Meer User Guide Documentation

Release 0.0.3

Champ Clark III

Nov 19, 2019

Contents

CHAPTER 1

What is Meer

"Meer" is a dedicated "spooler" for the Suricata IDS/IPS and Sagan log analysis engines. This means that as Suricata or Sagan write alerts out to a file, Meer can 'follow' that file and store the alert information into a database. You can think of the "spool" file as a 'queuing' system for alerts from Suricata or Sagan. Using a "spooling" system ensures the delivery of alerts to a back end database. This task was traditionally accomplished by using a file format called "unified2" which was developed by the SourceFire/Snort team and a program called Barnyard2. While unified2 has been useful, its binary nature makes it difficult to work with and it has not been extended in quite some time. Instead of following "unified2" files, Meer follows Suricata and Sagan's "EVE" (JSON) output format. Since the EVE output format is JSON, it is easier to work with. The EVE output also contains valuable information that does not exist in "unified2".

Meer is meant to be modular and simple. This project does not aim to replicate all features of Barnyard2. The idea is to replicate the more useful features and abandon the "cruft".

The primary Meer site is located at:

https://quadrantsec.com/meer

1.1 License

Meer is licensed under the GNU/GPL version 2.

CHAPTER 2

Installation

There are currently no binary packages of Meer available. However, installation from source is pretty straightforward.

2.1 Prerequisites

Before compiling and installing Meer, you will need to determine where you want your data to reside. Currently, Meer supports MariaDB, MySQL and PostgreSQL. In order to build Meer, you will need one or more of these installed with all development files. For example, Ubuntu/Debian systems can install via:

```
apt-get install mariadb-dev  # For MariaDB
apt-get install libmysqlclient-dev # For MySQL
apt-get install libpq-dev # For PostgreSQL
```

Meer uses a YAML configuration file. This means that Meer will need libyaml installed on the system. On Ubuntu/Debian systems, this can be installed via:

```
apt-get install libyaml-dev
```

Meer uses JSON-C to parse JSON output from Sagan and Suricata. On Ubuntu/Debian systems, this prerequisite can be installed via:

```
apt-get install libjson-c-dev
```

2.2 Source

Installation from source distributions files gives

Basic steps:

```
git clone https://github.com/beave/meer
cd meer
./autogen.sh
./configure
make
sudo make install
```

By default, this will install Meer into the `/usr/local/bin/` directory with the default Meer configuration file in the `/usr/local/etc/` directory. By default, Meer will compile with MySQL/MariaDB support.

2.2.1 Common configure options

`--prefix``=/usr/`
: Installs the Meer binary in the /usr/bin. The default is `/usr/local/bin`.

`--sysconfdir``=/etc`
: Installs the Meer configuration file (meer.yaml) in the /etc directory. The default is `/usr/local/etc/`.

`--disable-mysql`
: This flag disables MySQL or MariaDB support. By default –enable-mysql is used.

`--enable-postgresql`
: This flag enables PostgreSQL support. By default –disable-postgresql is used.

`--with-libjsonc-libraries`
: This option points Meer to where the json-c libraries reside.

`--with-libjsonc-includes`
: This option points Meer to where the json-c header files reside.

`--with-libyaml_libraries`
: This option points Meer to where the libyaml files reside.

`--with-libyaml-includes`
: This option points Meer to where the libyaml header files reside.

`--enable-redis`
: This option enables redis output support. It requires “hiredis” to be installed on the target.

CHAPTER 3

Command Line Options

The majority of controls for Meer are within the `meer.yaml` file.

`-d, --daemon`
This option tells Meer to fork to the background.

`-c, --config`
This option tells what configuration file to use. By default Meer uses `/usr/local/etc/meer.yaml`.

`-h, --help`
The Meer help screen.

`-q, --quiet`
This option to tells Meer to not output to the console. Logs are still sent to the /var/log/meer directory.

CHAPTER 4

Starting Meer

To start Meer as root type:

```
/usr/local/bin/meer
```

To start Meer with a specified configuration file as root type:

```
/usr/local/bin/meer --config /path/to/my/config
```

To start Meer with a specified configuration file in “quiet” mode as root type:

```
/usr/local/bin/meer --config /path/to/my/config --quiet
```

to start Meer in the background as “root” type:

```
/usr/local/bin/meer --daemon
```

CHAPTER 5

Database/File Configuration & Rights

5.1 Directories and rights

Meer keeps track of its actions in the `meer.log` and its position in the Sagan or Suricata EVE output file through a "waldo" file. This means that Meer will need an area to record data to. The default location of these files is in the `/var/log/meer` directory. You will need to create and assign this directory the proper rights.

Making directories & assigning rights:

```
sudo mkdir /var/log/meer
sudo chown suricata /var/log/meer # Chown to 'sagan' if using with Sagan!
```

You will need to adjust your `meer.yaml` rights in the `runas` option to match your permissions.

5.2 Setting up a new database

If you do not have a database already configured to receive alerts, the instructions below will help you get started. First, you will need to create a database. For the purpose of this document the target database will be known as `example_database`. Database schemas are stored in the `meer/sql` directory.

Creating the `example_database` with MySQL/MariaDB:

```
mysqladmin -u root -p create example_database
mysql -u root -p example_database < sql/create_mysql
```

When using PostgreSQL, use the `meer/sql/create_postgresql` schema file.

5.3 Using an old database

If you have a legacy database that you wish to convert, do the following with MySQL/MariaDB:

```
mysql -u root -p example_database < sql/extend_mysql
```

This will create new tables (https, flows, dns, etc).

5.4 Setting database rights

It is important to setup the proper rights when using Meer. Meer needs only INSERT and SELECT on all tables. It will need INSERT, SELECT and UPDATE on the `example_database.sensor` table.

With MySQL/MariaDB:

```
GRANT INSERT,SELECT ON example_database.* to myusername@127.0.0.1 identified by
↪'mypassword`;
GRANT INSERT,SELECT,UPDATE,INSERT ON example_database.sensor to myusername@127.0.0.1␣
↪identified by 'mypassword';
```

CHAPTER 6

Meer 'core' configuration:

Meers operations are mainly controlled by the `meer.yaml` file. The configuration file is split into two sections. The `meer-core` controls how Meer processes incoming data from EVE files. The `output-plugins` controls how data extracted from the EVE files is transported to a database backend.

6.1 'meer-core' example

```
meer-core:

   core:

     hostname: "mysensor"  # Unique name for this sensor (no spaces)
     interface: "eth0"     # Can be anything.  Sagan "syslog", suricata "eth0".

     runas: "suricata"     # User to "drop privileges" too.
     #runas: "sagan"

     classification: "/etc/suricata/classification.config"
     #classification: "/usr/local/etc/sagan-rules/classification.config"

     meer_log: "/var/log/meer/meer.log"   # Meer log file

     # Meer can decode various types of data from within an "alert".  This
     # section enabled/disabled various JSON decoders.

     metadata: enabled
     flow: enabled
     http: enabled
     tls: enabled
     ssh: enabled
     smtp: enabled
     email: enabled
     json: enabled       # Original JSON from EVE
```

(continues on next page)

(continued from previous page)

```
    # This enables the "fingerprint" option.  When used in conjunction with the
    # "fingerprint.rules" (https://github.com/quadrantsec/fingerprint-rules),
    # this will record things like operating system type,  type of system it is
    # (client/server), etc.  This data get routed differently and does not
    # generate "alerts".

    fingerprint: enabled
    fingerprint_log: "/tmp/fingerprint.eve"

    # "client_stats" are specific to Sagan and allow Sagan/Meer to record
    # informatin about systems sending Sagan data.  This has no affect on
    # Suricata.

    client_stats: disabled

    # If "dns" is enabled, Meer will do reverse DNS (PTR) lookups of an IP.
    # The "dns_cache" is the amount of time Meer should "cache" a PTR record.
    # The DNS cache prevents Meer from doing repeated lookups of an
    # already looked-up PTR record.  This reduces over-loading DNS servers.

    dns: enabled
    dns_cache: 900      # Time in seconds.

    # "health" checks are a set of signatures that are triggered every so
    # often to ensure a sensor is up and operational.  When these events
    # are triggered,  they are not stored into the database as normal alert
    # data.  For example,  with MySQL/MariaDB output enabled, they update the
    # "sensor.health" table with the current epoch time.  Think of these
    # events like a "ping" for your sensor.  This can be useful for detecting
    # when Meer, Suricata, or Sagan have "died" unexpectedly.

    health: enabled
    health_signatures: 20000001,20000002,20000003,20000004

    waldo_file: "/var/log/meer/meer.waldo"      # Where to store the last
                                                # position in the
                                                # "follow-eve" file.

    lock_file: "/var/log/meer/meer.lck"         # To prevent dueling processes.

    follow_eve: "/var/log/suricata/alert.json"  # The Suricata/Sagan file to monitor
    #follow_eve: "/var/log/sagan/alert.json
```

6.2 'meer-core' options

Below describes the options in the *meer-core* section of the `meer.yaml`.

6.2.1 hostname

This is stored in the database in the `sensor` table under the `hostname` column. The `interface` is appended to the `hostname`. This option is required.

6.2.2 interface

The `interface` is stored in the `sensor` table appended to the `hostname` and `interface` columns. This describes in what interface the data was collected. This can be any descriptive string. For example, "eth0", "syslog", etc. This option is required.

6.2.3 runas

This is the user name the Meer process should "drop privileges" to. You will likely want to run Meer as the same user name that is collecting information. For example, "sagan" or "suricata". The `runas` can protect your system from security flaws in Meer. **Do not run as "root"**. This option is required.

6.2.4 classification

The `classification` option tells Meer where to find classification types. This file typically ships with Sagan, Suricata, and Snort rules. It defines a 'classtype' (for example, "attempt-recon") and assigns a numeric priority to the event. This option is required.

6.2.5 meer_log

The `meer_log` is the location of the file for Meer to record errors and statistics to. The file will need to be writable by the same user specified in the `runas` option.

6.2.6 metadata

The `metadata` option tells Meer to decode "metadata" from Suricata or Sagan. If the "metadata" is present in the alert, Meer will decode it and store its contents in memory for later use.

6.2.7 flow

The `flow` option tells Meer to decode "flow" data from Suricata or Sagan. If the "flow" JSON is present in the alert, Meer will decode it and store its contents in memory for later use.

6.2.8 http

The `http` option tells Meer to decode "http" data from Suricata or Sagan. If the "http" JSON is present in the alert, Meer will decode it and store its contents in memory for later use.

6.2.9 tls

The `tls` option tells Meer to decode "tls" data from Suricata or Sagan. If the "tls" JSON is present in the alert, Meer will decode it and store its contents in memory for later use.

6.2.10 ssh

The `ssh` option tells Meer to decode "ssh" data from Suricata or Sagan. If the "ssh" JSON is present in the alert, Meer will decode it and store its contents in memory for later use.

6.2.11 smtp

The `smtp` option tells Meer to decode "smtp" data from Suricata or Sagan. If the "smtp" JSON is present in the alert, Meer will decode it and store its contents in memory for later use.

6.2.12 email

The `email` option tells Meer to decode "email" data from Suricata or Sagan. If the "email" JSON is present in the alert, Meer will decode it and store its contents in memory for later use. This is not to be confused with `smtp`. The data from `email` will contain information like e-mail file attachments, carbon copies, etc.

6.2.13 json

The `json` option tells Meer to store the original JSON/EVE event. This is the raw event that Meer has read in.

6.2.14 fingerprint

The `fingerprint` option tells Meer to decode "fingerprint" rules and route the data differently. Fingerprint rules do not work like normal rules. The data from these rules is used to passively fingerprint systems for operating systems and types (client/server). This information can be valuable to determine if an attack might have been successful or not. Fingerprint rules are located at https://github.com/quadrantsec/fingerprint-rules.

6.2.15 fingerprint_log

When fingerprint rules fire, this is the log file that is create and data sent to. This log file format is an JSON (EVE) log file and is meant to be routed to a Elasticsearch back end. The idea is to store this information for historical purposes.

6.2.16 dns

The `dns` option tells Meer to perform a DNS PTR (reverse) record lookup of the IP addresses involved in an alert. This option is useful because it records the DNS record at the time the event occurred.

6.2.17 dns_cache

When `dns` is enabled, Meer will internally cache records to avoid repetitive lookups. For example, if 1000 alerts come in from a single IP address, Meer will look up the DNS PTR record one time and use the cache for the other 999 times. This saves on lookup time and extra stress on the internal DNS server. If you do not want Meer to cache DNS data, simply set this option to 0. The `dns_cache` time is in seconds.

6.2.18 health

The `health` option is a set of signatures used to monitor the health of Meer and your Sagan or Suricata instances. When enabled, Meer will treat certain Sagan and Suricata signatures as "health" indicators rather than normal alerts. When a "health" signature occurs, Meer updates the `sensor` table `health` column with the epoch time the health signature triggered. This can be useful in quickly determining if a sensor is down or behind (back logged) on alerts.

6.2.19 health_signatures

When `health` is enabled, this option supplies a list of signature IDs (sid) to Meer of Suricata or Sagan "health" signatures.

6.2.20 waldo_file

The `waldo_file` is a file that Meer uses to keep track of its last location within a EVE/JSON file. This keeps Meer from re-reading data in between stop/starts. This option is required.

6.2.21 lock_file

The `lock_file` is used to help avoid multiple Meer processes from processing the same data. The lock_file should be unique per Meer instance. The lock file contains the process ID (PID) of instance of Meer. This option is required.

6.2.22 follow_eve

The `follow_eve` option informs Meer what file to "follow" or "monitor" for new alerts. You will want to point this to your Sagan or Suricata "alert" EVE output file. You can think of Meer "monitoring" this file similar to how "tail -f" operates. This option is required.

CHAPTER 7

Output Plugins

7.1 SQL output-plugins example

Below is an example of the "output-plugins" from the `meer.yaml`. This section controls the SQL output.

```
output-plugins:

  # MySQL/MariaDB output - Stores data from Suricata or Sagan into a semi-
  # traditional "Barnyard2/Snort"-like database.

  sql:

    enabled: yes
    driver: mysql        # "mysql" or "postgresql"
    port: 3306           # Change to 5432 for PostgreSQL
    debug: no
    server: 127.0.0.1
    port: 3306
    username: "XXXX"
    password: "XXXXXX"
    database: "snort_test"

    # Automatically reconnect to the database when disconnected.

    reconnect: enabled
    reconnect_time: 10

    # Store decoded JSON data that is similar to Unified2 "extra" data to the
    # "extra" table.

    extra_data: enabled

    # Store extra decoded JSON metadata from Suricata or Sagan.  This requires
    # your database to have the metadata, flow, http, etc. tables.  If all are
```

(continues on next page)

(continued from previous page)

```
    # disabled,  Meer will store data in strictly a Barnyard2/Snort method.
    # If you want to store this decoded information,  and you likely do,  make
    # sure you have the decoders enabled in the "core" section of this Meer
    # configuration file!

    metadata: enabled
    flow: enabled
    http: enabled
    tls: enabled
    ssh: enabled
    smtp: enabled
    email: enabled
    json: enabled

    # If you would like Meer to mimic the legacy "reference" tables from
    # Snort/Barnyard2, enable it here.  If you are using more than one database
    # to store Suricata or Sagan data, you will likely want to leave this
    # disabled. The legacy reference system is not very efficient and there are
    # better ways to keep track of this data.  This is also a memory hog and
    # performance killer.  See tools/reference_handler/reference_handler.pl to
    # build a centralized reference table.

    reference_system: disabled
    sid_file: "/etc/suricata/rules/sid-msg.map"    # Created with "create-sidmap"
    reference: "/etc/suricata/reference.config"

    #sid_file: "/usr/local/etc/sagan-rules/sagan-sid-msg.map"
    #reference: "/usr/local/etc/sagan-rules/reference.config"
```

7.1.1 enabled

When this option is set to `yes` or `no`, it enables or disables the SQL section of the Meer output plugin.

7.1.2 driver

This controls what SQL database driver Meer will use. Valid types are `mysql` (for both MySQL and MariaDB) and `postgresql`.

7.1.3 port

The port the target SQL server is listening on.

7.1.4 server

The IP address of the SQL server.

7.1.5 debug

When `debug` is enabled, Meer will display SQL statements and transactions to stdout and to the `meer_log`. This can be useful for debugging SQL errors and issues. By default, this is disabled.

7.1.6 username

The username to use during authentication with the SQL database.

7.1.7 password

The password to use during authentication with the SQL database.

7.1.8 reconnect

If Meer encounters an issue with connecting to the SQL database, if this option is `enabled`, Meer will continually try to reconnect until it is successful.

7.1.9 reconnect_time

This is how long to pause, in seconds, before attempting to reconnect to the SQL database if the `reconnect` option is enabled.

7.1.10 extra_data

When the `extra_data` option is enabled, Meer will record certain information (XFF, DNS data, SMTP data, etc) in the legacy `extra` table.

7.1.11 metadata

This option controls Meer's ability to record decoded alert metadata to the `metadata` SQL table. If "metadata" is detected within the EVE/JSON and the `metadata` decoder is enabled (controlled in the `meer-core`), then it will be recorded to the `metadata` SQL table.

7.1.12 flow

This option controls Meer's ability to record decoded alert flow to the `flow` SQL table. If "flow" is detected within the EVE/JSON and the `flow` decoder is enabled (controlled in the `meer-core`), then it will be recorded to the `flow` SQL table.

7.1.13 http

This option controls Meer's ability to record decoded alert http to the `http` SQL table. If "http" is detected within the EVE/JSON and the `http` decoder is enabled (controlled in the `meer-core`), then it will be recorded to the `http` SQL table.

7.1.14 tls

This option controls Meer's ability to record decoded alert tls to the `tls` SQL table. If "tls" is detected within the EVE/JSON and the `tls` decoder is enabled (controlled in the `meer-core`), then it will be recorded to the `tls` SQL table.

7.1.15 ssh

This option controls Meer's ability to record decoded alert ssh to the `ssh` SQL table. If "ssh" is detected within the EVE/JSON and the `ssh` decoder is enabled (controlled in the `meer-core`), then it will be recorded to the `ssh-client``and ``ssh-server` SQL tables.

smtp ~~~

This option controls Meer's ability to record decoded alert smtp to the `smtp` SQL table. If "smtp" is detected within the EVE/JSON and the `smtp` decoder is enabled (controlled in the `meer-core`), then it will be recorded to the `smtp` SQL table.

7.1.16 email

This option controls Meer's ability to record decoded alert email to the `email` SQL table. If "email" is detected within the EVE/JSON and the `email` decoder is enabled (controlled in the `meer-core`), then it will be recorded to the `email` SQL tables. This is not to be confused with the `smtp` table.

7.1.17 reference_system

The `reference_system` allows Meer to store alert reference data in a traditional "Barnyard2" format. If you are using a single database for all events, this option might be useful to you. If you are using UIs like Snorby, Squeel, etc. you will likely want to enable this option. If you are using multiple databases, then consider looking at the "reference_handler.pl" script that ships with Meer.

7.1.18 sid_file

The `sid_file` is a legacy "signature message map" file that points signature IDs to their references. If you want to use the legacy `reference_system`, you will need a "signature message map" (`sid_file`) for Meer to read.

7.2 "pipe" output

Below is an example of the "pipe" output plugin. This takes data being written to the EVE file and puts it into a named pipe (FIFO). This can be useful if you want a third party program (for example, Sagan - https://sagan.io) to analyze the data.

```
pipe:

    enabled: no
    pipe_location: /var/sagan/fifo/sagan.fifo
    pipe_size: 1048576                      # System must support F_GETPIPE_SZ/F_
↪SETPIPE_SZ
    metadata: enabled

    # Below are the "event_types" from Suricata/Sagan. This tells Meer what to send
    # to the named pipe/FIFO.

    alert: enabled
    files: enabled
    flow: enabled
    dns: enabled
```

(continues on next page)

(continued from previous page)

```
    http: enabled
    tls: enabled
    ssh: enabled
    smtp: enabled
    fileinfo: enabled
    dhcp: enabled
```

7.2.1 enabled

Enabled/disabled the 'pipe' output.

7.2.2 pipe_location

Location of the named pipe on the file system.

7.2.3 pipe_size

Number of bytes will set the size of the named pipe/FIFO to.

7.2.4 metadata

This option controls Meer's ability to record decoded alert metadata to the named pipe. If "metadata" is detected within the EVE/JSON and the `metadata` decoder is enabled (controlled in the `meer-core`), then it will be recorded to the named pipe.

7.2.5 flow

This option controls Meer's ability to record decoded alert flow to named pipe. If "flow" is detected within the EVE/JSON and the `flow` decoder is enabled (controlled in the `meer-core`), then it will be recorded to the named pipe.

7.2.6 http

This option controls Meer's ability to record decoded alert http to the named pipe. If "http" is detected within the EVE/JSON and the `http` decoder is enabled (controlled in the `meer-core`), then it will be recorded to the named pipe.

7.2.7 tls

This option controls Meer's ability to record decoded alert tls to the named pipe. If "tls" is detected within the EVE/JSON and the `tls` decoder is enabled (controlled in the `meer-core`), then it will be recorded to the named pipe.

7.2.8 ssh

This option controls Meer's ability to record decoded alert ssh to the named pipe. If "ssh" is detected within the EVE/JSON and the `ssh` decoder is enabled (controlled in the `meer-core`), then it will be recorded to the named pipe.

smtp ~~~

This option controls Meer's ability to record decoded alert smtp to the named pipe. If "smtp" is detected within the EVE/JSON and the `smtp` decoder is enabled (controlled in the `meer-core`), then it will be recorded to the named pipe.

7.2.9 email

This option controls Meer's ability to record decoded alert email to the named pipe. If "email" is detected within the EVE/JSON and the `email` decoder is enabled (controlled in the `meer-core`), then it will be recorded to the named pipe. This is not to be confused with the `smtp` table.

7.2.10 fileinfo

This option controls Meer's ability to record decoded alert fileinfo to the named pipe. If "fileinfo" is detected within the EVE/JSON and the `fileinfo` decoder is enabled (controlled in the `meer-core`), then it will be recorded to the named pipe.

7.2.11 dhcp

This option controls Meer's ability to record decoded alert dhcp to the named pipe. If "dhcp" is detected within the EVE/JSON and the `dhcp` decoder is enabled (controlled in the `meer-core`), then it will be recorded to the named pipe.

7.3 "external" output

This option allows signatures to call "external" programs. For example, if a signature the proper "metadata" (`metadata: meer external` or a set policy), Meer will fork a copy of the specified program and pass the EVE via stdin. This feature can be useful for creating custom firewalling routines or routing data to alternate programs. The "external" program can be written in any language that suites you.

```
##############################################################################
# external
#
# EVE data (JSON) is passed via stdin to the external program.   The
# external program can be written in any language you choose (shell script,
# Python, Perl, etc).
#
# This can be useful for automatic firewalling,  building block lists,
# replicating "snortsam" functionality, etc.  See the "tools/external"
# directory for example routines that use this feature.
#
# If this option is enabled, any rule that has the metadata of "meer
# external" (ie - "metadata:meer external") will cause the external script
# to be executed.  Execution can also be controlled by Snort metadata
```

(continues on next page)

(continued from previous page)

```
# "policies".
##############################################################################

external:

  enabled: no
  debug: no

  # Execution of an external program based on metadata "policy".  When Meer
  # encounters a "policy" (security-ips, balanced-ips, connectivity-ips,
  # and max-detect-ips),  Meer will execute the specified routine.
  # Currently only Snort rules have these types of polices.  This can be
  # useful when you want to execute an external script that will to "block"
  # or "firewall" based off the policy types.  This section only applies if
  # you are using Suricata with Snort rules.  Snort's polices are
  # below:

  # connectivity-ips  - You run a lot of real time applications (VOIP,
  # financial transactions, etc), and don't want to run any rules that
  # could affect the current performance of your sensor.  The rules in this
  # category make snort happy, additionally this category focuses on the high
  # profile most likely to affect the largest number of people type of
  # vulnerabilities.

  # balanced-ips - You are normal, you run normal stuff and you want normal
  # security protections.  This is the best policy to start from if you are
  # new, old, or just plain average.  If you don't have any special
  # requirements for super high speeds or super secure networks start here.

  # security-ips - You don't care about dropping your bosses email, everything
  # in your environment is tightly regulated and you don't tolerate people
  # stepping outside of your security policy.  This policy hates on IM, P2P,
  # vulnerabilities, malware, web apps that cause productivity loss, remote
  # access, and just about anything not related to getting work done.
  # If you run your network with an iron fist start here.

  # I can't seem to find any documentation on what "max-detect-ips" is :(

  policy-security-ips: enabled
  policy-max-detect-ips: enabled
  policy-connectivity-ips: enabled
  policy-balanced-ips: enabled

  program: "/usr/local/bin/external_program"
```

7.3.1 enabled

Keyword is used to enable/disable `external` output.

7.3.2 debug

When enabled, this option will display and log debugging information.

7.3.3 policy-security-ips

Execute `external` program when the `policy-security-ips` is encountered.

7.3.4 policy-max-detect-ips

Execute `external` program when the `policy-max-detect-ips` is encountered.

7.3.5 policy-connectivity-ips

Execute `external` program when the `policy-connectivity-ips` is encountered.

7.3.6 policy-balanced-ips

Execute `external` program when the `policy-balanced-ips` is encountered.

7.3.7 program

`external` program to execute when conditions are met.

7.4 Redis output

This controls how Meer logs to a Redis database. Meer can record alert records to Redis similar to how Suricata with Redis support enabled does. Redis is also used as a temporary storage engine for `client_stats` (Sagan only) and `fingerprint` data if enabled.

```
##############################################################################
# "redis" allows you to send Suricata/Sagan EVE data to a Redis database.
# This will mimic the way Suricata writes EVE data to Redis with the
# exception of "client_stats" which is a Sagan specific processor.
##############################################################################

redis:

  enabled: no
  debug: no
  server: 127.0.0.1
  port: 6379
  batch: 10                # Batch/pipelining mode. Max is 100. 1 == no batching.
  key: "suricata"          # Default 'key' or 'channel' to use.
  mode: list               # How to publish data to Redis.  Valid types are list/
↪lpush,
                           # rpush, channel|publish.

  # This controls event_types to send to Redis.

  alert: enabled
  files: enabled
  flow: enabled
  dns: enabled
```

(continues on next page)

(continued from previous page)

```
  http: enabled
  tls: enabled
  ssh: enabled
  smtp: enabled
  fileinfo: enabled
  dhcp: enabled

  # Fingerprint data can be temporarily stored in a Redis database.  When an alert
  # fires, this information can be used to determine the targets operating system,
  # type (client/server), etc.  This can be useful in determining the validity of
  # an event. If used in conjunction with the SQL output,  the fingerprint data for
  # the targeted system is stored in the 'fingerprint' table.

  fingerprint: enabled

  # This controls sending Sagan client tracking data to Redis.  This has no affect
  # on Suricata systems.

  client_stats: disabled
```

7.4.1 enabled

Enable or disable the Redis output.

7.4.2 debug

Enable or disabled Redis debugging.

7.4.3 server

The Redis server address you want to store data to.

7.4.4 port

Port of the target Redis server.

7.4.5 batch

The `batch` is the amount of data to collect before sending it to Redis. This has no affect when using Redis with either `client_stats` or `fingerprint` data.

7.4.6 key

The `key` is the default Redis channel or key to use.

7.4.7 mode

The `mode` controls how data is stored to Redis. Valid options are `list`, `lpush`, `rpush`, `channel` or `publish`. The default is `list`. The method Meer stores the data is compatible with Suricata's Redis output format. Note; This option does not have any affect on `client_stats` or `fingerprint` recording.

7.4.8 alert

Enable or disable storing `alert` data into Redis.

7.4.9 files

Enable or disable storing `files` data into Redis.

7.4.10 flow

Enable or disable storing `flow` data into Redis.

7.4.11 dns

Enable or disable storing `dns` data into Redis.

7.4.12 http

Enable or disable storing `http` data into Redis.

7.4.13 tls

Enable or disable storing `tls` data into Redis.

7.4.14 ssh

Enable or disable storing `ssh` data into Redis.

7.4.15 smtp

Enable or disable storing `smtp` data into Redis.

7.4.16 fileinfo

Enable or disable storing `fileinfo` data into Redis.

7.4.17 dhcp

Enable or disable storing `dhcp` data into Redis.

7.4.18 fingerprint

Enable or disable storing `fingerprint` data in the Redis database. This is a temporary storage system for `fingerprint` data. This allows correlation between device fingerprints (ie - operating systems, devices types, etc) with alerts.

7.4.19 client_stats

This is a Sagan only option. This option temporarily stores devices that are sending Sagan logs along with an example log entry. This has no affect with Suricata.

CHAPTER 8

Console Output

8.1 Console/Log Startup

At start up, the logs and console output give you information about the status of Meer. For example, you will want to note the `SQL information`, such as the driver and whether a successful connection was made. If there is a problem making a connection to your database, Meer will display the error that is causing the issues.

Another important item to note is the database sensor ID. This will be the ID number used in the database to store events.

Common issues are database rights and directory/file permission problems.

If Meer makes it to the `Waiting of new data...`, then Meer has successfully started.

```
[*] [11/09/2018 03:24:29] -  @@@@@@@@@@  @@@@@@@@ @@@@@@@@ @@@@@@@    Meer version 0.
↪0.3-git
[*] [11/09/2018 03:24:29] -  @@! @@! @@! @@!      @@!      @@!  @@@   Quadrant␣
↪Information Security
[*] [11/09/2018 03:24:29] -  @!! !!@ @!@ @!!!:!   @!!!:!   @!@!!@a    https://
↪quadrantsec.com
[*] [11/09/2018 03:24:29] -  !!:     !!: !!:      !!:      !!: :!a    Copyright (C)␣
↪2018
[*] [11/09/2018 03:24:29] -   :      :   : :: ::  : :: ::   :   : :
[*] [11/09/2018 03:24:29] -
[*] [11/09/2018 03:24:29] - Dropping privileges! [UID: 999 GID: 999]
[*] [11/09/2018 03:24:29] - Classifications file loaded [/etc/suricata/classification.
↪config].
[*] [11/09/2018 03:24:29] -
[*] [11/09/2018 03:24:29] - Decode 'json'    : enabled
[*] [11/09/2018 03:24:29] - Decode 'metadata': enabled
[*] [11/09/2018 03:24:29] - Decode 'flow'    : enabled
[*] [11/09/2018 03:24:29] - Decode 'http'    : enabled
[*] [11/09/2018 03:24:29] - Decode 'tls'     : enabled
[*] [11/09/2018 03:24:29] - Decode 'ssh'     : enabled
[*] [11/09/2018 03:24:29] - Decode 'smtp'    : enabled
```

(continues on next page)

(continued from previous page)

```
[*] [11/09/2018 03:24:29] - Decode 'email'   : enabled
[*] [11/09/2018 03:24:29] -
[*] [11/09/2018 03:24:29] - Waldo loaded. Current position: 191000
[*] [11/09/2018 03:24:29] -
[*] [11/09/2018 03:24:29] - --[ SQL information ]-------------------------------------
↪-------
[*] [11/09/2018 03:24:29] -
[*] [11/09/2018 03:24:29] - SQL Driver: MySQL/MariaDB
[*] [11/09/2018 03:24:29] - Extra data: enabled
[*] [11/09/2018 03:24:29] - Legacy Reference System': disabled
[*] [11/09/2018 03:24:29] -
[*] [11/09/2018 03:24:30] - Successfully connected to MySQL/MariaDB database.
[*] [11/09/2018 03:24:30] - Using Database Sensor ID: 1
[*] [11/09/2018 03:24:30] - Last CID: 586325
[*] [11/09/2018 03:24:30] -
[*] [11/09/2018 03:24:30] - Recird 'json'    : enabled
[*] [11/09/2018 03:24:30] - Record 'metadata': enabled
[*] [11/09/2018 03:24:30] - Record 'flow'    : enabled
[*] [11/09/2018 03:24:30] - Record 'http'    : enabled
[*] [11/09/2018 03:24:30] - Record 'tls'     : enabled
[*] [11/09/2018 03:24:30] - Record 'ssh'     : enabled
[*] [11/09/2018 03:24:30] - Record 'smtp'    : enabled
[*] [11/09/2018 03:24:30] - Record 'email'   : enabled
[*] [11/09/2018 03:24:30] -
[*] [11/09/2018 03:24:30] - ------------------------------------------------------------
↪-----------------
[*] [11/09/2018 03:24:30] - Skipping to record 191000 in /var/log/suricata/alert.json
[*] [11/09/2018 03:24:31] - Reached target record of 191000.  Processing new records.
[*] [11/09/2018 03:24:31] - Read in 191000 lines
[*] [11/09/2018 03:24:31] - Waiting for new data......
```

8.2 Console/Log Shutdown

Upon shutdown, the Meer console and logs provide information about the previous execution. For example, how efficient DNS caching performed, how much data was stored in flow, http, tls, ssh, smtp, and meta data tables, and how efficient SQL caches performed. It also displays the last `Waldo Position`, which indicates what position it left off in the file. Another important item to note is the `CID`, which is the last database position Meer left off.

```
[*] [11/09/2018 03:24:29] - --[ Meer Statistics ]-------------------------------------
↪--
[*] [11/09/2018 03:24:29] -
[*] [11/09/2018 03:24:29] -  - Decoded Statistics:
[*] [11/09/2018 03:24:29] -
[*] [11/09/2018 03:24:29] -  Waldo Postion : 191000
[*] [11/09/2018 03:24:29] -  JSON          : 8987
[*] [11/09/2018 03:24:29] -  Flow          : 8987
[*] [11/09/2018 03:24:29] -  HTTP          : 8889
[*] [11/09/2018 03:24:29] -  TLS           : 0
[*] [11/09/2018 03:24:29] -  SSH           : 16
[*] [11/09/2018 03:24:29] -  SMTP          : 0
[*] [11/09/2018 03:24:29] -  Email         : 0
[*] [11/09/2018 03:24:29] -  Metadata      : 8782
[*] [11/09/2018 03:24:29] -
[*] [11/09/2018 03:24:29] -  - DNS Statistics:
```

(continues on next page)

(continued from previous page)

```
[*] [11/09/2018 03:24:29] -
[*] [11/09/2018 03:24:29] -  DNS Lookups   : 7374
[*] [11/09/2018 03:24:29] -  DNS Cache Hits: 45780 (86.127%)
[*] [11/09/2018 03:24:29] -
[*] [11/09/2018 03:24:29] -  - MySQL/MariaDB Statistics:
[*] [11/09/2018 03:24:29] -
[*] [11/09/2018 03:24:29] -  Health Checks          : 17590
[*] [11/09/2018 03:24:29] -  INSERT                 : 88920
[*] [11/09/2018 03:24:29] -  SELECT                 : 104
[*] [11/09/2018 03:24:29] -  UPDATE                 : 26578
[*] [11/09/2018 03:24:29] -  Class Cache Misses     : 15
[*] [11/09/2018 03:24:29] -  Class Cache Hits       : 26562 (99.944%)
[*] [11/09/2018 03:24:29] -  Signature Cache Misses : 15
[*] [11/09/2018 03:24:29] -  Signature Cache Hits   : 26562 (99.944%)
[*] [11/09/2018 03:24:29] -
[*] [11/09/2018 03:24:29] - Last CID is : 586325.
[*] [11/09/2018 03:24:29] - Shutdown complete.
```

CHAPTER 9

Getting help

The primary Meer site is located at:

https://quadrantsec.com/meer

The Meer Github site is located at:

https://github.com/beave/meer

If you are having issues getting Meer to work, consider posting in the Meer mailing list. This list is good for general configuration, install, and usage questions.

https://groups.google.com/forum/#!forum/meer-users

If you need to report a compile or programming issue, please use our Github.com issues page. That is located at:

https://github.com/beave/meer/issues

Index

Symbols

A

C