owlh_documentation Documentation

Release 0.15.0

owlh team

Jul 06, 2020

Contents

CHAPTER 1

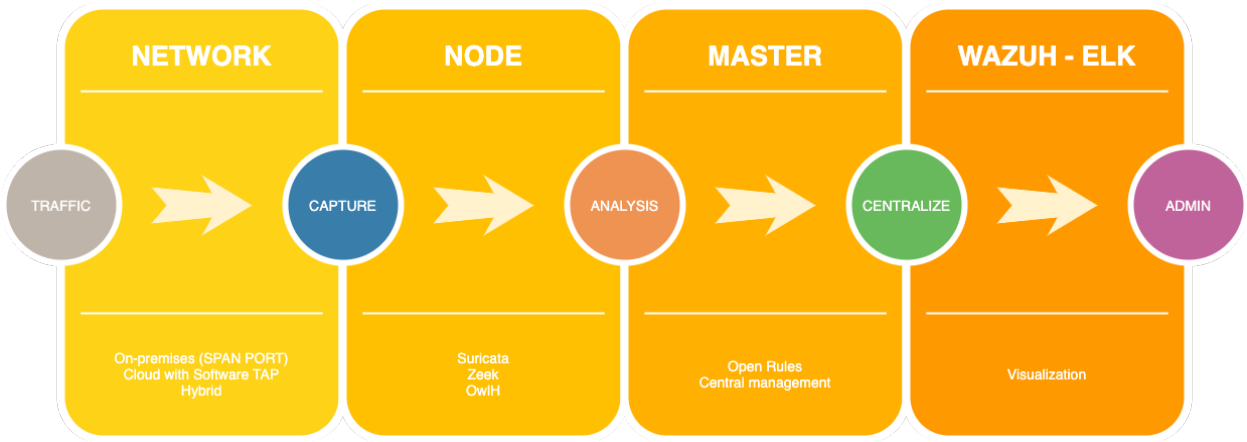
About OwlH

v0.14.x

This picture will summarize the process we are working in.

1.1 Getting Started with OwlH

1.1.1 Components



OwlH will help to integrate and manage multiple NIDS solutions, providing a centralized management solution. To accomplish this, we deploy different components.

OwlH provides flexibility and scalability to be integrated with 3rd party solutions as Moloch for forensics and many others. You can grow as needed in your network.

1.1.1.1 OwlH Master

Is an appliance running the centralized management API. All centralized stuff happens here. configurations, synchronizations.

see more about OwlH Master

1.1.1.2 OwlH Node

Is an appliance that will include NIDS software as Suricata and/or Zeek. This appliance will be able to listen network traffic, analyze it and forward analysis results to an storage and visualization platform like ELK or Splunk. It also helps with network traffic transport in our Software TAP configuration

see more about OwlH Node

1.1.1.3 OwlH UI

The graphical User Interface that will provide an easy access and visualization of all management capabilities

see more about OwlH User Interface

1.1.1.4 OwlH Client

Small and light weight client used to transport traffic from servers to an OwlH Node or OwlH Master when there is no way to access directly network traffic, ex. Cloud environments.

see more about OwlH Client

Check our Github repos

1.1.2 Capabilities

1.1.2.1 NIDS and Traffic analysis support

- Suricata Management
- Zeek Management
- Moloch Management

1.1.2.2 Top capabilities

- OpenRules
- Analyzer
- Groups

1.1.2.3 Traffic transport and management

- Software TAP NODE side
- Software TAP MASTER side
- Traffic dispatcher MASTER side
- Traffic Forwarder CLIENT side (Linux - Windows)

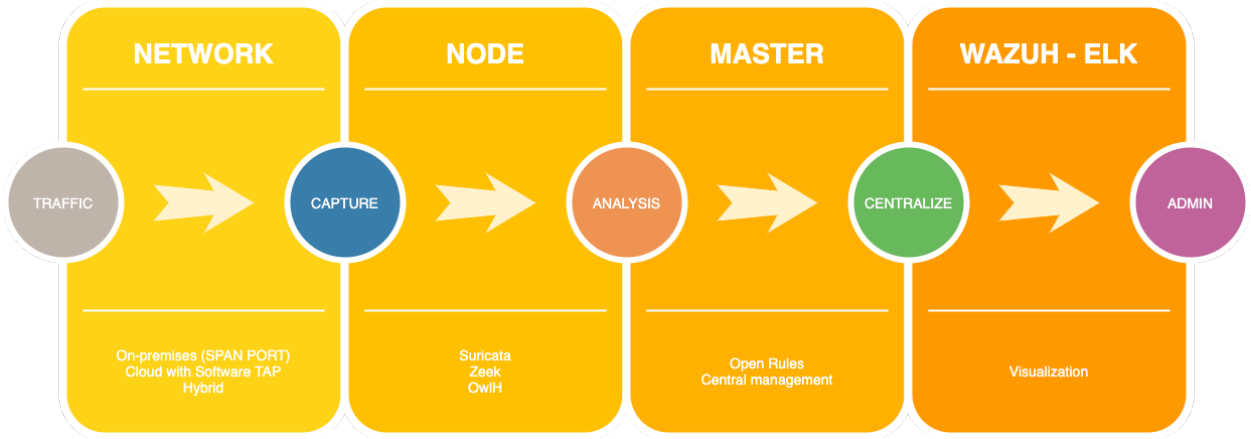
1.1.2.4 OwlH Plugins

- MAC management
- Known Ports management
- DNS data exfiltration analysis

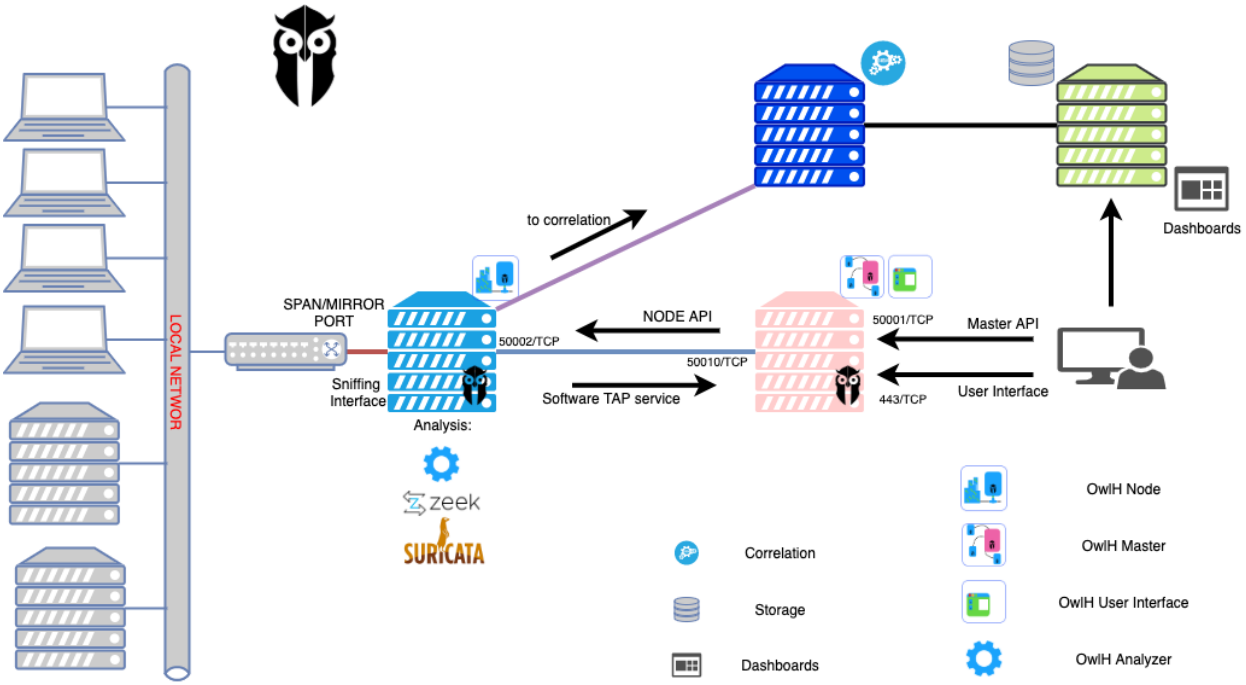
1.1.2.5 Others

- RBAC management
- User Authentication using LDAP
- Change Control records
- Internal incident records
- OwlH software update

1.1.3 Architecture

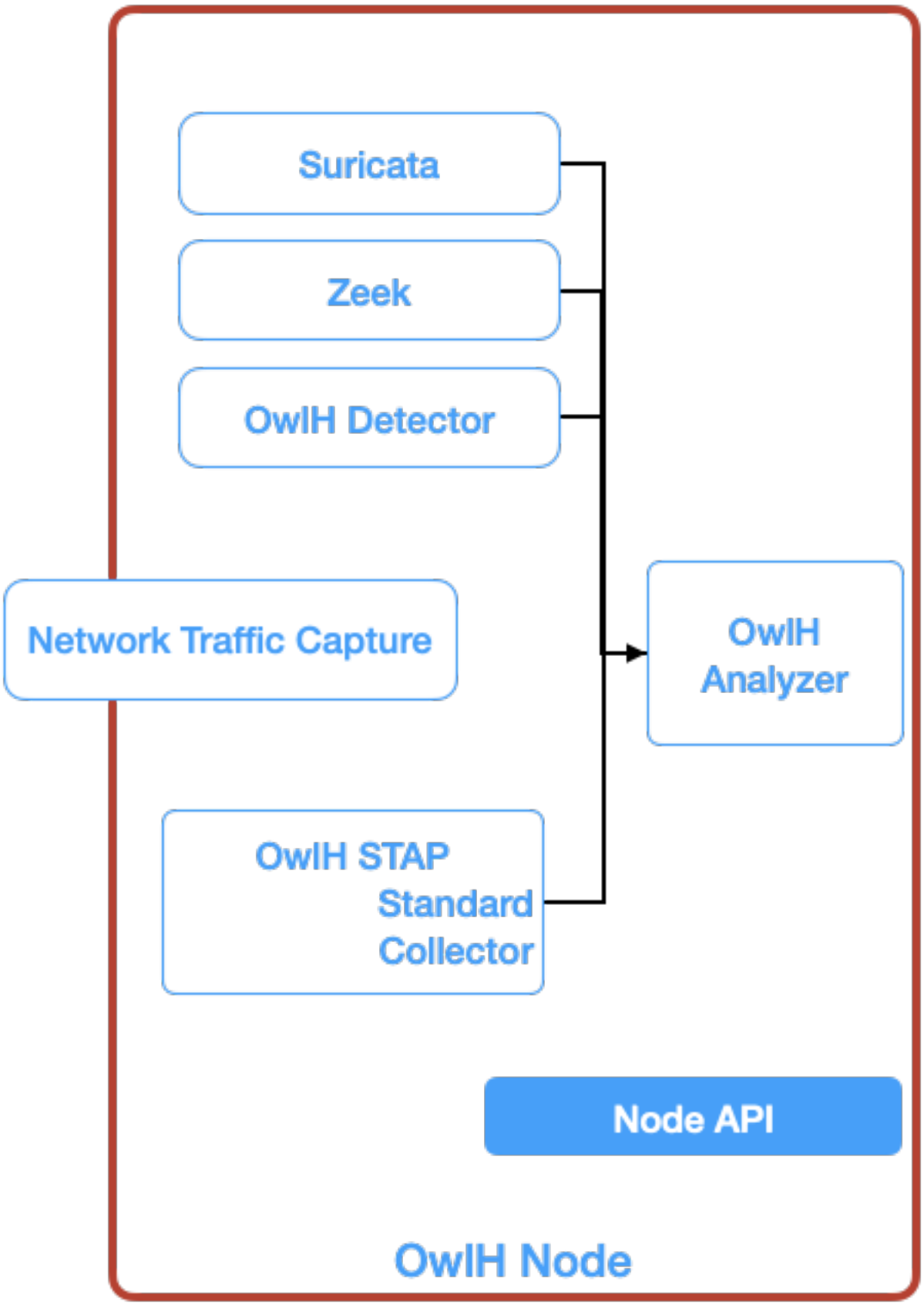


1.1.3.1 Standard data flow

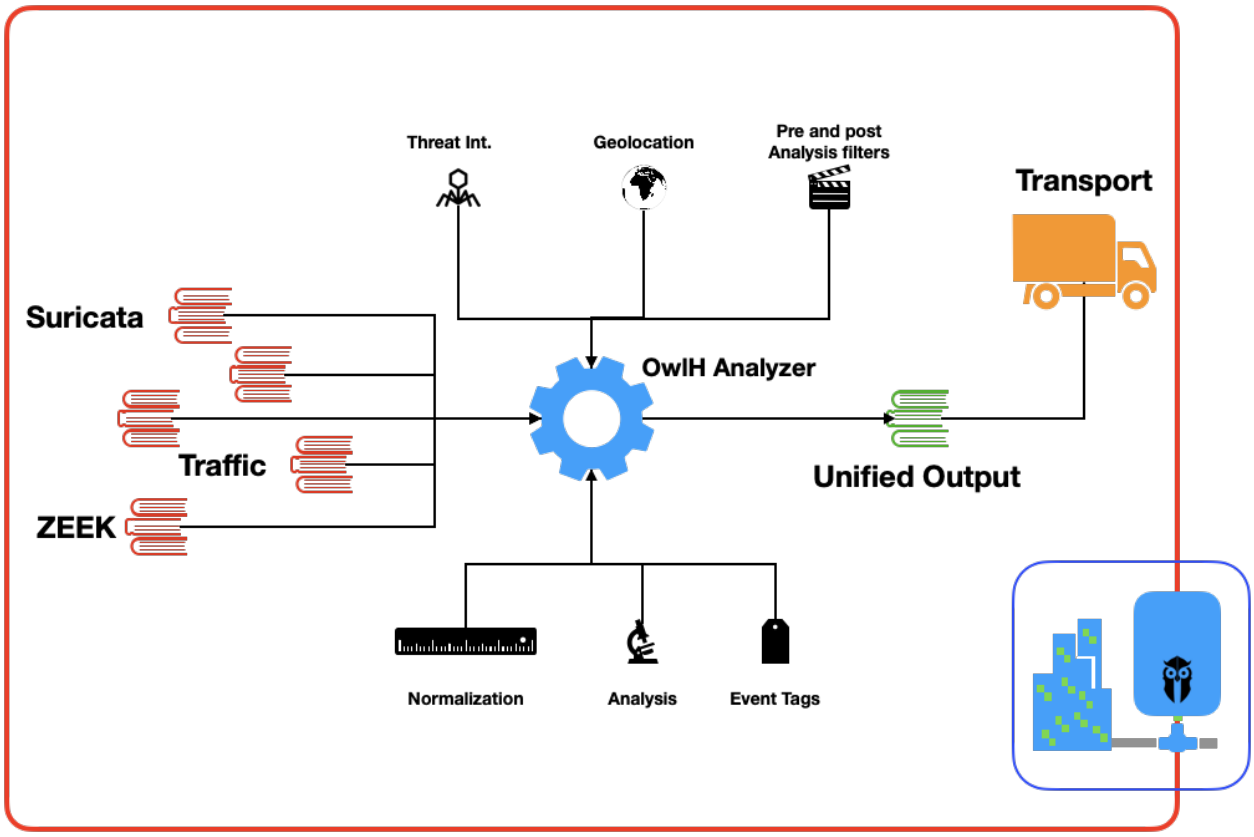


1.1.3.2 Inside OwlH Node

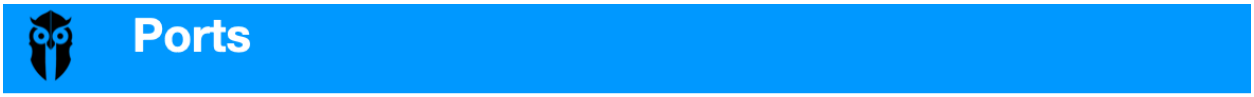
Traffic analysis by NIDS


Suricata
Zeek
OwlH Detector
Network Traffic Capture
OwlH
Analyzer
OwlH STAP
Standard
Collector
Node API
OwlH Node

NIDS output analysis and enrichment with OwlH Analyzer



1.1.3.3 Used ports



From	To	Dst Port
Owlh Master	OwlH Node	50002/TCP 22/TCP
OwlH Master	Wazuh Manager	1514/UDP
OwlH Node	OwlH Master	50001/TCP
OwlH Node	Wazuh Manager	1514/UDP
Admin User Station	OwlH Node	22/TCP
Admin User Station	OwlH Master	50001/TCP 22/TCP 443/TCP

Thanks to our flexible architecture, we can adapt to any scenario. Here we have some samples of running environments:

- Scenarios

Check our Github repos

1.2 Install OwlH

1.2.1 OwlH Installer

Install your OwlH components and related services with OwlH Installer.

- OwlH Installer

1.2.2 Install components

1.2.2.1 Standard Installation

Review your *scenario* an use these guides to help you to deploy

- OwlH Master
- OwlH Node
- OwlH UI
- OwlH Client

1.2.2.2 Advanced Installation

See more installation options available.

- Advanced installation

1.2.3 Configure

- User Interface - APP
- *API description*

1.2.4 Visualization

- OwlH dashboards integration Wazuh-ELK

1.2.5 Appendices

- main requirements

1.3 Update and upgrade OwlH

1.3.1 OwlH components

- OwlH Master
- OwlH Node
- OwlH UI
- OwlH Client

1.4 User Manual

1.4.1 First configuration

1.4.1.1 User Interface

- Access to your UI/APP
- Register a node
- **Create a ruleset for suricata**
  - Create a ruleset source
  - Create a local ruleset
  - Apply ruleset to a node or group of nodes

1.4.1.2 Analyzer

- Enable Analyzer

1.4.1.3 Suricata

- Configure Suricata

1.4.1.4 Zeek

- Configure Zeek as standalone

1.4.1.5 Wazuh

- Configure Wazuh to read the OwlH Analyzer output alerts.json file

1.4.2 Configuration Files

1.4.2.1 API service configuration files

Node configuration

```
appname = OwlHnode
runmode = dev
autorender = false
copyrequestbody = true
EnableDocs = true
ListenTCP4 = true
EnableHTTP = false
EnableHTTPS = true
HTTPSAddr = "0.0.0.0"
HTTPSPort = 50002
HTTPSCertFile = "conf/certs/ca.crt"
HTTPSKeyFile = "conf/certs/ca.key"
EnableDocs = true
```

Master configuration

```
appname = OwlHmaster
runmode = dev
autorender = false
copyrequestbody = true
EnableDocs = true
ListenTCP4 = true
EnableHTTP = false
EnableHTTPS = true
HTTPSAddr = "0.0.0.0"
HTTPSPort = 50001
HTTPSCertFile = "conf/certs/ca.crt"
```

(continues on next page)

(continued from previous page)

```
HTTPSKeyFile = "conf/certs/ca.key"
EnableDocs = true
```

1.4.2.2 Main Configuration files (main.conf)

there are main.conf files in owlh node and owlh master

you can find details about each file here:

Master Main Configuration file

this is master main.conf file description

If you need help

- email our support team - support@owlh.net
- join OwlH slack - OwlH Slack workspace
- ask for professional support and services - prohelp@owlh.net

OwlH - current v0.14.x - Mar - OwlH Changelog

documentation last updated - Jul 06, 2020

Node Main Configuration file

Global settings

- node
- loop
- files
- execute
- service
- deploy

Suricata

- suricata
- suricataBPF
- suricataRuleset
- suricataRulesetReload
- suristop
- suristart

- suripath
- suribin
- surirunning

Zeek

- zeek
- loaddatazeekpath
- loaddatazeekrunning

Wazuh

- wazuhStart
- wazuhStop
- loadDataWazuhPath
- loadDataWazuhBin
- loadDataWazuhRunning

Software TAP

- stap
- stapCollector
- stapPubKey

Data Bases

- stapConn
- groupConn
- pluginConn
- nodeConn
- monitorConn

Analyzer

- analyzer

If you need help

- email our support team - support@owlh.net
- join OwlH slack - OwlH Slack workspace
- ask for professional support and services - prohelp@owlh.net

OwlH - current v0.14.x - Mar - OwlH Changelog

documentation last updated - Jul 06, 2020

1.4.2.3 Analyzer configuration file

```
{
  "enable":true,
  "outputfile":"/var/log/owlh/alerts.json",
  "prefilterfile":"conf/prefilters.json",
  "postfilterfile":"conf/postfilters.json",
  "tagsfile":"conf/tags.json",
  "srcfiles": [
      "/var/log/suricata/eve.json",
      "/usr/local/zeek/logs/current/conn.log",
      "/usr/local/zeek/logs/current/dns.log"
  ],
  "feedfiles": [
      {
          "feedfile":"/usr/local/owlh/src/owlhnode/conf/feeds/otx.feed",
          "workers":4
      },
      {
          "feedfile":"/tmp/local.feed"
      },
      {
          "feedfile":"/usr/local/owlh/src/owlhnode/conf/feeds/xforce.feed",
          "workers":4
      },
      {
          "feedfile":"/usr/local/owlh/src/owlhnode/conf/feeds/falcon.feed",
          "workers":4
      }
  ]
}
```

1.4.3 OwlH API

The OwlH API is an open source **RESTful API** that allows for interaction with the `OwlH Master` and `OwlH Node` components from a web browser, command line tool like cURL or any script or program that can make web requests. The OwlH UI and APP relies on this totally. Use the API to easily perform everyday actions like adding a node, restarting the services or looking up status details.

1.4.3.1 OwlH MASTER RESTful API

Note: Work in progress.

1.4.3.2 OwlH NODE RESTful API

Note: Work in progress.

1.5 Troubleshooting

Warning: work in progress...

Note: If you are missing something in this documentation, please say hello in our slack #doc channel and let us know what is missing or should be good to have.

- join OwlH slack - OwlH Slack workspace

1.5.1 OwlH Node

Suricata doesn't create alerts

1.5.2 OwlH Master

1.5.3 OwlH UI

1.6 Looking for...

1.6.1 OwlH and Suricata

As usual, please keep in contact if there is any clarification or help needed.

- email our support team - support@owlh.net
- join OwlH slack - OwlH Slack workspace
- ask for professional support and services - prohelp@owlh.net

1.6.1.1 Main steps

- Install Suricata from OwlH Script
- **Default settings when you install from OwlH script**
  - configuration files
  - rules folder
  - bpf file and folder
  - socket - PID files
- **Choose between Suricata management models**
  - Manage by OwlH
  - Expert mode

1.6.1.2 Suricata output with OwlH

- Standard eve.json
- Socket output

1.6.1.3 Suricata Rules

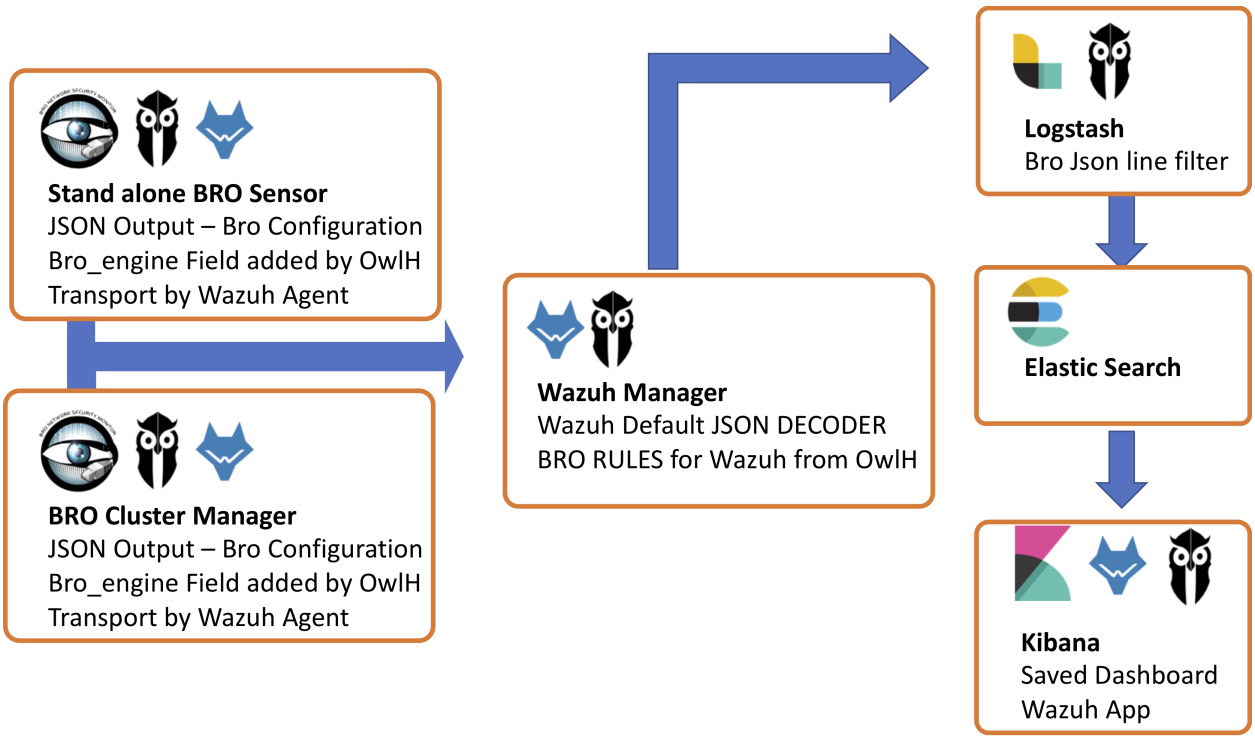
Use OpenRules to:

- create local rulesets based on 3rd party rulesets and custom rules
- synchronize each local ruleset with one or mode nodes
- schedule ruleset update
- edit rules from User Interface
- enable or disable rules
- search rules and find where are rules installed and stored

see OpenRules

1.6.2 OwlH and Zeek

1.6.2.1 Integration Logical Diagram



Components

- OwlH Node - Zeek IDS and Wazuh Agent
- Wazuh Manger
- Logstash Server
- Elastic and Kibana Server

Let's see what we need to modify on each component to be able to manage this Bro and Wazuh integration.

1.6.2.2 Configure - Zeek - OwlH Node

This system will require Bro working of course, and Wazuh agent installed. OwlH instructions will help to configure both Bro and Wazuh agent.

1.6.2.3 Zeek Logs Output format to JSON

Option 1 - Modify ASCII writer output

you can load the json_logs.bro configuration that will tell ASCII writer to write output in JSON format. You must include following line in your .bro configuration files. It can be /etc/bro/site/local.bro or you can follow our recomendation and write the configs in owlh.bro file (please, see below).

This will modify output and will store just json output, you won't have ASCII output.

```
@load tuning/json_logs.zeek
```

Zeek Event Enritchment to help Wazuh ruleset

It is a good idea to help wazuh rules to do their job, to include a field that will identify what kind of log line we are analyzing. Bro output doesn't include that info per line by default, so we are going to help wazuh by including the field 'bro_engine' that will tell wazuh what kind of log is it.

We are using redef function to include a custom field for each ::Info record of each Protocol. Here are just a few of them, we will include more by default in next releases.

```
redef record DNS::Info += {
    bro_engine:    string    &default="DNS"    &log;
};
redef record Conn::Info += {
    bro_engine:    string    &default="CONN"    &log;
};
redef record Weird::Info += {
    bro_engine:    string    &default="WEIRD"    &log;
};
redef record SSL::Info += {
    bro_engine:    string    &default="SSL"    &log;
};
redef record SSH::Info += {
    bro_engine:    string    &default="SSH"    &log;
};
```

Loading Zeek customizations at Zeek start

We include all OwlH customizations in OwlH_*.bro files, that helps to have a clear view of what OwlH does as well as we hope it will simplify configuration management.

Under /etc/bro/site we will create two files

- owlh.bro - Will include JSON call and @load for bro_engine field definition.
- owlh_types.bro - Will include all redef statments

You will only need to load OwlH.bro at the end of your local.bro file to include all these configurations

```
@load /etc/bro/site/OwlH.bro
```

owlh.bro looks like:

```
@load tuning/json-logs.zeek
@load owlh.zeek
```

and owlh.zeek:

```
redef record DNS::Info += {
    bro_engine:    string    &default="DNS"    &log;
};
redef record Conn::Info += {
    bro_engine:    string    &default="CONN"    &log;
```

(continues on next page)

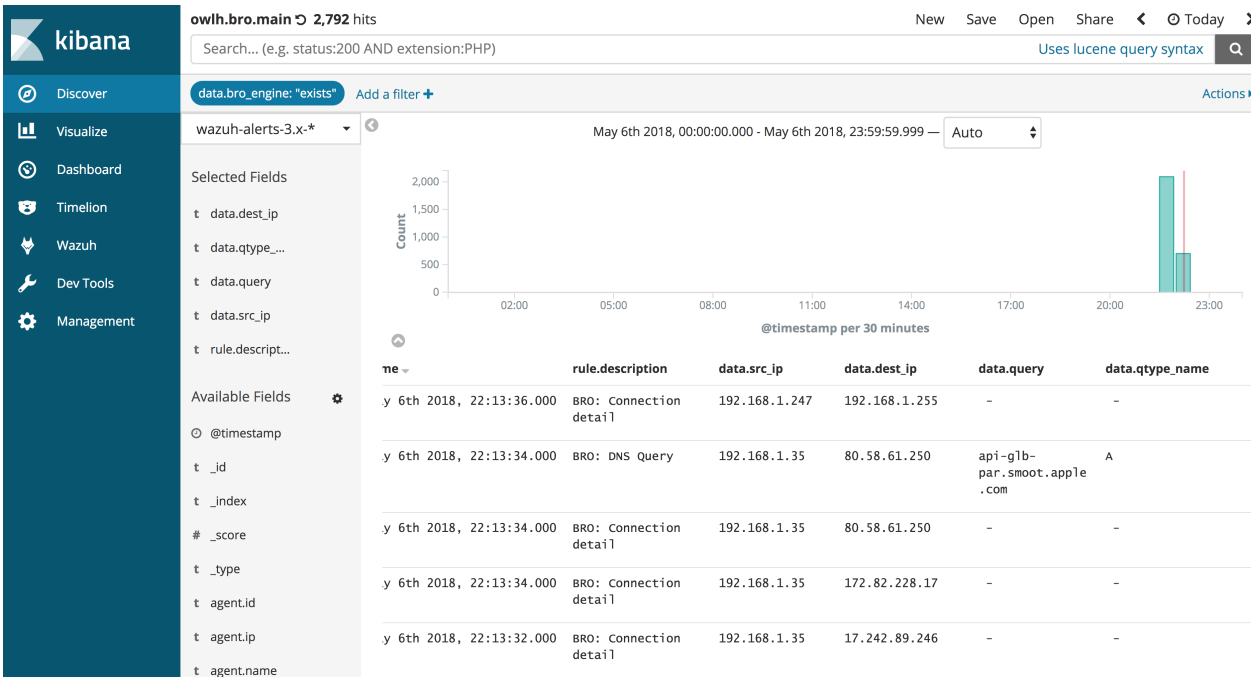
(continued from previous page)

```
};
redef record Weird::Info += {
    bro_engine:    string    &default="WEIRD"    &log;
};
redef record SSL::Info += {
    bro_engine:    string    &default="SSL"    &log;
};
redef record SSH::Info += {
    bro_engine:    string    &default="SSH"    &log;
};
```

1.6.2.4 Review your Kibana Dashboard

for integration with wazuh-elk you will need to verify that OwlH filebeat Module is loaded in Wazuh Manager servers and OwlH elasticsearch template and kibana dashboards are loaded.

Configure and integrate with Wazuh-ELk



And that's all folks.

1.6.3 OwlH and Moloch

1.6.3.1 Configure Moloch

- Install it on Master
- Install in a remote server

1.6.3.2 Moloch in Master

- Configure Moloch to read from owlh interface
- Configure STAP on Master to collect socket and replay to owlh interface

1.6.3.3 Moloch in remote server

- Configure NFS to publish a PCAP folder
- Configure Master to connect to Moloch server PCAP folder
- Configure OwlH Master Dispatcher to include Moloch PCAP folder in the pool
- Configure STAP on Master to collect socket and write to PCAP

1.7 Use Cases

What do you want to achieve with NIDS platform in your Network?

1.7.1 BASIC

- Monitor a single server traffic
- Monitor traffic from one or multiple network segments using a SPAN/Mirror Port

1.7.2 ADVANCED

- I have some remote/cloud servers but I can't use SPAN/Mirror facilities and I need to monitor server's traffic
- Transport traffic from remote servers in cloud environment for analysis, storage and forensic
- We have an hybrid cloud (AWS, Google Cloud, AZURE) and on-premises environment and need a centralized NIDS management and security view

1.7.3 INTEGRATE WITH WAZUH

Integrate with Wazuh

This will help you:

- Just send default Suricata alerts to Wazuh-ELK
- Unify Suricata and Zeek outputs, send to Wazuh-ELK and visualize with some cool dashboard

1.8 CHANGELOG

As per our latest version, in OwlH solution you may find:

OwlH - current v0.14.x - Mar - OwlH Changelog

1.8.1 If you need help

- email our support team - support@owlh.net
- join OwlH slack - OwlH Slack workspace
- ask for professional support and services - prohelp@owlh.net

OwlH - current v0.14.x - Mar - OwlH Changelog

documentation last updated - Jul 06, 2020

- email our support team - support@owlh.net
- join OwlH slack - OwlH Slack workspace

documentation last updated - Jul 06, 2020