The Suricata IDS

- The Suricata Engine is an Open Source Next Generation Intrusion Detection and Prevention Engine with the following features:

- **Highly Scalable**

    - Suricata is multi threaded; more than one instance can be invoked and it will balance the load of processing across every processor on a sensor that the IDS is configured to use.

    - Allows off-the shelf hardware to achieve 10 gigabit speeds (according to their website) on real life traffic.

- **Protocol Identification**

- The most common protocols are automatically recognized by Suricata as the stream starts, thus allowing rule writers to write a rule to the protocol, not to the ports.

- The IDS uses dedicated keywords that can be matched on protocol fields which range from http URI to a SSL certificate identifier.

- **File Identification, MD5 Checksums, and File Extraction**

    - Can identify a large variety of file types within the network traffic, tag a file for extraction and the file will be written to disk with a meta data file describing the capture situation and flow.

    - The file's MD5 checksum is calculated in real-time, so if we have a list of md5 hashes that we wish to allow in the network, or want to block, Suricata can identify those files.

- **IDS / IPS**

    - Implements a complete signature language to match on known threats, policy violations and malicious behaviour.

    - Detects many anomalies in the traffic it inspects using the specialized **Emerging Threats Suricata ruleset** and the **VRT ruleset**.

- **High Performance**
    - A single Suricata instance is capable of inspecting multi-gigabit traffic. The engine is built around a multi threaded, modern, clean and highly scalable code base.
    - There is native support for hardware acceleration from several vendors and through PF_RING and AF_PACKET. Experimental GPU acceleration uploads some CPU intensive tasks to your graphics card.

- **Automatic protocol detection**
    - Suricata will automatically detect protocols such as HTTP on any port and apply the proper detection and logging logic. This greatly helps with finding malware and CnC (Command-and-Control) channels.

- **Network Security Monitoring**
    - Can log HTTP requests, log and store TLS certificates, extract files from flows and store them to disk. The full pcap capture support allows easy analysis.
    - TLS/SSL Logging and Analysis: matches against most aspects of an SSL/TLS exchange within the ruleset language.
    - The TLS Parser can also log all key exchanges for analysis. This provides protection against less than a reputable certificate authority.
    - HTTP Logging: Will log all HTTP connections on any port to file for later analysis.

- **Lua scripting**
    - Advanced analysis and functionality available to detect things not possible within the ruleset syntax.

- Industry standard outputs
    - Through the Unified2 output format and the Barnyard2 tool, Suricata can be used with BASE, Snorby, Sguil, SQueRT and all other tools out there.

Documentation

- Extensive documentation can be found at:

    https://suricata-ids.org/docs/

Download and Installation

- The full source can be downloaded from:

  https://suricata-ids.org/download/

- Make sure you read the "quick start" and "installation" guides and then follow the instructions therein.

General Fedora Installation Notes

- Make sure the following packages are installed:

```
dnf install libyaml-devel
dnf install file-devel
```

- Then build and install as follows:

```
#make
#make install-full
```

- Get the rule set as follows:

```
/usr/bin/wget -qO - https://rules.emergingthreats.net/open/suricata-3.0/emerging.rules.tar.gz | tar -x -z -C "/usr/local/etc/suricata/" -f -
```

- Start running Suricata as follows (your interface may be different):

```
/usr/local/bin/suricata -c /usr/local/etc/suricata//suricata.yaml -i enp6s0
```

- If you encounter a library error like "**libhtp.so is not found**", you can run it with:

```
LD_LIBRARY_PATH=/usr/local/lib /usr/local/bin/suricata -c /usr/local/etc/suricata//suricata.yaml -i enp6s0
```

- Or to run as daemon (just like we did with snort):

```
LD_LIBRARY_PATH=/usr/local/lib /usr/local/bin/suricata -c /usr/local/etc/suricata//suricata.yaml -i enp6s0 -D
```

- It is highly recommended to use a rule manager for maintaining rules. The two most common are Oinkmaster and Pulledpork.

- See the following guide:

https://redmine.openinfosecfoundation.org/projects/suricata/wiki/Rule_Management_with_Oinkmaster

- The Suricata configuration file is in:

  `/usr/local/etc/suricata/suricata.yaml`

- The default log directory is:

  `/usr/local/var/log/suricata/`

- For JSON output support install the following:

  `dnf install jansson*`

- Make sure you enable JSON format in **suricata.yaml** and then do a full rebuild again.

```
  # 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
      #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
      #redis:
      #  server: 127.0.0.1
      #  port: 6379
      #  mode: list ## possible values: list (default), channel
      #  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 as to be reserved to high traffic suricata.
      #  pipelining:
      #    enabled: yes ## set enable to yes to enable query pipelining
      #    batch-size: 10 ## number of entry to keep in buffer
      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)
            http: yes                # enable dumping of http fields
            tls: yes                 # enable dumping of tls fields
            ssh: yes                 # enable dumping of ssh fields
            smtp: yes                # enable dumping of smtp fields
:
```

- To view build information:

LD_LIBRARY_PATH=/usr/local/lib /usr/local/bin/suricata --build-info

- The following (partial) screenshot shows "**libjansson support**" enabled:

```
  DAG enabled:                          no
  Napatech enabled:                     no

  Unix socket enabled:                  yes
  Detection enabled:                    yes

  libnss support:                       no
  libnspr support:                      no
  libjansson support:                   yes
  hiredis support:                      no
  Prelude support:                      no
  PCRE jit:                             yes
  LUA support:                          no
  libluajit:                            no
  libgeoip:                             no
  Non-bundled htp:                      no
  Old barnyard2 support:                no
  CUDA enabled:                         no
  Hyperscan support:                    no
  Libnet support:                       no

  Suricatasc install:                   yes

  Profiling enabled:                    no
  Profiling locks enabled:              no

Development settings:
  Coccinelle / spatch:                  no
  Unit tests enabled:                   no
  Debug output enabled:                 no
  Debug validation enabled:             no

Generic build parameters:
  Installation prefix:                  /usr/local
```