's restored when Suricata exits.

# Enabled by default.

#disable-offloading: false

#

# disable checksum validation. Same as setting '-k none' on the

# command-line.

#checksum-validation: none

# Netmap support

#

# Netmap operates with NIC directly in driver, so you need FreeBSD 11+ which has

# built-in Netmap support or compile and install the Netmap module and appropriate

# NIC driver for your Linux system.

# To reach maximum throughput disable all receive-, segmentation-,

# checksum- offloading on your NIC (using ethtool or similar).

# Disabling TX checksum offloading is \*required\* for connecting OS endpoint

# with NIC endpoint.

# You can find more information at https://github.com/luigirizzo/netmap

#

netmap:

# To specify OS endpoint add plus sign at the end (e.g. "eth0+")

- interface: eth2

# Number of capture threads. "auto" uses number of RSS queues on interface.

# Warning: unless the RSS hashing is symmetrical, this will lead to

# accuracy issues.

#threads: auto

# You can use the following variables to activate netmap tap or IPS mode.

# If copy-mode is set to ips or tap, the traffic coming to the current

# interface will be copied to the copy-iface interface. If 'tap' is set, the

# copy is complete. If 'ips' is set, the packet matching a 'drop' action

# will not be copied.

# To specify the OS as the copy-iface (so the OS can route packets, or forward

# to a service running on the same machine) add a plus sign at the end

# (e.g. "copy-iface: eth0+"). Don't forget to set up a symmetrical eth0+ -> eth0

# for return packets. Hardware checksumming must be \*off\* on the interface if

# using an OS endpoint (e.g. 'ifconfig eth0 -rxcsum -txcsum -rxcsum6 -txcsum6' for FreeBSD

# or 'ethtool -K eth0 tx off rx off' for Linux).

#copy-mode: tap

#copy-iface: eth3

# Set to yes to disable promiscuous mode

# disable-promisc: no

# Choose checksum verification mode for the interface. At the moment

# of the capture, some packets may have an invalid checksum due to

# the checksum computation being offloaded to the network card.

# Possible values are:

# - yes: checksum validation is forced

# - no: checksum validation is disabled

# - auto: Suricata uses a statistical approach to detect when

# checksum off-loading is used.

# Warning: 'checksum-validation' must be set to yes to have any validation

#checksum-checks: auto

# BPF filter to apply to this interface. The pcap filter syntax apply here.

#bpf-filter: port 80 or udp

#- interface: eth3

#threads: auto

#copy-mode: tap

#copy-iface: eth2

# Put default values here

- interface: default

Suricata의 고급 패킷 캡처 옵션에 대한 설정 내용을 설명합니다. 이 설정은 네트워크 인터페이스에서 데이터를 캡처하고 분석하는 방식을 세부적으로 조정할 수 있습니다.

#### **1. Capture 일반 설정**

* **NIC 오프로드 비활성화 (disable-offloading)**:
  + 기본적으로 NIC의 오프로드 기능을 비활성화하는 설정이 활성화되어 있습니다. NIC의 하드웨어 오프로드 기능을 비활성화하면 패킷 처리의 정확도가 향상될 수 있지만, 성능 저하가 발생할 수 있습니다.

#disable-offloading: false

* **체크섬 검증 비활성화 (checksum-validation)**:
  + 체크섬 검증을 비활성화할 수 있습니다. 체크섬 검증이 활성화되면 네트워크 패킷의 무결성을 보장하지만, 성능에 영향을 줄 수 있습니다.

#checksum-validation: none

#### **2. Netmap 지원**

**Netmap**은 네트워크 인터페이스와 직접 상호작용하며 패킷 처리 성능을 극대화하기 위한 모드입니다. FreeBSD 11+ 또는 Netmap 모듈을 설치한 Linux 시스템에서 사용할 수 있습니다. Netmap을 통해 직접 NIC와 작업하면 높은 처리량을 제공할 수 있습니다.

주요 설정 옵션:

* **interface**: 캡처할 인터페이스를 지정합니다. 예시에서는 eth2가 사용되고 있습니다.
* **threads**: 캡처 쓰레드 수를 지정하며, 기본적으로 "auto"는 RSS 큐의 수를 사용하여 자동으로 설정합니다. 성능을 위해 여러 쓰레드를 설정할 수 있지만, 정확도에 영향을 줄 수 있습니다.
* **copy-mode** 및 **copy-iface**: tap 또는 ips 모드를 사용하여 트래픽을 복사하는 인터페이스를 지정합니다. ips 모드는 'drop' 동작을 일으키는 패킷을 복사하지 않고, tap 모드는 패킷을 완전히 복사합니다.
* **disable-promisc**: 프라미스큐어스 모드를 비활성화할 수 있습니다. 이 모드가 활성화되면 인터페이스는 자신에게 직접 전송되지 않은 패킷도 수신할 수 있습니다.
* **checksum-checks**: 체크섬 검증을 수행할지 설정합니다. yes, no, auto 중 하나를 선택할 수 있으며, auto는 체크섬 오프로드가 사용되는지 감지하여 처리합니다.
* **bpf-filter**: 특정 패킷만 캡처할 수 있는 BPF 필터를 지정합니다. 예를 들어, 특정 포트나 프로토콜에 대해서만 트래픽을 모니터링할 수 있습니다.

netmap:

- interface: eth2

#threads: auto

#copy-mode: tap

#copy-iface: eth3

#disable-promisc: no

#checksum-checks: auto

#bpf-filter: port 80 or udp

- interface: default

Suricata의 고급 캡처 옵션을 사용하여 네트워크 트래픽 캡처 성능을 세부적으로 조정할 수 있습니다. **Netmap**을 통해 고성능 NIC와의 직접적인 상호작용이 가능하며, 다양한 설정을 통해 트래픽 처리 성능과 정확도 사이에서 균형을 맞출 수 있습니다.

# PF\_RING configuration: for use with native PF\_RING support

# for more info see http://www.ntop.org/products/pf\_ring/

pfring:

- interface: eth0

# Number of receive threads. If set to 'auto' Suricata will first try

# to use CPU (core) count and otherwise RSS queue count.

threads: auto

# Default clusterid. PF\_RING will load balance packets based on flow.

# All threads/processes that will participate need to have the same

# clusterid.

cluster-id: 99

# Default PF\_RING cluster type. PF\_RING can load balance per flow.

# Possible values are:

# - cluster\_flow: 6-tuple: <src ip, src\_port, dst ip, dst port, proto, vlan>

# - cluster\_inner\_flow: 6-tuple: <src ip, src port, dst ip, dst port, proto, vlan>

# - cluster\_inner\_flow\_2\_tuple: 2-tuple: <src ip, dst ip >

# - cluster\_inner\_flow\_4\_tuple: 4-tuple: <src ip, src port, dst ip, dst port >

# - cluster\_inner\_flow\_5\_tuple: 5-tuple: <src ip, src port, dst ip, dst port, proto >

# - cluster\_round\_robin (NOT RECOMMENDED)

cluster-type: cluster\_flow

# bpf filter for this interface

#bpf-filter: tcp

# If bypass is set then the PF\_RING hw bypass is activated, when supported

# by the network interface. Suricata will instruct the interface to bypass

# all future packets for a flow that need to be bypassed.

#bypass: yes

# Choose checksum verification mode for the interface. At the moment

# of the capture, some packets may have an invalid checksum due to

# the checksum computation being offloaded to the network card.

# Possible values are:

# - rxonly: only compute checksum for packets received by network card.

# - yes: checksum validation is forced

# - no: checksum validation is disabled

# - auto: Suricata uses a statistical approach to detect when

# checksum off-loading is used. (default)

# Warning: 'checksum-validation' must be set to yes to have any validation

#checksum-checks: auto

# Second interface

#- interface: eth1

# threads: 3

# cluster-id: 93

# cluster-type: cluster\_flow

# Put default values here

- interface: default

#threads: 2

Suricata의 **PF\_RING** 설정에 대해 설명하며, 이 기능은 고속 네트워크 패킷 캡처 및 처리 성능을 높이는 데 사용됩니다. PF\_RING은 특히 고속 네트워크에서 성능을 향상시키는 네이티브 지원을 제공합니다.

#### **1. PF\_RING 기본 설정**

* **interface**: 캡처할 인터페이스를 지정합니다. 여기서는 eth0 인터페이스가 사용됩니다. 추가적으로 eth1 등 다른 인터페이스를 추가로 설정할 수 있습니다.

**threads**: 수신 스레드 수를 설정합니다. 기본적으로 'auto'로 설정되어 있으며, 이는 CPU 코어 수 또는 RSS 큐 수에 따라 자동으로 스레드 수를 결정합니다.  
  
threads: auto

**cluster-id**: PF\_RING 클러스터 ID는 패킷을 흐름 기반으로 부하 분산하는 데 사용됩니다. 동일한 클러스터 ID를 공유하는 모든 스레드 및 프로세스가 동일한 패킷 플로우를 처리할 수 있게 설정됩니다.  
  
cluster-id: 99

* **cluster-type**: PF\_RING의 클러스터 유형을 정의합니다. 여기서는 **cluster\_flow**가 사용되며, 이는 6-튜플 (출발지 IP, 출발지 포트, 목적지 IP, 목적지 포트, 프로토콜, VLAN)에 기반하여 플로우별로 로드 밸런싱을 수행합니다.
  + 가능한 값들은 다음과 같습니다:
    - **cluster\_flow**: 기본 6-튜플 (src IP, src\_port, dst IP, dst\_port, proto, vlan).
    - **cluster\_inner\_flow**: 내부 플로우 6-튜플.
    - **cluster\_inner\_flow\_2\_tuple**: 2-튜플 (src IP, dst IP).
    - **cluster\_inner\_flow\_4\_tuple**: 4-튜플 (src IP, src\_port, dst IP, dst\_port).
    - **cluster\_inner\_flow\_5\_tuple**: 5-튜플 (src IP, src\_port, dst IP, dst\_port, proto).
    - **cluster\_round\_robin**: 라운드 로빈 방식 (권장하지 않음).

cluster-type: cluster\_flow

#### **2. 추가 설정 옵션**

**bpf-filter**: 인터페이스에 적용할 BPF(Berkeley Packet Filter) 필터를 지정합니다. 특정 프로토콜이나 포트에 대한 트래픽만 캡처하고 싶을 때 유용합니다. 예를 들어, TCP 트래픽만 캡처하고 싶다면 bpf-filter: tcp로 설정할 수 있습니다.  
yaml  
코드 복사  
#bpf-filter: tcp

**bypass**: PF\_RING 하드웨어 바이패스를 지원하는 경우, 이 설정을 활성화하여 네트워크 인터페이스가 바이패스할 플로우를 결정할 수 있습니다.  
  
#bypass: yes

* **checksum-checks**: 인터페이스에서 체크섬 검증 모드를 설정할 수 있습니다. 네트워크 카드가 체크섬을 처리하는 경우 유효하지 않은 체크섬을 가질 수 있기 때문에 체크섬 검증 모드가 중요합니다.
  + 가능한 값:
    - **rxonly**: 수신 패킷만 체크섬 검증.
    - **yes**: 강제로 체크섬 검증.
    - **no**: 체크섬 검증 비활성화.
    - **auto**: 통계적 접근을 사용하여 체크섬 오프로드 사용 여부를 감지.

#checksum-checks: auto

#### 

**PF\_RING** 설정을 통해 Suricata는 고속 네트워크에서 보다 효율적인 패킷 캡처 및 처리를 수행할 수 있습니다. 이 설정은 인터페이스 및 스레드 관리를 통해 네트워크 흐름을 분산하고, 체크섬 검증과 같은 추가 기능을 통해 정확도를 향상시킬 수 있습니다.

# For FreeBSD ipfw(8) divert(4) support.

# Please make sure you have ipfw\_load="YES" and ipdivert\_load="YES"

# in /etc/loader.conf or kldload'ing the appropriate kernel modules.

# Additionally, you need to have an ipfw rule for the engine to see

# the packets from ipfw. For Example:

#

# ipfw add 100 divert 8000 ip from any to any

#

# N.B. This example uses "8000" -- this number must mach the values

# you passed on the command line, i.e., -d 8000

#

ipfw:

# Reinject packets at the specified ipfw rule number. This config

# option is the ipfw rule number AT WHICH rule processing continues

# in the ipfw processing system after the engine has finished

# inspecting the packet for acceptance. If no rule number is specified,

# accepted packets are reinjected at the divert rule which they entered

# and IPFW rule processing continues. No check is done to verify

# this will rule makes sense so care must be taken to avoid loops in ipfw.

#

## The following example tells the engine to reinject packets

# back into the ipfw firewall AT rule number 5500:

#

# ipfw-reinjection-rule-number: 5500

**FreeBSD**에서 **ipfw**와 **divert** 서포트를 사용하기 위한 설정을 다룹니다. 해당 설정은 **Suricata**와 **ipfw(8)**, **divert(4)** 모듈 간의 통합을 목표로 합니다.

#### **1. 커널 모듈 로드 설정**

ipfw와 divert를 사용하기 위해선 먼저 관련 커널 모듈을 로드해야 합니다. 이 설정은 /etc/loader.conf 파일에 다음 두 줄을 추가하거나, 직접 **kldload** 명령어로 모듈을 로드할 수 있습니다:

ipfw\_load="YES"

ipdivert\_load="YES"

이 설정을 통해 **ipfw**와 **ipdivert** 모듈이 시스템 부팅 시 자동으로 로드됩니다.

#### **2. ipfw 룰 설정**

패킷을 **Suricata**로 리다이렉트하기 위해 **ipfw**에 특정 규칙을 추가해야 합니다. 예를 들어:

ipfw add 100 divert 8000 ip from any to any

이 규칙은 모든 트래픽을 divert 번호 **8000**으로 리다이렉트하며, Suricata는 이 번호를 통해 패킷을 수신합니다. 이 값은 **Suricata**를 시작할 때 명령줄에 전달한 값과 일치해야 합니다.

#### **3. ipfw 패킷 재삽입 규칙**

Suricata가 패킷을 처리한 후, **ipfw**의 규칙 처리 시스템에서 패킷이 다시 재삽입될 위치를 지정할 수 있습니다. 이를 위해 ipfw-reinjection-rule-number 설정을 사용합니다.

* 이 설정을 사용하지 않으면 패킷은 원래 **divert** 룰에서 처리된 후 다시 **ipfw** 시스템으로 돌아갑니다.

특정 규칙 번호로 패킷을 재삽입하려면 다음과 같이 설정할 수 있습니다:  
  
ipfw-reinjection-rule-number: 5500

* 이 예시는 Suricata가 패킷을 처리한 후 **ipfw**의 **5500번 규칙**에서 다시 패킷 처리를 계속하도록 설정하는 방법입니다.

#### **4. 주의 사항**

* **ipfw-reinjection-rule-number**를 설정할 때는 주의가 필요합니다. 잘못된 설정은 **ipfw** 내에서 무한 루프를 발생시킬 수 있기 때문에, 반드시 검토 후 설정을 적용해야 합니다.
* 패킷이 Suricata로 리다이렉트된 후 적절하게 재삽입되도록 **ipfw** 규칙 체계 내에서 정확한 흐름을 계획해야 합니다.

**FreeBSD**에서 **ipfw**와 **divert** 모듈을 사용해 Suricata와 통합할 때, 커널 모듈 로드, 패킷 리다이렉트, 재삽입 규칙 등을 적절히 설정하는 것이 중요합니다. 이를 통해 네트워크 트래픽을 효율적으로 분석 및 처리할 수 있습니다.

napatech:

# When use\_all\_streams is set to "yes" the initialization code will query

# the Napatech service for all configured streams and listen on all of them.

# When set to "no" the streams config array will be used.

#

# This option necessitates running the appropriate NTPL commands to create

# the desired streams prior to running Suricata.

#use-all-streams: no

# The streams to listen on when auto-config is disabled or when and threading

# cpu-affinity is disabled. This can be either:

# an individual stream (e.g. streams: [0])

# or

# a range of streams (e.g. streams: ["0-3"])

#

streams: ["0-3"]

# Stream stats can be enabled to provide fine grain packet and byte counters

# for each thread/stream that is configured.

#

enable-stream-stats: no

# When auto-config is enabled the streams will be created and assigned

# automatically to the NUMA node where the thread resides. If cpu-affinity

# is enabled in the threading section. Then the streams will be created

# according to the number of worker threads specified in the worker-cpu-set.

# Otherwise, the streams array is used to define the streams.

#

# This option is intended primarily to support legacy configurations.

#

# This option cannot be used simultaneously with either "use-all-streams"

# or "hardware-bypass".

#

auto-config: yes

# Enable hardware level flow bypass.

#

hardware-bypass: yes

# Enable inline operation. When enabled traffic arriving on a given port is

# automatically forwarded out its peer port after analysis by Suricata.

#

inline: no

# Ports indicates which Napatech ports are to be used in auto-config mode.

# these are the port IDs of the ports that will be merged prior to the

# traffic being distributed to the streams.

#

# When hardware-bypass is enabled the ports must be configured as a segment.

# specify the port(s) on which upstream and downstream traffic will arrive.

# This information is necessary for the hardware to properly process flows.

#

# When using a tap configuration one of the ports will receive inbound traffic

# for the network and the other will receive outbound traffic. The two ports on a

# given segment must reside on the same network adapter.

#

# When using a SPAN-port configuration the upstream and downstream traffic

# arrives on a single port. This is configured by setting the two sides of the

# segment to reference the same port. (e.g. 0-0 to configure a SPAN port on

# port 0).

#

# port segments are specified in the form:

# ports: [0-1,2-3,4-5,6-6,7-7]

#

# For legacy systems when hardware-bypass is disabled this can be specified in any

# of the following ways:

#

# a list of individual ports (e.g. ports: [0,1,2,3])

#

# a range of ports (e.g. ports: [0-3])

#

# "all" to indicate that all ports are to be merged together

# (e.g. ports: [all])

#

# This parameter has no effect if auto-config is disabled.

#

ports: [0-1,2-3]

# When auto-config is enabled the hashmode specifies the algorithm for

# determining to which stream a given packet is to be delivered.

# This can be any valid Napatech NTPL hashmode command.

#

# The most common hashmode commands are: hash2tuple, hash2tuplesorted,

# hash5tuple, hash5tuplesorted and roundrobin.

#

# See Napatech NTPL documentation other hashmodes and details on their use.

#

# This parameter has no effect if auto-config is disabled.

#

hashmode: hash5tuplesorted

**Napatech**와 **Suricata**를 통합하여 고성능 네트워크 트래픽 분석을 수행하기 위한 설정을 다룹니다. Napatech는 **FPGA** 기반의 네트워크 가속 솔루션을 제공하여 패킷 처리 성능을 극대화합니다.

#### **1. 모든 스트림 사용 설정 (use-all-streams)**

* **설정 가능 값**: yes 또는 no
* **기본값**: no

use-all-streams를 yes로 설정하면, Suricata는 **Napatech** 서비스에서 설정된 모든 스트림을 자동으로 감지하고 수신합니다. **NTPL** 명령어를 통해 원하는 스트림을 사전에 생성해야 하며, 이 설정이 no로 되어 있을 경우, 명시적으로 정의한 스트림만 사용할 수 있습니다.

#### **2. 스트림 정의 (streams)**

* **설정 예시**: streams: ["0-3"]

**use-all-streams** 옵션을 비활성화할 때 사용됩니다. 지정된 스트림 또는 스트림 범위를 Suricata가 모니터링하도록 정의합니다. 예를 들어, "0-3"은 0번부터 3번까지의 스트림을 모니터링합니다.

#### **3. 스트림 통계 (enable-stream-stats)**

* **설정 가능 값**: yes 또는 no
* **기본값**: no

이 옵션을 yes로 설정하면 각 스레드/스트림에 대한 세부 패킷 및 바이트 카운터가 활성화됩니다.

#### **4. 자동 구성 (auto-config)**

* **설정 가능 값**: yes 또는 no
* **기본값**: yes

auto-config가 yes로 설정되면, 스레드가 존재하는 **NUMA** 노드에 따라 스트림이 자동으로 생성됩니다. **cpu-affinity**가 활성화된 경우, 작업 스레드 수에 맞게 스트림이 생성됩니다.

#### **5. 하드웨어 레벨 플로우 우회 (hardware-bypass)**

* **설정 가능 값**: yes 또는 no
* **기본값**: yes

hardware-bypass는 플로우를 우회 처리하는 하드웨어 기능을 활성화합니다. **Napatech** 하드웨어에서 이 기능을 지원하며, 특정 플로우가 우회 처리되도록 설정할 수 있습니다.

#### **6. 인라인 모드 (inline)**

* **설정 가능 값**: yes 또는 no
* **기본값**: no

inline 모드를 활성화하면, 주어진 포트로 들어오는 트래픽이 Suricata 분석 후 자동으로 대응되는 포트로 전달됩니다.

#### **7. 포트 설정 (ports)**

* **설정 예시**: ports: [0-1,2-3]

이 옵션은 자동 구성 모드에서 사용되는 **Napatech** 포트들을 정의합니다. 각 세그먼트는 동일한 네트워크 어댑터에 있어야 하며, 트래픽을 적절히 처리하기 위해 설정됩니다.

#### **8. 해시 모드 (hashmode)**

* **설정 가능 값**: hash2tuple, hash2tuplesorted, hash5tuple, hash5tuplesorted, roundrobin

이 옵션은 패킷을 어떤 방식으로 스트림에 분배할지 결정합니다. 예를 들어, hash5tuplesorted는 5튜플 해시를 사용하여 패킷을 스트림에 분배합니다. 이는 패킷의 소스 IP, 포트, 프로토콜 등을 기준으로 스트림을 결정하는 방식입니다.

**Napatech**와 **Suricata**의 통합은 네트워크 분석 성능을 극대화할 수 있는 중요한 설정입니다. 이 설정 파일을 통해 스트림 처리, 플로우 우회, 인라인 모드 등 여러 기능을 최적화할 수 있습니다.