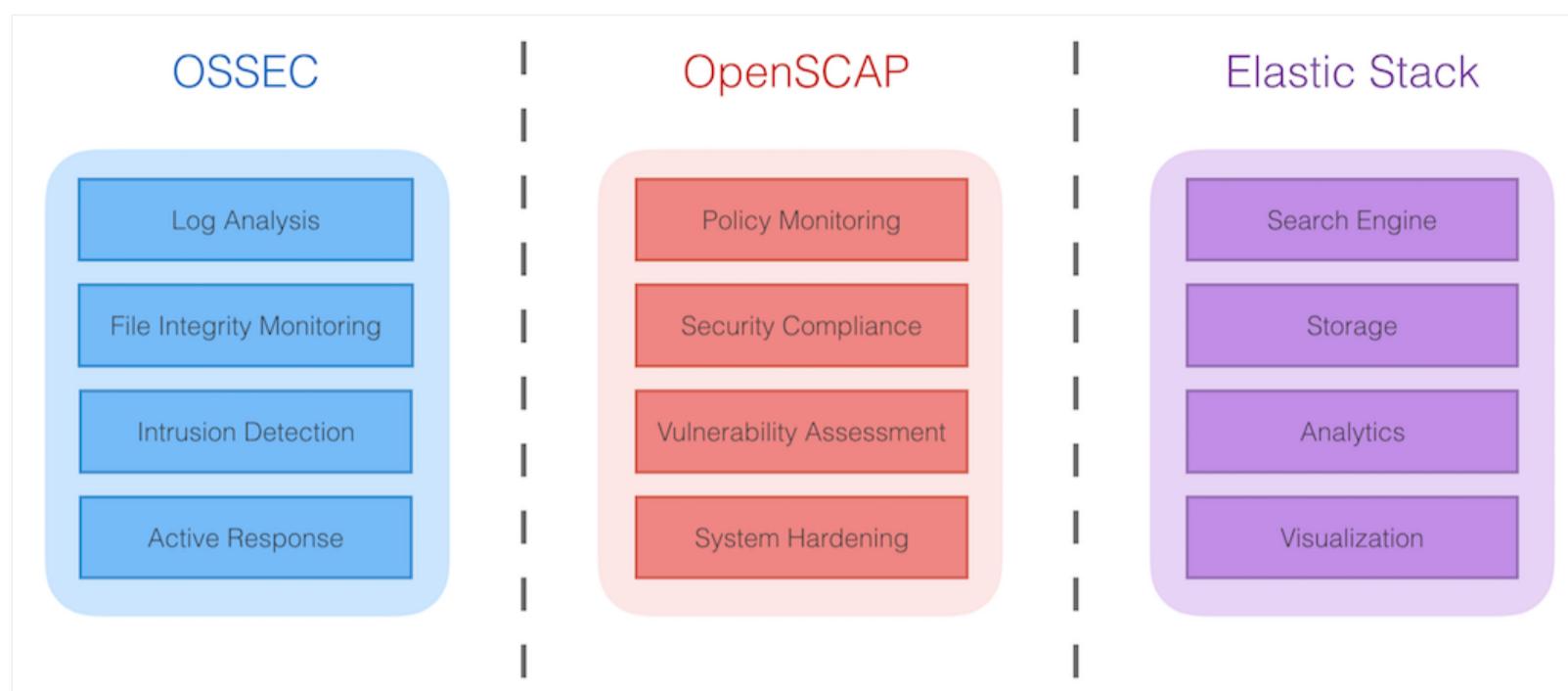




Documentation

Getting started

Wazuh is a security detection, visibility, and compliance open source project. It was born as a fork of OSSEC HIDS, later was integrated with Elastic Stack and OpenSCAP evolving into a more comprehensive solution. Below is a brief description of these tools and what they do:



OSSEC HIDS

[OSSEC HIDS](#) is a Host-based Intrusion Detection System (HIDS) used both for security detection, visibility, and compliance monitoring. It's based on a multi-platform agent that forwards system data (e.g log messages, file hashes, and detected anomalies) to a central manager, where it is further analyzed and processed, resulting in security alerts. Agents convey event data to the central manager via a secure and authenticated channel.

Additionally, OSSEC HIDS provide a centralized syslog server and an agentless configuration monitoring system, providing security insight into the events and changes on agentless devices such as firewalls, switches, routers, access points, network appliances, etc.

OpenSCAP

[OpenSCAP](#) is an [OVAL](#) (Open Vulnerability Assessment Language) and [XCCDF](#) (Extensible Configuration Checklist Description Format) interpreter used to check system configurations and to detect vulnerable applications.

It's a well-known tool designed to check the security compliance and hardening of the systems using industry standard security baselines for enterprise environments.

Elastic Stack

[Elastic Stack](#) is a software suite (Filebeat, Logstash, Elasticsearch, Kibana) used to collect, parse, index, store, search, and present log data. It provides a web frontend useful for gaining a high-level dashboard view of events, as well to realize advanced analytics and data mining deep into your store of event data.

Table of Contents

This document will help you understand Wazuh components and its architecture, Also, it will show you some common use cases.

- [Components](#)
 - [Wazuh agent](#)
 - [Wazuh server](#)
 - [Elastic Stack](#)
- [Architecture](#)
 - [Communications and data flow](#)
 - [Archival data storage](#)
- [Use cases](#)
 - [Signature-based log analysis](#)
 - [File integrity monitoring](#)
 - [Rootkits detection](#)

- o Security policy monitoring

Installation guide

This document will guide you through the installation process. For interactive help, our [email forum](#) is available. You can subscribe by sending an email to [Wazuh subscribe](#).

Wazuh installation involves two central components, the Wazuh server, and Elastic Stack. In addition, Wazuh agents will need to be deployed to the monitored hosts in your environment:

- **Wazuh server:** Runs the Wazuh manager, API and Filebeat (only necessary in distributed architecture). Collects and analyzes data from deployed agents.
- **Elastic Stack:** Runs the Elasticsearch engine, Logstash server and Kibana (including the Wazuh App). It reads, parses, indexes, and stores alert data generated by the Wazuh server.
- **Wazuh agent:** Runs on the monitored host, collecting system log and configuration data, and detecting intrusions and anomalies. It talks with the Wazuh server, to which it forwards collected data for further analysis.

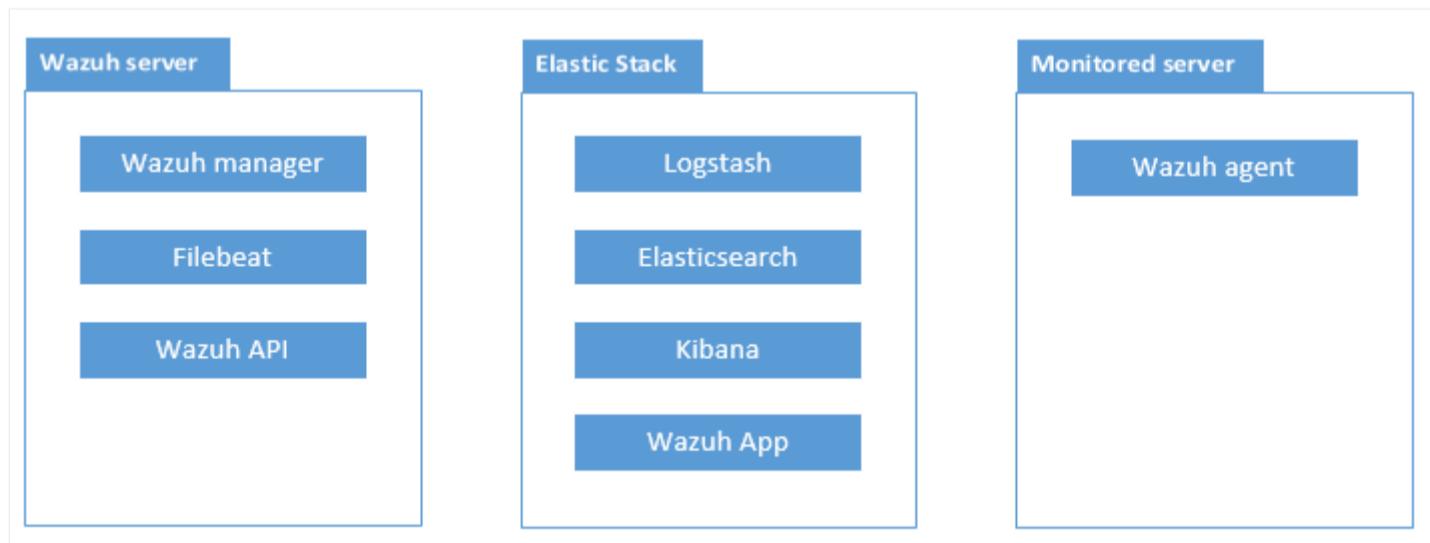
Distributed architectures do run the Wazuh server and Elastic Stack cluster (one or more servers) on different hosts. On the other hand, single-host architectures have Wazuh server and Elastic Stack installed in the same system. This guide covers both installation options.

The diagrams below list the components that are run per host, both for single-host and distributed architectures.

Single-host architecture:



Distributed architecture:



Note

Before installing the components please confirm time synchronization service is configured and working on your servers. This is most commonly done with **NTP**. More info for [Debian/Ubuntu](#) and [CentOS/RHEL/Fedora](#).

Contents

- [Installing Wazuh server](#)
- [Installing Elastic Stack](#)

- [Installing Wazuh agent](#)
- [Optional configurations](#)
 - [Setting up SSL for Filebeat and Logstash](#)
 - [Setting up SSL and authentication for Kibana](#)
 - [Securing the Wazuh API](#)
 - [Elasticsearch tuning](#)
- [Upgrading Wazuh](#)
- [Virtual Machine](#)
- [Packages List](#)
 - [Windows](#)
 - [Mac OS X](#)
 - [Red Hat](#)
 - [CentOS](#)
 - [Fedora](#)
 - [SUSE](#)
 - [Ubuntu](#)
 - [Debian](#)
 - [Solaris](#)
 - [OVA Wazuh 2.1.1 + ELK 5.6.3](#)
- [Unattended Installation](#)
 - [Global](#)
 - [Agent](#)
 - [Manager/local](#)
 - [API](#)

Docker

[Docker](#) is an open-source project that automates the deployment of different applications inside software containers. Docker containers wrap up a piece of software in a complete filesystem that contains everything it needs to run like: code, system tools, libraries, etc. This process guarantees that the system will always run the same, regardless the environment it is running.

We have created our own fork based on "[deviantony](#)" dockerfiles and "[xetus-oss](#)" dockerfiles. Thank you, Anthony Lapenna, for your contribution to the community. If you want to contribute to the Wazuh fork, please go to the [Wazuh docker repository](#)

The images we created are in the [Docker hub](#). You can install Wazuh with a single-host architecture using a set of Docker images that contains [Wazuh Manager](#), [Wazuh API](#), [Filebeat](#), [Logstash](#), [Elasticsearch](#), [Kibana](#)

This section will show you the process:

- [Docker installation](#)
 - [Docker engine](#)
 - [Docker compose](#)
- [Wazuh container](#)
 - [Requirements](#)
 - [Usage](#)
- [FAQ](#)
 - [How can I tune the Kibana configuration?](#)
 - [How can I tune the Logstash configuration?](#)
 - [How can I specify the amount of memory used by Logstash?](#)
 - [How can I tune the Elasticsearch configuration?](#)
 - [How can I store Wazuh data?](#)
 - [How can I store Elasticsearch data?](#)

Deploying with Puppet

Puppet is an open-source software tool that gives you an automatic way to inspect, deliver, operate and future-proof all of your software, no matter where it is executed. It runs on many Unix-like systems as well as on Microsoft Windows, and includes its own declarative language to describe system configuration. It is very simple to use and allows you to install and configure Wazuh easily.

Contents

- [Set up Puppet](#)
 - [Installing Puppet master](#)
 - [Installing Puppet agent](#)
 - [Setting up Puppet certificates](#)
- [Wazuh Puppet module](#)
 - [Install Wazuh module](#)
 - [Install manager via Puppet](#)
 - [Install agent via Puppet](#)
 - [Reference Wazuh puppet](#)

Deploying with Ansible

Ansible is an open source platform designed for automating tasks. It comes with Playbooks, a descriptive language based on YAML, that make easy to create and describe automation jobs. Also, Ansible communicates with every host over SSH, making it very secure. See [Ansible Overview](#) for more info.

Contents

- [Considerations](#)
- [Install Ansible](#)
 - [OpenSSH Compatibility](#)
 - [Windows hosts](#)
 - [Installation on CentOS/RHEL/Fedora](#)
 - [Installation on Debian/Ubuntu](#)
- [Remote Hosts](#)
 - [Using passwords](#)
 - [Windows authentication](#)
 - [Using SSH key-pairing](#)
 - [Installing public key](#)
 - [Add hosts to control](#)
 - [Test connection](#)
- [Roles](#)
 - [Wazuh Manager](#)
 - [Filebeat](#)
 - [Elasticsearch](#)
 - [Kibana](#)
 - [Logstash](#)
 - [Wazuh Agent](#)
- [Variables references](#)
 - [Elasticsearch](#)
 - [Kibana](#)
 - [Logstash](#)
 - [Filebeat](#)
 - [Wazuh Manager](#)
 - [Wazuh Agent](#)

Using Wazuh for PCI DSS

The **Payment Card Industry Data Security Standard (PCI DSS)** is a proprietary information security standard for organizations that handle branded credit cards from the major card companies including *Visa*, *MasterCard*, *American Express*, *Discover*, and *JCB*. The standard was created to increase controls around cardholder data to reduce credit card fraud.

Wazuh helps to implement PCI DSS by performing log analysis, file integrity checking, policy monitoring, intrusion detection, real-time alerting and active response. This guide explains how these capabilities help with each of the standard requirements:

- [Wazuh for PCI DSS Guide \(PDF\)](#)
- [Wazuh for PCI DSS Guide \(Excel\)](#)

In the following section, we will elaborate on some specific use cases. They explain how to use Wazuh capabilities to meet the standard requirements.

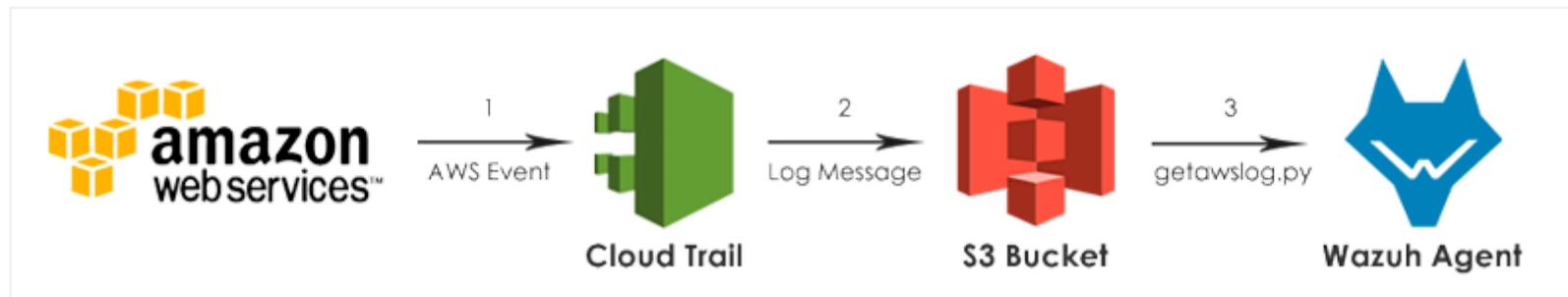
Contents

- [Log analysis](#)
- [Policy monitoring](#)
- [Rootkit detection](#)
- [File integrity monitoring](#)
- [Active response](#)
- [Elastic Stack](#)

Using Wazuh for AWS

This section provides instructions to integrate Wazuh with Amazon Web Services (AWS). It also explains different use cases as examples of how the rules developed by Wazuh can be used to alert on specific events from IAM, EC2 and VPC.

The diagram below shows how a log message about an AWS event flows from AWS to a Wazuh agent. Once the agent reads the message, it sends it to the Wazuh manager which analyses it with decoders and rules. When a rule matches, an alert is triggered if the rule severity is high enough.



1. CloudTrail is a web service that records AWS API calls for your account and writes them to log files. When an AWS event occurs, CloudTrail generates the log message. Using CloudTrail we can get more visibility into AWS user activity, tracking changes made to AWS resources.
2. Once an event takes place, CloudTrail writes it to a log file on Amazon S3, where log files can be stored durably and inexpensively.
3. The script `getawslog.py` downloads CloudTrail log files from Amazon S3 to the Wazuh agent, uncompresses them and appends the new data to a local text file which is monitored by the Wazuh agent and forwarded to the Wazuh manager just like any other log file.

This diagram makes it easier to understand the integration process described in the upcoming pages.

Contents

- [Integration with AWS](#)
 - [Turn on CloudTrail](#)
 - [Create a user with permission to access S3](#)
 - [Install Python Boto on your Wazuh manager](#)
 - [Configure user credentials with Python Boto](#)
 - [Run the python script to download the JSON data](#)
 - [Collect AWS log data](#)
- [Use Cases](#)
 - [IAM use cases](#)
 - [EC2 use cases](#)
 - [VPC Use cases](#)

Migrating from OSSEC

This document describes how to migrate your existing OSSEC installation (agent or manager) to Wazuh. For interactive help, our [email forum](#) is available. You can subscribe by sending an email to wazuh+subscribe@googlegroups.com.

! Note

OSSEC agents are compatible with Wazuh manager, but if you don't migrate your agents to Wazuh, you will lose some capabilities like [OpenSCAP](#) or some [syscheck features](#) in those agents.

The migration of Elastic stack, in the case that you already have it installed, is beyond the scope of Wazuh documentation. We recommend you visit our guides for [Installing Elastic Stack](#).

Follow the appropriate section depending on the type of your OSSEC installation:

Upgrade from	Type	Installation type	Upgrade to	Guide
OSSEC 2.8.3+	Manager	Packages	Wazuh 2.0	Migrating OSSEC manager installed from packages
OSSEC 2.8.3+	Manager	Sources	Wazuh 2.0	Install Wazuh server with RPM packages
				Install Wazuh server with Deb packages
OSSEC 2.8.3+	Agent	Packages	Wazuh 2.0	Migrating OSSEC agent installed from packages
OSSEC 2.8.3+	Agent	Sources	Wazuh 2.0	Install Wazuh agent with RPM packages
				Install Wazuh agent with Deb packages

! Warning

For cases where OSSEC was installed from sources, the configuration file `/var/ossec/etc/ossec.conf` **will be overwritten**. The *old* configuration file from the current installation is saved as `ossec.conf.rpmorig` or `ossec.conf.deborig`. You should compare the new file with the old one. Also, a backup of your previous ruleset will be saved at `/var/ossec/etc/backup_ruleset`. All the rules/decoders in files other than `local_rules.xml` or `local_decoder.xml` will be overwritten.

Release Notes

This section summarizes the most important features of each release.

Contents

- [2.1 Release Notes](#)
 - [Anti-flooding mechanism](#)
 - [Labels for agent alerts](#)
 - [Improved Authd performance](#)
 - [New features for internal logs](#)
 - [Updated external libraries](#)
 - [Wazuh API](#)
 - [Ruleset](#)

Installing Wazuh server

The Wazuh server can be installed on any Unix-like operating system, most commonly on Linux. It is more convenient install it via packages if one is available for your distribution. However, build and install it from sources is also pretty simple.

There are two components that you usually have to install on a Wazuh server: the manager and the API. In addition, for distributed architectures (where the Wazuh server sends data to a remote Elastic Stack cluster) you will need to install Filebeat.

There are several options to install a Wazuh server, depending on the operating system and whether or not you wish to build from source. Consult the table below and choose how to proceed:

Type	Description
RPM packages	Install Wazuh server on CentOS/RHEL/Fedora.
DEB packages	Install Wazuh server on Debian/Ubuntu.
Sources	Install Wazuh server from source code.

Note

Installing Wazuh Server on a 64-bit operating system is highly recommended since the Wazuh API is not available on 32-bit platforms. Without the Wazuh API, much of the functionality of the Wazuh Kibana App will not work. Similarly, if you are using Red Hat or CentOS for your Wazuh Server platform, make sure it is version 6 or higher to properly install Wazuh API.

Installing Elastic Stack

This guide describes the installation of an Elastic Stack server composed by Logstash, Elasticsearch, and Kibana. We will illustrate package-based installations of these components. You can also install them from binary tarballs, however, this is not preferred or supported under Wazuh documentation.

In addition to Elastic Stack components, you will also find here the instructions to install and configure the Wazuh App (deployed as a Kibana plugin).

Depending on your operating system you can choose to install Elastic Stack from RPM or DEB packages. Consult the table below and choose how to proceed:

Type	Description
RPM packages	Install Elastic Stack on CentOS/RHEL/Fedora.
DEB packages	Install Elastic Stack on Debian/Ubuntu.

Installing Wazuh agent

The Wazuh agent runs on the hosts that you want to monitor. It is multi-platform and provides the following capabilities: log and data collection, file integrity monitoring, rootkits and malware detection, and security policy monitoring. In addition, it talks to the Wazuh manager, sending data in near real-time through an encrypted and authenticated channel.

There are several options to install a Wazuh agent, depending on the operating system and whether or not you wish to build from source. Consult the table below and choose how to proceed for a given agent:

Type	Description
RPM packages	Install Wazuh agents on CentOS/RHEL/Fedora.
DEB packages	Install Wazuh agents on Debian/Ubuntu.
Windows installer	Install Wazuh agents on Windows.
Mac OS installer	Install Wazuh agents on Mac OS
Solaris installer	Install Wazuh agents on Solaris
Sources	Install Wazuh agents from source code.

Note

Deploying agents to a large number of servers or endpoints can be easier using automation tools like Puppet, Chef, SCCM or Ansible. Consider exploring these options if that is your case.

Optional configurations

This section advises on best practices related to setting up an efficient, stable and secure Wazuh environment. Here you can find how to set up an SSL communication in distributed architectures (where a Wazuh server talks to an Elastic Stack cluster), how to use Nginx to set up a secure proxy for Kibana plus adding authentication to your web user interface.

Contents

- [Setting up SSL for Filebeat and Logstash](#)
- [Setting up SSL and authentication for Kibana](#)
- [Securing the Wazuh API](#)
- [Elasticsearch tuning](#)

Upgrading Wazuh

This section describes how to upgrade an existing Wazuh installation. The upgrade process depends on the version that is currently installed and the version that you want to upgrade to:

Upgrade from	Upgrade to	Supported Upgrade Type
1.x	2.x	Upgrade from a legacy version
2.0.x	2.0.y	Upgrade from the same minor version (where $y > x$)
2.0.x	2.1.x	Upgrade from the same major version

⚠ Warning

Wazuh v2 uses different index names and templates than Wazuh v1. For that reason, you will not be able to see the alerts previous to the upgrade in Wazuh App. If you need to access them, you will need to reindex the previous indices.

💡 Note

If you find any issue during the upgrade process, feel free to ask for help in our [mailing list](#). The Wazuh team and other users of the Open Source community may be able to assist you.

Virtual Machine

We provide a pre-built virtual machine image (OVA), you can directly import using VirtualBox (where has been built it) and other OVA compatible virtualization system as well.

! Note

This VM only run on 64-bit systems and is not recommended to use in production environments. Instead, it can be a useful tool for proof of concepts and labs. Distributed architectures and multi-node Elastic Stack clusters are usually a better fit for production environments, where higher performance is required.

1. This virtual appliance, available at https://packages.wazuh.com/vm/wazuh2.1.1_5.6.3.ova, contains the following components:

- CentOS 7
- Wazuh 2.1.1-1
- RESTful API 2.1.1-1
- Elasticsearch 5.6.3
- Logstash 5.6.3
- Kibana 5.6.3 port “**5601**”
- WazuhAPP 2.1.1_5.6.3

2. Import the OVA in your virtualization platform, and run the virtual machine. The root password is “**wazuh**” and the username/password for the Wazuh API is “**foo/bar**”.

Although you don't need to change any Elastic Stack configuration settings, feel free to explore the options. You can find Elasticsearch installed in `/usr/share/elasticsearch`. Similarly, Logstash is installed in `/usr/share/logstash` and its config directory is `/etc/logstash/conf.d/`.

3. **Wazuh Manager** and the **Elastic Stack** are configured to work out of the box. You can now deploy the Wazuh agents, on those systems that you intend to monitor, and connect them to your virtual appliance. More documentation at:

- [How to install Wazuh agents.](#)

⚠ Warning

Before connecting any Wazuh agent, change the VM's network interface type from NAT (the factory default) to bridge for be reachable to your network. By default, the VM will try to get an IP address from your network's DHCP server. Alternatively, you can set a static IP address by configuring the proper network files on the CentOS operating system where it's this virtual machine based on.

4. You can start and stop wazuh-manager, elasticsearch, logstash, and kibana with the ‘systemctl’ command. Examples:

```
$ systemctl restart wazuh-manager  
$ systemctl stop elasticsearch  
$ systemctl start logstash  
$ systemctl status kibana
```

5. In order to connect to Kibana web user interface, you can login with http://OVA_IP_ADDRESS:5601 (where `OVA_IP_ADDRESS` is your system IP).

Unattended Installation

Unattended installation saves the user having to interact with the installation interface to complete the process, allowing the automation of agents deployments. To do this, you must modify the `preloaded-vars.conf` file uncommenting the configuration lines that you want to automate in the installation process.

- [Global](#)
- [Agent](#)
- [Manager/local](#)
- [API](#)

Global

	Defines the language to be used.	
USER_LANGUAGE	Allowed values	"en", "br", "cn", "de", "el", "en", "es", "fr", "hu", "it", "jp", "nl", "pl", "ru", "sr", "tr"
USER_NO_STOP	If it is set to anything, the confirmation messages are not going to be asked for.	
	Defines the role for the Wazuh instance that is being installed.	
USER_INSTALL_TYPE	Allowed values	"local", "agent", "server"
	Defines the location to install Wazuh.	
USER_DIR	Allowed values	Any path
	If it is set to "y", the directory to install Wazuh will be removed if exists.	
USER_DELETE_DIR	Allowed values	"y", "n"
	If it is set to "n", active response will be disabled.	
USER_ENABLE_ACTIVE_RESPONSE	Allowed values	"y", "n"
	If it is set to "n", syscheck will be disabled.	
USER_ENABLE_SYSCHECK	Allowed values	"y", "n"
	If it is set to "n", rootcheck will be disabled.	
USER_ENABLE_ROOTCHECK	Allowed values	"y", "n"
	If it is set to "n", OpenSCAP will be disabled.	
USER_ENABLE_OPENSCAP	Allowed values	"y", "n"
USER_UPDATE	If it is set to anything, the update installation will be done.	
USER_BINARYINSTALL	If it is set to anything, the installation is not going to compile the code, but use the binaries from ./bin/	

Agent

USER_AGENT_SERVER_IP	Specifies the IP address of the Wazuh server.
USER_AGENT_SERVER_NAME	Specifies the hostname of the Wazuh server.

USER_AGENT_CONFIG_PROFILE

Specifies the agent's config profile name. This is used to create a configuration profiles for this particular profile name.

Example:

```
USER_LANGUAGE="en"
USER_NO_STOP="y"
USER_INSTALL_TYPE="agent"
USER_DIR="/var/ossec"
USER_ENABLE_SYSCHECK="y"
USER_ENABLE_ROOTCHECK="y"
USER_ENABLE_OPENSCAP="y"
USER_ENABLE_ACTIVE_RESPONSE="y"
```

Manager/local

USER_ENABLE_EMAIL	Enables or disables alerts by e-mail.	
	Allowed values	"y", "n"
USER_EMAIL_ADDRESS	Defines the destination e-mail for the alerts.	
	Allowed values	A valid e-mail address.
USER_EMAIL_SMTP	Defines the SMTP server to send the e-mails.	
	Allowed values	A valid SMTP server.
USER_ENABLE_SYSLOG	Enables or disables remote syslog.	
	Allowed values	"y", "n"
USER_WHITE_LIST	List of IPs or networks that are going to be set to never be blocked.	

Example:

```
USER_LANGUAGE="en"
USER_NO_STOP="y"
USER_INSTALL_TYPE="server"
USER_DIR="/var/ossec"
USER_ENABLE_EMAIL="n"
USER_ENABLE_SYSCHECK="y"
USER_ENABLE_ROOTCHECK="y"
USER_ENABLE_OPENSCAP="y"
USER_WHITE_LIST="n"
USER_ENABLE_SYSLOG="y"
```

API

Parameters for `install_api.sh`:

REINSTALL	Reinstall Wazuh.	
	Allowed values	"y", "n"
REMOVE	Remove current installation.	
	Allowed values	"y", "n"
DIRECTORY	Installation directory.	
	Allowed values	Any path

Parameters for `configure_api.sh`:

PORT	The port used to connect to the Wazuh API.	
	Allowed values	Any valid port.
HTTPS	Enable HTTPS.	
	Allowed values	"y", "n"
AUTHD	Enable authd authentication.	
	Allowed values	"y", "n"
PROXY	Change proxy.	
	Allowed values	"y", "n"

Parameters for certificate generation:

COUNTRY	Certificate country.
STATE	Certificate state.
LOCALITY	Certificate locality.
ORG_NAME	Organization name.
ORG_UNIT	Organitation unit name.
COMMON_NAME	Common Name.
PASSWORD	Certificate password.

Parameters for basic auth:

USER	API user.
PASS	API password.

! Note

To automate deployments in Windows you can use the parameters of its [installer](#).

Docker installation

The first thing you need to do is install Docker if you don't have it already.

Docker engine

Docker requires a 64-bit operating system running kernel version 3.10 or higher.

To check your current kernel version, open a terminal and use `uname -r` to display your kernel version:

```
$ sudo uname -r  
3.10.0-229.el7.x86_64
```

Run the Docker installation script.

```
# curl -sSL https://get.docker.com/ | sh
```

If you would like to use Docker as a non-root user, you should now consider adding your user to the “docker” group with something like:

```
$ sudo usermod -aG docker your-user
```

Note

Remember that you will have to log out and log back in for this to take effect.

Docker compose

Docker compose 1.6 or newer is required. Install it like this:

```
# curl -L "https://github.com/docker/compose/releases/download/1.12.0/docker-compose-$(uname -s)-$(uname -m)"  
-o /usr/local/bin/docker-compose  
# chmod +x /usr/local/bin/docker-compose
```

Test the installation:

```
$ docker-compose --version  
docker-compose version 1.12.0, build b31ff33
```


Wazuh container

Requirements

Increase max_map_count on your host (Linux)

You need to increase `max_map_count` on your Docker host:

```
$ sudo sysctl -w vm.max_map_count=262144
```

To set this value permanently, update the `vm.max_map_count` setting in `/etc/sysctl.conf`. To verify after rebooting, run "sysctl `vm.max_map_count`".

⚠ Warning

If you don't set the **max_map_count** on your host, Elasticsearch will probably don't work.

Increase max_map_count on your host (Windows)

You need to increase `max_map_count` on your Docker host:

```
$ docker-machine ssh default
$ sudo sysctl -w vm.max_map_count=262144
$ exit
```

To set this value permanently, update the `vm.max_map_count` setting in `/var/lib/boot2docker/profile`:

```
$ docker-machine ssh default
$ sudo vi /var/lib/boot2docker/bootlocal.sh
```

Add the following line into the profile file:

```
sysctl -w vm.max_map_count=262144
```

Make the script runnable:

```
$ sudo chmod +x /var/lib/boot2docker/bootlocal.sh
```

To verify after rebooting, run "sysctl `vm.max_map_count`".

⚠ Warning

If you don't set the **max_map_count** on your host, Elasticsearch will probably don't work.

SELinux

On distributions which have SELinux enabled out-of-the-box, you will need to either re-context the files or put SELinux into Permissive mode for docker-elk to start properly. For example, on Red Hat and CentOS the following command will apply the proper context

```
# chcon -R system_u:object_r:admin_home_t:s0 docker-elk/
```

Docker for OSX

In Docker for OSX, there is a default memory limit of 2GB, in order to run *docker-compose up* successfully you have to change default memory settings from 2GB to at least 4 or 5GB. To do so, click on the Docker icon in the menu bar, then “Preferences...”, go to “Advanced” tab and set 5GB of memory, then click on “Apply & Restart” and run *docker-compose up*.

Usage

1. Copy the *docker-compose* file to your system:

```
$ curl -so docker-compose.yml https://raw.githubusercontent.com/wazuh/wazuh-docker/master/docker-compose.yml
```

2. Start Wazuh and Elastic Stack using *docker-compose*:

a. Foreground:

```
$ docker-compose up
```

b. Background:

```
$ docker-compose up -d
```

Then access the Kibana UI by hitting <http://localhost:5601> with a web browser.

By default, the stack exposes the following ports:

1514	Wazuh UDP
1515	Wazuh TCP
514	Wazuh UDP
55000	Wazuh API
5000	Logstash TCP input
9200	Elasticsearch HTTP
9300	Elasticsearch TCP transport
5601	Kibana

Note

Configuration is not dynamically reloaded, so you will need to restart the stack after any change in the configuration of a component.

FAQ

How can I tune the Kibana configuration?

The Kibana default configuration is stored in `kibana/config/kibana.yml` .:

```
kibana:  
  image: wazuh/wazuh-kibana  
  hostname: kibana  
  restart: always  
  ports:  
    - "5601:5601"  
  networks:  
    - docker_elk  
  depends_on:  
    - elasticsearch  
  environment:  
    - "WAZUH_KIBANA_PLUGIN_URL=http://your.repo/wazuhapp-2.0_5.3.0.zip"  
  entrypoint: sh wait-for-it.sh elasticsearch
```

How can I tune the Logstash configuration?

The logstash configuration is stored in `logstash/config/logstash.conf` .

The `logstash/config` folder is mapped onto the `/etc/logstash/conf.d` container so that you can create more than one file in that folder if you'd like to. However, you must be aware that config files will be read from that directory in alphabetical order.

How can I specify the amount of memory used by Logstash?

The Logstash container uses the `LS_HEAP_SIZE` environment variable to determine how much memory should be allocated as JVM heap memory (defaults to 2048m).

If you want to override the default configuration, edit the `LS_HEAP_SIZE` environment variable defined in the logstash section of `docker-compose.yml` :

```
logstash:  
  image: wazuh/wazuh-logstash:latest  
  command: -f /etc/logstash/conf.d/  
  volumes:  
    - ./logstash/config:/etc/logstash/conf.d  
  ports:  
    - "5000:5000"  
  networks:  
    - docker_elk  
  depends_on:  
    - elasticsearch  
  environment:  
    - LS_HEAP_SIZE=2048m
```

How can I tune the Elasticsearch configuration?

The Elasticsearch container uses the default configuration and it is not exposed by default.

If you want to override the default configuration, create a file `elasticsearch/config/elasticsearch.yml` and put your custom version of the configuration in it.

Then map your configuration file inside the container in the `docker-compose.yml` . Update the elasticsearch container declaration to:

```
elasticsearch:  
  image: wazuh/wazuh-elasticsearch:latest  
  ports:  
    - "9200:9200"  
    - "9300:9300"  
  environment:  
    ES_JAVA_OPTS: "-Xms1g -Xmx1g"  
  networks:  
    - docker_elk
```

How can I store Wazuh data?

The data stored in Wazuh will persist after container reboots but not after container removal.

In order to preserve Wazuh data even after removing the Wazuh container, you'll have to mount a volume on your Docker host. Update the Wazuh container declaration in the [docker-compose.yml](#) to look like this:

```
elasticsearch:  
  image: wazuh/wazuh:latest  
  hostname: wazuh-manager  
  ports:  
    - "1514:1514/udp"  
    - "1515:1515"  
    - "514:514"  
    - "55000:55000"  
  networks:  
    - docker_elk  
  volumes:  
    - /path/to/storage:/var/ossec/data
```

This will store Wazuh data inside `/path/to/storage` in the Docker host's local file system.

How can I store Elasticsearch data?

The data stored in Elasticsearch will persist after container reboots but not after container removal.

In order to preserve Elasticsearch data even after removing the Elasticsearch container, you'll have to mount a volume on your Docker host. Update the elasticsearch container declaration in the [docker-compose.yml](#) file to look like this:

```
elasticsearch:  
  image: wazuh/wazuh-elasticsearch:latest  
  hostname: elasticsearch  
  command: elasticsearch -Des.network.host=_non_loopback_ -Des.cluster.name: my-cluster  
  ports:  
    - "9200:9200"  
    - "9300:9300"  
  environment:  
    ES_JAVA_OPTS: "-Xms1g -Xmx1g"  
  networks:  
    - docker_elk  
  volumes:  
    - /path/to/storage:/usr/share/elasticsearch/data
```

This will store elasticsearch data inside `/path/to/storage` in the Docker host's local file system.

Considerations

Before we get started with Ansible, confirm the following network requirements are met:

- **Private network DNS:** If you intended to use hostname instead of IP Address for remote hosts definitions, be sure you have correctly setup your own DNS server and it responds correctly to your hosts FQDN hostname, otherwise use your hosts file.
- **Firewall open ports:** Ansible can work with any TCP port, by default Ansible uses TCP/22 port to work with Linux hosts, be sure this port is open in hosts and/or firewalls.

Install Ansible

OpenSSH Compatibility

Ansible version 1.3 and later uses native OpenSSH for remote communication and, also, uses ControlPersist, a feature available for OpenSSH v5.6. This will increase performance by speeding up SSH Session Creation, which is very useful for Ansible. Otherwise, you will need to consider the setup [Accelerated Mode](#) on Ansible.

Windows hosts

Windows hosts are supported by Ansible from version 1.7 via the remote execution of PowerShell. As opposed to Linux hosts, it is necessary to do some pre-work before being able to use Ansible in Windows hosts. Please refer to [Windows Support](#) on Ansible official documentation. Consider the following minimum requirements:

- [Pywinrm](#) version 0.2.2 or later is required for Ansible control machine.
- PowerShell 3.0 or later is required to be able to run Ansible modules on Windows hosts.

Installation on CentOS/RHEL/Fedora

Install using yum from [EPEL](#). Only CentOS/RedHat version 6 or 7, and Fedora distributions, are currently supported. Follow the next steps:

1. Install EPEL repository:

```
$ sudo yum -y install epel-release
```

2. Install ansible:

```
$ sudo yum install ansible
```

Installation on Debian/Ubuntu

For Debian and Ubuntu we will use the Ansible PPA repository. Follow the next steps:

1. Install required dependencies:

```
$ sudo apt-get update  
$ sudo apt-get install lsb-release software-properties-common
```

2. Setup ansible repository:

a. For Ubuntu:

```
sudo apt-add-repository -y ppa:ansible/ansible  
sudo apt-get update
```

b. For Debian:

```
echo "deb http://ppa.launchpad.net/ansible/ansible/ubuntu trusty main" | sudo tee -a  
/etc/apt/sources.list.d/ansible-debian.list  
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 93C4A3FD7BB9C367  
sudo apt-get update
```

3. Finally, install ansible:

```
$ sudo apt-get install ansible
```

Remote Hosts

Ansible was born with the idea to be an agentless automation platform. Ansible relies on SSH the connection to remote hosts, meaning that, you can connect to remote hosts as SSH does. To follow, we briefly explain two (2) of this methods.

Note

We recommend the use of [Using passwords](#) method, this avoid you share your public SSH Key among several hosts.

Using passwords

Ansible does most of the work via SSH, SSH share their authentication mechanisms with Ansible, so, in order to establish a connection with remote hosts, a user/password must be supplied. To follow there is a description of some useful options to use for SSH authentication:

```
-u <user> Set the connection user.  
-k Ask the password of the connection user.  
-b Execute task and operations with a privilege user.  
-K Ask for sudo password, intended for privilege escalation.
```

You can use the above args as follows:

```
ansible -m setup all -u foo -k -b -K
```

This will set the connection user as `foo`. Also, it will ask for the connection user password and privileged user password.

Windows authentication

Windows hosts use a different mechanism to perform authentication. Please refer to [Authentication Options](#) in order to setup the adequate option.

Using SSH key-pairing

You can setup a SSH key-pairing to provide a silent auth mechanism, first create a OpenSSH key-pair:

```
$ ssh-keygen
```

To improve security on this setup, please ensure you provide a password for this key.

```
Enter passphrase (empty for no passphrase): *****  
Enter same passphrase again: *****
```

Using ssh-agent, avoid asking the key password over and over again on every Ansible deploy. Ssh-agent will cached your key to be use in further actions, until you logout.

Installing public key

After creating the Control machine key, you need to install the public key into every remote hosts, copy the content of `.ssh/id_rsa.pub` of Control machine to `.ssh/authorized_keys` on your host. Make sure you know the user to store `authorized_keys`, this will be the user you use for any action via Ansible.

Set the correct permissions:

```
$ chmod 600 .ssh/authorized_keys
```

Add hosts to control

Adding hosts is easy, just put the hostname or IP Address on `/etc/ansible/hosts`.

```
$ cat /etc/ansible/hosts

hosts1.example.net
hosts2.example.net
```

Also, you can group hosts. This could be useful to execute tasks and roles to several hosts at once:

```
$ cat /etc/ansible/hosts

[wazuh-elasticsearch]
hosts1.example.net
hosts2.example.net
```

Note

You can see the [Ansible inventory documentation](#) for more info regarding hosts and groups.

Test connection

This will attempt a connection with the remote hosts using `ping` module.

```
$ ansible all -m ping
```

You will get a output like this.

```
hosts1.example.net | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
hosts2.example.net | SUCCESS => {
    "changed": false,
    "ping": "pong"
}
```

If you see the above, then Ansible is fully usable.

Roles

You can use these roles to deploy Elastic Stack components, Wazuh API, Wazuh Manager and Wazuh Agents, first clone our [GitHub repository](#) directly to your Ansible roles folder:

```
$ cd /etc/ansible/roles  
$ git clone https://github.com/wazuh/wazuh-ansible.git .
```

Below we explain briefly how to use these roles, please check out [Ansible Playbooks](#) for more information.

Contents

- [Wazuh Manager](#)
- [Filebeat](#)
- [Elasticsearch](#)
- [Kibana](#)
- [Logstash](#)
- [Wazuh Agent](#)

Variables references

Elasticseach

elasticsearch_cluster_name

Name of the Elasticsearch cluster

Default wazuh

elasticsearch_node_name

Name of the Elasticsearch node

Default node-1

elasticsearch_http_port

ElasticSearch listening port

Default 9200

elasticsearch_network_host

ElasticSearch, listening ip address

Default 127.0.0.1

elasticsearch_jvm_xms

JVM heap size

Default null

elastic_stack_version

Version of Elasticsearch to install

Default 5.6.3

elasticsearch_shards

Set number of shards for indices

Default 5

elasticsearch_replicas

Set number of replicas for indices

Default 1

Kibana

elasticsearch_http_port

Elasticsearch node port.

Default 9200

elasticsearch_network_host

IP address or hostname of Elasticsearch node.

Default 127.0.0.1

kibana_server_host

Listening IP address of Kibana.

Default 0.0.0.0

kibana_server_port

Listening port of Kibana.

Default 5601

elastic_stack_version

Version of Kibana to install

Default 5.6.3

Logstash

logstash_create_config

Generate or not Logstash config.

Defaults true

logstash_input_beats

When is set to true, it will configure Logstash to use Filebeat input. Otherwise it will use File input.

Defaults false

elasticsearch_network_host

Ip address or hostname of Elasticsearch node.

Default 127.0.0.1

elasticsearch_http_port

Port of Elasticsearch node.

Default 9200

elasticsearch_shards

Set number of shards for indices

Default 5

elasticsearch_replicas

Set number of shards for indices

Default 1

elastic_stack_version

Version of Logstash to install

Default 5.6.3

logstash_ssl

Using ssl between filebeat and logstash

Default false

logstash_ssl_dir

Folder where the SSL key and cert will be stored.

Default /etc/pki/logstash

logstash_ssl_certificate_file

SSL certificate file to be copied from Ansible server to logstash server.

Default null

logstash_ssl_key_file

SSL key file to be copied from Ansible server to logstash server.

Default null

Filebeat

filebeat_create_config:

Generate or not Filebeat config.

Default true

filebeat_prospectors:

Set filebeat projectors to fetch data.

Example:

```
filebeat_prospectors:  
- input_type: log  
  paths:  
    - "/var/ossec/logs/alerts/alerts.json"  
  document_type: json  
  json.message_key: log  
  json.keys_under_root: true  
  json.overwrite_keys: true
```

filebeat_output_elasticsearch_enabled:

Send output to Elasticsearch node(s).

Default false

filebeat_output_elasticsearch_hosts:

Elasticsearch node(s) to send output.

Example:

```
filebeat_output_elasticsearch_hosts:  
- "localhost:9200"  
- "10.1.1.10:9200"
```

filebeat_output_logstash_enabled:

Send output to Logstash node(s).

Default true

filebeat_output_logstash_hosts:

Logstash node(s) to send output.

Example:

```
filebeat_output_logstash_hosts:  
- "10.1.1.10:5000"  
- "10.1.1.11:5000"
```

filebeat_enable_logging:

Enable/disable logging.

Default true

filebeat_log_level:

Set filebeat log level.

Default debug

filebeat_log_dir:

Set filebeat log directory.

Default: /var/log/mybeat

filebeat_log_filename:

Set filebeat log filename.

Default mybeat.log

filebeat_ssl_dir:

Set the folder containing SSL certs.

Default /etc/pki/logstash

filebeat_ssl_certificate_file:

Set certificate filename.

Default null

filebeat_ssl_key_file:

Set certificate key filename.

Default null

filebeat_ssl_insecure:

Verify validity of the server certificate hostname.

Default false

Wazuh Manager

wazuh_manager_fqdn:

Set Wazuh Manager fqdn hostname.

Default wazuh-server

wazuh_manager_config:

This store the Wazuh Manager configuration.

Example:

```
wazuh_manager_config:
  json_output: 'yes'
  alerts_log: 'yes'
  logall: 'no'
  log_format: 'plain'
  connection:
    - type: 'secure'
      port: '1514'
      protocol: 'tcp'
  authd:
    enable: false
    port: 1515
    use_source_ip: 'no'
    force_insert: 'no'
    force_time: 0
    purge: 'no'
    use_password: 'no'
    ssl_agent_ca: null
    ssl_verify_host: 'no'
    ssl_manager_cert: null
    ssl_manager_key: null
    ssl_auto_negotiate: 'no'
    email_notification: 'no'
  mail_to:
    - 'admin@example.net'
  mail_smtp_server: localhost
  mail_from: wazuh-server@example.com
  extra_emails:
    - enable: false
      mail_to: 'admin@example.net'
      format: full
      level: 7
      event_location: null
      group: null
      do_not_delay: false
      do_not_group: false
      rule_id: null
  reports:
    - enable: false
      category: 'syscheck'
      title: 'Daily report: File changes'
      email_to: 'admin@example.net'
      location: null
      group: null
      rule: null
      level: null
      srcip: null
      user: null
      showlogs: null
  syscheck:
    frequency: 43200
    scan_on_start: 'yes'
    auto_ignore: 'no'
    alert_new_files: 'yes'
  ignore:
    - /etc/mtab
    - /etc/mnttab
    - /etc/hosts.deny
    - /etc/mail/statistics
    - /etc/random-seed
    - /etc/random.seed
    - /etc/adjtime
    - /etc/httpd/logs
    - /etc/utmpx
    - /etc/wtmpx
    - /etc/cups/certs
    - /etc/dumpdates
    - /etc/svc/volatile
```

```
no_diff:
  - /etc/ssl/private.key
directories:
  - dirs: /etc,/usr/bin,/usr/sbin
    checks: 'check_all="yes"'
  - dirs: /bin,/sbin
    checks: 'check_all="yes"'
rootcheck:
frequency: 43200
openscap:
timeout: 1800
interval: '1d'
scan_on_start: 'yes'
log_level: 1
email_level: 12
localfiles:
  - format: 'syslog'
    location: '/var/log/messages'
  - format: 'syslog'
    location: '/var/log/secure'
  - format: 'command'
    command: 'df -P'
    frequency: '360'
  - format: 'full_command'
    command: 'netstat -tln | grep -v 127.0.0.1 | sort'
    frequency: '360'
  - format: 'full_command'
    command: 'last -n 20'
    frequency: '360'
globals:
  - '127.0.0.1'
  - '192.168.2.1'
commands:
  - name: 'disable-account'
    executable: 'disable-account.sh'
    expect: 'user'
    timeout_allowed: 'yes'
  - name: 'restart-ossec'
    executable: 'restart-ossec.sh'
    expect: ''
    timeout_allowed: 'no'
  - name: 'win_restart-ossec'
    executable: 'restart-ossec.cmd'
    expect: ''
    timeout_allowed: 'no'
  - name: 'firewall-drop'
    executable: 'firewall-drop.sh'
    expect: 'srcip'
    timeout_allowed: 'yes'
  - name: 'host-deny'
    executable: 'host-deny.sh'
    expect: 'srcip'
    timeout_allowed: 'yes'
  - name: 'route-null'
    executable: 'route-null.sh'
    expect: 'srcip'
    timeout_allowed: 'yes'
  - name: 'win_route-null'
    executable: 'route-null.cmd'
    expect: 'srcip'
    timeout_allowed: 'yes'
active_responses:
  - command: 'restart-ossec'
    location: 'local'
    rules_id: '100002'
  - command: 'win_restart-ossec'
    location: 'local'
    rules_id: '100003'
```

```
- command: 'host-deny'
location: 'local'
level: 6
timeout: 600
syslog_outputs:
- server: null
port: null
format: null
```

wazuh_agent_configs:

This store the different settings and profiles for centralized agent configuration via Wazuh Manager.

Example:

```
- type: os
type_value: Linux
syscheck:
  frequency: 43200
  scan_on_start: 'yes'
  auto_ignore: 'no'
  alert_new_files: 'yes'
  ignore:
    - /etc/mtab
    - /etc/mnttab
    - /etc/hosts.deny
    - /etc/mail/statistics
    - /etc/svc/volatile
  no_diff:
    - /etc/ssl/private.key
directories:
  - dirs: /etc,/usr/bin,/usr/sbin
    checks: 'check_all="yes"'
  - dirs: /bin,/sbin
    checks: 'check_all="yes"'
rootcheck:
  frequency: 43200
  cis_distribution_filename: null
localfiles:
  - format: 'syslog'
    location: '/var/log/messages'
  - format: 'syslog'
    location: '/var/log/secure'
  - format: 'syslog'
    location: '/var/log/maillog'
  - format: 'apache'
    location: '/var/log/httpd/error_log'
  - format: 'apache'
    location: '/var/log/httpd/access_log'
  - format: 'apache'
    location: '/var/ossec/logs/active-responses.log'
- type: os
type_value: Windows
syscheck:
  frequency: 43200
  scan_on_start: 'yes'
  auto_ignore: 'no'
  alert_new_files: 'yes'
  windows_registry:
    - key: 'HKEY_LOCAL_MACHINE\Software\Classes\batfile'
      arch: 'both'
    - key: 'HKEY_LOCAL_MACHINE\Software\Classes\Folder'
localfiles:
  - format: 'Security'
    location: 'eventchannel'
  - format: 'System'
    location: 'eventlog'
```

cdb_lists:

Configure CDB lists used by the Wazuh Manager (located at [ansible-wazuh-manager/vars/cdb_lists.yml](#)).

Example:

```
cdb_lists:  
- name: 'audit-keys'  
  content: |  
    audit-wazuh-w:write  
    audit-wazuh-r:read  
    audit-wazuh-a:attribute  
    audit-wazuh-x:execute  
    audit-wazuh-c:command
```

⚠ Warning

We recommend the use of [Ansible Vault](#) to protect Wazuh, agentless and authd credentials.

agentless_credits:

Credentials and host(s) to be used by agentless feature.

Example:

```
agentless_credits:  
- type: ssh_integrity_check_linux  
  frequency: 3600  
  host: root@example.net  
  state: periodic  
  arguments: '/bin /etc/ /sbin'  
  passwd: qwerty
```

⚠ Warning

We recommend the use of [Ansible Vault](#) to protect Wazuh, agentless and authd credentials.

wazuh_api_user:

Wazuh API credentials.

Example:

```
wazuh_api_user:  
- foo:$apr1$/axqZYWQ$Xo/nz/IG3PdwV82EnfYKh/  
- bar:$apr1$hXE97ag.$8m0koHByattiGKUKPUgcZ1
```

⚠ Warning

We recommend the use of [Ansible Vault](#) to protect Wazuh, agentless and authd credentials.

authd_pass:

Wazuh authd service password.

Example:

```
authd_pass: foobar
```

Wazuh Agent

wazuh_manager_ip:

Set Wazuh Manager server IP address to be used by the agent.

Default null

wazuh_profile:

Configure what profiles this agent will have.

Default null

Multiple profiles can be included, separated by a comma and a space, by example:

```
wazuh_profile: "centos7, centos7-web"
```

wazuh_agent_authd:

Set the agent-authd facility. This will enable or not the automatic agent registration, you could set various options in accordance of the authd service configured in the Wazuh Manager.

```
wazuh_agent_authd:  
  enable: false  
  port: 1515  
  ssl_agent_ca: null  
  ssl_agent_cert: null  
  ssl_agent_key: null  
  ssl_auto_negotiate: 'no'
```

wazuh_notify_time

Set the <notify_time> option in the agent.

Default null

wazuh_time_reconnect

Set <time-reconnect> option in the agent.

Default null

wazuh_winagent_config

Set the Wazuh Agent installation regarding Windows hosts.

```
install_dir: 'C:\wazuh-agent\'  
version: '2.1.1'  
revision: '2'  
repo: https://packages.wazuh.com/windows/  
md5: fd9a3ce30cd6f9f553a1bc71e74a6c9f
```

wazuh_agent_config:

Wazuh Agent related configuration.

Example:

```

log_format: 'plain'
syscheck:
  frequency: 43200
  scan_on_start: 'yes'
  auto_ignore: 'no'
  alert_new_files: 'yes'
  ignore:
    - /etc/mtab
    - /etc/mnttab
    - /etc/hosts.deny
    - /etc/mail/statistics
    - /etc/random-seed
    - /etc/random.seed
    - /etc/adjtime
    - /etc/httpd/logs
    - /etc/utmpx
    - /etc/wtmpx
    - /etc/cups/certs
    - /etc/dumpdates
    - /etc/svc/volatile
no_diff:
  - /etc/ssl/private.key
directories:
  - dirs: /etc,/usr/bin,/usr/sbin
    checks: 'check_all="yes"'
  - dirs: /bin,/sbin
    checks: 'check_all="yes"'
windows_registry:
  - key: 'HKEY_LOCAL_MACHINE\Software\Classes\batfile'
    arch: 'both'
  - key: 'HKEY_LOCAL_MACHINE\Software\Classes\Folder'
rootcheck:
  frequency: 43200
openscap:
  disable: 'yes'
  timeout: 1800
  interval: '1d'
  scan_on_start: 'yes'
localfiles:
  - format: 'syslog'
    location: '/var/log/messages'
  - format: 'syslog'
    location: '/var/log/secure'
  - format: 'command'
    command: 'df -P'
    frequency: '360'
  - format: 'full_command'
    command: 'netstat -tln | grep -v 127.0.0.1 | sort'
    frequency: '360'
  - format: 'full_command'
    command: 'last -n 20'
    frequency: '360'

```

Warning

We recommend the use of [Ansible Vault](#) to protect authd credentials.

authd_pass:

Wazuh authd credentials for agent registration.

Example:

```
authd_pass: foobar
```

Integration with AWS

Prior to enabling the Wazuh rules for Amazon Web Services, follow the steps below to enable the AWS API to generate log messages and store them as JSON data files in an Amazon S3 bucket. A detailed description of each of the steps can be found below.

1. Turn on CloudTrail.
2. Create a user with permission to access S3.
3. Install Python Boto on your Wazuh manager.
4. Configure user credentials with Python Boto.
5. Run the python script to download the JSON data.
6. Collect AWS log data.

Turn on CloudTrail

Create a trail for your AWS account. Trails can be created using the AWS CloudTrail console or the AWS Command Line Interface (AWS CLI). Both methods follow the same steps. In this case we will be focusing on the first one:

- Turn on CloudTrail. Note that, by default, when creating a trail in one region in the CloudTrail console, it will actually apply to all regions.

Warning

Please do not enable *Enable log file validation* parameter; it's not supported by the provided python script.

- Create a new Amazon S3 bucket or specify an existing bucket to store all your log files. By default, log files from all AWS regions in your account will be stored in the bucket selected.

Note

When naming a new bucket, if you get this error `Bucket already exists. Select a different bucket name.`, then try a different name, since the one you have selected is already in use by another Amazon AWS user.

From now on, all the events in your Amazon AWS account will be logged. You can search log messages manually inside `CloudTrail/API activity history`. Note that every 7 minutes, a JSON file containing new log messages will be stored in your bucket.

Create a user with permission to access S3

Sign in to the `AWS Management Console` and open the `IAM console`. In the navigation panel, choose `Users` and then choose `Create New Users`. Type the username for the user you would like to create.

Note

Usernames can only use a combination of alphanumeric characters and these characters: plus (+), equal (=), comma (,), period (.), at (@), and hyphen (-). Names must be unique within an account.

The user will need access to the API, which requires an access key. To generate an access key for a new user, select `Generate an access key` and choose `Create`.

Warning

This is your only opportunity to view or download the secret access key, and you must provide this information to your user before they can use the AWS Console. If you don't download and save it now, you will need to create new access keys for the user later. You will not have access to the secret access key again after this step.

Give the user access to this specific S3 bucket (based on [Writing IAM Policies: How to Grant Access to an Amazon S3 Bucket](#))

Under the IAM console, select `Users` and go to the `Permissions` tab, in the `Inline Policies` section, push the `Create User Policy` button. Click the `Custom Policy` option and push the `Select` button.

On the next page enter a `Policy Name` e.g. ossec-cloudtrail-s3-access, and for `Policy Document` use the example provided below:

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Effect": "Allow",  
      "Action": [ "s3>ListBucket" ],  
      "Resource": [ "arn:aws:s3:::YOURBUCKETNAME" ]  
    },  
    {  
      "Effect": "Allow",  
      "Action": [  
        "s3:GetObject",  
        "s3>DeleteObject"  
      ],  
      "Resource": [ "arn:aws:s3:::YOURBUCKETNAME/*" ]  
    }  
  ]  
}
```

Install Python Boto on your Wazuh manager

To download and process the Amazon AWS logs that already are archived in S3 Bucket we need to install Python Boto on the Wazuh manager and configure it to connect to AWS S3.

Prerequisites for Python Boto installation using Pip

- Windows, Linux, OS X, or Unix
- Python 2 version 2.7+ or Python 3 version 3.3+
- Pip

Check if Python is already installed:

```
$ python --version
```

If Python 2.7 or later is not installed, then install it with your distribution's package manager as shown below:

- On Debian derivatives such as Ubuntu, use APT:

```
$ sudo apt-get install python2.7
```

- On Red Hat and derivatives, use yum:

```
$ sudo yum install python27
```

Open a command prompt or shell and run the following command to verify that Python has been installed correctly:

```
$ python --version  
Python 2.7.9
```

To install Pip on Linux:

- Download the installation script from pypa.io:

```
$ curl -O https://bootstrap.pypa.io/get-pip.py
```

- Run the script with Python:

```
$ sudo python get-pip.py
```

Now that Python and pip are installed, use pip to install boto:

```
$ sudo pip install boto
```

Configure user credentials with Python Boto

It is necessary to configure the AWS CLI on your Wazuh manager to use the credentials of the newly created user. Create a file called `/etc/boto.cfg` like this:

```
[Credentials]
aws_access_key_id = <your_access_key_here>
aws_secret_access_key = <your_secret_key_here>
```

Run the python script to download the JSON data

We use a python script to download JSON files from the S3 bucket and convert them into flat files that can be used with Wazuh. This script was written by Xavier Martens @xme <<https://blog.rootshell.be>> and contains minor modifications done by Wazuh. It is located in our [repository](#) at [wazuh/wazuh-ruleset/tools/amazon/getawslog.py](#).

Run the following command to use this script:

```
$ ./getawslog.py -b s3bucketname -d -j -D -l /path-with-write-permission/amazon.log
```

Where `s3bucketname` is the name of the bucket created when CloudTrail was activated (see the first step in this section: "Turn on CloudTrail") and `/path-with-write-permission/amazon.log` is the path where the flat log file is stored once has been converted by the script.

! Note

If you don't want to use an existing folder, create a new one manually before running the script.

! Warning

The above script will delete all logs on the Amazon S3 bucket after download.

If you want to maintain the logs files in the bucket, you need to use the script without `-D` parameter like the following example:

```
$ ./getawslog.py -b s3bucketname -d -j -l /path-with-write-permission/amazon.log -s /path-with-write-permission/awslogstat.db
```

Using `-s /path-with-write-permission/awslogstat.db` will track downloaded log files avoiding processing them again, without it the script will download previously processed log files adding its content again to `/path-with-write-permission/amazon.log`. Also you need to install `sqlite` module for python:

```
$ sudo pip install pysqlite
```

CloudTrail delivers log files to your S3 bucket approximately every 7 minutes. Create a cron job to periodically run the script. Note that running it more frequently than once every 7 minutes would be useless. CloudTrail does not deliver log files if no API calls are made on your account.

Run `crontab -e` and, at the end of the file, add the following line

```
*/5 * * * * /usr/bin/flock -n /tmp/cron.lock -c python path_to_script/getawslog.py -b s3bucketname -d -j -D -l /path-with-write-permission/amazon.log
```

! Note

This script downloads and deletes the files from your S3 Bucket. However, you can always review the log messages generated during the last 7 days within the CloudTrail console.

Collect AWS log data

Now the Wazuh manager needs to be configured to be able to collect the log messages generated by AWS. In other words, the file

`/path-with-write-permission/amazon.log` generated by the script mentioned above needs to be added to the configuration file

`/var/ossec/etc/ossec.conf` using the `<ossec_config>` tag as shown below.

```
<ossec_config>
  <localfile>
    <log_format>syslog</log_format>
    <location>/path-with-write-permission/amazon.log</location>
  </localfile>
</ossec_config>
```

! Note

The file `/path-with-write-permission/amazon.log` must be the same one you setup in the above step: [Run the python script to download the JSON data](#).

Finally, restart the Wazuh manager to apply changes:

a. For Systemd:

```
systemctl restart wazuh-manager
```

b. For SysV Init:

```
service wazuh-manager restart
```

Use Cases

Our rules focuses on providing the desired visibility within the Amazon Web Services platform.

The following describes some use cases for IAM, EC2 and VPC services. The structure followed is always the same. You will see the definition of the rule that matches with the log message generated by the AWS event. You can check how this log message flows in the diagram at the beginning of this section. Also, on each of the examples, you will see a screenshot of how Kibana shows the corresponding alert. Remember that an alert is triggered when the log message matches a specific rule if its level is high enough.

Contents

- [IAM use cases](#)
 - [Create user account](#)
 - [Create user account without permissions](#)
 - [User login failed](#)
 - [Possible break-in attempt](#)
 - [Login success](#)
- [EC2 use cases](#)
 - [Run a new instance in EC2](#)
 - [Start instances in EC2](#)
 - [Stop instances in EC2](#)
 - [Create Security Groups in EC2](#)
 - [Allocate a new Elastic IP address](#)
 - [Associate a new Elastic IP address](#)
- [VPC Use cases](#)
 - [Create VPC](#)

2.1 Release Notes

This section shows the most relevant new features of Wazuh v2.1. You will find more detailed information in our [changelog](#) file.

New features:

- [Anti-flooding mechanism](#)
- [Labels for agent alerts](#)
- [Improved Authd performance](#)
- [New features for internal logs](#)
- [Updated external libraries](#)
- [Wazuh API](#)
- [Ruleset](#)

Anti-flooding mechanism

This mechanism is designed to prevent disruptively large bursts of events on an agent from negatively impacting the network or the manager. It uses a leaky bucket queue that collects all generated events, and sends them to the manager at a rate not exceeding a configurable events per second threshold.

Learn more about this new mechanism at [Anti-flooding mechanism](#).

Labels for agent alerts

This feature allows agent-specific attributes to be included in each alert involving a given agent. This could include things like who is in charge of a particular agent, or the installation date of that agent. This provides a simple method to add valuable metadata to alert records.

For more details about this new feature see our [Labels section](#).

Improved Authd performance

The Authd program has been improved in this version. Before Wazuh 2.1, the Wazuh API and the `manage_agents` tools could not register an agent while `ossec-authd` was running. Now agent registration is simultaneously supported via all of these methods.

Additionally, since this new version of `ossec-authd` runs in the background, it can be enabled using the command `ossec-control enable auth`. Its options can be configured in the `auth` section of `ossec.conf`. The documentation includes a good configuration example.

Finally, the new `force_insert` and `force_time` options in Authd (`-F<time>` from the `ossec-authd` command line) allow for automatic deletion of any agents that match the name or IP address of a newly registered agent.

New features for internal logs

It is widely known that JSON is one of the most popular logging formats. Because of this, it is now possible to have internal logs written in JSON format, plain text, or both. This can be configured in the `logging` section of `ossec.conf`.

In addition, internal logs are rotated and compressed daily, simplifying their management. To control disk space use, there is also a configurable threshold for how long to retain rotated logs before automatic deletion. These parameters are configured in the `monitord` section of [Internal configuration](#).

Updated external libraries

External libraries used by Wazuh have been updated to improve their integration with our components.

Wazuh API

The request `/agents` returns information about the OS.

Also, it is possible to delete or restart a list of specific agents.

Ruleset

The previous Windows decoders extracted a wrong user (the subject user). New decoders for Windows extract the proper user and new fields.

Install Wazuh server with RPM packages

For CentOS/RHEL/Fedora platforms, installing Wazuh server components is just install relevant packages by previously adding the appropriate repositories.

Note

Many of the commands described below need to be executed with root user privileges.

Adding the Wazuh repository

The first thing you need is to add the Wazuh repository to your server. Alternatively, if you prefer to download the wazuh-manager package directly, you can find it [here](#).

To set up the repository, run the command that corresponds to your specific RPM-based Linux distribution:

a. For CentOS:

```
$ cat > /etc/yum.repos.d/wazuh.repo <<\EOF
[wazuh_repo]
gpgcheck=1
gpgkey=https://packages.wazuh.com/key/GPG-KEY-WAZUH
enabled=1
name=CentOS-$releasever - Wazuh
baseurl=https://packages.wazuh.com/yum/el/$releasever/$basearch
protect=1
EOF
```

b. For RHEL:

```
$ cat > /etc/yum.repos.d/wazuh.repo <<\EOF
[wazuh_repo]
gpgcheck=1
gpgkey=https://packages.wazuh.com/key/GPG-KEY-WAZUH
enabled=1
name=RHEL-$releasever - Wazuh
baseurl=https://packages.wazuh.com/yum/rhel/$releasever/$basearch
protect=1
EOF
```

c. For Fedora:

```
$ cat > /etc/yum.repos.d/wazuh.repo <<\EOF
[wazuh_repo]
gpgcheck=1
gpgkey=https://packages.wazuh.com/key/GPG-KEY-WAZUH
name=Fedora-$releasever - Wazuh
enabled=1
baseurl=https://packages.wazuh.com/yum/fc/$releasever/$basearch
protect=1
EOF
```

d. For Amazon Linux:

```
$ cat > /etc/yum.repos.d/wazuh.repo <<\EOF
[wazuh_repo]
gpgcheck=1
gpgkey=https://packages.wazuh.com/key/GPG-KEY-WAZUH
enabled=1
name=Amazon Linux - Wazuh
baseurl=https://packages.wazuh.com/yum/el/7/$basearch
protect=1
EOF
```

Installing Wazuh manager

The next will install Wazuh manager on your system:

```
$ yum install wazuh-manager
```

Once the process is completed, you can check the service status with:

a. For Systemd:

```
$ systemctl status wazuh-manager
```

b. For SysV Init:

```
$ service wazuh-manager status
```

Installing Wazuh API

1. NodeJS >= 4.6.1 is required in order to run the Wazuh API. If you do not have NodeJS installed, or your version is older than 4.6.1, we recommend you add the official NodeJS repository like this:

```
$ curl --silent --location https://rpm.nodesource.com/setup_6.x | bash -
```

and then, install nodejs:

```
$ yum install nodejs
```

2. Install the Wazuh API. It will update NodeJS if it is required:

```
$ yum install wazuh-api
```

3. Once the process is completed, you can check the service status with:

a. For Systemd:

```
$ systemctl status wazuh-api
```

b. For SysV Init:

```
$ service wazuh-api status
```

4. Python >= 2.7 is required in order to run the Wazuh API. It is installed by default or included in the official repositories in most Linux distributions.

It is possible to set a custom Python path for the API in `/var/ossec/api/configuration/config.js`, in case the stock version of Python in your distro is too old:

```
config.python = [
    // Default installation
    {
        bin: "python",
        lib: ""
    },
    // Package 'python27' for CentOS 6
    {
        bin: "/opt/rh/python27/root/usr/bin/python",
        lib: "/opt/rh/python27/root/usr/lib64"
    }
];
```

CentOS 6 and Red Hat 6 come with Python 2.6, you can install Python 2.7 in parallel maintaining older version:

a. For CentOS 6:

```
$ yum install -y centos-release-scl
$ yum install -y python27
```

b. For RHEL 6:

```
$ yum install python27

# You may need to first enable a repository in order to get python27, with a command like this:
#   yum-config-manager --enable rhui-REGION-rhel-server-rhscl
#   yum-config-manager --enable rhel-server-rhscl-6-rpms
```

Installing Filebeat

Filebeat is the tool on the Wazuh server that will securely forward the alerts and archived events to the Logstash service on the Elastic Stack server(s).

Warning

In a single-host architecture (where Wazuh server and Elastic Stack are installed in the same system), you may entirely skip installing Filebeat, since Logstash will be able to read the event/alert data directly from the local filesystem without the assistance of a forwarder.

The RPM package is suitable for installation on Red Hat, CentOS and other modern RPM-based systems.

1. Install the GPG keys from Elastic, and the Elastic repository:

```
$ rpm --import https://packages.elastic.co/GPG-KEY-elasticsearch

$ cat > /etc/yum.repos.d/elastic.repo << EOF
[elastic-5.x]
name=Elastic repository for 5.x packages
baseurl=https://artifacts.elastic.co/packages/5.x/yum
gpgcheck=1
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch
enabled=1
autorefresh=1
type=rpm-md
EOF
```

2. Install Filebeat:

```
$ yum install filebeat-5.6.5
```

3. Download the Filebeat config file from the Wazuh repository, which is preconfigured to forward Wazuh alerts to Logstash:

```
$ curl -so /etc/filebeat/filebeat.yml  
https://raw.githubusercontent.com/wazuh/wazuh/2.1/extensions/filebeat/filebeat.yml
```

4. Edit the file `/etc/filebeat/filebeat.yml` and replace `ELASTIC_SERVER_IP` with the IP address or the hostname of the Elastic Stack server. For example:

```
output:  
  logstash:  
    hosts: ["ELASTIC_SERVER_IP:5000"]
```

5. Enable and start the Filebeat service:

a. For Systemd:

```
$ systemctl daemon-reload  
$ systemctl enable filebeat.service  
$ systemctl start filebeat.service
```

b. For SysV Init:

```
$ chkconfig --add filebeat  
$ service filebeat start
```

Next steps

Once you have installed the manager, API and Filebeat (only needed for distributed architectures), you are ready to [install Elastic Stack](#).

Install Wazuh server with DEB packages

For Debian/Ubuntu platforms, installing Wazuh server components is just install relevant packages by previously adding the appropriate repositories.

Note

Many of the commands described below need to be executed with root user privileges.

Adding Wazuh Repositories

The first thing you need is to add the Wazuh repository to your server. Alternatively, if you prefer to download the wazuh-manager package directly, you can find it [here](#).

1. In order to perform this procedure properly, packages `curl`, `apt-transport-https` and `lsb-release` must be installed into your system. If they are not, install them:

```
$ apt-get update  
$ apt-get install curl apt-transport-https lsb-release
```

2. Install the GPG key:

```
$ curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | apt-key add -
```

3. Getting the distribution codename and adding the repository:

```
$ CODENAME=$(lsb_release -cs)  
$ echo "deb https://packages.wazuh.com/apt $CODENAME main" | tee /etc/apt/sources.list.d/wazuh.list
```

These are the supported codename values:

- For Debian: wheezy, jessie, stretch and sid
- For Ubuntu: trusty, vivid, wily, xenial and yakkety

4. Update the package information:

```
$ apt-get update
```

Installing Wazuh Manager

On your terminal, install the Wazuh manager:

```
$ apt-get install wazuh-manager
```

Once the process is completed, you can check the service status with:

a. For Systemd:

```
$ systemctl status wazuh-manager
```

b. For SysV Init:

```
$ service wazuh-manager status
```

Installing Wazuh API

1. NodeJS >= 4.6.1 is required in order to run the Wazuh API. If you do not have NodeJS installed, or your version is older than 4.6.1, we recommend you add the official NodeJS repository like this:

```
$ curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -
```

and then, install nodejs:

```
$ apt-get install nodejs
```

2. Install the Wazuh API. It will update NodeJS if it is required:

```
$ apt-get install wazuh-api
```

3. Once the process is completed, you can check the service status with:

a. For Systemd:

```
$ systemctl status wazuh-api
```

b. For SysV Init:

```
$ service wazuh-api status
```

4. Python >= 2.7 is required in order to run the API. It is installed by default or included in the official repositories in most Linux distributions.

It is possible to set a custom Python path for the API in `/var/ossec/api/configuration/config.js`, in case the stock version of Python in your distro is too old:

```
config.python = [
    // Default installation
    {
        bin: "python",
        lib: ""
    },
    // Package 'python27' for CentOS 6
    {
        bin: "/opt/rh/python27/root/usr/bin/python",
        lib: "/opt/rh/python27/root/usr/lib64"
    }
];
```

Installing Filebeat

Filebeat is the tool on the Wazuh server that will securely forward the alerts and archived events to the Logstash service on the Elastic Stack server(s).

Warning

In a single-host architecture (where Wazuh server and Elastic Stack are installed in the same system), you may entirely skip installing Filebeat, since Logstash will be able to read the event/alert data directly from the local filesystem without the assistance of a forwarder.

The DEB package is suitable for Debian, Ubuntu, and other Debian-based systems.

1. Install the GPG keys from Elastic, and the Elastic repository:

```
$ curl -s https://artifacts.elastic.co/GPG-KEY-elasticsearch | apt-key add -
$ echo "deb https://artifacts.elastic.co/packages/5.x/apt stable main" | tee
/etc/apt/sources.list.d/elastic-5.x.list
$ apt-get update
```

2. Install Filebeat:

```
$ apt-get install filebeat=5.6.5
```

3. Download the Filebeat config file from the Wazuh repository, which is preconfigured to forward Wazuh alerts to Logstash:

```
$ curl -so /etc/filebeat/filebeat.yml
https://raw.githubusercontent.com/wazuh/wazuh/2.1/extensions/filebeat/filebeat.yml
```

4. Edit the file `/etc/filebeat/filebeat.yml` and replace `ELASTIC_SERVER_IP` with the IP address or the hostname of the Elastic Stack server. For example:

```
output:
  logstash:
    hosts: ["ELASTIC_SERVER_IP:5000"]
```

5. Enable and start the Filebeat service:

a. For Systemd:

```
$ systemctl daemon-reload  
$ systemctl enable filebeat.service  
$ systemctl start filebeat.service
```

b. For SysV Init:

```
$ update-rc.d filebeat defaults 95 10  
$ service filebeat start
```

Next steps

Once you have installed the manager, API and Filebeat (only needed for distributed architectures), you are ready to [install Elastic Stack](#).

Install Wazuh server from sources

This guide describes how to install the manager and API from source code. In addition, for distributed architectures, you will find some guidance on how to install Filebeat.

Note

Many of the commands described below need to be executed with root user privileges.

Installing Wazuh manager

1. Install development tools and compilers. In Linux this can easily be done using your distribution's package manager:

a. For RPM-based distributions:

```
$ sudo yum install make gcc git  
  
# If you want to use Auth, also install:  
$ sudo yum install openssl-devel
```

b. For Debian-based distributions:

```
$ sudo apt-get install gcc make git libcurl4-openssl-dev curl  
  
# If you want to use Auth, also install:  
$ sudo apt-get install libssl-dev
```

2. Download and extract the latest version:

```
$ curl -Ls https://github.com/wazuh/wazuh/archive/v2.1.1.tar.gz | tar zx
```

3. Run the `install.sh` script, this will display a wizard that will guide you through the installation process using the Wazuh sources:

```
$ cd wazuh-*  
$ sudo ./install.sh
```

4. The script will ask about what kind of installation you want. Type `server` to install Wazuh Manager:

```
1- What kind of installation do you want (server, agent, local, hybrid or help)? server
```

5. Start the services using this command:

```
$ sudo /var/ossec/bin/ossec-control start
```

Installing Wazuh API

1. NodeJS >= 4.6.1 is required in order to run the Wazuh API. If you do not have NodeJS installed or your version is older than 4.6.1, we recommend you add the official repository as this has more recent versions.

a. For RPM-based distributions:

```
$ curl --silent --location https://rpm.nodesource.com/setup_6.x | bash -  
$ sudo yum -y install nodejs
```

b. For Debian-based distributions:

```
$ curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -  
$ sudo apt-get install -y nodejs
```

! Note

[Official guide to install NodeJS.](#)

2. Download and execute the installation script:

```
$ curl -s -o install_api.sh https://raw.githubusercontent.com/wazuh/wazuh-api/v2.1.1/install_api.sh &&  
bash ./install_api.sh download
```

3. Python >= 2.7 is required in order to run the API. It is installed by default or included in the official repositories of most Linux distributions. It is possible to set a custom Python path for the API to use, in [/var/ossec/api/configuration/config.js](#):

```
config.python = [
    // Default installation
    {
        bin: "python",
        lib: ""
    },
    // Package 'python27' for CentOS 6
    {
        bin: "/opt/rh/python27/root/usr/bin/python",
        lib: "/opt/rh/python27/root/usr/lib64"
    }
];
```

CentOS 6 and Red Hat 6 come with Python 2.6, you can install Python 2.7 in parallel maintaining older version:

a. For CentOS 6:

```
$ sudo yum install -y centos-release-scl
$ sudo yum install -y python27
```

b. For RHEL 6:

```
$ sudo yum install python27

# You may need to first enable a repository in order to get python27, with a command like this:
#   sudo yum-config-manager --enable rhui-REGION-rhel-server-rhscl
#   sudo yum-config-manager --enable rhel-server-rhscl-6-rpms
```

Note

You can also run an [unattended installation](#) of the Wazuh manager and API.

Installing Filebeat

While Filebeat can be installed from source ([see this doc](#)), the process is more complex than you may like, and it is beyond the scope of Wazuh documentation. We recommend installing Filebeat via repository package, otherwise, you can install it from a binary tarball, that's should work for any Linux distro. See more [here](#).

Warning

In a single-host architecture (where Wazuh server and Elastic Stack are installed in the same system), you may entirely skip installing Filebeat, since Logstash will be able to read the event/alert data directly from the local filesystem without the assistance of a forwarder.

Next steps

Once you have installed the manager, API and Filebeat (only needed for distributed architectures), you are ready to [install Elastic Stack](#).

Install Wazuh agent with RPM packages

The RPM package is suitable for installation on Red Hat, CentOS and other RPM-based systems.

Note

Many of the commands described below need to be executed with root user privileges.

Adding the Wazuh repository

The first thing you need is to add the Wazuh repository to your server. Alternatively, if you prefer to download the wazuh-agent package directly, you can find it [here](#).

Run the following command that corresponds to your specific Linux distribution:

a. For CentOS:

```
$ cat > /etc/yum.repos.d/wazuh.repo <<\EOF
[wazuh_repo]
gpgcheck=1
gpgkey=https://packages.wazuh.com/key/GPG-KEY-WAZUH
enabled=1
name=CentOS-$releasever - Wazuh
baseurl=https://packages.wazuh.com/yum/el/$releasever/$basearch
protect=1
EOF
```

a.1) For CentOS-5:

```
$ cat > /etc/yum.repos.d/wazuh.repo <<\EOF
[wazuh_repo]
gpgcheck=1
gpgkey=https://packages.wazuh.com/key/RPM-GPG-KEY-OSSEC-RHEL5
enabled=1
name=CentOS-$releasever - Wazuh
baseurl=https://packages.wazuh.com/yum/el/$releasever/$basearch
protect=1
EOF
```

b. For RHEL:

```
$ cat > /etc/yum.repos.d/wazuh.repo <<\EOF
[wazuh_repo]
gpgcheck=1
gpgkey=https://packages.wazuh.com/key/GPG-KEY-WAZUH
enabled=1
name=RHEL-$releasever - Wazuh
baseurl=https://packages.wazuh.com/yum/rhel/$releasever/$basearch
protect=1
EOF
```

b.1) For RHEL-5:

```
$ cat > /etc/yum.repos.d/wazuh.repo <<\EOF
[wazuh_repo]
gpgcheck=1
gpgkey=https://packages.wazuh.com/key/RPM-GPG-KEY-OSSEC-RHEL5
enabled=1
name=RHEL-$releasever - Wazuh
baseurl=https://packages.wazuh.com/yum/rhel/$releasever/$basearch
protect=1
EOF
```

c. For Fedora:

```
$ cat > /etc/yum.repos.d/wazuh.repo <<\EOF
[wazuh_repo]
gpgcheck=1
gpgkey=https://packages.wazuh.com/key/GPG-KEY-WAZUH
name=Fedora-$releasever - Wazuh
enabled=1
baseurl=https://packages.wazuh.com/yum/fc/$releasever/$basearch
protect=1
EOF
```

d. For Amazon Linux:

```
$ cat > /etc/yum.repos.d/wazuh.repo <<\EOF
[wazuh_repo]
gpgcheck=1
gpgkey=https://packages.wazuh.com/key/GPG-KEY-WAZUH
name=Amazon Linux - Wazuh
enabled=1
baseurl=https://packages.wazuh.com/yum/el/7/$basearch
protect=1
EOF
```

Installing Wazuh agent

On your terminal, install the Wazuh agent:

```
$ yum install wazuh-agent
```

Note

At this point your agent is installed and you just need to register and configure it to talk to your manager. For more information about this process please visit our user manual.

Install Wazuh agent with DEB packages

The DEB package is suitable for Debian, Ubuntu, and other Debian-based systems.

Note

Many of the commands described below need to be executed with root user privileges.

Adding the Wazuh repository

The first thing you need is to add the Wazuh repository to your server. Alternatively, if you prefer to download the wazuh-agent package directly, you can find it [here](#).

1. In order to perform this procedure properly, packages `curl`, `apt-transport-https` and `lsb-release` must be present on your system. If they are not, install them:

```
$ apt-get install curl apt-transport-https lsb-release
```

2. Install the Wazuh repository GPG key:

```
$ curl -s https://packages.wazuh.com/key/GPG-KEY-WAZUH | apt-key add -
```

3. Getting the distribution codename and adding the repository:

```
CODENAME=$(lsb_release -cs)
echo "deb https://packages.wazuh.com/apt $CODENAME main" \
| tee /etc/apt/sources.list.d/wazuh.list
```

These are the supported codename values:

- For Debian: wheezy, jessie, stretch and sid
- For Ubuntu: trusty, vivid, wily, xenial and yakkety

4. Update the package information:

```
$ apt-get update
```

Installing Wazuh agent

On your terminal, install the Wazuh agent:

```
$ apt-get install wazuh-agent
```

Note

At this point your agent is installed and you just need to register and configure it to talk to your manager. For more information about this process please visit our user manual.

Install Wazuh agent on Windows

! Note

You will need administrator privilege to install using the next guide.

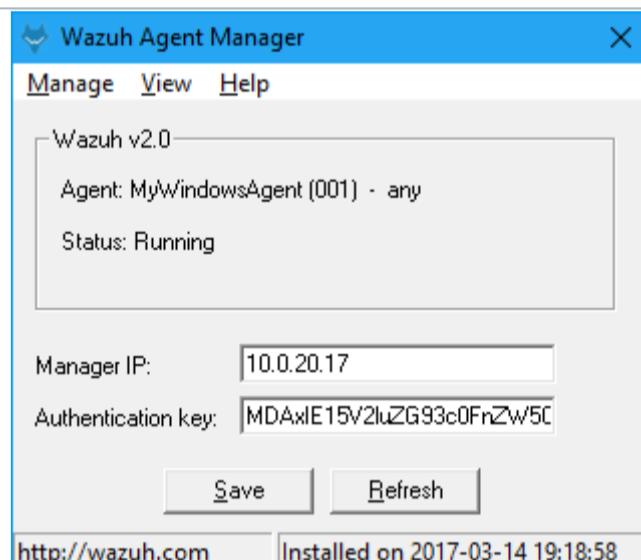
Download the Windows installer from our [packages list](#). You can install it via:

- [Using the GUI](#)
- [Using the command line](#)

Using the GUI

Double click on the downloaded file and follow the wizard. If unsure, leave default answers.

Once installed, the agent includes a graphical user interface that can be used to configure it, opening the log file or to start/stop the service.



By default all agent files can be found at the following location: [C:\Program Files\(x86\)\ossec-agent](http://C:\Program Files(x86)\ossec-agent).

! Note

At this point your agent is installed and you just need to register and configure it to talk to your manager. For more information about this process please visit our user manual.

Using the command line

Run the installer on a command line (the `/q` argument is used for unattended installations):

```
wazuh-agent-2.1.1-1.msi /q
```

You can automate the agent registration with authd using the following parameters:

Option	Description
APPLICATIONFOLDER	Installation path. Default C:Program Files (x86)ossec-agent.
SERVER_IP	Manager IP address. Optionally accepts a list of IPs separated by commas.
SERVER_HOSTNAME	Hostname from the manager. Optionally accepts a list of hostnames separated by commas.
SERVER_PORT	Manager connection port.
PROTOCOL	Communication protocol. Accepts UDP and TCP. Default is UDP.
AUTHD_SERVER	Authd IP address.

Option	Description
AUTHD_PORT	Authd connection port.
PASSWORD	Authd password.
NOTIFY_TIME	Time between manager checks.
TIME_RECONNECT	Time in seconds until a reconnection attempt.
CERTIFICATE	Certificate path.
AGENT_NAME	Agent's name. By default will be the computer name.
/l*v installer.log	Generates a log of the installation process.

Usage example:

```
wazuh-agent-2.1.1-1.msi /q SERVER_IP="192.168.1.1" AUTHD_SERVER="192.168.1.1" PASSWORD="TopSecret"  
AGENT_NAME="W2012"
```

Note

Unattended installations must be launched with administrator permissions.

Install Wazuh agent on Mac OS X

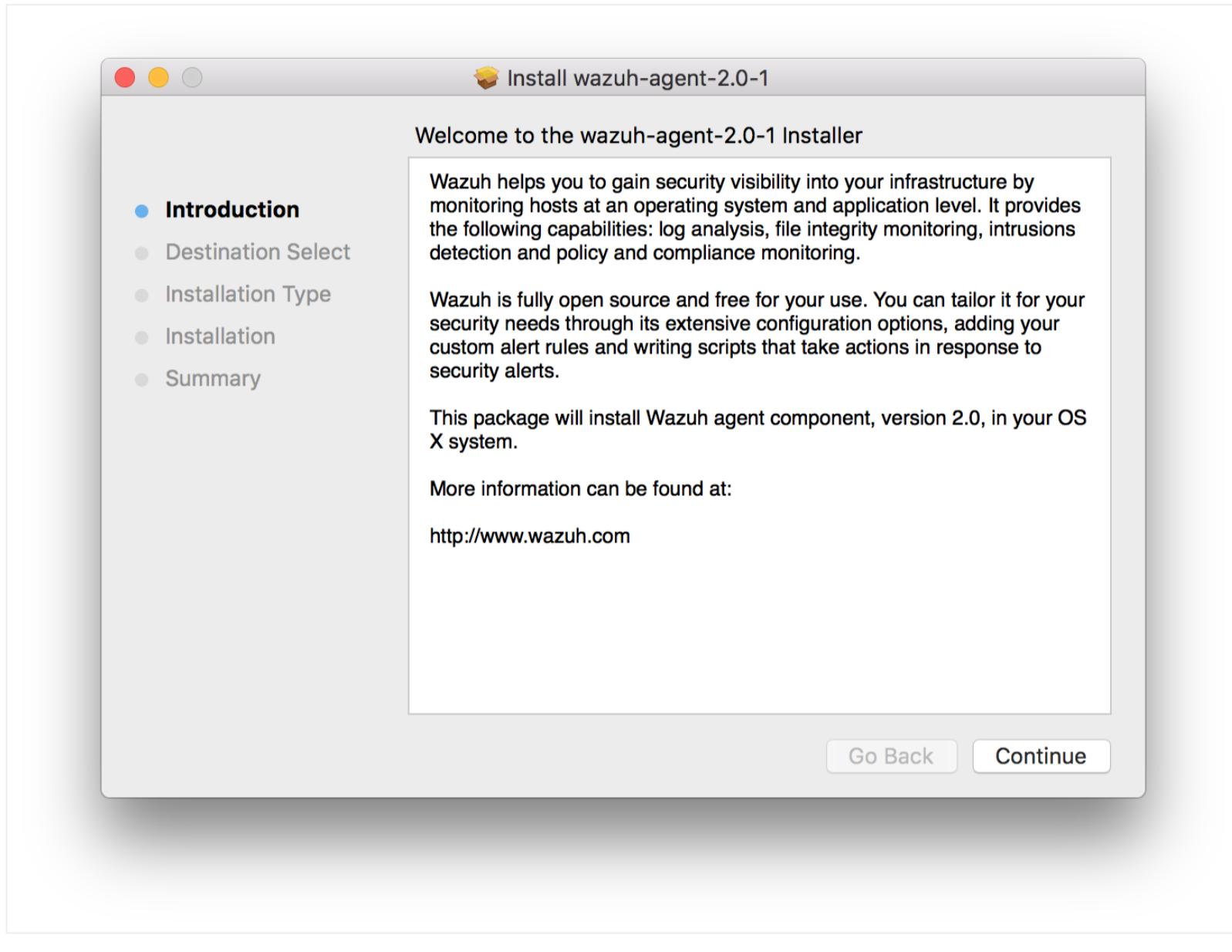
Mac OS X agent can be downloaded from our [packages list](#). The current version has been tested on Mac OS X 10.12 and should be compatible with other versions. You can install it via:

- The command line:

```
installer -pkg wazuh-agent-2.0-2.pkg -target /
```

- The GUI:

Double click on the downloaded file and follow the wizard. If unsure, leave default answers.



By default all agent files can be found at the following location: [/Library/Ossec/](#).

Note

At this point your agent is installed and you just need to register and configure it to talk to your manager. For more information about this process please visit our user manual.

Install Wazuh agent on Solaris

Solaris agent can be downloaded from our [packages list](#). Current version has been tested on Solaris 11 version 5.11 and Solaris 10 version 5.10. The installation step is:

- a. For Solaris 11 i386:

```
pkg install -d wazuh-agent_2.1.0-sol11-i386.p5p wazuh-agent
```

- b. For Solaris 10 i386:

```
pkgadd -d wazuh-agent_2.1.0-sol10-i386.pkg
```

- c. For Solaris 11 sparc:

```
pkg install -d wazuh-agent_2.1.0-sol11-sparc.p5p wazuh-agent
```

- d. For Solaris 10 sparc:

```
pkgadd -d wazuh-agent_2.1.0-sol10-sparc.pkg
```

Note

At this point your agent is installed and you just need to register and configure it to talk to your manager. For more information about this process please visit our user manual.

Install Wazuh agent from sources

This guide describes how to install an Wazuh agent from source code.

- [Installing Linux agent.](#)
- [Installing Windows agent.](#)

! Note

Many of the commands described below need to be executed with root user privileges.

Installing Linux agent

1. Install development tools and compilers. In Linux this can easily be done using your distribution's package manager:

a. For RPM-based distributions:

```
sudo yum install make gcc  
# If you want to use Auth, also install:  
sudo yum install openssl-devel
```

b. For Debian-based distributions:

```
sudo apt-get install gcc make libcurl4-openssl-dev curl  
# If you want to use Auth, also install:  
sudo apt-get install libssl-dev
```

2. Download and extract the latest version:

```
$ curl -Ls https://github.com/wazuh/wazuh/archive/v2.1.1.tar.gz | tar zx
```

3. Run the `install.sh` script, this will display a wizard that will guide you through the installation process using the Wazuh sources:

```
$ cd wazuh-*  
$ sudo ./install.sh
```

! Note

You can also run an [unattended installation](#).

4. The script will ask about what kind of installation you want. Type `agent` in order to install a Wazuh agent:

```
1- What kind of installation do you want (server, agent, local, hybrid or help)? agent
```

Note

At this point your agent is installed and you just need to register and configure it to talk to your manager. For more information about this process please visit our user manual.

Installing Windows agent

This section describes how to download and build the Wazuh HIDS Windows agent from sources. First of all, we will compile the agent on a Linux system to end up generating the .msi installer on Windows.

Note

The following procedure has been tested to work with Ubuntu 16.04 and other Debian based distributions as building environment, may work fine with other Debian/Ubuntu versions as well.

Set up Ubuntu build environment

Install these dependencies to build the Windows Wazuh agent installer on Ubuntu:

```
$ sudo apt-get install gcc-mingw-w64  
$ sudo apt-get install nsis  
$ sudo apt-get install make
```

Set up Windows build environment

To generate the installer we need to solve the following dependencies in Windows:

- [WiX Toolset](#).
- .NET framework 3.5.1 or higher.
- Microsoft Windows SDK.

Source code download

Download the Wazuh source code and unzip it:

```
$ curl -Ls https://github.com/wazuh/wazuh/archive/v2.1.1.tar.gz | tar zx  
$ cd wazuh-*/*src
```

Compiling the agent

Run the make command:

```
$ make TARGET=winagent
```

You should expect the following output at the end of the building process:

```
Done building winagent
```

Once the agent has been compiled, we should transfer the `src` folder to a Windows system. This folder could be compressed at first to speed up the process.

```
$ zip -r ../src *
```

Once in Windows, we only need to run [src/win32/wazuh-installer-build-msi.bat](#) to start the installer generation. If we don't want to sign the installer, we will have to comment or delete the signtool line.

Note

At this point the installer is ready. You can launch it with a normal or unattended installation. For more information about this process please visit our [installation section for Windows](#).

Upgrade from the same minor version

Use these instructions if you are upgrading your Wazuh installation within the same minor version. As an example, from 2.0.0 to 2.1.1.

Upgrade Wazuh manager

a. Upgrade Wazuh server on CentOS/RHEL/Fedora:

```
$ sudo yum upgrade wazuh-manager
```

b. Upgrade Wazuh server on Debian/Ubuntu:

```
$ sudo apt-get update && sudo apt-get install --only-upgrade wazuh-manager
```

Upgrade Wazuh agent

a. Upgrade Wazuh agent on CentOS/RHEL/Fedora:

```
$ sudo yum upgrade wazuh-agent
```

b. Upgrade Wazuh agent on Debian/Ubuntu:

```
$ sudo apt-get update && sudo apt-get install --only-upgrade wazuh-agent
```


Upgrade from the same major version

Use these instructions if you are upgrading your Wazuh installation within the same major version. As an example, from 2.0.1 to 2.1.1.

Upgrade Wazuh manager

Before upgrading the Wazuh manager, stop `ossec-authd` in case that it is running in background. Since Wazuh 2.1.0, `ossec-authd` should be configured in the [auth section](#) of `ossec.conf`.

- Upgrade Wazuh server on CentOS/RHEL/Fedora:

```
$ sudo yum upgrade wazuh-manager
```

- Upgrade Wazuh server on Debian/Ubuntu:

```
$ sudo apt-get update && sudo apt-get install --only-upgrade wazuh-manager
```

Upgrade Wazuh API

- Upgrade Wazuh API on CentOS/RHEL/Fedora:

```
$ sudo yum upgrade wazuh-api
```

- Upgrade Wazuh API on Debian/Ubuntu:

```
$ sudo apt-get update && sudo apt-get install --only-upgrade wazuh-api
```

Upgrade Wazuh agent

- Upgrade Wazuh agent on CentOS/RHEL/Fedora:

```
$ sudo yum upgrade wazuh-agent
```

- Upgrade Wazuh agent on Debian/Ubuntu:

```
$ sudo apt-get update && sudo apt-get install --only-upgrade wazuh-agent
```

Upgrade Wazuh Kibana App

- On your terminal, remove the current Wazuh Kibana App:

```
$ /usr/share/kibana/bin/kibana-plugin remove wazuh
```

2. Once the process is completed, you must stop Kibana with:

a. For Systemd:

```
$ systemctl stop kibana
```

b. For SysV Init:

```
$ service kibana stop
```

3. Remove the current kibana bundles:

```
$ rm -rf /usr/share/kibana/optimize/bundles
```

4. Upgrade Wazuh Kibana App (this can take a while):

```
$ /usr/share/kibana/bin/kibana-plugin install https://packages.wazuh.com/wazuhapp/wazuhapp.zip
```

5. Once the process is completed, you must start Kibana again with:

a. For Systemd:

```
$ systemctl start kibana
```

b. For SysV Init:

```
$ service kibana start
```

Wazuh Manager

This role will install and configure Wazuh Manager and Wazuh API, there are several variables you can use to customize the installation or configuration, by example:

- **json_output**: enabling or not JSON output (default: `yes`)
- **email_notification**: enabling email notifications (default: `no`)
- **mail_to**: email notifications recipients (array, defaults: `admin@example.net`)
- **mail_smtp_server**: SMTP server to be used by email notifications (defaults: `localhost`)
- **mail_from**: email notification sender (defaults: `ossec@example.com`)

By creating a YAML file `wazuh-manager.yml` you can be set the usage of this role:

```
- hosts: wazuh-manager
  roles:
    - ansible-wazuh-manager
    - ansible-role-filebeat
```

Setting the variables on a separate YAML file is recommended when configuring the installation. For this example we used:

`vars-production.yml`:

```
filebeat_output_logstash_hosts: '10.1.1.11:5000'

wazuh_manager_fqdn: "wazuh-server"

wazuh_manager_config:
  json_output: 'yes'
  alerts_log: 'yes'
  logall: 'no'
  log_format: 'plain'
  connection:
    - type: 'secure'
      port: '1514'
      protocol: 'tcp'
  authd:
    enable: true
    port: 1515
    use_source_ip: 'no'
    force_insert: 'no'
    force_time: 0
    purge: 'no'
    use_password: 'no'
    ssl_agent_ca: null
    ssl_verify_host: 'no'
    ssl_manager_cert: null
    ssl_manager_key: null
    ssl_auto_negotiate: 'no'
```

You can configure **Wazuh API** user credentials, this could be done by setting the file: `ansible-wazuh-manager/vars/wazuh_api_creds.yml` located on your Ansible control server, the credentials are in `htpasswd` format:

```
# Be sure you encrypt this file with ansible-vault
wazuh_api_user:
- foo:$apr1$/axqZYWQ$Xo/nz/IG3PdwV82EnfYKh/
- bar:$apr1$hXE97ag.$8m0koHByattiGKUKPUgcZ1
```

Also, you can configure **agentless** host credentials via the file: `ansible-wazuh-manager/vars/agentless_creds.yml`, set many as you need:

```
# Be sure you encrypt this file with ansible-vault.  
agentless_credits:  
- type: ssh_integrity_check_linux  
  frequency: 3600  
  host: root@example1.net  
  state: periodic  
  arguments: '/bin /etc/ /sbin'  
  passwd: qwerty  
- type: ssh_integrity_check_bsd  
  frequency: 3600  
  host: user@example2.net  
  state: periodic  
  arguments: '/bin /etc/ /sbin'  
  passwd: qwerty
```

And the `authd` service password could be set in the file `ansible-wazuh-manager/vars/authd_pass.yml`:

```
# Be sure you encrypt this file with ansible-vault  
authd_pass: foobar
```

⚠ Warning

We recommend the use of [Ansible Vault](#) to protect Wazuh API and agentless credentials.

Next, run the playbook:

```
$ ansible-playbook wazuh-manager.yml -e@vars-production.yml
```

The example above will install Wazuh Manager and Filebeat, Filebeat will be configured to forward data to `10.1.1.11:5000` as Logstash node, also it will set various `agentless` hosts configurations including their credentials, the Wazuh API and the `authd` will be configured as well.

Please review the [references](#) section to see all variables available for this role.

Filebeat

Filebeat can be used in conjunction with Wazuh Manager to send events and alerts to Logstash node, this role will install Filebeat, you can customize the installation with these variables:

- **filebeat_output_logstash_hosts:** define logstash node(s) to be use (default: `127.0.0.1:5000`).

Please review the [references](#) section to see all variables available for this role.

Elasticsearch

This role is intended to deploy Elasticsearch node, you have some variables that can be used to customize the installation:

- **elasticsearch_network_host**: defines listen ip address (default: `127.0.0.1`).
- **elasticsearch_http_port**: defines listen port (default: `9200`).
- **elasticsearch_jvm_xms**: amount of memory for java (default: `null`).
- **elastic_stack_version**: defines elk version to be installed.

You can create a YAML file `wazuh-elastic.yml` to be used by Ansible playbook:

```
- hosts: elasticsearch
  roles:
    - ansible-role-elasticsearch
```

You can set your custom variable definitions for different environments, for example:

a. For production environment `vars-production.yml`:

```
elasticsearch_network_host: '10.1.1.10'
```

b. For development environment `vars-development.yml`:

```
elasticsearch_network_host: '192.168.0.10'
```

Next, run the ansible playbook:

```
ansible-playbook wazuh-elastic.yml -e@vars-production.yml
```

The example above will install Elasticsearch and set the listen address to: `10.1.1.10` using `vars-production.yml`.

Please review the [references](#) section to see all variables available for this role.

Kibana

This is similar to the Elasticsearch role and is intended to deploy Kibana with the correct Wazuh APP version, you can customize the installation with the following:

- **elasticsearch_network_host**: defines Elasticsearch node ip address (default: `127.0.0.1`).
- **elasticsearch_http_port**: defines Elasticsearch node port (default: `9200`).
- **kibana_server_host**: defines Kibana listen address (default: `0.0.0.0`).
- **elastic_stack_version**: defines Kibana version to be installed.

You can create a YAML file `wazuh-kibana.yml` to be used by Ansible playbook:

```
- hosts: kibana
  roles:
    - ansible-role-kibana
```

You can set your custom variable definitions for different environments, for example:

a. For production environment `vars-production.yml`:

```
elasticsearch_network_host: '10.1.1.10'
```

b. For development environment `vars-development.yml`:

```
elasticsearch_network_host: '192.168.0.10'
```

Next, run the Ansible playbook:

```
ansible-playbook wazuh-kibana.yml -e@vars-production.yml
```

The example above will install Kibana and configure to use `10.1.1.10` as Elasticsearch node.

Please review the [references](#) section to see all variables available for this role.

Logstash

This role will install and configure Logstash with Wazuh templates on the hosts you selected, you can customize the installation with this vars:

- **elasticsearch_network_host**: defines Elasticsearch node ip address (default: `127.0.0.1`).
- **elasticsearch_http_port**: defines Elasticsearch node port (default: `9200`).
- **elastic_stack_version**: defines Logstash version to be installed.
- **logstash_input_beats**: defines the use of File input or Filebeat input. (defaults: `false`)

Create a YAML file `wazuh-logstash.yml` to be used by Ansible playbook:

```
- hosts: logstash
  roles:
    - ansible-role-logstash
```

You can set your custom variable definitions for different environments, for example:

a. For production enviroment `vars-production.yml`:

```
elasticsearch_network_host: '10.1.1.10'
logstash_input_beats: true
```

b. For development environment `vars-development.yml`:

```
elasticsearch_network_host: '127.0.0.1'
logstash_input_beats: false
```

Next, run the Ansible playbook:

```
ansible-playbook wazuh-logstash.yml -e@vars-production.yml
```

The example above will install Logstash and configure to use `10.1.1.10` as Elasticsearch node enabling the Filebeat input.

Please review the [references](#) section to see all variables available for this role.

Wazuh Agent

This role is designed to install and configure Wazuh Agent on different hosts, this agent is compatible with Linux and Windows machines. Also, has the ability to register the agent using the `ossec-authd` service on the Wazuh Manager, you can use several variables to customize the installation:

- `wazuh_manager_ip`: set Wazuh server to connect.
- `wazuh_agent_authd`: array with a set of options to register the Wazuh agent on the Wazuh server, will require the `ossec-authd` service started on the Wazuh server.

By example, create a YAML file `wazuh-agent.yml` to be used by Ansible playbook:

```
- hosts: all:!wazuh-manager
  roles:
    - ansible-wazuh-agent
```

You can maintain different environments using a variable definition YAML file for each one:

a. For production environment `vars-production.yml`:

```
wazuh_manager_ip: 10.1.1.12
wazuh_agent_authd:
  enable: true
  port: 1515
  ssl_agent_ca: null
  ssl_agent_cert: null
  ssl_agent_key: null
  ssl_auto_negotiate: 'no'
```

b. For development environment `vars-development.yml`:

```
wazuh_manager_ip: 192.168.0.10
wazuh_agent_authd:
  enable: true
  port: 1515
  ssl_agent_ca: null
  ssl_agent_cert: null
  ssl_agent_key: null
  ssl_auto_negotiate: 'no'
```

Next, run the ansible playbook:

```
$ ansible-playbook wazuh-agent.yml -e@vars-production.yml
```

The example above for production environment will install Wazuh agent in all host except `wazuh-manager`. then it will register against `wazuh-manager` with ip `10.1.1.12`.

Please review the [references](#) section to see all variables available for this role.

Upgrading Wazuh server

Follow next steps in order to update your `Wazuh v1.x` server to `Wazuh v2.x`.

1. First of all, stop running processes:

```
$ /var/ossec/bin/ossec-control stop  
$ systemctl stop wazuh-api
```

2. Only if you have a distributed architecture, remove logstash-forwarder (it's been replaced by Filebeat):

Deb systems:

```
$ apt-get remove logstash-forwarder
```

RPM systems:

```
$ yum remove logstash-forwarder
```

3. Install Wazuh server:

You could upgrade your current installation by following our installation guide.

- [Install Wazuh server with RPM packages](#)
- [Install Wazuh server with Deb packages](#)

Once the package is installed, review your `/var/ossec/etc/ossec.conf` file, as it will be overwritten. The one that was previously in use has been saved as `ossec.conf.rpmorig` or `ossec.conf.deborig`. It is recommended to compare the new file with the old one and import old settings when needed.

A backup of your custom rules and decoders will be saved at `/var/ossec/etc/backup_ruleset`. You need to reapply them again, we recommend use `/var/ossec/etc/decoders` and `/var/ossec/etc/rules` for custom rules and decoders, these directories won't be overwritten by future upgrades.

4. Run `/var/ossec/bin/manage_agents -V` to confirm that now you are running `Wazuh v2.x`:

```
$ /var/ossec/bin/manage_agents -V
```

Wazuh v2.0 - Wazuh Inc.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License (version 2) as published by the Free Software Foundation.

Upgrading Elastic Stack server

Although Wazuh v2.x is compatible both with Elastic Stack 2.x and 5.x, it is recommended to run it with version 5.x, our Wazuh Kibana App is not compatible with Elastic Stack 2.X. In any case, here is a brief description of the upgrade process, no matter which version of the cluster you decide to keep.

1. [Keep using Elastic Stack 2.x](#)
2. [Upgrade from Elastic Stack 2.x to 5.x](#)

Keep using Elastic Stack 2.x

In this scenario you will only need to configure Logstash to receive data from Filebeat (or directly read alerts generated by Wazuh server for a single-host architecture) and feed the Elasticsearch using the Wazuh alerts template:

Configure Logstash

1. Download the new logstash configuration:

```
$ curl -so /etc/logstash/conf.d/01-wazuh.conf  
https://raw.githubusercontent.com/wazuh/wazuh/2.1/extensions/logstash/01-wazuh.conf  
$ curl -so /etc/logstash/wazuh-elastic2-template.json  
https://raw.githubusercontent.com/wazuh/wazuh/2.1/extensions/elasticsearch/wazuh-elastic2-template.json
```

2. In the output section of `/etc/logstash/conf.d/01-wazuh.conf`, comment the line for `elastic5-template` and uncomment the line for `elastic2-template`:

```
output {  
  elasticsearch {  
    hosts => ["localhost:9200"]  
    index => "wazuh-alerts-%{+YYYY.MM.dd}"  
    document_type => "wazuh"  
    #      template => "/etc/logstash/wazuh-elastic5-template.json"  
    template => "/etc/logstash/wazuh-elastic2-template.json"  
    template_name => "wazuh"  
    template_overwrite => true  
  }  
}
```

3. Only if you are using a single-host architecture (where Wazuh server is running with Elastic Stack in the same host), edit `/etc/logstash/conf.d/01-wazuh.conf` commenting out the entire input section titled `Remote Wazuh Manager - Filebeat input` and uncommenting the entire input section titled `Local Wazuh Manager - JSON file input`:

```

# Wazuh - Logstash configuration file
## Remote Wazuh Manager - Filebeat input

#beats {
#    port => 5000
#    codec => "json_lines"
#    ssl => true
#    ssl_certificate => "/etc/logstash/logstash.crt"
#    ssl_key => "/etc/logstash/logstash.key"
# }
#}
# Local Wazuh Manager - JSON file input

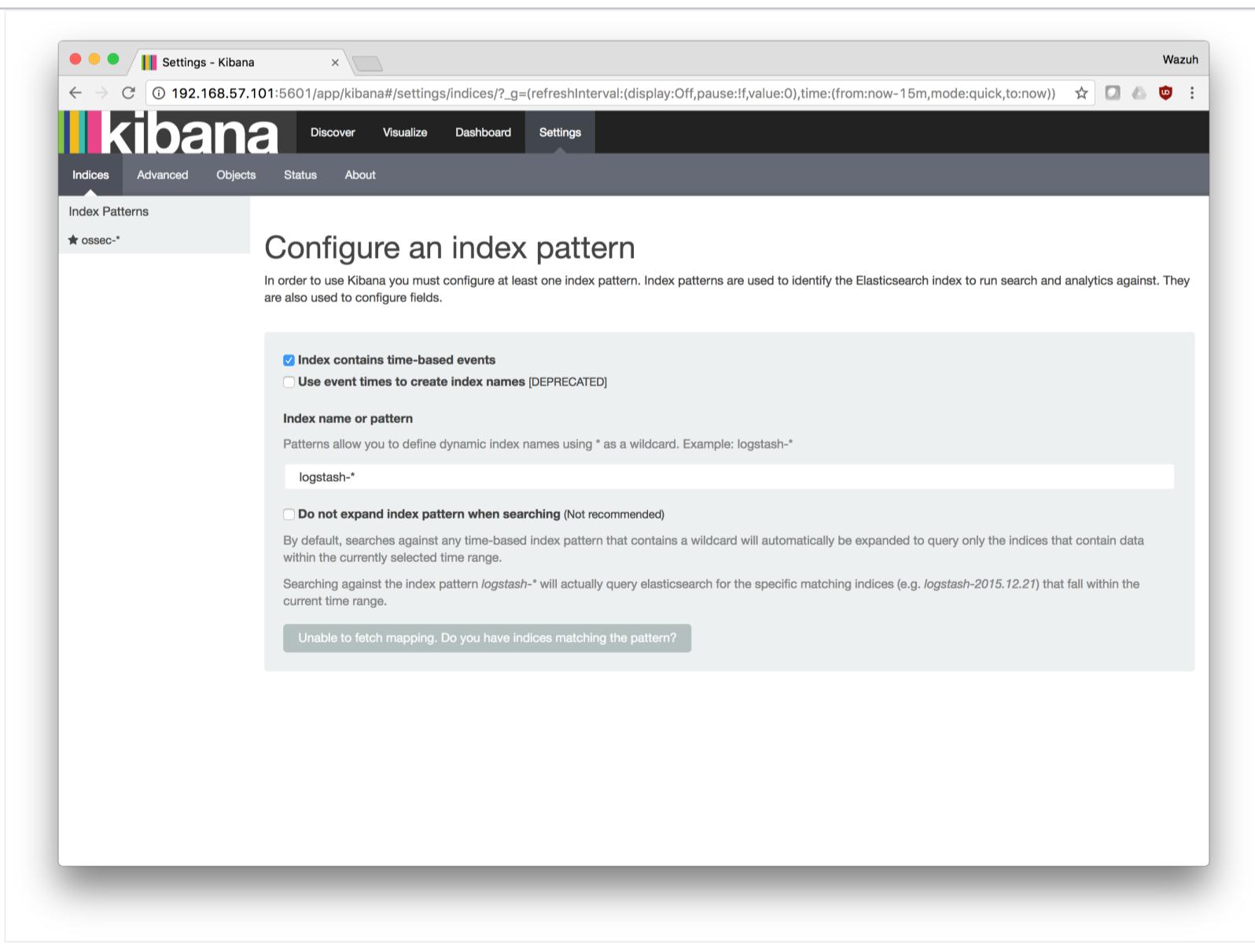

```

The above will setup Logstash to `read` the Wazuh `alerts.json` file directly from the `local` filesystem rather than receive forwarded data from Filebeat.

Configure Kibana

Next, in order to display Wazuh alerts data, we will configure Kibana index pattern.

1. Go to Settings and configure a new wildcard:



2. Set `wazuh-*` as index pattern and choose `timestamp` as time field, then click on create:

The screenshot shows the Kibana Settings interface. In the top navigation bar, the 'Settings' tab is selected. Below it, the 'Index Patterns' section lists an entry for 'ossec-*'. A modal window titled 'Configure an index pattern' is open, prompting the user to define a new index pattern. The 'Index name or pattern' field contains 'wazuh-*'. Under 'Time-field name', the value '@timestamp' is selected. At the bottom of the modal is a green 'Create' button.

3. Set as default wildcard by clicking on the Star:

The screenshot shows the Kibana Settings interface for the 'wazuh-*' index pattern. The 'Fields' tab is selected, displaying a table of 148 core fields. The table includes columns for 'name', 'type', 'format', 'analyzed', 'indexed', and 'controls'. Most fields are of type 'string', with a few like 'GeoLocation.latitude' being 'number'. The 'analyzed' column shows some fields as checked (✓) and others as uncheckable (✗). The 'indexed' column also shows mixed status. The 'controls' column contains edit icons for each row.

name	type	format	analyzed	indexed	controls
rule.id	string		✓	✓	
audit.directory.name	string		✓	✓	
dstport	string		✓	✓	
audit.tty	string		✓	✓	
type	string		✓	✓	
GeoLocation.latitude	number		✓	✓	
rule.cve	string		✓	✓	
path	string		✓	✓	
hostname	string		✓	✓	
audit.exe	string		✓	✓	
syscheck.perm_before	string		✓	✓	
action	string		✓	✓	
agent.name	string		✓	✓	

4. Go to the **Discover** tab in order to visualize the alerts data.

Upgrade from Elastic Stack 2.x to 5.x

Follow next steps to upgrade your Elastic Stack cluster to version 5.X:

1. Stop the running Logstash, Elasticsearch and Kibana instances:

a. For Systemd:

```
$ systemctl stop logstash.service  
$ systemctl stop elasticsearch.service  
$ systemctl stop kibana.service
```

b. For SysV Init:

```
$ service logstash stop  
$ service elasticsearch stop  
$ service kibana stop
```

2. Remove Logstash old configuration and template files:

For single-host architectures (Wazuh server and Elastic Stack running in the same system):

```
$ rm /etc/logstash/conf.d/01-ossec-singlehost.conf  
$ rm /etc/logstash/elastic-ossec-template.json
```

For distributed architectures (Elastic Stack standalone server):

```
$ rm /etc/logstash/conf.d/01-ossec.conf  
$ rm /etc/logstash/elastic-ossec-template.json
```

3. Remove deprecated settings from configuration file:

Removing deprecated settings on Elasticsearch will avoid errors & conflicts after the upgrade, To do that, comment the following lines on your `/etc/elasticsearch/elasticsearch.yml` file:

```
index.number_of_shards: 1  
index.number_of_replicas: 0
```

`ES_HEAP_SIZE` option is now deprecated. You should remove or comment out this option in your `/etc/sysconfig/elasticsearch` file:

```
# ES_HEAP_SIZE - Set it to half your system RAM memory  
ES_HEAP_SIZE=8g
```

Now you can go ahead and configure it following the Elastic [jvm.options guide](#)

4. At this point, you could install the new version of Elastic Stack. Depending on your operating system you can follow one of these installation instructions:

- [Install Elastic Stack with RPM packages](#)
- [Install Elastic Stack with DEB packages](#)

5. Let's check the software version of the different components to verify everything worked as expected:

a. For Logstash:

```
$ /usr/share/logstash/bin/logstash -V  
logstash 5.2.2
```

b. For Elasticsearch:

```
$ /usr/share/elasticsearch/bin/elasticsearch -V  
Version: 5.2.2, Build: f9d9b74/2017-02-24T17:26:45.835Z, JVM: 1.8.0_60
```

c. For Kibana:

```
$ /usr/share/kibana/bin/kibana -V  
5.2.
```

Note

Wazuh v2.x uses different indices and templates than Wazuh v1.x. For that reason, you will not be able to see the previous alerts using Kibana. If you need to access them, you will have to reindex the previous indices.

Upgrading Wazuh agents

Follow next steps in order to update your [Wazuh v1.x](#) agents to [Wazuh v2.x](#).

- a. On DEB or RPM based **Linux systems**, you can easily rely on the packages manager to upgrade your agents. The process differs very little from installing a new agent. More information available in our documentation at:

- [Install Wazuh agent with RPM packages](#)
- [Install Wazuh agent with Deb packages](#)

You can check your agent version running the following command:

```
/var/ossec/bin/manage_agents -V
```

```
Wazuh v2.0 - Wazuh Inc.
```

```
This program is free software; you can redistribute it and/or modify  
it under the terms of the GNU General Public License (version 2) as  
published by the Free Software Foundation.
```

- b. On **Windows**, **Mac OS** and other operating systems, we advise you to delete your previous version and install Wazuh v2.x from scratch. More information can be found at:

- [Install Wazuh agent on Windows](#)
- [Install Wazuh agent on Mac OS X](#)

Components

Wazuh's main components are the agent that runs on each monitored host, and the server that analyzes data received from the agents and from other agentless sources like syslog. In addition, the server forwards event data to an Elasticsearch cluster, where information is indexed and stored.

Wazuh agent

The Wazuh agent runs on Windows, Linux, Solaris, BSD, or Mac operating system. It is used to collect different types of system and application data. The agent forwards the collected data to the Wazuh server through an encrypted and authenticated channel. In order to establish this secure channel, a registration process involving unique pre-shared keys is utilized.

The agents can be used to monitor physical servers, virtual machines and cloud instances (e.g. Amazon AWS, Azure or Google Cloud). Pre-compiled agent installation packages are already available for these operating systems: Linux, AIX, Solaris, Windows, and Darwin (Mac OS X).

On Unix-based operating systems the agent runs multiple processes, these processes communicate each other through a local Unix domain socket, one of those processes is in charge of the communication and data sending to the Wazuh server. On Windows systems, there is only one agent process running multiple tasks using mutexes.

Different agent tasks or processes are used to monitor the system in different ways (e.g., monitoring file integrity, reading system log messages, and scanning system configurations).

The diagram below represents the internal tasks and processes that take place at the agent level:

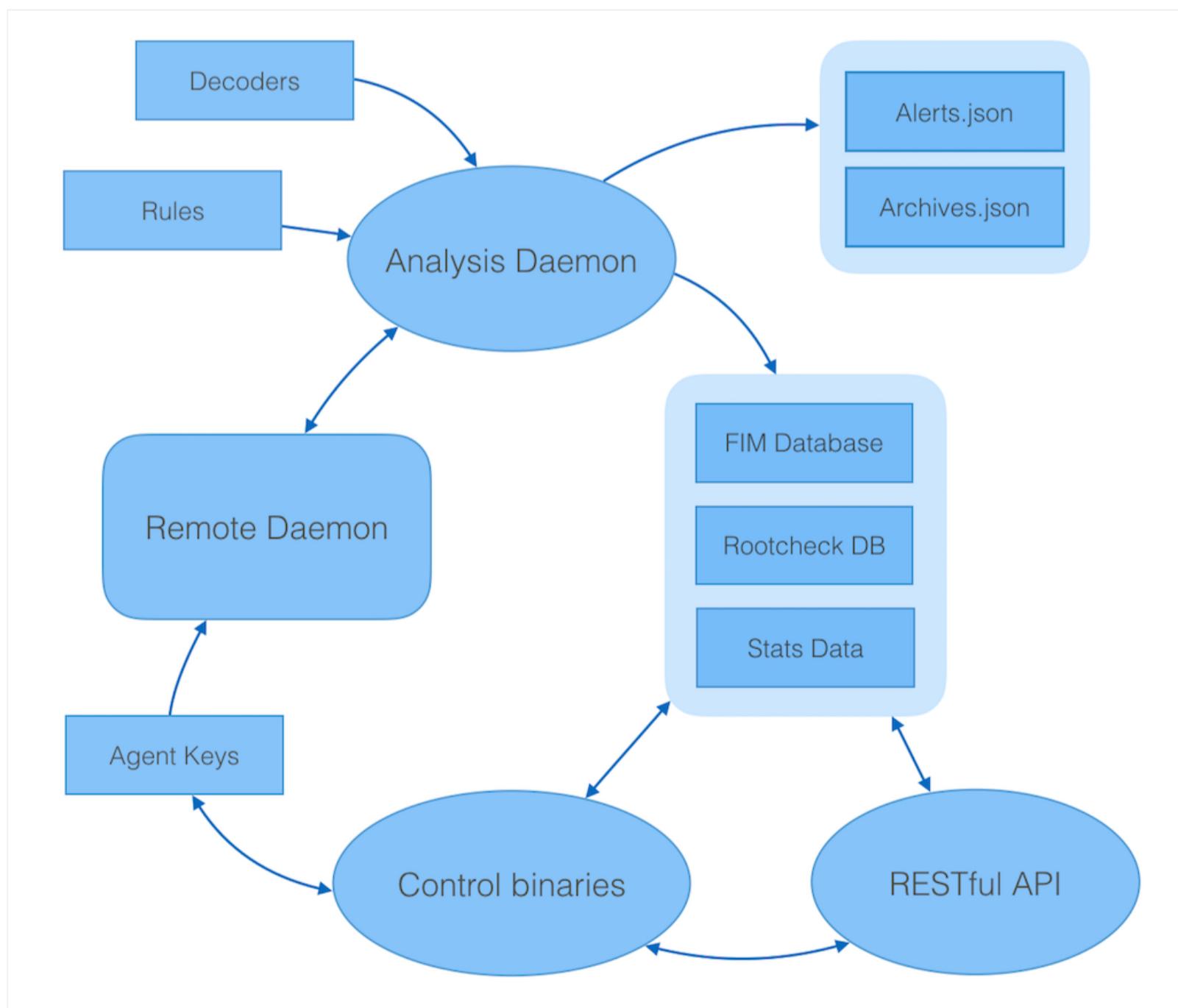


All agent processes have different purposes and settings. Here is a brief description of what is done by each of them:

- **Rootcheck:** This process performs multiple tasks related to the detection of rootkits, malware and system anomalies. It also runs certain basic security checks against system configuration files.
- **Log Collector:** This agent component is used to read operating system and application log messages, including flat log files, standard Windows event logs, and even Windows Event Channels. It can also be configured to periodically run and capture the output of specific commands.
- **Syscheck:** This process performs file integrity monitoring (FIM). It can also monitor registry keys on Windows systems. It is capable of detecting changes in a file's content, ownership, and other attributes, as well as noting creation and deletion of files. While it performs periodic FIM scans by default, it can also be configured to communicate with the operating system kernel to do real-time detection of file changes, as well to realize detailed changes report (diffs) of text files.
- **OpenSCAP:** This module uses published [OVAL](#) (Open Vulnerability Assessment Language) and [XCCDF](#) (Extensible Configuration Checklist Description Format) baseline security profiles, by periodically scan the system it can find vulnerable applications or configurations that do not follow well-known standards such as those defined in [CIS](#) (Center for Internet Security) benchmarks.
- **Agent Daemon:** This is the process that receives the data generated or collected by all other agent components. It compresses, encrypts and delivers the data to the server through an authenticated channel. This process runs in an isolated “chroot” (change root) environment, meaning that it has limited access to the monitored system. This improves the overall security of the agent because is the only process that connects to the network.

Wazuh server

The server component is in charge of analyzes the data received from the agents, triggering alerts when an event matches a rule (e.g. intrusion detected, file changed, configuration not compliant with policy, possible rootkit, etc...).



The server usually runs on a stand-alone physical machine, virtual machine, or cloud instance. It also locally runs agent components with the purpose of monitoring itself. Below is a list of the main server components:

- **Registration service:** This is used to register new agents by provisioning and distributing pre-shared authentication keys, these keys are unique to each agent. This process runs as a network service and supports authentication via TLS/SSL and/or by a fixed password.

- **Remote daemon service:** This is the service that receives data from the agents. It makes use of the pre-shared keys to validate each agent's identity and to encrypt communications with them.
- **Analysis daemon:** This is the process that performs data analysis. It utilizes decoders to identify the type of information being processed (e.g. Windows events, SSHD logs, web server logs...), and then extract relevant data elements from the log messages (e.g. source ip, event id, user...). Next, by using rules identify specific patterns in the decoded log records which would trigger alerts and possibly even call for automated countermeasures (active response) like an IP ban on the firewall.
- **RESTful API:** This provides an interface to manage and monitor the configuration and deployment status of agents. It is also used by the Wazuh web interface, which is a Kibana app.

Elastic Stack

Elastic Stack is a unified suite of popular open source projects for log management, including Elasticsearch, Logstash, Kibana, Filebeat, and others. The parts especially relevant to the Wazuh solution are:

- **Elasticsearch:** A highly scalable full-text search and analytics engine. Elasticsearch is distributed, what means that data (indices) are divided into shards and each shard can have zero or more replicas.
- **Logstash:** A tool to collect and parse logs to be saved into a storage system (e.g., Elasticsearch). Collected events can also be enriched and transformed using input, filter and output plugins.
- **Kibana:** A flexible and intuitive web interface for mining, analyzing, and visualizing data. It runs on top of the content indexed on an Elasticsearch cluster.
- **Filebeat:** A lightweight forwarder used to convey logs across a network, usually to Logstash or Elasticsearch.

Wazuh integrates with Elastic Stack to provide a feed of already decoded log messages to be indexed by Elasticsearch, as well as a real-time web console for alert and log data analysis. In addition, Wazuh user interface (running on top of Kibana) can be used for management and monitoring of your Wazuh infrastructure.

An Elasticsearch *index* is a collection of documents that have somewhat similar characteristics (like certain common fields and shared data retention requirements). Wazuh utilizes as many as three different indices, created daily, to store different type of events:

- **wazuh-alerts:** Index for alerts generated by the Wazuh server each time an event trips a rule.
- **wazuh-events:** Index for all events (archive data) received from the agents, whether or not they trip a rule.
- **wazuh-monitoring:** Index for data related to agent status over time. It is used by the web interface to represent when individual agents are or have been "Active", "Disconnected", or "Never connected".

An index is composed of documents. For the indices above, documents are individual alerts, archived events, or status events.

An Elasticsearch index is split up into one or more shards, and each shard can optionally have one or more replicas. Each primary and replica shard is an individual Lucene index. Thus an Elasticsearch index is made up of many Lucene indexes. When a search is run on an Elasticsearch index, the search is executed on all the shards in parallel, and the results are merged. Splitting up Elasticsearch indexes into multiple shards and replicas is used in multiple-node Elasticsearch clusters with the purpose of scaling out searches and high availability. Single-node Elasticsearch clusters normally have only one shard per index and no replicas.

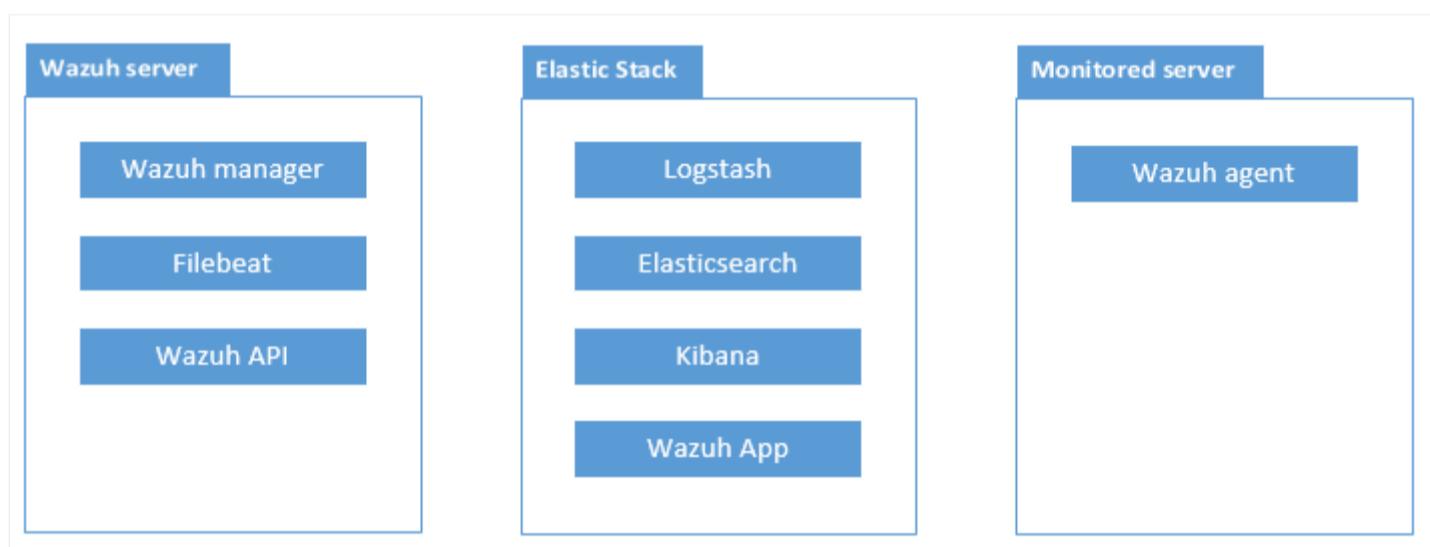
Architecture

The Wazuh architecture is based on agents running on monitored hosts that forward logs data to a central server. Also, agentless devices (such as firewalls, switches, routers, access points, etc.) are supported, they could actively submit log data via syslog and/or periodically probe their configuration changes to later forward the data to the central server. The central server decodes and analyzes the incoming information and passes the results along to an Elasticsearch cluster for indexing and storage.

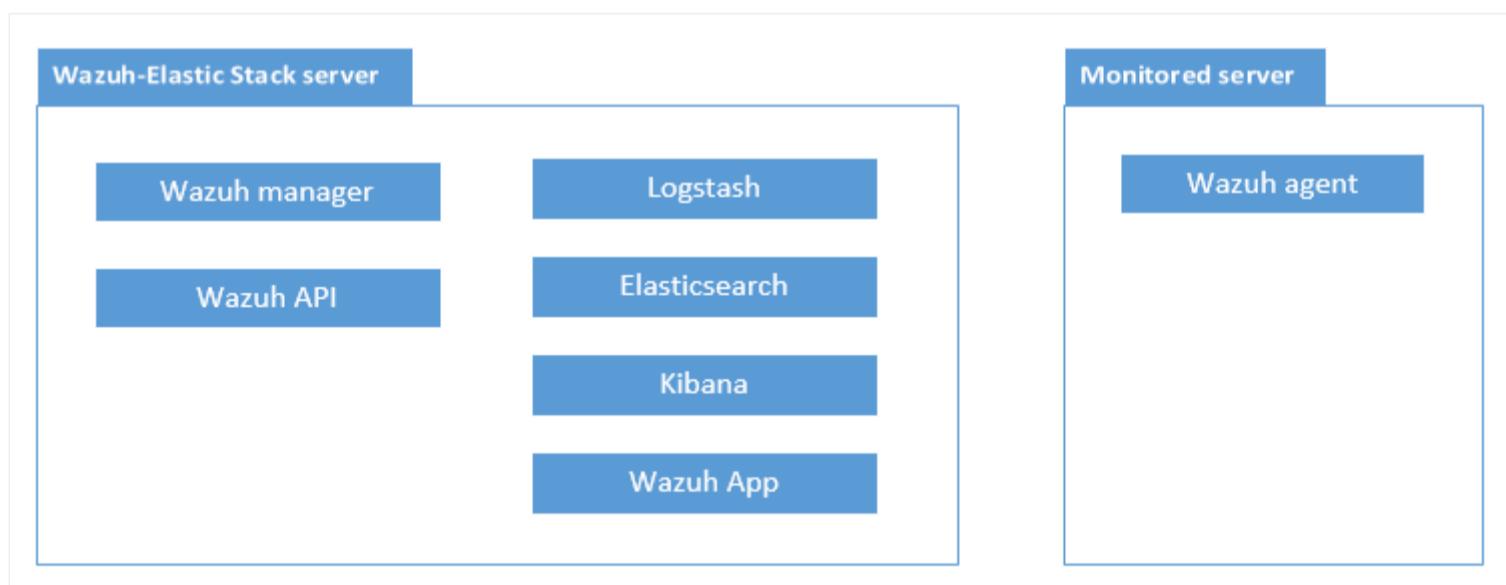
An Elasticsearch cluster is a collection of one or more nodes (servers) that communicate with each other to perform read and write operations on indexes. Small Wazuh deployments (<50 agents), can easily be handled by a single-node cluster. Multi-node clusters are recommended when there is a large number of monitored systems, when a large volume of data is planned on, and/or when high availability is required.

When the Wazuh server and the Elasticsearch cluster are on different hosts, Filebeat is used to securely forward Wazuh alerts and/or archived events to the Elasticsearch server(s) using TLS encryption.

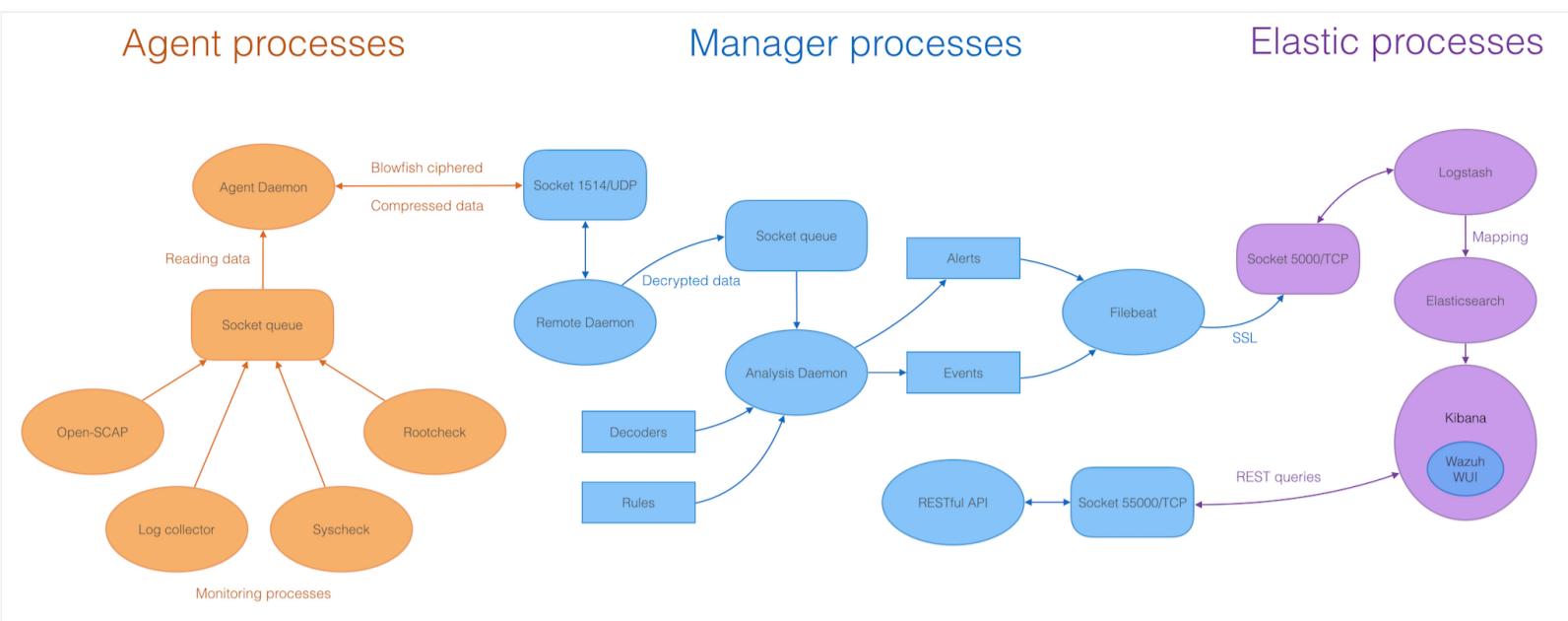
See below how components are distributed when the Wazuh server and the Elasticsearch cluster run on different hosts. Note that with multi-node clusters there will be multiple Elastic Stack servers to which Filebeat is capable of forwarding data:



In smaller Wazuh deployments, Wazuh and Elastic Stack with a single-node Elasticsearch cluster, can all be deployed on a single server. In this scenario, Logstash can read the Wazuh alerts and/or archived events directly from the local file system and feed them into the local Elasticsearch instance.



Communications and data flow



Agent-server communication

Wazuh agents use the OSSEC message protocol to send collected events to the Wazuh server over port 1514 (UDP or TCP). The Wazuh server then decodes and rule-checks the received events with the analysis engine. Events that trip a rule are augmented with alert data such as rule id and rule name. Events can be spooled to one or both of the following files, depending on whether or not tripped a rule:

- The file `/var/ossec/logs/archives/archives.json` contains all events whether they tripped a rule or not.
- The file `/var/ossec/logs/alerts/alerts.json` contains only events that tripped a rule.

Note

Alerts will be duplicated if you use both of these files. Also, note that both files receive fully decoded events data.

The OSSEC message protocol uses a 192-bit Blowfish encryption with a full 16-round implementation, at this time has no publicly known cryptographic weaknesses.

Wazuh-elastic communication

In larger deployments, the Wazuh server uses Filebeat to ship alert and event data to Logstash (5000/TCP) on the Elastic Stack server, using TLS encryption. For a single-host architecture, Logstash is able to read the events/alerts directly from the local filesystem without using Filebeat.

Logstash formats the incoming data, and optionally enriches it with GeoIP information, before sending it along to Elasticsearch (port 9200/TCP). Once the data is indexed into Elasticsearch, Kibana (port 5601/TCP) is used to mine and visualize the information.

The Wazuh App runs inside Kibana constantly querying the RESTful API (port 55000/TCP on the Wazuh manager) in order to display configuration, and status related information of the server and agents, as well to restart agents when desired. This communication is encrypted with TLS and authenticated with username and password.

Archival data storage

Alerts and non-alert events are together stored in files on the Wazuh server in addition to being sent to Elasticsearch. These files can be written in JSON format (.json) and/or in plain text format (.log - no decoded fields but more compact). These files are daily compressed and signed using MD5 and SHA1 checksums. The directory and filename structure is as follows:

```
root@wazuh-server:/var/ossec/logs/archives/2017/Jan# ls -l
total 176
-rw-r----- 1 ossec ossec 234350 Jan  2 00:00 ossec-archive-01.json.gz
-rw-r----- 1 ossec ossec    350 Jan  2 00:00 ossec-archive-01.json.sum
-rw-r----- 1 ossec ossec 176221 Jan  2 00:00 ossec-archive-01.log.gz
-rw-r----- 1 ossec ossec    346 Jan  2 00:00 ossec-archive-01.log.sum
-rw-r----- 1 ossec ossec 224320 Jan  2 00:00 ossec-archive-02.json.gz
-rw-r----- 1 ossec ossec    350 Jan  2 00:00 ossec-archive-02.json.sum
-rw-r----- 1 ossec ossec 151642 Jan  2 00:00 ossec-archive-02.log.gz
-rw-r----- 1 ossec ossec    346 Jan  2 00:00 ossec-archive-02.log.sum
-rw-r----- 1 ossec ossec 315251 Jan  2 00:00 ossec-archive-03.json.gz
-rw-r----- 1 ossec ossec    350 Jan  2 00:00 ossec-archive-03.json.sum
-rw-r----- 1 ossec ossec 156296 Jan  2 00:00 ossec-archive-03.log.gz
-rw-r----- 1 ossec ossec    346 Jan  2 00:00 ossec-archive-03.log.sum
```

Rotation and backups of archive files is recommended, according to the storage capacity of the Wazuh Manager server. Using *cron* jobs, you could easily arrange to keep only a certain time window of archive files locally on the Manager (e.g., last year or last three months).

On the other hand, you may choose to dispense with storing archive files at all and simply rely on Elasticsearch for archive storage, especially if you are already running periodic Elasticsearch snapshot backups and/or a multi-node Elasticsearch cluster with shard replicas for high availability. You could even use a *cron* job to move snapshotted indexes to a final data storage server and sign them using MD5 and SHA1 algorithms.

Use cases

Wazuh is often used to meet compliance requirements (such PCI DSS or HIPAA) and configuration standards (CIS hardening guides). It is also popular among IaaS users (eg. Amazon AWS, Azure or Google cloud), where deploying a host-based IDS in the running instances can be combined with the analysis of the infrastructure events (pulled directly from the cloud provider API).

Here is a list of common use cases:

1. [Signature-based log analysis](#)
2. [File integrity monitoring](#)
3. [Rootkits detection](#)
4. [Security policy monitoring](#)

Signature-based log analysis

Automated log analysis and management accelerate threat detection. There are many cases where evidence of an attack can be found in the logs of your devices, systems, and applications. Wazuh can be used to automatically aggregate and analyze log data.

The Wazuh agent (running on the monitored host) is usually the one in charge of reading operating system and application log messages, forwarding those to the Wazuh server, where the analysis takes place. When no agent is deployed, the server can also receive data via syslog, from network devices or applications.

Wazuh uses decoders to identify the source application of the log message and then analyzes it using application specific rules. Here is an example of a rule used to detect SSH authentication failure events:

```
<rule id="5716" level="5">
  <if_sid>5700</if_sid>
  <match>^Failed|error: PAM: Authentication</match>
  <description>SSHD authentication failed.</description>
  <group>authentication_failed,pci_dss_10.2.4,pci_dss_10.2.5,</group>
</rule>
```

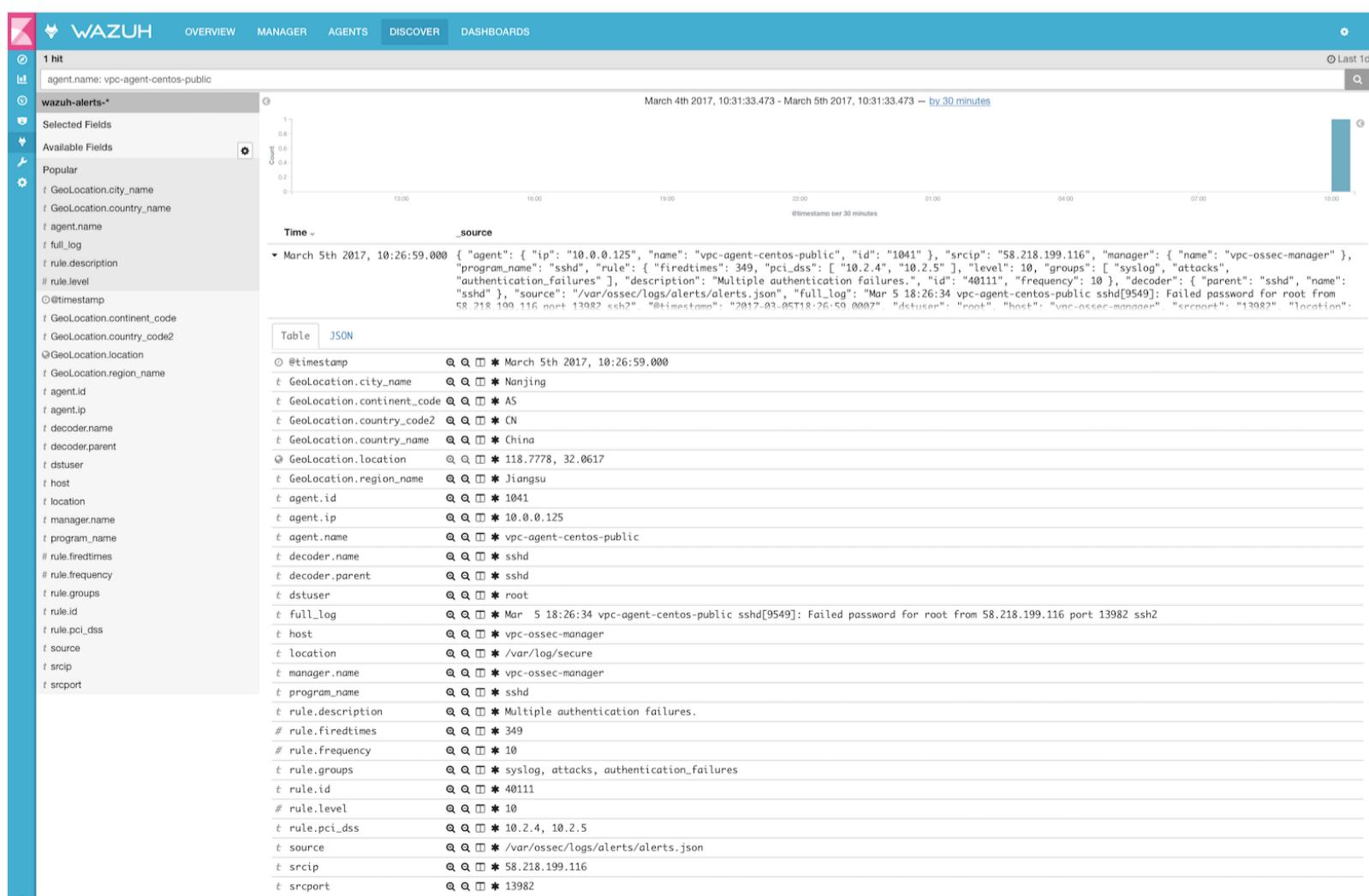
Rules include a `match` field, used to define the pattern the rule is going to be looking for. It also has a `level` field that specifies the resulting alert priority.

The manager will generate an alert every time an event collected by one of the agents (or via syslog) matches a rule (with level higher than zero).

Here is an example found in `/var/ossec/logs/alerts/alerts.json`:

```
{
  "agent": {
    "id": "1041",
    "ip": "10.0.0.125",
    "name": "vpc-agent-centos-public"
  },
  "decoder": {
    "name": "sshd",
    "parent": "sshd"
  },
  "dstuser": "root",
  "full_log": "Mar 5 18:26:34 vpc-agent-centos-public sshd[9549]: Failed password for root from 58.218.199.116 port 13982 ssh2",
  "location": "/var/log/secure",
  "manager": {
    "name": "vpc-ossec-manager"
  },
  "program_name": "sshd",
  "rule": {
    "description": "Multiple authentication failures.",
    "firedtimes": 349,
    "frequency": 10,
    "groups": [
      "syslog",
      "attacks",
      "authentication_failures"
    ],
    "id": "40111",
    "level": 10,
    "pci_dss": [
      "10.2.4",
      "10.2.5"
    ]
  },
  "srcip": "58.218.199.116",
  "srcport": "13982",
  "timestamp": "2017-03-05T10:26:59-0800"
}
}
```

Once generated by the manager, the alerts are sent to the Elastic Stack component where they are enriched with Geolocation information, stored and indexed. Kibana can be then used to search, analyze and visualize the data. See below an alert as displayed in the interface:



Wazuh provides a default ruleset, updated periodically, with over 1,600 rules for different applications.

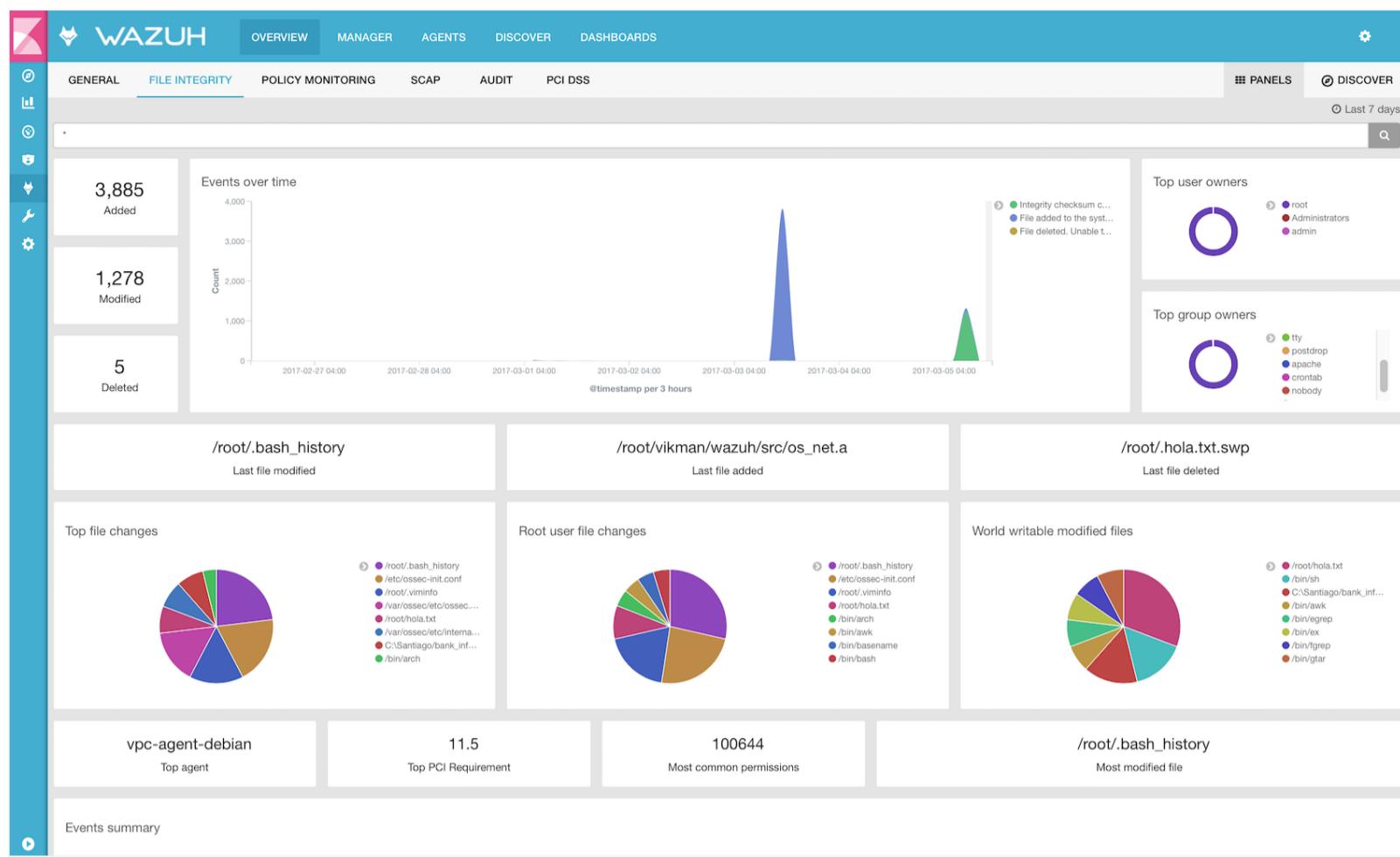
File integrity monitoring

File integrity monitoring (FIM) component detects and alerts when operating system and application files are modified. This capability is often used to detect access or changes to sensitive data. In fact, if your servers are in scope with PCI DSS, the requirement 11.5 states that you must install a file integrity monitoring solution to pass your audit.

Below is an example of an alert, generated when a monitored file is changed. Metadata includes MD5 and SHA1 checksums, file sizes (before and after the change), file permissions, file owner and content changes.

```
{
  "agent": {
    "id": "003",
    "ip": "10.0.0.121",
    "name": "vpc-agent-debian"
  },
  "decoder": {
    "name": "syscheck_integrity_changed"
  },
  "full_log": "Integrity checksum changed for: '/root/hola.txt'\nSize changed from '3089' to '3213'\nOld md5sum was: '20db2c4c9bdd937975371bc5ca25af92'\nNew md5sum is : '3841e727a28f733e6d34413afd49d607'\nOld sha1sum was: 'a6c57142a6e6e7e55c58b3174ee52b4f8ec996e3'\nNew sha1sum is :\n'99aa4b60467a932c89f32603e410a6c194fb1ac3'\n",
  "location": "syscheck",
  "manager": {
    "name": "vpc-ossec-manager"
  },
  "rule": {
    "description": "Integrity checksum changed.",
    "firetimes": 8,
    "groups": [
      "ossec",
      "syscheck"
    ],
    "id": "550",
    "level": 7,
    "pci_dss": [
      "11.5"
    ]
  },
  "syscheck": {
    "diff": "\0a1,2\n> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua\n> \n",
    "event": "modified",
    "gid_after": "0",
    "gname_after": "root",
    "inode_after": 398585,
    "md5_after": "3841e727a28f733e6d34413afd49d607",
    "md5_before": "20db2c4c9bdd937975371bc5ca25af92",
    "mtime_after": "2017-03-05T13:47:32",
    "mtime_before": "2017-03-05T13:44:15",
    "path": "/root/hola.txt",
    "perm_after": "100666",
    "sha1_after": "99aa4b60467a932c89f32603e410a6c194fb1ac3",
    "sha1_before": "a6c57142a6e6e7e55c58b3174ee52b4f8ec996e3",
    "size_after": "3213",
    "size_before": "3089",
    "uid_after": "1000",
    "uname_after": "admin"
  },
  "timestamp": "2017-03-05T13:44:18-0800"
}
```

A good summary of file changes, can be found in the file integrity monitoring dashboard, which provides drill-down capabilities to get all details of the alerts triggered.



Rootkits detection

Wazuh agent periodically scans the monitored system to detect rootkits both at a kernel and user level. This type of malware usually replaces or changes existing operating system components in order to alter the behavior of the system, it can hide other processes, files or network connections like itself.

Wazuh uses different detection mechanisms to look for system anomalies or well-known intrusions. This is done periodically by the *Rootcheck* component:

Action	Detection mechanism	Binary	System call
Detection of hidden processes	Comparing output of system binaries and system calls	ps	setsid()
			getpgid()
			kill()
Detection of hidden files	Comparing output of system binaries and system calls	ls	stat()
	Scanning /dev		opendir()
Detection of hidden ports	Comparing output of system binaries and system calls	netstat	readdir()
			bind()
Detection of known rootkits	Using a malicious file database		stat()
			fopen()
	Inspecting files content using signatures		opendir()
	Detecting file permission and ownership anomalies	fopen()	
			stat()

Below is an example of an alert generated when a hidden process is found. In this case, the affected system is running a Linux kernel-level rootkit (named Diamorphine):

```
{  
  "agent": {  
    "id": "1030",  
    "ip": "10.0.0.59",  
    "name": "diamorphine-POC"  
  },  
  "decoder": {  
    "name": "rootcheck"  
  },  
  "full_log": "Process '562' hidden from /proc. Possible kernel level rootkit.",  
  "location": "rootcheck",  
  "manager": {  
    "name": "vpc-ossec-manager"  
  },  
  "rule": {  
    "description": "Host-based anomaly detection event (rootcheck).",  
    "firedtimes": 4,  
    "groups": [  
      "ossec",  
      "rootcheck"  
    ],  
    "id": "510",  
    "level": 7  
  },  
  "timestamp": "2017-03-05T15:13:04-0800",  
  "title": "Process '562' hidden from /proc."  
}
```

Security policy monitoring

SCAP is a standardized compliance checking solution for enterprise-level infrastructure. It is a line of specifications maintained by the National Institute of Standards and Technology (NIST) with the purpose of maintaining enterprise systems security.

OpenSCAP is an auditing tool that utilizes the Extensible Configuration Checklist Description Format (XCCDF). XCCDF is a standard way of expressing checklist content and defines security checklists. It also combines with other specifications such as CPE, CVE, CCE, and OVAL, to create SCAP-expressed checklist that can be processed by SCAP-validated products.

Wazuh agent uses OpenSCAP internally to verify that systems conform to CIS hardening standards. Below is an example of an SCAP rule used to check if SSH daemon is configured to allow empty passwords:

```

<ns10:Rule id="xccdf_org.ssgproject.content_rule_sshd_disable_empty_passwords" selected="false"
severity="high">
  <ns10:title xml:lang="en-US">Disable SSH Access via Empty Passwords</ns10:title>
  <ns10:description xml:lang="en-US">To explicitly disallow remote login from accounts with empty passwords,
add or correct the following line in <html:code>/etc/ssh/sshd_config</html:code>:
<html:pre>PermitEmptyPasswords no</html:pre> Any accounts with empty passwords should be disabled
immediately, and PAM configuration should prevent users from being able to assign themselves empty passwords.
  </ns10:description>
  <ns10:reference href="http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf">AC-
3</ns10:reference>
  <ns10:reference href="http://iase.disa.mil/stigs/cci/Pages/index.aspx">765</ns10:reference>
  <ns10:reference href="http://iase.disa.mil/stigs/cci/Pages/index.aspx">766</ns10:reference>
  <ns10:rationale xml:lang="en-US">Configuring this setting for the SSH daemon provides additional assurance
that remote login via SSH will require a password, even in the event of misconfiguration elsewhere.
</ns10:rationale>
  <ns10:fix complexity="low" disruption="low" id="sshd_disable_empty_passwords" reboot="false"
strategy="enable" system="urn:xccdf:fix:script:sh">grep -q ^PermitEmptyPasswords /etc/ssh/sshd_config
&& sed -i "s/PermitEmptyPasswords.*/PermitEmptyPasswords no/g" /etc/ssh/sshd_config; if ! [ $? -eq
0 ]; then; echo "PermitEmptyPasswords no" &gt;&gt; /etc/ssh/sshd_config; fi
</ns10:fix>
  <ns10:check system="http://oval.mitre.org/XMLSchema/oval-definitions-5">
    <ns10:check-content-ref href="ssg-rhel6-oval.xml" name="oval:ssg:sshd_disable_empty_passwords:def:1" />
  </ns10:check>
  <ns10:check system="http://scap.nist.gov/schema/ocil/2">
    <ns10:check-content-ref href="ssg-rhel6-ocil.xml" name="ocil:ssg-
sshd_disable_empty_passwords_ocil:questionnaire:1" />
  </ns10:check>
</ns10:Rule>

```

SCAP checks are run periodically (default is once a day), and results are set to the Wazuh server where they are processed through OpenSCAP decoders and rules. Below is an example of an alert, generated when Linux audit policies (auditd) are not configured to monitor user actions:

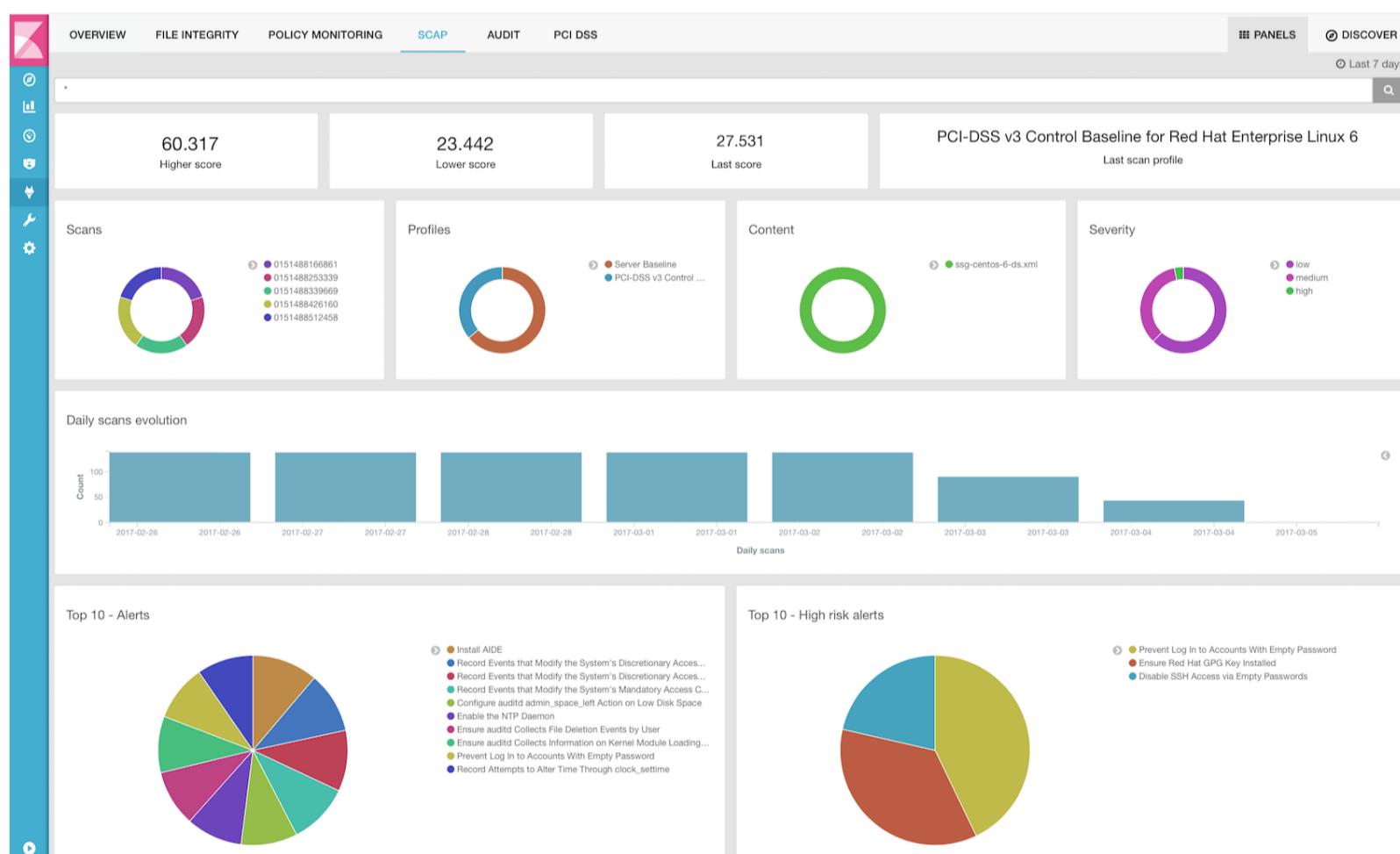
```
{
  "agent": {
    "id": "1040",
    "ip": "10.0.0.76",
    "name": "ip-10-0-0-76"
  },
  "decoder": {
    "name": "oscap",
    "parent": "oscap"
  },
  "full_log": "oscap: msg: \"xccdf-result\", scan-id: \"10401488754797\", content: \"ssg-centos-7-ds.xml\", title: \"Ensure auditd Collects System Administrator Actions\", id: \"xccdf_org.ssgproject.content_rule_audit_rules_sysadmin_actions\", result: \"fail\", severity: \"low\", description: \"At a minimum the audit system should collect administrator actions for all users and root. If the auditd daemon is configured to use the augenrules program to read audit rules during daemon startup (the default), add the following line to a file with suffix .rules in the directory /etc/audit/rules.d: -w /etc/sudoers -p wa -k actions If the auditd daemon is configured to use the auditctl utility to read audit rules during daemon startup, add the following line to /etc/audit/audit.rules file: -w /etc/sudoers -p wa -k actions\", rationale: \"The actions taken by system administrators should be audited to keep a record of what was executed on the system, as well as, for accountability purposes.\\" references: \"AC-2(7)(b) (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), AC-17(7) (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), AU-1(b) (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), AU-2(a) (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), AU-2(c) (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), AU-2(d) (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), AU-12(a) (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), AU-12(c) (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), IR-5 (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), 126 (http://iase.disa.mil/stigs/cci/Pages/index.aspx), Test attestation on 20121024 by DS (https://github.com/OpenSCAP/scap-security-guide/wiki/Contributors)\", identifiers: \"CCE-RHEL7-CCE-TBD (http://cce.mitre.org)\", oval-id: \"oval:ssg:def:370\", benchmark-id: \"xccdf_org.ssgproject.content_benchmark_RHEL-7\", profile-id: \"xccdf_org.ssgproject.content_profile_common\", profile-title: \"Common Profile for General-Purpose Systems\".", "location": "wodle_open-scap",
    "manager": {
      "name": "vpc-ossec-manager"
    },
    "oscap": {
      "check": {
        "description": "At a minimum the audit system should collect administrator actions for all users and root. If the auditd daemon is configured to use the augenrules program to read audit rules during daemon startup (the default), add the following line to a file with suffix .rules in the directory /etc/audit/rules.d: -w /etc/sudoers -p wa -k actions If the auditd daemon is configured to use the auditctl utility to read audit rules during daemon startup, add the following line to /etc/audit/audit.rules file: -w /etc/sudoers -p wa -k actions",
        "id": "xccdf_org.ssgproject.content_rule_audit_rules_sysadmin_actions",
        "identifiers": "CCE-RHEL7-CCE-TBD (http://cce.mitre.org)",
        "oval": {
          "id": "oval:ssg:def:370"
        },
        "rationale": "The actions taken by system administrators should be audited to keep a record of what was executed on the system, as well as, for accountability purposes.",
        "references": "AC-2(7)(b) (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), AC-17(7) (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), AU-1(b) (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), AU-2(a) (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), AU-2(c) (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), AU-2(d) (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), AU-12(a) (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), AU-12(c) (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), IR-5 (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), 126 (http://iase.disa.mil/stigs/cci/Pages/index.aspx), Test attestation on 20121024 by DS (https://github.com/OpenSCAP/scap-security-guide/wiki/Contributors)",
        "result": "fail",
        "severity": "low",
        "title": "Ensure auditd Collects System Administrator Actions"
      }
    }
  }
}
```

```

},
"scan": {
    "benchmark": {
        "id": "xccdf_org.ssgproject.content_benchmark_RHEL-7"
    },
    "content": "ssg-centos-7-ds.xml",
    "id": "10401488754797",
    "profile": {
        "id": "xccdf_org.ssgproject.content_profile_common",
        "title": "Common Profile for General-Purpose Systems"
    }
}
},
"rule": {
    "description": "OpenSCAP: Ensure auditd Collects System Administrator Actions (not passed)",
    "firedtimes": 3,
    "groups": [
        "oscap",
        "oscap-result"
    ],
    "id": "81529",
    "level": 5,
    "pci_dss": [
        "2.2"
    ]
},
"timestamp": "2017-03-05T15:00:03-0800"
}

```

In addition, Wazuh WUI can be used to visualize and analyze policy monitoring scan results. For example, here is a screenshot of data collected from a CentOS system when scanning it using **Server baseline** and **PCI DSS v3** pre-defined profiles:



Set up Puppet

Before we get started with Puppet, confirm the following network requirements are met:

- **Private network DNS:** Forward and reverse DNS must be configured, and every server must have a unique hostname. If you do not have DNS configured, you must use your hosts file for name resolution. We will assume that you will use your private network for communication within your infrastructure.
- **Firewall open ports:** The Puppet master must be reachable on TCP port 8140.

Contents

- [Installing Puppet master](#)
- [Installing Puppet agent](#)
- [Setting up Puppet certificates](#)

Wazuh Puppet module

This [module](#) has been authored by Nicolas Zin and updated by Jonathan Gazeley and Michael Porter. Wazuh has forked it with the purpose of maintaining it. Thank you to the authors for the contribution.

Install Wazuh module

Download and install the Wazuh module from Puppet Forge:

```
$ sudo puppet module install wazuh-wazuh
Notice: Preparing to install into /etc/puppet/modules ...
Notice: Downloading from https://forgeapi.puppetlabs.com ...
Notice: Installing -- do not interrupt ...
/etc/puppet/modules
└── wazuh-wazuh (v2.0.21)
    ├── puppet-selinux (v0.8.0)
    ├── puppetlabs-apt (v2.2.0)
    ├── puppetlabs-concat (v1.2.4)
    ├── puppetlabs-stdlib (v4.9.0)
    └── stahnma-epel (v1.1.1)
```

This module installs and configures Wazuh agent and manager.

Install manager via Puppet

The manager is configured by installing the `wazuh::server` class, and optionally using:

- `wazuh::command`: to define active response command (like `firewall-drop.sh`).
- `wazuh::activeresponse`: to link rules to active response commands.
- `wazuh::addlog`: to define additional log files to monitor.

Here is an example of a manifest `wazuh-manager.pp`:

```
node "server.yourhost.com" {
  class { 'wazuh::server':
    smtp_server => 'localhost',
    ossec_emailto => ['user@mycompany.com'],
  }

  wazuh::command { 'firewallblock':
    command_name      => 'firewall-drop',
    command_executable => 'firewall-drop.sh',
    command_expect     => 'srcip'
  }

  wazuh::activeresponse { 'blockWebattack':
    command_name => 'firewall-drop',
    ar_level      => 9,
    ar_agent_id   => 123,
    ar_rules_id   => [31153,31151],
    ar_repeated_offenders => '30,60,120'
  }

  wazuh::addlog { 'monitorLogFile':
    logfile => '/var/log/secure',
    logtype => 'syslog'
  }
}
```

Place the file at `/etc/puppetlabs/code/environments/production/manifests/` in your Puppet master and it will be executed in the specified node after the `runinterval` time set in `puppet.conf`.

Install agent via Puppet

The agent is configured by installing the `wazuh::client` class.

Here is an example of a manifest `wazuh-agent.pp` (please replace with your IP address):

```
node "client.yourhost.com" {

  class { "wazuh::client":
    ossec_server_ip => "192.168.209.166"
  }

}
```

Place the file at `/etc/puppetlabs/code/environments/production/manifests/` in your Puppet master and it will be executed in the specified node after the `runinterval` time set in `puppet.conf`.

Reference Wazuh puppet

Sections	Functions
Wazuh server class	<code>email_alert</code> <code>command</code> <code>activeresponse</code> <code>addlog</code>
Wazuh agent class	<code>addlog</code>
<code>ossec_scanpaths</code> configuration	

Contents

- [Scan paths configuration](#)
- [Wazuh agent class](#)
- [Wazuh server class](#)

Migrating OSSEC manager installed from packages

1. Backup your current configuration

Stop OSSEC:

```
$ /var/ossec/bin/ossec-control stop
```

Check if you have enough space to create a copy of `/var/ossec`:

```
$ du -h /var/ossec | tail -n1
$ df -h /var
```

Backup `/var/ossec`:

```
$ cp -rp /var/ossec /var/ossec_backup
```

2. Remove your current installation

Debian and Ubuntu:

```
$ apt-get remove ossec-hids --purge
```

CentOS and Red Hat:

```
$ yum remove ossec-hids
```

Remove directory:

```
$ rm -rf /var/ossec
```

3. Install Wazuh server

Follow the next guide in order to install Wazuh server:

- [Install Wazuh server with RPM packages](#)
- [Install Wazuh server with Deb packages](#)

4. Restore configuration

Stop OSSEC:

```
$ systemctl stop wazuh-manager
```

Restore mandatory files:

```
$ cp -p /var/ossec_backup/agentless/.passlist /var/ossec/agentless/
$ cp -p /var/ossec_backup/etc/client.keys /var/ossec/etc/
$ cp -p /var/ossec_backup/etc/ossec.conf /var/ossec/etc/ossec.conf.orig
$ cp -p /var/ossec_backup/etc/local_internal_options.conf /var/ossec/etc/local_internal_options.conf
$ cp -p /var/ossec_backup/etc/local_decoder.xml /var/ossec/etc/decoders/local_decoder.xml
$ cp -p /var/ossec_backup/etc/shared/agent.conf /var/ossec/etc/shared/agent.conf
$ cp -p /var/ossec_backup/rules/local_rules.xml /var/ossec/etc/rules/local_rules.xml
$ cp -p /var/ossec_backup/queue/rids/sender_counter /var/ossec/queue/rids/sender_counter
```

Restore optional files

The following files are required in order to preserve alerts log files and syscheck/rootcheck databases:

```
$ cp -rp /var/ossec_backup/logs/archives/* /var/ossec/logs/archives
$ cp -rp /var/ossec_backup/logs/alerts/* /var/ossec/logs/alerts
$ cp -rp /var/ossec_backup/queue/rootcheck/* /var/ossec/queue/rootcheck
$ cp -rp /var/ossec_backup/queue/syscheck/* /var/ossec/queue/syscheck
```

5. Review ossec.conf

The previous configuration file is saved as [/var/ossec/etc/ossec.conf.orig](#). You should review the new configuration file [/var/ossec/etc/ossec.conf](#) with the old one in case that you want to add some setting from the previous configuration.

6. Start Wazuh

```
$ /var/ossec/bin/ossec-control start
```

Migrating OSSEC agent installed from packages

1. Backup your current configuration

Stop OSSEC:

```
$ /var/ossec/bin/ossec-control stop
```

Check if you have enough space to create a copy of `/var/ossec`:

```
$ du -h /var/ossec | tail -n1  
$ df -h /var
```

Backup `/var/ossec`:

```
$ cp -rp /var/ossec /var/ossec_backup
```

2. Remove your current installation

Debian and Ubuntu:

```
$ apt-get remove ossec-hids-agent --purge
```

CentOS and Red Hat:

```
$ yum remove ossec-hids-agent
```

Remove directory:

```
$ rm -rf /var/ossec
```

3. Install Wazuh agent

Follow the next guide in order to install Wazuh agent:

- [Install Wazuh agent with RPM packages](#)
- [Install Wazuh agent with Deb packages](#)

4. Restore configuration

Stop OSSEC:

```
$ systemctl stop wazuh-agent
```

Restore files:

```
$ cp -p /var/ossec_backup/etc/ossec.conf /var/ossec/etc/ossec.conf.orig  
$ cp -p /var/ossec_backup/etc/local_internal_options.conf /var/ossec/etc/local_internal_options.conf  
$ cp -p /var/ossec_backup/etc/client.keys /var/ossec/etc/  
$ cp -p /var/ossec_backup/queue/rids/* /var/ossec/queue/rids/
```

5. Review ossec.conf

The previous configuration file is saved as `/var/ossec/etc/ossec.conf.orig`. You should review the new configuration file `/var/ossec/etc/ossec.conf` with the old one in case that you want to add some setting from the previous configuration.

Do not forget to restore the IP of the manager:

```
/var/ossec/etc/ossec.conf
```

```
<ossec_config>  
  <client>  
    <server-ip>MANAGER_IP</server-ip>
```

6. Start Wazuh

```
$ /var/ossec/bin/ossec-control start
```

Install Elastic Stack with RPM packages

The RPM packages are suitable for installation on Red Hat, CentOS and other RPM-based systems.

! Note

Many of the commands described below need to be executed with root user privileges.

Preparation

1. Oracle Java JRE is required by Logstash and Elasticsearch.

! Note

The following command accepts the necessary cookies to download Oracle Java JRE. Please, visit [Oracle Java 8 JRE Download Page](#) for more information.

```
$ curl -Lo jre-8-linux-x64.rpm --header "Cookie: oraclelicense=accept-securebackup-cookie"  
"https://download.oracle.com/otn-pub/java/jdk/8u191-b12/2787e4a523244c269598db4e85c51e0c/jre-8u191-linux-x64.rpm"
```

Now check if the package was download successfully:

```
$ rpm -qlp jre-8-linux-x64.rpm > /dev/null 2>&1 && echo "Java package downloaded successfully" || echo  
"Java package did not download successfully"
```

Finally, install the RPM package using yum:

```
$ yum install jre-8-linux-x64.rpm  
$ rm jre-8-linux-x64.rpm
```

2. Install the Elastic repository and its GPG key:

```
$ rpm --import https://packages.elastic.co/GPG-KEY-elasticsearch  
  
$ cat > /etc/yum.repos.d/elastic.repo << EOF  
[elastic-5.x]  
name=Elastic repository for 5.x packages  
baseurl=https://artifacts.elastic.co/packages/5.x/yum  
gpgcheck=1  
gpgkey=https://artifacts.elastic.co/GPG-KEY-elasticsearch  
enabled=1  
autorefresh=1  
type=rpm-md  
EOF
```

Elasticsearch

Elasticsearch is a highly scalable full-text search and analytics engine. For more info please see [Elasticsearch](#).

1. Install the Elasticsearch package:

```
$ yum install elasticsearch-5.6.5
```

2. Enable and start the Elasticsearch service:

a. For Systemd:

```
$ systemctl daemon-reload  
$ systemctl enable elasticsearch.service  
$ systemctl start elasticsearch.service
```

b. For SysV Init:

```
$ chkconfig --add elasticsearch  
$ service elasticsearch start
```

3. Load Wazuh Elasticsearch template:

```
$ curl https://raw.githubusercontent.com/wazuh/wazuh-kibana-app/2.1/server/startup/integration_files/template_file.json | curl -XPUT  
'http://localhost:9200/_template/wazuh' -H 'Content-Type: application/json' -d @-
```

4. Insert sample alert:

```
$ curl https://raw.githubusercontent.com/wazuh/wazuh-kibana-app/2.1/server/startup/integration_files/alert_sample.json | curl -XPUT "http://localhost:9200/wazuh-alerts-`date +%Y.%m.%d`/wazuh/sample" -H 'Content-Type: application/json' -d @-
```

! Note

It is recommended to edit the default configuration to improve the Elasticsearch performance. To do so, please see [Elasticsearch tuning](#).

Logstash

Logstash is the tool that will collect, parse, and forward to Elasticsearch for indexing and storage all logs generated by Wazuh server. For more info please see [Logstash](#).

1. Install the Logstash package:

```
$ yum install logstash-5.6.5
```

2. Download the Wazuh config for Logstash:

```
$ curl -so /etc/logstash/conf.d/01-wazuh.conf  
https://raw.githubusercontent.com/wazuh/wazuh/2.1/extensions/logstash/01-wazuh.conf
```

3. Download the Wazuh Logstash template:

```
$ curl -so /etc/logstash/wazuh-elasticsearch-template.json  
https://raw.githubusercontent.com/wazuh/wazuh/2.1/extensions/elasticsearch/wazuh-elasticsearch-template.json
```

4. Follow this step only if you are using a single-host architecture:

- Edit `/etc/logstash/conf.d/01-wazuh.conf`, commenting out the entire input section titled "Remote Wazuh Manager - Filebeat input" and uncommenting the entire input section titled "Local Wazuh Manager - JSON file input". This will set up Logstash to read the Wazuh `alerts.json` file directly from the local filesystem rather than expecting Filebeat on a separate server to forward the information in that file to Logstash.
- Because the Logstash user needs to read `alerts.json` file, please add it to OSSEC group by running:

```
$ usermod -a -G ossec logstash
```

Note

Follow the next steps if you use CentOS-6/RHEL-6 or Amazon AMI (logstash uses Upstart like service manager and need to be fixed, see [bug](#))

- 1) Edit the file `/etc/logstash/startup.options` and in the line `30` change the `LS_GROUP=logstash` to `LS_GROUP=ossec`.
- 2) Update the service with the new parameters run the command `/usr/share/logstash/bin/system-install`
- 3) Start Logstash again.

5. Enable and start the Logstash service:

a. For Systemd:

```
$ systemctl daemon-reload  
$ systemctl enable logstash.service  
$ systemctl start logstash.service
```

b. For SysV Init:

```
$ chkconfig --add logstash  
$ service logstash start
```

Note

If you are running Wazuh server and the Elastic Stack server on separate systems (**distributed architecture**), then it is important to configure encryption between Filebeat and Logstash. To do so, please see [Setting up SSL for Filebeat and Logstash](#).

Kibana is a flexible and intuitive web interface for mining and visualizing the events and archives stored in Elasticsearch. More info at [Kibana](#).

1. Install the Kibana package:

```
$ yum install kibana-5.6.5
```

2. Install the Wazuh App plugin for Kibana:

```
$ /usr/share/kibana/bin/kibana-plugin install https://packages.wazuh.com/wazuhapp/wazuhapp-2.1.1_5.6.5.zip
```

 **Warning**

The Kibana plugin installation process may take several minutes. Please wait patiently.

3. **Optional.** Kibana will listen only on the loopback interface (localhost) by default. To set up Kibana to listen on all interfaces, edit the file `/etc/kibana/kibana.yml`. Uncomment the setting `server.host` and change the value to:

```
server.host: "0.0.0.0"
```

 **Note**

It is recommended to set up an Nginx proxy for Kibana in order to use SSL encryption and to enable authentication. Instructions to set the proxy up can be found at [Setting up SSL and authentication for Kibana](#).

4. Enable and start the Kibana service:

a. For Systemd:

```
$ systemctl daemon-reload  
$ systemctl enable kibana.service  
$ systemctl start kibana.service
```

b. For SysV Init:

```
$ chkconfig --add kibana  
$ service kibana start
```

5. Disable the Elasticsearch repository:

We recommend to disable the Elasticsearch repository in order to prevent an upgrade to a newer Elastic Stack version due to possible breaking changes with our App, so you should do it as follow:

```
$ sed -i "s/^enabled=1/enabled=0/" /etc/yum.repos.d/elastic.repo
```

Connecting the Wazuh App with the API

Follow the next guide in order to connect the Wazuh App with the API:

- [Connect the Wazuh App with the API](#)

Install Elastic Stack with Debian packages

The DEB package is suitable for Debian, Ubuntu, and other Debian-based systems.

Note

Many of the commands described below need to be executed with root user privileges.

Preparation

1. Oracle Java JRE is required by Logstash and Elasticsearch:

a. For Debian:

```
$ echo "deb http://ppa.launchpad.net/webupd8team/java/ubuntu xenial main" | tee /etc/apt/sources.list.d/webupd8team-java.list
$ echo "deb-src http://ppa.launchpad.net/webupd8team/java/ubuntu xenial main" | tee -a /etc/apt/sources.list.d/webupd8team-java.list
$ apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv-keys EEA14886
```

b. For Ubuntu:

```
$ add-apt-repository ppa:webupd8team/java
```

2. Once the repository is added, install Java JRE:

```
$ apt-get update
$ apt-get install oracle-java8-installer
```

3. Install the Elastic repository and its GPG key:

```
$ apt-get install curl apt-transport-https
$ curl -s https://artifacts.elastic.co/GPG-KEY-elasticsearch | apt-key add -
$ echo "deb https://artifacts.elastic.co/packages/5.x/apt stable main" | tee /etc/apt/sources.list.d/elastic-5.x.list
$ apt-get update
```

Elasticsearch

Elasticsearch is a highly scalable full-text search and analytics engine. For more info please see [Elasticsearch](#).

1. Install the Elasticsearch package:

```
$ apt-get install elasticsearch=5.6.5
```

2. Enable and start the Elasticsearch service:

a. For Systemd:

```
$ systemctl daemon-reload  
$ systemctl enable elasticsearch.service  
$ systemctl start elasticsearch.service
```

b. For SysV Init:

```
$ update-rc.d elasticsearch defaults 95 10  
$ service elasticsearch start
```

3. Load Wazuh Elasticsearch template:

```
$ curl https://raw.githubusercontent.com/wazuh/wazuh-kibana-app/2.1/server/startup/integration_files/template_file.json | curl -XPUT  
'http://localhost:9200/_template/wazuh' -H 'Content-Type: application/json' -d @-
```

4. Insert sample alert:

```
$ curl https://raw.githubusercontent.com/wazuh/wazuh-kibana-app/2.1/server/startup/integration_files/alert_sample.json | curl -XPUT "http://localhost:9200/wazuh-alerts-`date +%Y.%m.%d`/wazuh/sample" -H 'Content-Type: application/json' -d @-
```

Note

It is recommended to edit the default configuration to improve the Elasticsearch performance. To do so, please see [Elasticsearch tuning](#).

Logstash

Logstash is the tool that will collect, parse, and forward to Elasticsearch for indexing and storage all logs generated by Wazuh server. For more info please see [Logstash](#).

1. Install the Logstash package:

```
$ apt-get install logstash=1:5.6.5-1
```

2. Download the Wazuh config for Logstash:

```
$ curl -so /etc/logstash/conf.d/01-wazuh.conf  
https://raw.githubusercontent.com/wazuh/wazuh/2.1/extensions/logstash/01-wazuh.conf
```

3. Download the Wazuh Logstash template:

```
$ curl -so /etc/logstash/wazuh-elasticsearch-template.json  
https://raw.githubusercontent.com/wazuh/wazuh/2.1/extensions/elasticsearch/wazuh-elasticsearch-template.json
```

4. Follow this step only if you are using a single-host architecture:

- Edit `/etc/logstash/conf.d/01-wazuh.conf`, commenting out the entire input section titled "Remote Wazuh Manager - Filebeat input" and uncommenting the entire input section titled "Local Wazuh Manager - JSON file input". This will set up Logstash to read the Wazuh `alerts.json` file directly from the local filesystem rather than expecting Filebeat on a separate server to forward the information in that file to Logstash.
- Because the Logstash user needs to read alerts.json file, please add it to OSSEC group by running:

```
$ usermod -a -G ossec logstash
```

5. Enable and start the Logstash service:

- For Systemd:

```
$ systemctl daemon-reload  
$ systemctl enable logstash.service  
$ systemctl start logstash.service
```

- For SysV Init:

```
$ update-rc.d logstash defaults 95 10  
$ service logstash start
```

Note

If you are running Wazuh server and the Elastic Stack server on separate systems (**distributed architecture**), then it is important to configure encryption between Filebeat and Logstash. To do so, please see [Setting up SSL for Filebeat and Logstash](#).

Kibana

Kibana is a flexible and intuitive web interface for mining and visualizing the events and archives stored in Elasticsearch. More info at [Kibana](#).

1. Install the Kibana package:

```
$ apt-get install kibana=5.6.5
```

2. Install the Wazuh App plugin for Kibana:

```
$ /usr/share/kibana/bin/kibana-plugin install https://packages.wazuh.com/wazuhapp/wazuhapp-2.1.1_5.6.5.zip
```

⚠ Warning

The Kibana plugin installation process may take several minutes. Please wait patiently.

3. **Optional.** Kibana will listen only on the loopback interface (localhost) by default. To set up Kibana to listen on all interfaces, edit the file `/etc/kibana/kibana.yml`. Uncomment the setting `server.host` and change the value to:

```
server.host: "0.0.0.0"
```

❗ Note

It is recommended to set up an Nginx proxy for Kibana in order to use SSL encryption and to enable authentication. Instructions to set the proxy up can be found at [Setting up SSL and authentication for Kibana](#).

4. Enable and start the Kibana service:

- a. For Systemd:

```
$ systemctl daemon-reload  
$ systemctl enable kibana.service  
$ systemctl start kibana.service
```

- b. For SysV Init:

```
$ update-rc.d kibana defaults 95 10  
$ service kibana start
```

5. Disable the Elastic repository:

We recommend to disable the Elasticsearch repository in order to prevent an upgrade to a newer Elastic Stack version due to possible breaking changes with our App, so you should do it as follow:

```
$ sed -i -r '/deb https:\/\/\artifacts.elastic.co\/packages\/5.x\/apt stable main/ s/^(.*)$/#\1/g'  
/etc/apt/sources.list.d/elastic-5.x.list
```

Connecting the Wazuh App with the API

Follow the next guide in order to connect the Wazuh App with the API:

- [Connect the Wazuh App with the API](#)

IAM use cases

AWS Identity and Access Management (IAM) enables you to securely control your users' access to AWS services and resources. Using IAM, you can create and manage AWS users and groups, and use permissions to allow and deny their access to AWS resources.

Following are some use cases for Wazuh rules built in for IAM events.

Create user account

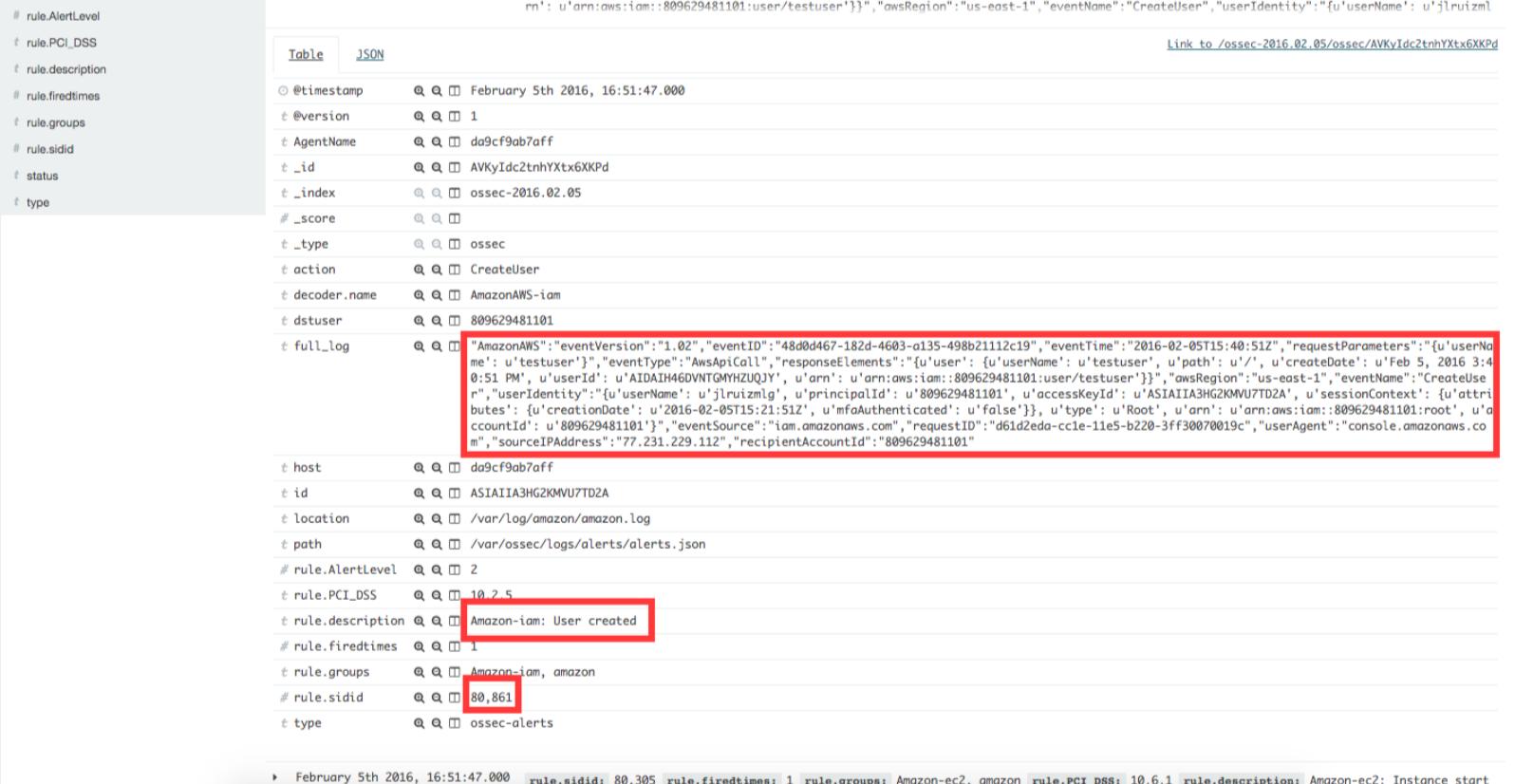
When we create a new user account in IAM, an AWS event is generated. As per the diagram at the beginning of this section, the log event flows to the Wazuh agent which passes it along to the Wazuh manager. The latter then analyzes the event and finds that it matches rule 80861. This results in an alert being generated, which can be seen in Kibana.

Definition of rule 80861

Copied to clipboard 

```
<rule id="80861" level="2">
  <if_sid>80860</if_sid>
  <action>CreateUser</action>
  <description>Amazon-iam: User created</description>
  <group>amazon,pci_dss_10.2.5,</group>
</rule>
```

Kibana will show this alert



rule.AlertLevel	2
rule.PCI_DSS	10.6.1
rule.description	Amazon-iam: User created
rule.firetimes	1
rule.groups	amazon,pci_dss_10.2.5
rule.sidid	80861
rule.type	ossec-alerts
<small>February 5th 2016, 16:51:47.000</small>	
<small>Link to /ossec-2016.02.05/ossec/AVKyIdc2tnhYXtx6XKpd</small>	

Create user account without permissions

If a user without permission to create new users, attempts to create a new user, then the log message generated will match **rule 80862** and Kibana will show the alert as follows:

Definition of rule 80862

Copied to clipboard

```
<rule id="80862" level="2">
  <if_sid>80861</if_sid>
  <match>"errorCode": "AccessDenied"</match>
  <description>Amazon-iam: User creation denied</description>
  <group>amazon,pci_dss_10.2.4,pci_dss_10.2.5,</group>
</rule>
```

Kibana will show this alert

# rule.AlertLevel	rule.PCI_DSS	m::809629481101:user/jlruizm is not authorized to perform: iam:CreateUser on resource: arn:aws:iam::809629481101:user/testuser2","responseElements": "None", "awsRegion": "us-east-1", "eventName": "CreateUser", "userIdentity": "{'userName': 'jlruizm', 'principalId': 'AIDAILRLBKOWLZF6JB550'}
# rule.description		Link to /ossec-2016.02.05/ossec/AVKyLBQKtnhYXtx6XKPn
# rule.firetimes		
t rule.groups		
# rule.sidid		
t status		
t type		
o @timestamp	February 5th 2016, 17:02:58.000	
t @version	1	
t AgentName	da9cf9ab7aff	
t _id	AVKyLBQKtnhYXtx6XKPn	
t _index	ossec-2016.02.05	
# _score		
t _type	ossec	
t action	CreateUser	
t decoder.name	AmazonAWS-iam	
t dstuser	AIDAILRLBKOWLZF6JB550	
t full_log	"AmazonAWS": "eventVersion": "1.02", "errorCode": "AccessDenied", "eventTime": "2016-02-05T15:56:45Z", "requestParameters": "None", "eventType": "AwsApiCall", "errorMessage": "User: arn:aws:iam::809629481101:user/jlruizm is not authorized to perform: iam:CreateUser on resource: arn:aws:iam::809629481101:user/testuser2", "responseElements": "None", "awsRegion": "us-east-1", "eventName": "CreateUser", "userIdentity": "{'userName': 'jlruizm', 'principalId': 'AIDAILRLBKOWLZF6JB550', 'accessKeyId': 'ASIAICYKRXTGWY4SJGA', 'invokedBy': 'signin.amazonaws.com', 'sessionContext': {'attributes': {'creationDate': '2016-02-05T15:56:16Z', 'mfaAuthenticated': 'false'}, 'type': 'IAMUser', 'arn': 'arn:aws:iam:809629481101:user/jlruizm', 'accountId': '809629481101'}}, "eventSource": "iam.amazonaws.com", "requestID": "0e74548d-cc21-11e5-92fb-396f3fb8acdb", "userAgent": "signin.amazonaws.com", "eventID": "a4fc3bcc-010d-48ed-803a-0bd92264308f", "sourceIPAddress": "192.34.63.158", "recipientAccountId": "809629481101"	
t host	da9cf9ab7aff	
t id	ASIAICYKRXTGWY4SJGA	
t location	/var/log/amazon/amazon.log	
t path	/var/ossec/logs/alerts/alerts.json	
# rule.AlertLevel	5	
t rule.PCI_DSS	10.2.4, 10.2.5	
t rule.description	rule.PCI_DSS	Amazon-iam: User creation denied
# rule.firetimes	1	
t rule.groups	Amazon-iam, amazon	
# rule.sidid	80,862	
t type	ossec-alerts	

User login failed

When a user tries to log in with an invalid password, a new event and log message will be generated. This log message will match rule 80802, generating an alert that will be shown in Kibana as follows:

Definition of rule 80802

Copied to clipboard

```
<rule id="80802" level="2">
  <if_sid>80801</if_sid>
  <match>'ConsoleLogin': u'Failure'</match>
  <description>Amazon-signin: User Login failed</description>
  <group>amazon,authentication_failed,pci_dss_10.2.4,pci_dss_10.2.5,</group>
</rule>
```

Kibana will show this alert

t location	⌚ @timestamp	⌚ Q Q ⌂ February 4th 2016, 13:38:04.000
t path	⌚ @version	⌚ Q Q ⌂ 1
# rule.AlertLevel	⌚ AgentName	⌚ Q Q ⌂ da9cf9ab7aff
t rule.PCI_DSS	⌚ _id	⌚ Q Q ⌂ AVKsSiB9NHDLXV6lWlY
t rule.description	⌚ _index	⌚ Q Q ⌂ ossec-2016.02.04
# rule.firetimes	⌚ _score	⌚ Q Q ⌂
# rule.frequency	⌚ _type	⌚ Q Q ⌂ ossec
t rule.groups	⌚ action	⌚ Q Q ⌂ ConsoleLogin
# rule.sidid	⌚ decoder.name	⌚ Q Q ⌂ AmazonAWS-signin
t type	⌚ decoder.parent	⌚ Q Q ⌂ AmazonAWS-signin
	⌚ dstuser	⌚ Q Q ⌂ jlruizm
	⌚ full_log	⌚ Q Q ⌂ "AmazonAWS": "eventVersion": "1.02", "eventId": "b922fe62-7002-49f9-be47-af4a31c4379c", "eventTime": "2016-02-04T12:29:41Z", "additionalEventData": "[{"MFALUsed": "No", "LoginTo": "https://console.aws.amazon.com/console/home?state=hashArgs%23&isAuthcode=true", "MobileVersion": "No"}, {"requestParameters": "None", "eventType": "AwsConsoleSignIn", "errorMessage": "Failed authentication", "responseElements": [{"ConsoleLogin": "Failure"}], "awsRegion": "us-east-1", "eventName": "ConsoleLogin", "userIdentity": {"userName": "jlruizm", "accessKeyId": "AIDAILRLBKOWLZF6JB550", "principalId": "AIDAILRLBKOWLZF6JB550", "accountId": "809629481101"}, "eventSource": "signin.amazonaws.com", "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/48.0.2564.97 Safari/537.36", "sourceIPAddress": "192.34.60.87", "recipientAccountId": "809629481101"}]
	⌚ host	⌚ Q Q ⌂ da9cf9ab7aff
	⌚ location	⌚ Q Q ⌂ /var/log/amazon/amazon.log
	⌚ path	⌚ Q Q ⌂ /var/ossec/logs/alerts/alerts.json
# rule.AlertLevel	⌚ rule.AlertLevel	⌚ Q Q ⌂ 5
t rule.PCI_DSS	⌚ rule.description	⌚ Q Q ⌂ Amazon-signin: User Login failed
# rule.firetimes	⌚ rule.firetimes	⌚ Q Q ⌂ 6
t rule.groups	⌚ rule.groups	⌚ Q Q ⌂ Amazon-iam, amazon, authentication_failed
# rule.sidid	⌚ rule.sidid	⌚ Q Q ⌂ 80,802
t type	⌚ type	⌚ Q Q ⌂ ossec-alerts

Possible break-in attempt

When more than 4 authentication failures occur in a **360** second time window, this fires [rule 80803](#) and generates an alert.

Definition of rule 80803

Copied to clipboard

```
<rule id="80803" level="10" frequency="4" timeframe="360">
  <if_matched_sid>80802</if_matched_sid>
  <description>Possible breakin attempt (high number of login attempts).</description>
  <group>amazon,authentication_failures,pci_dss_11.4,pci_dss_10.2.4,pci_dss_10.2.5,</group>
</rule>
```

Kibana will show this alert

t path	⌚ @version	⌚ Q Q ⌂ 1
# rule.AlertLevel	⌚ AgentName	⌚ Q Q ⌂ da9cf9ab7aff
Quick Count (1 / 1 records)	⌚ _id	⌚ Q Q ⌂ AVKsSiB9NHDLXV6lWlY
10	⌚ _index	⌚ Q Q ⌂ ossec-2016.02.04
	⌚ _score	⌚ Q Q ⌂
	⌚ _type	⌚ Q Q ⌂ ossec
t rule.PCI_DSS	⌚ action	⌚ Q Q ⌂ ConsoleLogin
t rule.description	⌚ decoder.name	⌚ Q Q ⌂ AmazonAWS-signin
# rule.firetimes	⌚ decoder.parent	⌚ Q Q ⌂ AmazonAWS-signin
# rule.frequency	⌚ dstuser	⌚ Q Q ⌂ jlruizm
t rule.groups	⌚ full_log	⌚ Q Q ⌂ "AmazonAWS": "eventVersion": "1.02", "eventId": "df9de854-9835-4939-a9c7-204707e4c8cf", "eventTime": "2016-02-04T12:31:57Z", "additionalEventData": "[{"MFALUsed": "No", "LoginTo": "https://console.aws.amazon.com/console/home?state=hashArgs%23&isAuthcode=true", "MobileVersion": "No"}, {"requestParameters": "None", "eventType": "AwsConsoleSignIn", "errorMessage": "Failed authentication", "responseElements": [{"ConsoleLogin": "Failure"}], "awsRegion": "us-east-1", "eventName": "ConsoleLogin", "userIdentity": {"userName": "jlruizm", "accessKeyId": "AIDAILRLBKOWLZF6JB550", "principalId": "AIDAILRLBKOWLZF6JB550", "accountId": "809629481101"}, "eventSource": "signin.amazonaws.com", "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/48.0.2564.97 Safari/537.36", "sourceIPAddress": "192.34.60.87", "recipientAccountId": "809629481101"}]
	⌚ host	⌚ Q Q ⌂ da9cf9ab7aff
	⌚ location	⌚ Q Q ⌂ /var/log/amazon/amazon.log
	⌚ path	⌚ Q Q ⌂ /var/ossec/logs/alerts/alerts.json
# rule.AlertLevel	⌚ rule.AlertLevel	⌚ Q Q ⌂ 10
t rule.PCI_DSS	⌚ rule.description	⌚ Q Q ⌂ Possible breakin attempt (high number of login attempts).
# rule.firetimes	⌚ rule.firetimes	⌚ Q Q ⌂ 1
# rule.frequency	⌚ rule.frequency	⌚ Q Q ⌂ 4
t rule.groups	⌚ rule.groups	⌚ Q Q ⌂ Amazon-iam, amazon, authentication_failures
# rule.sidid	⌚ rule.sidid	⌚ Q Q ⌂ 80,803
t type	⌚ type	⌚ Q Q ⌂ ossec-alerts

Login success

After a successful login, the [rule 80801](#) will match the log message generated by this event and a new alert will be shown in Kibana:

Definition of rule 80801

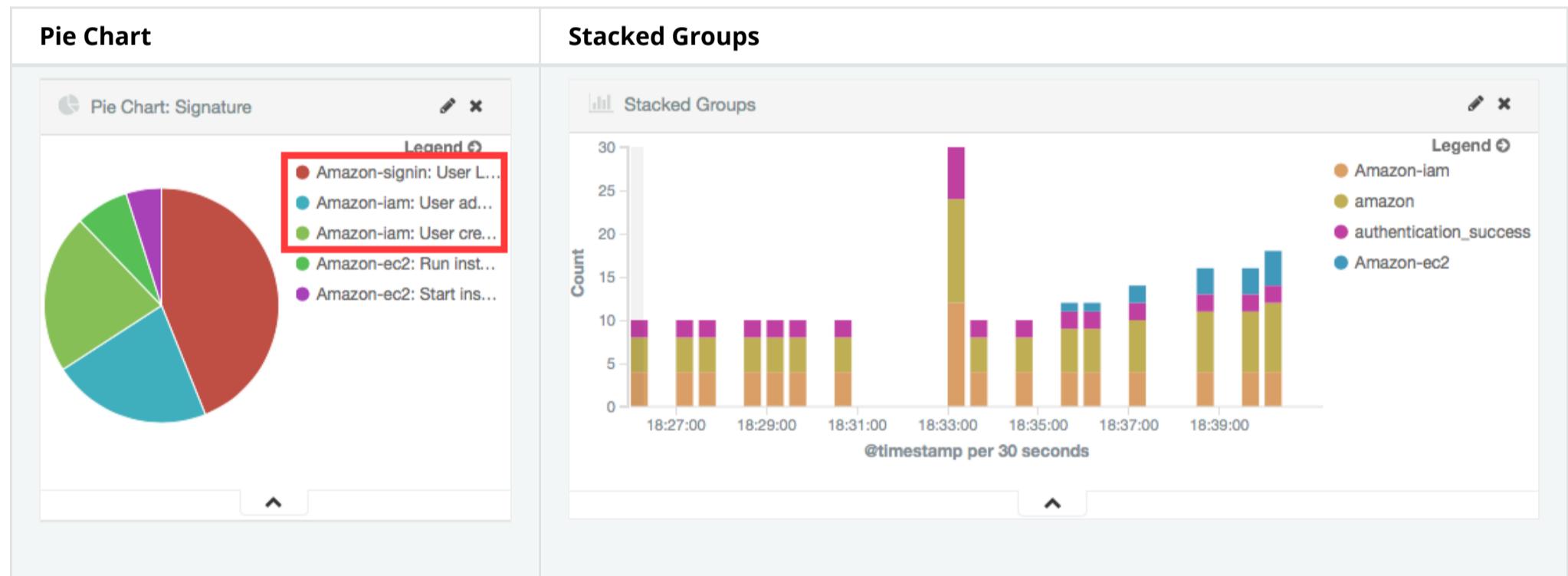
Copied to clipboard

```
<rule id="80801" level="2">
  <if_sid>80800</if_sid>
  <action>ConsoleLogin</action>
  <description>Amazon-signin: User Login Success</description>
  <group>amazon,authentication_success,pci_dss_10.2.5,</group>
</rule>
```

Kibana will show this alert

t location	⌚ @timestamp	⌚ Q Q ⌂ February 4th 2016, 13:48:32.000
t path	⌚ @version	⌚ Q Q ⌂ 1
# rule.AlertLevel	⌚ AgentName	⌚ Q Q ⌂ da9cf9ab7aff
t rule.PCI_DSS	⌚ _id	⌚ Q Q ⌂ AVKsu7gVNH01XV6lWll
# rule.description	⌚ _index	⌚ Q Q ⌂ ossec-2016.02.04
# rule.firetimes	⌚ _score	⌚ Q Q ⌂
# rule.frequency	⌚ _type	⌚ Q Q ⌂ ossec
t rule.groups	⌚ action	⌚ Q Q ⌂ ConsoleLogin
# rule.sidid	⌚ decoder.name	⌚ Q Q ⌂ AmazonAWS-signin
t type	⌚ decoder.parent	⌚ Q Q ⌂ AmazonAWS-signin
	⌚ dstuser	⌚ Q Q ⌂ jlruizm
	⌚ full_log	⌚ Q Q ⌂ "AmazonAWS": "eventVersion": "1.02", "eventId": "a2c63121-5afb-43bf-b8ef-04856b48438c", "eventTime": "2016-02-04T12:41:52Z", "additionalEventData": "[{"MFAUsed": "u'No'", "LoginTo": "u'https://console.aws.amazon.com/console/home?state=hashArgs%23&isauthcode=true'", "MobileVersion": "u'No'", "requestParameters": "None", "eventType": "AwsConsoleSignIn", "responseElements": "[{"ConsoleLogin": "u'Success'}]", "awsRegion": "us-east-1", "eventName": "ConsoleLogin", "userIdentity": "[{"userName": "jlruizm", "type": "u'IAMUser', "arn": "arn:aws:iam:809629481101:user/jlruizm", "principalId": "u'AIDAIRLBKOWLZF6JB550', "accountId": "u'809629481101'", "eventSource": "signin.amazonaws.com", "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_11_4) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/48.0.2564.97 Safari/537.36", "sourceIPAddress": "192.34.60.87", "recipientAccountId": "809629481101"}]"}]
	⌚ host	⌚ Q Q ⌂ da9cf9ab7aff
	⌚ location.host	⌚ Q Q ⌂ /var/log/amazon/amazon.log
	⌚ path	⌚ Q Q ⌂ /var/ossec/logs/alerts/alerts.json
# rule.AlertLevel	⌚ Q Q ⌂ 2	
t rule.PCI_DSS	⌚ Q Q ⌂ 10.2.5	
t rule.description	⌚ Q Q ⌂ Amazon-signin: User Login Success	
# rule.firetimes	⌚ Q Q ⌂ 1	
t rule.groups	⌚ Q Q ⌂ Amazon-iam, amazon, authentication_success	
# rule.sidid	⌚ Q Q ⌂ 80,801	
t type	⌚ Q Q ⌂ ossec-alerts	

The Kibana Dashboards will show:



EC2 use cases

Amazon EC2 (Elastic Compute Cloud) provides scalable computing capacity in the cloud. When using a service like this, it is highly desirable to monitor for attacks or other unauthorized actions being performed against your cloud assets. With CloudTrail and Wazuh's EC2 event analysis capabilities, this is very possible.

Following are some use cases for Wazuh rules built in for EC2.

Run a new instance in EC2

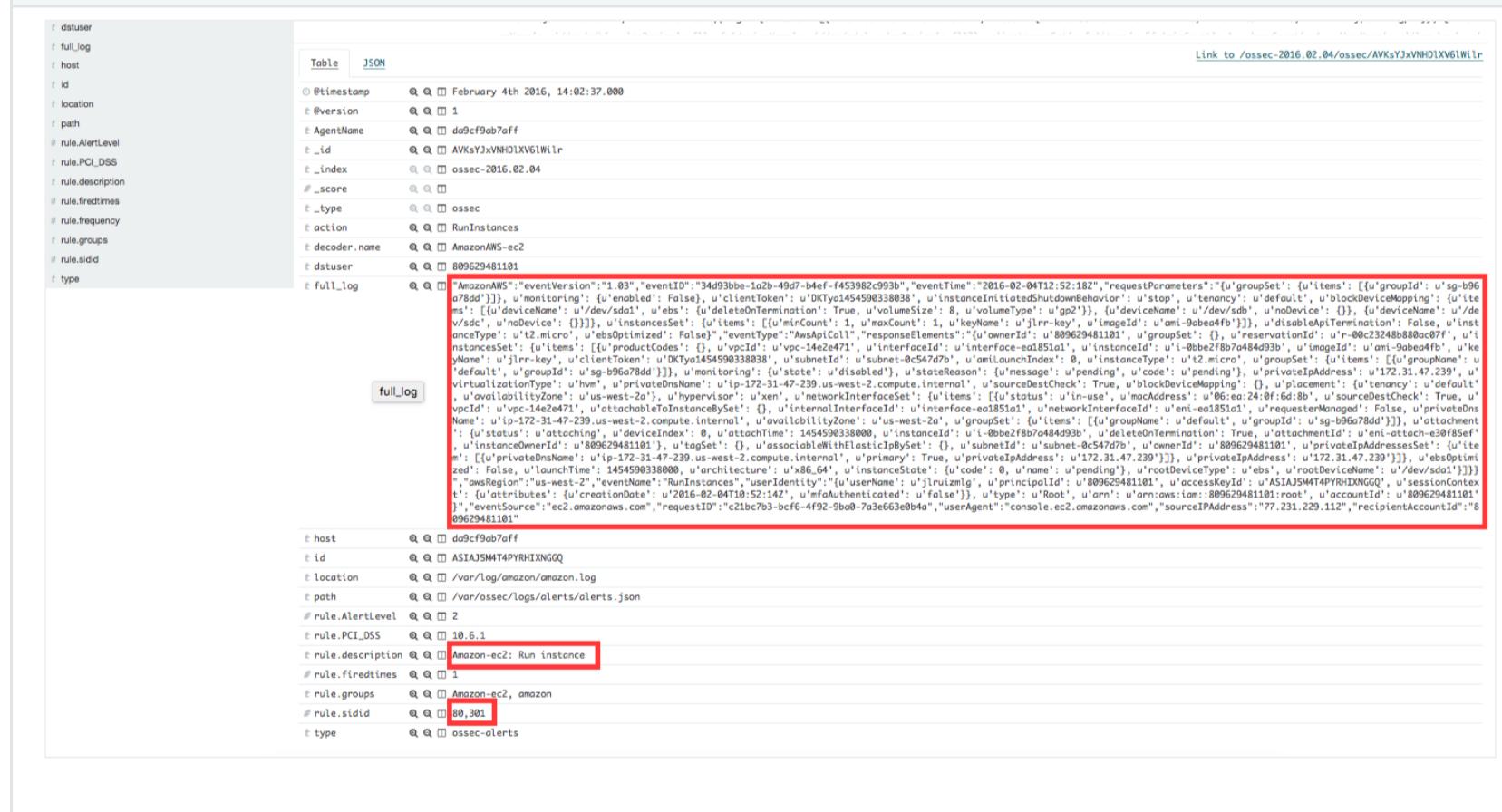
When a user runs a new instance in EC2, an AWS event is generated. As previously illustrated, the log message flows to the Wazuh agent which passes it on to Wazuh manager. The latter analyzes the log event and finds that it matches rule **80301**, which results in an alert being generated, as can be seen in Kibana.

Definition of rule 80301

Copied to clipboard 

```
<rule id="80301" level="2">
  <if_sid>80300</if_sid>
  <action>RunInstances</action>
  <description>Amazon-ec2: Run instance</description>
  <group>amazon,pci_dss_10.6.1,</group>
</rule>
```

Kibana will show this alert



When a user tries to run an instance **without relevant permissions**, then the log message will match **rule 80303** and an alert will be generated as seen below:

Definition of rule 80303

Copied to clipboard 

```
<rule id="80301" level="2">
  <if_sid>80301</if_sid>
  <match>"errorCode": "Client.UnauthorizedOperation"</match>
  <description>Amazon-ec2: Run instance unauthorized</description>
  <group>amazon,pci_dss_10.6.1,</group>
</rule>
```

Kibana will show this alert

Start instances in EC2

When an instance in EC2 is started, the log message will match [rule 80305](#) and an alert will be generated as shown below:

Definition of rule 80305

Copied to clipboard

```
<rule id="80305" level="2">
    <if_sid>80300</if_sid>
    <action>StartInstances</action>
    <description>Amazon-ec2: Instance started</description>
    <group>amazon,pci_dss_10.6.1,</group>
</rule>
```

Kibana will show this alert

Table		JSON
⌚ @timestamp	⌚ ⓘ ⓘ ⓘ	February 8th 2016, 16:14:04.000
⌚ @version	⌚ ⓘ ⓘ ⓘ	1
⌚ AgentName	⌚ ⓘ ⓘ ⓘ	da9cf9ab7aff
⌚ _id	⌚ ⓘ ⓘ ⓘ	AVLBcmY6dJ0hI551weQf
⌚ _index	⌚ ⓘ ⓘ ⓘ	ossec-2016.02.08
# _score	⌚ ⓘ ⓘ ⓘ	
⌚ _type	⌚ ⓘ ⓘ ⓘ	ossec
⌚ action	⌚ ⓘ ⓘ ⓘ	StartInstances
⌚ decoder.name	⌚ ⓘ ⓘ ⓘ	AmazonAWS-ec2
⌚ dstuser	⌚ ⓘ ⓘ ⓘ	809629481101
⌚ full_log	⌚ ⓘ ⓘ ⓘ	<pre>AmazonWS": "eventVersion": "1.03", "eventID": "9db8fc19-12bf-402c-9c11-1574083ea55f", "eventTime": "2016-02-08T15:05:49Z", "requestParameters": {"u'instancesSet': {"u'items': [{"u'instanceId': 'u'i-6fc0e8b6'}]}}, "eventType": "AwsApiCall", "responseElements": {"u'instancesSet': {"u'items': [{"u'instanceId': 'u'i-6fc0e8b6', u'currentState': {"u'code': 0, u'name': 'pending'}, u'previousState': {"u'code': 80, u'name': 'stopped'}}]}}, "awsRegion": "us-west-2", "eventName": "StartInstances", "userIdentity": {"u'userName': 'ujlruizmlo', 'u'principalId': 'u'809629481101', 'u'accessKeyId': 'u'ASIAJV62LU43JDYD7MQQ', 'u'sessionContext': {"u'attributes': {"u'creationDate': 'u'2016-02-08T14:18:09Z', 'u'mfaAuthenticated': 'u>false'}}}, "u'type': 'u'Root', 'u'arn': 'u'arn:aws:iam:809629481101:root', 'u'accountId': 'u'809629481101'", "eventSource": "ec2.amazonaws.com", "requestID": "96c9a3f6-b7ab-4c96-995a-32f686bcc49d", "userAgent": "console.ec2.amazonaws.com", "sourceIPAddress": "188.87.168.226", "recipientAccountId": "809629481101"</pre>
⌚ host	⌚ ⓘ ⓘ ⓘ	da9cf9ab7aff
⌚ id	⌚ ⓘ ⓘ ⓘ	ASIAJV62LU43JDYD7MQQ
⌚ location	⌚ ⓘ ⓘ ⓘ	/var/log/amazon/amazon.log
⌚ path	⌚ ⓘ ⓘ ⓘ	/var/ossec/logs/alerts/alerts.json
# rule.AlertLevel	⌚ ⓘ ⓘ ⓘ	2
⌚ rule.PCI_DSS	⌚ ⓘ ⓘ ⓘ	10.6.1
⌚ rule.description	⌚ ⓘ ⓘ ⓘ	Amazon-ec2: Instance started
# rule.firedtimes	⌚ ⓘ ⓘ ⓘ	1
⌚ rule.groups	⌚ ⓘ ⓘ ⓘ	Amazon-ec2, amazon
# rule.sidid	⌚ ⓘ ⓘ ⓘ	80,305
⌚ type	⌚ ⓘ ⓘ ⓘ	ossec-alerts

If a user tries to start instances **without relevant permissions**, rule 80306 will apply and an alert will be generated as shown below:

Definition of rule 80306

Copied to clipboard

```
<rule id="80306" level="5">
  <if_sid>80305</if_sid>
  <match>"errorCode":"Client.UnauthorizedOperation"</match>
  <description>Amazon-ec2: Start instance unauthorized</description>
  <group>amazon,pci_dss_10.6.1,</group>
</rule>
```

Kibana will show this alert

	host	id	location	path	rule.AlertLevel	rule.PCI_DSS	rule.description	rule.firetimes	rule.groups	rule.sidid	type
Table	JSON										
@timestamp	February 8th 2016, 18:18:33.000										
t @version	1										
t AgentName	da9cf9ab7aff										
t _id	AVLBSFnrJ0hI5S1wee6										
t _index	ossec-2016.02.08										
# _score											
t _type	ossec										
t action	Stop Instances										
t decoder.name	AmazonAWS-ec2										
t dstuser	AIDAIRLBRKOWLZF6JB550										
t full_log	{"AmazonAWS": "eventVersion": "1.03", "errorCode": "Client.UnauthorizedOperation", "eventTime": "2016-02-08T17:12:11Z", "requestParameters": {"u'force': False, u'instancesSet': [{"u'instanceId': 'u'-6fc0e08d6'}]}}, "eventType": "AwsApiCall", "errorMessage": "You are not authorized to perform this operation. Encoded authorization failure message: ipgqIGwNa1AlpTHcpV8zxFoxrj9n24ioNv1M1kpJBL1AQK92RVMlbgkccN-CpDF6IAiHXGoFyVKYy7deGfpCotgiCxZ1vVEbd01MmAbQ1Nf1y7dv0m1U8hv_fyxeD0tsPrtHmpZ_LK5YwX1X17TyMBFgwjjacljebuSd01079A8E17xdo02ge_a96LXCa1j-Elui3hbq09KF57h4glzP8oCry198V0Qx4Rpeel-3wTaGnohcVhZRlV1TWCl0DEt33y07Vj-k3-S19njfMg71ih728exj1In2H2Q2IchFFxLYXZwuARNnQX34s7b24TDKL1pmbyphqfDwLRjGUHTIBDXLgx-RRkwWJ8v6zpqVUKZYALUgzkCX00lkxjvpqUFmCwA7Nmzb1TwobJc4G1U1rGYAsXfYtJGKnhwl_o-NxCl-Jcvb-Zw4U3P8GHTsFzJ7yaf4BVN7UPX3yE_S0xa4AGcFlldzqz5Pun00IP70kZpEKSLv95zeVhKSvG11VZ15040RVdfjnNgD1lpJXItcgahsGABG8-25dPPE1V8bwFP64nbUYlhEKQ1Uw", "responseElements": "None", "awsRegion": "us-west-2", "eventName": "StopInstances", "userIdentity": {"u'username': 'jlruizm', 'u'principalId': 'u'AIDAIRLBRKOWLZF6JB550', 'u'accessKeyId': 'u'ASIAIW4R56USTFGP6ZQ', 'u'invokedBy': 'u'signin.amazonaws.com', 'u'sessionContext': {'u'attributes': {'u'creationDate': 'u'2016-02-08T17:08:41Z', 'u'mfaAuthenticated': 'u'false'}}}, 'u'type': 'u'TAMUser', 'u'arn': 'u'arn:aws:iam::809629481101:user/jlruizm', 'u'accountId': 'u'809629481101'}, "eventSource": "ec2.amazonaws.com", "requestID": "1a975fe4-776b-43c4-932e-0400348a7753", "userAgent": "signin.amazonaws.com", "eventID": "bcdf5721-fdc9-b43d-6dbd64c8eddf", "sourceIPAddress": "192.81.208.190", "recipientAccountId": "809629481101"}										
full_log											
t host	da9cf9ab7aff										
t id	ASIAIW4R56USTFGP6ZQ										
t location	/var/log/amazon/amazon.log										
t path	/var/ossec/logs/alerts/alerts.json										
# rule.AlertLevel	5										
t rule.PCI_DSS	10.6.1										
t rule.description	Amazon-ec2: Stop instance unauthorized										
# rule.firetimes	1										
t rule.groups	Amazon-ec2, amazon										
# rule.sidid	80,309										
t type	ossec-alerts										

Stop instances in EC2

When an instance in EC2 is stopped, [rule 80308](#) will apply and an alert will be generated as shown below:

Definition of rule 80308

Copied to clipboard

```
<rule id="80308" level="2">
  <if_sid>80300</if_sid>
  <action>StopInstances</action>
  <description>Amazon-ec2: Instance stopped</description>
  <group>amazon,pci_dss_10.6.1,</group>
</rule>
```

Kibana will show this alert

Quick Count (1 / 1 records)

StopInstances 100.0%

Visualize

Table JSON

t timestamp	February 8th 2016, 16:36:58.000
t @version	1
t AgentName	da9cf9ab7aff
t _id	AVLBh1oZdJ0hI551weev
t _index	ossec-2016.02.08
# _score	
t _type	ossec
t action	StopInstances
t decoder.name	AmazonAWS-ec2
t dstuser	809629481101
t full_log	<p>AmazonAWS: "eventVersion": "1.03", "eventId": "f102fe8b-d9e-4d00-9334-542c0f25f45f", "eventTime": "2016-02-08T15:29:46Z", "requestParameters": {"u'force': False, u'instancesSet': {u'items': [{u'instanceId': 'u'i-06721f96e0b968896'}]}}, "eventType": "AwsApiCall", "responseElements": {"u'instancesSet': {u'items': [{u'instanceId': 'u'i-06721f96e0b968896', u'currentState': {u'code': 64, u'name': 'u'stopping'}, u'previousState': {u'code': 16, u'name': 'u'running'}}]}}, "awsRegion": "us-west-2", "eventName": "StopInstances", "userIdentity": "u'jruizmla'", "u'username': 'jruizmla', 'u'accountId': '809629481101', "eventSource": "ec2.amazonaws.com", "requestID": "6ffde308-61c3-4008-84d4-45175era9d36", "userAgent": "console.ec2.amazonaws.com", "sourceIPAddress": "188.87.168.226", "recipientAccountId": "809629481101"}</p>
t host	da9cf9ab7aff
t id	ASIAJW62LU43JDYD7MQQ
t location	/var/log/amazon/amazon.log
t path	/var/ossec/logs/alerts/alerts.json
# rule.AlertLevel	2
t rule.PCI_DSS	10.6.1
t rule.description	Amazon-ec2: Instance stopped
# rule.firetimes	1
t rule.groups	Amazon-ec2, amazon
# rule.sidid	80,308
t type	ossec-alerts

If a user tries to stop instances **without relevant permissions**, [rule 80306](#) will apply and an alert will be generated as shown below:

Definition of rule 80309

Copied to clipboard

```
<rule id="80309" level="5">
  <if_sid>80308</if_sid>
  <action>StopInstances</action>
  <match>"errorCode": "Client.UnauthorizedOperation"</match>
  <description>Amazon-ec2: Stop instance unauthorized</description>
  <group>amazon,pci_dss_10.6.1,</group>
</rule>
```

Kibana will show this alert

host
id
location
path
rule.AlertLevel
rule.PCI_DSS
rule.description
rule.firetimes
rule.groups
rule.sidid
type

Table JSON

t timestamp	February 8th 2016, 18:18:33.000
t @version	1
t AgentName	da9cf9ab7aff
t _id	AVLB5FnrdJ0hI551weev
t _index	ossec-2016.02.08
# _score	
t _type	ossec
t action	StartInstances
t decoder.name	AmazonAWS-ec2
t dstuser	AIDAIRLRLBK0WLZF6JB550
t full_log	<p>AmazonAWS: "eventVersion": "1.03", "errorCode": "Client.UnauthorizedOperation", "eventTime": "2016-02-08T17:09:59Z", "requestParameters": {"u'instancesSet': {u'items': [{u'instanceId': 'u'i-06721f96e0b968896'}]}}, "eventType": "AwsApiCall", "errorMessage": "You are not authorized to perform this operation. Encoded authorization failure message: IQQRYYB-3NDX3jw4j4VABw3m3a1s2Eo0v5WWF97ktMPj1t2-XtY-GPS2zherAEBc60Fhtz5YJX1Gw6TuF3BWYdmnUu0L_yFGyUy4S6KX1XoR1b6sQnFwC5y45fzyFlqhBUTq447-17sEtTT0bxKy-z99ZFx2hu4wz-ftBtZ25xz6tC21YUQ0uR1BvZkem31Rv20VrwMvDPfhKs08nW0neMpV4Yx24_0cV2NghA-x2AGQewgzz1XX0YLFB0spiaXv3z0DawNUeM7Y90Tlh-Bjhxrs5JUjYtj9s1hWe0EjghHLF00-TJQVu0Fky13nIegMuHo1nuIFR0YmcnIXXX-1W08e0EYHng50wEF2Ve7Of_GMEe3APV5oYf1TSos02NXTnD14HU088_NB9w7CS576l07PjuCcoSHemil1vcfjmSnD9Z90WYtjnduL00UvM19vf cstNW3W11eJefb16r27WS1W1J1-Gdr5408Ykq0LNQmQjNvYFEnTeVWj15Fys6NSNx82FV9Xkd0-NBuw3r1wga1_WMb0xRnhZNBysjBvFg7smZr1_PCDkw", "responseElements": {"None", "awsRegion": "us-west-2"}, "eventName": "StartInstances", "userIdentity": "u'jruizmla'", "u'principalId': 'AIDAIRLRLBK0WLZF6JB550', "u'accessKeyId': 'u'ASIAINHAR56U5JTGPG6Z0', "u'invokedBy': 'signin.amazonaws.com', "u'sessionContext': {u'attributes': {u'creationDate': 'u'2016-02-08T17:08:41', 'u'mfaAuthenticated': 'u'false'}}}, "u'type': 'u'IAMUser', 'u'arn': 'u'arn:aws:iam:809629481101:user:jruizmla', 'u'accountId': 'u'809629481101', "eventSource": "ec2.amazonaws.com", "requestID": "b2764f63-f34f-4b01-f41-364b2d09edce", "userAgent": "signin.amazonaws.com", "eventId": "c220eb78-990e-4ec2-b882-ceed98c7b846", "sourceIPAddress": "192.81.208.198", "recipientAccountId": "809629481101"}</p>
t host	da9cf9ab7aff
t id	ASIAIW4R56U5TFGP6ZQ
t location	/var/log/amazon/amazon.log
t path	/var/ossec/logs/alerts/alerts.json
# rule.AlertLevel	5
t rule.PCI_DSS	10.6.1
t rule.description	Amazon-ec2: Start instance unauthorized
# rule.firetimes	1
t rule.groups	Amazon-ec2, amazon
# rule.sidid	80,308
t type	ossec-alerts

Create Security Groups in EC2

When a new security group is created, [rule 80404](#) will fire and an alert will be shown as follows:

Definition of rule 80404

Copied to clipboard

```
<rule id="80404" level="2">
  <if_sid>80300</if_sid>
  <action>CreateSecurityGroup</action>
  <description>Amazon-ec2: Create Security Group</description>
  <group>amazon,pci_dss_10.6.1,</group>
</rule>
```

Kibana will show this alert

		securityGroup", "userIdentity": "u'userName': u'jruizmig', u'principalId': u'809629481101', u'accessKeyId': u'ASIAJV62LU43JYD7MQQ', u'sessionContext': fu'a
l _index		Link to /ossec-2016.02.08/ossec/AVLB8AGSdJ0hI551wefB
# _score		
t _type		
t action		
t decoder.name		
t decoder.parent		
t distuser		
t full_log		
t host		
t id		
t location		
t path		
# rule.AlertLevel		
t rule.PCI_DSS		
t rule.description		
t rule.firetimes		
t rule.groups		
t rule.sidid		
t type		
Table		
JSON		
o @timestamp	Q Q □	February 8th 2016, 18:31:16.000
t @version	Q Q □	1
t AgentName	Q Q □	da9cf9ab7aff
t _id	Q Q □	AVLB8AGSdJ0hI551wefB
t _index	Q Q □	ossec-2016.02.08
# _score	Q Q □	
t _type	Q Q □	ossec
t action	Q Q □	CreateSecurityGroup
t decoder.name	Q Q □	AmazonANS-ec2
t distuser	Q Q □	809629481101
t full_log	Q Q □	"AmazonAWS": "eventVersion": "1.03", "eventId": "d3425663-0c5b-441f-9755-8a12b1413c1c", "eventTime": "2016-02-08T17:26:42Z", "requestParameters": {"u'groupName': 'Security Group', 'u'vepcId': 'u'vpc-f9d9e9c', 'u'groupDescription': 'u'Test Security Group'}, "eventType": "AwsApiCall", "responseElements": {"u'_return': True, 'u'groupId': 'u'sg-ddb165ba}', "u'awsRegion': 'us-west-2', "u'eventName': 'CreateSecurityGroup', "u'userIdentity': "u'userName': u'jruizmig', u'principalId': u'809629481101', u'accessKeyId': u'ASIAJV62LU43JYD7MQQ', u'sessionContext': {u'attributes': {u'creationDate': u'2016-02-08T14:18:09Z', 'u'infoAuthenticated': 'u'false'}}, 'u'type': 'u'Root', 'u'arn': 'aws:iam::809629481101:root', 'u'accountId': 'u'809629481101"}, "eventSource": "ec2.amazonaws.com", "requestID": "71e4484b-17c8-48bb-8883-08580b34466b", "userAgent": "console.ec2.amazonaws.com", "sourceIPAddress": "188.87.168.226", "recipientAccountId": "809629481101"
t host	Q Q □	da9cf9ab7aff
t id	Q Q □	ASIAJV62LU43JYD7MQQ
t location	Q Q □	/var/log/amazon/amazon.log
t path	Q Q □	/var/ossec/logs/alerts/alerts.json
# rule.AlertLevel	Q Q □	2
t rule.PCI_DSS	Q Q □	10.6.1
t rule.description	Q Q □	Amazon-ec2: Create Security Group
# rule.firetimes	Q Q □	1
t rule.groups	Q Q □	Amazon-ec2, amazon
# rule.sidid	Q Q □	80,404
t type	Q Q □	ossec-alerts

Allocate a new Elastic IP address

If a new Elastic IP is allocated, then rule 80411 will apply:

Definition of rule 80411

Copied to clipboard

```
<rule id="80411" level="2">
  <if_sid>80300</if_sid>
  <action>AllocateAddress</action>
  <description>Amazon-ec2: Allocate Address</description>
  <group>amazon,</group>
</rule>
```

Kibana will show this alert

	decoder.name	decoder.parent	dstuser	full_log	host	id	location	path	rule.AlertLevel	rule.PCI_DSS	rule.description	rule.firetimes	rule.groups	rule.sidid	type
o @timestamp	February 8th 2016, 18:31:16.000														
t @version	1														
t AgentName	da9cf9ab7aff														
t _id	AVL8AG5dJ0hI5S1wefC														
t _index	ossec-2016.02.08														
# _score															
t _type	ossec														
t action	AllocateAddress														
t decoder.name	AmazonAWS-ec2														
t dstuser	809629481101														
t full_log	AmazonAWS: "eventVersion": "1.03", "eventId": "996adb00-901b-4232-be93-8e265350e786", "eventTime": "2016-02-08T17:26:56Z", "requestParameters": {"u'domain': u'vpc'}}, "eventType": "AwsApiCall", "responseElements": {"u'publicIp': u'52.35.216.133', u'domain': u'vpc', u'AllocationId': 'eipalloc-8c1dfbe8'}}, "awsRegion": "us-west-2", "eventName": "AllocateAddress", "userIdentity": {"u'username': 'jlruizmig', 'principalId': '809629481101', 'accessKeyId': 'ASIAJV62LU43JDYD7MQQ', 'sessionContext': {'u'attributes': {'u'creationDate': '2016-02-08T14:18:09Z', 'u'mfaAuthenticated': 'false'}}}, "u'type': 'Root', 'u'arn': 'arn:aws:iam:809629481101:root', 'u'accountId': '809629481101'}, "eventSource": "ec2.amazonaws.com", "requestID": "888d90b7-7d65-4516-bf58-4d8185cc058", "userAgent": "console.ec2.amazonaws.com", "sourceIPAddress": "188.87.168.226", "recipientAccountId": "809629481101"														
t host	da9cf9ab7aff														
host	ASIAJV62LU43JDYD7MQQ														
t location	/var/log/amazon/amazon.log														
t path	/var/ossec/logs/alerts/alerts.json														
# rule.AlertLevel	2														
t rule.description	Amazon-ec2: Associate Address														
# rule.firetimes	1														
t rule.groups	Amazon-ec2, amazon														
# rule.sidid	80,411														
t type	ossec-alerts														

Associate a new Elastic IP address

If an Elastic IP address is associated, then [rule 80446](#) will apply, generating the corresponding alert:

Definition of rule 80446															
Copied to clipboard															
<rule id="80446" level="2">															
<if_sid>80300</if_sid>															
<action>AssociateAddress</action>															
<description>Amazon-ec2: Associate Address</description>															
<group>amazon,pci_dss_10.6.1,</group>															
</rule>															

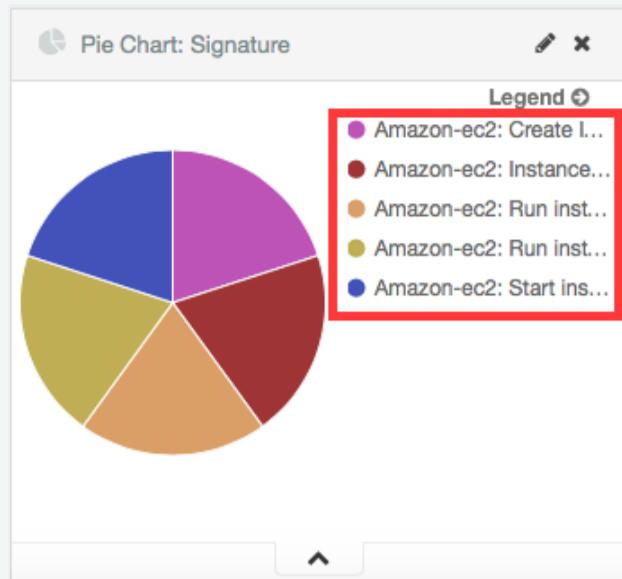
Kibana will show this alert

	full_log	host	id	location	path	rule.AlertLevel	rule.PCI_DSS	rule.description	rule.firetimes	rule.groups	rule.sidid	type		
o @timestamp	February 8th 2016, 18:47:55.000													
t @version	1													
t AgentName	da9cf9ab7aff													
t _id	AVL8AG5dJ0hI5S1wefg													
t _index	ossec-2016.02.08													
# _score														
t _type	ossec													
t action	AssociateAddress													
t decoder.name	AmazonAWS-ec2													
t dstuser	809629481101													
t full_log	AmazonAWS: "eventVersion": "1.03", "eventId": "978a861b-7358-4374-bdc4-d5251e5473cc", "eventTime": "2016-01-07T20:45:46Z", "requestParameters": {"u'networkInterfaceId': 'eni-5d39b10', 'u'allowReassociation': False, 'u'AllocationId': 'eipalloc-63012006', 'u'privateIpAddress': '172.31.36.210"}, "eventType": "AwsApiCall", "responseElements": {"u'_return': True, 'u'associationId': 'eipassoc-43972124"}, "awsRegion": "us-west-2", "eventName": "AssociateAddress", "userIdentity": {"u'username': 'jlruizmig', 'principalId': '809629481101', 'accessKeyId': 'ASIAIURBQE5FTI6Q7VQ', 'sessionContext': {'u'attributes': {'u'creationDate': '2016-01-07T18:46:59Z', 'u'mfaAuthenticated': 'false'}}}, "u'type': 'Root', 'u'arn': 'arn:aws:iam:809629481101:root', 'u'accountId': '809629481101"}, "eventSource": "ec2.amazonaws.com", "requestID": "0cd8e035-2915-4ccb-a00d-84cc054049e", "userAgent": "console.ec2.amazonaws.com", "sourceIPAddress": "76.66.104.185", "recipientAccountId": "809629481101"													
t host	da9cf9ab7aff													
t id	ASIAIURBQE5FTI6Q7VQ													
t location	/var/log/amazon/amazon.log													
t path	/var/ossec/logs/alerts/alerts.json													
# rule.AlertLevel	2													
t rule.description	Amazon-ec2: Associate Address													
# rule.firetimes	1													
t rule.groups	Amazon-ec2, amazon													
# rule.sidid	80,446													
t type	ossec-alerts													

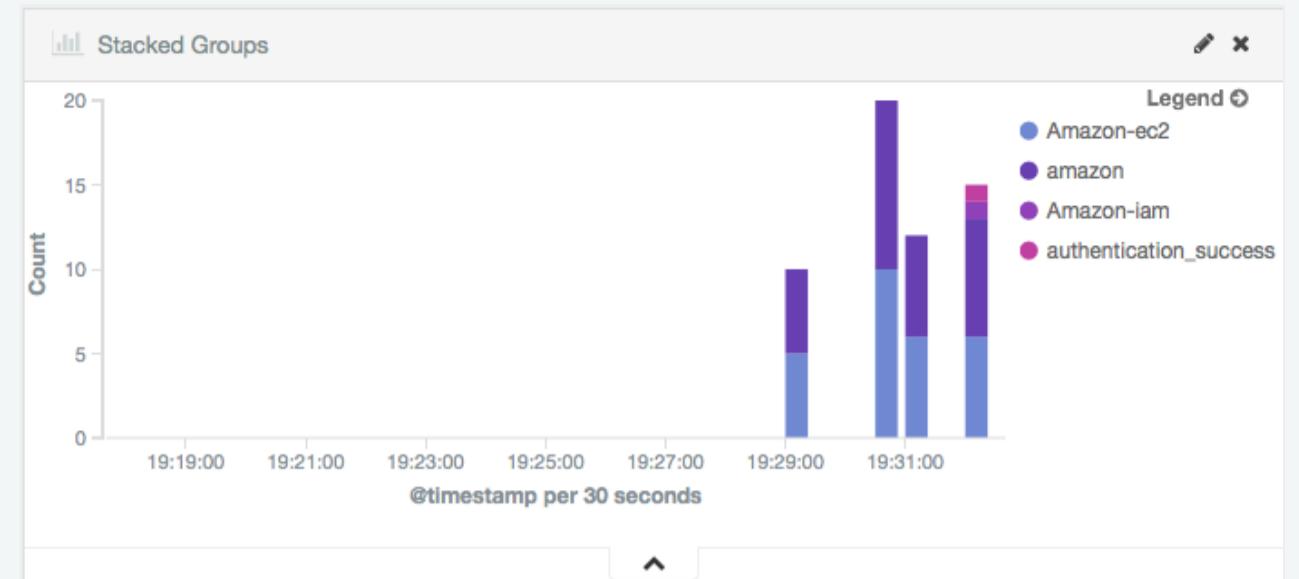
The Kibana Dashboards will show:

Pie Chart	Stacked Groups
-----------	----------------

Pie Chart



Stacked Groups



VPC Use cases

Using an Amazon VPC (Virtual Private Cloud), you can logically isolate your AWS assets from the rest of AWS. You can even set up your own virtual networking in the cloud. It is important to carefully monitor what happens with your VPC as it represent a critical part of your cloud infrastructure.

Create VPC

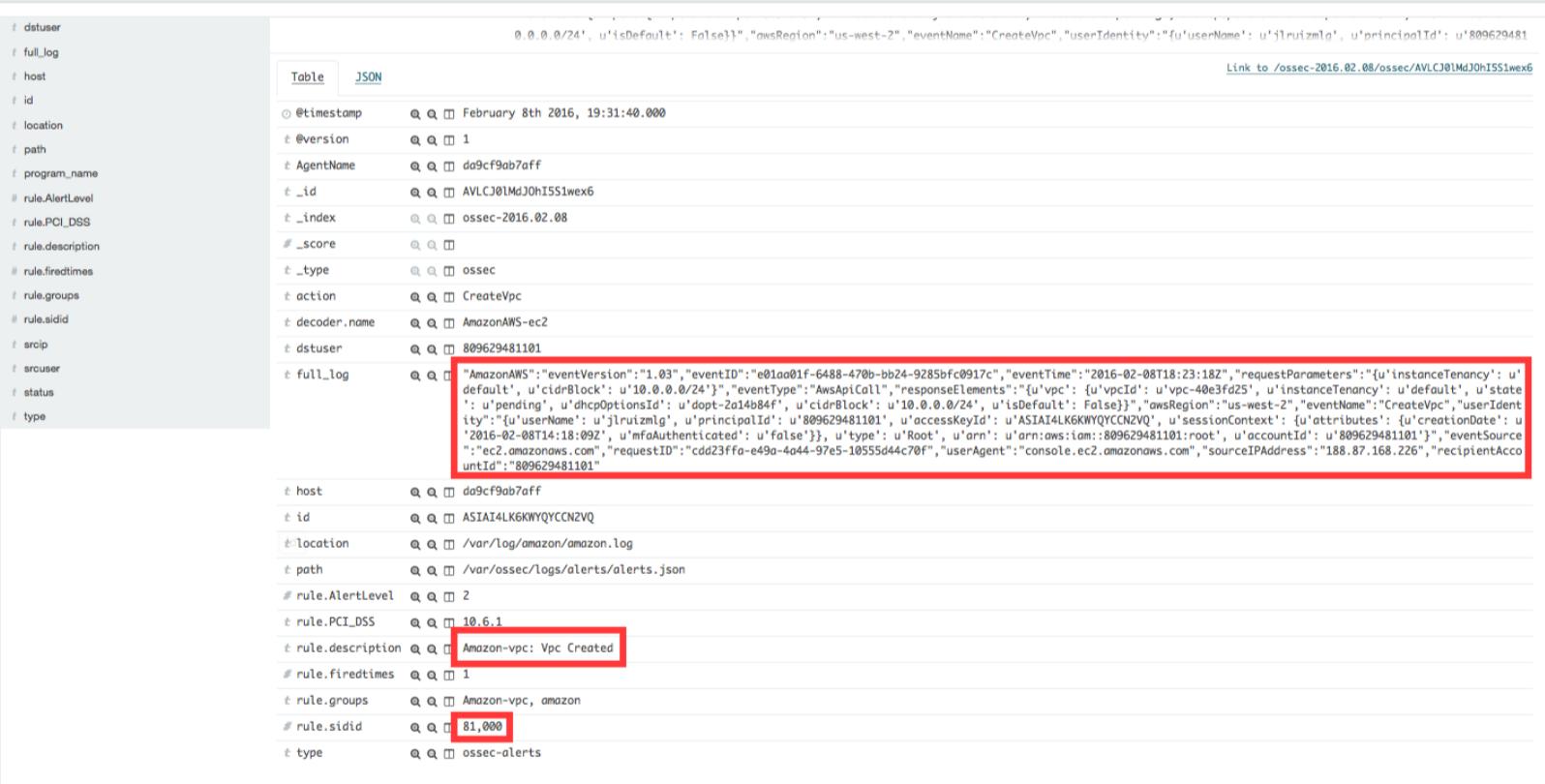
If a VPC is created, [rule 81000](#) will apply and an alert will be generated as shown below:

Definition of rule 81000

Copied to clipboard 

```
<rule id="81000" level="2">
  <if_sid>80300</if_sid>
  <action>CreateVpc</action>
  <description>Amazon-vpc: Vpc Created</description>
  <group>amazon,pci_dss_10.6.1,</group>
</rule>
```

Kibana will show this alert



The screenshot shows a Kibana search results page. On the left, there's a sidebar with various field names like dstuser, full_log, host, id, location, path, program_name, rule.AlertLevel, rule.PCI_DSS, rule.description, rule.firetimes, rule.groups, rule.sidid, srcip, srcuser, status, type. In the main area, there's a table with two rows. The first row is for the event itself, and the second row is for the alert definition. The alert definition row has several fields highlighted with red boxes: 'rule.description' contains 'Amazon-vpc: Vpc Created', 'rule.sidid' contains '81,000', and 'rule.PCI_DSS' contains '10.6.1'. The entire alert definition row is also enclosed in a red box.

If a user without proper permissions attempts to create a VPC, [rule 81001](#) will apply:

Definition of rule 81001

Copied to clipboard 

```
<rule id="81001" level="5">
  <if_sid>81000</if_sid>
  <match>"errorCode": "Client.UnauthorizedOperation"</match>
  <description>Amazon-Vpc: Vpc Created Unauthorized Operation</description>
  <group>amazon,pci_dss_10.6.1,</group>
</rule>
```

Kibana will show this alert

↳ _score	"errorMessage": "You are not authorized to perform this operation.", "responseElements": "None", "awsRegion": "us-west-2", "eventName": "CreateVpc", "userIdentity": "f0'userName': u'jlruizm', u'principalId': u'AIDAILRLBKOWLZF6JB550', u'accessKeyId': u'ASIAJASXIR4RVLYRQMA', u'ses	
↳ _type		
↳ action		
↳ data		
↳ decoder.name		
↳ decoder.parent		
↳ dstuser		
↳ full_log		
↳ host		
↳ id		
↳ location		
↳ path		
↳ program_name		
↳ rule.AlertLevel		
↳ rule.PCI_DSS		
↳ rule.description		
↳ rule.firetimes		
↳ rule.groups		
↳ rule.sidid		
↳ srcip		
↳ srcuser		
↳ status		
↳ type		
↳ @timestamp	February 8th 2016, 19:31:40.000	
↳ @version	1	
↳ AgentName	da9cf9ab7aff	
↳ _id	AVLCJ01MdJ0hIS51wex5	
↳ _index	ossec-2016.02.08	
↳ _score		
↳ _type	ossec	
↳ action	CreateVpc	
↳ decoder.name	AmazonAWS-ec2	
↳ dstuser	AIDAILRLBKOWLZF6JB550	
↳ full_log	AmazonAWS: "eventVersion": "1.03", "errorCode": "Client.UnauthorizedOperation", "eventTime": "2016-02-08T18:24:17Z", "requestParameters": {"u'instanceTenancy': 'default', "u'cidrBlock': '10.0.0.0/24"}", "eventType": "AwsApiCall", "errorMessage": "You are not authorized to perform this operation.", "responseElements": "None", "awsRegion": "us-west-2", "eventName": "CreateVpc", "userIdentity": "jlruizm", "principalId": "AIDAILRLBKOWLZF6JB550", "accessKeyId": "ASIAJASXIR4RVLYRQMA", "invokedBy": "signin.amazonaws.com", "sessionContext": {"u'attributes': {"u'creationDate': '2016-02-08T17:08:41Z', "u'mfaAuthenticated': 'false'}}, "u'type': 'IAMUser', "u'arn': 'arn:aws:iam::809629481101:user/jlruizm', "u'accountId': '809629481101'", "eventSource": "ec2.amazonaws.com", "requestID": "59d526fd-577b-46f6-8d81-c2d3ced01319", "userAgent": "signin.amazonaws.com", "eventID": "e3990d52-43a7-4937-oc1a-02f928920935", "sourceIPAddress": "192.81.214.229", "recipientAccountId": "809629481101"	
↳ host	da9cf9ab7aff	
↳ id	ASIAJASXIR4RVLYRQMA	
↳ location	/var/log/amazon/amazon.log	
↳ path	/var/ossec/logs/alerts/alerts.json	
↳ rule.AlertLevel	5	
↳ rule.PCI_DSS	10.6.1	
↳ rule.description	Amazon-Vpc: Vpc Created Unauthorized Operation	
↳ rule.firetimes	1	
↳ rule.groups	Amazon-vpc, amazon	
↳ rule.sidid	81,001	
↳ type	ossec-alerts	

Scan paths configuration

Leaving this unconfigured will result in Wazuh using the module defaults.

By default, it will monitor `/etc`, `/usr/bin`, `/usr/sbin`, `/bin` and `/sbin` on the Wazuh Server, with real time monitoring disabled and `report_changes` enabled.

To overwrite the defaults or add in new paths to scan, you can use here to overwrite the defaults.

To tell Wazuh to enable real time monitoring of the default paths:

wazuh::server::ossec_scanpaths:

```
path: /etc report_changes: 'no' realtime: 'no'  
path: /usr/bin report_changes: 'no' realtime: 'no'  
path: /usr/sbin report_changes: 'no' realtime: 'no'  
path: /bin report_changes: 'yes' realtime: 'yes'  
path: /sbin report_changes: 'yes' realtime: 'yes'
```

wazuh::server::ossec_ignorepaths:

By default, it will empty.

To overwrite the defaults or add in new paths to scan, you can use here to overwrite the defaults.

More information in about syscheck configuration in the [File integrity monitoring](#) section.

Note

Configuring the `ossec_scanpaths` variable will overwrite the default paths. If you want to add a new directory to monitor, you must also add the above default paths to be monitored as well.

Wazuh agent class

\$ossec_active_response

Allows active response on this host.

Default true

\$ossec_rootcheck

Enable rootcheck on this host.

Default true

\$ossec_rootcheck_frequency

Setup rootcheck frequency.

Default 36000

\$ossec_rootcheck_checkports

Enable rootcheck checkports option.

Default true

\$ossec_rootcheck_checkfiles

Enable rootcheck checkfiles option.

Default true

\$ossec_server_ip

Wazuh Manager IP.

\$ossec_server_hostname

Hostname of the server.

\$ossec_server_port

Server port configuration.

Default 1514

\$ossec_scanpaths

Agents can be Linux or Windows. For this reason, don't have ossec_scanpaths by default.

Default []

\$ossec_emailnotification

Whether to send email notifications or not.

Default yes

\$ossec_ignorepaths

Files/Directory list to ignore

Default []

\$ossec_local_files

Files list for log analysis

This files are listed in params.pp in section \$default_local_files

\$ossec_syscheck_frequency

Frequency that syscheck is executed default every 12 hours

Default 43200

\$ossec_prefilter

Command used to prevent prelinking from creating false positives.

This option can potentially impact performance negatively. The configured command will be run for each and every file checked.

Default false

\$ossec_service_provider

This option associate Operative System Family

\$ossec_config_profiles

Specify the agent.conf profile(s) to be used by the agent.

Default []

\$selinux

Whether to install a SELinux policy to allow rotation of OSSEC logs.

Default false

\$agent_name

Configure agent host name of the system

Default \$::hostname

\$agent_ip_address

Configure agent IPv4 address for the default network interface.

Default \$::ipaddress

\$manage_repo

Install Wazuh through Wazuh repositories.

Default true

\$manage_epel_repo

Install epel repo and inotify-tools

Default true

\$agent_package_name

Define package name defined in params.pp

\$agent_package_version

Define package version

Default installed

\$agent_service_name

Define service name defined in params.pps

\$manage_client_keys

Manage client keys option.

Default export

\$agent_auth_password

Define password for agent-auth

Default undef

\$ar_repeated_offenders

A comma separated list of increasing timeouts in minutes for repeat offenders.

There can be a maximum of 5 entries.

Default empty

\$enable_wodle_openscap

Enable openscap configuration in ossec.conf

Default false

\$wodle_openscap_content

Depending linux distribution assign profile xccdf.

\$ossec_conf_template

Path of ossec configuration agent template.

Default wazuh/wazuh_agent.conf.erb

function wazuh::addlog

\$log_name

Configure Wazuh log name

\$agent_log

Path to log file.

Default false

\$logfile

Path to log file.

\$logtype

The OSSEC log_format of the file.

Default syslog

Wazuh server class

`class wazuh::server`

\$smtp_server

SMTP mail server.

\$ossec_emailto

Email to address. `['user1@mycompany.com', 'user2@mycompany.com']`

\$ossec_emailfrom

Email from address.

Default ossec@\${domain}

\$ossec_active_response

Enable or disable active-response.

Default true

\$ossec_rootcheck

Enable rootcheck.

Default true

\$ossec_rootcheck_frequency

Frequency that the rootcheck is going to be executed (in seconds).

Default 36000

\$ossec_rootcheck_checkports

Look for the presence of hidden ports.

Default true

\$ossec_rootcheck_checkfiles

Scan the whole filesystem looking for unusual files and permission problems.

Default true

\$ossec_global_host_information_level

Alerting level for the events generated by the host change monitor (from 0 to 16).

Default 8

\$ossec_global_stat_level

Alerting level for the events generated by the statistical analysis (from 0 to 16).

Default 8

\$ossec_email_alert_level

Threshold defining minimum severity for a rule to fire an email alert. Some rules circumvent this threshold (`alert_email` option).

Default 7

\$ossec_ignorepaths

Specify paths to ignore ossec scan

Default []

\$ossec_scanpaths

Define paths to ossec scan

\$ossec_white_list

Allow white listing of IP addresses.

Default []

\$ossec_extra_rules_config

Using it, after enabling the Wazuh ruleset (either manually or via the automated script), take a look at the changes made to the ossec.conf file. You will need to put these same changes into the “\$ossec_extra_rules_config” array parameter when calling the wazuh::server class.

Default []

\$ossec_local_files

Define path log files to scan with ossec

\$ossec_emailnotification

Whether or not to send email notifications.

Default yes

\$ossec_email_maxperhour

Global Configuration with maximum number of emails per hour.

Default 12

\$ossec_email_idsname

Define email ID name

Default undef

\$ossec_syscheck_frequency

Frequency that syscheck is executed default every 22 hours

Default 79200

\$ossec_auto_ignore

Specifies if syscheck will ignore files that change too often (after the third change)

Default yes

\$ossec_prefilter

Command to run to prevent prelinking from creating false positives.

Note

This option can potentially impact performance negatively. The configured command will be run for each and every file checked.

Default false

\$ossec_service_provider

Set service provider to Redhat on Redhat systems.

Default \$::ossec::params::ossec_service_provide

\$ossec_server_port

Port to allow communication between manager and agents.

Default: '1514'

\$server_package_version

Modified client.pp and server.pp to accept package versions as a parameter.

Default installed

\$manage_repos

Install Wazuh through Wazuh repositories.

Default true

\$manage_epel_repo

Install epel repo and inotify-tools

Default true

\$manage_client_keys

Manage client keys option.

Default true

\$agent_auth_password

Define password for agent-auth

Default undef

\$ar_repeated_offenders

A comma separated list of increasing timeouts in minutes for repeat offenders.

There can be a maximum of 5 entries.

Default empty

\$syslog_output

Allows a Wazuh manager to send the OSSEC alerts to one or more syslog servers

Default false

\$syslog_output_server

The IP Address of the syslog server.

Default undef

\$syslog_output_format

Format of alert output.

Default undef

\$enable_wodle_openscap

Enable openscap configuration in ossec.conf

Default false

\$local_decoder_template

Allow to use a custom local_decoder.xml in the manager.

Default wazuh/local_decoder.xml.erb

\$local_rules_template

Allow to use a custom local_rules.xml in the manager.

Default wazuh/local_rules.xml.erb

\$shared_agent_template

Enable the configuration to deploy through agent.conf

Default `wazuh/ossec_shared_agent.conf.erb`

\$manage_paths

Follow the instructions on [ossec-scanpaths](#).

Default [{path' => '/etc,/usr/bin,/usr/sbin', 'report_changes' => 'no', 'realtime' => 'no'}, {path' => '/bin,/sbin', 'report_changes' => 'yes', 'realtime' => 'yes'}]

! Note

Consequently, if you add or remove any of the Wazuh rules later on, you'll need to ensure you add/remove the appropriate bits in the \$ossec_extra_rules_config array parameter as well.

```
function wazuh::email_alert
```

\$alert_email

Email to send to.

\$alert_group

An array of rule group names.

Default false

! Note

No email will be sent for alerts with a severity below the global `[$ossec_email_alert_level]`, unless the rule has alert_email set.

function wazuh::command

\$command_name

Human readable name for wazuh::activeresponse usage.

\$command_executable

Name of the executable. OSSEC comes preloaded with disable-account.sh, host-deny.sh, ipfw.sh, pf.sh, route-null.sh, firewall-drop.sh, ipfw_mac.sh, ossec-tweeter.sh, restart-ossec.sh.

\$command_expect

Default srcip

\$timeout_allowed

Default true

function wazuh::activeresponse

\$command_name

Human readable name for wazuh::activeresponse usage.

\$ar_location

It can be set to local, server, defined-agent, all.

Default local

\$ar_level

Can take values between 0 and 16.

Default 7

\$ar_rules_id

List of rule IDs.

Default []

\$ar_timeout

Usually active response blocks for a certain amount of time.

Default 300

\$ar_repeated_offenders

A comma separated list of increasing timeouts in minutes for repeat offenders. There can be a maximum of 5 entries.

Default empty

function wazuh::addlog

\$log_name

Configure Wazuh log name

\$agent_log

Path to log file.

Default false

\$logfile

Path to log file.

\$logtype

The OSSEC log_format of the file.

Default syslog

Overview

Wazuh is an open source project that provides security visibility, compliance and infrastructure monitoring capabilities. The project was born as a fork of OSSEC HIDS, and has evolved into a comprehensive solution by implementing new functionalities and integrating other tools like OpenSCAP and Elasticsearch.

This manual describes how to configure and use each one of Wazuh components: Wazuh server, Wazuh agent, and Elastic Stack.

Wazuh server

The Wazuh server is based on a suite of applications, every application/component is designed to accomplish certain task, but when they working together it could analyze the data receive from agents, triggering alerts when an event matches a rule, register new clients/agents, sending data to Elastic Stack server, all paired with a RESTful API.

Components

- **Wazuh Manager:** It receives data and analyzes data from the agents. To do that it uses decoders and rules that have been crafted to trigger security alerts. The manager is also used to distribute configuration files to the agents, and to monitor their status. In addition it can send control messages to trigger automatic actions at an agent level.
- **Registration Service:** It use a secure mechanisms to register a client without any intervention on server side.
- **RESTful API:** It provides an interface to manage and monitor the configuration of the manager and agents, also it could register clients/agents. It can be also used to inspect the manager log messages, decoders and rules. In addition it provides useful information related to the agents, including their status, operating system details, and file integrity monitoring and rootcheck alerts.
- **Filebeat:** It is used in distributed architectures (where the Wazuh server and Elastic Stack live in different systems) to forward the alerts data to Logstash. This component has its own [documentation](#) developed by Elastic.

Elastic Stack

Elastic Stack is used to indexing, browse and visualize Wazuh alerts data. In addition, the Wazuh app for Kibana can be used to visualize configuration settings, rules, and decoders, agents status, information, and provides dashboards for policy, compliance and file integrity monitoring.

Components

- **Wazuh app:** is a Kibana plugin designed to display Wazuh related information providing a RESTful API web interface making administration of Wazuh Manager and Wazuh Agents easy and powerful.
- **Logstash:** is used to ingest data coming from one or more Wazuh servers via Filebeat, feeding the Elasticsearch cluster. In addition it enriches alerts adding Geolocation metadata. More information at [Logstash official documentation](#).
- **Elasticsearch:** is a highly scalable full-text search and analytics engine. It is used to index alerts data, and historical agents statuts information. More information at [Elasticsearch official documentation](#).
- **Kibana:** is a flexible and intuitive web interface for mining, analyzing, and visualizing data. In combination with our Wazuh Kibana app, it is used as Wazuh web user interface (WUI). More information at [Kibana official documentation](#).

Wazuh agents

The Wazuh agent runs on monitored systems and is in charge of collecting log and event data, performing policy monitoring scans, detecting malware and rootkits and alert when watched files are modified. It communicates with the Wazuh server through an encrypted and authenticated channel.

Components

- **Rootcheck:** perform rootkit and malware detection on every system where the agent is installed.
- **Log monitoring/analysis:** collect and analyze system logs on the system finding any suspicious activity.
- **Syscheck:** runs periodically to check if any configured file (or registry entry on Windows) has changed.

- **OpenSCAP**: designed to check weak and vulnerable applications and configurations.

Wazuh server administration

Wazuh manager is the system that analyzes the data received from all the agents, triggering alerts when an event matches a rule for example: intrusion detected, file changed, configuration not compliant with policy, possible rootkit, etc. It is also an agent, so it has all the features that an agent has. Also, the manager can forward the alerts it triggered through syslog, emails or integration with external APIs.

Contents

- [Remote service](#)
 - [Configuration](#)
- [Defining an alert level threshold](#)
 - [Configuration](#)
- [Integration with external APIs](#)
 - [Configuration](#)
 - [Integration with Slack](#)
 - [Integration with PagerDuty](#)
- [Configuring syslog output](#)
 - [Configuration](#)
- [Generating automatic reports](#)
- [Configuring email alerts](#)
 - [Generic email options](#)
 - [Granular email options](#)
 - [Force forwarding an alert by email](#)

Registering agents

Next, we describe how this process work and more specifically the different methods you can use to register agents against Wazuh server.

Contents

- [The registration process](#)
 - [Agent keys](#)
 - [Registration methods](#)
- [Using the registration service](#)
 - [Simple method](#)
 - [Secure methods](#)
 - [Forcing insertion](#)

Capabilities

This section explain more in detail how each capability works, how can be configured, frequently asked questions and some examples that allow a better understanding of all the features. If you find a problem, error or if you want to ask related questions please let us know on our [mailing list](#).

The capabilities are:

- [Log data collection](#)
 - [How it works](#)
 - [Configuration](#)
 - [FAQ](#)
- [File integrity monitoring](#)
 - [How it works](#)
 - [Configuration](#)
 - [FAQ](#)
- [Anomaly and malware detection](#)
 - [How it works](#)
 - [Configuration](#)
 - [FAQ](#)
- [Monitoring security policies](#)
 - [Rootcheck](#)
 - [OpenSCAP](#)
- [Monitoring system calls](#)
 - [How it works](#)
 - [Configuration](#)
- [Command monitoring](#)
 - [How it works](#)
 - [Configuration](#)
 - [FAQ](#)
- [Active response](#)
 - [How it works](#)
 - [Configuration](#)
 - [FAQ](#)
- [Agentless monitoring](#)
 - [How it works](#)
 - [Configuration](#)
 - [FAQ](#)
- [Anti-flooding mechanism](#)
 - [Why it is an anti-flooding mechanism needed?](#)
 - [How it works: Leaky bucket](#)
 - [Use case: Leaky bucket](#)
 - [Anti-flooding in agent modules](#)
- [Agent labels](#)
 - [How it works](#)
 - [Use case](#)

Ruleset

This documentation explains how to install, update, and contribute to Wazuh Ruleset. These rules are used by the system to detect attacks, intrusions, software misuse, configuration problems, application errors, malware, rootkits, system anomalies or security policy violations. OSSEC provides an out-of-the-box set of rules that we update and augment, in order to increase Wazuh detection capabilities.

Contents

- [Getting started](#)
 - [GitHub repository](#)
 - [Directory layout](#)
- [Update ruleset](#)
 - [Usage examples](#)
 - [Configure weekly updates](#)
- [Custom rules and decoders](#)
 - [Adding new decoders and rules](#)
 - [Changing an existing rule](#)
 - [Changing an existing decoder](#)
- [Dynamic fields](#)
 - [Traditional decoders](#)
 - [Dynamic decoders](#)
- [Ruleset XML syntax](#)
 - [Decoders Syntax](#)
 - [Rules Syntax](#)
 - [Regular Expression Syntax](#)
- [Testing decoders and rules](#)
- [Using CDB lists](#)
 - [Creating a CDB list](#)
 - [Using the CDB list in the rules](#)
- [Contribute to the ruleset](#)

RESTful API

The Wazuh API is an open source RESTful API that allows the Wazuh manager to be interacted with from a web browser, command line tool like cURL, or any script or program that can make web requests. The Wazuh Kibana app relies on this heavily, and Wazuh's goal is to accommodate complete remote management of the Wazuh infrastructure via the Wazuh Kibana app. Use the API to easily perform everyday actions like adding an agent, restarting the manager/agent(s), or looking up syscheck details.

For more details, see [Use Cases](#).

Wazuh API capabilities:

- Agent management
- Manager control & overview
- Rootcheck control & search
- Syscheck control & search
- Ruleset information
- Statistical information
- HTTPS and user authentication
- Error handling

Contents

- [Getting started](#)
 - [Starting and stopping the API](#)
 - [Hello world!](#)
 - [Basic concepts](#)
 - [Use cases](#)
- [Configuration](#)
 - [Configuration script](#)
 - [Configuration file](#)
 - [Basic Authentication](#)
 - [Manually enable https support](#)
- [Reference](#)
 - [Request List](#)
 - [Agents](#)
 - [Decoders](#)
 - [Manager](#)
 - [Rootcheck](#)
 - [Rules](#)
 - [Syscheck](#)
- [Examples](#)
 - [cURL](#)
 - [Python](#)
 - [PowerShell](#)

Reference

This part of the user manual will cover the configuration files used by Wazuh and define the setting options. Directions are given on how to customize the functionality of Wazuh to the unique environment and specific requirements of each installation.

Contents

- [Local configuration](#)
- [Centralized configuration](#)
- [Internal configuration](#)
- [Daemons](#)
- [Tools](#)

Client keys file

The `client.keys` file stores the data used to authenticate secure agents.

Location [🔗](#)

UNIX systems

Folder `etc` inside the installation directory.

Windows agents

Installation directory.

File format

This file contains one line per each agent entry. In the case of agents, only one line is allowed, and this line must match exactly one entry in the `client.keys` file at manager, otherwise the agent will be rejected.

```
<ID> <Name> <Address> <Password>
```

ID

Agent identifier number.

Allowed characters	Digits only
Allowed size	3 to 8 digits
Padding	0-padded
Unique value	Yes
Reserved values	ID "000"

Name

Name of the agent.

Allowed characters	Alphanumeric characters, <code>-</code> , <code>_</code> and <code>.</code>
Allowed size	Up to 128 bytes
Unique value	Yes

Address

Allowed source address range in CIDR format. If specified, the manager will only accept the agent if its source IP matches this address.

Format	CIDR. Netmask is optional.
Unique value	Yes
Reserved values	None
Aliases	<code>any</code> = <code>0.0.0.0/0</code>

Password

String that will take part in the external message encryption.

Allowed characters	Printable characters
Allowed size	Up to 128 bytes
Unique value	No

Void entries

Key entries can be invalidated so the related agent is considered removed: the line is discarded.

- Line starting with `#` or whitespace.
- Agent name starting with `#` or `!`.

Examples

```
001 server1 any bb8a28997c6c3964eacb3d32308072f6661f567a41105b2b0b09f1a82331b937
002 dbserver 10.0.1.2 363a99a6e9c9a8b6bb766d676453538e0cb20162f84b36472d99cfbef4928440
003 data2 10.1.2.0/24 3d263f5cc513072fe6b63ab221d1facf132918235c97f19efd9446257d16ea4a
004 !data3 any ed52060a133343dbc74474c19aad8fb7dddd9a4b5965ebbe9edb2a73fd11a17
```

Standard OSSEC message format

This page aims to describe the format of messages that OSSEC (and Wazuh) accepts and sends between its components:

Input logs

Strings ingested by *Logcollector* from log files, Windows events, program outputs, etc.

Standard OSSEC events

Data sent **locally** between OSSEC components, usually to *Agent daemon* (in agents) or *Analysis daemon* (in manager).

Secure messages

Data delivered **remotely** between *Agent daemon* and *Remote daemon*.

Input logs

Input logs are messages ingested by Logcollector. They can be in Syslog format or any other custom format. In the former case the message header is parsed by the pre-decoder.

Syslog message

```
Nov 9 16:06:26 localhost salute: Hello world.
```

These logs go through a pre-decoding stage that tries to extract some data from the logs, in the case they are in Syslog format.

- Date: `Nov 9 16:06:26`
- Host name: `localhost`
- Program name: `salute`
- Log: `Hello world.`

If the message is not in Syslog format, the log will be the full text.

The string in *log* is the input for decoders defined in XML format or plugin decoders.

Standard OSSEC event

OSSEC events are transmitted between software components of the same machine, using the local datagram socket at `/var/ossec/queue/ossec/queue`. For instance:

- Log events from *Logcollector* to *Agent daemon*.
- File integrity monitoring events from *Syscheck* to *Agent daemon*.
- Policy monitoring events from *Rootcheck* to *Agent daemon*.
- All of the previous events from *Remote daemon* to *Analysis daemon*.

The format of these events is:

```
<Queue>:<Location>:<Message>
```

Queue

1-byte event type. It defines the decoding mode for *Analysis daemon*.

The most common queue types are:

1

Local file log, including Syslog messages, Windows event logs, outputs from commands, OpenSCAP results and custom logs.

2

Remote Syslog messages, received by the Syslog server at *Remote daemon*.

4

Secure messages. They are events from *Remote daemon* to *Analysis daemon*, that contain a standard OSSEC message plus the source agent ID.

8

Syscheck event. *Analysis daemon* parses it using the Syscheck decoder.

9

Rootcheck event. *Analysis daemon* parses it using the Rootcheck decoder.

Location

Log source, typically the path to the file where the log was found.

Message

Content of the log.

Example:

```
1:/var/log/syslog:Nov 9 16:06:26 localhost salute: Hello world.
```

Secure message format

Secure messages are those messages sent through a network between an agent (*Agent daemon*) and the manager (*Analysis daemon*). They are:

- Encrypted.
- Compressed.
- Randomized.
- Numerated.

Step by step procedure

Block

The *block* is the result of joining a header and the input event:

```
<Block> = <Random> <Global counter> ":" <Local counter> <Event>
```

Random

5-byte 0-padded random unsigned integer.

Size	5 digits
Padding	0-padded

Global counter

Most significative part of the message counter.

Size	10 digits
Padding	0-padded

Local counter

Least significative part of the message counter.

Size	4 digits
Padding	0-padded

Event

Input message.

Hash

The *hash* is the 32-byte MD5 digest:

```
<Hash> = MD5(<Block>)
```

Compressed data

This object is the result of compressing the *hash* and the *block* (appended) through the *DEFLATE* algorithm, using *zlib*:

```
<CData> = Compress(<Hash> <Block>)
```

Padding

The compressed data is a byte array that must:

1. Have a size multiple of 8.
2. Start with one or more `!`.

So the `<Padding>` object is a string of 1 to 8 `!` symbols, so that the array resulting of appending both `<Padding>` and `<CData>` has a size multiple of 8.

```
<Padding> = 1..8 "!"  
Length(<Padding> <Block>) = 0 (mod 8)
```

Encrypted data

The padded data is encrypted using Blowfish:

```
<Encrypted> = Blowfish(<Padding> <Block>)
```

The initialization vector and the encryption key are described in [Encryption system](#).

Payload

The payload is the final message that will be sent to the peer (secure manager or agent). It starts with `:` and, if and only if the agent entry allows more than one host (address `any` or netmask different from 32), the agent ID between two `!` symbols:

```
<Payload> =  
  ":" <Encrypted>,           if <Netmask> = 32  
  !" <Agent ID> ":" <Encrypted>, otherwise
```

Complete encryption formula

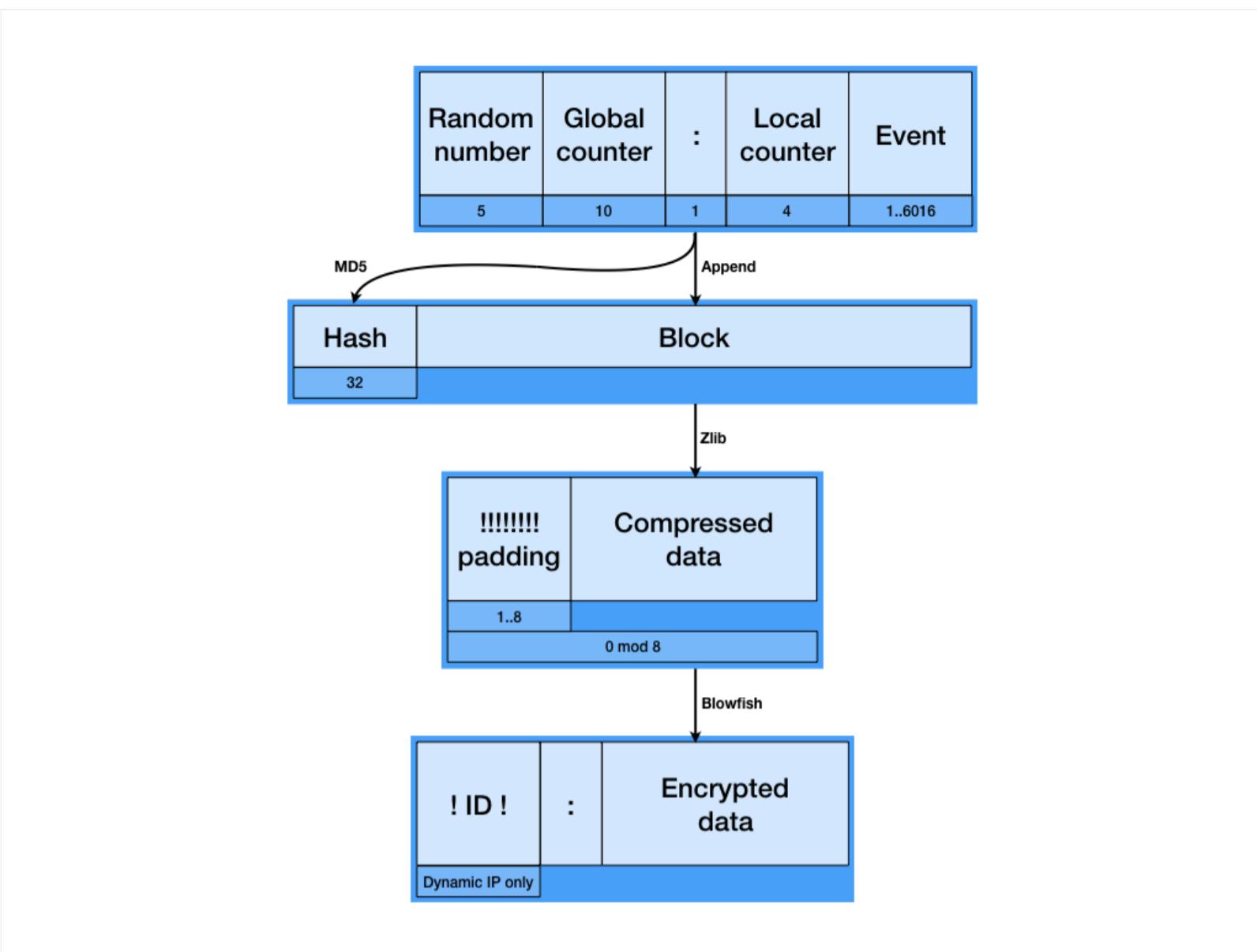
For agents with restricted address:

```
 ":" Blowfish(<!-padding> Gzip(MD5(<Random> <Global> ":" <Local> <Event>)) <Random> <Global> ":" <Local>  
<Event>))
```

For agents with unrestricted address (address `any` or netmask different from 32):

```
 !" <ID> ":" Blowfish(<!-padding> Gzip(MD5(<Random> <Global> ":" <Local> <Event>)) <Random> <Global> ":"  
<Local> <Event>))
```

This is the **encryption flow chart**:



Network protocol

The procedure to send a payload via network depends on the connection protocol:

UDP protocol

The datagram is the payload itself:

```
Send(<Payload>)
```

TCP protocol

Messages are not delimited by the network, so the payload size must be prepended to the payload:

```
Send(<Size> <Payload>)
```

The **Size** has the following format:

Size	4 bytes
Sign	Unsigned
Endianness	Little-endian

Encryption system

The encryption system uses a constant initialization vector and a key:

Initialization vector

8-byte hexadecimal array:

```
<IV> = FE DC BA 98 76 54 32 10
```

Encryption key

The key is built by appending and cutting hexadecimal strings depending on some agent attributes (see [Client keys file](#)):

```
<Key> = MD5(<Pass>) MD5(MD5(<Name>) MD5(<ID>))[0:15]
```

To clarify: the second MD5 hash is cut to its first 15 bytes (from 0 to 14th).

Remote service

You can set how Wazuh Manager could publish their remote service used by Agents:

Configuration

All configuration of Remote Service is done via `ossec.conf` using `<remote>` XML tag, all the available options are detailed in [Remote config](#)

You can change what ip address use to listen the service:

```
<ossec_config>
  <remote>
    <local_ip>10.0.0.10</local_ip>
  </remote>
</ossec_config>
```

This will set the default listen ip address to 10.0.0.10.

When you change any value on `ossec.conf` file, you need to restart the service to enabling previously changed values.

a. For Systemd:

```
systemctl restart wazuh-manager
```

b. For SysV Init:

```
service wazuh-manager restart
```


Defining an alert level threshold

Every possible event on the Wazuh Agent is set with certain level, by default is 1, all events from this level will trigger and alert into Wazuh Manager.

Configuration

All configuration of Remote Service is done via `ossec.conf` using `<alerts>` XML tag, all the available options are detailed in [Alerts reference](#)

```
<ossec_config>
  <alerts>
    <log_alert_level>6</log_alert_level>
  </alerts>
</ossec_config>
```

This will set to level 6 the minimum severity level for alerts to be stored to alerts.log and/or alerts.json.

When you change any value on `ossec.conf` file, you need to restart the service to enabling previously changed values.

a. For Systemd:

```
systemctl restart wazuh-manager
```

b. For SysV Init:

```
service wazuh-manager restart
```


Integration with external APIs

Integrator is a new daemon that allows the connection of Wazuh to external APIs and alerting tools such as Slack and PagerDuty.

Configuration

Integrator is not enabled by default. Integrator is enabled using the following command:

```
$ /var/ossec/bin/ossec-control enable integrator  
$ /var/ossec/bin/ossec-control restart
```

Integrations are configured in the file `etc/ossec.conf`, which is located inside your Wazuh installation directory. Add the following inside `<ossec_config>` `</ossec_config>` to configure this integration:

```
<integration>  
  <name> </name>  
  <hook_url> </hook_url>  
  <api_key> </api_key>  
  
  <!-- Optional filters -->  
  
  <rule_id> </rule_id>  
  <level> </level>  
  <group> </group>  
  <event_location> </event_location>  
</integration>
```

Integration with Slack

```
<integration>  
  <name>slack</name>  
  <hook_url>https://hooks.slack.com/services/...</hook_url>  
</integration>
```

Integration with PagerDuty

```
<integration>  
  <name>pagerduty</name>  
  <api_key>MYKEY</api_key>  
</integration>
```


Configuring syslog output

Wazuh may be configured to send alerts to syslog as follows:

Configuration

Syslog output is configured in `ossec.conf`. All the available options are detailed in [Syslog output](#)

```
<ossec_config>
  <syslog_output>
    <level>9</level>
    <server>192.168.1.241</server>
  </syslog_output>

  <syslog_output>
    <server>192.168.1.240</server>
  </syslog_output>
</ossec_config>
```

The above configuration will send alerts to `192.168.1.240` and, if the alert level is higher than 9, will also send the alert to `192.168.1.241`.

After the configuration of the `ossec.conf` file, the client-syslog must be enabled followed by a restart of Wazuh using the following command:

```
/var/ossec/bin/ossec-control enable client-syslog
```

a. For Systemd:

```
systemctl restart wazuh-manager
```

b. For SysV Init:

```
service wazuh-manager restart
```


Generating automatic reports

Daily reports are summaries of the alerts for the day. You can configure your own report. Configuration of report is done in the `ossec.conf` file using the `report` option. More information: [Report](#), make sure you have plenty configure your SMTP server in order to receive emails, see [Configuring email alerts](#) and [SMTP server with authentication](#) for reference.

```
<ossec_config>
  <reports>
    <category>syscheck</category>
    <title>Daily report: File changes</title>
    <email_to>example@test.com</email_to>
  </reports>
</ossec_config>
```



The above configuration will send a daily report of all `syscheck` alerts.

Rules may also be filtered by level, source, username, rule id, etc.

For example:

```
<ossec_config>
  <reports>
    <level>10</level>
    <title>Daily report: Alerts with level higher than 10</title>
    <email_to>example@test.com</email_to>
  </reports>
</ossec_config>
```



The above configuration will send a report with all rules that fired with a level higher than 10.

Example of generated report

From: Wazuh

12:01 AM (10 hours ago)

to me

Report 'Daily report: File changes' completed.

->Processed alerts: 368
->Post-filtering alerts: 58
->First alert: 2017 Mar 08 06:31:26
->Last alert: 2017 Mar 08 13:11:42

Top entries for 'Level':

Severity 5	47	
Severity 7	11	

Top entries for 'Group':

ossec	58	
pci_dss_11.5	58	
syscheck	58	

Top entries for 'Location':

localhost->syscheck	51	
(ubuntu) 192.168.1.242->syscheck	7	

Top entries for 'Rule':

554 - File added to the system.	47	
550 - Integrity checksum changed.	11	

Top entries for 'Filenames':

/boot/grub/grub.cfg	1	
/etc/apt/apt.conf.d/01autoremove-kernels	1	
/etc/group	1	
/etc/group-	1	
/etc/gshadow	1	
/etc/gshadow-	1	
/etc/passwd	1	
/etc/passwd-	1	
/etc/postfix/main.cf	1	
/etc/shadow	1	
/etc/shadow-	1	

Configuring email alerts

Wazuh can be configured to send the alerts to an email. You can configure the system to send emails when certain rules are triggered or configure it to send a daily report.

Mail example:

```
From: Wazuh <you@example.com>           5:03 PM (2 minutes ago)
to: me
-----
Wazuh Notification.
2017 Mar 08 17:03:05

Received From: localhost->/var/log/secure
Rule: 5503 fired (level 5) -> "PAM: User login failed."
Src IP: 192.168.1.37
Portion of the log(s):

Mar 8 17:03:04 localhost sshd[67231]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0
tty=ssh ruser= rhost=192.168.1.37
uid: 0
euid: 0
tty: ssh

--END OF NOTIFICATION
```

Generic email options

In order to configure Wazuh to send alerts through email, you need to configure the email settings inside the `<global>` section:

```
<ossec_config>
  <global>
    <email_notification>yes</email_notification>
    <email_to>me@test.com</email_to>
    <smtp_server>mail.test.com..</smtp_server>
    <email_from>wazuh@test.com</email_from>
  </global>
  ...
</ossec_config>
```

To see all the available options to configure it, go to [global section](#)

After the global configuration, we need to configure the `email_alert_level`. This option establishes the minimum level to send an alert. By default is set to 7.

```
<ossec_config>
  <alerts>
    <email_alert_level>10</email_alert_level>
  </alerts>
  ...
</ossec_config>
```

This example will set the minimum level to 10. More information: [alerts section](#).

When you have configured the `alert_level`, Wazuh needs to be restarted for the change take effect

a) For Systemd:

```
systemctl status wazuh-manager
```

b) For SysV Init:

```
service wazuh-manager status
```

⚠ Warning

Wazuh doesn't handle SMTP authentication. If you want to use an email with it, you need to [configure a server relay](#).

Granular email options

Wazuh also allows a very granular configuration options for your alerts through email. Here you will find some examples of the granular configuration. More info: [email_alerts section](#)

⚠ Warning

The minimum level you configured inside `alerts` section, will be also valid here.

So, for example, if you configure your system to send an email when the rule 526 is triggered, if that rule has a level lower than the configured on the previous section the alert will not be sent.

Email alert based on level

The general configuration will be:

```
<email_alerts>
  <email_to>you@example.com</email_to>
  <level>4</level>
  <do_not_delay />
</email_alerts>
```

This will send to `you@example` and email if the any rule with level greater or equal to 10 is triggered. Remember, if the level here is less than the `email_alert_level` configured on the previous section, this will not be sent.

Email alert based on level and agent

The general configuration will be:

```
<email_alerts>
  <email_to>you@example.com</email_to>
  <event_location>server1</event_location>
  <do_not_delay />
</email_alerts>
```

This will send to `you@example` and email if the for the rules triggered on the `server1`. Also, `event_location` can be configured to monitor a specific log, hostname or network (IP)

Email based on rules ID

```
<email_alerts>
  <email_to>you@example.com</email_to>
  <rule_id>515, 516</rule_id>
  <do_not_delay />
</email_alerts>
```

This will send an email if the rules 515 or 516 are triggered on any agent.

Email based on the group

Each rule can have one or more groups configured. We can use this groups to filter the rules that we want to send through email:

```
<email_alerts>
  <email_to>you@example.com</email_to>
  <group>pci_dss_10.6.1</group>
</email_alerts>
```

This will send an alert if any rule part of the `pci_dss_10.6.1` group is triggered on any machine.

Multiples options and multiples email

This example will show you the real capacity of this capability:

```
<ossec_config>
  <email_alerts>
    <email_to>alice@test.com</email_to>
    <event_location>server1|server2</event_location>
  </email_alerts>
  <email_alerts>
    <email_to>is@test.com</email_to>
    <event_location>/log/secure$</event_location>
  </email_alerts>
  <email_alerts>
    <email_to>bob@test.com</email_to>
    <event_location>192.168.</event_location>
  </email_alerts>
  <email_alerts>
    <email_to>david@test.com</email_to>
    <level>12</level>
  </email_alerts>
</ossec_config>
```

This configuration will send:

- An email to `alice@test.com` if any alert on server1 or server2 is triggered
- An email to `is@test.com` if the alerts came from `/log/secure/`
- An email to `bob@test.com` if the alerts came from any machine on the `192.168.0.0/24` network
- An email to `david@test.com` if the alerts have a level equals or higher than 12.

Force forwarding an alert by email

It's also possible to force the mail alert on the rule declaration. In order to do so, you need to use `option`

The possible values for this option are:

- **alert_by_email**: Always alert by email.
- **no_email_alert**: Never alert by email.
- **no_log**: Do not log this alert.

So for example this rule:

```
<rule id="502" level="3">
  <if_sid>500</if_sid>
  <options>alert_by_email</options>
  <match>Ossec started</match>
  <description>Ossec server started.</description>
</rule>
```

This will send an email every time this rule is triggered. I doesn't matter the level minimum level configured on the `<alerts>` section in `ossec.conf`

Using the command line

There are some actions you can do via command line:

Contents

- [Register Agent](#)
 - [Forcing insertion](#)
- [Listing Agents](#)
- [Remove Agents](#)

Using the RESTful API

Similar to command-line you can do certain task via RESTful API:

Contents

- [Register Agents](#)
- [Listing Agents](#)
- [Remove Agents](#)

Using Wazup App

Listing Agents

You can list and see all Registered Agents with only accessing to the Wazuh app and go to *Agents* tab:

The screenshot shows the Wazuh interface with the 'AGENTS' tab selected. At the top, there's a summary box for the last registered agent ('vpc-agent-centos-public') and another for higher activity ('vpc-ossec-manager'). Below this, a search bar and filter options are available. The main table lists agents with columns for ID, Name, IP, and Status. The listed agents are:

ID	Name	IP	Status
000	vpc-ossec-manager	127.0.0.1	Active
003	vpc-agent-debian	10.0.0.121	Active
005	vpc-agent-ubuntu-public	10.0.0.126	Active

Show Agent

Clicking on certain Agent will show you more information about it.

The screenshot shows the Wazuh interface with the 'AGENTS' tab selected, displaying detailed information for the agent '1041 - vpc-agent-centos-public'. Key details shown include the agent's name, IP address, version, operating system, and last keep alive time. Below this, several dashboards provide real-time monitoring data:

- Top 5 alerts:** A donut chart showing alert types: sshd, unix_chkpwd, PAM, syslog, and Host-based anomaly detection.
- Top 5 groups:** A donut chart showing group types: syslog, authentication_failed, pam, sshd, and access_control.
- Top 5 PCI DSS Requirements:** A donut chart showing requirements: 10.2.4, 10.2.5, 10.6.1, 2.2.4, and 11.4.
- Alert level evolution:** A line graph showing the count of alerts per hour over the last 24 hours.
- Alerts:** A bar chart showing the count of agent alerts over time.

Checking connection with Manager

First you need to be sure that the Agent is pointing to Manager Address this is set on `ossec.conf` using `<client>` XML tag, for more see [Client reference](#).

```
<ossec_config>
  <client>
    <server-ip>10.0.0.10</server-ip>
    <protocol>udp</protocol>
  </client>
</ossec_config>
```

This will set 10.0.0.10 as Wazuh Manager server, after you need restart the Agent:

a. For Systemd:

```
systemctl restart wazuh-agent
```

b. For SysV Init:

```
service wazuh-agent restart
```

After you register the agent and it be successfully connected, you could see a list of connected agents into Manager with:

```
$ /var/ossec/bin/agent_control -lc
```

This will display every registered Agent, also you can check if a Agent is correctly connected verifying if the UDP connection to Manager is established:

```
$ netstat -vatunp|grep ossec-agentd
```

This could match with the Agent and Manager ip addresses.

Getting started

The default number of rules and decoders is limited. For this reason, we centralize, test and maintain decoders and rules submitted by open source contributors. We also create new rules and rootchecks periodically and add them to this repository so they can be used by the user community. Some examples are the new rules for Netscaler and Puppet.

GitHub repository

In the ruleset repository you will find:

- **New rules, decoders and rootchecks**

We update and maintain the out-of-the-box rules provided by OSSEC, both to eliminate false positives and to increase accuracy. In addition, we map the rules to PCI-DSS compliance controls, making it easy to identify when an alert is related to a specific compliance requirement.

- **Tools**

We provide some useful tools for testing.

Resources

- Visit our repository to view the rules in detail at [Github Wazuh Ruleset](#)
- Find a complete description of the available rules at [Wazuh Ruleset Summary](#)

Rule and Rootcheck example

Log analysis rule for Netscaler with PCI DSS compliance mapping:

```
<rule id="80102" level="10" frequency="6">
  <if_matched_sid>80101</if_matched_sid>
  <same_source_ip />
  <description>Netscaler: Multiple AAA failed to login the user</description>
  <group>authentication_failures,netscaler-aaa,pci_dss_10.2.4,pci_dss_10.2.5,pci_dss_11.4,</group>
</rule>
```

Rootcheck rule for SSH Server with mapping to CIS security benchmark and PCI DSS compliance:

```
[CIS - Debian Linux - 2.3 - SSH Configuration - Empty passwords permitted {CIS: 2.3 Debian Linux} {PCI_DSS: 4.1}] [any] [http://www.ossec.net/wiki/index.php/CIS_DebianLinux]
f:/etc/ssh/sshd_config -> !r:^# && r:^PermitEmptyPasswords\.+yes;
```

Directory layout

The ruleset folder structure is shown below:

```
/var/ossec/
  └── etc/
    ├── decoders/
    │   └── local_decoder.xml
    └── rules/
        └── local_rules.xml
  └── ruleset/
      ├── decoders/
      └── rules/
```

Inside the `ruleset/` folder you will find all the common rules and decoders. All files inside this folder **will be overwritten** or modified in the Wazuh update process, so please do not edit files or add custom files in this folder.

If we need to perform some **custom** changes, we will use the `etc/` folder. You can add here your own decoders/rules files or use the default `local_decoder.xml` and `local_rules.xml` files.

Update ruleset

Run the `update_ruleset.py` script to update the Wazuh ruleset. You should not need to make any other changes to accommodate the updated rules.

Usage examples

Update Decoders, Rules and Rootchecks:

```
$ /var/ossec/bin/update_ruleset.py
```

All script options:

```
Restart:  
  -r, --restart      Restart OSSEC when required.  
  -R, --no-restart   Do not restart OSSEC when required.  
  
Backups:  
  -b, --backups     Restore last backup.  
  
Additional Params:  
  -f, --force-update Force to update the ruleset. By default, only it is updated the new/changed  
decoders/rules/rootchecks.  
  -o, --ossec-path   Set OSSEC path. Default: '/var/ossec'  
  -s, --source       Select ruleset source path (instead of download it).  
  -j, --json         JSON output. It should be used with '-s' or '-S' argument.  
  -d, --debug        Debug mode.
```

Configure weekly updates

Run `update_ruleset.py` weekly and keep your Wazuh Ruleset installation up to date by adding a crontab job to your system.

One way to do this would be to run `sudo crontab -e` and, at the end of the file, add the following line

```
@weekly root cd /var/ossec/bin && ./update_ruleset.py -r
```


Custom rules and decoders

It is possible to modify the default rules and decoders from the Wazuh Ruleset and also to add new ones in order to increase Wazuh's detection capabilities.

Adding new decoders and rules

Note

We will use `local_decoder.xml` and `local_rules.xml` to implement small changes. For larger scale changes/additions to the stock decoders and rules, we recommend you create a new decoder and/or rule file.

We are going to describe these procedures using an easy example. Here is a log from a program called `example`:

```
Dec 25 20:45:02 MyHost example[12345]: User 'admin' logged from '192.168.1.100'
```

First, we need to decode this information, so we add the new decoder to `/var/ossec/etc/decoders/local_decoder.xml`:

```
<decoder name="example">
  <program_name>^example</program_name>
</decoder>

<decoder name="example">
  <parent>example</parent>
  <regex>User '(\w+)' logged from '(\d+\.\d+\.\d+\.\d+)'</regex>
  <order>user, srcip</order>
</decoder>
```

Now, we will add the following rule to `/var/ossec/etc/rules/local_rules.xml`:

```
<rule id="100010" level="0">
  <program_name>example</program_name>
  <description>User logged</description>
</rule>
```

We can check if it works by using `/var/ossec/bin/ossec-logtest`:

```
**Phase 1: Completed pre-decoding.
full event: 'Dec 25 20:45:02 MyHost example[12345]: User 'admin' logged from '192.168.1.100''
hostname: 'MyHost'
program_name: 'example'
log: 'User 'admin' logged from '192.168.1.100'''

**Phase 2: Completed decoding.
decoder: 'example'
dstuser: 'admin'
srcip: '192.168.1.100'

**Phase 3: Completed filtering (rules).
Rule id: '100010'
Level: '0'
Description: 'User logged'
```

Changing an existing rule

You can modify the standard rules.

⚠ Warning

Changes to any rule file inside the `/var/ossec/ruleset/rules` folder will be lost in the update process. Use the following procedure to preserve your changes.

If we want to change the level value of the SSH rule `5710` from 5 to 10, we will do the following:

1. Open the rule file `/var/ossec/ruleset/rules/0095-sshd_rules.xml`.
2. Find and copy the following code from the rule file:

```
<rule id="5710" level="5">
  <if_sid>5700</if_sid>
  <match>illegal user|invalid user</match>
  <description>sshd: Attempt to login using a non-existent user</description>
  <group>invalid_login,authentication_failed,pci_dss_10.2.4,pci_dss_10.2.5,pci_dss_10.6.1,</group>
</rule>
```

3. Paste the code into `/var/ossec/etc/rules/local_rules.xml`, modify the level value, and add `overwrite="yes"` to indicate that this rule is overwriting an already defined rule:

```
<rule id="5710" level="10" overwrite="yes">
  <if_sid>5700</if_sid>
  <match>illegal user|invalid user</match>
  <description>sshd: Attempt to login using a non-existent user</description>
  <group>invalid_login,authentication_failed,pci_dss_10.2.4,pci_dss_10.2.5,pci_dss_10.6.1,</group>
</rule>
```

Changing an existing decoder

You can also modify the standard decoders.

⚠ Warning

Changes in any decoder file in the `/var/ossec/ruleset/decoders` folder will be lost in the update process. Use the following procedure to preserve your changes.

Unfortunately, there is no facility for overwriting decoders in the way described for rules above. However, we can perform changes in any decoder file as follows:

If we want to change something in the decoder file `0310-ssh_decoders.xml`, we will do the following:

1. Copy the decoder file `/var/ossec/ruleset/decoders/0310-ssh_decoders.xml` from the default folder to the user folder `/var/ossec/etc/decoders` in order to keep the changes.
2. Exclude the original decoder file `ruleset/decoders/0310-ssh_decoders.xml` from the OSSEC loading list. To do this, use the tag `<decoder_exclude>` in the `ossec.conf` file. Thus, the specified decoder will not be loaded from the default decoder folder, and the decoder file saved in the user folder will be loaded instead.

```
<ruleset>
  <!-- Default ruleset -->
  <decoder_dir>ruleset/decoders</decoder_dir>
  <rule_dir>ruleset/rules</rule_dir>
  <rule_exclude>0215-policy_rules.xml</rule_exclude>
  <list>etc/lists/audit-keys</list>

  <!-- User-defined ruleset -->
  <decoder_dir>etc/decoders</decoder_dir>
  <rule_dir>etc/rules</rule_dir>
  <decoder_exclude>ruleset/decoders/0310-ssh_decoders.xml</decoder_exclude>
</ruleset>
```

3. Perform the changes in the file `/var/ossec/etc/decoders/0310-ssh_decoders.xml`.

Warning

Note that at this point, if updates to the public Wazuh Ruleset include changes to 0310-ssh_decoders.xml, they will not apply to you since you are no longer loading that decoder file from the standard location that gets updates. At some point you may have to manually migrate your customized material from 0310-ssh_decoders.xml to a newer copy of that file. Consider internally documenting your changes in 0310-ssh_decoders.xml so that they are easy to find if they have to be migrated later.

Dynamic fields

Traditional decoders

An important step for the detection and processing of threats is the extraction of information from each event received. Wazuh uses decoders to identify event types and then extract the most relevant fields, thus enriching events and allowing them to be more deeply analyzed and indexed.

Traditionally, OSSEC has provided thirteen predefined fields for storing extracted information (*user, srcip, dstip, srcport, dstport, protocol, action, id, url, data, extra_data, status, system_name*), of which only eight can be extracted simultaneously.

Static fields:

```
<decoder name="web-accesslog">
  <type>web-log</type>
  <prematch>^\d+\.\d+\.\d+\.\d+ - </prematch>
  <regex>^(\d+\.\d+\.\d+\.\d+) - \S+ [\S+ -\d+] </regex>
  <regex>"\w+ (\S+) HTTP\S+ (\d+) </regex>
  <order>srcip,url,id</order>
</decoder>
```

Dynamic decoders

It is often necessary to extract more than eight relevant fields from an event, and often the actual data items extracted have no relationship to the limited list of predefined field names. Knowing that we cannot afford to operate within these constraints, Wazuh has extended OSSEC to allow the decoding of an unlimited number of fields with field names that clearly relate to what is being extracted. Even nested field names are supported.

Dynamic fields:

```
<decoder name="auditd-config_change">
  <parent>auditd</parent>
  <regex offset="after_regex">^auid=(\S+) ses=(\S+) op="(.+)"</regex>
  <order>audit.auid,audit.session,audit.op</order>
</decoder>
```

Wazuh transforms any field name included in the `<order>` tag into a JSON field.

The next example shows how the auditd decoder extracts the information from an alert:

```
** Alert 1486483073.60589: - audit,audit_configuration,
2017 Feb 07 15:57:53 wazuh-example->/var/log/audit/audit.log
Rule: 80705 (level 3) -> 'Auditd: Configuration changed'
type=CONFIG_CHANGE msg=audit(1486483072.194:20): auid=0 ses=6 op="add rule" key="audit-wazuh-a" list=4 res=1
audit.type: CONFIG_CHANGE
audit.id: 20
audit.auid: 0
audit.session: 6
audit.op: add rule
audit.key: audit
audit.list: 4
audit.res: 1
```

JSON Output:

```
{
  "rule": {
    "level": 3,
    "description": "Auditd: Configuration changed",
    "id": 80705,
    "firedtimes": 2,
    "groups": [
      "audit",
      "audit_configuration"
    ],
  },
  "agent": {
    "id": "000",
    "name": "wazuh-example"
  },
  "manager": {
    "name": "wazuh-example"
  },
  "full_log": "type=CONFIG_CHANGE msg=audit(1486483072.194:20): auid=0 ses=6 op=\"add rule\" key=\"audit-wazuh-a\" list=4 res=1",
  "audit": {
    "type": "CONFIG_CHANGE",
    "id": "20",
    "auid": "0",
    "session": "6",
    "op": "add rule",
    "key": "audit",
    "list": "4",
    "res": "1"
  },
  "decoder": {
    "parent": "auditd",
    "name": "auditd"
  },
  "timestamp": "2017 Feb 07 15:57:53",
  "location": "/var/log/audit/audit.log"
}
}
```

Note

By default, the number of fields that can be extracted simultaneously from an `<order>` tag is **64**. This value can be modified by changing the variable `analysisd.decoder_order_size` seen in `/var/ossec/etc/internal_options.conf`. If you need to change this value, copy the `analysisd.decoder_order_size` section from `/var/ossec/etc/internal_options.conf` to `/var/ossec/etc/local_internal_options.conf` and change it there, since Wazuh software updates can replace `/var/ossec/etc/internal_options.conf`

Testing decoders and rules

The tool *ossec-logtest* allow us to test how an event is decoded and if an alert is generated.

Run the tool */var/ossec/bin/ossec-logtest* and paste the following log:

```
Mar  8 22:39:13 ip-10-0-0-10 sshd[2742]: Accepted publickey for root from 73.189.131.56 port 57516
```

```
$ /var/ossec/bin/ossec-logtest
```

```
Mar  8 22:39:13 ip-10-0-0-10 sshd[2742]: Accepted publickey for root from 73.189.131.56 port 57516
```

**Phase 1: Completed pre-decoding.

```
    full event: 'Mar  8 22:39:13 ip-10-0-0-10 sshd[2742]: Accepted publickey for root from 73.189.131.56
port 57516'
        hostname: 'ip-10-0-0-10'
        program_name: 'sshd'
        log: 'Accepted publickey for root from 73.189.131.56 port 57516'
```

**Phase 2: Completed decoding.

```
    decoder: 'sshd'
    dstuser: 'root'
    srcip: '73.189.131.56'
```

**Phase 3: Completed filtering (rules).

```
    Rule id: '5715'
    Level: '3'
    Description: 'sshd: authentication success.'
```

**Alert to be generated.

Warning

The decoder name showed in *Phase 2* will be the name of the parent decoder.

In addition, you can use the option “-v” to show more information about the rules:

```
$ /var/ossec/bin/ossec-logtest -v

Mar  8 22:39:13 ip-10-0-0-10 sshd[2742]: Accepted publickey for root from 73.189.131.56 port 57516

**Phase 1: Completed pre-decoding.
  full event: 'Mar  8 22:39:13 ip-10-0-0-10 sshd[2742]: Accepted publickey for root from 73.189.131.56
port 57516'
    hostname: 'ip-10-0-0-10'
    program_name: 'sshd'
    log: 'Accepted publickey for root from 73.189.131.56 port 57516'

**Phase 2: Completed decoding.
  decoder: 'sshd'
  dstuser: 'root'
  srcip: '73.189.131.56'

**Rule debugging:
  Trying rule: 1 - Generic template for all syslog rules.
    *Rule 1 matched.
      *Trying child rules.
        Trying rule: 600 - Active Response Messages Grouped
        Trying rule: 2100 - NFS rules grouped.
        Trying rule: 2507 - OpenLDAP group.
        Trying rule: 2550 - rshd messages grouped.
        Trying rule: 2701 - Ignoring procmail messages.
        Trying rule: 2800 - Pre-match rule for smartd.
        Trying rule: 5100 - Pre-match rule for kernel messages
        Trying rule: 5200 - Ignoring hpiod for producing useless logs.
        Trying rule: 2830 - Crontab rule group.
        Trying rule: 5300 - Initial grouping for su messages.
        Trying rule: 5905 - useradd failed.
        Trying rule: 5400 - Initial group for sudo messages
        Trying rule: 9100 - PPTPD messages grouped
        Trying rule: 9200 - Squid syslog messages grouped
        Trying rule: 2900 - Dpkg (Debian Package) log.
        Trying rule: 2930 - Yum logs.
        Trying rule: 2931 - Yum logs.
        Trying rule: 2940 - NetworkManager grouping.
        Trying rule: 2943 - nouveau driver grouping
        Trying rule: 3100 - Grouping of the sendmail rules.
        Trying rule: 3190 - Grouping of the smf-sav sendmail milter rules.
        Trying rule: 3300 - Grouping of the postfix reject rules.
        Trying rule: 3320 - Grouping of the postfix rules.
        Trying rule: 3390 - Grouping of the clamsmtpd rules.
        Trying rule: 3395 - Grouping of the postfix warning rules.
        Trying rule: 3500 - Grouping for the spamd rules
        Trying rule: 3600 - Grouping of the imapd rules.
        Trying rule: 3700 - Grouping of mailscanner rules.
        Trying rule: 3800 - Grouping of Exchange rules.
        Trying rule: 3900 - Grouping for the courier rules.
        Trying rule: 4300 - Grouping of PIX rules
        Trying rule: 4500 - Grouping for the Netscreen Firewall rules
        Trying rule: 4700 - Grouping of Cisco IOS rules.
        Trying rule: 4800 - SonicWall messages grouped.
        Trying rule: 5500 - Grouping of the pam_unix rules.
        Trying rule: 5556 - unix_chkpwd grouping.
        Trying rule: 5600 - Grouping for the telnetd rules
        Trying rule: 5700 - SSHD messages grouped.
          *Rule 5700 matched.
            *Trying child rules.
              Trying rule: 5709 - sshd: Useless SSHD message without an user/ip and context.
              Trying rule: 5711 - sshd: Useless/Duplicated SSHD message without a user/ip.
              Trying rule: 5721 - sshd: System disconnected from sshd.
              Trying rule: 5722 - sshd: ssh connection closed.
              Trying rule: 5723 - sshd: key error.
              Trying rule: 5724 - sshd: key error.
              Trying rule: 5725 - sshd: Host ungracefully disconnected.
```

```
Trying rule: 5727 - sshd: Attempt to start sshd when something already bound to the port.
Trying rule: 5729 - sshd: Debug message.
Trying rule: 5732 - sshd: Possible port forwarding failure.
Trying rule: 5733 - sshd: User entered incorrect password.
Trying rule: 5734 - sshd: sshd could not load one or more host keys.
Trying rule: 5735 - sshd: Failed write due to one host disappearing.
Trying rule: 5736 - sshd: Connection reset or aborted.
Trying rule: 5750 - sshd: could not negotiate with client.
Trying rule: 5756 - sshd: subsystem request failed.
Trying rule: 5707 - sshd: OpenSSH challenge-response exploit.
Trying rule: 5701 - sshd: Possible attack on the ssh server (or version gathering).
Trying rule: 5706 - sshd: insecure connection attempt (scan).
Trying rule: 5713 - sshd: Corrupted bytes on SSHD.
Trying rule: 5731 - sshd: SSH Scanning.
Trying rule: 5747 - sshd: bad client public DH value
Trying rule: 5748 - sshd: corrupted MAC on input
Trying rule: 5702 - sshd: Reverse lookup error (bad ISP or attack).
Trying rule: 5710 - sshd: Attempt to login using a non-existent user
Trying rule: 5716 - sshd: authentication failed.
Trying rule: 5718 - sshd: Attempt to login using a denied user.
Trying rule: 5726 - sshd: Unknown PAM module, PAM misconfiguration.
Trying rule: 5737 - sshd: cannot bind to configured address.
Trying rule: 5738 - sshd: pam_loginuid could not open loginuid.
Trying rule: 5704 - sshd: Timeout while logging in.
Trying rule: 5717 - sshd: configuration error (moduli).
Trying rule: 5728 - sshd: Authentication services were not able to retrieve user credentials.
Trying rule: 5730 - sshd: SSHD is not accepting connections.
Trying rule: 5739 - sshd: configuration error (AuthorizedKeysCommand)
Trying rule: 5740 - sshd: connection reset by peer
Trying rule: 5741 - sshd: connection refused
Trying rule: 5742 - sshd: connection timed out
Trying rule: 5743 - sshd: no route to host
Trying rule: 5744 - sshd: port forwarding issue
Trying rule: 5745 - sshd: transport endpoint is not connected
Trying rule: 5746 - sshd: get_remote_port failed
Trying rule: 5749 - sshd: bad packet length
Trying rule: 5715 - sshd: authentication success.

*Rule 5715 matched.
*Trying child rules.
```

```
Trying rule: 40101 - System user successfully logged to the system.
Trying rule: 40112 - Multiple authentication failures followed by a success.
```

**Phase 3: Completed filtering (rules).

```
  Rule id: '5715'
  Level: '3'
  Description: 'sshd: authentication success.'
```

**Alert to be generated.

Using CDB lists

OSSEC is able to check if a field extracted during the decoding phase is in a CDB list (constant database). The main use case of this feature is to create a white/black list of users, IPs or domain names.

Creating a CDB list

Creating the list file

The list file is a plain text file where each line has the following format:

```
key1:value1  
key2:value2
```

Each key must be unique and is terminated with a colon `:`.

For IP addresses the dot notation is used for subnet matches:

key	CIDR	Possible matches
192.168.::	192.168.0.0/16	192.168.0.0 - 192.168.255.255
172.16.19.::	172.16.19.0/24	172.16.19.0 - 172.16.19.255
10.1.1.1::	10.1.1.1/32	10.1.1.1

Example of IP address list file:

```
192.168.:: Matches 192.168.0.0 - 192.168.255.255  
172.16.19.:: Matches 172.16.19.0 - 172.16.19.255  
10.1.1.1:: Matches 10.1.1.1
```

We recommend to use the directory `/var/ossec/etc/lists` to store your lists.

Adding the list to ossec.conf

Each list must be defined in the ossec.conf file using the following syntax:

```
<ossec_config>  
  <ruleset>  
    <list>etc/lists/list-IP</list>
```

Restart OSSEC to apply the changes.

Making the CDB list

The list files must be compiled before they can be used. The tool `/var/ossec/bin/ossec-makelists` will process and compile all the lists if needed.

Remember to compile the lists every time that you update them. It is not necessary to restart OSSEC to apply the changes.

Using the CDB list in the rules

A rule would use the following syntax to look up a key within a CDB list.

Positive key match

This example is a search for the key stored in the field attribute and will match if it *IS* present in the database:

```
<list field="user" lookup="match_key">etc/lists/list-user</list>
```

The `lookup="match_key"` is the default and can be left out as in this example:

```
<list field="user">etc/lists/list-user</list>
```

In case the field is an IP address, you must use `address_match_key`:

```
<list field="srcip" lookup="address_match_key">etc/lists/list-IP</list>
```

Negative key match

This example is a search for the key stored in the field attribute and will match if it *IS NOT* present in the database:

```
<list field="user" lookup="not_match_key">etc/lists/list-user</list>
```

In case the field is an IP address, you must use `not_address_match_key`:

```
<list field="srcip" lookup="not_address_match_key">etc/lists/list-IP</list>
```

Key and value match

This example is a search for the key stored in the field attribute, and on a positive match the returned value of the key will be processed using the regex in the `check_value` attribute:

```
<list field="user" lookup="match_key_value" check_value="^block">etc/lists/list-user</list>
```

In case the field is an IP address, you must use `not_address_match_key`:

```
<list field="srcip" lookup="address_match_key_value" check_value="^reject">etc/lists/list-IP</list>
```

Contribute to the ruleset

If you have created new rules, decoders or rootchecks and you would like to contribute to our repository, please fork our [Github repository](#) and submit a pull request.

If you are not familiar with Github, you can also share them through our [mailing list](#), to which you can subscribe by sending an email to wazuh+subscribe@googlegroups.com. Also, do not hesitate to request new rules or rootchecks that you would like to see running in Wazuh. Our team will do our best to make it happen.

Note

In our repository you will find that most of the rules contain one or more groups called pci_dss_X. This is the PCI DSS control related to the rule. We have produced a document that can help you tag each rule with its corresponding PCI requirement: [PCI tagging](#)

Local configuration

The `ossec.conf` file is the main configuration file on the Wazuh manager, and it also plays a role on the agents. It is located at `/var/ossec/etc/ossec.conf` both in the manager and agent. It is recommended you back up this file before making changes to it, as an error in the configuration can completely prevent Wazuh services from starting up.

The `ossec.conf` file is in XML format, and all configuration options are nested in their appropriate section of the file. In this file, the outermost XML tag is `<ossec_config>`. For example, here is an example of the proper location of the `alerts` configuration section:

```
<ossec_config>
  <alerts>
    <!--
      alerts options here
    -->
  </alerts>
</ossec_config>
```

The `agent.conf` file is very similar to `ossec.conf` except that it is used to centrally distribute configuration information to agents. See more [here](#).

Wazuh can be installed in two possible ways: the Wazuh manager uses the “server/manager” installation type and agents use the “agent” installation type.

Configuration sections	Supported installations
active-response	manager, agent
agentless	manager
alerts	manager
auth	manager
client	agent
client_buffer	agent
command	manager
database_output	manager
email_alerts	manager
global	manager
integration	manager
labels	manager, agent
localfile	manager, agent
logging	manager, agent
remote	manager
reports	manager
rootcheck	manager, agent
ruleset	manager
syscheck	manager, agent
syslog_output	manager
wodle name="open-scap"	manager, agent

All of the above sections must be located within the top-level `<ossec_config>` tag.

Internal configuration

The main configuration is located in the `ossec.conf` file, however some internal configuration features are located in the `/var/ossec/etc/internal_options.conf` file.

Generally, this file is reserved for debugging issues and for troubleshooting. **Any error in this file may cause your installation to malfunction or fail to run.**

Warning

This file will be overwritten during upgrades. In order to maintain custom changes, you must use the `/var/ossec/etc/local_internal_options.conf` file.

- [Agent](#)
- [Analysisd](#)
- [DBD](#)
- [Logcollector](#)
- [Mайл](#)
- [Monitord](#)
- [Remoted](#)
- [Syscheck](#)
- [Rootcheck](#)
- [Wazuh_database](#)
- [Wazuh_modules](#)
- [Windows](#)

Agent

agent.tolerance	Description	Time in seconds since the agent is full until trigger a flooding alert.
	Default value	15
	Allowed value	Any integer between 0 and 600.
agent.warn_level	Description	Percentage of occupied capacity in Agent buffer to trigger a warning alert.
	Default value	90
	Allowed value	Any integer between 1 and 100.
agent.normal_level	Description	Percentage of occupied capacity in Agent buffer to come back to normal state.
	Default value	70
	Allowed value	Any integer between 0 and <code>agent.warn_level - 1</code> .
agent.min_eps	Description	Minimum events per second permitted in <code><client_buffer></code> configuration.
	Default value	50
	Allowed value	Any integer between 1 and 1000.
agent.debug	Description	Run the unix agent's processes in debug mode.
	Default value	0
	Allowed value	0 : No debug output 1: Standard debug output 2: Verbose debug outputNext,Previous

Analysisd

analysisd.default_timeframe	Description	Analysisd default rule timeframe.
	Default value	360
	Allowed value	Any integer between 60 and 360
analysisd.stats_maxdiff	Description	Analysisd stats maximum diff.
	Default value	999000
	Allowed value	Any integer between 10 and 99999
analysisd.stats_mindiff	Description	Analysisd stats minimum diff.
	Default value	1250
	Allowed value	Any integer between 10 and 99999
analysisd.stats_percent_diff	Description	Analysisd stats percentage (how much to differ from average).
	Default value	150
	Allowed value	Any integer between 5 and 9999
analysisd.fts_list_size	Description	Analysisd FTS list size.
	Default value	32
	Allowed value	Any integer between 12 and 512
analysisd.fts_min_size_for_str	Description	Analysisd FTS minimum string size.
	Default value	14
	Allowed value	Any integer between 6 and 128
analysisd.log_fw	Description	Analysisd Enable the firewall log (at logs/firewall/firewall.log).
	Default value	1
	Allowed value	0, 1
analysisd.decoder_order_size	Description	Maximum number of fields in a decoder (order tag).
	Default value	64
	Allowed value	Any integer between 10 and 64
analysisd.geoip_jsonout	Description	Output GeolP data at JSON alerts.
	Default value	0
	Allowed value	0, 1
analysisd.label_cache_maxage	Description	Time in seconds without reload labels in cache from agents.
	Default value	0
	Allowed value	Any integer between 0 and 60.
analysisd.show_hidden_labels	Description	Make hidden labels visible in alerts.
	Default value	0
	Allowed value	0, 1
analysisd.debug	Description	Debug level (manager installations)
	Default value	0
	Allowed value	0: No debug output
		1: Standard debug output
		2: Verbose debug output

DBD

dbd.reconnect_attempts	Description	The number of times ossec-dbd will attempt to reconnect to the database.
	Default value	10
	Allowed value	Any integer between 1 and 9999

Logcollector

logcollector.loop_timeout	Description	File polling interval.
	Default value	2
	Allowed value	Any integer between 1 and 120
logcollector.open_attempts	Description	Number of attempts to open a log file.
	Default value	8
	Allowed value	Any integer between 2 and 298
logcollector.remote_commands	Description	Enable/disable Logcollector to accept remote commands from the manager.
	Default value	0
	Allowed value	0, 1
logcollector.vcheck_files	Description	Number of readings before checking files.
	Default value	64
	Allowed value	Any integer between 0 and 1024
logcollector.max_lines	Description	Maximum number of logs read from the same file in each iteration.
	Default value	10000
	Allowed value	Any integer between 100 and 100000.
logcollector.debug	Description	Debug level (used in manager or unix agent installations)
	Default value	0
	Allowed value	0: No debug output
		1: Standard debug output
		2: Verbose debug output

Maild

maild.strict_checking	Description	Toggle to enable or disable strict checking.
	Default value	1
	Allowed value	0, 1
maild.grouping	Description	Toggle to enable or disable grouping of alerts into a single email.
	Default value	1
	Allowed value	0, 1
maild.full_subject	Description	Toggle to enable or disable full subject in alert emails.

	Default value	0	
	Allowed value	0, 1	
maild.geoip	Description	Toggle to enable or disable GeolP data in alert emails.	
	Default value	1	
	Allowed value	0, 1	

Monitord

	Description	Amount of seconds to wait before compressing or signing the files.
monitord.day_wait	Default value	10
	Allowed value	Any integer between 5 and 240
	Description	Toggle to enable or disable log file compression.
monitord.compress	Default value	1
	Allowed value	0, 1
	Description	Toggle to enable or disable signing the log files.
monitord.sign	Default value	1
	Allowed value	0, 1
	Description	Toggle to enable or disable monitoring of agents.
monitord.monitor_agents	Default value	1
	Allowed value	0, 1
	Description	Number of days to keep rotated internal logs.
monitord.keep_log_days	Default value	31
	Allowed value	0, 500

Remoted

	Description	Flush rate for the receive counter.
remoted.recv_counter_flush	Default value	128
	Allowed value	Any integer between 10 and 999999
	Description	Compression averages printout.
remoted.comp_average_printout	Default value	19999
	Allowed value	Any integer between 10 and 999999
	Description	Toggle to enable or disable verification of msg id.
remoted.verify_msg_id	Default value	0
	Allowed value	0, 1
	Description	Toggle to enable or disable acceptance of empty client.keys.
remoted.pass_empty_keyfile	Default value	1
	Allowed value	0, 1
	Description	Debug level (manager installation)
remoted.debug	Default value	0
	Allowed value	0: No debug output

1: Standard debug output

2: Verbose debug output

Syscheck

syscheck.sleep	Description	Number of seconds to sleep after reading syscheck.sleep_after number of files.
	Default value	2
	Allowed value	Any integer between 0 and 64
syscheck.sleep_after	Description	Number of files to read before sleeping for syscheck.sleep seconds.
	Default value	15
	Allowed value	Any integer between 1 and 9999
syscheck.debug	Description	Debug level (used in manager and unix agent installations).
	Default value	0
	Allowed value	0: No debug output
		1: Standard debug output
		2: Verbose debug output
		Next Previous

Rootcheck

rootcheck.sleep	Description	Number of milliseconds to sleep after reading one PID or suspicious port. ! New in version 2.1.
	Default value	50
	Allowed values	Any integer from 0 to 50.

Wazuh_database

The Wazuh core uses list-based databases to store information related to agent keys and FIM / Rootcheck event data. Such information is highly optimized to be handled by the core.

In order to provide well-structured data that could be accessed by the user or the Wazuh API, new **SQLite-based databases** have been introduced in the Wazuh manager. The Database Synchronization Module is a **user-transparent component** that collects the following information from the core:

- Agent's name, address, encryption key, last connection time, operating system, agent version and shared configuration hash.
- FIM data: creation, modification and deletion of regular files and Windows registry entries.
- Rootcheck detected defects: issue message, first detection date and last alert time.
- Static core settings, such as maximum permitted agents or SSL being enabled for Authd.

Note

The Wazuh Database Synchronization Module starts automatically on the server and local profiles and requires no configuration, however, some optional settings are available.

The module uses *inotify* from Linux to monitor changes to every log file in real-time. Databases will be updated as soon as possible when a change is detected. **If inotify is not supported**, (for example, on operating systems other than Linux) every log file will be scanned continuously, looking for changes, with a default delay of one minute between scans.

How to disable the module

To disable the Wazuh Database Synchronization Module, the sync directives must be set to 0 in the [etc/local_internal_options.conf](#) file as shown below:

```
wazuh_database.sync_agents=0
wazuh_database.sync_syscheck=0
wazuh_database.sync_rootcheck=0
```

Once these settings have been adjusted, save the file and **restart Wazuh**. With the above settings, the Database Synchronization Module will not be loaded when Wazuh starts.

wazuh_database.sync_agents	Description	Synchronize agent database with client.keys.
	Default value	1
	Allowed value	0, 1
wazuh_database.sync_syscheck	Description	Synchronize f.i.m. data with Syscheck database.
	Default value	1
	Allowed value	0, 1
wazuh_database.sync_rootcheck	Description	Synchronize policy monitoring data with Rootcheck database.
	Default value	1
	Allowed value	0, 1
wazuh_database.full_sync	Description	Full data synchronization.
	Default value	0
	Allowed value	0, 1
wazuh_database.sleep	Description	Interval to sleep between cycles. Only necessary if inotify not available.
	Default value	60
	Allowed value	Any integer between 0 and 86400 (seconds)
wazuh_database.max_queued_events	Description	Max number of queued events (only if inotify is available).
	Default value	0 (use system default value)
	Allowed value	Any integer between 0 and 2147483647

Wazuh_modules

wazuh_modules.task_nice	Description	Indicates the priority of the tasks. Lower Value, Higher priority.
	Default value	10
	Allowed value	Any integer between -20 and 19
wazuh_modules.max_eps	Description	Maximum number of events per second sent by OpenSCAP Wazuh Module.
	Default value	1000

	Allowed value	Any integer between 100 and 1000
	Description	Debug level
	Default value	0
wazuh_modules.debug		0: No debug output
	Allowed value	1: Standard debug output
		2: Verbose debug outputNext,Previous

Windows

	Description	Debug level (used in windows agent installations).
	Default value	0
windows.debug		0: No debug output
	Allowed value	1: Standard debug output
		2: Verbose debug outputNext,Previous

Daemons

Daemons	Descriptions	Supported installations
ossec-agentd	Client side daemon that communicates with the server	Agent
ossec-agentlessd	Runs integrity checking on systems without an agent installed	manager
ossec-analysisd	Receives log messages and compares them to the rules	manager
ossec-authd	Adds agents to Wazuh manager	manager
ossec-csyslogd	Forwards Wazuh alerts via syslog	manager
ossec-dbd	Inserts alert logs into a database	manager
ossec-execd	Executes active responses	manager, agent
ossec-logcollector	Monitors configured files and commands for new log messages	manager, agent
ossec-maild	Sends Wazuh alerts via email	manager
ossec-monitord	Monitors agent connectivity and compresses log files	manager
ossec-remoted	Communicates with agents	manager
ossec-reportd	Creates reports from Wazuh alerts	manager
ossec-syscheckd	Checks configured files for security changes	manager, agent
wazuh-modulesd	Wazuh module manager	manager, agent

Tools

Tools	Descriptions	Supported installations
ossec-control	Manages the status of Wazuh processes	manager, agent
agent-auth	Adds agents to a Wazuh manager	agent
agent_control	Allows queries of the manager to get information about any agent	manager
manage_agents	Provides an interface to handle authentication keys for agents	manager, agent
ossec-logtest	Allows testing and verification of rules against provided log records	manager
ossec-makelists	Compiles cdb databases	manager
rootcheck_control	Allows management of policy monitoring and system auditing database	manager
syscheck_control	Provides an interface for managing the integrity checking database	manager
syscheck_update	Updates the integrity check database	manager
clear_stats	Clears the events stats	manager
ossec-regex	Validates a regex expression	manager
update-ruleset.py	Update Decoders, Rules and Rootchecks	manager
util.sh	Adds a file to be monitored by ossec-logcollector	manager agent
verify-agent-conf	Verifies the Wazuh agent.conf configuration	manager

Register Agent

1. On the **manager**, run *manage_agents*:

```
$ /var/ossec/bin/manage_agents

*****
* Wazuh v2.0 Agent manager.          *
* The following options are available: *
*****
(A)dd an agent (A).
(E)xtract key for an agent (E).
(L)ist already added agents (L).
(R)emove an agent (R).
(Q)uit.

Choose your action: A,E,L,R or Q:
```

2. Press *A* and *Enter* to add an agent. You'll be asked for the agent's name (use the agent hostname or another arbitrary name), its IP and the agent ID (you can leave this field empty to auto-assign an ID).

In this example, we'll add an agent with name "Example", dynamic IP (*any*) and automatic ID:

```
Choose your action: A,E,L,R or Q: A

- Adding a new agent (use '\q' to return to the main menu).
Please provide the following:
 * A name for the new agent: Example
 * The IP Address of the new agent: any
 * An ID for the new agent[001]:
Agent information:
ID:001
Name:Example
IP Address:any

Confirm adding it?(y/n): y
Agent added with ID 001.
```

3. Extract the new agent's key. You will need it for the agent:

```
Choose your action: A,E,L,R or Q: E

Available agents:
ID: 001, Name: Example, IP: any
Provide the ID of the agent to extract the key (or '\q' to quit): 001

Agent key information for '001' is:
MDAxIDE4NWVlNjE1Y2YzYiBhbnkgMGNmMDFiYTM3NmMxY2JjNjU0NDAwYmFhZDY1ZWU1YjcyMGI2NDY3ODhkNGQzMjM5ZTdlNGVmNzQzMGFjMDA4Nw==
```

4. Exit from *manage_agents* by pressing *Q* and *Enter*.

5. Now on the **agent** run *manage_agents*:

```
$ /var/ossec/bin/manage_agents

*****
* Wazuh v2.0 Agent manager.      *
* The following options are available: *
*****
(I)mport key from the server (I).
(Q)uit.
Choose your action: I or Q:
```

6. Press *I* and *Enter* to import a key. Then paste the key that you extracted on the manager:

```
Choose your action: I or Q: I

* Provide the Key generated by the server.
* The best approach is to cut and paste it.
*** OBS: Do not include spaces or new lines.

Paste it here (or '\q' to quit):
MDAxIDE4NWVlNjE1Y2YzYiBhbnkgMGNmMDFiYTM3NmMxY2JjNjU0NDAwYmFhZDY1ZWU1YjcyMGI2NDY3ODhkNGQzMjM5ZTdLNGVmNzQzM
FjMDA4Nw=

Agent information:
ID:013
Name:Example
IP Address:any

Confirm adding it?(y/n): y
Added.
```

7. Press *Q* and *Enter* to exit from *manage_agents*.

8. Edit the wazuh-agent configuration in `/var/ossec/etc/ossec.conf` to add the wazuh-manager IP address. In `<client>` section change the `MANAGE_IP` value to the wazuh-manager address:

```
<client>
    <server-ip>MANAGE_IP</server-ip>
</client>
```

9. Restart the agent:

```
/var/ossec/bin/ossec-control restart
```

Forcing insertion

If you try to add an agent with an IP that another agent is already registered with, `manage_agents` will generate an error. You can use the argument `-d` in order to force the insertion.

Example

We have installed the agent *Server1* with IP 10.0.0.10 and ID 005. For some reason, we had to reinstall the server, so now we must install a new agent and we need to connect it to the manager. In this case, we can use the argument `-d 0` meaning that the previous agent (005) will be removed (with a backup) and a new agent will be created re-using the IP. The new agent will have a new ID:

```
/var/ossec/bin/manage_agents -n Server1 -a 10.10.10.10 -d 0
```

Listing Agents

The binary `/var/ossec/bin/agent_control` allows us to retrieve the list of available agents:

```
$ /var/ossec/bin/agent_control -l
Wazuh agent_control. List of available agents:
ID: 000, Name: vpc-ossec-manager (server), IP: 127.0.0.1, Active/Local
ID: 1040, Name: ip-10-0-0-76, IP: 10.0.0.76, Active
ID: 003, Name: vpc-agent-debian, IP: 10.0.0.121, Active
ID: 005, Name: vpc-agent-ubuntu-public, IP: 10.0.0.126, Active
ID: 006, Name: vpc-agent-windows, IP: 10.0.0.124, Active
ID: 1024, Name: ip-10-0-0-252, IP: 10.0.0.252, Never connected
ID: 1028, Name: vpc-debian-it, IP: any, Never connected
ID: 1030, Name: diamorphine-POC, IP: 10.0.0.59, Active
ID: 015, Name: vpc-agent-centos, IP: 10.0.0.123, Active
ID: 1031, Name: WIN-UENN0U6R5SF, IP: 10.0.0.124, Never connected
ID: 1032, Name: vpc-agent-ubuntu, IP: 10.0.0.122, Active
ID: 1033, Name: vpc-agent-debian8, IP: 10.0.0.128, Active
ID: 1034, Name: vpc-agent-redhat, IP: 10.0.0.127, Active
ID: 1035, Name: vpc-agent-centos7, IP: 10.0.0.101, Never connected
ID: 1041, Name: vpc-agent-centos-public, IP: 10.0.0.125, Active
```

List of agentless devices:

Remove Agents

The binary `/var/ossec/bin/manage_agents` allow us to remove agents.

Confirmation before removing the agent:

```
/var/ossec/bin/manage_agents

*****
* Wazuh v2.0 Agent manager.          *
* The following options are available: *
*****
(A)dd an agent (A).
(E)xtract key for an agent (E).
(L)ist already added agents (L).
(R)emove an agent (R).
(Q)uit.

Choose your action: A,E,L,R or Q: r

Available agents:
ID: 003, Name: DB_Agent, IP: any
Provide the ID of the agent to be removed (or '\q' to quit): 003
Confirm deleting it?(y/n): y
Agent '003' removed.

** You must restart OSSEC for your changes to take effect.

manage_agents: Exiting.
```

No confirmation before removing the agent:

```
$ /var/ossec/bin/manage_agents -r 001

*****
* Wazuh v2.0 Agent manager.          *
* The following options are available: *
*****
(A)dd an agent (A).
(E)xtract key for an agent (E).
(L)ist already added agents (L).
(R)emove an agent (R).
(Q)uit.

Choose your action: A,E,L,R or Q:
Available agents:
ID: 001, Name: new, IP: any
Provide the ID of the agent to be removed (or '\q' to quit): 001
Confirm deleting it?(y/n): y
Agent '001' removed.

** You must restart OSSEC for your changes to take effect.

manage_agents: Exiting.
```


Decoders Syntax

Options

- [decoder](#)
- [parent](#)
- [accumulate](#)
- [program_name](#)
- [prematch](#)
- [regex](#)
- [order](#)
- [fts](#)
- [ftscomment](#)

decoder

The attributes list below defines a decoder.

Default Value	n/a
Allowed values	n/a

The attributes list below defines a decoder.

Attribute	Description
id	The ID of the decoder
name	The name of the decoder
type	The type of the decoder
status	The status of the decoder

parent

It is used to link a subordinate codeblock to his parent.

Default Value	n/a
Allowed values	Any decoder name

accumulate

Allow OSSEC to track events over multiple log messages based on a decoded id.

! Note

Requires a regex populating the id field.

Default Value	n/a
Allowed values	n/a

program_name

It defines the name of the program with which the decoder is associated.

Default Value	n/a
Allowed values	Any sregex expression

prematch

It attempts to find a match within the log for the string defined.

Default Value	n/a
Allowed values	Any sregex expression

regex

Default Value	n/a
Allowed values	Any regex expression

order

It defines what the parenthesis groups contain and the order in which they were received.

Default Value	n/a	
Static fields	srcuser	Extracts the source username
	dstuser	Extracts the destination (target) username
	user	An alias to dstuser (only one of the two can be used)
	srcip	Source ip
	dstip	Destination ip
	srcport	Source port
	dstport	Destination port
	protocol	Protocol
	id	Event id
	url	Url of the event
	action	Event action (deny, drop, accept, etc)
	status	Event status (success, failure, etc)
Dynamic fields	extra_data	Any extra data
	Any string not included in the previous list	

fts

It is used to designate a decoder as one in which the first time it matches the administrator would like to be alerted.

Default Value	n/a	
Allowed values	location	Where the log came from
	srcuser	Extracts the source username
	dstuser	Extracts the destination (target) username
	user	An alias to dstuser (only one of the two can be used)
	srcip	Source ip
	dstip	Destination ip
	srcport	Source port
	extra_data	Any extra data
	any	Any string not included in the previous list
	all	All strings
	none	No strings
	empty	Empty string

dstport	Destination port
protocol	Protocol
id	Event id
url	Url of the event
action	Event action (deny, drop, accept, etc)
status	Event status (success, failure, etc)
extra_data	Any extra data

ftscomment

It adds a comment to a decoder when `<fts>` tag is used.

Default Value	n/a
Allowed values	Any string

Rules Syntax

Available options

- [rule](#)
- [match](#)
- [regex](#)
- [decoded_as](#)
- [category](#)
- [field](#)
- [srcip](#)
- [dstip](#)
- [extra_data](#)
- [user](#)
- [program_name](#)
- [hostname](#)
- [time](#)
- [weekday](#)
- [id](#)
- [url](#)
- [if_sid](#)
- [if_group](#)
- [if_level](#)
- [if_matched_sid](#)
- [if_matched_group](#)
- [same_id](#)
- [same_source_ip](#)
- [same_source_port](#)
- [same_dst_port](#)
- [same_location](#)
- [same_user](#)
- [description](#)
- [list](#)
- [info](#)
- [options](#)
- [check_diff](#)
- [group](#)

rule

level	Definition	Specifies the level of the rule. Alerts and responses use this value.
	Allowed values	0 to 16
id	Definition	Specifies the ID of the rule.
	Allowed values	Any number from 1 to 9999
maxsize	Definition	Specifies the maximum size of the event.

	Allowed values	from 1 to 99999
frequency	Definition	Number of times the rule must have matched before firing. Triggers when 2 more than this setting.
	Allowed values	Any number from 1 to 999
timeframe	Definition	The timeframe in seconds. This option is intended to be used with the frequency option.
	Allowed values	Any number from 1 to 999
ignore	Definition	The time (in seconds) to ignore this rule after firing it (to avoid floods).
	Allowed values	Any number from 1 to 999
overwrite	Definition	Used to supercede an OSSEC rule with local changes.
	Allowed values	yes

match

Any string to match against the log event.

Default Value	n/a
Allowed values	Any sregex expression

regex

Any regex to match against the log event.

Default Value	n/a
Allowed values	Any regex expression

decoded_as

Default Value	n/a
Allowed values	Any decoder name

category

The decoded category to match: ids, syslog, firewall, web-log, squid or windows.

Default Value	n/a
Allowed values	Any category

field

Any regex to be compared to a field extracted by the decoder.

name	Specifies the name of the field extracted by the decoder.
-------------	---

srcip

Any IP address or CIDR block to be compared to an IP decoded as srcip. Use "!" to negate it.

Default Value	n/a
Allowed values	Any srcip

dstip

Any IP address or CIDR block to be compared to an IP decoded as dstip. Use "!" to negate it.

Default Value	n/a
Allowed values	Any dstip

extra_data

Any string that is decoded into the extra_data field.

Default Value	n/a
Allowed values	Any string.

user

Any username (decoded as the username).

Default Value	n/a
Allowed values	Any sregex expression

program_name

Program name is decoded from syslog process name.

Default Value	n/a
Allowed values	Any sregex expression

hostname

Any hostname (decoded as the syslog hostname) or log file.

Default Value	n/a
Allowed values	Any sregex expression

time

Time that the event was generated.

Default Value	n/a
Allowed values	Any time range (hh:mm-hh:mm)

weekday

Week day that the event was generated.

Default Value	n/a
Allowed values	monday - sunday, weekdays, weekends

id

Any ID (decoded as the ID).

Default Value	n/a
Allowed values	Any sregex expression

url

Any URL (decoded as the URL).

Default Value	n/a
Allowed values	Any sregex expression

if_sid

Matches if the ID has matched.

Default Value	n/a
Allowed values	Any rule id

if_group

Matches if the group has matched before.

Default Value	n/a
Allowed values	Any Group

if_level

Matches if the level has matched before.

Default Value	n/a
Allowed values	Any level from 1 to 16

if_matched_sid

Matches if an alert of the defined ID has been triggered in a set number of seconds.

This option is used in conjunction with frequency and timeframe.

Default Value	n/a
Allowed values	Any rule id

Note

Rules at level 0 are discarded immediately and will not be used with the if_matched_rules. The level must be at least 1, but the <no_log> option can be added to the rule to make sure it does not get logged.

if_matched_group

Matches if an alert of the defined group has been triggered in a set number of seconds.

This option is used in conjunction with frequency and timeframe.

Default Value	n/a
Allowed values	Any Group

same_id

Specifies that the decoded id must be the same. This option is used in conjunction with frequency and timeframe.

Default Value	n/a
Allowed values	n/a

same_source_ip

Specifies that the decoded source ip must be the same. This option is used in conjunction with frequency and timeframe.

Default Value	n/a
Allowed values	n/a

same_source_port

Specifies that the decoded source port must be the same. This option is used in conjunction with frequency and timeframe.

Default Value	n/a
Allowed values	n/a

same_dst_port

Specifies that the decoded destination port must be the same. This option is used in conjunction with frequency and timeframe.

Default Value	n/a
Allowed values	n/a

same_location

Specifies that the location must be the same. This option is used in conjunction with frequency and timeframe.

Default Value	n/a
Allowed values	n/a

same_user

Specifies that the decoded user must be the same. This option is used in conjunction with frequency and timeframe.

Default Value	n/a
Allowed values	n/a

description

Rule description.

Default Value	n/a
Allowed values	Any string

list

Preform a CDB lookup using an ossec list. This is a fast on disk database which will always find keys within two seeks of the file.

Default Value	n/a
Allowed values	Path to the CDB file to be used for lookup from the OSSEC directory. Must also be included in the ossec.conf file.

Attribute	Description	
field	key in the CDB: srcip, srcport, dstip, dstport, extra_data, user, url, id, hostname, program_name, status, action, dynamic field.	
lookup	match_key	key to search within the cdb and will match if they key is present. Default.
	not_match_key	key to search and will match if it is not present in the database.

	match_key_value	searched for in the cdb. It will be compared with regex from attribute check_value.
	address_match_key	IP and the key to search within the cdb and will match if they key is present.
	not_address_match_key	IP the key to search and will match if it IS NOT present in the database
	address_match_key_value	IP to search in the cdb. It will be compared with regex from attribute check_value.
check_value	regex for matching on the value pulled out of the cdb when using types: address_match_key_value, match_key_value	

info

Extra information may be added through the following attributes:

Default Value	n/a
Allowed values	Any string

Attribute	Allowed values	Description
type	text	This is the default when no type is selected. Additional information about the alert/event.
	link	Link to more information about the alert/event.
	cve	The CVE Number related to this alert/event.
	ovsdb	The ovsdb id related to this alert/event.

options

Additional rule options

Attribute	Description
alert_by_email	Always alert by email.
no_email_alert	Never alert by email.
no_log	Do not log this alert.

check_diff

Used to determine when the output of a command changes.

Default Value	n/a
Allowed values	n/a

group

Add additional groups to the alert. Groups are optional tags added to alerts.

They can be used by other rules by using if_group or if_matched_group, or by alert parsing tools to categorize alerts.

Default Value	n/a
Allowed values	Any String

Regular Expression Syntax

There are two types of regular expressions: regex (*OS_Regex*) and sregex (*OS_Match*).

Regex (OS_Regex) syntax

This is a fast and simple library for regular expressions in C.

This library is designed to be simple while still supporting the most common regular expressions.

Supported expressions

Expressions	Valid characters
\w	A-Z, a-z, 0-9, ‘-‘, ‘@‘ characters
\d	0-9 character
\s	Spaces “ ”
\t	Tabs
\p	(*)+,-.;<=>?[]!"#\$%& {}()
\W	Anything not w
\D	Anything not d
\S	Anything not s
.	Anything

Modifiers

Expressions	Actions
+	To match one or more times
*	To match zero or more times

Special characters

Expressions	Actions
^	To specify the beginning of the text
\$	To specify the end of the text
	To create a logical or between multiple patterns

Characters escaping

To utilize the following characters they must be escaped with:

\$	()	\	
\\$	\(\)	\\	\

Sregex (OS_Match) syntax

This is faster than OS_Regex, but only supports simple string matching and the following special characters.

Special characters

Expressions	Actions

Expressions	Actions
^	To specify the beginning of the text
\$	To specify the end of the text
	To create a logic: or, between multiple patterns
!	To negate the expression

active-response

XML section name

```
<active-response>
</active-response>
```

In the active response configuration section, you bind an existing command to one or more rules or rule types and specify additional criteria for when to actually execute the command. It is possible to have as many responses as needed, but each must be in their own separate `<active-response>` section.

Options

- [disabled](#)
- [command](#)
- [location](#)
- [agent_id](#)
- [level](#)
- [rules_group](#)
- [rules_id](#)
- [timeout](#)
- [repeated_offenders](#)

disabled

This is a special-case option, in that it occurs alone in its own active-response section for the sole purpose of enabling or disabling the active response facility in Wazuh. In the absence of a section like this, active response is by default enabled on Unix-like systems, and disabled on Windows systems.

Setting it to `yes` on an agent will disable active-response for that agent only, while setting it in the manager's `ossec.conf` file will disable active-response on the manager and all agents.

Note

This option is available on server, local, and agent installations.

Default value	<code>no</code> for Unix-like systems and <code>yes</code> for Windows systems
Allowed values	The options accepted are <code>yes</code> and <code>no</code>

command

This is used to link the response to the command.

Default value	n/a
Allowed values	Any defined active response command name

location

This indicates on which system(s) the command should be executed.

Default value	n/a	
Allowed values	local	This runs the command on the agent that generated the event.
	server	This runs the command on the Wazuh manager.

	defined-agent	This runs the command on a specific agent identified by agent_id
	all	This runs the command on the Wazuh manager and on all agents. Use with caution.

Example for `defined-agent`:

If the application that interfaces with your edge firewall runs on one of your agents, you might have a firewall-block-edge command that runs a command on that agent to blacklists an offending IP on the edge firewall.

agent_id

The ID of the agent to execute the active response command (used when defined-agent is set).

Default value	n/a
Allowed values	Any agent id number, as long as defined-agent has been specified as the location.

level

This defines a minimum severity level required for the command to be executed.

Default value	n/a
Allowed values	Any level from 1 to 16

rules_group

This requires that a rule must belong to one or more rule groups for the command to be executed.

Default value	n/a
Allowed values	Any rule group is allowed. Multiple groups should be separated with a pipe character (" ").

Note

Observe that all groups must be finished by comma.

rules_id

This limits command execution to only when one or more listed rules fire.

Default value	n/a
Allowed values	Any rule identification. Multiple IDs can be specified if separated by a comma.

timeout

This specifies how long in seconds until the reverse command is executed. When `repeated_offenders` is used, `timeout` only applies to the first offense.

Default value	n/a
Allowed values	A positive number (seconds)

repeated_offenders

This is a comma-separated list of increasing timeouts in minutes for repeat offenders. There can be a maximum of 5 entries. This must be configured directly in the **ossec.conf** file of the agent, even when using a manager/agent setup with centralized configuration of other settings via **agent.conf**.

Default value	n/a
Allowed values	A positive number (minutes)

Example of configuration

```
<active-response>
  <disabled>no</disabled>
  <command>host-deny</command>
  <location>defined-agent</location>
  <agent-id>032</agent-id>
  <level>10</level>
  <rules_group>sshd,|pci_dss_11.4,</rules_group>
  <timeout>1</timeout>
  <repeated_offenders>1,5,10</repeated_offenders>
</active-response>
```

agentless

XML section name

```
<agentless>
</agentless>
```

Agentless monitoring allows you to run integrity checking on systems without an agent installed.

Options

- [type](#)
- [frequency](#)
- [host](#)
- [state](#)
- [arguments](#)

type

Default value	n/a	
Allowed values	ssh_integrity_check_bsd	Require a list of directories in <arguments>. Wazuh will file integrity scan the files in those directories.
	ssh_integrity_check_linux	System will alert if they have changed.
	ssh_generic_diff	Supply an <arguments> value that consists of a set of commands to run. Their output is then processed, looking for changes or rule matches.
	ssh_pixconfig_diff	Specifically for checking if the config of a Cisco PIX/router changes. No <arguments> required

frequency

This controls the number of seconds between each check of the agentless device.

Default value	n/a
Allowed values	An integer in seconds

host

This defines the username and the name of the agentless host.

Default value	n/a
Allowed values	Any username and host (<code>username@hostname</code>)

state

This determines whether the check type is periodic or periodic_diff.

Default value	n/a	
Allowed values	periodic	Output from each check is analyzed with the Wazuh ruleset as if a monitored log.

periodic_diff

Output from each agentless check is compared to the output of the previous run.

Changes are alerted on, similar to file integrity monitoring.

arguments

This defines the arguments passed to the agentless check

Default value	n/a
Allowed values	This is a space-delimited list of files or directories to be monitored.

Example of configuration

```
<agentless>
  <type>ssh_integrity_check_linux</type>
  <frequency>300</frequency>
  <host>admin@192.168.1.108</host>
  <state>periodic_diff</state>
  <arguments>/etc /usr/bin /usr/sbin</arguments>
</agentless>
```

alerts

XML section name

```
<alerts>  
</alerts>
```

Configure here the minimum alert levels for logging or sending alerts. You can also enable or disable the geolocation feature.

Options

- [log_alert_level](#)
- [email_alert_level](#)
- [use_geoip](#)

log_alert_level

This is the minimum severity level for alerts to be stored to alerts.log and/or alerts.json.

Default value	3
Allowed values	Any level from 1 to 16

email_alert_level

This is the minimum severity level for an alert to generate an email notification.

Warning

This is the minimum level for an alert to trigger an email. This overrides granular email alert levels. Setting this to 10 would prevent the sending of emails for alerts with levels lower than 10 even when there are settings in the granular email configuration referencing levels lower than 10. Individual rules can override this with the *alert_by_email* option, which forces an email alert regardless of global or granular alert level thresholds.

Default value	12
Allowed values	Any level from 1 to 16

use_geoip

Enable or disable GeoIP lookups.

Default value	no
Allowed values	The options are yes or no .

Default configuration

```
<alerts>  
  <log_alert_level>3</log_alert_level>  
  <email_alert_level>12</email_alert_level>  
</alerts>
```


auth

XML section name

```
<auth>  
</auth>
```

This section has options for the registering service.

! New in version 2.1.

Options

- [disabled](#)
- [port](#)
- [use_source_ip](#)
- [force_insert](#)
- [force_time](#)
- [purge](#)
- [use_password](#)
- [ssl_agent_ca](#)
- [ssl_verify_host](#)
- [ssl_manager_cert](#)
- [ssl_manager_key](#)
- [ssl_auto_negotiate](#)

disabled

Disables the execution of the Auth daemon.

Default value	no
Allowed values	<ul style="list-style-type: none">• yes• no

port

TCP port number to listen to connections.

Default value	1515
Allowed values	0 - 65535

use_source_ip

Use client's source IP address instead of "any" to add agent.

Default value	no
Allowed values	<ul style="list-style-type: none">• yes• no

force_insert

Force insertion: remove old agent with same name or IP.

Default value	no
Allowed values	<ul style="list-style-type: none">• yes• no

force_time

When forcing to remove old agents with same name or IP, this option specifies that the deletion will be performed only if the agent's keepalive has more than a number of seconds.

Default value	0
Allowed values	<ul style="list-style-type: none">• Positive number• 0

Value `0` means to force always.

purge

Delete definitely agents when removing.

Default value	no
Allowed values	<ul style="list-style-type: none">• yes• no

When set to `no` removed agents will remain in the client keys file, marked as removed. However, when set to `yes`, client keys file will be purged.

use_password

Enable shared password authentication.

Default value	no
Allowed values	<ul style="list-style-type: none">• yes• no

When enabled, the shared password will be read from file at `/var/ossec/etc/authd.pass`.

If this file does not exist, a **random password** will be generated.

ssl_agent_ca

Full path to CA certificate used to verify clients.

Allowed values	A full path
-----------------------	-------------

ssl_verify_host

When CA certificate is specified, this option enables source host verification. This means that the client source IP will be validated using the *Common Name* field.

Default value	no
----------------------	----

Allowed values	<ul style="list-style-type: none"> • yes • no
-----------------------	---

ssl_manager_cert

Full path to server SSL certificate.

Default value	/var/ossec/etc/sslmanager.cert
Allowed values	A full path

ssl_manager_key

Full path to server SSL key.

Default value	/var/ossec/etc/sslmanager.key
Allowed values	A full path

ssl_auto_negotiate

Auto select SSL/TLS method.

Default value	no
Allowed values	<ul style="list-style-type: none"> • yes • no

By default only TLS v1.2 is allowed. When set to **yes** the system will negotiate the most secure common method with the client.

In older systems, where the **manager does not support TLS v1.2**, this option will be enabled automatically.

Default configuration

```

<auth>
  <disabled>no</disabled>
  <port>1515</port>
  <use_source_ip>no</use_source_ip>
  <force_insert>no</force_insert>
  <force_time>0</force_time>
  <purge>no</purge>
  <use_password>no</use_password>
  <!-- <ssl_agent_ca></ssl_agent_ca> -->
  <ssl_verify_host>no</ssl_verify_host>
  <ssl_manager_cert>/var/ossec/etc/sslmanager.cert</ssl_manager_cert>
  <ssl_manager_key>/var/ossec/etc/sslmanager.key</ssl_manager_key>
  <ssl_auto_negotiate>no</ssl_auto_negotiate>
</auth>

```

client

XML section name

```
<client>  
</client>
```

Configure the connection parameters related to connecting to the manager.

Options

- [server-ip](#)
- [server-hostname](#)
- [port](#)
- [protocol](#)
- [config-profile](#)
- [notify_time](#)
- [time-reconnect](#)
- [local_ip](#)
- [disable-active-response](#)

server-ip

Specify the IP address of the Wazuh manager.

Default value	n/a
Allowed values	Any valid IP address is allowed.

server-hostname

Specify the hostname of the Wazuh manager.

Default value	n/a
Allowed values	Any resolvable hostname is allowed.

⚠ Warning

This parameter is incompatible with [server-ip](#).

port

Specify the port on the manager to send events to. This must match the associated listening port configured on the Wazuh manager.

Default value	1514
Allowed values	Any port number from 1 to 65535 is allowed.

protocol

Specifies the protocol to use when connecting to manager.

Default value	udp
Allowed values	udp, tcp

config-profile

Specify the agent.conf profile(s) to be used by the agent.

Default value	n/a
Allowed values	Multiple profiles can be included, separated by a comma and a space.

notify_time

Specify the time in seconds between agent checkins to the manager. More frequent checkins speed up dissemination of an updated agent.conf file to agents, but also could put undue load on the manager if there are a large number of agents.

Default value	600
Allowed values	A positive number (seconds)

time-reconnect

This is the time in seconds until a reconnection attempt. This should be set to a higher number than notify_time. For example, a notify_time time of 60 combined with a time-reconnect of 300 would mean that agents will cause the agent to attempt to check in once per minute, but if a checkin attempt fails to get a response from the manager, the agent will wait five minutes before trying again. Once it again succeeds, checkins will resume their normal one-minute interval.

Default value	1800
Allowed values	A positive number (seconds)

Warning

Notice that the notify_time value uses an underscore while the time-reconnect value uses a dash. This is an unfortunate legacy naming inconsistency, and is easy to mix up.

local_ip

When the agent has multiple network interfaces, this parameter specifies which IP address will communicate with the manager from.

Default value	n/a
Allowed values	Any valid IP address is allowed.

disable-active-response

Deprecated: This is an obsolete method to disable active response.

Warning

The recommended way is using the [active-response](#) section.

Default value	no
Allowed values	The options accepted are yes and no

Example of configuration

```
<client>
  <server-ip>192.168.1.100</server-ip>
  <config-profile>webserver, debian8</config-profile>
  <protocol>tcp</protocol>
  <notify_time>300</notify_time>
  <time-reconnect>900</time-reconnect>
</client>
```


client_buffer

XML section name

```
<client_buffer>
</client_buffer>
```

Configure the agent bucket parameters in order to avoid events flooding.

Options

- [disable](#)
- [length](#)
- [events_per_second](#)

disable

This parameter allows to disable the Agent Buffer and send events to the manager without any congestion control.

Default value	no
Allowed values	The options accepted are yes and no .

⚠ Warning

Disabling this functionality in large environments, agents may collapse the manager and the network.

length

The capacity of Agent Buffer in number of events.

Default value	5000
Allowed values	Any number between 1 and 100000.

events_per_second

Specifies the number of events sent to the manager per second.

Default value	500
Allowed values	Any number between 1 and 1000.

Default configuration

```
<client_buffer>
  <!-- Agent buffer options -->
  <disable>no</disable>
  <length>5000</length>
  <events_per_second>500</events_per_second>
</client_buffer>
```


command

XML section name

```
<command>
</command>
```

In a command configuration section, you define a command to be used by one or more active responses. It is possible to have as many commands as needed, but each one must be in their own separate <command> section.

Options

- [name](#)
- [executable](#)
- [expect](#)
- [timeout_allowed](#)

name

This field specifies the name of the command which is called in the [active-response](#) section.

Default value	n/a
Allowed values	Any name
use	Required

executable

This must be a file (with the execute permission set) inside `/var/ossec/active-response/bin`. You don't need to provide the path.

Default value	n/a
Allowed values	Any file name
use	Required

expect

This is a list of zero or more names of extracted fields that are to be passed as parameters to the command. If any of the listed fields were not extracted in a certain instance, those field values would be passed as a dash (`-`) instead of as no value at all. A good example is the firewall-block command which expects the `srcip` field so it knows which IP to block. Multiple expected field names are comma separated.

Default value	n/a
Allowed values	Names of extracted fields, like <code>srcip</code> or <code>username</code> , separated by commas if there is more than one.
use	Required

Note

You can specify no fields by using `<expect></expect>`. That is the valid setting when no options need to be passed to the active-response command.

timeout_allowed

If yes, this indicates that the command is stateful, and will be called again in a certain length of time and instructed to undo its original action.

Default value	yes
Allowed values	yes/no

Example of configuration

```
<!-- For Unix systems -->
<command>
  <name>host-deny</name>
  <executable>host-deny.sh</executable>
  <expect>srcip</expect>
  <timeout_allowed>yes</timeout_allowed>
</command>

<!-- For Windows systems -->
<command>
  <name>win_route-null</name>
  <executable>route-null.cmd</executable>
  <expect>srcip</expect>
  <timeout_allowed>yes</timeout_allowed>
</command>
```

database_output

XML section name

```
<database_output>  
</database_output>
```



MySQL and PostgreSQL database output is supported. It is configured with the options below.

Options

- [hostname](#)
- [username](#)
- [password](#)
- [database](#)
- [type](#)

hostname

Specify the IP address of the database server.

Default value	n/a
Allowed values	Any valid IP address

username

Specify the username to access the database.

Default value	n/a
Allowed values	Any valid username

password

Specify the password to access the database.

Default value	n/a
Allowed values	Any password

database

Specify the name of the database in which to store the alerts.

Default value	n/a
Allowed values	Database name

type

Type of database

Default value	n/a
Allowed values	mysql/postgresql

Note

Wazuh must be compiled with the database type that is to be used.

Example of configuration

```
<database_output>
  <hostname>192.168.1.122</hostname>
  <username>MySQLAdmin</username>
  <password>secret1234</password>
  <database>Alerts_DB</database>
  <type>mysql</type>
</database_output>
```

email_alerts

XML section name

```
<email_alerts>
</email_alerts>
```

This extends the email options configured in the `<global>` section.

! Note

Global email configuration is necessary to use granular email options.

Options

- [email_to](#)
- [level](#)
- [group](#)
- [event_location](#)
- [format](#)
- [rule_id](#)
- [do_not_delay](#)
- [do_not_group](#)

email_to

This specifies a single email address to which to send email alerts. If you want to send alerts to multiple addresses, each address must be listed in a separate `<email_to>` section. Lists are not allowed.

Default value	n/a
Allowed values	Any valid email address is allowed.

level

This is the minimum alert severity level for which emails will be sent.

! Note

The `level` option should be set at or above the `email_alert_level` in the `<alerts>` section of the configuration.

Default value	n/a
Allowed values	Any alert level 0 to 16 is allowed.

group

This limits the sending of emails to only when rules are tripped that belongs to one of the listed groups.

Default value	n/a
Allowed values	Any rule group is allowed. Multiple groups should be separated with a pipe character (" ").

! Note

Observe that all groups must be finished by comma.

event_location

The alert must match this event location to be forwarded. Do not specify this option repeatedly, as only the last instance would be used.

Default value	n/a
Allowed values	Any single agent name, hostname, IP address, or log file is allowed

format

This specifies the email format.

Default value	full	
Allowed values	full	Send normal emails.
	sms	Use a compact format more suitable for SMS.

rule_id

This limits the sending of emails to only when rules are tripped that have one of the listed rule IDs.

Default value	n/a
Allowed values	One or more rule IDs can be used here, separated by a comma and a space (", ").

do_not_delay

This causes email alerts to be sent right away, rather than to be delayed for the purpose of batching multiple alerts together.

Default value	n/a
Allowed values	XML tag with no value

do_not_group

This disables grouping of multiple alerts into the same email.

Default value	n/a
Allowed values	XML tag with no value

Warning

Notice that **do_not_delay** and **do_not_group** are special empty-element XML tags, so they stand alone, not having a starting and ending version of the tag. This is indicated by the tag name containing "/" at the end of the name.

Example of configuration

```
<email_alerts>
  <email_to>recipient@example.wazuh.com</email_to>
  <email_to>recipient2@example.wazuh.com</email_to>
  <level>12</level>
  <group>sshd,</group>
  <do_not_delay/>
</email_alerts>
```

global

XML section name

```
<global>
</global>
```

Global configuration generally applies to features that affect the system as a whole, rather than just one component.

Options

- [alerts_log](#)
- [email_notification](#)
- [email_to](#)
- [email_from](#)
- [email_reply_to](#)
- [smtp_server](#)
- [email_maxperhour](#)
- [email_idsname](#)
- [custom_alert_output](#)
- [stats](#)
- [logall](#)
- [memory_size](#)
- [white_list](#)
- [host_information](#)
- [jsonout_output](#)
- [prelude_output](#)
- [picviz_output](#)
- [picviz_socket](#)
- [zeromq_output](#)
- [zeromq_uri](#)
- [geoip_db_path](#)

alerts_log

This enable or disable writing alerts to [/var/ossec/logs/alerts/alerts.log](#).

Default value	yes
Allowed values	yes, no

Warning

Disabling JSON and plain text formated alerts simultaneously is not compatible with the integrator, syslog client and email features.

email_notification

This enable or disables email alerting.

Default value	no
Allowed values	yes, no

email_to

This specifies the email recipient for alerts.

! Note

To use granular email configurations, a base configuration is necessary in the section.

Default value	n/a
Allowed values	Any valid email address

Use this section repeatedly for multiple email addresses, once per addresses.

email_from

This controls the “source” address in email alerts.

Default value	n/a
Allowed values	Any valid email address

email_reply_to

This controls the “reply-to” address in email alerts.

Default value	n/a
Allowed values	Any valid email address

smtp_server

This controls what SMTP server to forward email alerts to for delivery.

Default value	n/a
Allowed values	<ul style="list-style-type: none">• Valid hostname or IP address.• Full path to a sendmail-like executable.

email_maxperhour

This specifies the maximum number of emails to be sent per hour. All emails in excess of this setting will be queued for later distribution.

! Note

At the end of the hour any queued emails will be sent together in one email. This is true whether mail grouping is enabled or disabled.

Default value	12
Allowed values	Any number from 1 to 9999

email_idsname

The name will be added to the email headers with the specified value.

Default value	n/a
Allowed values	Any name

custom_alert_output

This specifies the format of alerts written to `alerts.log`. Check the allowed values for `custom_alert_output` in the following table:

Variable name	Description
\$TIMESTAMP	The time the event was processed by OSSEC.
\$FTELL	Unknown
\$RULEALERT	Unknown
\$HOSTNAME	Hostname of the system generating the event.
\$LOCATION	The file the log messages was saved to.
\$RULEID	The rule id of the alert.
\$RULELEVEL	The rule level of the alert.
\$RULECOMMENT	Unknown
\$SRCIP	The source IP specified in the log message.
\$DSTUSER	The destination user specified in the log message.
\$FULLLOG	The original log message.
\$RULEGROUP	The groups containing the rule.

stats

This controls the severity level assigned to events generated by statistical analysis.

Default value	8
Allowed values	Any level from 0 to 16

logall

This controls whether or not to store all events received even when they do not trip a rule. This results in output to `/var/ossec/logs/archives/archives.log`

Default value	no
Allowed values	yes or no

memory_size

This sets the memory size for the event correlation engine.

Default value	1024
Allowed values	Any size from 16 to 5096

white_list

This is a list of IP addresses that should never be blocked with active response. Repeat this option for multiple IPs, one IP per line. This option is only valid in server and local installs.

Default value	n/a
Allowed values	Any IP address or netblock

host_information

This controls the severity level for events generated by the host change monitor.

Default value	8
---------------	---

Allowed values	Can be used any level from 0 to 16
-----------------------	------------------------------------

jsonout_output

This enables/disables writing of JSON-formated alerts to `/var/ossec/logs/alerts/alerts.json`. This will include the same events that would be sent to `alerts.log`, but in JSON format.

Default value	no
Allowed values	The options allowed are yes or no .

prelude_output

Enables or disables Prelude output.

Default value	yes
Allowed values	The options allowed are yes or no .

picviz_output

Enable PicViz output.

Default value	n/a
Allowed values	yes

picviz_socket

This is the full path of the socket that Wazuh will write alerts/events to for PicViz to read.

Default value	n/a
Allowed values	file and path that Wazuh will create and feed events to

zeromq_output

Enable ZeroMQ output.

Default value	n/a
Allowed values	The options allowed are yes or no .

zeromq_uri

This is the ZeroMQ URI that the publisher socket will bind to.

Default value	n/a
Allowed values	This URI format is defined by the ZeroMQ project.

For example, this will listen for ZeroMQ subscribers on IP address 127.0.0.1:11111.

```
<zeromq_uri>tcp://localhost:11111/</zeromq_uri>
```

This will listen on port 21212 for ZeroMQ subscribers, binding to the IP address assigned to eth0.

```
<zeromq_uri>tcp://eth0:21212/</zeromq_uri>
```

This will listen for zeromq on the Unix Domain socket `/alerts-zmq`.

```
<zeromq_uri>ipc:///alerts-zmq</zeromq_uri>
```

geoip_db_path

This is the full path to the MaxMind GeoIP IPv4 database file.

Default value	n/a
Allowed values	Path to the GeoIP IPv4 database file location

Example

```
<geoip_db_path>/etc/GeoLiteCity.dat</geoip_db_path>
```

Default configuration

```
<global>
  <jsonout_output>yes</jsonout_output>
  <alerts_log>yes</alerts_log>
  <logall>no</logall>
  <logall_json>no</logall_json>
  <email_notification>no</email_notification>
  <smtp_server>smtp.example.wazuh.com</smtp_server>
  <email_from>ossecm@example.wazuh.com</email_from>
  <email_to>recipient@example.wazuh.com</email_to>
  <email_maxperhour>12</email_maxperhour>
</global>
```

integration

XML section name

```
<integration>
</integration>
```

This configures the manager to connect Wazuh to external APIs and alerting tools such as Slack and PagerDuty.

Options

- [name](#)
- [hook_url](#)
- [api_key](#)
- [level](#)
- [rule_id](#)
- [group](#)
- [event_location](#)

name

This indicates the type of the service to integrate with.

Default value	n/a
Allowed values	slack, pagerdty

hook_url

This is the URL provided by Slack when integration is enabled on the Slack side. This is **mandatory for Slack**.

Default value	n/a
Allowed values	Slack URL

api_key

This is the key that you would have retrieved from the PagerDuty API. This is **mandatory for PagerDuty**.

! Note

You must restart Wazuh after changing this configuration.

Default value	n/a
Allowed values	PagerDuty Api key

Optional filters

level

This filter alerts by rule level. It will push only alerts with the specified level or above.

Default value	n/a
Allowed values	Any alert level from 0 to 16

rule_id

This filters alerts by rule ID.

Default value	n/a
Allowed values	Comma-separated rule IDs

group

This filters alerts by rule group. For the VirusTotal integration, only rules from the *syscheck* group are available.

Default value	n/a
Allowed values	Any rule group or comma-separated rule groups.

! Note

Observe that all groups must be finished by comma.

event_location

This filters alerts by the location of where the event originated. [OS_Regex Syntax](#)

Default value	n/a
Allowed values	Any single agent name, hostname, ip address, or log file.

Example of configuration

```
<integration>
  <name>slack</name>
  <hook_url>https://hooks.slack.com/services/T000/B000/XXXXXX</hook_url>
  <level>10</level>
  <group>multiple_drops, |authentication_failures,</group>
</integration>
```

labels

XML section name

```
<labels>
</labels>
```

This section permits include extra information about agents in their alerts. When email notifications are enabled, labeled data is sent within alerts by email without any additional configuration.

Options

- [label](#)

label

This option specifies the information that appears in alerts. It has the format `key:value`, which means that it is necessary to include the attribute `key` to work properly. Thinking in the JSON alerts, it is possible to nest labels splitting this attribute by dots.

Atributes:

key	The title that will describe the information of the label.	
	Allowed value	string
hidden	Permits not to show the specified label by default.	
	Allowed value	yes

Note

In `internal_options.conf` there is the possibility of show hidden labels in the alerts.

Example of configuration

```
<labels>
  <label key="aws.instance-id">i-052a1838c</label>
  <label key="aws.sec-group">sg-1103</label>
  <label key="network.ip">172.17.0.0</label>
  <label key="network.mac">02:42:ac:11:00:02</label>
  <label key="installation" hidden="yes">January 1st, 2017</label>
</labels>
```


localfile

XML section name

```
<localfile>
</localfile>
```

This configuration section is used to specify the collection of log data from files, Windows events, and from output of commands.

Options

- [location](#)
- [command](#)
- [alias](#)
- [frequency](#)
- [only-future-events](#)
- [query](#)
- [log_format](#)

location

Specify the location of a log or wildcarded group of logs to be read. `strftime` format strings may be used for log file names.

For instance, a log file named `file.log-2017-01-22` could be referenced with `file.log-%Y-%m-%d` (assuming today is Jan 22nd, 2017).

Wildcards may be used on non-Windows systems. When wildcards are used, the log files must exist at the time `ossec-logcollector` is started. It will not automatically begin monitoring new log files.

Note that `strftime` format strings and wildcards cannot be used on the same entry.

Default value	n/a
Allowed values	Any log file

command

A command to be run. All output from this command will be read as one or more log messages depending on whether command or full_command is used.

Default value	n/a
Allowed values	any command line, optionally including arguments

alias

This is an alias to identify the command. This will replace the command in the log message.

Default value	n/a
Allowed values	any string

For example `<alias>usb-check</alias>` would replace:

```
ossec: output: 'reg QUERY HKLM\SYSTEM\CurrentControlSet\Enum\USBSTOR':
```

with:

```
ossec: output: 'usb-check':
```

frequency

The minimum time in seconds between command runs. The command will probably not run every `frequency` seconds exactly, but the time between runs will not be shorter than this setting. This is used with `command` and `full_command`.

Default value	n/a
Allowed values	any positive number of seconds

only-future-events

This is for use only with the `eventchannel` log format. By default, when Wazuh starts, it will read all log content from a given Windows Event Channel since Wazuh was last stopped. Set this option to `yes` to override this behavior if desired. Then Wazuh would only receive events that occur after the Wazuh agent is started.

Default value	n/a
Allowed values	yes or no

query

This is for use only with the `eventchannel` log format. It is possible to specify an XPATH query following the event schema in order to filter the events that Wazuh will process.

Default value	n/a
Allowed values	Any XPATH query following the event schema

For example, the following configuration will only process events with an ID of 7040:

```
<localfile>
  <location>System</location>
  <log_format>eventchannel</log_format>
  <query>Event/System[EventID=7040]</query>
</localfile>
```

log_format

This is the format of the log being read.

Note

For most text log files that have one entry per line, you can just use `syslog`.

Default value	syslog	
Allowed values	syslog	This format is for plain text files in a syslog-like format. Also can be used when the logs are single line messages.
	snort-full	This is used for Snort's full-output format.
	snort-fast	This is used for Snort's fast-output format.
	squid	This is used for squid logs.
	iis	This is used for IIS logs.
	eventlog	This is used for the classic Microsoft Windows event log format.

eventchannel	This is used for Microsoft Windows event logs, using the new EventApi. Monitorize: standard "Windows" eventlogs and "Application and Services" logs.
mysql_log	This is used for MySQL logs. It does not support multi-line logs.
postgresql_log	This is used for PostgreSQL logs. It does not support multi-line logs.
nmapg	Used for monitoring files conforming to the grepable output from nmap .
apache	Apache's default log format.
command	Read in arbitrary output from the command (as run by root). Command defined by the command tag. Each line of output will be treated as a separate log.
full_command	Read in arbitrary output from the command (as run by root) Command defined by the command tag. The entire output will be treated as a single log item.
djb-multilog	Read files in the format produced by the multilog service logger in daemontools.
multi-line	Allow applications that log multiple lines per event to be monitored. Require the number of lines to be consistent. multi-line: is followed by the number of lines in each log entry. Each line will be combined with the previous lines until all lines are gathered. There may be multiple timestamps in a finalized event. The format is: <log_format>multi-line: NUMBER</log_format>

⚠ Warning

The eventchannel log format cannot be used on Windows agents older than Vista since they do not produce that kind of log.

⚠ Warning

Agents will ignore command and full_command log sources unless they have logcollector.remote_commands=1 set in their /var/ossec/etc/internal_options.conf or /var/ossec/etc/local_internal_options.conf file. This is a security precaution since it may not be permissible in all environments to allow the Wazuh manager to run arbitrary commands on agents in their root security context.

Example:

Multi-line log message in original log file:

```
Aug 9 14:22:47 hostname log line one
Aug 9 14:22:47 hostname log line two
Aug 9 14:22:47 hostname log line four
Aug 9 14:22:47 hostname log line three
Aug 9 14:22:47 hostname log line five
```

Log message as analyzed by ossec-analysisd:

```
Aug 9 14:22:47 hostname log line one Aug 9 14:22:47 hostname log line two Aug 9 14:22:47 hostname log line
three Aug 9 14:22:47 hostname log line four Aug 9 14:22:47 hostname log line five
```

Example of configuration

Linux configuration:

```
<!-- For monitor log files -->
<localfile>
  <log_format>syslog</log_format>
  <location>/var/log/syslog</location>
  <expect>srcip</expect>
  <timeout_allowed>yes</timeout_allowed>
</command>

<!-- For monitor commands output -->
<localfile>
  <log_format>command</log_format>
  <command>df -P</command>
  <frequency>360</frequency>
</localfile>
```

Windows configuration:

```
<!-- For monitor Windows eventchannel -->
<localfile>
  <location>Security</location>
  <log_format>eventchannel</log_format>
  <only-future-events>yes</only-future-events>
  <query>Event/System[EventID != 5145 and EventID != 5156]</query>
</localfile>
```

logging

XML section name

```
<logging>  
</logging>
```

This section allows to configure internal logs.

Options

- [log_format](#)

log_format

Specifies the log format between JSON output (.json) or plain text (.log). It also can be established both formats at the same time separated by comma.

Default value	plain
Allowed values	<ul style="list-style-type: none">• plain• json• plain, json

Default configuration

```
<!-- Choose between plain or json format (or both) for internal logs -->  
<logging>  
  <log_format>plain</log_format>  
</logging>
```


remote

XML section name

```
<remote>  
</remote>
```

Configuration of manager to listen for events from the agents.

Options

- [connection](#)
- [port](#)
- [protocol](#)
- [allowed-ips](#)
- [deny-ips](#)
- [local_ip](#)
- [ipv6](#)

connection

Specifies a type of incoming connection to accept: secure or syslog.

Default value	secure
Allowed values	secure, syslog

port

Specifies the port to use to listen for events.

Default value	1514 if secure, 514 if syslog
Allowed values	Any port number from 1 to 65535

protocol

Specifies the protocol to use. It is available for secure connections and syslog events.

Default value	udp
Allowed values	udp, tcp

Note

It is not possible to use both protocols simultaneously.

allowed-ips

List of IP addresses that are allowed to send syslog messages to the server (one per line).

Default value	n/a
Allowed values	Any IP address or network

Note

It is necessary to list at least one IP address when using the syslog connection type.

deny-ips

List of IP addresses that are not allowed to send syslog messages to the server (one per line).

Default value	n/a
Allowed values	Any IP address or network

local_ip

Local ip address to use to listen for connections.

Default value	All interfaces
Allowed values	Any internal ip address

ipv6

Local ipv6 address to listen for connections.

Default value	n/a
Allowed values	Any IPv6 address

Example of configuration

```
<remote>
  <connection>syslog</connection>
  <port>514</port>
  <protocol>udp</protocol>
  <allowed_ips>192.168.1.0/24</allowed_ips>
  <local_ip>192.168.1.5</local_ip>
</remote>

<remote>
  <connection>secure</connection>
  <port>1514</port>
  <protocol>udp</protocol>
</remote>
```

reports

XML section name

```
<reports>  
</reports>
```

Configuration options for reporting of alerts.

Options

- [group](#)
- [category](#)
- [rule](#)
- [level](#)
- [location](#)
- [srcip](#)
- [user](#)
- [title](#)
- [email_to](#)
- [showlogs](#)

group

Filter by group/category. It only accepts one group/category.

Default value	n/a
Allowed values	Any group used is allowed.

category

Filter by group/category.

Default value	n/a
Allowed values	Any category used is allowed.

rule

Rule ID to filter for.

Default value	n/a
Allowed values	Any Rule ID in Wazuh Rules is allowed

level

Alert level to filter for. The report will include all levels above and including level specified.

Default value	n/a
Allowed values	Any Alert level from 1 to 16 can be used

location

Filter by the log location or agent name.

Default value	n/a
Allowed values	Any file path, hostname or network is allowed

srcip

Filter by the source ip of the event.

Default value	n/a
Allowed values	Any hostname or network can be used.

user

Filter by the user name. This will match either the srcuser or dstuser.

Default value	n/a
Allowed values	Any username

title

Name of the report. **This is a required field.**

Default value	n/a
Allowed values	Any text

email_to

The email address to send the completed report. **This is a required field.**

Default value	n/a
Allowed values	Any email address

showlogs

Enable or disable the inclusion of logs when creating the report.

Default value	no
Allowed values	yes, no

Example of configuration

```
<reports>
  <group>authentication_failed,</group>
  <srcip>192.168.1.10</srcip>
  <title>Auth_Report</title>
  <email_to>recipient@example.wazuh.com</email_to>
  <showlogs>yes</showlogs>
</reports>
```

rootcheck

XML section name

```
<rootcheck>  
</rootcheck>
```

Configuration options for policy monitoring and anomaly detection.

Options

- [base_directory](#)
- [rootkit_files](#)
- [rootkit_trojans](#)
- [windows_audit](#)
- [system_audit](#)
- [windows_apps](#)
- [windows_malware](#)
- [scanall](#)
- [frequency](#)
- [disabled](#)
- [check_dev](#)
- [check_files](#)
- [check_if](#)
- [check_pids](#)
- [check_policy](#)
- [check_ports](#)
- [check_sys](#)
- [check_trojans](#)
- [check_unixaudit](#)
- [check_winapps](#)
- [check_winmalware](#)
- [skip_nfs](#)

base_directory

The base directory that will be appended to the following options:

`rootkit_files` `rootkit_trojans` `windows_malware` `windows_audit` `windows_apps` `systems_audit`

Default value	/var/ossec
Allowed values	Path to a directory

rootkit_files

Change the location of the rootkit files database.

Default value	/var/ossec/etc/shared/rootkit_files.txt
Allowed values	A file with the rootkit files signatures

rootkit_trojans

Change the location of the rootkit trojans database.

Default value	/var/ossec/etc/shared/rootkit_trojans.txt
Allowed values	A file with the trojans signatures

windows_audit

Specifies the path to a Windows audit definition file.

Default value	n/a
Allowed values	Path to a Windows audit definition file

system_audit

Specifies the path to an audit definition file for Unix-like systems.

Default value	n/a
Allowed values	Audit definition file for Unix-like systems

windows_apps

Specifies the path to a Windows application definition file.

Default value	n/a
Allowed values	Path to a Windows application def. file

windows_malware

Specifies the path to a Windows malware definitions file.

Default value	n/a
Allowed values	Path to a Windows malware definitions file

scanall

Tells rootcheck to scan the entire system. This option may lead to some false positives.

Default value	no
Allowed values	yes, no

frequency

Frequency that the rootcheck is going to be executed (in seconds).

Default value	36000
Allowed values	A positive number (seconds)

disabled

Disables the execution of rootcheck.

Default value	no
Allowed values	yes, no

check_dev

Enable or disable the checking of /dev.

Default value	yes
Allowed values	yes, no

check_files

Enable or disable the checking of files.

Default value	yes
Allowed values	yes, no

check_if

Enable or disable the checking of network interfaces.

Default value	yes
Allowed values	yes, no

check_pids

Enable or disable the checking of process ID's.

Default value	yes
Allowed values	yes, no

check_policy

Enable or disable the checking of policy.

Default value	yes
Allowed values	yes, no

check_ports

Enable or disable the checking of network ports.

Default value	yes
Allowed values	yes, no

check_sys

Enable or disable checking for anomalous file system objects.

Default value	yes
Allowed values	yes, no

check_trojans

Enable or disable checking for trojans.

Default value	yes
Allowed values	yes, no

check_unixaudit

Enable or disable the checking of unixaudit.

Default value	yes
Allowed values	yes, no

check_winapps

Enable or disable the checking of winapps.

Default value	yes
Allowed values	yes, no

check_winaudit

Enable or disable the checking of winaudit.

Default value	1
Allowed values	0 , 1

check_winmalware

Enable or disable checking for Windows malware.

Default value	yes
Allowed values	yes, no

skip_nfs

Enable or disable the scanning of network mounted filesystems (Works on Linux and FreeBSD). Currently, skip_nfs will exclude checking files on CIFS or NFS mounts.

Default value	yes
Allowed values	yes, no

Default Unix configuration

```
<!-- Policy monitoring -->
<rootcheck>
<disabled>no</disabled>
<check_unixaudit>yes</check_unixaudit>
<check_files>yes</check_files>
<check_trojans>yes</check_trojans>
<check_dev>yes</check_dev>
<check_sys>yes</check_sys>
<check_pids>yes</check_pids>
<check_ports>yes</check_ports>
<check_if>yes</check_if>

<!-- Frequency that rootcheck is executed - every 12 hours -->
<frequency>43200</frequency>

<rootkit_files>/var/ossec/etc/shared/rootkit_files.txt</rootkit_files>
<rootkit_trojans>/var/ossec/etc/shared/rootkit_trojans.txt</rootkit_trojans>

<system_audit>/var/ossec/etc/shared/system_audit_rcl.txt</system_audit>
<system_audit>/var/ossec/etc/shared/system_audit_ssh.txt</system_audit>
<system_audit>/var/ossec/etc/shared/cis_debian_linux_rcl.txt</system_audit>

<skip_nfs>yes</skip_nfs>
</rootcheck>
```

ruleset

XML section name

```
<ruleset>
</ruleset>
```

Configuration options for enabling or disabling rules and decoders.

Options

- [rule_include](#)
- [rule_dir](#)
- [rule_exclude](#)
- [decoder_include](#)
- [decoder_dir](#)
- [decoder_exclude](#)
- [list](#)

rule_include

Load a single rule file.

Default value	n/a
Allowed values	Path and filename of rule to load

rule_dir

Load a directory of rules. The files will be loaded in alphabetical order and any duplicate filenames will be skipped.

Default value	ruleset/rules
Allowed values	Path to a directory of rule files.

Attributes

An optional pattern can be included in the opening tag. The pattern is a regex match string used to determine if a file should be loaded.

rule_exclude

Exclude a single rule file.

Default value	n/a
Allowed values	Path and filename of rule to exclude

decoder_include

Load a single decoder file.

Default value	n/a
Allowed values	Path and filename of decoder to load

decoder_dir

Load a directory of decoders. The files will be loaded in alphabetical order and any duplicate filenames will be skipped.

Default value	ruleset/decoders
Allowed values	Path to a directory of decoder files

Attributes

An optional pattern can be included in the opening tag. The pattern is a regex match string used to determine if a file should be loaded.

decoder_exclude

Exclude a single decoder file.

Default value	n/a
Allowed values	Path and filename of decoder to exclude

list

Load a single CDB reference for use by other rules.

Default value	n/a
Allowed values	Path to a list file to be loaded and compiled.

Note

Do not include the file extension. Wazuh will read the .cdb version of the file (the version generated by ossec-makelists from the .txt version of the file).

Example of configuration

```
<ruleset>
  <rule_include>ruleset/rules/my_rules.xml</rule_include>
  <rule_dir pattern="_rules.xml$">ruleset/rules</rule_dir>
  <rule_exclude>0215-policy_rules.xml</rule_exclude>
  <decoder_include>ruleset/decoders/my_decoder.xml</decoder_include>
  <decoder_dir pattern=".xml$">ruleset/decoders</decoder_dir>
  <decoder_exclude>ruleset/decoders/my_decoder.xml</decoder_exclude>
  <list>etc/lists/blocked_hosts</list>
</ruleset>
```

syscheck

XML section name

```
<syscheck>
</syscheck>
```

Configuration options for file integrity monitoring.

Options

- [directories](#)
- [ignore](#)
- [nodiff](#)
- [frequency](#)
- [scan_time](#)
- [scan_day](#)
- [auto_ignore](#)
- [alert_new_files](#)
- [scan_on_start](#)
- [windows_registry](#)
- [registry_ignore](#)
- [prefilter_cmd](#)
- [skip_nfs](#)

directories

Use this option to add or remove directories to be monitored. The directories must be comma separated.

All files and subdirectories within the noted directories will also be monitored.

Drive letters without directories are not valid. At a minimum the '.' should be included ([D:\.](#)).

This is to be set on the system to be monitored (or in the [agent.conf](#), if appropriate).

Default value	/etc,/usr/bin,/usr/sbin,/bin,/sbin
Allowed values	Any directory

Atributes:

realtime	This will enable real-time/continuous monitoring on Linux (using the inotify system calls) and Windows systems. Real time only works with directories, not individual files.	
	Allowed values	yes, no
report_changes	Report file changes. This is limited to text files at this time.	
	Allowed values	yes, no
check_all	All the following check_* options are used together.	
	Allowed values	yes, no

check_sum	Check the MD5 and SHA-1 hashes of the files.	
	Same as using both <code>check_sha1sum="yes"</code> and <code>check_md5sum="yes"</code> .	
	Allowed values	yes, no
check_sha1sum	Check only the SHA-1 hash of the files.	
	Allowed values	yes, no
	Check only the MD5 hash of the files.	
check_md5sum	Check only the MD5 hash of the files.	
	Allowed values	yes, no
	Check the size of the files.	
check_size	Allowed values	yes, no
	Check the owner of the files.	
	On Windows, uid will always be 0.	
check_group	Allowed values	yes, no
	Check the group owner of the files/directories.	
	Available for UNIX. On Windows, gid will always be 0 and the group name will be blank.	
check_perm	Allowed values	yes, no
	Check the UNIX permission of the files/directories.	
	On Windows, this will only check the POSIX permissions.	
check_mtime	Allowed values	yes, no
	Check the modification time of a file.	
	! New in version 2.0.	
check_inode	Allowed values	yes, no
	Check the file inode.	
	Available for UNIX. On Windows, inode will always be 0.	
restrict	! New in version 2.0.	
	Allowed values	yes, no
	Limit checks to files containing the entered string in the file name.	
restrict	Any directory or file name (but not a path) is allowed	
	Allowed value	string

ignore

List of files or directories to be ignored (one entry per line). Multiple lines may be entered to include multiple files or directories. These files and directories are still checked, but the results are ignored.

Default value	/etc/mtab
Allowed values	Any directory or file name

Attributes:

type	This is a simple regex pattern to filter out files so alerts are not generated

Allowed values	sregex
----------------	--------

nodiff

List of files to not compute the diff (one entry per line). It could be used for sensitive files like a private key, credentials stored in a file or database configuration, avoiding data leaking by sending the file content changes through alerts.

Default value	/etc/ssl/private.key
Allowed values	Any file name

Attributes:

type	This is a simple regex pattern to filter out files so alerts are not generated	
	Allowed values	sregex

frequency

Frequency that the syscheck will be run (in seconds).

Default value	21600
Allowed values	A positive number, time in seconds

scan_time

Time to run the scans. Times may be represented as 21pm or 8:30.

Default value	n/a
Allowed values	Time of day

Note

This may delay the initialization of real-time scans.

scan_day

Day of the week to run the scans(one entry per line). Multiple lines may be entered to include multiple registry entries.

Default value	n/a
Allowed values	Day of the week

auto_ignore

Specifies whether or not syscheck will ignore files that change too many times (after the third change).

Default value	yes
Allowed values	yes, no

Note

It is valid on: server and local.

alert_new_files

Specifies if syscheck should alert when new files are created.

Default value	no
----------------------	----

Allowed values	yes, no
-----------------------	---------

! Note

It is valid on: server and local.

scan_on_start

Specifies if syscheck scans immediately when started.

Default value	yes
Allowed values	yes, no

windows_registry

Use this option to monitor specified Windows registry entries (one entry per line). Multiple lines may be entered to include multiple registry entries.

Default value	HKEY_LOCAL_MACHINE\Software
Allowed values	Any registry entry

Atributes:

arch	Select the Registry view depending on the architecture.	
	Default value	32bit
	Allowed values	32bit, 64bit, both

! Note

New entries will not trigger alerts, only changes to existing entries.

registry_ignore

List of registry entries to be ignored. (one entry per line). Multiple lines may be entered to include multiple registry entries.

Default value	..CryptographyRNG
Allowed values	Any registry entry

prefilter_cmd

Run to prevent prelinking from creating false positives.

Default value	n/a
Allowed values	Command to prevent prelinking

Example:

```
<prefilter_cmd>/usr/sbin/prelink -y</prefilter_cmd>
```

! Note

This option may negatively impact performance as the configured command will be run for each file checked.

skip_nfs

Specifies if syscheck should scan network mounted filesystems (Works on Linux and FreeBSD). Currently, skip_nfs will exclude checking files on CIFS or NFS mounts.

Default value	no
Allowed values	yes, no

Default Unix configuration

```
<!-- File integrity monitoring -->
<syscheck>
  <disabled>no</disabled>

  <!-- Frequency that syscheck is executed default every 12 hours -->
  <frequency>43200</frequency>

  <scan_on_start>yes</scan_on_start>

  <!-- Generate alert when new file detected -->
  <alert_new_files>yes</alert_new_files>

  <!-- Don't ignore files that change more than 3 times -->
  <auto_ignore>no</auto_ignore>

  <!-- Directories to check (perform all possible verifications) -->
  <directories check_all="yes">/etc,/usr/bin,/usr/sbin</directories>
  <directories check_all="yes">/bin,/sbin,/boot</directories>

  <!-- Files/directories to ignore -->
  <ignore>/etc/mtab</ignore>
  <ignore>/etc/hosts.deny</ignore>
  <ignore>/etc/mail/statistics</ignore>
  <ignore>/etc/random-seed</ignore>
  <ignore>/etc/random.seed</ignore>
  <ignore>/etc/adjtime</ignore>
  <ignore>/etc/httpd/logs</ignore>
  <ignore>/etc/utmpx</ignore>
  <ignore>/etc/wtmpx</ignore>
  <ignore>/etc/cups/certs</ignore>
  <ignore>/etc/dumpdates</ignore>
  <ignore>/etc/svc/volatile</ignore>

  <!-- Check the file, but never compute the diff -->
  <nodiff>/etc/ssl/private.key</nodiff>

  <skip_nfs>yes</skip_nfs>
</syscheck>
```

syslog_output

XML section name

```
<syslog_output>  
</syslog_output>
```

Configuration options for sending alerts to a syslog server.

Options

- [server](#)
- [port](#)
- [level](#)
- [group](#)
- [rule_id](#)
- [location](#)
- [use_fqdn](#)
- [format](#)

server

The IP Address of the syslog server.

Default value	n/a
Allowed values	Any valid IP address

port

The port to forward alerts to.

Default value	514
Allowed values	Any valid port

level

The minimum level of the alerts to be forwarded.

Default value	n/a
Allowed values	Any level from 1 to 16

group

Group of the alerts to be forwarded.

Default value	n/a
Allowed values	Any valid group. Separate multiple groups with the pipe (" ") character.

Note

Observe that all groups must be finished by comma.

rule_id

The rule_id of the alerts to be forwarded.

Default value	n/a
Allowed values	Any valid rule_id

location

The location of the alerts to be forwarded.

Default value	n/a
Allowed values	Any valid log file location

use_fqdn

Toggle for full or truncated hostname configured on the server. By default, ossec truncates the hostname at the first period ('.') when generating syslog messages.

Default value	no
Allowed values	yes, no

format

Format of alert output. When `jsonout_output` in `global` section is enabled, alerts are read from `alerts.json` instead of `alerts.log` for JSON format.

Default value	default	
Allowed values	default	
cef		will output data in the ArcSight Common Event Format.
splunk		will output data in a Splunk-friendly format.
json		will output data in the JSON format that can be consumed by a variety of tools.

Example of configuration

```
<syslog_output>
  <server>192.168.1.3</server>
  <level>7</level>
  <format>json</format>
</syslog_output>
```

wodle name="open-scap"

XML section name

```
<wodle name="open-scap">  
</wodle>
```

Configuration options of the OpenSCAP wodle.

Options

- [disabled](#)
- [timeout](#)
- [interval](#)
- [scan-on-start](#)
- [content](#)

Options	Allowed values
disabled	yes, no
timeout	A positive number (seconds)
interval	A positive number
scan-on-start	yes, no
content	N/A

disabled

Disables the OpenSCAP wodle.

Default value	no
Allowed values	yes, no

timeout

Timeout for each evaluation.

Default value	1800
Allowed values	A positive number (seconds)

interval

Interval between OpenSCAP executions.

Default value	1d
Allowed values	A positive number that should contain a suffix character indicating a time unit, such as, s (seconds), m (minutes), h (hours), d (days).

scan-on-start

Run evaluation immediately when service is started.

Default value	yes
Allowed values	yes, no

content

Define an evaluation.

Attributes

type	Select content type: xccdf or oval.
path	Use the specified policy file (DataStream, XCCDF or OVAL). Default path: /var/ossec/wodles/oscap/policies
timeout	Timeout for the evaluation (in seconds). Use of this attribute overwrites the generic timeout.
xccdf-id	XCCDF id.
oval-id	OVAL id.
datastream-id	Datastream id.
cpe	CPE dictionary file. Default path: /var/ossec/wodles/oscap/policies
profile	Select profile.

Example of configuration

```
<wodle name="open-scap">

<timeout>1800</timeout>
<interval>1d</interval>
<scan-on-start>yes</scan-on-start>

<content type="xccdf" path="ssg-centos7-ds.xml"/>
<content type="xccdf" path="ssg-centos6-ds.xml"/>

</wodle>
```

Verifying configuration

Configuration section	command
Syscheck/Rootcheck	/var/ossec/bin/ossec-syscheckd -t
local files	/var/ossec/bin/ossec-logcollector -t
Wodles	/var/ossec/bin/wazuh-modulesd -t
global/rules/decoders (manager only)	/var/ossec/bin/ossec-analysisd -t
Client (agent only)	/var/ossec/bin/ossec-agentd -t

agent-auth

The agent-auth program is the client application used with [ossec-authd](#) to automatically add agents to a Wazuh manager.

Warning

By default there is no authentication or authorization involved in this transaction, so it is recommended that this daemon only be run when a new agent is being added.

-A <agent_name>	Agent name to be used.	
	Default Value	hostname
-a	Auto negotiate the most secure common SSL/TLS method with the client.	
	Default	TLS v1.2 only (if supported by the server).
-D	Directory where Wazuh is installed.	
	Default Value	/var/ossec
-d	Run in debug mode, can be repeated to increase the verbosity of messages.	
-g <group>	Run as a group.	
-h	Display the help message	
-k <path>	Display the full path to the agent key.	
-m <manager_ip>	IP address of the manager.	
-P <password>	Use the specified password instead of searching for it at authd.pass . If not provided in the file nor on the console, the client will connect to the server without a password (insecure mode).	
-p <port>	Port ossec-authd is running on.	
	Default Value	1515
-t	Test configuration.	
-V	Display version and license information.	
-v <path>	Display the full path to the CA certificate used to verify the server.	
-x <path>	Display the full path to the agent certificate.	

agent_control

The agent_control program allows you to query the manager for information about any agent and also allows you to initiate a syscheck/rootcheck scan on an agent the next time it checks in.

-h	Display the help message
-l	List available agents whether they are active or not.
-lc	List active agents
-i <agent_id>	Extract information from an agent
-R <agent_id>	Restart the Wazuh processes on the agent
-r	Run the integrity/rootcheck checking on agents. This must be used in conjunction with options -a or -u.
-a	Utilizes all agents
-u <agent_id>	Perform the requested action on the specified agent.

manage_agents

The manage_agents program is available in both a version for server and agent installations.

The purpose of manage_agents is to provide an easy-to-use interface to handle authentication keys for Wazuh agents. These authentication keys are required for secure (encrypted and authenticated) communication between the Wazuh server and its affiliated agent instances.

Usage

```
manage_agents -[Vhlj] [-a <ip> -n <name>] [-F sec] [-e id] [-r id] [-i id] [-f file]
```

Options

-V	Version and license message.	
-h	This help message.	
-j	Use JSON output.	
-l	List available agents.	
-a <ip>	Add new agent.	
	Supported installations	Manager
-n <name>	Name for new agent. Only valid along with -a .	
	Supported installations	Manager
-e <id>	Extracts key for an agent.	
	Supported installations	Manager
-r <id>	Remove an agent.	
	Supported installations	Manager
-i <id>	Import authentication key.	
	Supported installations	Agent
-F <sec>	Remove agents with duplicated IP if disconnected since <sec> seconds. With -F 0 it will always remove duplicated agents.	
	Supported installations	Manager
-f <file>	Bulk generate client keys from file. <file> contains lines in IP,NAME format.	
	Supported installations	Manager

ossec-control

The ossec-control script is used to start, stop, configure, or check on the status of Wazuh processes. This script can enable or disable client-syslog, database logging, agentless configurations, integration with slack and pagerduty, and debug mode.

start	Start the Wazuh processes.		
stop	Stop the Wazuh processes.		
restart	Restart the Wazuh processes.		
reload	Restart all Wazuh processes except ossec-execd. This allows an agent to reload without losing active response status. This option is not available on a local Wazuh installation.		
status	Determine which Wazuh processes are running.		
enable	Allowed options	database	Enable the ossec-dbd daemon for logging to a database. Server and local
		client-syslog	Enable ossec-csyslogd for logging to remote syslog. Server and local
		agentless	Enable ossec-agentlessd for running commands on systems without Wazuh agents. Server and local
		integrator	Enable integrator for connection to external APIs and alerting tools. Server
		auth	Enable the ossec-authd daemon for add agents to the manager. Server
		debug	Run all Wazuh daemons in debug mode.
disable	Allowed options	database	Disable the ossec-dbd daemon for logging to a database. Server and local
		client-syslog	Disable ossec-csyslogd for logging to remote syslog. Server and local
		agentless	Disable ossec-agentlessd for running commands on systems without Wazuh agents. Server and local
		integrator	Disable integrator for connection to external APIs and alerting tools. Server
		auth	Enable the ossec-authd daemon for add agents to the manager. Server
		debug	Turn off debug mode.

Note

To use the database option, Database support must be compiled in during initial installation.

ossec-logtest

The ossec-logtest program is a useful tool when working with Wazuh rules. This tool allows the testing and verification of rules against provided log examples in a way that simulates the action of ossec-analysisd. This can also assist with writing and debugging custom rules and troubleshooting false positives and negatives.

-a	Analysis of input lines as though they are live events.
-c <config>	Run using <code>config</code> as the configuration file. Default Value /var/ossec/etc/ossec.conf
-D <dir>	Specifies the chroot before it completes loading all rules,decoders, and lists and processing standard input.
-d	Run as a Print debug output to the terminal. This option may be repeated to increase the verbosity of the debug messages.group.
-h	Display the help message.
-t	Test configuration. This will display file details on the rules to be loaded by ossec-analysisd, decoders, and lists as they are loaded and the order they were processed.
-U <rule-id:alert-level:decoder-name>	This option will cause ossec-logtest to return a zero exit status if the test results for the provided log line match the criteria in the arguments. Only one log line should be supplied for this to be useful.
-V	Display the version and license information for Wazuh and ossec-logtest.
-v	Display the verbose results.

Note

- The ossec-logtest code requires access to all ossec configuration files.

Note

-v is the key option to troubleshoot a rule or decoder problem.

ossec-makelists

The `ossec-makelists` utility to compile cdb databases. This will scan `ossec.conf` for database files, check the mtime, and recompile all out of date databases.

-c <config>	Run with configuration file of <code>config</code> .
	Default Value <code>/var/ossec/etc/ossec.conf</code>
-d	Run in debug mode, may be repeated to increase the verbosity of the debug messages.
-F	Force the rebuild of all configured databases.
-g <group>	Run as a group.
-h	Display the help message.
-t	Test configuration.
-u <user>	Run as a user.
-V	Display the version and license information.

rootcheck_control

The `rootcheck_control` tool allows for the management of the policy monitoring and system auditing database that is stored on the server side.

Anomalies detected by the rootcheck functionality can be listed, and categorized into resolved and outstanding issues.

This tool can also display the last time that ossec-rootcheck was run.

-h	Display the help message.
-l	List the available agents.
-lc	List only the currently connected agents.
-u <id> / -u all	Update the database for the identified or all agents.
-i <agent_id>	Print the database for the agent.
-r	Used with -i to print all the resolved issues.
-q	Used with -i to print all the outstanding issues.
-L	Used with -i print the last scan.
-s	Change the output to CSV format.

syscheck_control

The syscheck_control program provides an interface for managing and viewing the integrity checking database.

-h	Display the help message.	
-l	List the available agents.	
-lc	List only the currently connected agents.	
-u <id> / -u all	Update the database for the identified or all agents.	
-i <agent_id>	Print the database for the agent.	
-r -i	List the modified registry entries for the agent. Supported installations Windows agents	
-f <file>	Used with -i to print information about a modified file.	
-z	Used with -f to zero the auto-ignore counter.	
-d	Used with -f to ignore that file.	
-s	Change the output to CSV format.	

syscheck_update

The syscheck_update program wipes the integrity check database. All information about files that were added to the integrity check database will be deleted leaving an empty database which will be populated again the next time the syscheck daemon runs on the agents or the server.

-h	Display the help message.
-l	List the available agents.
-a	Update the database for all agents.
-u <id> / -u local	Update the database for the specified agent or the local database.

clear_stats

The clear_stats program clears the events stats.

-h	Display the help message.
-a	Clear all the average stats.
-d	Clear the daily averages.
-w	Clear the weekly averages.

ossec-regex

The ossec-regex program is used to validate a regex expression.

The pattern should be enclosed in single quotes to help prevent any unintended interactions with the shell.

The syntax for ossec-regex is as follows:

```
/var/ossec/bin/ossec-regex '<pattern>'
```

It then reads strings from stdin and outputs matches to stdout.

```
+OSRegex_Execute and +OS_Regex are displayed if a match is successful.
```


verify-agent-conf

The verify-agent-conf program verifies the Wazuh [agent.conf](#) configuration.

This program displays an error if there is a problem or exits silently if the configuration is correct.

Log analysis

Here we will use Wazuh log collection and analysis capabilities to meet the following PCI DSS controls:

10.2.4: Invalid logical access attempts.

10.2.5: Use of and changes to identification and authentication mechanisms —including but not limited to creation of new accounts and escalation of privileges— and all changes, additions, or deletions to accounts with root or administrative privileges.

These controls require us to log invalid logical access attempts, multiple invalid login attempts (possible brute force attacks), privilege escalations, changes to accounts, etc. In order to achieve this, we have added PCI DSS tags to OSSEC log analysis rules, mapping them to the corresponding requirement(s). This makes it easy to analyze and visualize our PCI DSS related alerts.

The syntax used for rule tagging is **pci_dss_** followed by the number of the requirement (e.g., **pci_dss_10.2.4** and **pci_dss_10.2.5**).

Here are some examples of OSSEC rules tagged for PCI requirements 10.2.4 and 10.2.5:

```
<!--apache: access attempt -->
<rule id="30105" level="5">
  <if_sid>30101</if_sid>
  <match>denied by server configuration</match>
  <description>Attempt to access forbidden file or directory.</description>
  <group>access_denied,pci_dss_6.5.8,pci_dss_10.2.4,</group>
</rule>

<!-- syslog-sudo: elevation of privileges -->
<rule id="5401" level="5">
  <if_sid>5400</if_sid>
  <match>incorrect password attempt</match>
  <description>Failed attempt to run sudo</description>
  <group>pci_dss_10.2.4,pci_dss_10.2.5,</group>
</rule>

<rule id="5402" level="3">
  <if_sid>5400</if_sid>
  <regex> ; USER=root ; COMMAND=| ; USER=root ; TSID=\S+ ; COMMAND=</regex>
  <description>Successful sudo to ROOT executed</description>
  <group>pci_dss_10.2.5,pci_dss_10.2.2,</group>
</rule>

<!-- ssh: identification and authentication mechanisms -->
<rule id="5712" level="10" frequency="6" timeframe="120" ignore="60">
  <if_matched_sid>5710</if_matched_sid>
  <description>SSHD brute force trying to get access to </description>
  <description>the system.</description>
  <same_source_ip />
  <group>authentication_failures,pci_dss_11.4,pci_dss_10.2.4,pci_dss_10.2.5,</group>
</rule>

<rule id="5720" level="10" frequency="6">
  <if_matched_sid>5716</if_matched_sid>
  <same_source_ip />
  <description>Multiple SSHD authentication failures.</description>
  <group>authentication_failures,pci_dss_10.2.4,pci_dss_10.2.5,pci_dss_11.4,</group>
</rule>
```

Use cases

In this scenario, we try to open the file `cardholder_data.txt`. Since our current user doesn't have read access to the file, we run `sudo` to elevate privileges.

```
[agent@centos ~]$ ls -l
total 0
drwxrwxr-x. 2 agent agent 6 Jan 5 18:34 centos
drwxr-x--- 2 root root 33 Jan 5 18:32 credit_cards
drwxrwxr-x. 2 agent agent 6 Jan 5 18:34 user_data
[agent@centos ~]$ sudo cat credit_cards/cardholder_data.txt
Number: 0000-0000-0000-0000
Holder: Mr. John Smith
```

Using the `sudo` log analysis decoder and rules, Wazuh will generate an alert for this particular action and write it to `alerts.log`. Using the rule tags we can see which PCI DSS requirements are specifically related to this alert.

```
root@ubuntu:~# tail -n10 /var/ossec/logs/alerts/alerts.log

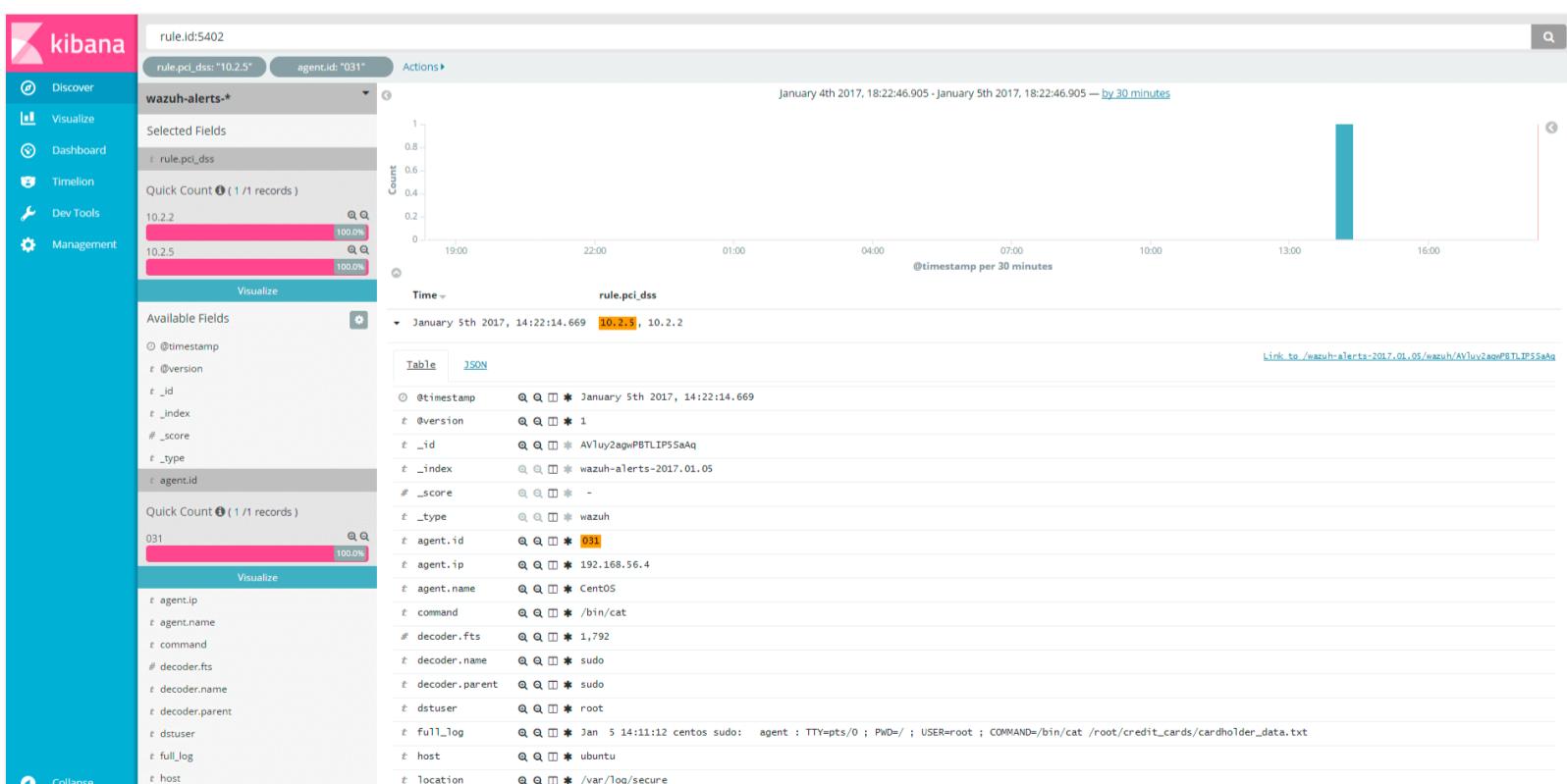
** Alert 1483621881.263207: - syslog,sudo,pci_dss_10.2.5,pci_dss_10.2.2,
2017 Jan 05 14:11:21 (CentOS) 192.168.56.4->/var/log/secure
Rule: 5402 (level 3) -> 'Successful sudo to ROOT executed'
User: root
Jan 5 14:11:12 centos sudo:    agent : TTY=pts/0 ; PWD=/ ; USER=root ; COMMAND=/bin/cat
/root/credit_cards/cardholder_data.txt
tty: pts/0
pwd: /
command: /bin/cat
```

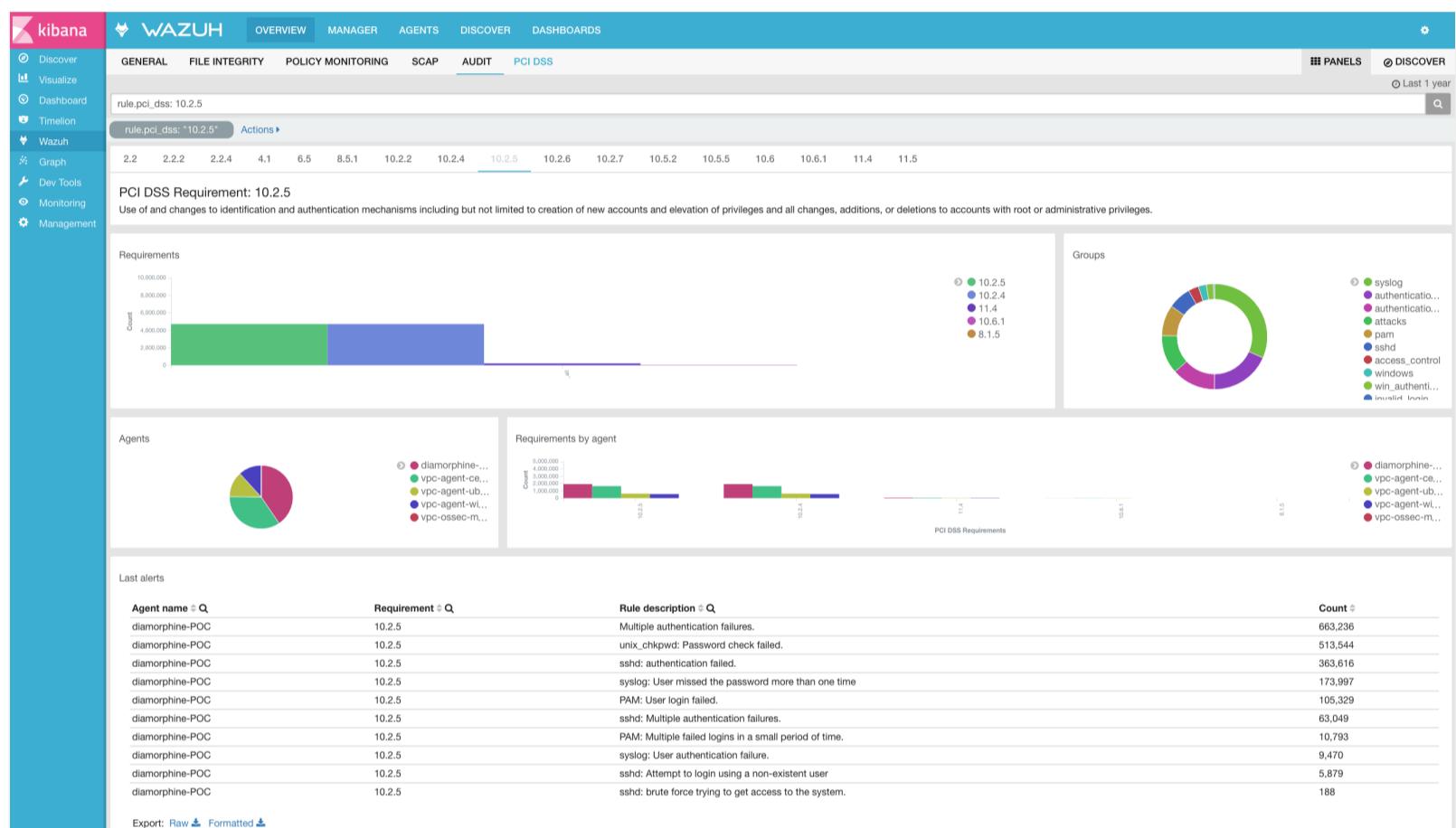
Since we have JSON output enabled, we can also see the alert in `alerts.json`:

```
root@ubuntu:~# tail -n1 /var/ossec/logs/alerts/alerts.json | jq
```

```
{
  "rule": {
    "level": 3,
    "description": "Successful sudo to ROOT executed",
    "id": 5402,
    "firedtimes": 1,
    "groups": [
      "syslog",
      "sudo"
    ],
    "pci_dss": [
      "10.2.5",
      "10.2.2"
    ]
  },
  "agent": {
    "id": "031",
    "name": "CentOS",
    "ip": "192.168.56.4"
  },
  "manager": {
    "name": "ubuntu"
  },
  "srcuser": "agent",
  "dstuser": "root",
  "full_log": "Jan 5 14:11:12 centos sudo: agent : TTY=pts/0 ; PWD=/ ; USER=root ; COMMAND=/bin/cat /root/credit_cards/cardholder_data.txt",
  "program_name": "sudo",
  "tty": "pts/0",
  "pwd": "/",
  "command": "/bin/cat",
  "decoder": {
    "fts": 1792,
    "parent": "sudo",
    "name": "sudo"
  },
  "timestamp": "2017 Jan 05 14:11:21",
  "location": "/var/log/secure"
}
}
```

Kibana displays information in an organized way, allowing filtering by different types of alert fields, including compliance controls. We have also developed a couple of PCI DSS dashboards for convenient viewing of relevant alerts.





Policy monitoring

The rootcheck module can be used to enforce and monitor your security policy. This is the process of verifying that all systems conform to a set of predefined rules surrounding configuration settings and approved application usage.

There are several PCI DSS requirements to verify that systems are properly hardened. An example would be:

2.2: Develop configuration standards for all system components. Assure that these standards address all known security vulnerabilities and are consistent with industry-accepted system hardening standards.

Sources of industry-accepted system hardening standards may include, but are not limited to: Center for Internet Security (CIS), International Organization for Standardization (ISO), SysAdmin Audit Network Security (SANS), Institute National Institute of Standards Technology (NIST).

Wazuh includes out-of-the-box, CIS baselines for Debian and Red Hat. Other baselines could be created for other systems or applications as well, just by adding the corresponding rootcheck file:

```
<rootcheck>
  <system_audit>/var/ossec/etc/shared/cis_debian_linux_rcl.txt</system_audit>
  <system_audit>/var/ossec/etc/shared/cis_rhel_linux_rcl.txt</system_audit>
  <system_audit>/var/ossec/etc/shared/cis_rhel5_linux_rcl.txt</system_audit>
</rootcheck>
```

Other PCI DSS requirements ask us to check that applications (especially network services) are configured in a secure way. One example is the following control:

2.2.4: Configure system security parameters to prevent misuse.

The following are good examples of rootcheck rules developed to check the configuration of SSH services:

```
[SSH Configuration - Protocol version 1 enabled {PCI_DSS: 2.2.4}] [any]
f:/etc/ssh/sshd_config -> !r:^# && r:Protocol\.+1;

[SSH Configuration - Root login allowed {PCI_DSS: 2.2.4}] [any]
f:/etc/ssh/sshd_config -> !r:^# && r:PermitRootLogin\.+yes;
```

In [Wazuh](#), the rootcheck rules use this syntax in the rootcheck name: **{PCI_DSS: X.Y.Z}**, mapping all rootchecks to their relevant PCI DSS requirement.

Use cases

In order to check SSH security settings and help meet requirement 2.2.4, we have developed the rootchecks `system_audit_ssh`. In our example, when Wazuh runs a rootcheck scan, it is able to detect certain security deficiencies in the SSH configuration.

```
[root@manager ossec]# cat etc/ossec.conf | grep system_audit_ssh -B 4 -A 2
```

```
<rootcheck>
  <rootkit_files>/var/ossec/etc/shared/rootkit_files.txt</rootkit_files>
  <rootkit_trojans>/var/ossec/etc/shared/rootkit_trojans.txt</rootkit_trojans>
  <system_audit>/var/ossec/etc/shared/system_audit_rcl.txt</system_audit>
  <system_audit>/var/ossec/etc/shared/ssh/system_audit_ssh.txt</system_audit>
</rootcheck>
```

If enabled, the file `archives.log` stores every log parsed by the Wazuh engine, whether it becomes an alert or not:

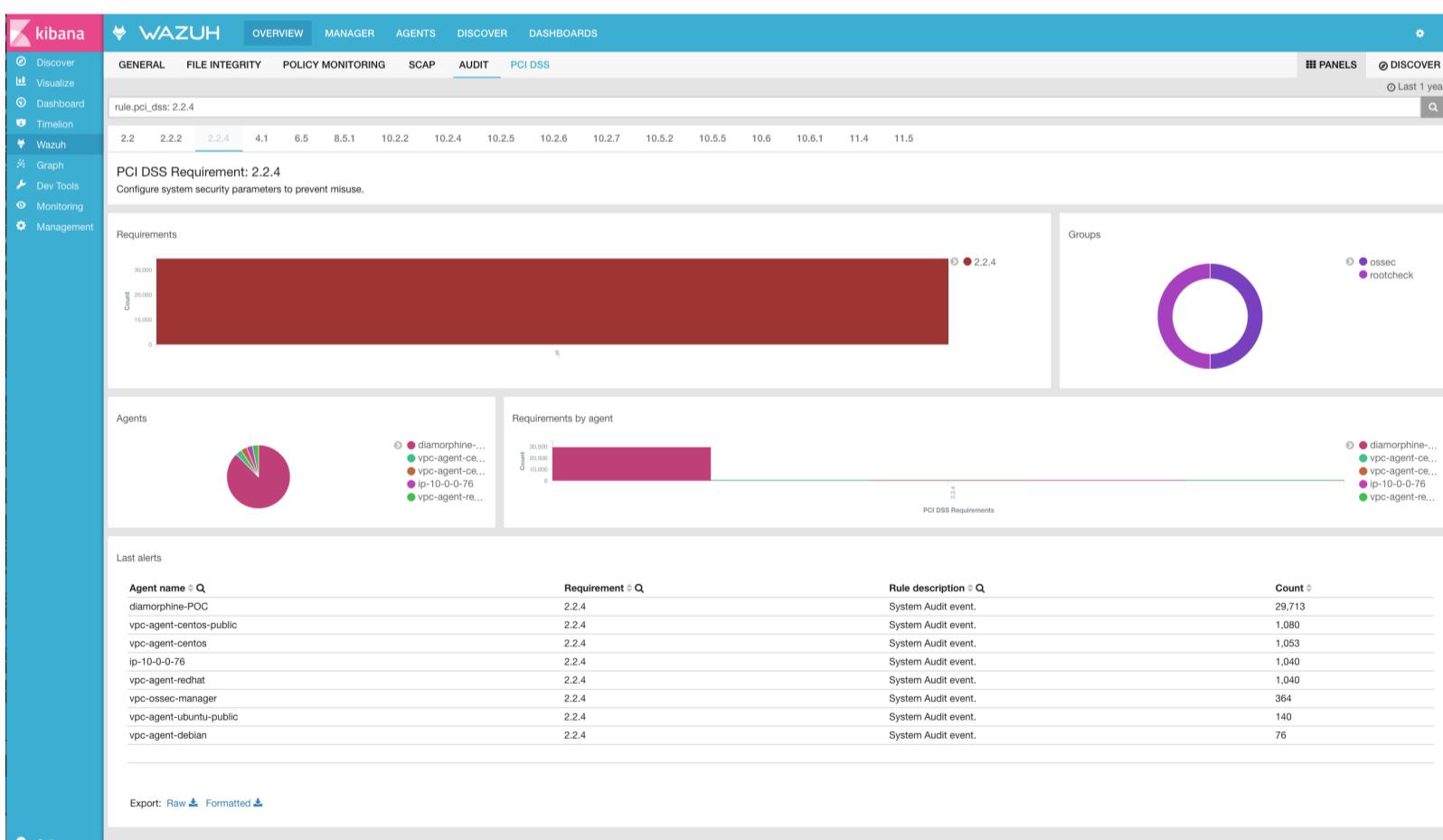
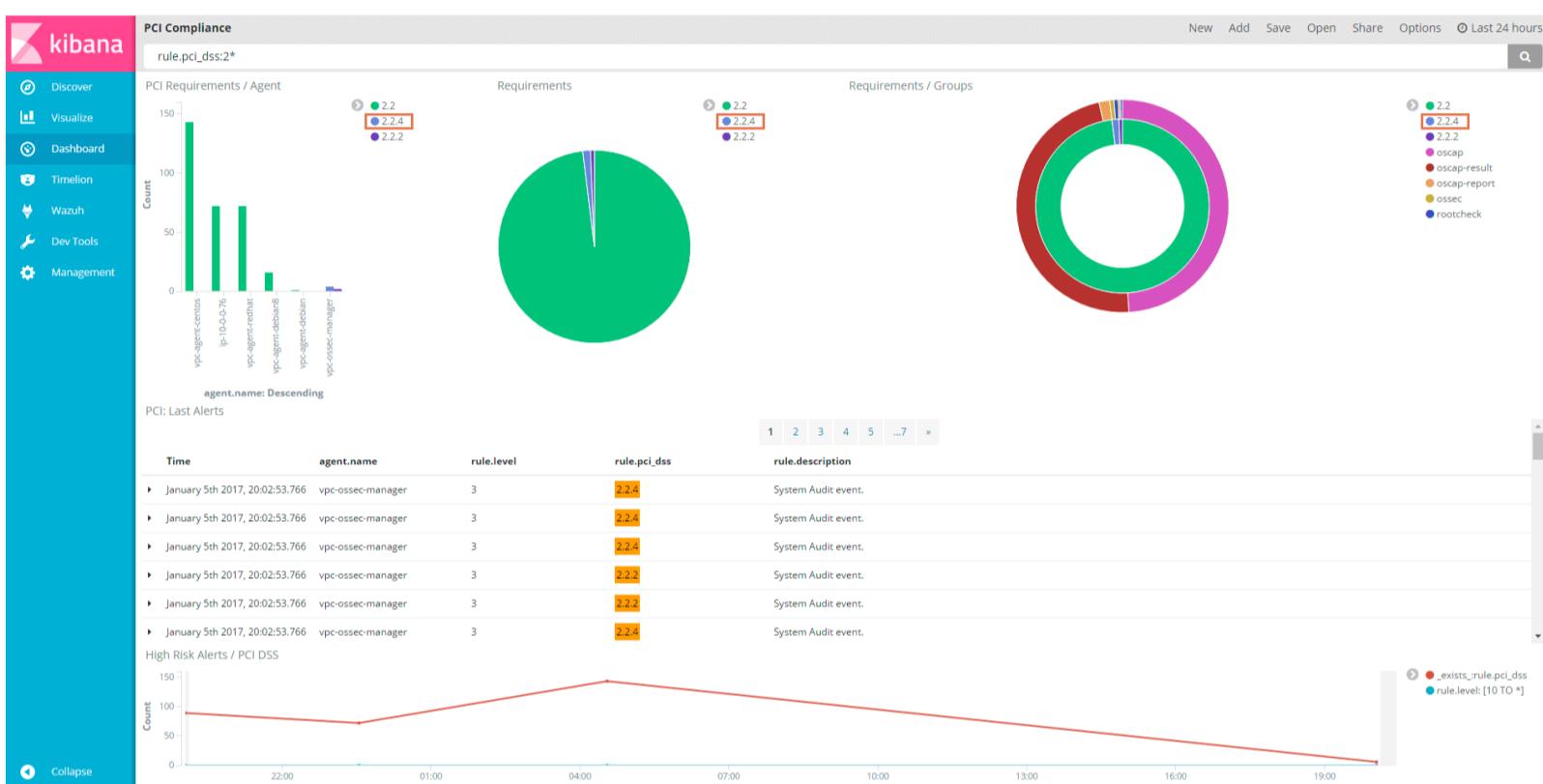
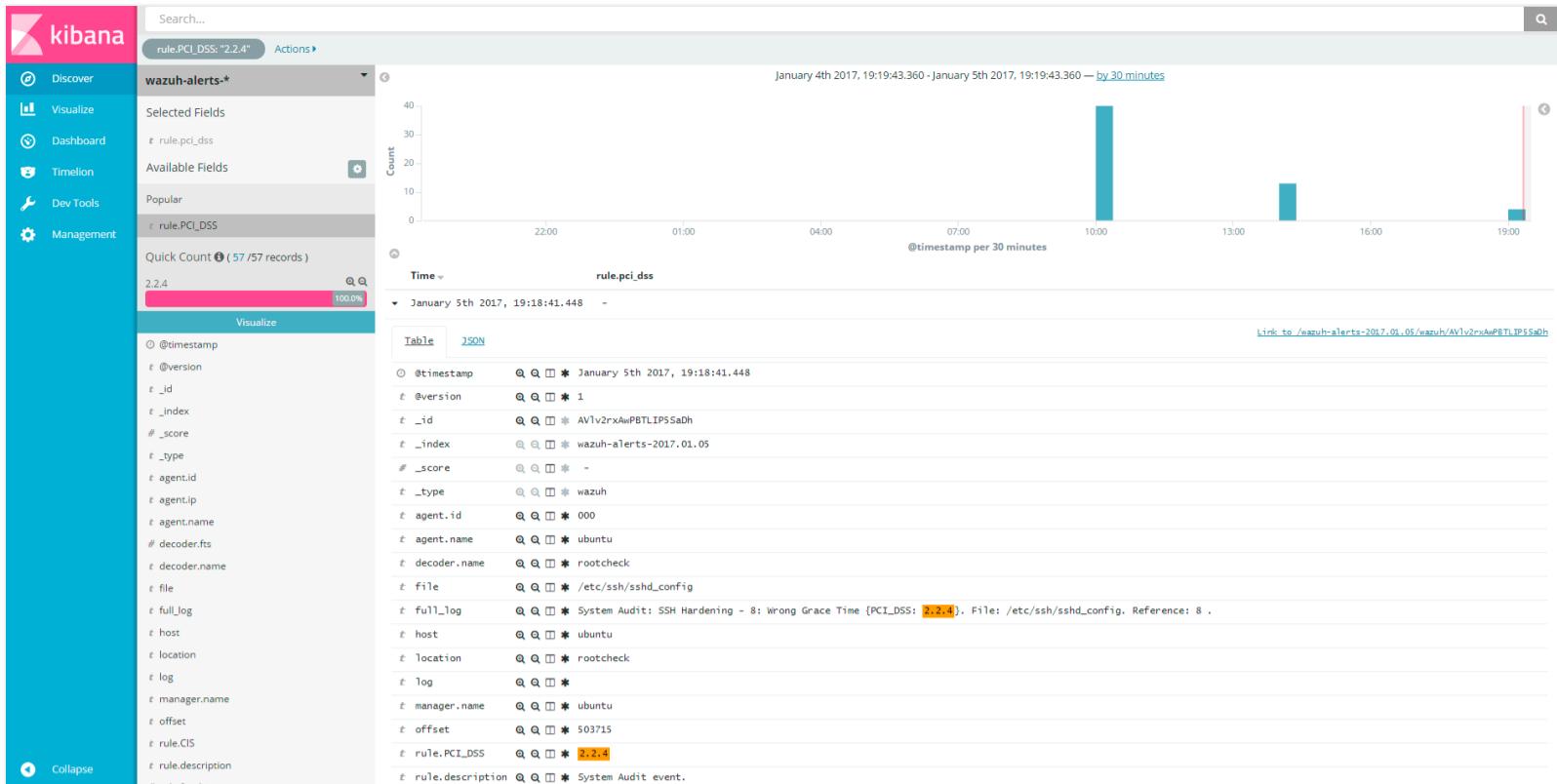
```
[root@manager ossec]# tail -f logs/archives/archives.log
2016 Jan 29 12:58:02 manager->rootcheck Ending rootcheck scan.
2016 Jan 29 13:07:18 manager->ossec-monitord ossec: Ossec started.
2016 Jan 29 13:08:34 manager->rootcheck Starting rootcheck scan.
2016 Jan 29 13:08:36 manager->rootcheck System Audit: SSH Hardening - 3: Root can log in {PCI_DSS: 2.2.4}.
File: /etc/ssh/sshd_config. Reference: 3 .
2016 Jan 29 13:08:36 manager->rootcheck System Audit: SSH Hardening - 4: No Public Key authentication {PCI_DSS: 2.2.4}. File: /etc/ssh/sshd_config. Reference: 4 .
2016 Jan 29 13:08:36 manager->rootcheck System Audit: SSH Hardening - 5: Password Authentication {PCI_DSS: 2.2.4}. File: /etc/ssh/sshd_config. Reference: 5 .
2016 Jan 29 13:08:36 manager->rootcheck System Audit: SSH Hardening - 6: Empty passwords allowed {PCI_DSS: 2.2.4}. File: /etc/ssh/sshd_config. Reference: 6 .
2016 Jan 29 13:08:36 manager->rootcheck System Audit: SSH Hardening - 7: Rhost or shost used for authentication {PCI_DSS: 2.2.4}. File: /etc/ssh/sshd_config. Reference: 7 .
2016 Jan 29 13:08:36 manager->rootcheck System Audit: SSH Hardening - 8: Wrong Grace Time {PCI_DSS: 2.2.4}.
File: /etc/ssh/sshd_config. Reference: 8 .
2016 Jan 29 13:08:36 manager->rootcheck System Audit: SSH Hardening - 9: Wrong Maximum number of authentication attempts {PCI_DSS: 2.2.4}. File: /etc/ssh/sshd_config. Reference: 9 .
```

In this case, all the logs above are alerts, so we will see an instance of the last alert in JSON:

```
[root@manager ossec]# tail -n 1 logs/alerts/alerts.json | pjson
```

```
{
  "rule": {
    "level": 3,
    "description": "System Audit event.",
    "id": 516,
    "firedtimes": 7,
    "groups": [
      "ossec",
      "rootcheck"
    ],
    "pci_dss": [
      "2.2.4"
    ]
  },
  "agent": {
    "id": "000",
    "name": "manager"
  },
  "manager": {
    "name": "manager"
  },
  "full_log": "System Audit: SSH Hardening - 9: Wrong Maximum number of authentication attempts {PCI_DSS: 2.2.4}. File: /etc/ssh/sshd_config. Reference: 9 .",
  "title": "SSH Hardening - 9: Wrong Maximum number of authentication attempts",
  "file": "/etc/ssh/sshd_config",
  "decoder": {
    "name": "rootcheck"
  },
  "timestamp": "2016 Jan 29 13:08:36",
  "location": "rootcheck"
}
```

Kibana shows the full information about the alert:



Rootkit detection

Rootkit and trojan detection is performed using two files: `rootkit_files.txt` and `rootkit_trojans.txt`. In addition, other low-level tests are performed to detect kernel-level rootkits. You can use these capabilities by adding references to these files in `ossec.conf`:

```
<rootcheck>
  <rootkit_files>/var/ossec/etc/shared/rootkit_files.txt</rootkit_files>
  <rootkit_trojans>/var/ossec/etc/shared/rootkit_trojans.txt</rootkit_trojans>
</rootcheck>
```

These are the options available for the `rootcheck` component:

- **`rootkit_files`**: Contains the Unix-based application level rootkit signatures.
- **`rootkit_trojans`**: Contains the Unix-based application level trojan signatures.
- **`check_files`**: Enable or disable the rootkit checks. Default yes.
- **`check_trojans`**: Enable or disable the trojan checks. Default yes.
- **`check_dev`**: Check for suspicious files in the `/dev` filesystem. Default yes.
- **`check_sys`**: Scan the whole system for low level anomalies. Default yes.
- **`check_pids`**: Check processes for anomalies. Default yes.
- **`check_ports`**: Check all listening ports for anomalies. Default yes.
- **`check_if`**: Check interfaces for anomalies. Default yes.

Rootcheck helps you to meet PCI DSS requirement 11.4 related to intrusions, trojans, and malware in general:

11.4: Use intrusion-detection and/or intrusion-prevention techniques to detect and/or prevent intrusions into the network. Keep all intrusion-detection and prevention engines, baselines, and signatures up to date. Intrusion detection and/or intrusion prevention techniques (such as IDS/IPS) compare the traffic coming into the network with known “signatures” and/or behaviors of thousands of compromise types (hacker tools, Trojans, and other malware), and send alerts and/or stop the attempt as it happens.

Use cases

Wazuh performs several tests to detect rootkits. One of them is to check for files hidden in `/dev`. The `/dev` directory should only contain device-specific files such as the primary IDE hard disk (`/dev/hda`), the kernel random number generators (`/dev/random` and `/dev/urandom`), etc. Any additional files, outside of the expected device-specific files, should be inspected because many rootkits use `/dev` as a storage partition to hide files. In the following example we have created the file `.hid` which is detected by OSSEC and generates the corresponding alert.

```
[root@manager /]# ls -a /dev | grep '^\.'
.
..
.hid
[root@manager /]# tail -n 25 /var/ossec/logs/alerts/alerts.log
Rule: 502 (level 3) -> 'Ossec server started.'
ossec: Ossec started.

** Alert 1454086362.26393: mail - ossec,rootcheck
2016 Jan 29 16:52:42 manager->rootcheck
Rule: 510 (level 7) -> 'Host-based anomaly detection event (rootcheck).'
File '/dev/.hid' present on /dev. Possible hidden file.
```


File integrity monitoring

File integrity monitoring (Syscheck) is performed by comparing the cryptographic checksum and other attributes of a known good file against the checksum and attributes of that file after it has been modified.

First, the Wazuh agent scans the system at an interval you specify, and it sends the checksums of the monitored files and registry keys (for Windows systems) to the Wazuh server. Then, the server stores the checksums and looks for modifications by comparing the newly received checksums against the historical checksum values for those files and/or registry keys. An alert is sent if the checksum (or another file attribute) changes. Wazuh also supports near real-time file integrity checking where this is desired.

[Syscheck](#) can be used to meet PCI DSS requirement 11.5:

11.5 Deploy a change-detection mechanism (for example, file-integrity monitoring tools) to alert personnel to unauthorized modification (including changes, additions, and deletions) of critical system files, configuration files, or content files; and configure the software to perform critical file comparisons at least weekly.

Use cases

In this example, we have configured Wazuh to detect changes in the file `/root/credit_cards`.

```
<syscheck>
  <directories check_all="yes" report_changes="yes">/root/credit_cards</directories>
</syscheck>
```

So, when we modify the file, Wazuh generates an alert.

```
[root@centos ~]# ls -l credit_cards
total 4
-rw-r--r--. 1 root root 14 Jan 10 19:33 cardholder_data.txt
[root@centos ~]# cat credit_cards/cardholder_data.txt
User1 = card4
[root@centos ~]# echo "User1 = card5" > credit_cards/cardholder_data.txt
[root@centos ~]# cat credit_cards/cardholder_data.txt
User1 = card5
```

As you can see, syscheck alerts are tagged with the requirement 11.5.

```
root@ubuntu:~# tail -n28 /var/ossec/logs/alerts/alerts.log
```

```
** Alert 1484071804.77110: - ossec,syscheck,pci_dss_11.5,
2017 Jan 10 19:10:04 (CentOS) 192.168.56.4->syscheck
Rule: 550 (level 7) -> 'Integrity checksum changed.'
Integrity checksum changed for: '/root/credit_cards/cardholder_data.txt'
Old md5sum was: '713f9c28cee03fc39f611d8e6ded6333'
New md5sum is : '313eba655eba3ebd814deee1b7bd7be1'
Old sha1sum was: '41f840a0f1335144d973e2bebb496e48fd3592e9'
New sha1sum is : 'a4e70ed0ca7bf67b4f5559a9d34a0d6a200927b2'

File: /root/credit_cards/cardholder_data.txt
New size: 14
New permissions: 100644
New user: root (0)
New group: root (0)
Old MD5: 713f9c28cee03fc39f611d8e6ded6333
New MD5: 313eba655eba3ebd814deee1b7bd7be1
Old SHA1: 41f840a0f1335144d973e2bebb496e48fd3592e9
New SHA1: a4e70ed0ca7bf67b4f5559a9d34a0d6a200927b2
Old date: Tue Jan 10 19:02:07 2017
New date: Tue Jan 10 19:09:58 2017
New inode: 1110
What changed: 1c1
< User1 = card4
---
> User1 = card5
```

The screenshot shows the Kibana interface with a sidebar on the left containing navigation links: Discover, Visualize, Dashboard, Timeline, Dev Tools, and Management. The main area displays a table of search results for an alert. The table has two columns: 'Field' and 'Value'. The 'Field' column includes @timestamp, @version, _id, _index, _score, _type, agent.id, agent.ip, agent.name, decoder.name, full_log, host, location, log, manager.name, offset, rule.description, rule.firedtimes, rule.frequency, rule.groups, rule.id, rule.level, source, script, syscheck.diff, syscheck.event, syscheck.gid_after, syscheck.gname_after, syscheck.inode_after, syscheck.inode_before, syscheck.md5_after, syscheck.md5_before, syscheck.mtime_after, and syscheck.mtime_before. The 'Value' column contains the corresponding alert details, such as the timestamp (January 10th, 2017, 19:14:13.298), version (1), _id (AvmJ1mvWzRse0Rq7pd2), _index (wazuh-alerts-2017.01.10), _score (~), _type (wazuh), agent.id (031), agent.ip (192.168.56.4), agent.name (CentOS), decoder.name (syscheck_integrity_changed), full_log (Integrity checksum changed for: '/root/credit_cards/cardholder_data.txt'). The full_log value also includes detailed checksum information: Old md5sum was: '713f9c28cee03fc39f611d8e6ded6333', New md5sum is : '313eba655eba3ebd814deee1b7bd7be1', Old sha1sum was: '41f840a0f1335144d973e2bebb496e48fd3592e9', and New sha1sum is : 'a4e70ed0ca7bf67b4f5559a9d34a0d6a200927b2'. The table also shows host (ubuntu), location (syscheck), log (~), manager.name (ubuntu), offset (112915), rule.description (Integrity checksum changed.), rule.firedtimes (4), rule.frequency (ossec, syscheck), rule.groups (ossec, syscheck), rule.id (550), rule.level (7), rule.pci_dss (11.5), source (/var/ossec/logs/alerts/alerts.json), syscheck.diff (1c1), syscheck.event (modified), and syscheck.gid_after (0).

Kibana

4 hits

rule.pci_dss:"11.5" AND syscheck.path:"/root/credit_cards/cardholder_data.txt"

New Save Open Share Last 24 hours

wazuh-alerts*

Selected Fields: _source

Available Fields: Popular, rule.pci_dss, @timestamp, @version, _id, _index, _score, _type, agent.id, agent.ip, agent.name, decoder.name, full_log, host, location, log, manager.name, offset, rule.description, rule.freetimes, rule.groups, rule.id, rule.level, source, syscheck.event, syscheck.gid_after

Count: 4

Time: January 9th 2017, 19:35:22.590 - January 10th 2017, 19:35:22.590 — by 30 minutes

@timestamp per 30 minutes

_source

January 10th, 19:14:13.298 syscheck.path: /root/credit_cards/cardholder_data.txt rule.pci_dss: 11.5 syscheck.uname_after: root syscheck.mtime_after: January 10th, 20:14:00.000 syscheck.md5_before: 713f9c28cee03fc39f611d8e6ded6333 syscheck.gid_after: 0 syscheck.size_after: 14 syscheck.diff: 1c1 < User1 = card4 --- > User1 = card5 syscheck.mtime_before: January 10th 2017, 20:13:26.000 syscheck.shai_after: a4e70ed0ca7bf67b4f559a9d34a0d6a200927b2 syscheck.gname_after: root syscheck.uid_after: 0 syscheck.event: modified syscheck.perm_after: 100644 syscheck.shai_before: 41f840a0f1335144d973e2bebb496e48fd3592e9 syscheck.md5_after: 313eba655eba8ebd814deee1b7bd7b7e1 syscheck.inode_after: 1110 agent.ip: 192.168.56.4 agent.name: CentOS agent.id: 031 offset: 112915 manager.name: ubuntu log: 7b4f5559a9d34a0d6a200927b2 syscheck.inode_after: 1110 agent.ip: 192.168.56.4 agent.name: CentOS agent.id: 031 offset: 111725 manager.name: ubuntu log: 655eba8ebd814deee1b7bd7b7e1 syscheck.inode_after: 1110 agent.ip: 192.168.56.4 agent.name: CentOS agent.id: 031 offset: 110535 log: 4 agent.name: CentOS agent.id: 031 manager.name: ubuntu offset: 93747 log: rule.freetimes: 1 rule.level: 7 rule.description: Integrity checksum changed.

January 10th, 19:13:38.291 syscheck.path: /root/credit_cards/cardholder_data.txt rule.pci_dss: 11.5 syscheck.mtime_after: January 10th, 20:13:26.000 syscheck.uname_after: root syscheck.size_after: 14 syscheck.md5_before: 313eba655eba8ebd814deee1b7bd7b7e1 syscheck.gid_after: 0 syscheck.size_after: 14 syscheck.diff: 1c1 < User1 = card4 --- > User1 = card5 syscheck.mtime_before: January 10th 2017, 20:09:58.000 syscheck.shai_after: a4e70ed0ca7bf67b4f559a9d34a0d6a200927b2 syscheck.gname_after: root syscheck.uid_after: 0 syscheck.event: modified syscheck.perm_after: 100644 syscheck.md5_after: 713f9c28cee03fc39f611d8e6ded6333 syscheck.shai_before: a4e70ed0ca7bf67b4f559a9d34a0d6a200927b2 syscheck.gname_after: root syscheck.event: modified syscheck.shai_before: 41f840a0f1335144d973e2bebb496e48fd3592e9 syscheck.md5_after: 313eba655eba8ebd814deee1b7bd7b7e1 syscheck.inode_after: 1110 agent.ip: 192.168.56.4 agent.name: CentOS agent.id: 031 offset: 111725 manager.name: ubuntu log: 7b4f5559a9d34a0d6a200927b2 syscheck.inode_after: 1110 agent.ip: 192.168.56.4 agent.name: CentOS agent.id: 031 offset: 110535 log: 655eba8ebd814deee1b7bd7b7e1 syscheck.inode_after: 1110 agent.ip: 192.168.56.4 agent.name: CentOS agent.id: 031 offset: 110535 log: 4 agent.name: CentOS agent.id: 031 manager.name: ubuntu offset: 93747 log: rule.freetimes: 1 rule.level: 7 rule.description: Integrity checksum changed.

January 10th, 19:10:13.261 syscheck.path: /root/credit_cards/cardholder_data.txt rule.pci_dss: 11.5 syscheck.mtime_after: January 10th, 20:09:58.000 syscheck.size_after: 14 syscheck.md5_before: 713f9c28cee03fc39f611d8e6ded6333 syscheck.gid_after: 0 syscheck.size_after: 14 syscheck.diff: 1c1 < User1 = card4 --- > User1 = card5 syscheck.mtime_before: January 10th 2017, 20:02:07.000 syscheck.shai_after: a4e70ed0ca7bf67b4f559a9d34a0d6a200927b2 syscheck.gname_after: root syscheck.uid_after: 0 syscheck.event: modified syscheck.perm_after: 100644 syscheck.md5_after: 713f9c28cee03fc39f611d8e6ded6333 syscheck.shai_before: 41f840a0f1335144d973e2bebb496e48fd3592e9 syscheck.gname_after: a4e70ed0ca7bf67b4f559a9d34a0d6a200927b2 syscheck.event: modified syscheck.shai_before: 41f840a0f1335144d973e2bebb496e48fd3592e9 syscheck.md5_after: 313eba655eba8ebd814deee1b7bd7b7e1 syscheck.inode_after: 1110 agent.ip: 192.168.56.4 agent.name: CentOS agent.id: 031 offset: 110535 log: 7b4f5559a9d34a0d6a200927b2 syscheck.inode_after: 1110 agent.ip: 192.168.56.4 agent.name: CentOS agent.id: 031 offset: 110535 log: 655eba8ebd814deee1b7bd7b7e1 syscheck.inode_after: 1110 agent.ip: 192.168.56.4 agent.name: CentOS agent.id: 031 offset: 110535 log: 4 agent.name: CentOS agent.id: 031 manager.name: ubuntu offset: 93747 log: rule.freetimes: 1 rule.level: 7 rule.description: Integrity checksum changed.

January 10th, 19:03:36.219 syscheck.path: /root/credit_cards/cardholder_data.txt rule.pci_dss: 11.5 syscheck.mtime_after: January 10th, 20:02:07.000 syscheck.size_after: 14 syscheck.md5_before: 713f9c28cee03fc39f611d8e6ded6333 syscheck.gid_after: 0 syscheck.size_after: 14 syscheck.diff: 1c1 < User1 = card4 --- > User1 = card5 syscheck.mtime_before: January 10th 2017, 19:56:55.000 syscheck.shai_after: 41f840a0f1335144d973e2bebb496e48fd3592e9 syscheck.gname_after: root syscheck.uid_after: 0 syscheck.event: modified syscheck.perm_after: 100644 syscheck.md5_after: 713f9c28cee03fc39f611d8e6ded6333 syscheck.inode_after: 1110 agent.ip: 192.168.56.4 agent.name: CentOS agent.id: 031 offset: 110535 log: 4 agent.name: CentOS agent.id: 031 manager.name: ubuntu offset: 93747 log: rule.freetimes: 1 rule.level: 7 rule.description: Integrity checksum changed.

Kibana

WAZUH

OVERVIEW MANAGER AGENTS DISCOVER DASHBOARDS

rule.pci_dss: 11.5

GENERAL FILE INTEGRITY POLICY MONITORING SCAP AUDIT PCI DSS

PANELS DISCOVER Last 1 year

PCI DSS Requirement: 11.5

Deploy a change detection mechanism (for example, file integrity monitoring tools) to alert personnel to unauthorized modification of critical system files, configuration files, or content files; and configure the software to perform critical file comparisons at least weekly.

Requirements

Count: 11.5

Groups

Agents

Last alerts

Agent name	Requirement	Rule description	Count
vpc-agent-debian	11.5	File added to the system.	5,725
vpc-agent-debian	11.5	Integrity checksum changed.	65
vpc-agent-debian	11.5	File deleted. Unable to retrieve checksum.	10
vpc-agent-centos-public	11.5	Integrity checksum changed.	1,240
vpc-agent-centos-public	11.5	File added to the system.	25
vpc-agent-windows	11.5	File added to the system.	25
vpc-agent-windows	11.5	Integrity checksum changed.	23
vpc-agent-centos	11.5	File deleted. Unable to retrieve checksum.	9
vpc-agent-centos	11.5	Integrity checksum changed.	13
vpc-agent-centos	11.5	File added to the system.	4

Export: Raw Formatted

Active response

Although [active response](#) is not explicitly discussed in PCI DSS, it is important to mention that an automated remediation to security violations and threats is a powerful tool that reduces your risk.

Active response allows a scripted action to be performed whenever an event matches certain rules in your Wazuh ruleset. Remedial action could be a firewall block or drop, traffic shaping or throttling, account lockout, etc...

Elastic Stack

Wazuh integration with Elastic Stack comes with out-of-the-box dashboards for PCI DSS compliance and CIS benchmarking. You can do forensic and historical analysis of your alerts and store your data for years using a reliable and scalable platform. Optionally, this can even include your archived events rather than only your alert events.

The following requirements can be met with a combination of Wazuh + Elastic Stack:

- 10.5:** Secure audit trails so they cannot be altered.
- 10.6.1:** Review the following at least daily: All security events, Logs of all critical system components, etc.
- 10.7:** Retain audit trail history for at least one year, with a minimum of three months immediately available for analysis.

Log data collection

Log data collection is the real-time process of making sense out of the records generated by servers or devices. This component can receive logs through text files or Windows event logs. It can also directly receive logs via remote syslog (useful for firewalls, etc...) The purpose of this process is the identification of application or system errors, misconfigurations, intrusion attempts, policy violations or security issues.

The memory and CPU requirements of the Wazuh agent are insignificant because it mostly just forwards events to the manager. However, on the Wazuh manager, CPU and memory consumption can increase quickly depending on the events per second (EPS) that the manager has to analyze.

Contents

- [How it works](#)
 - [Log collection](#)
 - [Analysis](#)
 - [Alert](#)
- [Configuration](#)
 - [Basic usage](#)
 - [Monitoring logs using regular expressions for file names](#)
 - [Monitoring date-based logs](#)
 - [Reading logs from Windows Event Log](#)
 - [Reading events from Windows Event Channel](#)
 - [Filtering events from Windows Event Channel with queries](#)
 - [Using environment variables](#)
- [FAQ](#)
 - [Are the logs analyzed on each agent?](#)
 - [How often does the manager monitor the logs?](#)
 - [How long are the logs stored on the server?](#)
 - [How does this help me with regulatory compliance?](#)
 - [What is the CPU usage like on the agents?](#)
 - [From where can Wazuh get log messages?](#)
 - [Can I send firewall, VPN, authentication logs to Wazuh?](#)
 - [What information should Wazuh extract from my logs?](#)
 - [Can I ignore events that are not important?](#)

File integrity monitoring

File integrity monitoring is the capability that allows us to know if any file has changed. The component responsible for this task is called **syscheck**. This component compares the cryptographic checksum and other attributes of a known good file or Windows registry key against the checksum and attributes of the same after it has been modified.

Contents

- [How it works](#)
- [Configuration](#)
 - [Basic usage](#)
 - [Configuring scheduled scans](#)
 - [Configuring real-time monitoring](#)
 - [Configure to report changes](#)
 - [Configure to ignore files](#)
 - [Ignoring files via rules](#)
 - [Changing severity](#)
- [FAQ](#)
 - [How often does syscheck run?](#)
 - [What is the CPU usage like on the agents?](#)
 - [Where are all the checksums stored?](#)
 - [Can I ignore files in a directory?](#)
 - [Can Wazuh report changes in the content of a text file?](#)
 - [How does Wazuh verify the integrity of files?](#)
 - [Does Wazuh monitor any directories by default?](#)
 - [Can I force an immediate syscheck scan?](#)
 - [Does Syscheck start when Wazuh start?](#)
 - [Does Wazuh alert when a new file is created?](#)

Anomaly and malware detection

Anomaly detection refers to the action of finding patterns in the system that do not match the expected behavior. Once malware (e.g., a rootkit) is installed on a system, it modifies the system to be hidden from the user. Although malware uses a variety of techniques for this purpose, Wazuh uses a broad spectrum approach to finding anomalous patterns that indicate possible intruders.

The main component responsible for this task is *rootcheck*. *Syscheck* also plays an important role.

Contents

- [How it works](#)
 - [File integrity monitoring](#)
 - [Check running processes](#)
 - [Check hidden ports](#)
 - [Check unusual files and permissions](#)
 - [Check hidden files using system calls](#)
 - [Scan the `/dev` directory](#)
 - [Scan network interfaces](#)
 - [Rootkit checks](#)
- [Configuration](#)
 - [Basic example](#)
 - [Ignoring false positives](#)
- [FAQ](#)
 - [How often does rootcheck run?](#)
 - [How does rootcheck know the rootkit files to look for?](#)
 - [Does rootcheck inspect running processes?](#)
 - [What about hidden files?](#)

Monitoring security policies

Policy monitoring is the process of verifying that all systems conform to a set of predefined rules regarding configuration settings and approved application usage.

Wazuh uses two components to perform this task: *Rootcheck* and *OpenSCAP*.

Contents

- [Rootcheck](#)
 - [How it works](#)
 - [Configuration](#)
 - [FAQ](#)
- [OpenSCAP](#)
 - [How it works](#)
 - [Configuration](#)
 - [FAQ](#)

Monitoring system calls

The [Linux Audit system](#) provides a way to track security-relevant information on your machine. Based on preconfigured rules, Audit proves detailed real-time logging about the events that are happening on your system. This information is crucial for mission-critical environments to determine the violator of the security policy and the actions they performed.

Contents

- [How it works](#)
 - [Control rules](#)
 - [File System Rules](#)
 - [System Call Rules](#)
- [Configuration](#)
 - [Basic usage](#)
 - [Monitoring accesses to a directory](#)
 - [Monitoring user actions](#)
 - [Privilege escalation](#)

Command monitoring

Sometimes what we want to monitor is not included in the logs. To solve that problem, Wazuh incorporates the ability to monitor the output of specific commands and treat that output just like log file content.

Contents

- [How it works](#)
 - Configure Wazuh agents to accept remote commands from the manager
 - Configure a command to monitor
 - Process the output
- [Configuration](#)
 - Basic usage
 - Monitor running Windows processes
 - Disk space utilization
 - Check if the output changed
 - Load average
 - Detect USB Storage
- [FAQ](#)
 - Can I monitor commands on Linux or Windows?
 - What are the possibilities of this capability?
 - Can I check if an application is running on an agent?

Active response

Active response performs various countermeasures to address active threats such as blocking access to an agent from the threat source. This automated remediation is called active response in Wazuh.

Active response executes a script in response to being triggered by a specific alert based on alert level or rule group. Any number of scripts can be initiated in response to a trigger, however these responses should be carefully considered as poor implementation of rules and responses could increase the vulnerability of the system.

Contents

- [How it works](#)
 - [When is an active response triggered?](#)
 - [Where are active response actions executed?](#)
 - [Active response configuration](#)
 - [Default Active response scripts](#)
- [Configuration](#)
 - [Basic usage](#)
 - [Windows automatic remediation](#)
 - [Block an IP with PF](#)
 - [Add an IP to the iptables deny list](#)
 - [Active response for a specified period of time](#)
 - [Active response that will not be undone](#)
- [FAQ](#)
 - [Can I use a custom script for active response?](#)
 - [Can I configure active response for only one host?](#)
 - [Can active response remove the action after a time?](#)

Agentless monitoring

Agentless monitoring allows you to monitor devices or systems with no agent using SSH, such as: routers, firewalls, switches and linux/bsd systems.

Agentless monitoring allows users who have software installation restriction meet security and compliance requirements.

Alerts will be triggered when the checksum on the output changes and will show either the checksum or the exact diff output of the change.

Contents

- [How it works](#)
 - [Connection](#)
 - [Monitoring](#)
 - [Checking the setup](#)
 - [Alert](#)
- [Configuration](#)
 - [Integrity check BSD](#)
 - [Integrity check Linux](#)
 - [Generic Diff](#)
 - [Pix config](#)
- [FAQ](#)
 - [Is it possible to monitor the output of a command on a remote device?](#)
 - [Can I monitor directories on a remote system?](#)

Anti-flooding mechanism

This mechanism is designed to prevent disruptively large bursts of events on an agent from negatively impacting the network or the manager. It uses a leaky bucket queue that collects all generated events, and sends them to the manager at a rate not exceeding a configurable events per second threshold, helping to avoid the loss of events or unexpected behavior of Wazuh components.

Additionally, agent modules can be configured to limit the rate at which they produce events, reducing the risk of the leaky bucket's buffer being saturated.

- [Why it is an anti-flooding mechanism needed?](#)
- [How it works: Leaky bucket](#)
- [Use case: Leaky bucket](#)
- [Anti-flooding in agent modules](#)

Why it is an anti-flooding mechanism needed?

In the Wazuh architecture, Wazuh agents collect information from log files, command outputs, different kinds of scans, etc. They then send all the collected information, broken into individual events, to their manager. Without any congestion control, an agent could potentially send events at a rate as high as the system is physically capable of transmitting them (hundreds or thousands per second).

Due to this fact, a wrong configuration in an agent may generate enough events to saturate a network or its manager. Here are some misconfiguration scenarios that could lead to this problem:

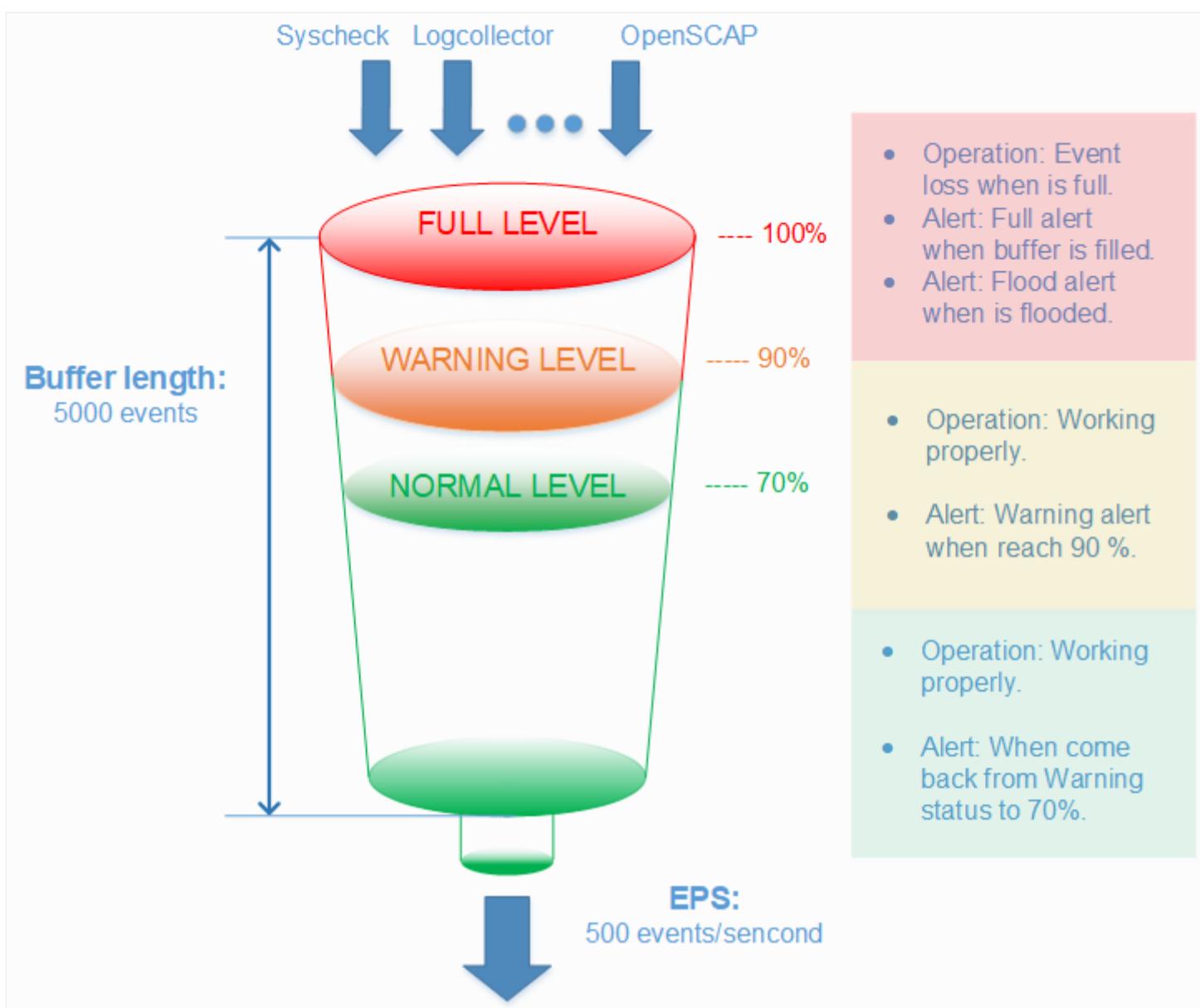
- Realtime FIM (Syscheck) of a directory with files that keep changing: Events are generated every time a file under a Syscheck-monitored directory changes. If Syscheck monitors a directory which changes constantly, it will generate a large amount of events. In addition, if the monitored directory contains any file to which Wazuh writes when it generates an event, like `/var/ossec/queue/`, it will cause an infinite loop.
- Windows Filtering Platform: A Windows firewall event (ID 5156) is generated each time an outbound network connection is allowed. When this event is enabled in Windows, and Wazuh is configured to monitor all Windows Security Log events, then when the agent connects its manager, it will generate a Windows firewall event that in turn causes the agent to connect again to its manager, leading to an infinite loop. Thus a disruptively high high rate of events is transmitted by the agent to its manager, causing problems at the agent, network, and/or manager level.
- Applications that retry on errors with no rate limiting: Some applications when encountering poorly anticipated error conditions like disk full, may generate an error log entry and retry over and over again hundreds of times per second, generative a massive event volume.

In order to better handle these kinds of situations, the following controls have been deployed:

- Agent-to-manager anti-flooding mechanism: This provides event congestion control with an agent-side leaky bucket queue to guard against saturation of the network or of the manager by an agent.
- Internal agent anti-flooding control: This mechanism uses internal limits in different components of the agent, making them limit the rate at which they generate events so as to avoid saturation.

How it works: Leaky bucket

As already described, the leaky bucket is a congestion control located in agents and focused on agent-to-manager communication. It collects events generated on an agent in a buffer of a configurable size (default 5000 events), and sends them to the manager at a rate no higher than a configurable number of events per second (default 500 EPS). These values need to take into account the needs of your specific agents, manager, and network environment. The following graphic shows how the bucket works.



There are several levels of control for the bucket with the aim of being aware of the buffer status, and being able to foresee a flooding situation.

- Warning alert: The first control will trigger an alert on the manager when the occupied capacity of the buffer has reached a certain threshold. By default it is set at 90 percent.
- Full alert: After the first control, if the buffer gets filled, another alert will be triggered on the manager. This new alert is more serious than a warning alert because **a full bucket will drop incoming events**.
- Flood alert: This alert is generated if more than a configurable amount of time passess between a full alert event and the buffer level dropping below the **warning level**.
- Normal alert: This alert is generated to announce that the buffer level has returned to normal (by default $\leq 70\%$) after having previously triggered a warning alert or higher.

The leaky bucket is totally configurable to adapt to any environment, with the use of the following configuration options.

Measured configuration

In the `<client_buffer>` section of [Local configuration](#) it is possible to disable the buffer, configure the size of the buffer (in number of events), and configure its throughput limit measured in EPS.

- Disable buffer: This parameter disables the use of the leaky bucket, resulting in no restriction on the rate of events transmitted by the agent to the manger. This is how previous versions of the agent worked.
- Buffer length: The buffer length is the maximum number of events that can be held in the leaky bucket at one time. It should be configured according to the expected rate at which an agent may generate events. This value is set to 5000 events by default, which is a generous buffer size for most environments.
- Events per second: This is the maximum rate at which events will be pulled from the agent's buffer and transmitted to its manager. The default is a generous 500 EPS, but this should be set taking into account the capacity of the network and the number of agents that a manager is serving.

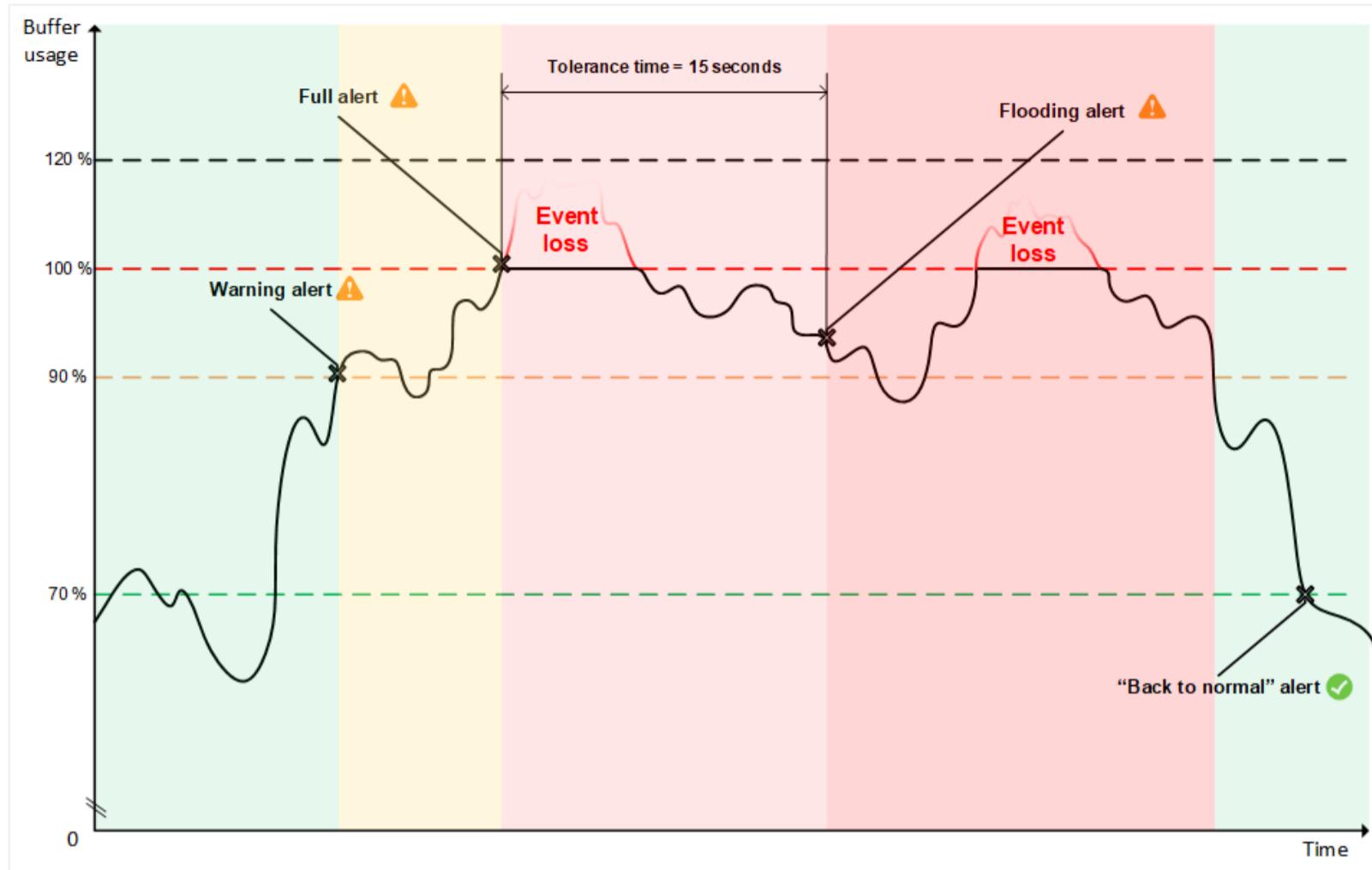
This configuration is also available in [Centralized configuration](#) which means it can be set in `agent.conf` with the aim of configuring agents' bucket options from the manager side. When an agent is configured by `agent.conf`, that configuration overrides its own local configuration. To allow the agent to have final say about a minimum number of EPS it will be allowed to transmit regardless of the EPS limit handed to it from the manager via `agent.conf`, another variable called `agent.min_eps` can be set in the agent's [Internal configuration](#).

Threshold configuration

In [Internal configuration](#), there are more advanced options related to buffer operation. Specifically, the warning and normal level thresholds, plus the tolerance time for triggering a flooding alert can be configured.

Use case: Leaky bucket

In this section, it will be shown how the leaky bucket acts facing an extreme situation. For this purpose, the following graphic shows different phases of the buffer's usage when it is receiving more events than expected, and how it acts step by step to manage the situation.



Normal status (green area)

As the graphic shows in the left area, the buffer is working normally, receiving and sending events. In this situation no buffer alerts are triggered on the manager. However, a large amount of events can provoke an increase in the buffer usage, causing it to reach the `warning level`, here set at 90 percent.

Warning status (orange area)

Once it has reached the `warning level`, an alert like this one is triggered on the manager side:

```
** Alert 1501604235.59814: - wazuh,agent_flooding,  
2017 Aug 01 18:17:15 (fedora) any->ossec-agent  
Rule: 521 (level 7) -> 'Agent buffer is close to an overflow state.'  
wazuh: Agent buffer: '90%'.  
wazuh: Agent buffer: '90%'.
```

Despite this alert, **no events have been dropped** because there is still **free space** in the buffer.

Reached 100% (light red area)

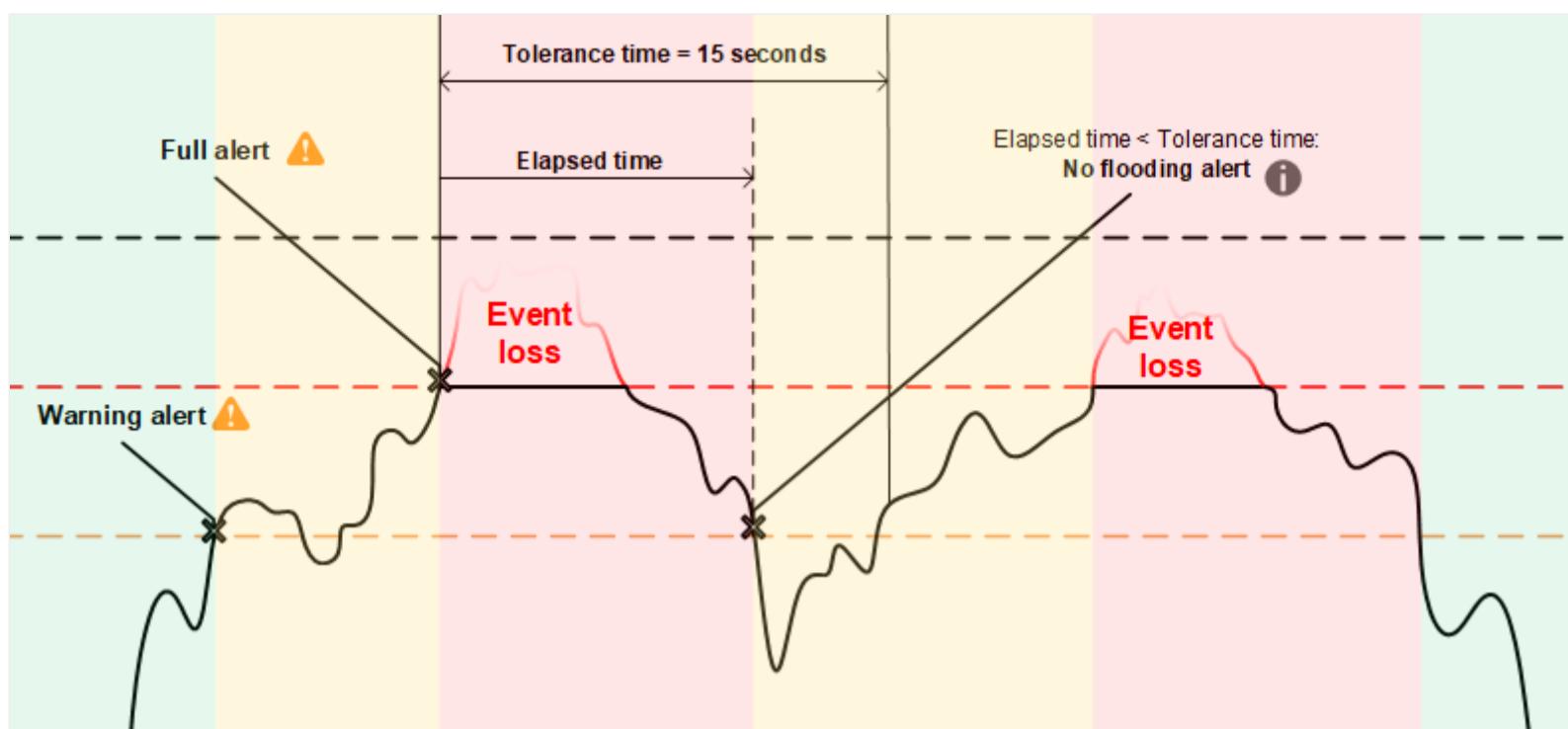
When the buffer continues receiving events faster than they are removed, it will eventually reach 100% of its capacity, triggering another alert on the manager:

```
** Alert 1501604236.60027: - wazuh,agent_flooding,  
2017 Aug 01 18:17:16 (fedora) any->ossec-agent  
Rule: 522 (level 9) -> 'Agent buffer is full. Events may be lost.'  
wazuh: Agent buffer: 'full'.  
wazuh: Agent buffer: 'full'.
```

It is important to understand that when the buffer is full, all newly arriving events **will be dropped** until free space opens up in the buffer again. For example, if in one second, 1000 events arrive to a full buffer with a throughput limit of 500 EPS, 500 of these events will be stored and the other 500 **will be dropped**.

When the buffer reaches 100% full, a timer is started, which is compared to the `tolerance time` set in `internal_options.conf`. At this point, two possible things could happen:

- Usage of the buffer decreases to below the **warning level** before the timer reaches the **tolerance time**. Consequently, no alert about flooding appears on the manager. The graphic illustrates this situation.



- Buffer usage stays above the **warning level** until the tolerance time has elapsed. Now it appears that the buffer may not come back to a normal status by itself. For that reason, a more severe alert is triggered on the manager.

Flooding status (red area)

As already mentioned, a severe alert is triggered when **tolerance time** has elapsed. This alert has the following appearance:

```
** Alert 1501604250.60248: mail - wazuh,agent_flooding,
2017 Aug 01 18:17:30 (fedora) any->ossec-agent
Rule: 523 (level 12) -> 'Agent buffer is flooded. Check the agent configuration.'
wazuh: Agent buffer: 'flooded'.
```

⚠ Warning

Note the alert description warns the user to check the agent since it is probable that it will not recover to a normal status by itself. Remember that **a flooded agent is surely dropping events**.

Returning to normal status

The right area of the graphic shows how the buffer returns to a normal status after it hits 100% full. This could happen because a module ceases generating excessive events, either because something has completed or because the offending module was shut down manually.

In order to let the manager know when an agent is working properly again, another alert is triggered when a maxed-out buffer's usage decreases back down to less than the **normal level** (70% by default). The alert looks like this:

```
** Alert 1501604257.60486: - wazuh,agent_flooding,
2017 Aug 01 18:17:37 (fedora) any->ossec-agent
Rule: 524 (level 3) -> 'Agent buffer is back to normal load.'
wazuh: Agent buffer: 'normal'.
```

When the bucket is in this status **no events are dropped**.

Anti-flooding in agent modules

In order to avoid agent buffer saturation followed by event loss, the event production rates of Wazuh agent daemons that could cause this saturation have been limited.

- Logcollector: If a log file is written faster than logcollector can read it, this can cause the agent trouble. For this reason, the agent will restrict itself to reading no more than a configurable maximum number of lines from the same file per read cycle.
- OpenSCAP Wodle: This module previously sent the entire set of scan results as soon as a scan would complete. Now it sends the scan information to the manager at a regulated speed so as to reduce the likelihood of maxing out the buffer.

These are advanced configurations located at [Internal configuration](#). The variables defined for this purpose are called `logcollector.max_lines` and `wazuh_modules.max_eps`. Be careful when changing these values.

Agent labels

This feature provides the opportunity to customize alerts information from agents, giving us the chance of include specific information of each agent which could be useful when dealing with alerts. In addition, in large environments it could be used to distinguish groups of agents by any common characteristic like their time zone, for example.

The following sections shows how this feature works as well as an use case.

- [How it works](#)
- [Use case](#)

How it works

Configuring labels for being shown at alerts is too simple. It can be done using a simple XML structure which allows to add information with the format `key:value`, how to configure them it is explained at [Labels section](#) section of `ossec.conf`. It is remarkable that it exists the possibility of using dots to split "key" names nesting them in JSON formatted alerts.

It is also allowed to centralize it using `agent.conf`. This way, from the manager side it is possible to set labels for specific agents. Note that whether it exists a duplicated label key in `ossec.conf` and `agent.conf`, the second one will override the first one. For more information about usage of centralized configuration see its dedicated section: [Centralized configuration](#).

In addition, more technical configuration is available in [Internal configuration](#). Particularly, `analysisd.label_cache_maxage` and `analysisd.show_hidden_labels`.

Use case

It is interesting to think about a scenario where the use of labels could be useful. Let's imagine we have a large environment deployed in Amazon Web Service (AWS) and monitored by Wazuh. In that situation, we want to know in the manager the following information about each agent when an alert is triggered:

- AWS instance-id.
- AWS Security group.
- Network IP address.
- Network MAC.
- Date of installation (hidden).

To include these labels into alerts of a specific agent, it is necessary to set the following configuration in `ossec.conf`:

```
<labels>
  <label key="aws.instance-id">i-052a1838c</label>
  <label key="aws.sec-group">sg-1103</label>
  <label key="network.ip">172.17.0.0</label>
  <label key="network.mac">02:42:ac:11:00:02</label>
  <label key="installation" hidden="yes">January 1st, 2017</label>
</labels>
```

Whether the configuration is set from the manager, the configuration has to be set in `agent.conf` using this format:

```
<agent_config name="92603de31548">
  <labels>
    <label key="aws.instance-id">i-052a1838c</label>
    <label key="aws.sec-group">sg-1103</label>
    <label key="network.ip">172.17.0.0</label>
    <label key="network.mac">02:42:ac:11:00:02</label>
    <label key="installation" hidden="yes">January 1st, 2017</label>
  </labels>
</agent_config>
```

When an alert is fired for this agent, the previous configuration will add to alerts the specified information:

```
** Alert 1488922301.778562: mail - ossec,syscheck,pci_dss_11.5,  
2017 Jun 07 13:31:43 (92603de31548) 192.168.66.1->syscheck  
aws.instance-id: i-052a1838c  
aws.sec-group: sg-1103  
network.ip: 172.17.0.0  
network.mac: 02:42:ac:11:00:02  
Rule: 550 (level 7) -> 'Integrity checksum changed.'  
Integrity checksum changed for: '/var/ossec/etc/ossec.conf'  
Size changed from '3663' to '3664'  
Old md5sum was: '98b351df146410f174a967d726f9965e'  
New md5sum is : '7f4f5846dcaa0013a91bd6d3ac4a1915'  
Old sha1sum was: 'c6368b866a835b15baf20976ae5ea7ea2788a30e'  
New sha1sum is : 'c959321244bdcec824ff0a32cad6d4f1246f53e9'
```

And the same alert in JSON format shows the advantage of using splitting in "key" names:

```
{
  "timestamp": "2017-03-07T13:31:41-0800",
  "rule": {
    "level": 7,
    "description": "Integrity checksum changed.",
    "id": "550",
    "firedtimes": 1,
    "groups": [
      "ossec",
      "syscheck"
    ],
    "pci_dss": [
      "11.5"
    ]
  },
  "agent": {
    "id": "001",
    "name": "92603de31548",
    "ip": "192.168.66.1",
    "labels": {
      "aws": {
        "instance-id": "i-052a1838c",
        "sec-group": "sg-1103"
      },
      "network": {
        "ip": "172.17.0.0",
        "mac": "02:42:ac:11:00:02"
      }
    }
  },
  "manager": {
    "name": "ubuntu"
  },
  "full_log": "Integrity checksum changed for: '/var/ossec/etc/ossec.conf' Size changed from '3663' to '3664'\nOld md5sum was: '98b351df146410f174a967d726f9965e' New md5sum is : '7f4f5846dcaa0013a91bd6d3ac4a1915' Old\nsha1sum was: 'c6368b866a835b15baf20976ae5ea7ea2788a30e' New sha1sum is :\n'c959321244bdcec824ff0a32cad6d4f1246f53e9' ,\n  "syscheck": {
    "path": "/var/ossec/etc/ossec.conf",
    "size_before": "3663",
    "size_after": "3664",
    "perm_after": "100640",
    "uid_after": "0",
    "gid_after": "999",
    "md5_before": "98b351df146410f174a967d726f9965e",
    "md5_after": "7f4f5846dcaa0013a91bd6d3ac4a1915",
    "sha1_before": "c6368b866a835b15baf20976ae5ea7ea2788a30e",
    "sha1_after": "c959321244bdcec824ff0a32cad6d4f1246f53e9",
    "event": "modified"
  },
  "decoder": {
    "name": "syscheck_integrity_changed"
  },
  "location": "syscheck"
}
}
```

Finally, when we have enabled email reports. It will be sent an email as follows:

Wazuh Notification.

2017 Mar 07 13:31:41

Received From: (92603de31548) 192.168.66.1->syscheck

Rule: 550 fired (level 7) -> "Integrity checksum changed."

Portion of the log(s):

```
aws.instance-id: i-052a1838c
aws.sec-group: sg-1103
network.ip: 172.17.0.0
network.mac: 02:42:ac:11:00:02
Integrity checksum changed for: '/var/ossec/etc/ossec.conf'
Old md5sum was: '98b351df146410f174a967d726f9965e'
New md5sum is : '7f4f5846dcaa0013a91bd6d3ac4a1915'
Old sha1sum was: 'c6368b866a835b15baf20976ae5ea7ea2788a30e'
New sha1sum is : 'c959321244bdcec824ff0a32cad6d4f1246f53e9'
```

Register Agents

Two requests are needed to register an agent using the API:

- POST /agents ([reference](#))
- PUT /agents/:agent_name ([reference](#))

We have prepared a few scripts in different languages to help with the task of registering an agent with the API:

- [Register an agent using a shell script.](#)
- [Register an agent using a Python script.](#)
- [Register an agent using a PowerShell script.](#)

Basically, the scripts perform the following steps:

Step 1: Add the agent to the manager.

```
$ curl -u foo:bar -k -X POST -d 'name>NewAgent&ip=10.0.0.8' https://API_IP:55000/agents
{"error":0,"data":"001"}
```

Step 2: Get the agent key.

```
$ curl -u foo:bar -k -X GET https://API_IP:55000/agents/001/key
{"error":0,"data":"MDAxIE5ld0FnZW50IDEwLjAuMC44IDM0MGQ1NjNkODQyNjcxMWIyYzUzZTE1MGIzYjEyYWVlMTU10DgxMzVhNDE3MWQ1Y2IzZDY4M2Y0YjA0ZWVjYzM="}
```

Step 3: Copy the key to the agent.

```
$ /var/ossec/bin/manage_agents -i
MDAxIE5ld0FnZW50IDEwLjAuMC44IDM0MGQ1NjNkODQyNjcxMWIyYzUzZTE1MGIzYjEyYWVlMTU10DgxMzVhNDE3MWQ1Y2IzZDY4M2Y0YjA0ZWVjYzM=
```

⚠ Warning

If you paste the command directly in the terminal, the agent key will be saved in the bash history. Use *manage_agents* without arguments or from a script.

Step 4: Restart the agent.

```
$ /var/ossec/bin/ossec-control restart
```


Listing Agents

The request [GET /agents](#) returns the list of available agents.

```
curl -u foo:bar -k http://127.0.0.1:55000/agents?pretty
```

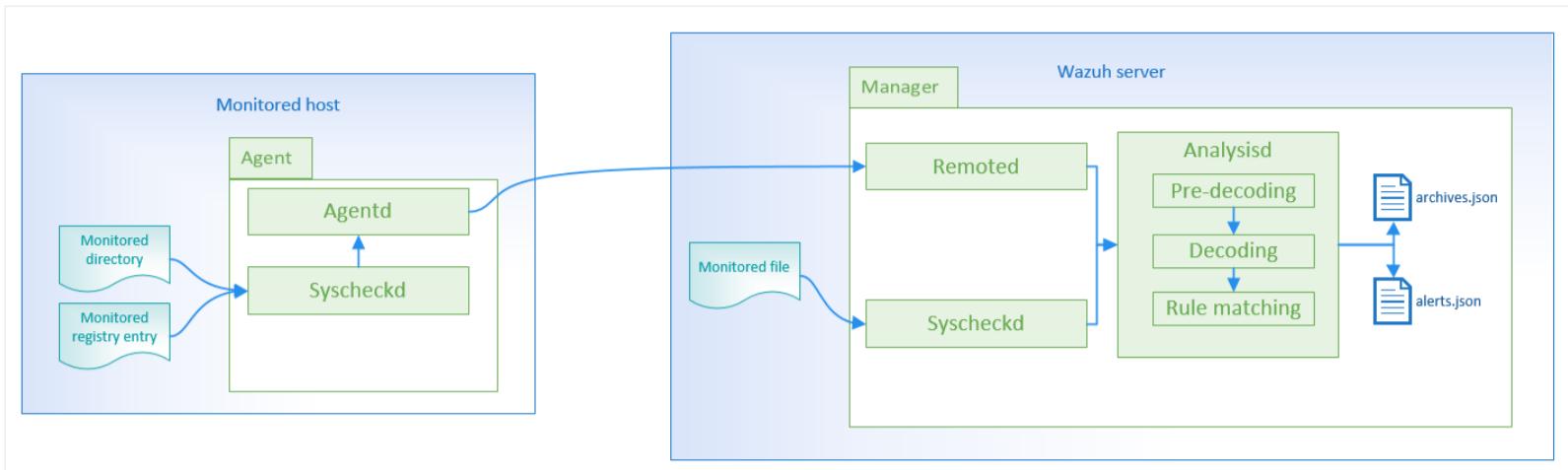
```
{
  "error": 0,
  "data": {
    "totalItems": 15,
    "items": [
      {
        "status": "Active",
        "ip": "127.0.0.1",
        "id": "000",
        "name": "vpc-ossec-manager"
      },
      {
        "status": "Active",
        "ip": "10.0.0.121",
        "id": "003",
        "name": "vpc-agent-debian"
      },
      {
        "status": "Active",
        "ip": "10.0.0.126",
        "id": "005",
        "name": "vpc-agent-ubuntu-public"
      },
      {
        "status": "Active",
        "ip": "10.0.0.124",
        "id": "006",
        "name": "vpc-agent-windows"
      },
      {
        "...": ...
      }
    ]
  }
}
```


Remove Agents

The request `DELETE /agents/:agent_id` removes the specified agent.

```
$ curl -u foo:bar -k -X DELETE http://127.0.0.1:55000/agents/002
{"error":0,"data":"Agent removed"}
```


How it works



1. Wazuh agent scans the system and sends the checksums and attributes of the monitored files and Windows registry keys to the Wazuh manager. We can configure:

- **Frequency:** By default, syscheck runs every 6 hours, but this is entirely configurable.
- **Real-time monitoring:** Wazuh supports real-time file integrity monitoring on servers running Windows or Linux. Note that the real-time option is only configurable for directories, not for single files.

2. Wazuh manager stores the checksums/attributes and looks for modifications by comparing the new values to the old values. It's possible to configure syscheck to report a diff summary of the actual changes made to text files.

3. An alert is generated if changes are detected on monitored files and/or registry keys.

It's possible to handle false positives using configuration options like `ignore`. You can also create rules that profile files that are to be excluded from FIM alerts.

Alert example, generated by **syscheck**:

```
** Alert 1460948255.25442: mail - ossec,syscheck,pci_dss_11.5,
2016 Apr 17 19:57:35 (ubuntu) 10.0.0.144->syscheck
Rule: 550 (level 7) -> 'Integrity checksum changed.'
Integrity checksum changed for: '/test/hello'
Size changed from '12' to '17'
Old md5sum was: 'e59ff97941044f85df5297e1c302d260'
New md5sum is : '7947eba5d9cc58d440fb06912e302949'
Old sha1sum was: '648a6a6ffffdaa0badb23b8baf90b6168dd16b3a'
New sha1sum is : '379b74ac9b2d2b09ff6ad7fa876c79f914a755e1'
```


Configuration

1. [Basic usage](#)
2. [Configuring scheduled scans](#)
3. [Configuring real-time monitoring](#)
4. [Configure to report changes](#)
5. [Configure to ignore files](#)
6. [Ignoring files via rules](#)
7. [Changing severity](#)

Basic usage

Syscheck is configured in `ossec.conf`. If you want more information about detailed configuration options, go to [Syscheck](#). Usually you use the following sections: `frequency`, `directories`, `ignore`, `alert_new_files`

To configure syscheck, a list of files and directories must be provided. The `check_all` option checks md5, sha1, owner, and permissions of the file.

```
<syscheck>
  <directories check_all="yes">/etc,/usr/bin,/usr/sbin</directories>
  <directories check_all="yes">/root/users.txt,/bsd,/root/db.html</directories>
</syscheck>
```

Configuring scheduled scans

Syscheck has an option to configure the frequency to scan the system. This is the `frequency` option. In this example we configure syscheck to run every 10 hours.

```
<syscheck>
  <frequency>36000</frequency>
  <directories>/etc,/usr/bin,/usr/sbin</directories>
  <directories>/bin,/sbin</directories>
</syscheck>
```

Configuring real-time monitoring

Real-time monitoring is configured with the `realtime` option. This option only works with directories, not for individual files. Real-time change detection is paused during periodic syscheck scans, and reactivates as soon as scans complete.

```
<syscheck>
  <directories check_all="yes" realtime="yes">c:/tmp</directories>
</syscheck>
```

Configure to report changes

Using `report_changes` option, we can see what specifically changed in text files. Be careful about which folders you set up to `report_changes`, because in order to report changes, Wazuh must copy every single file you want to monitor to a private location.

```
<syscheck>
  <directories check_all="yes" realtime="yes" report_changes="yes">/test</directories>
</syscheck>
```

Configure to ignore files

Files and directories can be omitted using the ignore option (or registry_ignore for Windows registry entries): In order to avoid false positives, syscheck can be configured to ignore certain files that we don't want to monitor with `ignore` tag (or registry_ignore for Windows registry entries).

```
<syscheck>
  <ignore>/etc/random-seed</ignore>
  <ignore>/root/dir</ignore>
  <ignore type="sregex">.log$|.tmp</ignore>
</syscheck>
```

Ignoring files via rules

It is possible to ignore files using rules:

```
<rule id="100345" level="0">
  <if_group>syscheck</if_group>
  <match>/var/www/htdocs</match>
  <description>Ignore changes to /var/www/htdocs</description>
</rule>
```

Changing severity

With a custom rule it is possible to alter the level of a syscheck alert when changes to a specific file or file pattern are detected:

```
<rule id="100345" level="12">
  <if_group>syscheck</if_group>
  <match>/var/www/htdocs</match>
  <description>Changes to /var/www/htdocs - Critical file!</description>
</rule>
```

FAQ

1. [How often does syscheck run?](#)
2. [What is the CPU usage like on the agents?](#)
3. [Where are all the checksums stored?](#)
4. [Can I ignore files in a directory?](#)
5. [Can Wazuh report changes in the content of a text file?](#)
6. [How does Wazuh verify the integrity of files?](#)
7. [Does Wazuh monitor any directories by default?](#)
8. [Can I force an immediate syscheck scan?](#)
9. [Does Syscheck start when Wazuh start?](#)
10. [Does Wazuh alert when a new file is created?](#)

How often does syscheck run?

Syscheck frequency is configurable by the user with [frequency](#). By default is configured to run every 6 hours.

What is the CPU usage like on the agents?

Syscheck scans are designed to run slowly to avoid too much CPU or memory use.

Where are all the checksums stored?

All the checksums are stored on the manager [/var/ossec/queue/syscheck](#)

Can I ignore files in a directory?

Yes, you can use the [ignore](#) option to avoid false positives. Example: [ignore-false-positives](#)

Can Wazuh report changes in the content of a text file?

Yes, this is possible with the [report_changes](#) option. For [directories](#) only. This option gives us the exact content that has been changed in a text file. Be selective about which folders you use [report_changes](#) on, because this requires syscheck to copy every single file you want to monitor with [report_changes](#) to a private location for comparison purposes. Example: [report changes](#)

How does Wazuh verify the integrity of files?

Wazuh manager stores and looks for modifications to all the checksums and file attributes received from the agents for the monitored files. Wazuh manager compares the new checksums/attributes against the stored ones. An alert is generated if anything changes.

Does Wazuh monitor any directories by default?

Yes. By default Wazuh monitors [/etc](#), [/usr/bin](#), [/usr/sbin](#), [/bin](#) and [/sbin](#) on Unix-like systems and [C:\Windows\System32](#) on Windows.

Can I force an immediate syscheck scan?

Yes, you can force an agent to perform a system integrity check with ::

`/var/ossec/bin/agent_control -r -a /var/ossec/bin/agent_control -r -u <agent_id>`

More info at [Ossec control section](#)

Does Syscheck start when Wazuh start?

By default syscheck scan when Wazuh start, but you can change this with the [scan_on_start](#) option

Does Wazuh alert when a new file is created?

Yes, but you need to configure it. Use the [alert_new_files](#) option

Rootcheck

Wazuh monitors configuration files to ensure they are compliant with your security policies, standards or hardening guides. Agents perform periodic scans to detect applications that are known to be vulnerable, unpatched, misconfigured.

- [How it works](#)
- [Configuration](#)
 - [Basic usage](#)
 - [Configure periodic scans](#)
 - [Root access to SSH](#)
- [FAQ](#)
 - [Can I specify my own audit file for policy monitoring?](#)

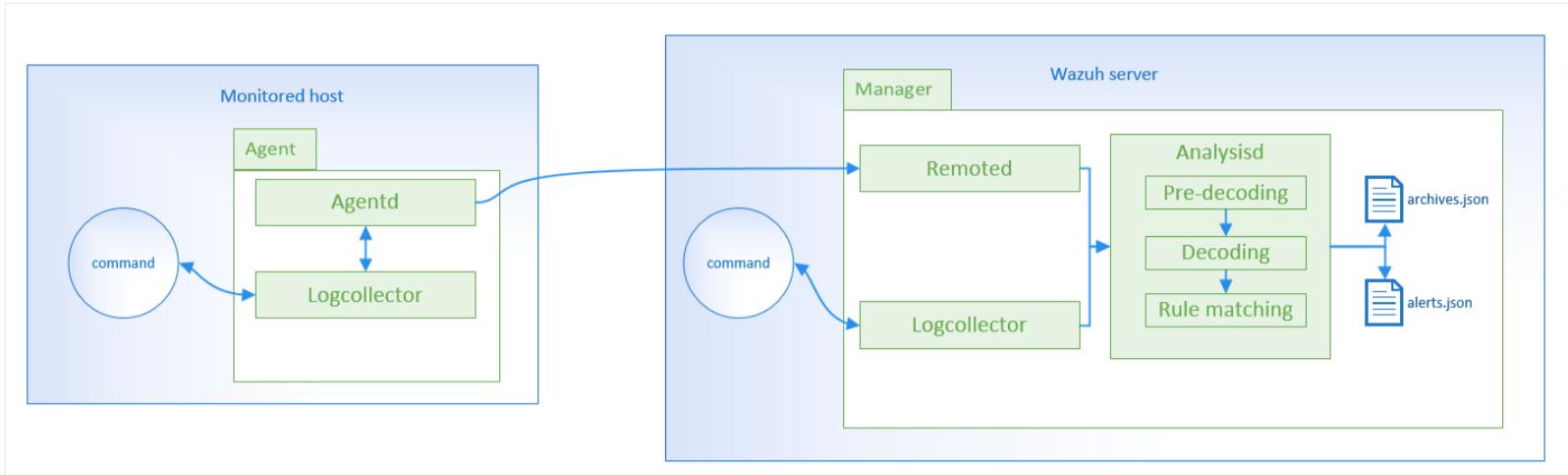
OpenSCAP

The **OpenSCAP wodle** is an integration of [OpenSCAP](#) with *Wazuh HIDS* that provides the ability to perform configuration and vulnerability scans of an agent. It is primarily used for:

- Verifying **security compliance**: OpenSCAP policies define the requirements that all systems in an organization must meet in order to be in line with applicable *security policies* and/or *security benchmarks*.
- Performing **vulnerability assessments**: OpenSCAP identifies and classifies vulnerabilities in a system.
- Performing **specialized assessments**: OpenSCAP can perform specific custom system checks (i.e., checking for suspicious file names and suspicious file locations.)

- [How it works](#)
 - Requirements
 - Default policies
 - Wodle flow
- [Configuration](#)
 - Basic usage
 - Evaluate PCI-DSS compliance on RHEL7
 - Auditing Security Vulnerabilities of Red Hat Products
 - Overwriting the timeout
 - Using profiles
 - Using CPE dictionary
 - Using IDs
- [FAQ](#)
 - Is there a noticeable performance impact when the OpenSCAP wodle is enabled on an agent?
 - Are evaluations executed in parallel?
 - How does the interval work?
 - Are the policies evaluated when OSSEC starts?
 - Where are the policies?

How it works



The following is required to set up the monitoring of a specific command's output on agents:

Configure Wazuh agents to accept remote commands from the manager

Agents do have the ability to run commands pushed from the manager (via the files in the `shared` directory). Before this feature can be used, it needs to be explicitly allowed at an agent level. This can be done by setting the `logcollector.remote_commands` in `local_internal_options.conf` on each agent.

Example:

```
# Logcollector - Whether or not to accept remote commands from the manager
logcollector.remote_commands=1
```

Configure a command to monitor

The commands to run and monitor, can be configured either inside the local `ossec.conf` of individual agents, but the ideal location would be the appropriate config section of `agent.conf` on the manager.

Example:

```
<localfile>
  <log_format>full_command</log_format>
  <command>.....</command>
  <frequency>120</frequency>
</localfile>
```

Process the output

After configuring the system to monitor the command's output as if it were log data, we can create custom rules like for [Log analysis](#), in order to process the output and alert when is needed.

Configuration

1. [Basic usage](#)
2. [Monitor running Windows processes](#)
3. [Disk space utilization](#)
4. [Check if the output changed](#)
5. [Load average](#)
6. [Detect USB Storage](#)

Basic usage

Command monitoring is configured in the [localfile](#) section of [ossec.conf](#). It can also be centrally configured in [agent.conf](#).

Monitor running Windows processes

Imagine you want to monitor the running process and alert if an important one is not running.

Example with notepad.exe:

1. Configure the agent to accept remote commands from the manager, in the agent's **local_internal_options.conf**.

```
# Logcollector - Whether or not to accept remote commands from the manager
logcollector.remote_commands=1
```

2. Define the command to list running processes, in the manager's **agent.conf** file

```
<localfile>
  <log_format>full_command</log_format>
  <command>tasklist</command>
  <frequency>120</frequency>
</localfile>
```

The **frequency** defines how often the command will be run (in seconds).

3. Define the rules

```
<rule id="100010" level="6">
  <if_sid>530</if_sid>
  <match>^ossec: output: 'tasklist'</match>
  <description>Important process not running.</description>
  <group>process_monitor,</group>
</rule>
<rule id="100011" level="0">
  <if_sid>100010</if_sid>
  <match>notepad.exe</match>
  <description>Processes running as expected</description>
  <group>process_monitor,</group>
</rule>
```

The first rule (100010) will generate an alert ("Important process not running"), unless it is overridden by its child rule (100011) that matches *notepad.exe* in the command output. You could add as many additional child rules as you need to enumerate all important processes you want to monitor. You could also adapt this example to monitor Linux process by changing the `<command>` from "tasklist" to a Linux command that lists processes, like "ps -auxw".

Disk space utilization

We can configure the `dh` command in the manager's **agent.conf** file or in the agent's **ossec.conf** file:

```
<localfile>
  <log_format>command</log_format>
  <command>df -P</command>
</localfile>
```

Wazuh already has a rule to monitor this:

```
<rule id="531" level="7" ignore="7200">
  <if_sid>530</if_sid>
  <match>ossec: output: 'df -P': /dev/</match>
  <regex>100%</regex>
  <description>Partition usage reached 100% (disk space monitor).</description>
  <group>low_diskspace,pci_dss_10.6.1,</group>
</rule>
```

The system will alert once the disk space usage on any partition reaches 100%

Check if the output changed

In this case we use the Linux “netstat” command and the [check_diff option](#) to monitor for changes in listening tcp sockets.

Configuration in [agent.conf](#) or [ossec.conf](#):

```
<localfile>
  <log_format>full_command</log_format>
  <command>netstat -tan |grep LISTEN|grep -v 127.0.0.1</command>
</localfile>
```

Wazuh already has a rule to monitor this:

```
<rule id="533" level="7">
  <if_sid>530</if_sid>
  <match>ossec: output: 'netstat -tan'</match>
  <check_diff />
  <description>Listened ports status (netstat) changed (new port opened or closed).</description>
  <group>pci_dss_10.2.7,pci_dss_10.6.1,</group>
</rule>
```

If the output changes, the system will generate an alert, indicating a network listener has disappeared or a new one has appeared, which could indicate something is broken or a network backdoor has been installed.

Load average

You can configure Wazuh to monitor the Linux [uptime](#) command and alert when it is higher than a given threshold, like 2 in this example.

Configuration in [agent.conf](#) or [ossec.conf](#):

```
<localfile>
  <log_format>command</log_format>
  <command>uptime</command>
</localfile>
```

And the custom rule to alert when is higher than 2:

```
<rule id="100101" level="7" ignore="7200">
  <if_sid>530</if_sid>
  <match>ossec: output: 'uptime': </match>
  <regex>load averages: 2.</regex>
  <description>Load average reached 2..</description>
</rule>
```

Detect USB Storage

Wazuh can be configured to alert when a USB storage device is connected. This example is for a Windows agent.

Configure your agent to monitor the USBSTOR registry entry, by adding this to the manager's `agent.conf` :::

```
<agent_config os="Windows">
  <localfile>
    <log_format>full_command</log_format>
    <command>reg QUERY HKLM\SYSTEM\CurrentControlSet\Enum\USBSTOR</command>
  </localfile>
</agent_config>
```

Next create your custom rule:

```
<rule id="140125" level="7">
  <if_sid>530</if_sid>
  <match>ossec: output: 'reg QUERY</match>
  <check_diff />
  <description>New USB device connected</description>
</rule>
```

FAQ

1. [Can I monitor commands on Linux or Windows?](#)
2. [What are the possibilities of this capability?](#)
3. [Can I check if an application is running on an agent?](#)

Can I monitor commands on Linux or Windows?

You can monitor command output on both Linux and Windows systems.

What are the possibilities of this capability?

Some examples: Disk space utilization, detection if an important process is running or not, load average, change in network listeners...

Can I check if an application is running on an agent?

Yes, it's possible to monitor running applications. [Example](#)

How it works

Connection

First of all, agentless monitoring must be enabled:

```
/var/ossec/bin/ossec-control enable agentless
```

In order to connect the manager to the device using SSH authentication, the following script should be used: `register_host.sh`, which is located in: `/var/ossec/agentless/`. This script has two options: `list` and `add`.

Using the `list` option will list all hosts already included.

```
/var/ossec/agentless/register_host.sh list
```

Using the `add` option will specify a new device to be added to the manager. `NOPASS` may be entered as the password to use public key authentication rather than using a password. For Cisco devices, such as routers or firewalls, `enablepass` should be used to specify the enable password.

```
/var/ossec/agentless/register_host.sh add root@example_address.com example_password [enablepass]
```

Public key authentication can be used with the following command:

```
sudo -u ossec ssh-keygen
```

Once created, the public key must be copied into the remote device.

Monitoring

After devices have been added to the list, the manager must be configured to monitor them. To view additional configuration options for the `ossec.conf` file, please refer to [agentless](#).

The four types of agentless checks.

BSD Integrity Check

For BSD systems, set the `type` as `ssh_integrity_check_bsd` as referenced below. A space-separated list of directories may be referenced in the configuration section using the arguments tag. Using this configuration, Wazuh will do an integrity check on the remote box.

```
<agentless>
  <type>ssh_integrity_check_bsd</type>
  <frequency>20000</frequency>
  <host>root@test.com</host>
  <state>periodic</state>
  <arguments>/bin /var/</arguments>
</agentless>
```

Linux Integrity Check

For Linux systems, set the `type` as `ssh_integrity_check_linux` as referenced below. A space-separated list of directories may be referenced in the configuration section using the arguments tag. Using this configuration, Wazuh will do an integrity check on the remote box.

```
<agentless>
  <type>ssh_integrity_check_linux</type>
  <frequency>36000</frequency>
  <host>root@test.com</host>
  <state>periodic</state>
  <arguments>/bin /etc/ /sbin</arguments>
</agentless>
```

Generic Diff

A set of commands can also be configured to run on a remote device. Wazuh will alert you if the output of those commands changes. In order to use this option, set `type` as `ssh_generic_diff`, as shown below.

```
<agentless>
  <type>ssh_generic_diff</type>
  <frequency>20000</frequency>
  <host>root@test.com</host>
  <state>periodic_diff</state>
  <arguments>ls -la /etc; cat /etc/passwd</arguments>
</agentless>
```

Note

To use `su` in a command as an argument, `use_su` must be set before the hostname. In the previous example, this would appear as:

```
<host>use_su root@example_address.com</host>
```

Pix config

This option will alert if a Cisco PIX/router configuration changes. Set the `type` to `ssh_pixconfig_diff`, as shown below.

```
<agentless>
  <type>ssh_pixconfig_diff</type>
  <frequency>36000</frequency>
  <host>pix@pix/fw.local</host>
  <state>periodic_diff</state>
</agentless>
```

Checking the setup

Finally, the `expect` package must be present on the manager for this feature to work.

When the `expect` package is present and Wazuh is restarted, the following is shown in the `/var/ossec/logs/ossec.log` file:

```
ossec-agentlessd: INFO: Test passed for 'ssh_integrity_check_linux'.
```

When Wazuh has connected to the remote device, the following will be shown in the same log file:

```
ossec-agentlessd: INFO: ssh_integrity_check_linux: root@example_adress.com: Starting.
ossec-agentlessd: INFO: ssh_integrity_check_linux: root@example_adress.com: Finished.
```

Alert

Once configured as above, Wazuh alerts will be triggered when changes occur within the directories, configuration or outputs based on the above examples:

Examples of alerts are as follows:

Integrity check BSD/Linux example alert:

```
** Alert 1486811998.93230: - ossec,syscheck,pci_dss_11.5,
2017 Feb 11 03:19:58 ubuntu->(ssh_integrity_check_linux) root@192.168.1.3->syscheck
Rule: 550 (level 7) -> 'Integrity checksum changed.'
Integrity checksum changed for: '/etc/.hidden'
Size changed from '0' to '10'
Old md5sum was: 'd41d8cd98f00b204e9800998ecf8427e'
New md5sum is : 'cc7bd56aba1122d0d5f9c7ef7f96de23'
Old sha1sum was: 'da39a3ee5e6b4b0d3255bfef95601890af80709'
New sha1sum is : 'b570fbdf7d6ad1d1e95ef57b74877926e2cdf196'

File: /etc/.hidden
Old size: 0
New size: 10
New permissions: 1204
New user: 0
New group: 0
Old MD5: d41d8cd98f00b204e9800998ecf8427e
New MD5: cc7bd56aba1122d0d5f9c7ef7f96de23
Old SHA1: da39a3ee5e6b4b0d3255bfef95601890af80709
New SHA1: b570fbdf7d6ad1d1e95ef57b74877926e2cdf196
```

Generic Diff example alert:

```
** Alert 1486811190.88243: - ossec,syscheck,agentless,pci_dss_11.5,pci_dss_10.6.1,
2017 Feb 11 03:06:30 ubuntu->(ssh_generic_diff) root@192.168.1.3->agentless
Rule: 555 (level 7) -> 'Integrity checksum for agentless device changed.'
ossec: agentless: Change detected:
3c3
< drwxr-xr-x. 77 root root 8192 Feb 27 10:44 .
---
> drwxr-xr-x. 77 root root 8192 Feb 27 10:47 .
176a177
> -rw-r--r--. 1 root root 0 Feb 27 10:47 test
```

Configuration

1. [Integrity check BSD](#)
2. [Integrity check Linux](#)
3. [Generic Diff](#)
4. [Pix config](#)

Agentless monitoring is configured in the `ossec.conf` file in the section [Agentless](#).

Integrity check BSD

In this example, the configuration is set to monitor the `/bin` and `/var` directories

```
<agentless>
  <type>ssh_integrity_check_bsd</type>
  <frequency>20000</frequency>
  <host>root@test.com</host>
  <state>periodic</state>
  <arguments>/bin /var</arguments>
</agentless>
```

Integrity check Linux

For Linux systems, set the `type` as `ssh_integrity_check_linux` as referenced below. A space-separated list of directories may be referenced in the configuration section using the arguments tag. Using this configuration, Wazuh will do an integrity check on the remote box.

The below example is configured to monitor the `/bin` and `/etc /sbin` directories

```
<agentless>
  <type>ssh_integrity_check_linux</type>
  <frequency>36000</frequency>
  <host>root@test.com</host>
  <state>periodic</state>
  <arguments>/bin /etc/ /sbin</arguments>
</agentless>
```

Generic Diff

In this example, the configuration is set to execute `ls -la /etc` and `cat /etc/passwd` commands every 20000 seconds. An alert will be triggered if the output of those commands change.

```
<agentless>
  <type>ssh_generic_diff</type>
  <frequency>20000</frequency>
  <host>root@test.com</host>
  <state>periodic_diff</state>
  <arguments>ls -la /etc; cat /etc/passwd</arguments>
</agentless>
```

Pix config

In this example, the configuration is set to trigger an alert when a Cisco PIX or router configuration changes.

```
<agentless>
  <type>ssh_pixconfig_diff</type>
  <frequency>36000</frequency>
  <host>pix@pix/fw.local</host>
  <state>periodic_diff</state>
</agentless>
```

FAQ

1. [Is it possible to monitor the output of a command on a remote device?](#)
2. [Can I monitor directories on a remote system?](#)

Is it possible to monitor the output of a command on a remote device?

Yes, using the `ssh_generic_diff` option [example](#)

Can I monitor directories on a remote system?

Yes, using the `ssh_integrity_check_bsd` or `ssh_integrity_check_linux` options.

How it works

The [Security Content Automation Protocol \(SCAP\)](#) is a specification for expressing and manipulating security data in standardized ways. SCAP jointly uses several specifications in order to automate continuous monitoring, vulnerability management, and reporting on results of security compliance scans.

Components of the security compliance evaluation process:

- **SCAP scanner:** This is an application that reads a SCAP policy and checks whether or not the system is compliant with it. There are many [tools](#) to scan your systems against SCAP policies. This wodle is an integration with the NIST-certified scanner called [OpenSCAP](#).
- **Security policies (SCAP content):** These determine how a system must be set up and what to check for. These policies contain machine-readable descriptions of the rules which your system will be required to follow.
- **Profiles:** Each security policy can contain multiple profiles, which provide sets of rules and values in line with a specific security baseline. You can think of a profile as a particular subset of rules within the policy; the profile determines which rules defined in the policy will be actually used and what values will be used during the evaluation.
- **Evaluation (scan):** This is the process performed by the OpenSCAP scanner on an agent according to a specific security policy and profile. It usually takes only a few minutes, depending on the number of rules selected in the profile.

Requirements

This wodle is executed on the agent, so each agent must meet the following requirements:

OpenSCAP In order to perform SCAP evaluations, we need the scanner. As we mentioned above, we use OpenSCAP. You can install it with this command:

a. For RPM-based distributions:

```
sudo yum install openscap-scanner
```

b. For Debian-based distributions:

```
sudo apt-get install libopenscap8 xsltproc
```

Python 2.6+ Python is a core part of this wodle. Currently all Linux distributions come with python, so it should not be an inconvenience.

Default policies

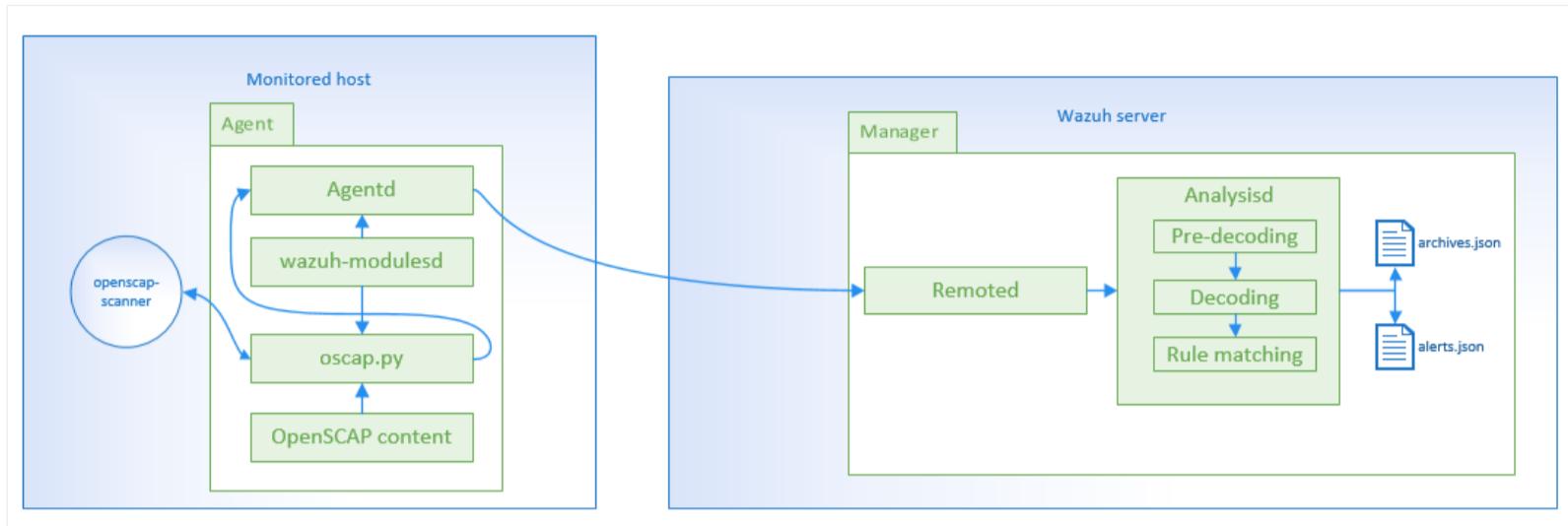
These are the Security Policy includes by default on Wazuh:

SO	Version	File name	Main profiles	Vulnerability assessment
CentOS	6	ssg-centos-6-ds.xml	Server, PCI	N/A
	7	ssg-centos-7-ds.xml	Common, PCI	N/A
RedHat	6	ssg-rhel-6-ds.xml	Server, PCI	N/A
		cve-redhat-6-ds.xml	N/A	Y
	7	ssg-rhel-7-ds.xml	Common , PCI	N/A
		cve-redhat-7-ds.xml	N/A	Y
Debian	8	ssg-debian-8-ds.xml	Common	N/A
Ubuntu	xenial	ssg-ubuntu-1604-ds.xml	Common	N/A
	trusty	cve-debian-oval.xml	N/A	Y

SO	Version	File name	Main profiles	Vulnerability assessment
	precise	cve-debian-oval.xml	N/A	Y
Fedora	24	ssg-fedora-ds.xml	Common	N/A

Each agent must have its policies in `/var/ossec/wodles/oscapp/content`.

Wodle flow



The agent will run *openscap-scanner* periodically according to the configuration. Each result of the scan will be sent to the Manager and it will generate an alert if the status of the result is fail. It is possible to tuning the rules to send the pass result too.

```
{  
  "timestamp": "2017-03-20T15:59:43-0700",  
  "rule": {  
    "level": 7,  
    "description": "OpenSCAP: Set Lockout Time For Failed Password Attempts (not passed)",  
    "id": "81530",  
    "firedtimes": 5,  
    "groups": [  
      "oscap",  
      "oscap-result"  
    ],  
    "pci_dss": [  
      "2.2"  
    ]  
  },  
  "agent": {  
    "id": "1040",  
    "name": "ip-10-0-0-76",  
    "ip": "10.0.0.76"  
  },  
  "manager": {  
    "name": "vpc-ossec-manager"  
  },  
  "full_log": "oscap: msg: \"xccdf-result\", scan-id: \"10401490050781\", content: \"ssg-centos-7-ds.xml\", title: \"Set Lockout Time For Failed Password Attempts\", id: \"xccdf_org.ssgproject.content_rule_accounts_passwords_pam_faillock_unlock_time\", result: \"fail\", severity: \"medium\", description: \"To configure the system to lock out accounts after a number of incorrect login attempts and require an administrator to unlock the account using pam_faillock.so, modify the content of both /etc/pam.d/system-auth and /etc/pam.d/password-auth as follows: add the following line immediately before the pam_unix.so statement in the AUTH section: auth required pam_faillock.so preauth silent deny= unlock_time= fail_interval= add the following line immediately after the pam_unix.so statement in the AUTH section: auth [default=die] pam_faillock.so authfail deny= unlock_time= fail_interval= add the following line immediately before the pam_unix.so statement in the ACCOUNT section: account required pam_faillock.so\", rationale: \"Locking out user accounts after a number of incorrect attempts prevents direct password guessing attacks. Ensuring that an administrator is involved in unlocking locked accounts draws appropriate attention to such situations.\\" references: \"AC-7(b)  
(http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), 47  
(http://iase.disa.mil/stigs/cci/Pages/index.aspx)\", identifiers: \"CCE-26884-7 (http://cce.mitre.org)\\", oval-id: \"oval:ssg:def:166\", benchmark-id: \"xccdf_org.ssgproject.content_benchmark_RHEL-7\", profile-id: \"xccdf_org.ssgproject.content_profile_pci-dss\", profile-title: \"PCI-DSS v3 Control Baseline for CentOS Linux 7\".",  
  "oscap": {  
    "scan": {  
      "id": "10401490050781",  
      "content": "ssg-centos-7-ds.xml",  
      "benchmark": {  
        "id": "xccdf_org.ssgproject.content_benchmark_RHEL-7"  
      },  
      "profile": {  
        "id": "xccdf_org.ssgproject.content_profile_pci-dss",  
        "title": "PCI-DSS v3 Control Baseline for CentOS Linux 7"  
      }  
    },  
    "check": {  
      "title": "Set Lockout Time For Failed Password Attempts",  
      "id": "xccdf_org.ssgproject.content_rule_accounts_passwords_pam_faillock_unlock_time",  
      "result": "fail",  
      "severity": "medium",  
      "description": "To configure the system to lock out accounts after a number of incorrect login attempts and require an administrator to unlock the account using pam_faillock.so, modify the content of both /etc/pam.d/system-auth and /etc/pam.d/password-auth as follows: add the following line immediately before the pam_unix.so statement in the AUTH section: auth required pam_faillock.so preauth silent deny= unlock_time= fail_interval= add the following line immediately after the pam_unix.so statement in the AUTH section: auth [default=die] pam_faillock.so authfail deny= unlock_time= fail_interval= add the following line immediately before the pam_unix.so statement in the ACCOUNT section: account required pam_faillock.so",  
      "rationale": "Locking out user accounts after a number of incorrect attempts prevents direct password guessing attacks. Ensuring that an administrator is involved in unlocking locked accounts draws appropriate attention to such situations."  
    }  
  }  
}
```

```

appropriate attention to such situations.",  

    "references": "AC-7(b) (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf),  

47 (http://iase.disa.mil/stigs/cci/Pages/index.aspx),"  

    "identifiers": "CCE-26884-7 (http://cce.mitre.org)",  

    "oval": {  

        "id": "oval:ssg:def:166"  

    }  

},  

"decoder": {  

    "parent": "oscap",  

    "name": "oscap"  

},  

"location": "wodle_open-scap"
}

```

When the scan finishes, a report event is sent which generates an alert:

```

{
    "timestamp": "2017-03-20T15:59:43-0700",
    "rule": {
        "level": 5,
        "description": "OpenSCAP Report overview: Score less than 80",
        "id": "81542",
        "firedtimes": 2,
        "groups": [
            "oscap",
            "oscap-report"
        ],
        "pci_dss": [
            "2.2"
        ]
    },
    "agent": {
        "id": "1040",
        "name": "ip-10-0-0-76",
        "ip": "10.0.0.76"
    },
    "manager": {
        "name": "vpc-ossec-manager"
    },
    "full_log": "oscap: msg: \"xccdf-overview\", scan-id: \"10401490050797\", content: \"ssg-centos-7-ds.xml\", benchmark-id: \"xccdf_org.ssgproject.content_benchmark_RHEL-7\", profile-id: \"xccdf_org.ssgproject.content_profile_common\", profile-title: \"Common Profile for General-Purpose Systems\", score: \"75.000000\".",
    "oscap": {
        "scan": {
            "id": "10401490050797",
            "content": "ssg-centos-7-ds.xml",
            "benchmark": {
                "id": "xccdf_org.ssgproject.content_benchmark_RHEL-7"
            },
            "profile": {
                "id": "xccdf_org.ssgproject.content_profile_common",
                "title": "Common Profile for General-Purpose Systems"
            },
            "score": "75.000000"
        }
    },
    "decoder": {
        "parent": "oscap",
        "name": "oscap"
    },
    "location": "wodle_open-scap"
}

```


Configuration

1. [Basic usage](#)
2. [Evaluate PCI-DSS compliance on RHEL7](#)
3. [Auditing Security Vulnerabilities of Red Hat Products](#)
4. [Overwriting the timeout](#)
5. [Using profiles](#)
6. [Using CPE dictionary](#)
7. [Using IDs](#)

Basic usage

To configure the options for OpenSCAP go to [ossec.conf](#), or for more details about specific options, see the [OpenSCAP section](#).

In this example, we configure Wazuh to run OpenSCAP each day, with a timeout of 30 minutes.

```
<wodle name="open-scap">
  <disabled>no</disabled>
  <timeout>1800</timeout>
  <interval>1d</interval>
  <scan-on-start>yes</scan-on-start>

  <content type="xccdf" path="ssg-centos-7-ds.xml">
    <profile>xccdf_org.ssgproject.content_profile_pci-dss</profile>
    <profile>xccdf_org.ssgproject.content_profile_common</profile>
  </content>
</wodle>
```

Evaluate PCI-DSS compliance on RHEL7

This section describes how to evaluate the Payment Card Industry Data Security Standard (PCI-DSS) compliance on Red Hat Enterprise Linux 7 agents.

Step 1: Configure agents

Each agent must be properly identified in order to know which policy and profile to execute.

Agent [ossec.conf](#):

```
<client>
  <server-ip>10.0.1.4</server-ip>
  <config-profile>redhat7</config-profile>
</client>
```

Step 2: Configure manager

We want to execute the PCI-DSS profile of the SSG RH7 policy only on Red Hat 7 servers.

Manager [shared/agent.conf](#):

```
<agent_config profile="redhat7">

  <wodle name="open-scap">
    <content type="xccdf" path="ssg-rhel7-ds.xml">
      <profile>xccdf_org.ssgproject.content_profile_pci-dss</profile>
    </content>
  </wodle>

</agent_config>
```

Step 3: Restart manager and agents

To apply the new configuration, restart the manager and agents:

```
$ /var/ossec/bin/ossec-control restart  
$ /var/ossec/bin/agent_control -R -a
```

If you prefer, you can restart a specific agent with the option `-u <id>` where **id** is the agent's id number.

Step 4: See alerts

When the evaluation is complete you will see the results as OSSEC alerts:

`/var/ossec/logs/alerts/alerts.log`

```
** Alert 1463752181.32768: - oscap,rule-result,pci_dss_2.2,  
2016 May 20 13:49:41 (RH_Agent) 10.0.1.7->wodle_open-scap  
Rule: 81529 (level 5) -> 'OpenSCAP rule failed (severity low).'  
oscap: msg: "rule-result", id: "47T7_Qd08gm4y8TSoD53", policy: "ssg-rhel7-ds.xml", profile:  
"xccdf_org.ssgproject.content_profile_pci-dss", rule_id:  
"xccdf_org.ssgproject.content_rule_sshd_set_idle_timeout", result: "fail", title: "Set SSH Idle Timeout  
Interval", ident: "CCE-26611-4", severity: "low".
```

```
** Alert 1463752181.33254: - oscap,report-overview,pci_dss_2.2,  
2016 May 20 13:49:41 (RH_Agent) 10.0.1.7->wodle_open-scap  
Rule: 81542 (level 4) -> 'OpenSCAP Report overview: Score less than 80'  
oscap: msg: "report-overview", id: "47T7_Qd08gm4y8TSoD53", policy: "ssg-rhel7-ds.xml", profile:  
"xccdf_org.ssgproject.content_profile_pci-dss", score: "56.835060" / "100.000000", severity of failed rules:  
"high": "1", "medium": "9", "low": "34", "n/a": "0".
```

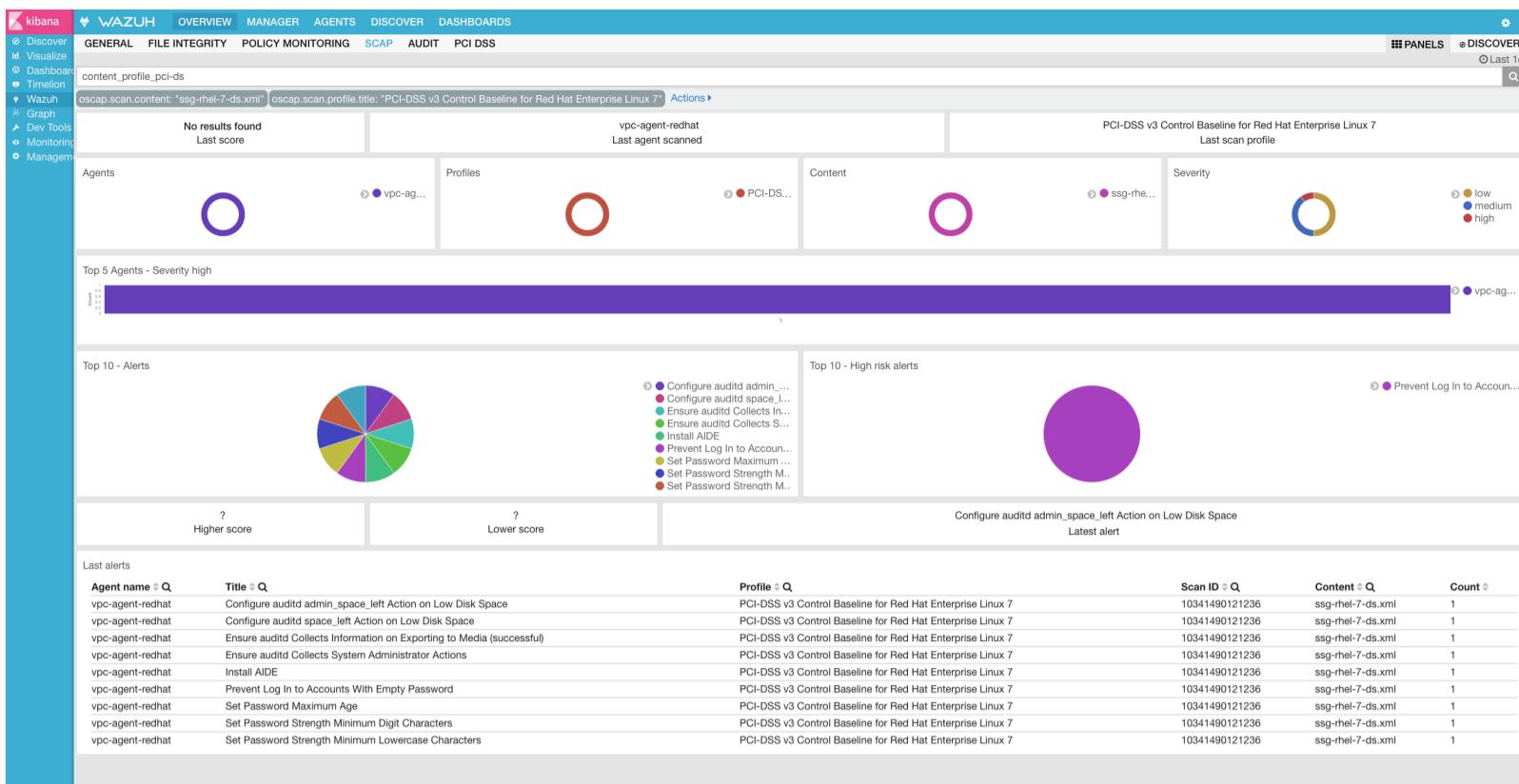
Kibana

Note that each field is extracted to facilitate searches and analysis.

○ @timestamp	Q Q D * March 21st 2017, 11:33:56.000
t _id	Q Q D * AVryIf3-Azwu4u-9YlQK
t _index	Q Q D * wazuh-alerts-2017.03.21
# _score	Q Q D * -
t _type	Q Q D * wazuh
t agent.id	Q Q D * 1034
t agent.ip	Q Q D * 10.0.0.127
t agent.name	Q Q D * vpc-agent-redhat
t decoder.name	Q Q D * oscap
t decoder.parent	Q Q D * oscap
t full_log	Q Q D * oscap: msg: "xccdf-result", scan-id: "10341490121236", content: "ssg-rhel7-ds.xml", title: "Set Password Maximum Age", id: "xccdf_org.ssgproject.content_rule_accounts_maximum_age_login_defs", result: "fail", severity: "medium", description: "To specify password maximum age for new accounts, edit the file /etc/login.defs and add or correct the following line, replacing DAYS appropriately: PASS_MAX_DAYS DAYS A value of 180 days is sufficient for many environments. The DOD requirement is 60.", rationale: "Setting the password maximum age ensures users are required to periodically change their passwords. This could possibly decrease the utility of a stolen password. Requiring shorter password lifetimes increases the risk of users writing down the password in a convenient location subject to physical compromise." references: "IA-5(f) (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), IA-5(g) (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), 180 (http://iase.disa.mil/stigs/cci/Pages/index.aspx), 199 (http://iase.disa.mil/stigs/cci/Pages/index.aspx), 76 (), Test attestation on 20121026 by OS (https://github.com/OpenSCAP/scap-security-guide/wiki/contributors)", identifiers: "CCE-27051-2 (http://cce.mitre.org)", oval-id: "oval:ssg:def:510", benchmark-id: "xccdf_org.ssgproject.content_benchmark_RHEL-7", profile-id: "xccdf_org.ssgproject.content_profile_pci-dss", profile-title: "PCI-DSS v3 Control Baseline for Red Hat Enterprise Linux 7".
t host	Q Q D * vpc-ossec-manager
t location	Q Q D * wodle_open-scap
t manager.name	Q Q D * vpc-ossec-manager
t oscap.check.description	Q Q D * To specify password maximum age for new accounts, edit the file /etc/login.defs and add or correct the following line, replacing DAYS appropriately: PASS_MAX_DAYS DAYS A value of 180 days is sufficient for many environments. The DOD requirement is 60.
t oscap.check.id	Q Q D * xccdf_org.ssgproject.content_rule_accounts_maximum_age_login_defs
t oscap.check.identifiers	Q Q D * CCE-27051-2 (http://cce.mitre.org)
t oscap.check.oval.id	Q Q D * oval:ssg:def:510
t oscap.check.rationale	Q Q D * Setting the password maximum age ensures users are required to periodically change their passwords. This could possibly decrease the utility of a stolen password. Requiring shorter password lifetimes increases the risk of users writing down the password in a convenient location subject to physical compromise.
t oscap.check.references	Q Q D * IA-5(f) (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), IA-5(g) (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), IA-5(l)(d) (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), 180 (http://iase.disa.mil/stigs/cci/Pages/index.aspx), 199 (http://iase.disa.mil/stigs/cci/Pages/index.aspx), 76 (), Test attestation on 20121026 by OS (https://github.com/OpenSCAP/scap-security-guide/wiki/contributors)
t oscap.check.result	Q Q D * fail
t oscap.check.severity	Q Q D * medium
t oscap.check.title	Q Q D * Set Password Maximum Age
t oscap.scan.benchmark.id	Q Q D * xccdf_org.ssgproject.content_benchmark_RHEL-7
t oscap.scan.content	Q Q D * ssg-rhel7-ds.xml
t oscap.scan.id	Q Q D * 10341490121236
t oscap.scan.profile.id	Q Q D * xccdf_org.ssgproject.content_profile_pci-dss
t oscap.scan.profile.title	Q Q D * PCI-DSS v3 Control Baseline for Red Hat Enterprise Linux 7
t rule.description	Q Q D * OpenSCAP: Set Password Maximum Age (not passed)
# rule.firedtimes	Q Q D * 3
t rule.groups	Q Q D * oscap, oscap-result
t rule.id	Q Q D * 81530
# rule.level	Q Q D * 7
t rule_pci_dss	Q Q D * 2.2
# source	Q Q D *

Step 5: Dashboards

Finally, you can explore all results using the OpenSCAP dashboards for Kibana.



Auditing Security Vulnerabilities of Red Hat Products

The Red Hat Security Response Team provides OVAL definitions for all vulnerabilities (identified by CVE name) that affect Red Hat Enterprise Linux 3, 4, 5, 6 and 7. This enables users to perform a vulnerability scan and diagnose whether a system is vulnerable or not.

Step 1: Configure agents

Each agent must be properly identified in order to know which policy and profile to execute.

Agent `ossec.conf`:

```
<client>
  <server-ip>10.0.1.4</server-ip>
  <config-profile>redhat7</config-profile>
</client>
```

Step 2: Configure manager

We want to execute the RedHat security policy only on Red Hat 7 servers.

Manager `shared/agent.conf`:

```
<agent_config profile="redhat7">

  <wodle name="open-scap">
    <content type="xccdf" path="com.redhat.rhsa-RHEL7.ds.xml"/>
  </wodle>

</agent_config>
```

Step 3: Restart manager and agents

To apply the new configuration, restart the manager and agents:

```
$ /var/ossec/bin/ossec-control restart
$ /var/ossec/bin/agent_control -R -a
```

If you prefer, you can restart a specific agent with option `-u <id>`.

Step 4: See alerts

When the evaluation is completed you will see the results as OSSEC alerts:

```
/var/ossec/logs/alerts/alerts.log
```

```

** Alert 1463757700.70731: mail - oscap,rule-result,pci_dss_2.2,
2016 May 20 15:21:40 (RH_Agent) 10.0.1.7->wodle_open-scap
Rule: 81531 (level 9) -> 'OpenSCAP rule failed (severity high).'
oscap: msg: "rule-result", id: "I0iLEGFi4iTkjnL9LWQ", policy: "com.redhat.rhsa-RHEL7.ds.xml", profile: "no-profiles", rule_id: "xccdf_com.redhat.rhsa_rule_oval-com.redhat.rhsa-def-20160722", result: "fail", title: "RHSA-2016:0722: openssl security update (Important)", ident: "RHSA-2016-0722, CVE-2016-0799, CVE-2016-2105, CVE-2016-2106, CVE-2016-2107, CVE-2016-2108, CVE-2016-2109, CVE-2016-2842", severity: "high".

```

```

** Alert 1463757700.71339: - oscap,report-overview,pci_dss_2.2,
2016 May 20 15:21:40 (RH_Agent) 10.0.1.7->wodle_open-scap
Rule: 81540 (level 1) -> 'OpenSCAP Report overview.'
oscap: msg: "report-overview", id: "I0iLEGFi4iTkjnL9LWQ", policy: "com.redhat.rhsa-RHEL7.ds.xml", profile: "no-profiles", score: "92.617447" / "100.000000", severity of failed rules: "high": "8", "medium": "14", "low": "0", "n/a": "0".

```

Kibana

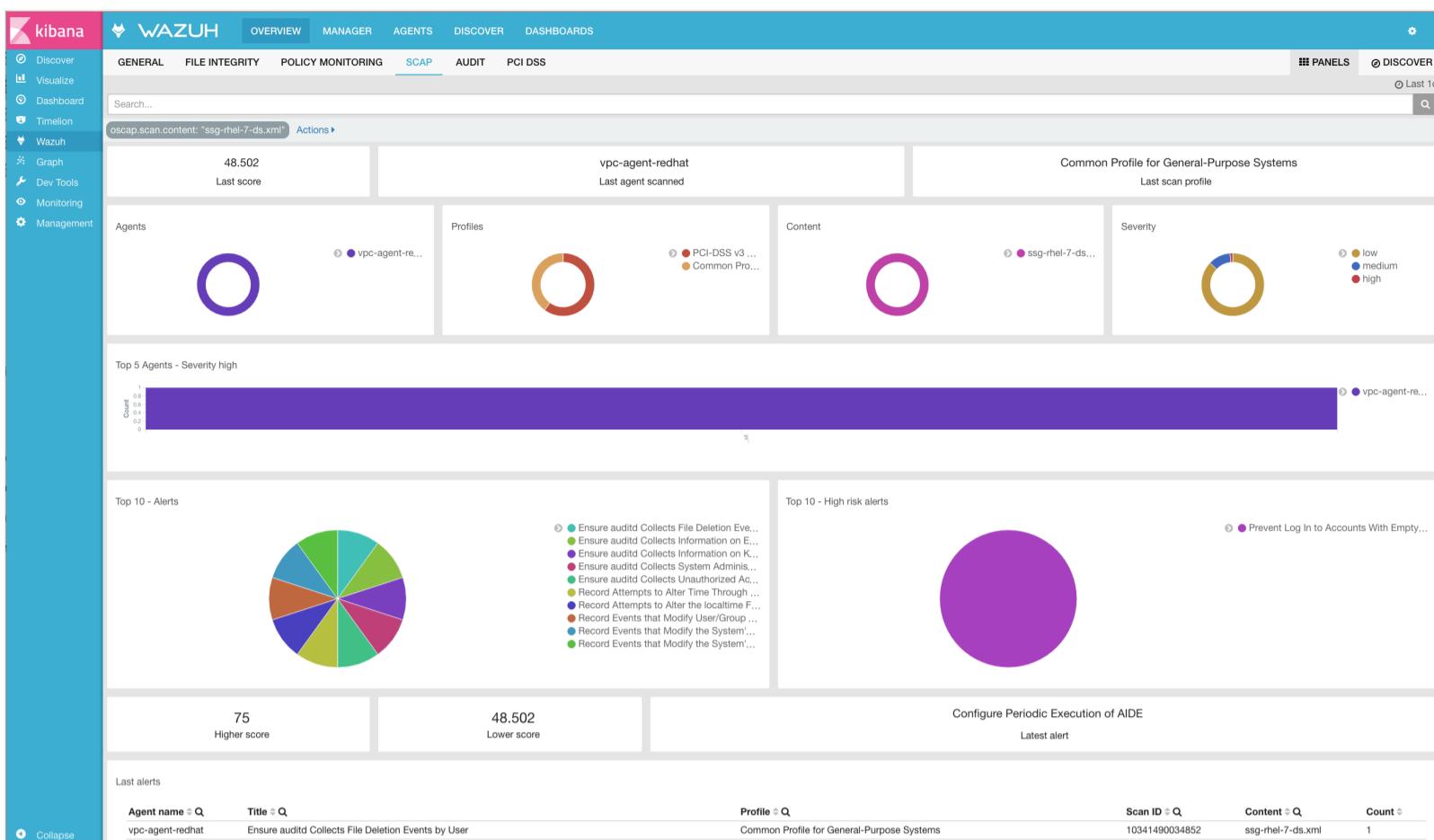
Note that each field is extracted to facilitate searches and analysis.

Table	JSON
o @timestamp	Q Q D * March 20th 2017, 11:33:57.000
t _id	Q Q D * Avrs-6VJAzww4u-9VMZD
t _index	Q Q D * wazuh-alerts-2017.03.20
# _score	Q Q D *
t _type	Q Q D * wazuh
t agent.id	Q Q D * 1034
t agent.ip	Q Q D * 10.0.0.127
t agent.name	Q Q D * vpc-agent-redhat
t decoder.name	Q Q D * oscap
t decoder.parent	Q Q D * oscap
t full_log	Q Q D * oscap: msg: "xccdf-result", scan-id: "10341490034836", content: "ssg-rhel-7-ds.xml", title: "Install libreswan Package", id: "xccdf_org.ssgproject.content_rule_package_libreswan_installed", result: "fail", severity: "low", description: "The Libreswan package provides an implementation of IPsec and IKE, which permits the creation of secure tunnels over untrusted networks. The libreswan package can be installed with the following command: \$ sudo yum install libreswan", rationale: "Providing the ability for remote users or systems to initiate a secure VPN connection protects information when it is transmitted over a wide area network." references: "AC-17 (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), MA-4 (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), SC-9 (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), 1130 (http://iae.disa.mil/stigs/cci/PageS/index.aspx), 1131 (http://iae.disa.mil/stigs/cci/Pages/index.aspx)", identifiers: "CCE-RHEL7-CCE-TBD (http://cce.mitre.org)", oval-id: "oval:ssg:def:473", benchmark-id: "xccdf_org.ssgproject.content_benchmark_RHEL-7", profile-id: "xccdf_org.ssgproject.content_profile_pci-dss", profile-title: "PCI-DSS v3 Control Baseline for Red Hat Enterprise Linux 7".
t host	Q Q D * vpc-ossec-manager
t location	Q Q D * wodle_open-scap
t manager.name	Q Q D * vpc-ossec-manager
t oscap.check.description	Q Q D * The Libreswan package provides an implementation of IPsec and IKE, which permits the creation of secure tunnels over untrusted networks. The libreswan package can be installed with the following command: \$ sudo yum install libreswan
t oscap.check.id	Q Q D * xccdf_org.ssgproject.content_rule_package_libreswan_installed
t oscap.check.identifiers	Q Q D * CCE-RHEL7-CCE-TBD (http://cce.mitre.org)
t oscap.check.oval.id	Q Q D * oval:ssg:def:473
t oscap.check.rationale	Q Q D * Providing the ability for remote users or systems to initiate a secure VPN connection protects information when it is transmitted over a wide area network.
t oscap.check.references	Q Q D * AC-17 (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), MA-4 (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), SC-9 (http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-53r4.pdf), 1130 (http://iae.disa.mil/stigs/cci/PageS/index.aspx), 1131 (http://iae.disa.mil/stigs/cci/Pages/index.aspx)
t oscap.check.result	Q Q D * fail
t oscap.check.severity	Q Q D * Low
t oscap.check.title	Q Q D * Install libreswan Package
t oscap.scan.benchmark.id	Q Q D * xccdf_org.ssgproject.content_benchmark_RHEL-7
t oscap.scan.content	Q Q D * ssg-rhel-7-ds.xml
t oscap.scan.id	Q Q D * 10341490034836
t oscap.scan.profile.id	Q Q D * xccdf_org.ssgproject.content_profile_pci-dss
t oscap.scan.profile.title	Q Q D * PCI-DSS v3 Control Baseline for Red Hat Enterprise Linux 7
t rule.description	Q Q D * OpenSCAP: Install libreswan Package (not passed)
# rule.firedtimes	Q Q D *
t rule.groups	Q Q D * oscap, oscap-result
t rule.id	Q Q D * 81529
# rule.level	Q Q D *
t rule_pci_dss	Q Q D * 2.2
# source	Q Q D *

Table	JSON
o @timestamp	Q Q D * March 21st 2017, 11:33:56.000
t _id	Q Q D * AvryIf3-Azw4u-9Ylqn
t _index	Q Q D * wazuh-alerts-2017.03.21
# _score	Q Q D *
t _type	Q Q D * wazuh
t agent.id	Q Q D * 1034
t agent.ip	Q Q D * 10.0.0.127
t agent.name	Q Q D * vpc-agent-redhat
t decoder.name	Q Q D * oscap
t decoder.parent	Q Q D * oscap
t full_log	Q Q D * oscap: msg: "xccdf-overview", scan-id: "10341490121236", content: "ssg-rhel-7-ds.xml", benchmark-id: "xccdf_org.ssgproject.content_benchmark_RHEL-7", profile-id: "xccdf_org.ssgproject.content_profile_pci-dss", profile-title: "PCI-DSS v3 Control Baseline for Red Hat Enterprise Linux 7", score: "48.501732".
t host	Q Q D * vpc-ossec-manager
t location	Q Q D * wodle_open-scap
t manager.name	Q Q D * vpc-ossec-manager
t oscap.scan.benchmark.id	Q Q D * xccdf_org.ssgproject.content_benchmark_RHEL-7
t oscap.scan.content	Q Q D * ssg-rhel-7-ds.xml
t oscap.scan.id	Q Q D * 10341490121236
t oscap.scan.profile.id	Q Q D * xccdf_org.ssgproject.content_profile_pci-dss
t oscap.scan.profile.title	Q Q D * PCI-DSS v3 Control Baseline for Red Hat Enterprise Linux 7
t oscap.scan.score	Q Q D * 48.502
t rule.description	Q Q D * OpenSCAP Report overview: Score less than 50
# rule.firedtimes	Q Q D *
t rule.groups	Q Q D * oscap, oscap-report
t rule.id	Q Q D * 81543
# rule.level	Q Q D *
t rule_pci_dss	Q Q D * 2.2
# source	Q Q D *

Step 5: Dashboards

Finally, you can explore all scan results using the OpenSCAP dashboards for Kibana.



Overwriting the timeout

It is possible to overwrite the timeout for a specific evaluation:

```
<wodle name="open-scap">

<timeout>1800</timeout>

<content type="xccdf" path="ssg-centos7-ds.xml">
    <timeout>120</timeout>
</content>

<content type="xccdf" path="ssg-centos6-ds.xml"/>

</wodle>
```

Using profiles

We can limit the evaluation to only specific profiles of a policy:

```
<wodle name="open-scap">

<content type="xccdf" path="ssg-centos7-ds.xml">
    <profile>xccdf_org.ssgproject.content_profile_standard</profile>
    <profile>xccdf_org.ssgproject.content_profile_pci-dss</profile>
</content>

<content type="xccdf" path="ssg-centos6-ds.xml"/>

</wodle>
```

Using CPE dictionary

You can also optionally specify the CPE dictionary file, which is used to determine which checks are relevant to specific platforms.

```
<wodle name="open-scap">

  <content type="xccdf" path="policy="ssg-centos7-ds.xml">
    <cpe>file.xml</cpe>
  </content>

  <content type="xccdf" path="ssg-centos6-ds.xml" />

</wodle>
```

Using IDs

You can select a specific ID of the datastream file:

```
<wodle name="open-scap">

  <content type="xccdf" path="ssg-centos7-ds.xml">
    <datastream-id>id</datastream-id>
    <xccdf-id>id</xccdf-id>
  </content>

  <content type="xccdf" path="ssg-centos6-ds.xml" />

</wodle>
```

FAQ

1. [Is there a noticeable performance impact when the OpenSCAP wodle is enabled on an agent?](#)
2. [Are evaluations executed in parallel?](#)
3. [How does the interval work?](#)
4. [Are the policies evaluated when OSSEC starts?](#)
5. [Where are the policies?](#)

Is there a noticeable performance impact when the OpenSCAP wodle is enabled on an agent?

The OpenSCAP wodle is designed to be very efficient, but the performance will depend on how fast oscap is (the scanner). Depending on the chosen policy, oscap can consume significant resources. We recommend you test your policies on a test agent before deploying them to production systems.

Are evaluations executed in parallel?

No, each evaluation is executed sequentially. Also, each profile of an evaluation is executed sequentially. This makes scans take somewhat longer but also reduces the load on agents caused by those scans.

How does the interval work?

The interval is the intended amount of time between the commencements of subsequent OpenSCAP scans on an agent. If a scan takes longer than the configured interval, an “interval overtaken” log message will be written to `/var/ossec/log/ossec.log`, and when the scan is finished, it will start again immediately.

Are the policies evaluated when OSSEC starts?

Yes, by default, policies are evaluated when the wodle starts. You can change this by setting `<scan-on-start>` to ‘no’. In this case, the next evaluation will be executed after the interval specified. The wodle state is saved when OSSEC is stopped.

Where are the policies?

Each agent must have its policies in `/var/ossec/wodles/oscap/policies`.

Setting up SSL for Filebeat and Logstash

If you are running Wazuh server and Elastic Stack on separate systems & servers (distributed architecture), then it is important to configure SSL encryption between Filebeat and Logstash. This not applies to single-server architectures.

! Note

Many of the commands described below need to be executed with root user privileges.

Generating a self-signed SSL certificate

1. First, we need an SSL certificate and key.

On the **machine with Logstash server** installed, create a copy of the OpenSSL example configuration file. The file location may vary depending on your operating system:

- a. On Debian or Ubuntu:

```
$ cp /etc/ssl/openssl.cnf custom_openssl.cnf
```

- b. On CentOS or Red Hat:

```
$ cp /etc/pki/tls/openssl.cnf custom_openssl.cnf
```

! Note

Typically you will run the Logstash server in your Elastic Stack server or, if you have set up a distributed Elasticsearch cluster, in one of its nodes.

2. Edit the custom configuration file, `custom_openssl.cnf`.

Find the section `[v3_ca]` and add a line like this, including your Elastic server's IP address:

```
[ v3_ca ]
subjectAltName = IP: YOUR_SERVER_IP
```

For example:

```
[ v3_ca ]
subjectAltName = IP: 192.168.1.2
```

3. Generate the SSL certificate and key:

```
$ openssl req -x509 -batch -nodes -days 3650 -newkey rsa:2048 -keyout /etc/logstash/logstash.key -out
/etc/logstash/logstash.crt -config custom_openssl.cnf
```

4. You may remove the custom configuration file:

```
$ rm custom_openssl.cnf
```

Configure Logstash server

At this point you should have your SSL certificate and key at `/etc/logstash/logstash.crt` and `/etc/logstash/logstash.key` respectively. Now we'll configure Logstash to use it across with Filebeat.

1. Edit file `/etc/logstash/conf.d/01-wazuh.conf` and uncomment the lines related to SSL under `input/beats`. The active input section should now look like this:

```
input {  
    beats {  
        port => 5000  
        codec => "json_lines"  
        ssl => true  
        ssl_certificate => "/etc/logstash/logstash.crt"  
        ssl_key => "/etc/logstash/logstash.key"  
    }  
}
```

2. Restart Logstash. The command depends on the OS init system:

a. For Systemd:

```
$ systemctl restart logstash.service
```

b. For legacy SysV Init:

```
$ service logstash restart
```

Configure Filebeat

Now we will configure Filebeat to verify the Logstash server's certificate.

1. On the **machine with Filebeat installed** (Wazuh server), fetch the Logstash server's SSL certificate file at `/etc/logstash/logstash.crt` and copy it into `/etc/filebeat/logstash.crt`.

Here is an **example** you might use to copy the SSL certificate from the Logstash server to Wazuh server where Filebeat is installed:

```
$ scp root@LOGSTASH_SERVER_IP:/etc/logstash/logstash.crt /etc/filebeat
```

2. Edit the file `/etc/filebeat/filebeat.yml` and uncomment the lines related to SSL inside `logstash`. The file should remain like this:

```
output:  
logstash:  
  hosts: ["192.168.1.2:5000"]  
  ssl:  
    certificateAuthorities: ["/etc/filebeat/logstash.crt"]
```

3. Restart Filebeat. The command depends on the OS init system:

a. For Systemd:

```
$ systemctl restart filebeat.service
```

b. For legacy SysV Init:

```
$ service filebeat restart
```

 Note

More detailed information is available in the [Securing communication with Logstash](#) guide from Elastic.

Setting up SSL and authentication for Kibana

By default, the communications between Kibana (including the Wazuh app) and the web browser on end-user systems are not encrypted. It's strongly recommended to configure Kibana to use SSL encryption and to enable authentication, next we briefly describe how to do this with a NGINX setup.

NGINX is a popular open-source web server and reverse proxy, known for its high performance, stability, rich feature set, simple configuration, and low resource consumption. Here we will use it as a reverse proxy to provide to the end users an encrypted and authenticated access to Kibana.

Note

Many of the commands described below need to be executed with root user privileges.

Contents

1. [NGINX SSL proxy for Kibana \(RPM-based distributions\)](#)
2. [NGINX SSL proxy for Kibana \(Debian-based distributions\)](#)

NGINX SSL proxy for Kibana (RPM-based distributions)

1. First, install NGINX:

- a. For CentOS:

```
$ cat > /etc/yum.repos.d/nginx.repo <<\EOF
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/centos/$releasever/$basearch/
gpgcheck=0
enabled=1
EOF

$ yum install nginx
```

- a. For RHEL:

```
$ cat > /etc/yum.repos.d/nginx.repo <<\EOF
[nginx]
name=nginx repo
baseurl=http://nginx.org/packages/rhel/$releasever/$basearch/
gpgcheck=0
enabled=1
EOF

$ yum install nginx
```

Note

For more information, see [NGINX: Official Red Hat/CentOS packages](#).

2. Install your SSL certificate and private key:

a. If you have a valid **signed certificate**, copy your key file `<ssl_key>` and your certificate file `<ssl_pem>` to their proper locations:

```
$ mkdir -p /etc/pki/tls/certs /etc/pki/tls/private
$ cp <ssl_pem> /etc/pki/tls/certs/kibana-access.pem
$ cp <ssl_key> /etc/pki/tls/private/kibana-access.key
```

b. Otherwise, create a **self-signed certificate**. Remember to set the `Common Name` field to your server name. For instance, if your server is `example.com`, you would do the following:

```
$ mkdir -p /etc/pki/tls/certs /etc/pki/tls/private
$ openssl req -x509 -batch -nodes -days 365 -newkey rsa:2048 -keyout /etc/pki/tls/private/kibana-access.key -out /etc/pki/tls/certs/kibana-access.pem
```

3. Configure NGINX as an HTTPS reverse proxy to Kibana:

```
$ cat > /etc/nginx/conf.d/default.conf <<\EOF
server {
    listen 80;
    listen [::]:80;
    return 301 https://$host$request_uri;
}

server {
    listen 443 default_server;
    listen [::]:443;
    ssl on;
    ssl_certificate /etc/pki/tls/certs/kibana-access.pem;
    ssl_certificate_key /etc/pki/tls/private/kibana-access.key;
    access_log          /var/log/nginx/nginx.access.log;
    error_log           /var/log/nginx/nginx.error.log;
    location / {
        auth_basic "Restricted";
        auth_basic_user_file /etc/nginx/conf.d/kibana.htpasswd;
        proxy_pass http://localhost:5601;
    }
}
EOF
```

! Note

We configure nginx in order to encapsulate the IP address of the Kibana server. This configuration allows us to redirect Kibana requests to HTTPS, when you use this configuration it's recommended to edit the file `/etc/kibana/kibana.yml` and set the field `server.host` to `localhost`, then you must restart the Kibana service to apply this change.

4. Allow NGINX to connect to Kibana port if you're using SELinux:

```
$ semanage port -a -t http_port_t -p tcp 5601
```

! Note

We assume that you have `policycoreutils-python` installed to manage SELinux.

Enable authentication by htpasswd

1. Install the package `httpd-tools`:

```
$ yum install httpd-tools
```

2. Generate the `.htpasswd` file. Replace `wazuh` with your chosen username (it must match with `auth_basic_user_file`):

```
$ htpasswd -c /etc/nginx/conf.d/kibana.htpasswd wazuh
```

3. Restart NGINX:

- a. For Systemd:

```
$ systemctl restart nginx
```

- b. For SysV Init:

```
$ service nginx restart
```

Now try to access the Kibana web interface via HTTPS. It should prompt you for the username and password that you just created.

NGINX SSL proxy for Kibana (Debian-based distributions)

1. Install NGINX:

```
$ apt-get install nginx
```

2. Install your SSL certificate and private key:

- a. If you have a valid signed certificate, copy your key file `<ssl_key>` and your certificate file `<ssl_pem>` to their proper locations:

```
$ mkdir -p /etc/ssl/certs /etc/ssl/private
$ cp <ssl_pem> /etc/ssl/certs/kibana-access.pem
$ cp <ssl_key> /etc/ssl/private/kibana-access.key
```

- b. Otherwise, create a **self-signed certificate**. Remember to set the `Common Name` field to your server name. For instance, if your server is `example.com`, you would do the following:

```
$ mkdir -p /etc/ssl/certs /etc/ssl/private
$ openssl req -x509 -batch -nodes -days 365 -newkey rsa:2048 -keyout /etc/ssl/private/kibana-access.key -out /etc/ssl/certs/kibana-access.pem
```

3. Configure NGINX as an HTTPS reverse proxy to Kibana:

```
$ cat > /etc/nginx/sites-available/default <<\EOF
server {
    listen 80;
    listen [::]:80;
    return 301 https://$host$request_uri;
}

server {
    listen 443 default_server;
    listen [::]:443;
    ssl on;
    ssl_certificate /etc/ssl/certs/kibana-access.pem;
    ssl_certificate_key /etc/ssl/private/kibana-access.key;
    access_log          /var/log/nginx/nginx.access.log;
    error_log           /var/log/nginx/nginx.error.log;
    location / {
        auth_basic "Restricted";
        auth_basic_user_file /etc/nginx/conf.d/kibana.htpasswd;
        proxy_pass http://localhost:5601/;
    }
}
EOF
```

! Note

We configure nginx in order to encapsulate the IP address of the Kibana server. This configuration allows us to redirect Kibana requests to HTTPS, when you use this configuration it's recommended to edit the file `/etc/kibana/kibana.yml` and set the field `server.host` to `localhost`, then you must restart the Kibana service to apply this change.

Enable authentication by htpasswd

1. Install the package `apache2-utils`:

```
$ apt-get install apache2-utils
```

2. Generate the `.htpasswd` file. Replace `<user>` with your chosen username:

```
$ htpasswd -c /etc/nginx/conf.d/kibana.htpasswd <user>
```

3. Restart NGINX:

a. For Systemd:

```
$ systemctl restart nginx
```

b. For SysV Init:

```
$ service nginx restart
```

Now try to access the Kibana web interface via HTTPS. It should prompt you for the username and password that you just created.

Securing the Wazuh API

By default, the communications between the Wazuh Kibana App and the Wazuh API are not encrypted. You should take the following actions to secure the Wazuh API.

1. Change default credentials:

By default you can access by typing user “foo” and password “bar”. We recommend you to generate new credentials. This can be done very easily, with the following steps:

```
$ cd /var/ossec/api/configuration/auth  
$ sudo node htpasswd -c user myUserName
```

2. Enable HTTPS:

In order to enable HTTPS you need to generate or provide a certificate. You can learn how to generate your own certificate or generate it automatically using the script [/var/ossec/api/scripts/configure_api.sh](#).

3. Bind to localhost:

In case you do not need to access to the API externally, you should bind the API to [localhost](#) using the option [config.host](#) placed in the configuration file [/var/ossec/api/configuration/config.js](#).

Elasticsearch tuning

This guide summarizes the relevant configurations that allow us to optimize Elasticsearch.

1. [Memory locking](#)
2. [Shards and replicas](#)

Memory locking

Elasticsearch performs poorly when the system is swapping the memory. It is vitally important to the health of your node that none of the JVM is ever swapped out to disk.

We will set the `bootstrap.memory_lock` setting to true, so Elasticsearch will lock the process address space into RAM, preventing any Elasticsearch memory from being swapped out.

Step 1: Set `bootstrap.memory_lock`

Uncomment or add this line to the `/etc/elasticsearch/elasticsearch.yml` file:

```
bootstrap.memory_lock: true
```

Step 2: Edit limit of system resources

Where to configure systems settings depends on which package and operating system you choose to use for Elasticsearch installation.

- In a case where **systemd** is used, system limits need to be specified via systemd. First, create the folder executing the command:
`mkdir -p /etc/systemd/system/elasticsearch.service.d/`, add a file called `elasticsearch.conf` and specify any changes in that file:

```
[Service]
LimitMEMLOCK=infinity
```

- In other case, edit the proper file `/etc/sysconfig/elasticsearch` for RPM or `/etc/default/elasticsearch` for Debian:

```
MAX_LOCKED_MEMORY=unlimited
```

Step 3: Limit memory

The previous configuration might cause node instability or even node death (with an `OutOfMemory` exception) if Elasticsearch tries to allocate more memory than is available. JVM heap limits will help us to define the memory usage and prevent this situation.

There are two rules to apply when setting the Elasticsearch heap size:

- No more than 50% of available RAM.
- No more than 32 GB.

In addition, you must take into account the memory usage by the operating system, services and software running on the host.

By default, Elasticsearch is configured with a 1 GB heap. You can change the heap size via JVM flags using the `/etc/elasticsearch/jvm.options` file:

```
# Xms represents the initial size of total heap space
# Xmx represents the maximum size of total heap space

-Xms4g
-Xmx4g
```

⚠ Warning

Ensure that the min (Xms) and max (Xmx) sizes are the same, this prevents JVM heap resizing at runtime, a very costly process.

Step 4: Restart Elasticsearch

Finally, restart Elasticsearch service:

a. For Systemd:

```
systemctl daemon-reload  
systemctl restart elasticsearch
```

b. For SysV Init:

```
service elasticsearch restart
```

After starting Elasticsearch, you can see whether this setting was successfully applied by checking the value of `mlockall` in the output of the next request:

```
curl -XGET 'localhost:9200/_nodes?filter_path=**.mlockall&pretty'
```

```
{  
  "nodes" : {  
    "sRuGbIQRRfC54wzwIHjJWQ" : {  
      "process" : {  
        "mlockall" : true  
      }  
    }  
  }  
}
```

The request has failed when you see the above output have `"mlockall" : false` field. You will also see a line with more information in the logs (`/var/log/elasticsearch/elasticsearch.log`) with the words *Unable to lock JVM Memory*.

Reference:

- [Memory lock check.](#)
- [bootstrap.memory_lock.](#)
- [Enable bootstrap.memory_lock.](#)
- [Heap: Sizing and Swapping.](#)
- [Limiting memory usage.](#)

Shards and replicas

Elasticsearch provides the ability to split an index in multiple pieces called shards. Each shard is in itself a fully-functional and independent “index” that can be hosted on any node in the cluster. Sharding is important for two primary reasons:

- It allows you to horizontally split/scale your content volume.
- It allows you to distribute and parallelize operations across shards thus increasing performance/throughput.

Also, Elasticsearch allows you to make one or more copies of your index’s shards into what are called replica shards, or replicas for short. Replication is important for two primary reasons:

- It provides high availability in case a shard/node fails.
- It allows you to scale out your search volume/throughput since searches can be executed on all replicas in parallel.

⚠ Warning

The number of shards and replicas can be defined per index at the time the index is created. After the index is created, you may change the number of replicas dynamically anytime but you cannot change the number of shards after-the-fact.

How many shards should my index have?

Due to is not possible to *reshard* (changing the number of shards) without reindexing, you must answer to this question before create your first index. However, we can't decide how many shards use without talk about nodes. In general, you will get the optimal performance by using the same number of shards as nodes. So, a cluster with 3 nodes will have 3 shards while a cluster with one node will only have a shard.

How many replicas should my index have?

Let's consider what will happen in a cluster with 3 nodes and 3 shards:

- No replica: Each node has 1 shard. If a node falls down, we will have an incomplete index of 2 shards.
- 1 replica: Each node has 1 shard and 1 replica. If a node falls down, we will have a complete index.
- 2 replicas: Each node has 1 shard and 2 replicas (the full index). Now, the cluster can work with just one node. This looks like the best solution but also it increase the storage requirements.

Setting number of shards and replicas

The default installation of Elastic Stack with [RPM](#) or [Debian](#) packages will configure each index with 5 primary shards and 1 replica.

In case you want to change these settings you need to edit the Elasticsearch template. In the following example, we configure the proper values for shards and replicas in a cluster with only 1 node.

⚠ Warning

We assume that your index has not yet been created, otherwise you will have to [reindex](#) after editing the template.

1. Download the Wazuh Elasticsearch template:

```
curl https://raw.githubusercontent.com/wazuh/wazuh/2.1/extensions/elasticsearch/wazuh-elasticsearch-template.json -o w-elastic-template.json
```

2. Edit the template in order to set 1 shard a 0 replicas:

```
nano w-elastic-template.json

{
  "order": 0,
  "template": "wazuh*",
  "settings": {
    "index.refresh_interval": "5s",
    "number_of_shards": 1,
    "number_of_replicas": 0
  },
  "mappings": {
    "...": ...
  }
}
```

3. Load the template:

```
curl -XPUT 'http://localhost:9200/_template/wazuh' -H 'Content-Type: application/json' -d @w-elastic-template.json
```

Changing number of replicas

The number of replicas can be changed dynamically using the Elasticsearch API.

In a cluster with 1 node, the number of replicas should be 0:

```
curl -XPUT 'localhost:9200/wazuh-*/_settings?pretty' -H 'Content-Type: application/json' -d'
{
  "settings": {
    "number_of_replicas" : 0
  }
}'
```

Reference:

- [Shards & Replicas](#).

Getting started

This guide provides all the basic information you need to start using the Wazuh API.

Starting and stopping the API

The API starts at boot time. To control or check the wazuh-api service, use the systemctl or service command.

Systemd systems

```
systemctl start/status/stop/restart wazuh-api
```

SysVinit systems

```
service wazuh-api start/status/stop/restart
```

Hello world!

In order to check if everything is working as expected, you can use cURL to do a *request*:

```
$ curl -u foo:bar -k https://127.0.0.1:55000?pretty
{
  "error": 0,
  "data": "Welcome to Wazuh HIDS API"
}
```

Explanation:

- `curl`: This is a command-line tool for sending requests and commands over HTTP and HTTPS.
- `-u foo:bar`: Specify a username and password to authenticate with the API.
- `-k`: Allow connections to SSL sites with self signed certs.
- `https://127.0.0.1:55000`: This is the API URL to use if you are running the command on the manager itself.
- `?pretty`: This parameter makes the JSON output more human-readable.

Basic concepts

Basic concepts about making API requests and understanding their responses:

- The *base URL* for each request is `https://IP:55000/` or `http://IP:55000/`, depending on if you enabled and set up SSL in the API.
- All responses are in *JSON format* with the following structure:

Field	Description
error	0 if everything was fine and an error code otherwise.
data	data requested. Only if error is equal to 0.
message	error description. Only if error is different to 0.

- Example response without errors:

```
{ "error": "0", "data": "Welcome to Wazuh HIDS API" }
```

- Example response with errors:

```
{ "error": "603", "message": "The requested URL was not found on this server" }
```

- Responses containing collections of data will return a maximum of 500 elements. You should use the *offset* and *limit* parameters to iterate through large collections.
- All responses have an HTTP status code: 2xx (success), 4xx (client error), 5xx (server error), etc.
- All requests accept the parameter *pretty* to convert the JSON response to a more human-readable format.
- The API log is stored on the manager as `/var/ossec/logs/api.log`.

Use cases

This section will present several use cases to give you a taste for the API's potential. You can find details about all possible API requests in the [reference](#) section.

Exploring the ruleset

Often when an alert fires, it is helpful to know details about the rule itself. The following request enumerates the attributes of rule **1002**:

```
curl -u foo:bar -k "https://127.0.0.1:55000/rules/1002?pretty"
```

```
{
  "error": 0,
  "data": {
    "totalItems": 1,
    "items": [
      {
        "status": "enabled",
        "pci": [],
        "description": "Unknown problem somewhere in the system.",
        "file": "syslog_rules.xml",
        "level": 2,
        "groups": [
          "syslog",
          "errors"
        ],
        "id": 1002,
        "details": {
          "options": "alert_by_email",
          "match": "$BAD_WORDS"
        }
      }
    ]
  }
}
```

It can also be helpful to know what rules are available that match specific criteria. For example, we can show all rules with a group of **web**, a PCI tag of **10.6.1**, and containing the word **failures**, which returns only one rule in this case.

```
curl -u foo:bar -k "https://127.0.0.1:55000/rules?group=web&pci=10.6.1&search=failures&pretty"
```

```
{
  "error": 0,
  "data": {
    "totalItems": 1,
    "items": [
      {
        "status": "enabled",
        "pci": [
          "10.6.1",
          "10.2.4",
          "10.2.5",
          "11.4"
        ],
        "description": "Multiple web authentication failures.",
        "file": "nginx_rules.xml",
        "level": 10,
        "groups": [
          "authentication_failures",
          "nginx",
          "web"
        ],
        "id": 31316,
        "details": {
          "same_source_ip": null,
          "frequency": "6",
          "if_matched_sid": "31315",
          "timeframe": "240"
        }
      }
    ]
  }
}
```

Mining the file integrity monitoring database of an agent

You can use the API to show information about all the files monitored by syscheck. For example, you can enumerate all monitored files on agent *000* (the manager) with extension *.py* that have been modified. In order to be concise, "*?limit=1*" has been used in this example to limit the results to a single record.

```
curl -u foo:bar -k "https://127.0.0.1:55000/syscheck/000/files?offset=0&limit=1&event=modified&search=.py&pretty"
```

```
{
  "error": 0,
  "data": {
    "totalItems": 1,
    "items": [
      {
        "uid": 0,
        "scanDate": "2016-07-14 10:58:45",
        "user": "root",
        "file": "/home/example.py",
        "modificationDate": "2016-07-14 10:58:18",
        "octalMode": "100777",
        "inode": 270323,
        "event": "modified",
        "size": 8,
        "sha1": "a38c98822f783fd45c256fe8fc928300c169d138",
        "group": "root",
        "gid": 0,
        "permissions": "-rwxrwxrwx",
        "md5": "b7f912e271b6c3e86ba2787f227d984c"
      }
    ]
  }
}
```

In case you need to find a file using its md5/sha1 hash, you can do so with a simple request like this:

```
curl -u foo:bar -k "https://127.0.0.1:55000/syscheck/000/files?hash=9d0ac660826f4245f3444b0247755c7229f1f9fe&pretty"
```

```
{  
  "error": 0,  
  "data": {  
    "totalItems": 1,  
    "items": [  
      {  
        "uid": 0,  
        "scanDate": "2016-07-14 08:49:27",  
        "user": "root",  
        "file": "/etc/default/cron",  
        "modificationDate": "2014-10-25 22:04:09",  
        "octalMode": "100644",  
        "inode": 262805,  
        "event": "added",  
        "size": 955,  
        "sha1": "9d0ac660826f4245f3444b0247755c7229f1f9fe",  
        "group": "root",  
        "gid": 0,  
        "permissions": "-rw-r--r--",  
        "md5": "eae0d979b5007d2af41540d8c2631359"  
      }  
    ]  
  }  
}
```

Listing outstanding rootcheck issues

Rootcheck requests are very similar to the syscheck ones. In order to get all rootcheck issues with an *outstanding* status you can run this request:

```
curl -u foo:bar -k "https://127.0.0.1:55000/rootcheck/000?status=outstanding&offset=0&limit=1&pretty"
```

```
{  
  "error": 0,  
  "data": {  
    "totalItems": 3,  
    "items": [  
      {  
        "status": "outstanding",  
        "oldDay": "2016-07-14 08:49:28",  
        "readDay": "2016-07-14 08:49:28",  
        "event": "System Audit: SSH Hardening - 1: Port 22 {PCI_DSS: 2.2.4}. File: /etc/ssh/sshd_config"  
      }  
    ]  
  }  
}
```

Starting the manager and dumping its configuration

It is possible to use the API to interact with the Manager in many ways. For example, you can stop/start/restart it or get its state with this simple request:

```
curl -u foo:bar -k -X PUT "https://127.0.0.1:55000/manager/restart?pretty"
```

```
{
  "error": 0,
  "data": [
    {
      "status": "running",
      "daemon": "wazuh-moduled"
    },
    {
      "status": "running",
      "daemon": "ossec-maild"
    },
    {
      "status": "running",
      "daemon": "ossec-execd"
    },
    {
      "status": "running",
      "daemon": "ossec-analysisd"
    },
    {
      "status": "running",
      "daemon": "ossec-logcollector"
    },
    {
      "status": "running",
      "daemon": "ossec-remoted"
    },
    {
      "status": "running",
      "daemon": "ossec-syscheckd"
    },
    {
      "status": "running",
      "daemon": "ossec-monitord"
    }
  ]
}
```

You can even dump the manager's current configuration with the below request (response shortened for brevity):

```
curl -u foo:bar -k "https://127.0.0.1:55000/manager/configuration?pretty"
```

```
{
  "error": 0,
  "data": {
    "global": {
      "email_notification": "no",
      "white_list": [
        "127.0.0.1",
        "^localhost.localdomain$",
        "10.0.0.2"
      ],
      "jsonout_output": "yes",
      "logall": "yes"
    },
    "...": { "...": "..."}
  }
}
```

Playing with agents

Of course we can work with agents. This enumerates **active** agents:

```
curl -u foo:bar -k "https://127.0.0.1:55000/agents?offset=0&limit=1&status=active&pretty"
```

```
{  
  "error": 0,  
  "data": {  
    "totalItems": 1,  
    "items": [  
      {  
        "status": "Active",  
        "ip": "127.0.0.1",  
        "id": "000",  
        "name": "LinMV"  
      }  
    ]  
  }  
}
```

Adding an agent is now easier than ever. Just send a request with the agent name and its IP.

```
curl -u foo:bar -k -X POST -d '{"name": "NewHost", "ip": "10.0.0.8"}' -H 'Content-Type:application/json' "https://127.0.0.1:55000/agents?pretty"
```

```
{  
  "error": 0,  
  "data": "019"  
}
```

You can fetch an agent's key like this:

```
curl -u foo:bar -k "https://127.0.0.1:55000/agents/019/key?pretty"
```

```
{  
  "error": 0,  
  "data"::  
"MDE5IGFkZmFmZGFkZmFkZmFkZmEgMTg1LjE2LjIxMS44OCBjN2Y2YzFhMjc4NWI1NjBhOWZiZGJiNjY20DMwMzdLODNkMjQwNDc5NmUxMDI2  
Yzk1ZTBmMmY2MDQ5ZDU1Mjlj"  
}
```

Conclusion

We hope you now better appreciate the potential of the Wazuh API. Remember to check out the [reference](#) document to discover all the available API requests. A nice summary can also be found here: [summary](#).

Configuration

The API will bind to port 55000/tcp by default and requires username/password authentication. The default username and password is "foo" and "bar".

Configuration script

Run the script `/var/ossec/api/scripts/configure_api.sh` to configure the basic settings.

Configuration file

You can configure certain API settings in the file `/var/ossec/api/configuration/config.js`:

```
// Path
config.ossec_path = "/var/ossec";
// The host to bind the API to.
config.host = "0.0.0.0";
// TCP Port used by the API.
config.port = "55000";
// Use HTTP protocol over TLS/SSL. Values: yes, no.
config.https = "yes";
// Use HTTP authentication. Values: yes, no.
config.basic_auth = "yes";
// In case the API runs behind a proxy server, turn to "yes" this feature. Values: yes, no.
config.BehindProxyServer = "no";
```

Make sure to restart wazuh-api service after editing the config file using the command below appropriate for your system:

```
systemctl restart wazuh-api
service wazuh-api restart
```

Basic Authentication

It is generally recommended to generate new credentials to replace foo:bar. This can be done very easily with the following steps, substituting your desired username for **myUserName**:

```
$ cd /var/ossec/api/configuration/auth
$ sudo node htpasswd -c user myUserName
```

Do not forget to restart the API to apply the changes:

```
$ systemctl restart wazuh-api
$ service wazuh-api restart
```

Manually enable https support

Generate key and certificate request (the OpenSSL package is required):

```
$ cd /var/ossec/api/configuration/ssl
$ sudo openssl genrsa -des3 -out server.key 1024
$ sudo openssl req -new -key server.key -out server.csr
```

By default, the key's password must be entered every time you run the server. If you don't want to enter the password every time, you can remove it by running these commands:

```
$ sudo cp server.key server.key.org  
$ sudo openssl rsa -in server.key.org -out server.key
```

Next generate your self-signed certificate:

```
$ sudo openssl x509 -req -days 365 -in server.csr -signkey server.key -out server.crt
```

And remove temporary files:

```
$ sudo rm server.csr  
$ sudo rm server.key.org
```

Reference

This API reference is organized by resources:

- [Agents](#)
- [Decoders](#)
- [Manager](#)
- [Rootcheck](#)
- [Rules](#)
- [Syscheck](#)

Also, it is provided an [Request List](#) with all available requests.

Request List

Agents

- DELETE /agents ([Delete a list of agents](#))
- DELETE /agents/:agent_id ([Delete an agent](#))
- GET /agents ([Get all agents](#))
- GET /agents/:agent_id ([Get an agent](#))
- GET /agents/:agent_id/key ([Get agent key](#))
- GET /agents/summary ([Get agents summary](#))
- GET /agents/summary/os ([Get OS summary](#))
- POST /agents ([Add agent](#))
- POST /agents/insert ([Insert agent](#))
- POST /agents/restart ([Restart a list of agents](#))
- PUT /agents/:agent_id/restart ([Restart an agent](#))
- PUT /agents/:agent_name ([Add agent \(quick method\)](#))
- PUT /agents/restart ([Restart all agents](#))

Decoders

- GET /decoders ([Get all decoders](#))
- GET /decoders/:decoder_name ([Get decoders by name](#))
- GET /decoders/files ([Get all decoders files](#))
- GET /decoders/parents ([Get all parent decoders](#))

Manager

- GET /manager/configuration ([Get manager configuration](#))
- GET /manager/info ([Get manager information](#))
- GET /manager/logs ([Get ossec.log](#))
- GET /manager/logs/summary ([Get summary of ossec.log](#))
- GET /manager/stats ([Get manager stats](#))
- GET /manager/stats/hourly ([Get manager stats by hour](#))
- GET /manager/stats/weekly ([Get manager stats by week](#))
- GET /manager/status ([Get manager status](#))

Rootcheck

- DELETE /rootcheck ([Clear rootcheck database](#))
- DELETE /rootcheck/:agent_id ([Clear rootcheck database of an agent](#))

- GET /rootcheck/:agent_id ([Get rootcheck database](#))
- GET /rootcheck/:agent_id/cis ([Get rootcheck CIS requirements](#))
- GET /rootcheck/:agent_id/last_scan ([Get last rootcheck scan](#))
- GET /rootcheck/:agent_id/pci ([Get rootcheck pci requirements](#))
- PUT /rootcheck ([Run rootcheck scan in all agents](#))
- PUT /rootcheck/:agent_id ([Run rootcheck scan in an agent](#))

Rules

- GET /rules ([Get all rules](#))
- GET /rules/:rule_id ([Get rules by id](#))
- GET /rules/files ([Get files of rules](#))
- GET /rules/groups ([Get rule groups](#))
- GET /rules/pci ([Get rule pci requirements](#))

Syscheck

- DELETE /syscheck ([Clear syscheck database](#))
- DELETE /syscheck/:agent_id ([Clear syscheck database of an agent](#))
- GET /syscheck/:agent_id ([Get syscheck files](#))
- GET /syscheck/:agent_id/last_scan ([Get last syscheck scan](#))
- PUT /syscheck ([Run syscheck scan in all agents](#))
- PUT /syscheck/:agent_id ([Run syscheck scan in an agent](#))

Agents

Add

Add agent

Add a new agent.

Request:

POST

/agents

Parameters:

Param	Type	Description
<code>name</code>	String	Agent name.
<code>ip</code>	String	<p>If you do not include this param, the API will get the IP automatically. If you are behind a proxy, you must set the option config.BehindProxyServer to yes at config.js.</p> <p>Allowed values:</p> <ul style="list-style-type: none"> • IP • IP/NET • ANY
<code>force</code>	Number	Remove old agent with same IP if disconnected since <force> seconds.

Example Request:

```
curl -u foo:bar -k -X POST -d '{"name": "NewHost", "ip": "10.0.0.9"}' -H 'Content-Type:application/json' "https://127.0.0.1:55000/agents?pretty"
```

Example Response:

```
{  
  "error": 0,  
  "data": "005"  
}
```

Add agent (quick method)

Adds a new agent with name :agent_name. This agent will use ANY as IP.

Request:

PUT

```
/agents/:agent_name
```

Parameters:

Param	Type	Description
agent_name	String	Agent name.

Example Request:

```
curl -u foo:bar -k -X PUT "https://127.0.0.1:55000/agents/myNewAgent?pretty"
```

Example Response:

```
{  
  "error": 0,  
  "data": "006"  
}
```

Insert agent

Insert an agent with an existing id and key.

Request:

POST

```
/agents/insert
```

Parameters:

Param	Type	Description
name	String	Agent name.
ip	String	If you do not include this param, the API will get the IP automatically. If you are behind a proxy, you must set the option config.BehindProxyServer to yes at config.js. Allowed values: <ul style="list-style-type: none">• IP• IP/NET• ANY

Param	Type	Description
id	String	Agent ID.
key	String	Agent key. Minimum length: 64 characters. Allowed values: ^[a-zA-Z0-9]+\$
force	Number	Remove old agent with same IP if disconnected since <force> seconds.

Example Request:

```
curl -u foo:bar -k -X POST -d
'{"name": "NewHost_2", "ip": "10.0.10.10", "id": "123", "key": "1abcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyzabcdefghijklmnopqrstuvwxyz64"}' -H 'Content-Type:application/json' "https://127.0.0.1:55000/agents/insert?pretty"
```

Example Response:

```
{
  "error": 0,
  "data": "123"
}
```

Delete

Delete a list of agents

Removes a list of agents. You must restart OSSEC after removing an agent.

Request:

DELETE

/agents

Parameters:

Param	Type	Description
ids	String[]	Array of agent ID's.

Example Request:

```
curl -u foo:bar -k -X DELETE -H "Content-Type:application/json" -d '{"ids": ["001", "002"]}' "https://127.0.0.1:55000/agents?pretty"
```

Example Response:

```
{
  "error": 0,
  "data": {
    "msg": "All selected agents were removed"
  }
}
```

Delete an agent

Removes an agent. You must restart OSSEC after removing an agent.

Request:

DELETE

```
/agents/:agent_id
```

Parameters:

Param	Type	Description
agent_id	Number	Agent ID.

Example Request:

```
curl -u foo:bar -k -X DELETE "https://127.0.0.1:55000/agents/002?pretty"
```

Example Response:

```
{
  "error": 0,
  "data": {
    "msg": "Some agents were not removed",
    "ids": [
      "002"
    ]
  }
}
```

Info

Get OS summary

Returns a summary of OS.

Request:

GET

```
/agents/summary/os
```

Parameters:

Param	Type	Description
offset	Number	First element to return in the collection.
limit	Number	Maximum number of elements to return.
sort	String	Sorts the collection by a field or fields (separated by comma). Use +/- at the beginning to ascending or descending order.
search	String	Looks for elements with the specified string.

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/agents/summary/os?pretty"
```

Example Response:

```
{  
  "error": 0,  
  "data": {  
    "totalItems": 3,  
    "items": [  
      "debian",  
      "ubuntu",  
      "windows"  
    ]  
  }  
}
```

Get agents summary

Returns a summary of the available agents.

Request:

GET

```
/agents/summary
```

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/agents/summary?pretty"
```

Example Response:

```
{  
  "error": 0,  
  "data": {  
    "Active": 3,  
    "Never connected": 3,  
    "Total": 6,  
    "Disconnected": 0  
  }  
}
```

Get all agents

Returns a list with the available agents.

Request:

GET

```
/agents
```

Parameters:

Param	Type	Description
offset	Number	First element to return in the collection.
limit	Number	Maximum number of elements to return.
sort	String	Sorts the collection by a field or fields (separated by comma). Use +/- at the beginning to ascending or descending order.
search	String	Looks for elements with the specified string.

Param	Type	Description
status	string	<p>Filters by agent status.</p> <p>Allowed values:</p> <ul style="list-style-type: none"> • active • never connected • disconnected
os.platform	String	Filters by OS platform
os.version	String	Filters by OS version

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/agents?pretty&offset=0&limit=5&sort=-ip,name"
```

Example Response:

```
{
  "error": 0,
  "data": {
    "totalItems": 6,
    "items": [
      {
        "status": "Active",
        "ip": "any",
        "os": {
          "platform": "windows",
          "version": "10.0.14393",
          "name": "Microsoft Windows Server 2016 Datacenter"
        },
        "id": "004",
        "name": "win_server"
      },
      {
        "status": "Active",
        "ip": "any",
        "os": {
          "platform": "ubuntu",
          "version": "16.04.2 LTS",
          "name": "Ubuntu"
        },
        "id": "003",
        "name": "u16"
      },
      {
        "status": "Never connected",
        "ip": "any",
        "id": "006",
        "name": "myNewAgent"
      },
      {
        "status": "Never connected",
        "ip": "10.0.10.10",
        "id": "123",
        "name": "NewHost_2"
      },
      {
        "status": "Never connected",
        "ip": "10.0.0.9",
        "id": "005",
        "name": "NewHost"
      }
    ]
  }
}
```

Get an agent

Returns the information of an agent.

Request:

GET

/agents/:agent_id

Parameters:

Param	Type	Description
agent_id	Number	Agent ID.

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/agents/000?pretty"
```

Example Response:

```
{  
    "error": 0,  
    "data": {  
        "status": "Active",  
        "name": "ip-10-0-0-10",  
        "ip": "127.0.0.1",  
        "dateAdd": "2017-08-05 14:47:01",  
        "version": "Wazuh v2.1.0",  
        "lastKeepAlive": "9999-12-31 23:59:59",  
        "os": {  
            "major": "8",  
            "name": "Debian GNU/Linux",  
            "platform": "debian",  
            "uname": "Linux ip-10-0-0-10 3.16.0-4-amd64 #1 SMP Debian 3.16.36-1+deb8u2 (2016-10-19) x86_64",  
            "version": "8",  
            "codename": "jessie"  
        },  
        "id": "000"  
    }  
}
```

Key

Get agent key

Returns the key of an agent.

Request:

GET

/agents/:agent_id/key

Parameters:

Param	Type	Description
agent_id	Number	Agent ID.

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/agents/001/key?pretty"
```

Example Response:

```
{  
    "error": 0,  
    "data": "  
MDAxIFdlYlNlcnZlcjEgMTAuMC4wLjYyIDNlZjEwYTQ2MGZmZDEwNDlhNDhiMmI1NjRjZmFiNGQxNmFiYzIzMzQ2NDM3MWY0ODQwZDQ0ZDJj  
N2RkNDkwZTE"  
}
```

Restart

Restart a list of agents

Restarts a list of agents.

Request:

POST

```
/agents/restart
```

Parameters:

Param	Type	Description
ids	String[]	Array of agent ID's.

Example Request:

```
curl -u foo:bar -k -X POST -H "Content-Type:application/json" -d '{"ids":["001","002"]}' "https://127.0.0.1:55000/agents/restart?pretty"
```

Example Response:

```
{
  "error": 0,
  "data": {
    "msg": "Some agents were not restarted",
    "ids": [
      "001",
      "002"
    ]
  }
}
```

Restart all agents

Restarts all agents.

Request:

PUT

```
/agents/restart
```

Example Request:

```
curl -u foo:bar -k -X PUT "https://127.0.0.1:55000/agents/restart?pretty"
```

Example Response:

```
{
  "data": "Restarting all agents",
  "error": 0
}
```

Restart an agent

Restarts the agent.

Request:

PUT

/agents/:agent_id/restart

Parameters:

Param	Type	Description
agent_id	Number	Agent unique ID.

Example Request:

```
curl -u foo:bar -k -X PUT "https://127.0.0.1:55000/agents/000/restart?pretty"
```

Example Response:

```
{  
    "data": "Restarting agent",  
    "error": 0  
}
```

Decoders

Info

Get all decoders

Returns all decoders included in ossec.conf.

Request:

GET

/decoders

Parameters:

Param	Type	Description
offset	Number	First element to return in the collection.
limit	Number	Maximum number of elements to return.
sort	String	Sorts the collection by a field or fields (separated by comma). Use +/- at the beginning to ascending or descending order.
search	String	Looks for elements with the specified string.
file	String	Filters by filename.
path	String	Filters by path.
		Filters the decoders by status. Allowed values: <ul style="list-style-type: none">• enabled• disabled• all
status	String	

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/decoders?pretty&offset=0&limit=2&sort=+file,position"
```

Example Response:

```
{  
  "error": 0,  
  "data": {  
    "totalItems": 480,  
    "items": [  
      {  
        "status": "enabled",  
        "name": "wazuh",  
        "details": {  
          "prematch": "^wazuh:",  
          "file": "0005-wazuh_decoders.xml",  
          "position": 0,  
          "path": "/var/ossec/ruleset/decoders"  
        },  
        {  
          "status": "enabled",  
          "name": "agent-buffer",  
          "details": {  
            "regex": "^\s+",  
            "prematch": "Agent buffer:",  
            "parent": "wazuh",  
            "order": "status"  
          },  
          "file": "0005-wazuh_decoders.xml",  
          "position": 1,  
          "path": "/var/ossec/ruleset/decoders"  
        }  
      ]  
    }  
  }  
}
```

Get all decoders files

Returns all decoders files included in ossec.conf.

Request:

GET

/decoders/files

Parameters:

Param	Type	Description
offset	Number	First element to return in the collection.
limit	Number	Maximum number of elements to return.
sort	String	Sorts the collection by a field or fields (separated by comma). Use +/- at the beginning to ascending or descending order.
search	String	Looks for elements with the specified string.

Param	Type	Description
status	String	<p>Filters the decoders by status.</p> <p>Allowed values:</p> <ul style="list-style-type: none"> • enabled • disabled • all
file	String	Filters by filename.
path	String	Filters by path.
download	String	Downloads the file

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/decoders/files?pretty&offset=0&limit=10&sort=-path"
```

Example Response:

```
{
  "error": 0,
  "data": {
    "totalItems": 85,
    "items": [
      {
        "status": "enabled",
        "path": "/var/ossec/ruleset/decoders",
        "file": "0060-cisco-estreamer_decoders.xml"
      },
      {
        "status": "enabled",
        "path": "/var/ossec/ruleset/decoders",
        "file": "0150-mysql_decoders.xml"
      },
      {
        "status": "enabled",
        "path": "/var/ossec/ruleset/decoders",
        "file": "0215-portsentry_decoders.xml"
      },
      {
        "status": "enabled",
        "path": "/var/ossec/ruleset/decoders",
        "file": "0325-suhosin_decoders.xml"
      },
      {
        "status": "enabled",
        "path": "/var/ossec/ruleset/decoders",
        "file": "0190-openvpn_decoders.xml"
      },
      {
        "status": "enabled",
        "path": "/var/ossec/ruleset/decoders",
        "file": "0405-mongodb_decoders.xml"
      },
      {
        "status": "enabled",
        "path": "/var/ossec/ruleset/decoders",
        "file": "0090-dragon-nids_decoders.xml"
      },
      {
        "status": "enabled",
        "path": "/var/ossec/ruleset/decoders",
        "file": "0085-dovecot_decoders.xml"
      },
      {
        "status": "enabled",
        "path": "/var/ossec/ruleset/decoders",
        "file": "0335-telnet_decoders.xml"
      },
      {
        "status": "enabled",
        "path": "/var/ossec/ruleset/decoders",
        "file": "0165-netscreen_decoders.xml"
      }
    ]
  }
}
```

Get all parent decoders

Returns all parent decoders included in ossec.conf

Request:

GET

/decoders/parents

Parameters:

Param	Type	Description
offset	Number	First element to return in the collection.
limit	Number	Maximum number of elements to return.
sort	String	Sorts the collection by a field or fields (separated by comma). Use +/- at the beginning to ascending or descending order.
search	String	Looks for elements with the specified string.

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/decoders/parents?pretty&offset=0&limit=2&sort=-file"
```

Example Response:

```
{
  "error": 0,
  "data": {
    "totalItems": 121,
    "items": [
      {
        "status": "enabled",
        "name": "local_decoder_example",
        "details": {
          "program_name": "local_decoder_example"
        },
        "file": "local_decoder.xml",
        "position": 0,
        "path": "/var/ossec/etc/decoders"
      },
      {
        "status": "enabled",
        "name": "jenkins",
        "details": {
          "prematch": "^\\w+ \\d+, \\d+ \\d+:\\d+:\\d+ \\w\\w \\S+ \\w+\\s"
        },
        "file": "0415-jenkins_decoders.xml",
        "position": 0,
        "path": "/var/ossec/ruleset/decoders"
      }
    ]
  }
}
```

Get decoders by name

Returns the decoders with the specified name.

Request:

GET

/decoders/:decoder_name

Parameters:

Param	Type	Description
decoder_name	String	Decoder name.
offset	Number	First element to return in the collection.
limit	Number	Maximum number of elements to return.
sort	String	Sorts the collection by a field or fields (separated by comma). Use +/- at the beginning to ascending or descending order.
search	String	Looks for elements with the specified string.

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/decoders/apache-errorlog?pretty"
```

Example Response:

```
{
  "error": 0,
  "data": {
    "totalItems": 3,
    "items": [
      {
        "status": "enabled",
        "name": "apache-errorlog",
        "details": {
          "program_name": "^apache2|^httpd"
        },
        "file": "0025-apache_decoders.xml",
        "position": 0,
        "path": "/var/ossec/ruleset/decoders"
      },
      {
        "status": "enabled",
        "name": "apache-errorlog",
        "details": {
          "prematch": "[warn] |[notice] |[error] "
        },
        "file": "0025-apache_decoders.xml",
        "position": 1,
        "path": "/var/ossec/ruleset/decoders"
      },
      {
        "status": "enabled",
        "name": "apache-errorlog",
        "details": {
          "prematch": "[\\w+ \\w+ \\d+ \\d+:\\d+:\\d+.\\d+ \\d+] [\\S*:warn] |[\\w+ \\w+ \\d+ \\d+:\\d+:\\d+.\\d+ \\d+] [\\S*:notice] |[\\w+ \\w+ \\d+ \\d+:\\d+:\\d+.\\d+ \\d+] [\\S*:error] |[\\w+ \\w+ \\d+ \\d+:\\d+:\\d+.\\d+ \\d+] [\\S*:info] "
        },
        "file": "0025-apache_decoders.xml",
        "position": 2,
        "path": "/var/ossec/ruleset/decoders"
      }
    ]
  }
}
```

Manager

Configuration

Get manager configuration

Returns ossec.conf in JSON format.

Request:

GET

```
/manager/configuration
```

Parameters:

Param	Type	Description
section	String	Indicates the ossec.conf section: global, rules, syscheck, rootcheck, remote, alerts, command, active-response, localfile.
field	String	Indicates a section child, e.g, fields for rule section are: include, decoder_dir, etc.

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/manager/configuration?section=global&pretty"
```

Example Response:

```
{
  "error": 0,
  "data": {
    "email_notification": "no",
    "alerts_log": "yes",
    "jsonout_output": "yes",
    "smtp_server": "smtp.example.wazuh.com",
    "email_to": "recipient@example.wazuh.com",
    "logall": "no",
    "email_maxperhour": "12",
    "white_list": [
      "127.0.0.1",
      "^localhost.localdomain$",
      "10.0.0.2"
    ],
    "email_from": "ossecm@example.wazuh.com",
    "logall_json": "no"
  }
}
```

Info

Get manager information

Returns basic information about Manager.

Request:

GET

```
/manager/info
```

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/manager/info?pretty"
```

Example Response:

```
{
  "error": 0,
  "data": {
    "installation_date": "Sat Aug  5 14:46:32 UTC 2017",
    "version": "v2.1.0",
    "openssl_support": "yes",
    "max_agents": "8000",
    "ruleset_version": "v2.1.0",
    "path": "/var/ossec",
    "tz_name": "UTC",
    "type": "server",
    "tz_offset": "+0000"
  }
}
```

Get manager status

Returns the Manager processes that are running.

Request:

GET

```
/manager/status
```

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/manager/status?pretty"
```

Example Response:

```
{
  "error": 0,
  "data": {
    "wazuh-modulesd": "running",
    "ossec-authd": "stopped",
    "ossec-monitord": "running",
    "ossec-logcollector": "running",
    "ossec-execd": "running",
    "ossec-remoted": "running",
    "ossec-syscheckd": "running",
    "ossec-analysisd": "running",
    "ossec-maild": "stopped"
  }
}
```

Logs

Get ossec.log

Returns the 3 last months of ossec.log.

Request:

GET

```
/manager/logs
```

Parameters:

Param	Type	Description
-------	------	-------------

Param	Type	Description
offset	Number	First element to return in the collection.
limit	Number	Maximum number of elements to return.
sort	String	Sorts the collection by a field or fields (separated by comma). Use +/- at the beginning to ascending or descending order.
search	String	Looks for elements with the specified string.
		Filters by type of log. Allowed values: <ul style="list-style-type: none">• all• error• info
category	string	Filters by category of log.

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/manager/logs?offset=0&limit=5&pretty"
```

Example Response:

```
{
  "data": {
    "totalItems": 16480,
    "items": [
      "2016/07/15 09:33:49 ossec-syscheckd: INFO: Syscheck scan frequency: 3600 seconds",
      "2016/07/15 09:33:49 ossec-syscheckd: INFO: Starting syscheck scan (forwarding database).",
      "2016/07/15 09:33:49 ossec-syscheckd: INFO: Starting syscheck database (pre-scan).",
      "2016/07/15 09:33:42 ossec-logcollector: INFO: Started (pid: 2832).",
      "2016/07/15 09:33:42 ossec-logcollector: INFO: Monitoring output of command(360): df -P"
    ]
  },
  "error": 0
}
```

Get summary of ossec.log

Returns a summary about the 3 last months of ossec.log.

Request:

GET

/manager/logs/summary

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/manager/logs/summary?pretty"
```

Example Response:

```
{  
  "error": 0,  
  "data": {  
    "wazuh-modulesd": {  
      "info": 2,  
      "all": 2,  
      "error": 0  
    },  
    "ossec-testrule": {  
      "info": 172,  
      "all": 172,  
      "error": 0  
    },  
    "wazuh-modulesd:oscap": {  
      "info": 2,  
      "all": 2,  
      "error": 0  
    },  
    "ossec-rootcheck": {  
      "info": 6,  
      "all": 6,  
      "error": 0  
    },  
    "ossec-monitord": {  
      "info": 3,  
      "all": 3,  
      "error": 0  
    },  
    "ossec-logcollector": {  
      "info": 25,  
      "all": 27,  
      "error": 2  
    },  
    "ossec-execd": {  
      "info": 4,  
      "all": 4,  
      "error": 0  
    },  
    "ossec-remoted": {  
      "info": 416,  
      "all": 1047,  
      "error": 631  
    },  
    "ossec-syscheckd": {  
      "info": 51,  
      "all": 51,  
      "error": 0  
    },  
    "ossec-analysisd": {  
      "info": 389,  
      "all": 389,  
      "error": 0  
    },  
    "wazuh-modulesd:database": {  
      "info": 2,  
      "all": 2,  
      "error": 0  
    }  
  }  
}
```

Stats

Get manager stats

Returns OSSEC statistical information of current date.

Request:

GET

```
/manager/stats
```

Parameters:

Param	Type	Description
date	String	Selects the date for getting the statistical information. Format: YYYYMMDD

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/manager/stats?pretty"
```

Example Response:

```
{
  "data": [
    {
      "hour": 5,
      "firewall": 0,
      "alerts": [
        {
          "level": 3,
          "sigid": 5715,
          "times": 4
        },
        {
          "level": 2,
          "sigid": 1002,
          "times": 2
        },
        {
          "...": ...
        }
      ],
      "totalAlerts": 107,
      "syscheck": 1257,
      "events": 1483
    },
    {
      "...": ...
    }
  ],
  "error": 0
}
```

Get manager stats by hour

Returns OSSEC statistical information per hour. Each item in averages field represents the average of alerts per hour.

Request:

GET

```
/manager/stats/hourly
```

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/manager/stats/hourly?pretty"
```

Example Response:

```
{  
  "data": {  
    "averages": [  
      100,  
      357,  
      242,  
      500,  
      422,  
      "...",  
      123  
    ],  
    "interactions": 0  
  },  
  "error": 0  
}
```

Get manager stats by week

Returns OSSEC statistical information per week. Each item in hours field represents the average of alerts per hour and week day.

Request:

GET

```
/manager/stats/weekly
```

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/manager/stats/weekly?pretty"
```

Example Response:

```
{  
  "data": {  
    "Wed": {  
      "hours": [  
        223,  
        "...",  
        456  
      ],  
      "interactions": 0  
    },  
    "Sun": {  
      "hours": [  
        332,  
        "...",  
        313  
      ],  
      "interactions": 0  
    },  
    "Thu": {  
      "hours": [  
        888,  
        "...",  
        123  
      ],  
      "interactions": 0  
    },  
    "Tue": {  
      "hours": [  
        536,  
        "...",  
        345  
      ],  
      "interactions": 0  
    },  
    "Mon": {  
      "hours": [  
        444,  
        "...",  
        556  
      ],  
      "interactions": 0  
    },  
    "Fri": {  
      "hours": [  
        131,  
        "...",  
        432  
      ],  
      "interactions": 0  
    },  
    "Sat": {  
      "hours": [  
        134,  
        "...",  
        995  
      ],  
      "interactions": 0  
    }  
  "error": 0  
}
```

Rootcheck

Clear

Clear rootcheck database

Clears the rootcheck database for all agents.

Request:

`DELETE`

`/rootcheck`

Example Request:

```
curl -u foo:bar -k -X DELETE "https://127.0.0.1:55000/rootcheck?pretty"
```

Example Response:

```
{
  "data": "Rootcheck database deleted",
  "error": 0
}
```

Clear rootcheck database of an agent

Clears the rootcheck database for an agent.

Request:

`DELETE`

`/rootcheck/:agent_id`

Parameters:

Param	Type	Description
<code>agent_id</code>	Number	Agent ID.

Example Request:

```
curl -u foo:bar -k -X DELETE "https://127.0.0.1:55000/rootcheck/000?pretty"
```

Example Response:

```
{
  "data": "Rootcheck database deleted",
  "error": 0
}
```

Info

Get last rootcheck scan

Return the timestamp of the last rootcheck scan.

Request:

`GET`

`/rootcheck/:agent_id/last_scan`

Parameters:

Param	Type	Description
agent_id	Number	Agent ID.

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/rootcheck/000/last_scan?pretty"
```

Example Response:

```
{
  "error": 0,
  "data": {
    "start": "2017-08-05 14:48:07",
    "end": "2017-08-05 14:47:33"
  }
}
```

Get rootcheck CIS requirements

Returns the CIS requirements of all rootchecks of the agent.

Request:**GET**

```
/rootcheck/:agent_id/cis
```

Parameters:

Param	Type	Description
offset	Number	First element to return in the collection.
limit	Number	Maximum number of elements to return.
sort	String	Sorts the collection by a field or fields (separated by comma). Use +/- at the beginning to ascending or descending order.
search	String	Looks for elements with the specified string.

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/rootcheck/000/cis?offset=0&limit=10&pretty"
```

Example Response:

```
{
  "error": 0,
  "data": {
    "totalItems": 2,
    "items": [
      "1.4 Debian Linux",
      "4.13 Debian Linux"
    ]
  }
}
```

Get rootcheck database

Returns the rootcheck database of an agent.

Request:

GET

```
/rootcheck/:agent_id
```

Parameters:

Param	Type	Description
agent_id	Number	Agent ID.
pci	String	Filters by pci requirement.
cis	String	Filters by CIS.
offset	Number	First element to return in the collection.
limit	Number	Maximum number of elements to return.
sort	String	Sorts the collection by a field or fields (separated by comma). Use +/- at the beginning to ascending or descending order.
search	String	Looks for elements with the specified string.

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/rootcheck/000?offset=0&limit=2&pretty"
```

Example Response:

```
{
  "error": 0,
  "data": {
    "totalItems": 8,
    "items": [
      {
        "status": "outstanding",
        "oldDay": "2017-08-05 14:47:18",
        "readDay": "2017-08-05 14:48:08",
        "event": "File '/var/lib/test' is owned by root and has written permissions to anyone."
      },
      {
        "status": "outstanding",
        "oldDay": "2017-08-05 14:47:18",
        "cis": "1.4 Debian Linux",
        "readDay": "2017-08-05 14:48:07",
        "event": "System Audit: CIS - Debian Linux - 1.4 - Robust partition scheme - /opt is not on its own partition {CIS: 1.4 Debian Linux}. File: /opt. Reference: https://benchmarks.cisecurity.org/tools2/linux/CIS_Debian_Benchmark_v1.0.pdf ."
      }
    ]
  }
}
```

Get rootcheck pci requirements

Returns the PCI requirements of all rootchecks of the agent.

Request:

GET

```
/rootcheck/:agent_id/pci
```

Parameters:

Param	Type	Description
offset	Number	First element to return in the collection.
limit	Number	Maximum number of elements to return.
sort	String	Sorts the collection by a field or fields (separated by comma). Use +/- at the beginning to ascending or descending order.
search	String	Looks for elements with the specified string.

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/rootcheck/000/pci?offset=0&limit=10&pretty"
```

Example Response:

```
{
  "error": 0,
  "data": {
    "totalItems": 2,
    "items": [
      "2.2.2",
      "2.2.4"
    ]
  }
}
```

Run

Run rootcheck scan in all agents

Runs syscheck and rootcheck on all agent, due to OSSEC launches both processes at once.

Request:

PUT

```
/rootcheck
```

Example Request:

```
curl -u foo:bar -k -X PUT "https://127.0.0.1:55000/rootcheck?pretty"
```

Example Response:

```
{
  "data": "Restarting Syscheck/Rootcheck on all agents",
  "error": 0
}
```

Run rootcheck scan in an agent

Runs syscheck and rootcheck on an agent, due to OSSEC launches both processes at once.

Request:

PUT

/rootcheck/:agent_id

Parameters:

Param	Type	Description
agent_id	Number	Agent ID.

Example Request:

```
curl -u foo:bar -k -X PUT "https://127.0.0.1:55000/rootcheck/000?pretty"
```

Example Response:

```
{  
    "error": 0,  
    "data": "Restarting Syscheck/Rootcheck locally"  
}
```

Rules

Info

Get all rules

Returns all rules.

Request:

GET

/rules

Parameters:

Param	Type	Description
offset	Number	First element to return in the collection.
limit	Number	Maximum number of elements to return.
sort	String	Sorts the collection by a field or fields (separated by comma). Use +/- at the beginning to ascending or descending order.
search	String	Looks for elements with the specified string.
		Filters the rules by status. Allowed values: <ul style="list-style-type: none">• enabled• disabled• all
status	String	
group	String	Filters the rules by group.
level	Range	Filters the rules by level. level=2 or level=2-5.
path	String	Filters the rules by path.

Param	Type	Description
file	String	Filters the rules by file name.
pci	String	Filters the rules by pci requirement.

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/rules?offset=0&limit=2&pretty"
```

Example Response:

```
{
  "error": 0,
  "data": {
    "totalItems": 1701,
    "items": [
      {
        "status": "enabled",
        "pci": [],
        "description": "Generic template for all syslog rules.",
        "file": "0010-rules_config.xml",
        "level": 0,
        "path": "/var/ossec/ruleset/rules",
        "groups": [
          "syslog"
        ],
        "id": 1,
        "details": {
          "category": "syslog",
          "noalert": "1"
        }
      },
      {
        "status": "enabled",
        "pci": [],
        "description": "Generic template for all firewall rules.",
        "file": "0010-rules_config.xml",
        "level": 0,
        "path": "/var/ossec/ruleset/rules",
        "groups": [
          "firewall"
        ],
        "id": 2,
        "details": {
          "category": "firewall",
          "noalert": "1"
        }
      }
    ]
  }
}
```

Get files of rules

Returns the files of all rules.

Request:

GET

/rules/files

Parameters:

Param	Type	Description
offset	Number	First element to return in the collection.
limit	Number	Maximum number of elements to return.
sort	String	Sorts the collection by a field or fields (separated by comma). Use +/- at the beginning to ascending or descending order.
search	String	Looks for elements with the specified string.
status	String	Filters files by status. Allowed values: <ul style="list-style-type: none">• enabled• disabled• all
path	String	Filters the rules by path.
file	String	Filters the rules by filefile.
download	String	Downloads the file

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/rules/files?offset=0&limit=10&pretty"
```

Example Response:

```
{
  "error": 0,
  "data": {
    "totalItems": 95,
    "items": [
      {
        "status": "enabled",
        "path": "/var/ossec/ruleset/rules",
        "file": "0010-rules_config.xml"
      },
      {
        "status": "enabled",
        "path": "/var/ossec/ruleset/rules",
        "file": "0015-ossec_rules.xml"
      },
      {
        "status": "enabled",
        "path": "/var/ossec/ruleset/rules",
        "file": "0016-wazuh_rules.xml"
      },
      {
        "status": "enabled",
        "path": "/var/ossec/ruleset/rules",
        "file": "0020-syslog_rules.xml"
      },
      {
        "status": "enabled",
        "path": "/var/ossec/ruleset/rules",
        "file": "0025-sendmail_rules.xml"
      },
      {
        "status": "enabled",
        "path": "/var/ossec/ruleset/rules",
        "file": "0030-postfix_rules.xml"
      },
      {
        "status": "enabled",
        "path": "/var/ossec/ruleset/rules",
        "file": "0035-spamd_rules.xml"
      },
      {
        "status": "enabled",
        "path": "/var/ossec/ruleset/rules",
        "file": "0040-imapd_rules.xml"
      },
      {
        "status": "enabled",
        "path": "/var/ossec/ruleset/rules",
        "file": "0045-mailscanner_rules.xml"
      },
      {
        "status": "enabled",
        "path": "/var/ossec/ruleset/rules",
        "file": "0050-ms-exchange_rules.xml"
      }
    ]
  }
}
```

Get rule groups

Returns the groups of all rules.

Request:

GET

/rules/groups

Parameters:

Param	Type	Description
offset	Number	First element to return in the collection.
limit	Number	Maximum number of elements to return.
sort	String	Sorts the collection by a field or fields (separated by comma). Use +/- at the beginning to ascending or descending order.
search	String	Looks for elements with the specified string.

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/rules/groups?offset=0&limit=10&pretty"
```

Example Response:

```
{
  "error": 0,
  "data": {
    "totalItems": 242,
    "items": [
      "access_control",
      "access_denied",
      "accesslog",
      "account_changed",
      "active_response",
      "adduser",
      "agent",
      "agent_flooding",
      "agentless",
      "amazon"
    ]
  }
}
```

Get rule pci requirements

Returns the PCI requirements of all rules.

Request:

GET

/rules/pci

Parameters:

Param	Type	Description
offset	Number	First element to return in the collection.
limit	Number	Maximum number of elements to return.
sort	String	Sorts the collection by a field or fields (separated by comma). Use +/- at the beginning to ascending or descending order.
search	String	Looks for elements with the specified string.

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/rules/pci?offset=0&limit=10&pretty"
```

Example Response:

```
{  
    "error": 0,  
    "data": {  
        "totalItems": 38,  
        "items": [  
            "1.1.1",  
            "1.3.4",  
            "1.4",  
            "10.1",  
            "10.2.1",  
            "10.2.2",  
            "10.2.4",  
            "10.2.5",  
            "10.2.6",  
            "10.2.7"  
        ]  
    }  
}
```

Get rules by id

Returns the rules with the specified id.

Request:

GET

```
/rules/:rule_id
```

Parameters:

Param	Type	Description
id	Number	rule.
offset	Number	First element to return in the collection.
limit	Number	Maximum number of elements to return.
sort	String	Sorts the collection by a field or fields (separated by comma). Use +/- at the beginning to ascending or descending order.
search	String	Looks for elements with the specified string.

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/rules/1002?pretty"
```

Example Response:

```
{  
  "error": 0,  
  "data": {  
    "totalItems": 1,  
    "items": [  
      {  
        "status": "enabled",  
        "pci": [],  
        "description": "Unknown problem somewhere in the system.",  
        "file": "0020-syslog_rules.xml",  
        "level": 2,  
        "path": "/var/ossec/ruleset/rules",  
        "groups": [  
          "syslog",  
          "errors"  
        ],  
        "id": 1002,  
        "details": {  
          "options": "alert_by_email",  
          "match": "$BAD_WORDS"  
        }  
      }  
    ]  
  }  
}
```

Syscheck

Clear

Clear syscheck database

Clears the syscheck database for all agents.

Request:

[DELETE](#)

```
/syscheck
```

Example Request:

```
curl -u foo:bar -k -X DELETE "https://127.0.0.1:55000/syscheck?pretty"
```

Example Response:

```
{  
  "data": "Syscheck database deleted",  
  "error": 0  
}
```

Clear syscheck database of an agent

Clears the syscheck database for an agent.

Request:

[DELETE](#)

```
/syscheck/:agent_id
```

Parameters:

Param	Type	Description
agent_id	Number	Agent ID.

Example Request:

```
curl -u foo:bar -k -X DELETE "https://127.0.0.1:55000/syscheck/000?pretty"
```

Example Response:

```
{  
  "data": "Syscheck database deleted",  
  "error": 0  
}
```

Info

Get last syscheck scan

Return the timestamp of the last syscheck scan.

Request:

GET

```
/syscheck/:agent_id/last_scan
```

Parameters:

Param	Type	Description
agent_id	Number	Agent ID.

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/syscheck/000/last_scan?pretty"
```

Example Response:

```
{  
  "error": 0,  
  "data": {  
    "start": "2017-08-05 14:48:04",  
    "end": "2017-08-05 14:48:07"  
  }  
}
```

Get syscheck files

Returns the syscheck files of an agent.

Request:

GET

```
/syscheck/:agent_id
```

Parameters:

Param	Type	Description
agent_id	Number	Agent ID.
offset	Number	First element to return in the collection.
limit	Number	Maximum number of elements to return.
sort	String	Sorts the collection by a field or fields (separated by comma). Use +/- at the beginning to ascending or descending order.
search	String	Looks for elements with the specified string.
event	String	<p>Filters files by event. Allowed values:</p> <ul style="list-style-type: none">• added• readded• modified• deleted
file	String	Filters file by filename.
filetype	String	Selects type of file. Allowed values: <ul style="list-style-type: none">• file• registry
summary	String	Returns a summary grouping by filename. Allowed values: <ul style="list-style-type: none">• yes• no
md5	String	Returns the files with the specified md5 hash.
sha1	String	Returns the files with the specified sha1 hash.
hash	String	Returns the files with the specified hash (md5 or sha1).

Example Request:

```
curl -u foo:bar -k -X GET "https://127.0.0.1:55000/syscheck/000?offset=0&limit=2&pretty"
```

Example Response:

```
{
  "error": 0,
  "data": {
    "totalItems": 0,
    "items": []
  }
}
```

Run

Run syscheck scan in all agents

Runs syscheck and rootcheck on all agent, due to OSSEC launches both processes at once.

Request:

PUT

/syscheck

Example Request:

```
curl -u foo:bar -k -X PUT "https://127.0.0.1:55000/syscheck?pretty"
```

Example Response:

```
{  
  "data": "Restarting Syscheck/Rootcheck on all agents",  
  "error": 0  
}
```

Run syscheck scan in an agent

Runs syscheck and rootcheck on an agent, due to OSSEC launches both processes at once.

Request:

PUT

/syscheck/:agent_id

Parameters:

Param	Type	Description
agent_id	Number	Agent ID.

Example Request:

```
curl -u foo:bar -k -X PUT "https://127.0.0.1:55000/syscheck/000?pretty"
```

Example Response:

```
{  
  "error": 0,  
  "data": "Restarting Syscheck/Rootcheck locally"  
}
```

Examples

CURL

cURL is a command-line tool for sending http/https requests and commands. It can be used to interact with the API. It is pre-installed on many Linux and Mac systems. Some examples:

GET

```
$ curl -u foo:bar -k https://127.0.0.1:55000
```



```
{"error": "0", "data": "Welcome to Wazuh HIDS API"}
```

PUT

```
$ curl -u foo:bar -k -X PUT https://127.0.0.1:55000/agents/new_agent
```

```
{"error": 0, "data": "004"}
```

POST

```
$ curl -u foo:bar -k -X POST -d '{"name": "NewHost", "ip": "10.0.0.8"}' -H 'Content-Type:application/json' "https://127.0.0.1:55000//agents"
```

```
{"error": 0, "data": "004"}
```

DELETE

```
$ curl -u foo:bar -k -X DELETE https://127.0.0.1:55000/rootcheck/001
```

```
{"error": "0", "data": "Policy and auditing database updated"}
```

Python

It is very easy to interact with the API using Python:

Code:

```

#!/usr/bin/env python

import json
import requests # To install requests, use: pip install requests

# Configuration
base_url = 'https://IP:55000'
auth = requests.auth.HTTPBasicAuth('foo', 'bar')
verify = False
requests.packages.urllib3.disable_warnings()

# Request
url = '{0}{1}'.format(base_url, "/agents/000")
r = requests.get(url, auth=auth, params=None, verify=verify)
print(json.dumps(r.json(), indent=4, sort_keys=True))
print("Status: {0}".format(r.status_code))

```

Output:

```

{
  "error": "0",
  "data": {
    "id": "000",
    "ip": "127.0.0.1",
    "lastKeepAlive": "Not available",
    "name": "LinMV",
    "os": "Linux LinMV 3.16.0-4-amd64 #1 SMP Debian 3.16.7-ckt11-1 (2015-05-24) x86_64",
    "rootcheckEndTime": "Unknown",
    "rootcheckTime": "Unknown",
    "status": "Active",
    "syscheckEndTime": "Unknown",
    "syscheckTime": "Unknown",
    "version": "OSSEC HIDS v2.8"
  }
}
Status: 200

```

For a fuller example, see [/var/ossec/api/examples/api-client.py](#).

PowerShell

The **Invoke-RestMethod** cmdlet sends requests to the API and handles the response easily. This cmdlet was introduced in Windows PowerShell 3.0.

Code:

```

function Ignore-SelfSignedCerts {
    add-type @"
        using System.Net;
        using System.Security.Cryptography.X509Certificates;

        public class PolicyCert : ICertificatePolicy {
            public PolicyCert() {}
            public bool CheckValidationResult(
                ServicePoint sPoint, X509Certificate cert,
                WebRequest wRequest, int certProb) {
                return true;
            }
        }
    "@
    [System.Net.ServicePointManager]::CertificatePolicy = new-object PolicyCert
}

# Configuration
$base_url = "https://IP:55000"
$username = "foo"
$password = "bar"
$base64AuthInfo = [Convert]::ToBase64String([Text.Encoding]::ASCII.GetBytes(("{}:{}" -f $username, $password)))
Ignore-SelfSignedCerts

# Request
$url = $base_url + "/syscheck/000/last_scan"
$method = "get"
try{
    $r = Invoke-RestMethod -Headers @{"Authorization=Basic {0}" -f $base64AuthInfo} -Method $method -Uri $url
}catch{
    $r = $_.Exception
}

Write-Output $r

```

Output:

```

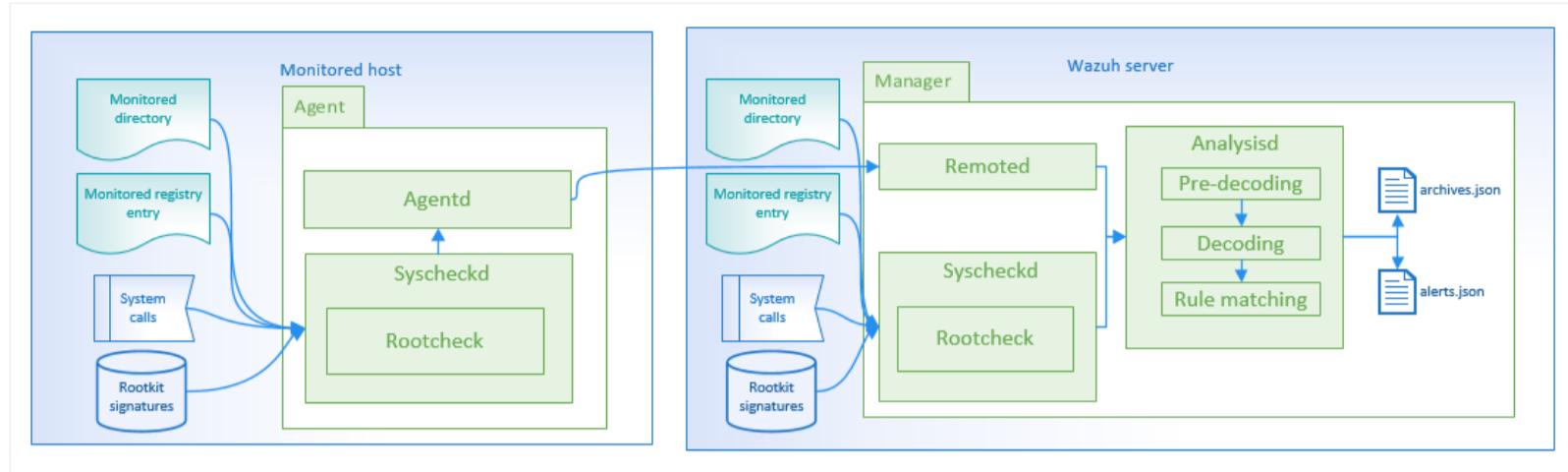
error data
-----
0  @{syscheckTime=Wed Feb 24 09:55:04 2016; syscheckEndTime=Wed Feb 24 10:00:42 2016}

```

For a fuller example, see </var/ossec/api/examples/api-client.ps1>.

How it works

This section describes the checks performed by Wazuh to find the anomalies caused by an intruder or malware.



File integrity monitoring

Malware replaces files, directories and commands, so performing file integrity checking on the main directories allows us to detect these actions. More info [File Integrity Monitoring Section](#)

Example:

```
** Alert 1460948255.25442: mail - ossec,syscheck,pci_dss_11.5,
2016 Apr 17 19:57:35 (ubuntu) 10.0.0.144->syscheck
Rule: 550 (level 7) -> 'Integrity checksum changed.'
Integrity checksum changed for: '/test/hello'
Size changed from '12' to '17'
Old md5sum was: 'e59ff97941044f85df5297e1c302d260'
New md5sum is : '7947eba5d9cc58d440fb06912e302949'
Old sha1sum was: '648a6a6ffffdaa0badb23b8baf90b6168dd16b3a'
New sha1sum is : '379b74ac9b2d2b09ff6ad7fa876c79f914a755e1'
```

Check running processes

A malicious process can prevent itself from being seen in a system's list of processes (trojan version of *ps* command). Rootcheck inspects all process IDs (PID) looking for discrepancies with different system calls (*getsid*, *getpgid*).

Example:

Diamorphine is a kernel-mode rootkit able to hide itself from *ps* and also able to hide other processes. If we install this package and hide a process, we will get an alert like this:

```
** Alert 1460225922.841535: mail - ossec,rootcheck
2017 Feb 15 10:00:42 (localhost) 192.168.1.240->rootcheck
Rule: 510 (level 7) -> 'Host-based anomaly detection event (rootcheck).'
Process '495' hidden from /proc. Possible kernel level rootkit.
```

Check hidden ports

Malware can use use hidden ports to communicate with the attacker. Rootcheck checks every port in the system using *bind()* and if it is not possible to bind to a port and it is not in the *netstat* output, a malware could be using that port.

Check unusual files and permissions

Scan the entire file system looking for unusual files and permissions. Files owned by root with write permissions for other user accounts, uid files, hidden directories, and files are all inspected.

Check hidden files using system calls

Scan the entire system comparing the differences between the *stat size* and the file size when using the *fopen + read* calls. The number of nodes in each directory is also compared with the output of *opendir + readdir*. If any results do not match, you might have malware installed.

Alert Example:

```
** Alert 1460225922.51190: mail - ossec,rootcheck
2017 Feb 15 10:30:42 (localhost) 192.168.1.240->rootcheck
Rule: 510 (level 7) -> 'Host-based anomaly detection event (rootcheck).'
Files hidden inside directory '/etc'. Link count does not match number of files (128,129)
```

Scan the /dev directory

The */dev* directory should only contain device-specific files. Any additional file should be inspected because malware uses this partition to hide files.

Example:

If you create a hidden file on */dev*, Wazuh should alert because there is a hidden file in a directory that should only contain device-specific files. This is the alert generated in that case:

```
** Alert 1487182293.37491: - ossec,rootcheck,
2017 Feb 15 10:11:33 localhost->rootcheck
Rule: 510 (level 7) -> 'Host-based anomaly detection event (rootcheck).'
File '/dev/.hiddenfile' present on /dev. Possible hidden file.
title: File present on /dev.
file: /dev/.hiddenfile
```

Scan network interfaces

Scan for any network interfaces on the system with *promiscuous mode* enabled. If the interface is in *promiscuous mode*, the output of the *ifconfig* command will show that. If not, we might have a malware installed.

Rootkit checks

Rootcheck performs several checks using its own database of rootkit signatures: *rootkit_files.txt*, *rootkit_trojans.txt* and *win_malware_rcl.txt*. Unfortunately, the signatures are out of date.

Configuration

1. [Basic example](#)
2. [Ignoring false positives](#)

Basic example

To configure the options for syscheck and rootcheck go to [ossec.conf](#). If you want more information about the exact configuration options go to [Syscheck section](#) and [Rootcheck section](#). Also see the following sections: [frequency](#), [rootkit_files](#), [rootkit_trojans](#)

A basic example to configure the database for rootkits (files and trojans):

```
<rootcheck>
  <rootkit_files>/var/ossec/etc/shared/rootkit_files.txt</rootkit_files>
  <rootkit_trojans>/var/ossec/etc/shared/rootkit_trojans.txt</rootkit_trojans>
</rootcheck>
```

Ignoring false positives

```
<rule id="100100" level="0">
  <if_group>rootcheck</if_group>
  <match>/dev/.blkid.tab</match>
  <description>Ignore false positive for /dev/.blkid.tab</description>
</rule>
```


FAQ

1. [How often does rootcheck run?](#)
2. [How does rootcheck know the rootkit files to look for?](#)
3. [Does rootcheck inspect running processes?](#)
4. [What about hidden files?](#)

How often does rootcheck run?

The rootcheck scan frequency is configurable with [frequency](#). By default it runs every 2 hours.

How does rootcheck know the rootkit files to look for?

The rootcheck engine has databases of rootkit signatures: *rootkit_files.txt*, *rootkit_trojans.txt* and *win_malware_rcl.txt*. Unfortunately, the signatures are out of date.

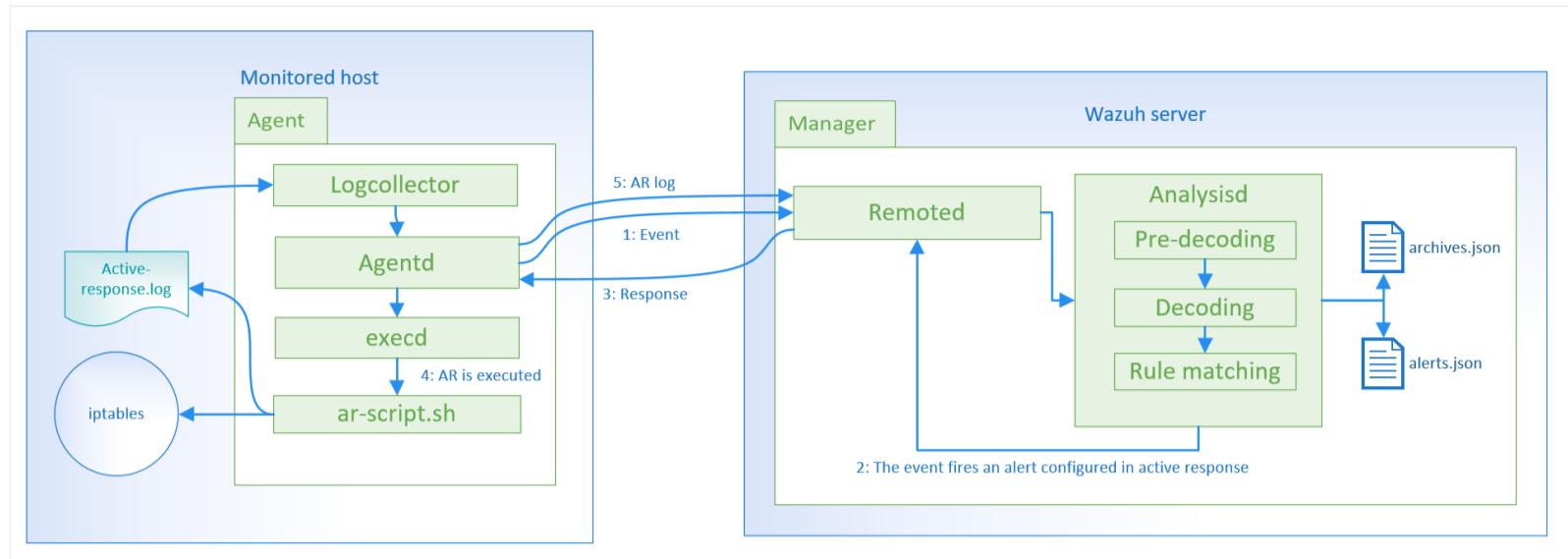
Does rootcheck inspect running processes?

Yes, rootcheck inspects all running processes looking for discrepancies with different system calls.

What about hidden files?

The rootcheck engine scans the entire system comparing the differences between the *stat size* and the files size when using the *fopen + read* calls. If any results do not match, you might have a malware installed.

How it works



When is an active response triggered?

An active response can be triggered by a specific alert, alert level or rule group as configured in the ossec.conf file, where a script is configured to execute with a certain rule/group. **Active responses** are either stateful or stateless responses. Stateful responses will undo the action after a specified period of time while stateless responses are one-time actions.

Where are active response actions executed?

Active response specifies where their associated command will be executed: on the agent that triggered the alert, on the manager, on another specified agent, or on all agents plus the manager.

Active response configuration

An active response is configured in [ossec.conf](#) as follows:

1. Create a command

In order to configure an active response, a **command** must be defined that will initiate a certain script in response to a trigger.

To configure the active response, define the name of a **command** using the pattern below and then reference the script to be initiated. Next, define what data element(s) will be passed to the script.

Custom scripts that have the ability to receive parameters from the command line may also be used for an active response.

Example:

```
<command>
  <name>host-deny</name>
  <executable>host-deny.sh</executable>
  <expect>srcip</expect>
  <timeout_allowed>yes</timeout_allowed>
</command>
```

In this example, the command is called **host-deny** which initiates the **host-deny.sh** script. The data element is defined as **srcip** and this command is configured with a specified period of time, making it a stateful response.

! Note

More information about options for creating a [command](#)

2. Define the active response

The active response configuration defines when and where a command is going to be executed. A command will be triggered when a specific rule with a specific id, severity level or source matches the active response criteria. This configuration will further define where the action of the command will be initiated, meaning in which environment (Agent, Manager, Local, or everywhere).

Example:

```
<active-response>
  <command>host-deny</command>
  <location>local</location>
  <level>7</level>
  <timeout>600</timeout>
</active-response>
```

In this example, the active response is configured to execute the command that was defined in the previous step. The where of the action is defined as the local host and the when is defined as any time the rule has a level higher than 6. The timeout that was allowed in the command configuration is also defined in the above example.

Note

More information about all the options you can define for the [Active response](#)

You can view the active response log at `/var/ossec/logs/active-response.log`.

Default Active response scripts

Wazuh is preconfigured with the following scripts:

Script name	Description
disable-account.sh	disables an account by setting <code>passwd -l</code>
firewall-drop.sh	adds an IP to the iptables deny list
firewalld-drop.sh	adds an IP to firewalld drop list
host-deny.sh	adds an IP to the /etc/hosts.deny file
ip-customblock.sh	Custom OSSEC block, easily modifiable for custom response
ipfw_mac.sh	Firewall-drop response script created for the Mac OS
ipfw.sh	Firewall-drop response script created for ipfw
npf.sh	Firewall-drop response script created for npf
ossec-slack.sh	in order to post modifications
ossec-tweeter.sh	in order to post modifications
pf.sh	Firewall-drop response script created for pf
restart-ossec.sh	Automatically restarts Wazuh when ossec.conf has been changed
route-null.sh	Adds an IP to null route

Configuration

1. [Basic usage](#)
2. [Windows automatic remediation](#)
3. [Block an IP with PF](#)
4. [Add an IP to the iptables deny list](#)
5. [Active response for a specified period of time](#)
6. [Active response that will not be undone](#)

Basic usage

Active response is configured in `ossec.conf`, within the [Active Response](#) and [Command](#) sections.

In this example, a command with the name “`restart-ossec`” is configured to use the “`restart-ossec.sh`” script with no data element. The **Active response** is configured to initiate the “`restart-ossec`” command on the local host when the rule with ID 10005 fires. This is a *Stateless* response as no timeout parameter is defined.

Command:

```
<command>
  <name>restart-ossec</name>
  <executable>restart-ossec.sh</executable>
  <expect></expect>
</command>
```

Active response:

```
<active-response>
  <command>restart-ossec</command>
  <location>local</location>
  <rules_id>10005</rules_id>
</active-response>
```

Windows automatic remediation

In this example, a command with the name “`win_route-null`” is configured to use the “`route-null.cmd`” script using the data element “`srcip`”. The **Active response** is configured to initiate the “`win_route-null`” command on the local host when the rule has a higher alert level than 7. This is a *Stateful* response with a timeout set at 900 seconds.

Command:

```
<command>
  <name>win_route-null</name>
  <executable>route-null.cmd</executable>
  <expect>srcip</expect>
  <timeout_allowed>yes</timeout_allowed>
</command>
```

Active response:

```
<active-response>
  <command>win_route-null</command>
  <location>local</location>
  <level>8</level>
  <timeout>900</timeout>
</active-response>
```

Block an IP with PF

In this example, a command with the name “*pf-block*” is configured to use the “*pf.sh*” script using the data element “*script*”. The **Active response** is configured to initiate the “*pf-block*” command on agent “001” when a rule in either the “*authenticaiton_failed*” or “*authentication_failures*” rule group fires. This is a *Stateless* response as no timeout parameter is defined.

Command:

```
<command>
  <name>pf-block</name>
  <executable>pf.sh</executable>
  <expect>srcip</expect>
</command>
```

Active response:

```
<active-response>
  <command>pf-block</command>
  <location>defined-agent</location>
  <agent_id>001</agent_id>
  <rules_group>authentication_failed,authentication_failures</rules_group>
</active-response>
```

Add an IP to the iptables deny list

In this example, a command with the name “*firewall-drop*” is configured to use the “*firewall-drop.sh*” script using the data element “*script*”. The **Active response** is configured to initiate the “*firewall-block*” command on all systems when a rule in either the “*authenticaiton_failed*” or “*authentication_failures*” rule group fires. This is a *Stateful* response with a timeout of 700 seconds. The repeated offenders parameter increases the timeout period for each subsequent offence by a specific IP address.

Note: This parameter is specified in minutes rather than seconds.

Command:

```
<command>
  <name>firewall-drop</name>
  <executable>firewall-drop.sh</executable>
  <expect>srcip</expect>
</command>
```

Active response:

```
<active-response>
  <command>firewall-block</command>
  <location>all</location>
  <rules_group>authentication_failed,authentication_failures</rules_group>
  <timeout>700</timeout>
  <repeated_offenders>30,60,120</repeated_offenders>
</active-response>
```

Active response for a specified period of time

The action of a stateful response continues for a specified period of time.

In this example, a command with the name “*host-deny*” is configured to use the “*host-deny.sh*” script using the data element “*script*”. The **Active response** is configured to initiate the “*host-deny*” command on the local host when a rule with a higher alert level than 6 is fired.

Command:

```
<command>
  <name>host-deny</name>
  <executable>host-deny.sh</executable>
  <expect>srcip</expect>
  <timeout_allowed>yes</timeout_allowed>
</command>
```

Active response:

```
<active-response>
  <command>host-deny</command>
  <location>local</location>
  <level>7</level>
  <timeout>600</timeout>
</active-response>
```

More info: [command](#)

Active response that will not be undone

The action of a stateless command is a one-time action that will not be undone.

In this example, a command with the name “*mail-test*” is configured to use the “*mail-test.sh*” script with no data element. The **Active response** is configured to initiate the “*mail-test*” command on the server when the rule with ID 1002 fires.

Command:

```
<command>
  <name>mail-test</name>
  <executable>mail-test.sh</executable>
  <timeout_allowed>no</timeout_allowed>
  <expect />
</command>
```

Active response:

```
<active-response>
  <command>mail-test</command>
  <location>server</location>
  <rules_id>1002</rules_id>
</active-response>
```

FAQ

1. [Can I use a custom script for active response?](#)
2. [Can I configure active response for only one host?](#)
3. [Can active response remove the action after a time?](#)

Can I use a custom script for active response?

Yes. You can create your own script and configure a command and active response to refer to it.

Can I configure active response for only one host?

Yes, using the location option. More info: [Active Response options](#)

Can active response remove the action after a time?

Yes, using the *timeout_allowed* option on the command and the *timeout* option on the active response. More info: [Example](#)

The registration process

Every Wazuh Agent send data to Wazuh Manager via secure way called OSSEC message protocol this encrypts messages using a pre-shared key. Initially when you successfully install a new Wazuh Agent this cannot communicate with Wazuh Manager for lack of the pre-shared key.

The registration process consists of a mechanism to create a trust relationship between the Manager and a Agent, this process could be done in a Manager itself or with a registration service, this service it listen on the Manager, a Agent could request a pre-shared key using some credentials and it will reply with the pre-shared key and store the new Agent in local database.

Another aproach is using the RESTful API, this is just a wrapper for local registration on Wazuh Manager.

Agent keys

The manager uses the file `/var/ossec/etc/client.keys` to store the registration record of each agent, which includes ID, name, IP, and key. Example:

```
001 Server1 any e20e0394dca71bacdea57d4ca25d203f836eca12eeaca1ec150c2e5f4309a653a
002 ServerProd 192.246.247.247 b0c5548beda537daddb4da698424d0856c3d4e760eaced803d58c07ad1a95f4c
003 DBServer 192.168.0.1/24 8ec4843da9e61647d1ec3facab542acc26bd0e08ffc010086bb3a6fc22f6f65b
```

The agents also have the file `/var/ossec/etc/client.keys` containing only their own registration record. Example for `Server1` agent:

```
001 Server1 any e20e0394dca71bacdea57d4ca25d203f836eca12eeaca1ec150c2e5f4309a653a
```

Basic data for registering an agent

In order to register an agent, it is necessary to provide the name and the IP of the agent.

There are several ways to set the agent IP:

- **Any IP:** Allow the agent to connect from any IP address. Example: `Server1` has `any` IP.
- **Fixed IP:** Allow the agent to connect only from the specified IP. Example: `ServerProd` has the IP `192.246.247.247`.
- **Range IP:** Allow the agent to connect from the specified range of IPs. Example: `DBServer` has the IP range `192.168.0.1/24`.

Some registration methods automatically detect the IP of the agent during the registration process.

Registration methods

Here are three ways to register an agent:

Type	Method	Description
Manually	Using the command line	Register an agent manually using <code>manage_agents</code> binary.
Automatically	Using the registration service	Register an agent automatically using <code>ossec-authd</code> binary.
	Using the RESTful API	Register an agent by scripting (bash, python, powershell) and the API.

Using the registration service

It's possible to register agents automatically with authd. Choose the method that best meets your needs:

Method	Description	
Simple method	The easiest method. There is no authentication or host verification.	
Use a password to authorize agents	Allows agents to authenticate via a shared password. This method is easy but does not perform host validation.	
Verify manager via SSL	The manager's certificate is signed by a CA that agents use to validate the server. This may include host checking.	
Verify agents via SSL	Host validation	The same as above, but the manager verifies the agent's certificate and address. There should be one certificate per agent.
	No host validation	The manager validates the agent by CA but not the host address. This method allows the use of a shared agent certificate.

Simple method

Get an SSL certificate

The first step is to get an SSL key and certificate. This is required in order to make authd work.

1. If you have a valid SSL certificate with its key, copy them into the *etc* folder:

```
# (Manager)
cp <ssl_cert> /var/ossec/etc/sslmanager.cert
cp <ssl_key> /var/ossec/etc/sslmanager.key
```

2. Otherwise, you can create a self-signed certificate:

```
# (Manager)
openssl req -x509 -batch -nodes -days 365 -newkey rsa:2048 -keyout /var/ossec/etc/sslmanager.key -out
/var/ossec/etc/sslmanager.cert
```

Register the agent

1. Start the authd server:

```
# (Manager)
/var/ossec/bin/ossec-authd
```

2. Run the auth client on the agent. You must enter the authd server's IP address, like this:

```
/var/ossec/bin/agent-auth -m 192.168.1.2
```

Some hints

By default, authd adds agents with a dynamic IP (like using "any" on [manage_agents](#)). If you want to add agents with static IP addresses, use [-i](#) at server-side:

```
# (Manager)
/var/ossec/bin/ossec-authd -i
```

On the other hand, **duplicate IPs are not allowed**, so an agent won't be added if there is already another agent registered with the same IP. By using the `-f` option, authd can be told to **force a registration** if it finds an older agent with the same IP - the older agent's registration will be deleted:

```
# (Manager)
/var/ossec/bin/ossec-authd -i -f 0
```

The `0` means the minimum time, in seconds, since the last connection of the old agent (the one to be deleted). In this case, `0` means to delete the old agent's registration regardless of how recently it has checked in.

Secure methods

Launching the authd daemon with default options would allow any agent to register itself, and then connect to a manager. The following options provide some mechanisms to authorize connections:

Method		Description
Use a password to authorize agents		Allows agents to authenticate via a shared password. This method is easy but does not perform host validation.
Verify manager via SSL		The manager's certificate is signed by a CA that agents use to validate the server. It may include host checking.
Verify agents via SSL	Host validation	The same as above, but the manager verifies the agent's certificate and address. There should be one certificate per agent.
	No host validation	The manager validates the agent by CA but not the host address. This method allows the use of a shared agent certificate.

Note

These methods can be combined.

Use a password to authorize agents

Note

Reference [ossec-authd](#)

The manager can be protected from unauthorized registrations by using a password. We can choose one ourselves or let authd generate a random password.

1. To specify a password manually, just write it to the file `etc/authd.pass`. For example, if the key were "TopSecret":

```
# (Manager)
echo "TopSecret" > /var/ossec/etc/authd.pass
/var/ossec/bin/ossec-authd -P

Accepting connections. Using password specified on file: /var/ossec/etc/authd.pass
```

2. If you don't specify a password, then authd will create a password itself and tell you what it is:

```
# (Manager)
/var/ossec/bin/ossec-authd -P

Accepting connections. Random password chosen for agent authentication: abcd1234
```

On the agent side, the key can be put in a file of the same name or specified as a command-line argument.

1. Using the file `etc/authd.pass`:

```
# (Agent)
echo "abcd1234" > /var/ossec/etc/authd.pass
/var/ossec/bin/agent-auth -m 192.168.1.2
```

2. Entering the password at the command line:

```
# (Agent)
/var/ossec/bin/agent-auth -m 192.168.1.2 -P "abcd1234"
```

Use SSL to verify hosts

Create a Certificate of Authority

First we are going to create a certificate of authority (CA) that we will use to sign the certificates for the manager and agents. Hosts will receive a copy of this certificate in order to verify the remote certificate:

```
openssl req -x509 -new -nodes -newkey rsa:2048 -keyout rootCA.key -out rootCA.pem -batch
```

⚠ Warning

The file `rootCA.key` that we have just created is the **private key** of the certificate of authority. It is needed to sign other certificates and it is critical to keep it secure. Note that we will never copy this file to other hosts.

Verify manager via SSL

1. Issue and sign a certificate for the authd server, entering the hostname or the IP address that agents will use to connect to the server. For example, if the server's IP is 192.168.1.2:

```
openssl req -new -nodes -newkey rsa:2048 -keyout sslmanager.key -out sslmanager.csr -subj
'/C=US/CN=192.168.1.2'
openssl x509 -req -days 365 -in sslmanager.csr -CA rootCA.pem -CAkey rootCA.key -out sslmanager.cert -
CAcreateserial
```

2. Copy the newly created certificate and the key to the manager's `etc` folder and start `ossec-authd`:

```
# (Manager)
cp sslmanager.cert sslmanager.key /var/ossec/etc
ossec-authd
```

3. Copy the CA (but not the key) to the agent's `etc` folder and run `agent-auth`:

```
# (Agent)
cp rootCA.pem /var/ossec/etc
agent-auth -m 192.168.1.2 -v /var/ossec/etc/rootCA.pem
```

Verify agents via SSL

Verify agents via SSL (no host validation)

In this example, we are going to create a certificate for agents without specifying their hostname, so that the same certificate can be used by many agents. This verifies that agents have a certificate signed by our CA, no matter where they are connecting from.

1. Issue and sign a certificate for the agent. Note that we will not enter the *common name* field:

```
openssl req -new -nodes -newkey rsa:2048 -keyout sslagent.key -out sslagent.csr -batch  
openssl x509 -req -days 365 -in sslagent.csr -CA rootCA.pem -CAkey rootCA.key -out sslagent.cert -  
CAcreateserial
```

2. Copy the CA (but not the key) to the manager's `etc` folder (if not already there) and start `ossec-authd`:

```
# (Manager)  
cp rootCA.pem /var/ossec/etc  
ossec-authd -v /var/ossec/etc/rootCA.pem
```

3. Copy the newly created certificate and key to the agent's `etc` folder and run `agent-auth`. For example, if the server's IP is 192.168.1.2:

```
# (Agent)  
cp sslagent.cert sslagent.key /var/ossec/etc  
agent-auth -m 192.168.1.2 -x /var/ossec/etc/sslagent.cert -k /var/ossec/etc/sslagent.key
```

Verify agents via SSL (host validation)

This is an alternative method to the last section. In this case, we will bind the agent's certificate to the agent IP address as seen by the manager.

1. Issue and sign a certificate for the agent. Then enter its hostname or IP address into the *common name* field. For example, if the agent's IP is 192.168.1.3:

```
openssl req -new -nodes -newkey rsa:2048 -keyout sslagent.key -out sslagent.csr -subj  
'/C=US/CN=192.168.1.3'  
openssl x509 -req -days 365 -in sslagent.csr -CA rootCA.pem -CAkey rootCA.key -out sslagent.cert -  
CAcreateserial
```

2. Copy the CA (but not the key) to the manager's `etc` folder (if not already there) and start `ossec-authd`. Note that we use the `-s` option in order to enable agent host verification:

```
# (Manager)  
cp rootCA.pem /var/ossec/etc  
ossec-authd -v /var/ossec/etc/rootCA.pem -s
```

3. Copy the newly created certificate and key to the agent's `etc` folder and run `agent-auth`. For example, if the server's IP is 192.168.1.2:

```
# (Agent)  
cp sslagent.cert sslagent.key /var/ossec/etc  
agent-auth -m 192.168.1.2 -x /var/ossec/etc/sslagent.cert -k /var/ossec/etc/sslagent.key
```

Forcing insertion

If you try to add an agent with an IP already listed in an existing registration, `ossec-authd` will generate an error. You can use the argument `-f` to force the insertion.

Example

We previously installed and registered the Wazuh agent on *Server1* with IP 10.0.0.10 and ID 005. For some reason, we then had to completely re-install *Server1* and thus we now need to install and reregister the Wazuh agent on *Server1*. In this case, we can use the "`-f 0`" parameter which results in the previous agent (005) being removed (with a backup) and a new agent being successfully registered. The new agent will have a new ID.

SMTP server with authentication

In case that your SMTP server has authentication (like Gmail), we need to configure a server relay because Wazuh does not support it by default. For this purpose we will use [Postfix](#). The following guide describes the minimal configuration to perform in Postfix to allow Wazuh sends emails to a SMTP with authentication:

1. Install the needed packages:

Ubuntu

```
apt-get install postfix mailutils libsasl2-2 ca-certificates libsasl2-modules
```

CentOS

```
yum update && yum install postfix mailx cyrus-sasl cyrus-sasl-plain
```

2. Set Postfix config file [/etc/postfix/main.cf](#). Add this lines to the end of the file:

Ubuntu

```
relayhost = [smtp.gmail.com]:587
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_sasl_security_options = noanonymous
smtp_tls_CAfile = /etc/ssl/certs/thawte_Primary_Root_CA.pem
smtp_use_tls = yes
```

CentOS

```
relayhost = [smtp.gmail.com]:587
smtp_sasl_auth_enable = yes
smtp_sasl_password_maps = hash:/etc/postfix/sasl_passwd
smtp_sasl_security_options = noanonymous
smtp_tls_CAfile = /etc/ssl/certs/ca-bundle.crt
smtp_use_tls = yes
```

3. Configure email address and password:

```
echo [smtp.gmail.com]:587 USERNAME@gmail.com:PASSWORD > /etc/postfix/sasl_passwd
postmap /etc/postfix/sasl_passwd
chmod 400 /etc/postfix/sasl_passwd
```

4. Secure DB password

```
chown root:root /etc/postfix/sasl_passwd /etc/postfix/sasl_passwd.db
chmod 0600 /etc/postfix/sasl_passwd /etc/postfix/sasl_passwd.db
```

5. Reload Postfix

```
systemctl reload postfix
```

6. Test you configuration with:

```
echo "Test mail from postfix" | mail -s "Test Postfix" you@example.com
```

You should receive an email on you@example.com

7. Configure Wazuh in the </var/ossec/etc/ossec.conf>:

```
<global>
  <email_notification>yes</email_notification>
  <smtp_server>localhost</smtp_server>
  <email_from>USERNAME@gmail.com</email_from>
  <email_to>you@example.com</email_to>
</global>
```

ossec-agentd

The ossec-agentd program is the client-side daemon that communicates with the server. It runs as `ossec` and is chrooted to `/var/ossec`.

-c <config>	Run using <config>, as the configuration file.	
	Default value	/var/ossec/etc/ossec.conf
-D <dir>	Chroot to <dir>	
	Default value	/var/ossec
-d	Run in debug mode. This option may be repeated to increase the verbosity of the debug messages.	
-f	Run in the foreground.	
-g <group>	Run as a group.	
-h	Display the help message.	
-t	Test configuration.	
-u <user>	Run as a user.	
	Default value	ossecm
-V	Display the version and license information	

ossec-agentlessd

The ossec-agentlessd program allows integrity checks to be run on systems without an agent installed.

-c <config>	Read the configuration from file <config>
-D <dir>	Chroot to <dir>
-d	Run in debug mode. This option may be repeated to increase the verbosity of the debug messages.
-f	Run in the foreground.
-g <group>	Run as a group.
-h	Display the help message.
-t	Test configuration.
-u <user>	Run as a user.
-V	Display the version and license information

ossec-analysisd

The ossec-analysisd program receives the log messages and compares them to the rules. It then creates an alert when a log message matches an applicable rule.

-c <config>	Configuration file ossec-analysisd should use.
-D <dir>	Chroot to <dir>
-d	Run in debug mode. This option may be repeated to increase the verbosity of the debug messages.
-f	Run in the foreground.
-g <group>	Run as a group.
-h	Display the help message.
-t	Test configuration.
-u	Run as a user.
-v	Display the version and license information

ossec-authd

The ossec-authd program will automatically add an agent to a Wazuh manager and provide the key to the agent. The [agent-auth](#) application is the client application used with `ossec-authd`. `ossec-authd` creates an agent with an ip address of "any" instead of using its actual IP.

Warning

By default there is no authentication or authorization involved in this transaction, so it is recommended that this daemon only be run when a new agent is being added.

-V	Version and license message.
-h	This help message.
-d	Debug mode. Use this parameter multiple times to increase the debug level.
-t	Test configuration.
-f	Run in foreground.
-i	Use client's source IP address instead of any.
-F <time>	Remove old agent with same name or IP if its keepalive has more than <time> seconds.
-F no	Disable force insertion.
-r	Do not keep removed agents (delete).
-g <group>	Group to run as. Default ossec
-D <dir>	Directory to chroot into. Default /var/ossec
-p <port>	Manager port. Default 1515
-P	Enable shared password authentication, at /var/ossec/etc/authd.pass or random.
-v <path>	Full path to CA certificate used to verify clients.
-s	Used with -v, enable source host verification.
-x <path>	Full path to server certificate. Default /var/ossec/etc/sslmanager.cert.
-k <path>	Full path to server key. Default /var/ossec/etc/sslmanager.key.
-a	Auto negotiate the most secure common SSL/TLS method with the client. Default TLS v1.2 only (if supported by the server).

ossec-csyslogd

The ossec-csyslogd program forwards alerts via syslog.

-c <config>	Run using <config>, as the configuration file.	
	Default value	/var/ossec/etc/ossec.conf
-D <dir>	Chroot to <dir>	
	Default value	/var/ossec
-d	Run in debug mode. This option may be repeated to increase the verbosity of the debug messages.	
-f	Run in the foreground.	
-g <group>	Run as a group.	
-h	Display the help message.	
-t	Test configuration.	
-u <user>	Run as a user.	
	Default value	ossecm
-V	Display the version and license information	

ossec-dbd

The `ossec-dbd` program inserts the alert logs into a database. These alerts can be inserted into either postgresql or mysql.

-c <config>	Run using <config>, as the configuration file.	
	Default value	/var/ossec/etc/ossec.conf
-D <dir>	Chroot to <dir>	
	Default value	/var/ossec
-d	Run in debug mode. This option may be repeated to increase the verbosity of the debug messages.	
-f	Run in the foreground.	
-g <group>	Run as a group.	
-h	Display the help message.	
-t	Test configuration.	
-u <user>	Run as a user.	
	Default value	ossecm
-v	Display the version and license information	

ossec-execd

The ossec-execd program runs active responses by initiating the configured scripts.

-c <config>	Run using <config>, as the configuration file.	
	Default value	/var/ossec/etc/ossec.conf
-d	Run in debug mode. This option may be repeated to increase the verbosity of the debug messages.	
-f	Run in the foreground.	
-g <group>	Run as a group.	
-h	Display the help message.	
-t	Test configuration.	
-V	Display the version and license information	

ossec-logcollector

The ossec-logcollector program monitors configured files and commands for new log messages.

-c <config>	Run using <config>, as the configuration file.	
	Default value	/var/ossec/etc/ossec.conf
-d	Run in debug mode. This option may be repeated to increase the verbosity of the debug messages.	
-f	Run in the foreground.	
-g <group>	Run as a group.	
-h	Display the help message.	
-t	Test configuration.	
-V	Display the version and license information	

ossec-maild

The ossec-maild program sends alerts via email. It is started by [ossec-control](#).

-c <config>	Run using <config>, as the configuration file.	
	Default value	/var/ossec/etc/ossec.conf
-D <dir>	Chroot to <dir>	
	Default value	/var/ossec
-d	Run in debug mode. This option may be repeated to increase the verbosity of the debug messages.	
-f	Run in the foreground.	
-g <group>	Run as a group.	
-h	Display the help message.	
-t	Test configuration.	
-u <user>	Run as a user.	
	Default value	ossecm
-V	Display the version and license information	

ossec-monitord

The ossec-monitord program monitors agent connectivity and compresses daily log files.

-c <config>	Run using <config>, as the configuration file.	
	Default value	/var/ossec/etc/ossec.conf
-D <dir>	Chroot to <dir>	
	Default value	/var/ossec
-d	Run in debug mode. This option may be repeated to increase the verbosity of the debug messages.	
-f	Run in the foreground.	
-g <group>	Run as a group.	
-h	Display the help message.	
-t	Test configuration.	
-u <user>	Run as a user.	
	Default value	ossecm
-V	Display the version and license information	

ossec-remoted

The ossec-remoted program is the server side daemon that communicates with the agents. It runs as ossecr and is chrooted to `/var/ossec` by default.

-c <config>	Run using <config>, as the configuration file.	
	Default value	/var/ossec/etc/ossec.conf
-D <dir>	Chroot to <dir>	
	Default value	/var/ossec
-d	Run in debug mode. This option may be repeated to increase the verbosity of the debug messages.	
-f	Run in the foreground.	
-g <group>	Run as a group.	
-h	Display the help message.	
-t	Test configuration.	
-u <user>	Run as a user.	
	Default value	ossecm
-V	Display the version and license information	

ossec-reportd

`ossec-reportd` is a program to create reports from Wazuh alerts. It accepts alerts on `stdin`, and outputs a report on `stderr`.

! Note

Since `ossec-reportd` outputs to `stderr`, some utilities like `less` will not work if you do not redirect the output. To do this, end the `ossec-reportd` with `2>81` to redirect `stderr` to `stdout`. Following this redirect, `more` or `less` can be used with ease.

-D <dir>	Chroot to <dir> .	
-d	Run in debug mode. This option may be repeated to increase the verbosity of the debug messages.	
	Filter the results.	
-f <filter> <value>	Allowed values	group rule level location user srcip filename
-h	Display the help message.	
-n <string>	Create a description for the report.	
-r <filter> <value>	Show related entries.	
-s	Show the alerts related to the summary.	
-V	Display the version and license information	

ossec-syscheckd

The ossec-syscheckd program checks configured files for changes to the checksums, permissions and ownership. It is run using ossec-control.

-c <config>	Run using <config>, as the configuration file.	
	Default value	/var/ossec/etc/ossec.conf
-d	Run in debug mode. This option may be repeated to increase the verbosity of the debug messages.	
-f	Run in the foreground.	
-h	Display the help message.	
-t	Test configuration.	
-V	Display the version and license information	

wazuh-modulesd

The wazuh-modulesd program manages the Wazuh modules described below.

Elastic Stack integration

Elastic Stack is a combination of popular open source projects for log management, including Elasticsearch, Logstash, Kibana, and others.

Wazuh Ruleset

Wazuh rules are based on of the OSSEC ruleset. This ruleset has been revised and expanded with enhancements, corrections and additions to deepen Wazuh functionality and detection capabilities.

RESTful API

The RESTful API controls the Wazuh manager using REST requests. This allows the Wazuh manager to be interacted with from a web browser, command line tool like cURL, or any script or program that can make web requests. This API may be used to easily perform everyday actions like adding an agent, restarting the manager/agent(s), or looking up syscheck details.

-d	Increase debug mode.
-f	Run in the foreground.
-h	Display the help message.
-t	Test configuration.

How it works

Note

This guide is based on the official [Audit guide](#).

Audit uses a set of rules to define what is to be captured in the log files. There are three types of Audit rules that can be specified:

- **Control rules** allow the Audit system's behavior and some of its configuration to be modified.
- **File system rules**, also known as file watches, allow the auditing of access to a particular file or a directory.
- **System call rules** allow logging of system calls that specified programs makes.

Audit rules can be specified interactively with the *auditctl* command-line utility, but to make changes persistent, edit */etc/audit/audit.rules*.

Warning

All commands that interact with the Audit service and the Audit log files require root privileges, so you will need to be root or use sudo to execute these commands.

Control rules

Some examples that illustrate how to modify the behaviour of the Audit system:

auditctl -b	Set the maximum amount of existing Audit buffers in the kernel.
auditctl -e	Enable/disable the Audit system or lock its configuration.
auditctl -s	Report the status of the Audit system.
auditctl -l	List all currently loaded Audit rules.
auditctl -D	Delete all currently loaded Audit rules.

File System Rules

To define a file system rule, use the following syntax:

```
-w <path> -p <permissions> -k <key_name>
```

where:

-w <path>	Specify what file or directory to audit with <path>		
	<permissions> are the permissions that are to auditing, including the following:		
-p <permissions>	Values	r	read access to a file or a directory.
		w	write access to a file or a directory.
		x	execute access to a file or a directory.
		a	change in the file's or directory's attribute.
-k <key_name>	<key_name> is an optional string to identify which rule/set of rules generates a particular log line. This argument is required by Wazuh in order to analyze the logs more accurately.		

For example, to define a rule that logs all write access to, and every attribute change of, the */etc/passwd* file, execute the following command::

```
$ auditctl -w /etc/passwd -p wa -k passwd_changes
```

System Call Rules

To define a system call rule, use the following syntax::

```
-a action,<filter> -S system_call -F field=value -k key_name
```

where:

-a <action>, <filter>	Tells the kernel's rule matching engine to append a rule at the end of the rule list. We must specify which rule list to append it to and what action to take when it triggers.		
	<action>	always	read access to a file or a directory.
		never	write access to a file or a directory.
	The <filter> value specifies which kernel rule-matching filter is applied to the event		
	<filter>	task	Only audit events fork or clone syscalls. This is rarely used in practice.
		exit	All syscall and file system audit requests are evaluated.
		user	This is used to remove some events that originate in user space. By default, any event originating in user space is allowed.
		exclude	This is used to exclude certain events from being logged. <i>msgtype</i> is used to tell the kernel which message to filter out. For more granular control over which events to audit: use the user and exit filters instead.
-S <system_call>	This specifies which <i>system_call</i> to audit. Multiple system calls can be specified in a single rule. A list of all system calls can be found with the command <code>ausyscall --dump</code> .		
-F <field=value>	Use <i>field=value</i> to specify additional criteria to narrow down which events to audit, based on: architecture, group ID, process ID, etc... Multiple -F options can be used in a single rule.		
-k <key_name>	<key_name>is an optional string to identify which rule/set of rules generates a particular log line. This argument is required by Wazuh in order to analyze the logs more accurately.		

For example, to define a rule that creates a log entry every time a file is deleted or renamed by a system user whose ID is 500 or larger, use the following. Note that the *-F auid!=4294967295* option is used to exclude users whose login UID is not set.

```
$ auditctl -a always,exit -S unlink -S unlinkat -S rename -S renameat -F auid>=500 -F auid!=4294967295 -k delete
```

It is also possible to define a file system rule using the system call rule syntax. The following command creates a rule for system calls that is analogous to the **-w /etc/shadow -p wa** file system rule::

```
$ auditctl -a always,exit -F path=/etc/shadow -F perm=wa
```

Configuration

1. [Basic usage](#)
2. [Monitoring accesses to a directory](#)
3. [Monitoring user actions](#)
4. [Privilege escalation](#)

Basic usage

Manager

Audit generates numerous events and it is hard to distinguish if those events correspond to a *write access*, *read access*, *execute access*, *attribute change*, or *system call rule*, using Wazuh decoders and rules. This is why we use the *key* argument in audit rules to facilitate the processing of events by Wazuh. As previously explained, each audit rule has the option to add a descriptive *key* value to identify what rule generated a particular audit log entry. We will use a CDB list to determine the types of audit rule that has fire. This list will have the following syntax:

```
key_name:value
```

where:

- **Key_name** is the string you used in the argument *-k* of a *file system rule* or a *call system rule*.
- **Value** is one of the following values:

```
write: File system rules with -p w.  
read: File system rules with -p r.  
execute: File system rules with -p x.  
attribute: File system rules with -p a.  
command: System call rules.
```

By default, OSSEC includes a CDB list with the following keys:

```
$ cat /var/ossec/etc/lists/audit-keys  
  
audit-wazuh-w:write  
audit-wazuh-r:read  
audit-wazuh-a:attribute  
audit-wazuh-x:execute  
audit-wazuh-c:command
```

You can add your own key with its value to the list like this:

```
echo "my_key_write_type:write" >> /var/ossec/etc/lists/audit-keys
```

Each time you modify a CDB list, you must compile it:

```
/var/ossec/bin/ossec-makelists
```

Agent

Installing Audit

In order to use the Audit system, you must have the audit package installed on your system. If you do not have this package installed, execute the following command as the root user to install it.

Red Hat, CentOS and Fedora:

```
$ yum install audit
```

Debian and Ubuntu based Linux distributions:

```
$ apt-get install auditd
```

Editing ossec.conf

Wazuh must be aware of the events detected by Audit. So, it needs to be configured to read the audit log file:

```
<localfile>
  <log_format>audit</log_format>
  <location>/var/log/audit/audit.log</location>
</localfile>
```

Restarting OSSEC

Finally, we must restart Wazuh agent in order to apply the changes:

```
$ /var/ossec/bin/ossec-control restart
```

Now everything is ready to process audit events. You only need to create the proper audit rules (via *auditctl* or */etc/audit/audit.rules*). In the next section we will describe some good use cases.

Monitoring accesses to a directory

In this example, we are going to monitor every kind of access under the */home* directory:

```
auditctl -w /home -p w -k audit-wazuh-w
auditctl -w /home -p a -k audit-wazuh-a
auditctl -w /home -p r -k audit-wazuh-r
auditctl -w /home -p x -k audit-wazuh-x
```

Now we start getting alerts on account of the new audit rules:

```

** Alert 1487891035.24299: - audit,audit_configuration,
2017 Feb 23 15:03:55 localhost->/var/log/audit/audit.log
Rule: 80705 (level 3) -> 'Auditd: Configuration changed'
type=CONFIG_CHANGE msg=audit(1487891033.538:2936): auid=1000 ses=346
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 op="add_rule" key="audit-wazuh-w" list=4 res=1
audit.type: CONFIG_CHANGE
audit.id: 2936
audit.key: audit
audit.list: 4
audit.res: 1

** Alert 1487891043.24730: - audit,audit_configuration,
2017 Feb 23 15:04:03 localhost->/var/log/audit/audit.log
Rule: 80705 (level 3) -> 'Auditd: Configuration changed'
type=CONFIG_CHANGE msg=audit(1487891041.427:2937): auid=1000 ses=346
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 op="add_rule" key="audit-wazuh-a" list=4 res=1
audit.type: CONFIG_CHANGE
audit.id: 2937
audit.key: audit
audit.list: 4
audit.res: 1

** Alert 1487891047.25161: - audit,audit_configuration,
2017 Feb 23 15:04:07 localhost->/var/log/audit/audit.log
Rule: 80705 (level 3) -> 'Auditd: Configuration changed'
type=CONFIG_CHANGE msg=audit(1487891045.481:2938): auid=1000 ses=346
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 op="add_rule" key="audit-wazuh-r" list=4 res=1
audit.type: CONFIG_CHANGE
audit.id: 2938
audit.key: audit
audit.list: 4
audit.res: 1

** Alert 1487891049.25592: - audit,audit_configuration,
2017 Feb 23 15:04:09 localhost->/var/log/audit/audit.log
Rule: 80705 (level 3) -> 'Auditd: Configuration changed'
type=CONFIG_CHANGE msg=audit(1487891049.144:2939): auid=1000 ses=346
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 op="add_rule" key="audit-wazuh-x" list=4 res=1
audit.type: CONFIG_CHANGE
audit.id: 2939
audit.key: audit
audit.list: 4
audit.res: 1

```

Note

While it would be possible to define the previous rules as one single rule that specifies `-p warx`, we intentionally separate them out so each rule has its own unique **key** value that is important for analysis.

Let's see what happens when we execute the following commands:

New File

Command:

```
$ touch /home/malware.py
```

Alert:

```
** Alert 1487891161.28457: - audit,audit_watch_write,audit_watch_create,
2017 Feb 23 15:06:01 localhost->/var/log/audit/audit.log
Rule: 80790 (level 3) -> 'Audit: Created: /home/malware.py'
type=SYSCALL msg=audit(1487891161.190:2942): arch=c000003e syscall=2 success=yes exit=3 a0=7ffce677b7b7
a1=941 a2=1b6 a3=7ffce6779690 items=2 ppid=60621 pid=60761 auid=1000 uid=0 gid=0 euid=0 suid=0
fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=346 comm="touch" exe="/usr/bin/touch"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="audit-wazuh-w" type=CWD
msg=audit(1487891161.190:2942): cwd="/" type=PATH msg=audit(1487891161.190:2942): item=0
name="/home/" inode=16777403 dev=fd:00 mode=040755 ouid=0 ogid=0 rdev=00:00
obj=system_u:object_r:home_root_t:s0 objtype=PARENT type=PATH msg=audit(1487891161.190:2942):item=1
name="/home/malware.py" inode=18369115 dev=fd:00 mode=0100644 ouid=0 ogid=0 rdev=00:00
obj=unconfined_u:object_r:home_root_t:s0 objtype=CREATE
audit.type: SYSCALL
audit.id: 2942
audit.syscall: 2
audit.success: yes
audit.exit: 3
audit.ppid: 60621
audit.pid: 60761
audit.auid: 1000
audit.uid: 0
audit.gid: 0
audit.euid: 0
audit.suid: 0
audit.fsuid: 0
audit.egid: 0
audit.sgid: 0
audit.fsgid: 0
audit.tty: pts0
audit.session: 346
audit.command: touch
audit.exe: /usr/bin/touch
audit.key: audit-wazuh-w
audit.cwd: /
audit.directory.name: /home/
audit.directory.inode: 16777403
audit.directory.mode: 040755
audit.file.name: /home/malware.py
audit.file.inode: 18369115
audit.file.mode: 0100644
```

Write Access

Command:

```
$ nano /home/malware.py
```

Alert:

```
** Alert 1487891353.48010: - audit,audit_watch_write,
2017 Feb 23 15:09:13 localhost->/var/log/audit/audit.log
Rule: 80781 (level 3) -> 'Audit: Watch - Write access: /home/malware.py'
type=SYSCALL msg=audit(1487891353.291:2956): arch=c000003e syscall=2 success=yes exit=3 a0=9e2e80
a1=441 a2=1b6 a3=63 items=2 ppid=60621 pid=60819 auid=1000 uid=0 gid=0 euid=0 suid=0 egid=0
sgid=0 fsgid=0 tty=pts0 ses=346 comm="nano" exe="/usr/bin/nano"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="audit-wazuh-w"
type=CWD msg=audit(1487891353.291:2956): cwd="/" type=PATH msg=audit(1487891353.291:2956): item=0
name="/home/" inode=16777403 dev=fd:00 mode=040755 ouid=0 ogid=0 rdev=00:00
obj=system_u:object_r:home_root_t:s0 objtype=PARENT type=PATH msg=audit(1487891353.291:2956): item=1
name="/home/malware.py" inode=18369115 dev=fd:00 mode=0100644 ouid=0 ogid=0 rdev=00:00
obj=unconfined_u:object_r:home_root_t:s0 objtype=NORMAL
audit.type: SYSCALL
audit.id: 2956
audit.syscall: 2
audit.success: yes
audit.exit: 3
audit.ppid: 60621
audit.pid: 60819
audit.auid: 1000
audit.uid: 0
audit.gid: 0
audit.euid: 0
audit.suid: 0
audit.fsid: 0
audit.egid: 0
audit.sgid: 0
audit.fsgid: 0
audit.tty: pts0
audit.session: 346
audit.command: nano
audit.exe: /usr/bin/nano
audit.key: audit-wazuh-w
audit.cwd: /
audit.directory.name: /home/
audit.directory.inode: 16777403
audit.directory.mode: 040755
audit.file.name: /home/malware.py
audit.file.inode: 18369115
audit.file.mode: 0100644
```

Change Permissions

Command:

```
$ chmod u+x /home/malware.py
```

Alert:

```
** Alert 1487891409.49498: - audit,audit_watch_attribute,
2017 Feb 23 15:10:09 localhost->/var/log/audit/audit.log
Rule: 80787 (level 3) -> 'Audit: Watch - Change attribute: /home/malware.py'
type=SYSCALL msg=audit(1487891408.563:2957): arch=c000003e syscall=268 success=yes exit=0
a0=fffffffffffff9c
a1=22f50f0 a2=1e4 a3=7fffe879a7d0 items=1 ppid=60621 pid=60820 auid=1000 uid=0 gid=0 euid=0
suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty pts0 ses=346 comm="chmod" exe="/usr/bin/chmod"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="audit-wazuh-a" type=CWD
msg=audit(1487891408.563:2957): cwd="/" type=PATH msg=audit(1487891408.563:2957): item=0
name="/home/malware.py" inode=18369115 dev=fd:00 mode=0100644 uid=0 ogid=0 rdev=00:00
obj=unconfined_u:object_r:home_root_t:s0 objtype=NORMAL
audit.type: SYSCALL
audit.id: 2957
audit.syscall: 268
audit.success: yes
audit.exit: 0
audit.ppid: 60621
audit.pid: 60820
audit.auid: 1000
audit.uid: 0
audit.gid: 0
audit.euid: 0
audit.suid: 0
audit.fsuid: 0
audit.egid: 0
audit.sgid: 0
audit.fsgid: 0
audit.tty: pts0
audit.session: 346
audit.command: chmod
audit.exe: /usr/bin/chmod
audit.key: audit-wazuh-a
audit.cwd: /
audit.file.name: /home/malware.py
audit.file.inode: 18369115
audit.file.mode: 0100644
```

Read access

Command:

```
$ /home/malware.py
```

Alert:

```
** Alert 1487891459.53222: - audit,audit_watch_read,
2017 Feb 23 15:10:59 localhost->/var/log/audit/audit.log
Rule: 80784 (level 3) -> 'Audit: Watch - Read access: /home/malware.py'
type=SYSCALL msg=audit(1487891458.283:2960): arch=c000003e syscall=2 success=yes exit=3 a0=14d1e20
a1=0 a2=fffffffffffff80 a3=7ffdd01083d0 items=1 ppid=60621 pid=60821 auid=1000 uid=0 gid=0 euid=0
suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=346 comm="bash" exe="/usr/bin/bash"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="audit-wazuh-r" type=CWD
msg=audit(1487891458.283:2960): cwd="/" type=PATH msg=audit(1487891458.283:2960): item=0
name="/home/malware.py" inode=18369115 dev=fd:00 mode=0100744 ouid=0 ogid=0 rdev=00:00
obj=unconfined_u:object_r:home_root_t:s0 objtype=NORMAL
audit.type: SYSCALL
audit.id: 2960
audit.syscall: 2
audit.success: yes
audit.exit: 3
audit.ppid: 60621
audit.pid: 60821
audit.auid: 1000
audit.uid: 0
audit.gid: 0
audit.euid: 0
audit.suid: 0
audit.fsuid: 0
audit.egid: 0
audit.sgid: 0
audit.fsgid: 0
audit.tty: pts0
audit.session: 346
audit.command: bash
audit.exe: /usr/bin/bash
audit.key: audit-wazuh-r
audit.cwd: /
audit.file.name: /home/malware.py
audit.file.inode: 18369115
audit.file.mode: 0100744
```

Delete file

Command:

```
$ rm /home/malware.py
```

Alert:

```
** Alert 1487891497.54463: - audit,audit_watch_write,audit_watch_delete,
2017 Feb 23 15:11:37 localhost->/var/log/audit/audit.log
Rule: 80791 (level 3) -> 'Audit: Deleted: /home/malware.py'
type=SYSCALL msg=audit(1487891496.026:2961): arch=c000003e syscall=263 success=yes exit=0
a0=fffffffffffff9c a1=13b00c0 a2=0 a3=7ffe1b582dc0 items=2 ppid=60621 pid=60824 auid=1000
uid=0 gid=0 euid=0 suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=346 comm="rm" exe="/usr/bin/rm"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="audit-wazuh-w"
type=CWD msg=audit(1487891496.026:2961): cwd="/" type=PATH msg=audit(1487891496.026:2961): item=0
name="/home/" inode=16777403 dev=fd:00 mode=040755 ouid=0 ogid=0 rdev=00:00
obj=system_u:object_r:home_root_t:s0 objtype=PARENT type=PATH msg=audit(1487891496.026:2961): item=1
name="/home/malware.py" inode=18369115 dev=fd:00 mode=0100744 ouid=0 ogid=0 rdev=00:00
obj=unconfined_u:object_r:home_root_t:s0 objtype=DELETE
audit.type: SYSCALL
audit.id: 2961
audit.syscall: 263
audit.success: yes
audit.exit: 0
audit.ppid: 60621
audit.pid: 60824
audit.auid: 1000
audit.uid: 0
audit.gid: 0
audit.euid: 0
audit.suid: 0
audit.fsuid: 0
audit.egid: 0
audit.sgid: 0
audit.fsgid: 0
audit.tty: pts0
audit.session: 346
audit.command: rm
audit.exe: /usr/bin/rm
audit.key: audit-wazuh-w
audit.cwd: /
audit.directory.name: /home/
audit.directory.inode: 16777403
audit.directory.mode: 040755
audit.file.name: /home/malware.py
audit.file.inode: 18369115
audit.file.mode: 0100744
```

Monitoring user actions

Here we choose to audit all commands run by a user who has admin privileges. The audit configuration for this is quite simple:

```
$ auditctl -a exit,always -F euid=0 -F arch=b64 -S execve -k audit-wazuh-c
$ auditctl -a exit,always -F euid=0 -F arch=b32 -S execve -k audit-wazuh-c
```

If the root user executes nano, the alert will look like this:

```

** Alert 1487892032.56406: - audit,audit_command,
2017 Feb 23 15:20:32 localhost->/var/log/audit/audit.log
Rule: 80792 (level 3) -> 'Audit: Command: /usr/bin/nano'
type=SYSCALL msg=audit(1487892031.893:2963): arch=c000003e syscall=59 success=yes exit=0 a0=14e4990
a1=14e4a30 a2=14d4ef0 a3=7ffdd01083d0 items=2 ppid=60621 pid=60840 auid=1000 uid=0 gid=0 euid=0
suid=0 fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=346 comm="nano" exe="/usr/bin/nano"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="audit-wazuh-c" type=EXECVE
msg=audit(1487892031.893:2963): argc=1 a0="nano" type=CWD msg=audit(1487892031.893:2963):
 cwd="/" type=PATH msg=audit(1487892031.893:2963): item=0 name="/bin/nano" inode=18372489 dev=fd:00
mode=0100755 ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:bin_t:s0 objtype=NORMAL type=PATH
msg=audit(1487892031.893:2963): item=1 name="/lib64/ld-linux-x86-64.so.2" inode=33595530 dev=fd:00
mode=0100755 ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:ld_so_t:s0 objtype=NORMAL
audit.type: SYSCALL
audit.id: 2963
audit.syscall: 59
audit.success: yes
audit.exit: 0
audit.ppid: 60621
audit.pid: 60840
audit.auid: 1000
audit.uid: 0
audit.gid: 0
audit.euid: 0
audit.suid: 0
audit.fsuid: 0
audit.egid: 0
audit.sgid: 0
audit.fsgid: 0
audit.tty: pts0
audit.session: 346
audit.command: nano
audit.exe: /usr/bin/nano
audit.key: audit-wazuh-c
audit.cwd: /
audit.file.name: /bin/nano
audit.file.inode: 18372489
audit.file.mode: 0100755

```

Privilege escalation

By default, Wazuh is able to detect privilege escalation by analyzing the corresponding log in `/var/log/auth.log`. The below example shows the homer user executing a root action:

```
$ homer@springfield:/ $ sudo ls /var/ossec/etc
```

Wazuh detects the action, extracting the `srcuser`, `dstuser` and `command` among other fields:

```

** Alert 1487892460.79075: - syslog,sudo,pci_dss_10.2.5,pci_dss_10.2.2,
2017 Feb 23 15:27:40 localhost->/var/log/secure
Rule: 5402 (level 3) -> 'Successful sudo to ROOT executed'
User: root
Feb 23 15:27:40 localhost sudo:      rromero : TTY=pts/0 ; PWD=/home/rromero ; USER=root ; COMMAND=/bin/ls
/var/ossec/etc
tty: pts/0
pwd: /home/rromero
command: /bin/ls

```

However, you may find this level of detail inadequate, in which case you can use Audit.

If you have created a rule to monitor root actions, like in the previous use case, every action with `sudo` will be logged, but the `auid` field will inconveniently be 0 (root user) instead of that of the actual user who initiated the escalated action. You generally want to know who originally initiated a command, regardless of if it was escalated or not.

In order to keep the track of the user after sudo, it is necessary to configure *PAM*.

⚠ Warning

Be very careful with PAM configuration, as a bad configuration could make your system inaccessible.

Add the following line to every PAM service that needs it:

```
session required      pam_loginuid.so
```

A common configuration should include: *login*, *common-session*, *cron* and *sshd*:

```
$ grep -R "pam_loginuid.so" /etc/pam.d/  
  
/etc/pam.d/login:session    required      pam_loginuid.so  
/etc/pam.d/common-session:session required      pam_loginuid.so  
/etc/pam.d/cron:session     required      pam_loginuid.so  
/etc/pam.d/sshd:session     required      pam_loginuid.so
```

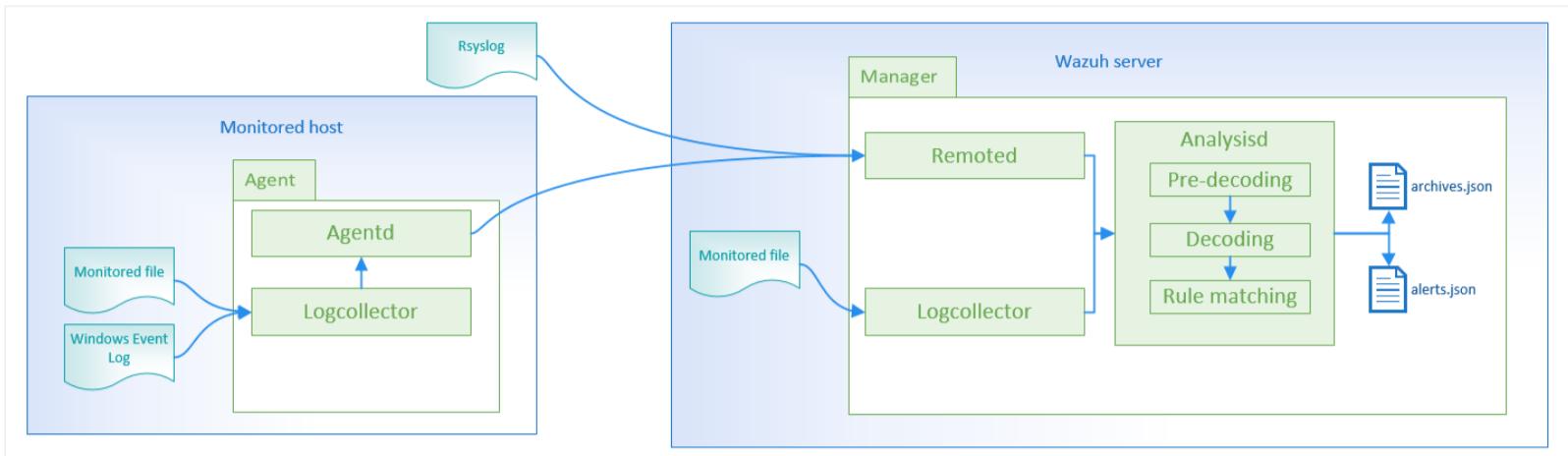
After configuring PAM, if we execute the previous command with the user *homer* we will see that the field *auid* is 1004, the id of the user *homer*.

```
$ homer@springfield:/$ sudo ls /var/ossec/etc
```

```
** Alert 1487892803.121460: - audit,audit_command,
2017 Feb 23 15:33:23 localhost->/var/log/audit/audit.log
Rule: 80792 (level 3) -> 'Audit: Command: /usr/bin/ls'
type=SYSCALL msg=audit(1487892802.652:3054): arch=c000003e syscall=59 success=yes exit=0 a0=7f711f7d4ef8
a1=7f711f7d6358 a2=7f711f7df2e0 a3=7 items=2 ppid=60910 pid=60911 auid=1000 uid=0 gid=0 euid=0 suid=0
fsuid=0 egid=0 sgid=0 fsgid=0 tty=pts0 ses=346 comm="ls" exe="/usr/bin/ls"
subj=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023 key="audit-wazuh-c" type=EXECVE
msg=audit(1487892802.652:3054): argc=2 a0="ls" a1="/var/ossec/etc" type=CWD msg=audit(1487892802.652:3054):
 cwd="/home/rromero" type=PATH msg=audit(1487892802.652:3054): item=0 name="/bin/ls" inode=16912203 dev=fd:00
 mode=0100755 ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:bin_t:s0 objtype=NORMAL type=PATH
msg=audit(1487892802.652:3054): item=1 name="/lib64/ld-linux-x86-64.so.2" inode=33595530 dev=fd:00
mode=0100755 ouid=0 ogid=0 rdev=00:00 obj=system_u:object_r:ld_so_t:s0 objtype=NORMAL
audit.type: SYSCALL
audit.id: 3054
audit.syscall: 59
audit.success: yes
audit.exit: 0
audit.ppid: 60910
audit.pid: 60911
audit.auid: 1000
audit.uid: 0
audit.gid: 0
audit.euid: 0
audit.suid: 0
audit.fsid: 0
audit.egid: 0
audit.sgid: 0
audit.fsgid: 0
audit.tty: pts0
audit.session: 346
audit.command: ls
audit.exe: /usr/bin/ls
audit.key: audit-wazuh-c
audit.cwd: /home/rromero
audit.file.name: /bin/ls
audit.file.inode: 16912203
audit.file.mode: 0100755
```

How it works

The below figure illustrates the event flow:



Log collection

The log source can be:

Log files

The Log analysis engine can be configured to monitor specific files on the servers.

Configuration example:

Linux:

```
<localfile>
  <location>/var/log/example.log</location>
  <log_format>syslog</log_format>
</localfile>
```

Windows:

```
<localfile>
  <location>C:\myapp\example.log</location>
  <log_format>syslog</log_format>
</localfile>
```

Windows event log

Wazuh can monitor classic Windows event logs, as well as the newer Windows event channels:

Configuration example:

Event log:

```
<localfile>
  <location>Security</location>
  <log_format>eventlog</log_format>
</localfile>
```

Event channel:

```
<localfile>
  <location>Microsoft-Windows-PrintService/Operational</location>
  <log_format>eventchannel</log_format>
</localfile>
```

Remote syslog

For other devices like firewalls, you can configure the log analysis component to receive log events through syslog.

Configuration example:

```
<ossec_config>
  <remote>
    <connection>syslog</connection>
    <allowed-ips>192.168.2.0/24</allowed-ips>
  </remote>
<ossec_config>
```

`<connection>syslog</connection>` indicates the manager will accept incoming syslog messages from across the network, and `<allowed-ips>192.168.2.0/24</allowed-ips>` defines the network from which syslog messages will be accepted.

Log Example:

```
2016-03-15T15:22:10.078830+01:00 tron su:pam_unix(su-l:auth):authentication failure;logname=tm uid=500
euid=0 tty=pts/0 ruser=tm rhost= user=root
1265939281.764 1 172.16.167.228 TCP_DENIED /403 734 POST
http://lbcore1.metacafe.com/test/SystemInfoManager.php - NONE/- text/html
[Sun Mar 06 08:52:16 2016] [error] [client 187.172.181.57] Invalid URI in request GET: index.php HTTP/1.0
```

Analysis

Pre-decoding

In this phase, only static information is extracted from well-known fields.

```
Feb 14 12:19:04 localhost sshd[25474]: Accepted password for rromero from 192.168.1.133 port 49765 ssh2
```

Extracted information:

- *hostname*: 'localhost'
- *program_name*: 'sshd'

Decoding

The Decode phase identifies/evaluates the type of a log message and then extracts known fields for that message type. Example of a log and its extracted info:

```
Feb 14 12:19:04 localhost sshd[25474]: Accepted password for rromero from 192.168.1.133 port 49765 ssh2
```

Extracted information:

- *program_name*: sshd
- *dstuser*: rromero
- *srcip*: 192.168.1.133

Rule matching

The next step is to check if any of the rules match.

For the previous example, rule 5715 is matched:

```
<rule id="5715" level="3">
  <if_sid>5700</if_sid>
  <match>^Accepted|authenticated.$</match>
  <description>sshd: authentication success.</description>
  <group>authentication_success,pci_dss_10.2.5,</group>
</rule>
```

! Note

More information about [Wazuh Ruleset](#)

Alert

Once the rule is matched, the manager will create an alert:

```
** Alert 1487103546.21448: - syslog,sshd,authentication_success,pci_dss_10.2.5,
2017 Feb 14 12:19:06 localhost->/var/log/secure
Rule: 5715 (level 3) -> 'sshd: authentication success.'
Src IP: 192.168.1.133
User: rromero
Feb 14 12:19:04 localhost sshd[25474]: Accepted password for rromero from 192.168.1.133 port 49765 ssh2
```

It will be stored in `/var/ossec/logs/alerts/alerts.json` and/or `/var/ossec/logs/alerts/alerts.log`.

By default, it will generate alerts on events that are important or of security relevance. To store all events even if they do not match a rule, you need to enable the `<log_all>` option.

Alerts will be stored at `/var/ossec/logs/alerts.alerts.(json|log)` and events at `/var/ossec/logs/archives/archives.(json|log)`. It uses log rotation and creates an individual directory for each year and month.

Archived logs are not automatically deleted. You choose when to manually or automatically (i.e., cron job) delete logs according to your own legal and regulatory requirements.

Configuration

1. [Basic usage](#)
2. [Monitoring logs using regular expressions for file names](#)
3. [Monitoring date-based logs](#)
4. [Reading logs from Windows Event Log](#)
5. [Reading events from Windows Event Channel](#)
6. [Filtering events from Windows Event Channel with queries](#)
7. [Using environment variables](#)

Basic usage

Log data collection is configured in `ossec.conf`, mainly in the following sections: `localfile`, `remote` and `global`. Also, it is possible to configure it in `agent.conf` to centralize the distribution of these configuration settings to relevant agents.

This is a basic usage example. Provide the name of the file to be monitored and the format:

```
<localfile>
  <location>/var/log/messages</location>
  <log_format>syslog</log_format>
</localfile>
```

Monitoring logs using regular expressions for file names

Wazuh supports posix regular expressions. For example, to analyze every file that ends with a `.log` inside the `/var/log` directory, use the following configuration:

```
<localfile>
  <location>/var/log/*.log</location>
  <log_format>syslog</log_format>
</localfile>
```

Monitoring date-based logs

For log files that change according to the date, you can also specify a **strftime** format to replace the day, month, year, etc. For example, to monitor the log files like `C:\Windows\app\log-08-12-15.log`, where 08 is the year, 12 is the month and 15 the day (and it is rolled over every day), do:

```
<localfile>
  <location>C:\Windows\app\log-%y-%m-%d.log</location>
  <log_format>syslog</log_format>
</localfile>
```

Reading logs from Windows Event Log

To monitor a Windows event log, you need to provide the format as “eventlog” and location is the name of the event log:

```
<localfile>
  <location>Security</location>
  <log_format>eventlog</log_format>
</localfile>
```

Reading events from Windows Event Channel

You can additionally monitor specific Windows event channels. The location is the name of the event channel. This is the only way to monitor the Applications and Services logs. If the file name contains a "%4", replace it with "/":

```
<localfile>
  <location>Microsoft-Windows-PrintService/Operational</location>
  <log_format>eventchannel</log_format>
</localfile>
```

Filtering events from Windows Event Channel with queries

It is possible to filter the events from an event channel:

```
<localfile>
  <location>System</location>
  <log_format>eventchannel</log_format>
  <query>Event/System[EventID=7040]</query>
</localfile>
```

Using environment variables

You can use environment variables like `%WinDir%` in the location pattern. The following is an example of reading logs from an IIS server:

```
<localfile>
  <location>%WinDir%\System32\LogFiles\W3SVC3\ex%y%m%d.log</location>
  <log_format>iis</log_format>
</localfile>
```

FAQ

1. [Are the logs analyzed on each agent?](#)
2. [How often does the manager monitor the logs?](#)
3. [How long are the logs stored on the server?](#)
4. [How does this help me with regulatory compliance?](#)
5. [What is the CPU usage like on the agents?](#)
6. [From where can Wazuh get log messages?](#)
7. [Can I send firewall, vpn, authentication logs to Wazuh?](#)
8. [What information should Wazuh extract from my logs?](#)
9. [Can I ignore events that are not important?](#)

Are the logs analyzed on each agent?

No, the manager gets the logs from all the agents and then analyzes the messages.

How often does the manager monitor the logs?

The manager monitors logs in real time.

How long are the logs stored on the server?

Archived logs are not automatically deleted. You choose when to manually or automatically (i.e., cron job) delete logs according to your own legal and regulatory requirements.

How does this help me with regulatory compliance?

Log analysis is a requirement for : [PCI DSS Compliance](#), HIPAA Compliance, FISMA Compliance and SOX Compliance.

What is the CPU usage like on the agents?

The memory and CPU requirements of the agent are insignificant because it mostly just forwards events to the manager. However, on the manager, CPU and memory consumption can increase quickly depending on the events per second (EPS) that the manager has to analyze.

From where can Wazuh get log messages?

Wazuh can read log messages from text log files, Windows event logs and event channels, and also via remote syslog. Logs are monitored in real time.

Can I send firewall, VPN, authentication logs to Wazuh?

Yes. Wazuh has the capability to receive and process logs from devices that send logs using the syslog protocol. You can create custom decoders and rules for your device-specific logs.

What information should Wazuh extract from my logs?

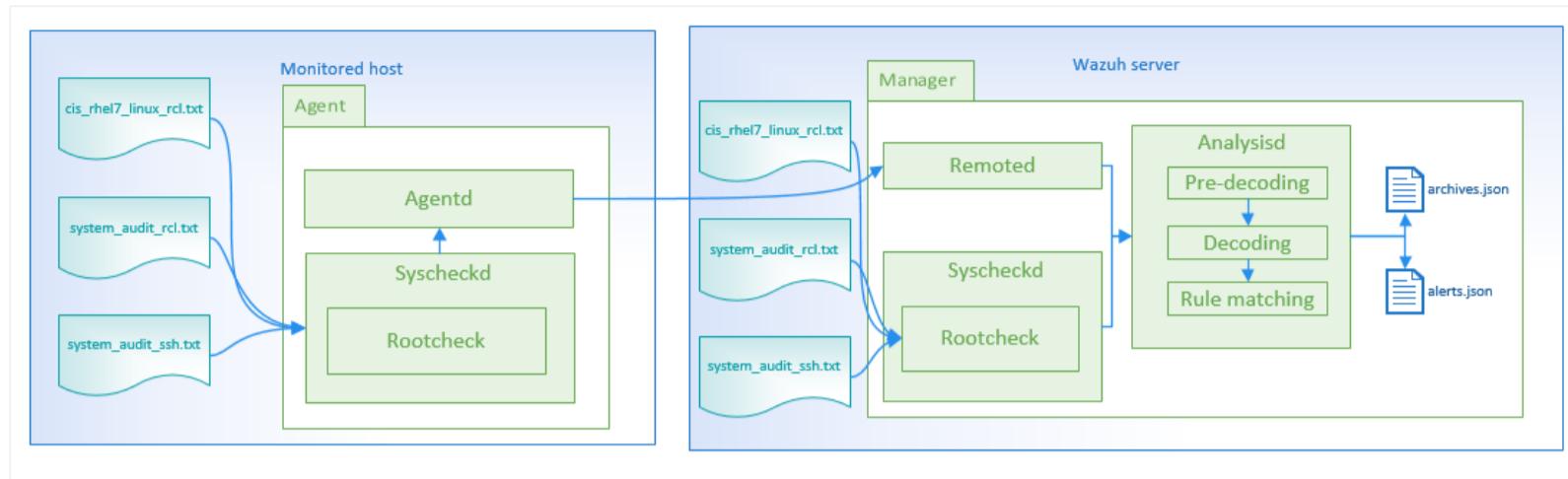
This depends on your needs. Once you know the format of your application logs and the typical events, you can create decoders and rules for them.

Can I ignore events that are not important?

You can configure the rules to ignore certain events. More info: [Custom rules](#)

How it works

Rootcheck allows to define policies in order to check if the agents meet the requirement specified.



The *rootcheck* engine can perform the following checks:

- check if a process is running
- check if a file is present
- check if the content of a file contains a pattern, or if a Windows registry key contains a string or is simply present.

Using these checks, the following policies have been developed:

Policy	Description
cis_debian_linux_rcl.txt	Based on CIS Benchmark for Debian Linux v1.0
cis_rhel5_linux_rcl.txt	Based on CIS Benchmark for Red Hat Enterprise Linux 5 v2.1.0
cis_rhel6_linux_rcl.txt	Based on CIS Benchmark for Red Hat Enterprise Linux 6 v1.3.0
cis_rhel7_linux_rcl.txt	Based on CIS Benchmark for Red Hat Enterprise Linux 7 v1.1.0
cis_rhel_linux_rcl.txt	Based on CIS Benchmark for Red Hat Enterprise Linux v1.0.5
cis_sles11_linux_rcl.txt	Based on CIS Benchmark for SUSE Linux Enterprise Server 11 v1.1.0
cis_sles12_linux_rcl.txt	Based on CIS Benchmark for SUSE Linux Enterprise Server 12 v1.0.0
system_audit_rcl.txt	Web vulnerabilities and exploits
win_audit_rcl.txt	Check registry values
system_audit_ssh.txt	SSH Hardening
win_applications_rcl.txt	Check if malicious applications are installed

Alerts related to policy monitoring:

- 512: Windows Audit
- 514: Windows Application
- 516: Unix Audit

The policy and compliance monitoring databases are normally maintained on the manager, which distributes them to all the agents.

Example of an existing policy rule:

```

# PermitRootLogin not allowed
# PermitRootLogin indicates if the root user can log in via ssh.
$sshd_file=/etc/ssh/sshd_config;

[SSH Configuration - 1: Root can log in] [any] [1]
f:$sshd_file -> !r:^# && r:PermitRootLogin\.+yes;
f:$sshd_file -> r:^#\s*PermitRootLogin;

```

Alert example:

```
** Alert 1487185712.51190: - ossec,rootcheck,
2017 Feb 15 11:08:32 localhost->rootcheck
Rule: 516 (level 3) -> 'System Audit event.'
System Audit: CIS - RHEL7 - 6.2.9 - SSH Configuration - Empty passwords permitted {CIS: 6.2.9 RHEL7}
{PCI_DSS: 4.1}. File: /etc/ssh/sshd_config. Reference:
https://benchmarks.cisecurity.org/tools2/linux/CIS_Red_Hat_Enterprise_Linux_7_Benchmark_v1.1.0.pdf .
title: CIS - RHEL7 - 6.2.9 - SSH Configuration - Empty passwords permitted
file: /etc/ssh/sshd_config
```

Configuration

1. [Basic usage](#)
2. [Configure periodic scans](#)
3. [Root access to SSH](#)

Basic usage

To configure the options for rootcheck, go to the [Rootcheck section](#) in [ossec.conf](#). The most common configuration options are: [frequency](#) and [system-audit](#)

Basic example to configure audit polices:

```
<rootcheck>
  <system_audit>./db/system_audit_rcl.txt</system_audit>
  <system_audit>./db/cis_debian_linux_rcl.txt</system_audit>
  <system_audit>./db/cis_rhel_linux_rcl.txt</system_audit>
</rootcheck>
```

Configure periodic scans

This is a basic configuration to run a scan every 10 hours.

```
<rootcheck>
  <frequency>36000</frequency>
  <system_audit>/var/ossec/etc/shared/system_audit_rcl.txt</system_audit>
  <system_audit>/var/ossec/etc/shared/cis_debian_linux_rcl.txt</system_audit>
  <system_audit>/var/ossec/etc/shared/cis_rhel_linux_rcl.txt</system_audit>
  <system_audit>/var/ossec/etc/shared/cis_rhel5_linux_rcl.txt</system_audit>
</rootcheck>
```

Root access to SSH

1. First you need to create your custom audit file (audit_test.txt):

```
# PermitRootLogin not allowed
# PermitRootLogin indicates if the root user can log in by ssh.
$sshd_file=/etc/ssh/sshd_config;

[SSH Configuration - 1: Root can log in] [any] [1]
f:$sshd_file -> !r:^# && r:PermitRootLogin\..+yes;
f:$sshd_file -> r:^#\s*PermitRootLogin;
```

2. Reference our new file in the rootcheck options:

```
<rootcheck>
  <system_audit>/var/ossec/etc/shared/audit_test.txt</system_audit>
</rootcheck>
```


FAQ

1. [Can I specify my own audit file for policy monitoring?](#)

Can I specify my own audit file for policy monitoring? 

Yes, you can use the *system_audit* option for that. Example [SSH rule](#)

