OSSEC HIDS Documentation

Release 3.3

OSSEC Project

Jan 21, 2020

Contents:

OSSEC is an open source host based intrustion detection system. It performs log monitoring, file integrity monitoring, Windows registry monitoring, rootkit detection, real-time alerting, and active-response. It runs on Microsoft Windows, and most modern Unix-like systems including Linux, FreeBSD, OpenBSD, and Solaris.

CHAPTER 1

About OSSEC

1.1 OSSEC Architecture

1.1.1 Linux and unix-like systems

OSSEC runs as multiple processes, the exact number differing between agent, server, and local installations. Most processes communicates through unix sockets under the *queue* directory inside of the OSSEC installation location. When possible the OSSEC processes run with limited privileges and chroot to the install location. This is modeled after the Privilege Separation tehniques used in OpenBSD.

1.1.2 Windows

OSSEC runs as a single service.

1.1.3 Agent/Server Communication

The OSSEC server listens on 1514/udp via *ossec-remoted*. Agents send messages to the server via *ossec-agentd*. The communication is two-way, but initiated by the agent.

1.1.4 Agentless and Network Devices

OSSEC has the ability to communicate with systems that cannot have the agent software installed. This is typically done through SSH, and a few pre-made Expect scripts are provided for a number of systems.

In addition to the agentless support, OSSEC can receive syslog messages from any number of devices and process them as if the messages were delivered via an agent.

1.1.5 User List

User	Process
root	ossec-syscheckd, ossec-execd, ossec-logcollector
ossec	ossec-analysisd, ossec-monitord, ossec-agentlessd
ossecr	ossec-remoted
ossecm	ossec-maild, ossec-csyslogd

1.1.6 Process List

Process	Description	Install Type
ossec-analysisd	Master program. Analyzes data from the logs, syscheck, rootcheck, etc. Runs as an unprivileged (ossec) user under chroot.	Server/Standalone
ossec-execd	Execute active responses by calling the configured scripts. Runs as root.	All
ossec-maild	Send e-mail alerts. Runs as an unprivileged user (ossecm) under chroot.	Server/Standalone
ossec-remoted	Server side socket for server/client communications. Runs as an unprivileged user (ossecr) under chroot.	Server
ossec-agentd	Agent side socket for server/client communications. Runs as an unprivileged user (ossec) under chroot.	Agent
ossec-logcollector	Monitor log files and windows event logs (do not use tail).	All
ossec-syscheckd	Does integrity checking and rootkit detection (rootcheck is a module of it).	All
ossec-csyslogd	Client syslog tool to forward OSSEC alerts to remote syslog servers (including SIEM and log management systems).	Server/Standalone
ossec-monitord	Monitor agent connectivity and compress daily log files.	Server/Standalone

1.2 Supported Systems

1.2.1 Operating Systems

The following operating systems are supported by the OSSEC agent:

- GNU/Linux (all distributions, including RHEL, Ubuntu, Slackware, Debian, etc)
- Windows 7, 8, 10
- Windows Server 2008r2, 2012, 2016
- FreeBSD (all current versions)
- OpenBSD (all current versions)
- NetBSD (all current versions)
- Solaris 10
- AIX 5.2 and 5.3
- Mac OS X 10.x
- HP-UX 11

1.2.2 Devices Supported via syslog

These systems/devices are also supported via remote syslog:

- Cisco PIX, ASA and FWSM (all versions)
- Cisco IOS routers (all versions)
- Juniper Netscreen (all versions)
- SonicWall firewall (all versions)
- Checkpoint firewall (all versions)
- Cisco IOS IDS/IPS module (all versions)
- Sourcefire (Snort) IDS/IPS (all versions)
- Checkpoint Smart Defense (all versions)
- Bluecoat proxy (all versions)
- Cisco VPN concentrators (all versions)

1.2.3 Devices and Operating Systems via Agentless

Using OSSEC agentless options, the following systems are also supported (for log analysis and file integrity checking):

- Cisco PIX, ASA and FWSM (all versions)
- Cisco IOS routers (all versions)
- Juniper Netscreen (all versions)
- SonicWall firewall (all versions)
- Checkpoint firewall (all versions)
- All operating systems specified in the “operating systems” section

1.2.4 Notes about support

The OSSEC project is made up of volunteers, mostly working on the project in their free time. Some systems will be harder to support due to availability. While we may all want a spare AIX or HP-UX system in our network closets, it’s a luxury not all of us can afford.

CHAPTER 2

OSSEC Manual

2.1 Installation

2.1.1 Installation Types

OSSEC can be installed in an agent/server combination or as a stand-alone system. The stand-alone installation is essentially a server installation without the pieces that interact with agents. The server installation includes the agent functionality for the local system.

Server

In an OSSEC server/agent installation, the agents pass log messages to the server for processing. Rules and decoders are installed only on the server. Alerts are generated and distributed from the server.

Agent

OSSEC agents tail the local log files and forward the messages to the OSSEC server. Local file integrity monitoring messages are also forwarded to the server.

Hybrid

A hybrid installation is both a server and agent. As a server it processes logs for a number of agents, and as an agent it forwards alerts to another server.

Local

A local, or stand-alone, installation resides entirely on a singular system. It is not associated with a server or agents. Decoders and rules will be stored on a local installation.

2.1.2 Installations requirements

For UNIX systems, OSSEC only requires gnu make, gcc, and libc. OpenSSL is a suggested, but optional, prerequisite. However, you always have the option to pre-compile it on one system and move the binaries to the final box.

PCRE2

New in version 3.3.

PCRE2 support has been added to version 3.3. The build system can either use the system's PCRE2 libraries, or the library can be built as part of the installation process.

The default build process expects the *pcre2-10.32* source to be installed in *src/external*:

```
$ cd ossec-hids-*/src
$ wget https://ftp.pcre.org/pub/pcre/pcre2-10.32.tar.gz
$ tar xzf pcre2-10.32.tar.gz -C src/external
```

To use the system's PCRE2, set the *PCRE2_SYSTEM* variable to yes:

```
# cd ossec-hids-*
# PCRE2_SYSTEM=yes ./install.sh
```

If the system's PCRE2 library is used, verify that JIT is enabled. If it is not, set *USE_PCRE2_JIT* to no:

```
# PCRE2_SYSTEM=yes USE_PCRE2_JIT=no ./install.sh
```

zlib

zlib is included with OSSEC in *src/external/zlib-1.2.11*. In previous versions this included version was used by default during the build process, but this changed to using the system zlib. Ensure the correct zlib development packages are installed.

To use the included version of zlib, simply set *ZLIB_SYSTEM* to *no*:

```
# cd ossec-hids-*
# ZLIB_SYSTEM=no ./install.sh
```

Ubuntu

On Ubuntu you will need the *build-essential* package in order to compile and install OSSEC.

To install the package run the following command.

```
# apt-get install build-essential zlib1g-dev
```

To use the system's pcre2 libraries, install the libpcre2 development package:

```
# apt-get install libpcre2-dev
```

If database support is needed *mysql-dev* or *postgresql-dev* should be installed. Run the following command to install these packages.

```
# apt-get install mysql-dev postgresql-dev
```

To use the SQLite features, the *libsqlite3-dev* package is necessary.

New in version 3.0.

```
# apt-get install libsqlite3-dev
```

RedHat/CentOS

RedHat should have most of the packages needed by default. The zlib development package should be installed:

```
# yum install zlib-devel
```

To use the system's pcre2 libraries, add the pcre2 development package:

```
# yum install pcre2-devel
```

If database support is needed the package mysql-devel and/or postgresql-devel will need to be installed.

```
# yum install mysql-devel postgresql-devel
```

To use the SQLite features, the *sqlite-devel* package is necessary.

New in version 3.0.

```
# yum install sqlite-devel
```

OpenSuse

The zlib development package should be installed:

```
# zypper install zlib-devel
```

To use the system's pcre2 libraries, add the pcre2 development package:

```
# zypper install pcre2-devel
```

If database support is needed the package mysql-devel and/or postgresql-devel will need to be installed.

```
# zypper install postgresql-devel mysql-devel
```

Debian

Warning: The Debian instructions are probably out of date. Contributions updating this section would be appreciated.

Debian has replaced bash with dash, and this may cause issues during installation. Dash does not appear to support all of the features available in other shells, and may display an error when trying to set the server's IP address on an agent system. The error can be ignored, but the server ip address will need to be set.

Do this by making sure something like the following information is in the agent's ossec.conf:

```
<ossec_config>
  <client>
    <server-ip>SERVER'S IP</server-ip>
  </client>
```

This can also be avoided by using bash to run `install.sh`:

```
# bash ./install.sh
```

Alpine Linux

To build OSSEC on Alpine Linux the following packages should be installed:

```
# apk add libc-dev pcre2-dev bsd-compat-headers libevent-dev openssl-dev zlib-dev␣
↪linux-headers
```

Even with the *linux-headers* package installed, make will not be able to find *a.out.h*. Until the installer is fixed, link *a.out.h* to *src/headers*:

```
$ ln -s /usr/include/linux/a.out.h ./headers/a.out.h
```

OpenBSD

To build OSSEC on OpenBSD the following packages should be installed:

```
# pkg_add pcre2 gmake
```

To add database support for a server the *postgresql-client* or *mariadb-client* will need to be installed.

For sqlite support, install the *sqlite3* package.

FreeBSD

TBD

2.1.3 Build Options

OSSEC has a number of options that can be set or changed at build time. These options can change the behavior or resource utilization by the OSSEC processes.

Required

TARGET

The *TARGET* is the type of system being built.

This is set during the compilation, either through the *install.sh* script or manually in the *src* directory:

```
$ cd ossec-hids-*/src
$ make TARGET=server
```

Available options:

- **server:** OSSEC Management Server
- **local:** Local OSSEC installation
- **agent:** OSSEC Agent
- **hybrid:** Hybrid OSSEC server
- **winagent:** Windows Agent

Optional

A number of options are set in the *Makefile*, but can be modified at compile time.

ZLIB_SYSTEM

If set to *yes* the system installed zlib will be used

Default: yes

PCRE2_SYSTEM

If set to *yes* the system installed pcre2 will be used

Default: no

LUA_ENABLE

If set to *yes* the installation process will build lua

Default: no

MAXAGENTS

The maximum number of supported agents

Default: 2048

USE_PRELUDE

Enable forwarding alerts to Prelude-IDS

Default: no

USE_ZEROMQ

Enable forwarding events via ZeroMQ

Default: no

USE_GEOIP

Enable Lookups against the GeoIP database

Default: no

USE_INOTIFY

Enable realtime support in *syscheck* via *inotify* in Linux

Default: enabled on Linux and Windows, disabled on other operating systems

USE_PCRE2_JIT

Enable the use of pcre2 Just In Time (JIT) support. If the installation process compiles PCRE2 this will be enabled. If the system installed PCRE2 is used, it must support JIT if this is enabled.

Default: yes

REUSE_ID

Allow *authd* to re-use agent IDs

Default: no

Settings

A listing of the default settings.

```
$ cd ossec-hids-*/src
$ make settings

General settings:
    TARGET:           server
    V:
    DEBUG:
    DEBUGAD:
    PREFIX:           /var/ossec
    MAXAGENTS:        2048
    REUSE_ID:         no
    DATABASE:
    ONEWAY:           no
    CLEANFULL:        no
User settings:
    OSSEC_GROUP:      ossec
    OSSEC_USER:       ossec
    OSSEC_USER_MAIL:  ossecm
    OSSEC_USER_REM:   ossecr
ZLIB settings:
    ZLIB_SYSTEM:      yes
    ZLIB_INCLUDE:
    ZLIB_LIB:         os_zlib.a
PCRE2 settings:
```

(continues on next page)

(continued from previous page)

```
    PCRE2_SYSTEM:       y
    PCRE2_INCLUDE:
Lua settings:
    LUA_PLAT:           posix
    LUA_ENABLE:         no
USE settings:
    USE_ZEROMQ:         no
    USE_GEOIP:          no
    USE_PRELUDE:        no
    USE_OPENSSL:        auto
    USE_INOTIFY:        no
    USE_SQLITE:
    USE_PCRE2_JIT:      yes
Mysql settings:
    includes:
    libs:
Pgsql settings:
    includes:
    libs:
Defines:
    -DMAX_AGENTS=2048 -DOSSECHIDS -DDEFAULTDIR="/var/ossec" -DUSER="ossec" -DREMUSER=
↪"ossecr" -DGROUPGLOBAL="ossec" -DMAILUSER="ossecm" -DOpenBSD -pthread -DZLIB_SYSTEM␣
↪-DUSE_PCRE2_JIT -DLIBOPENSSL_ENABLED
Compiler:
    CFLAGS            -I/usr/local/include -DMAX_AGENTS=2048 -DOSSECHIDS -DDEFAULTDIR="/
↪var/ossec" -DUSER="ossec" -DREMUSER="ossecr" -DGROUPGLOBAL="ossec" -DMAILUSER=
↪"ossecm" -DOpenBSD -pthread -DZLIB_SYSTEM -DUSE_PCRE2_JIT -DLIBOPENSSL_ENABLED -
↪Wall -Wextra -I./ -I./headers/
    LDFLAGS            -lm -L/usr/local/lib -lpcre2-8 -lssl -lcrypto -lz
    CC                cc
    MAKE              gmake
```

2.1.4 Install from source

For source installations the *install.sh* script will ask questions about the installation, compile, and install the necessary files and users.

Please check the *Installations requirements* to ensure everything necessary is installed. This document will assume the installation requirements are met.

Manager/Agent Installation

1. Download the latest version and verify its checksum.
2. Extract the compressed OSSEC tarball. and run the `install.sh` script. It will guide you through the installation and compile the source (not shown).

   ```
   $ tar -zxvf ossec-hids-*.tar.gz (or gunzip -d; tar -xvf)
   $ cd ossec-hids-*
   $ sudo ./install.sh
   ```

3. The OSSEC manager listens on UDP port 1514. All firewalls between the agents and the manager will need to allow this traffic.
4. Start OSSEC HIDS by running the following command:

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

Manual Installation

OSSEC can also be installed in a more manual fashion without the use of *install.sh*. This installation method requires manual configuration of the *ossec.conf* file. The *ossec*, *ossecm*, and *ossecr* users will still be created automatically.

After the source tarball is downloaded and extracted:

```
$ cd ossec-hids-*/src
$ make TARGET=<server|local|agent>
$ sudo make install
```

Build options can still be passed to *make* (*USE_ZEROMQ*, *USE_GEOIP*, etc.).

2.1.5 Package Installation

The OSSEC project has made RPM and deb packages available. Links to the packages can be found on the OSSEC download page

RPM Installation

OSSEC's RPMs are made available by AtomiCorp.

The RPMs can be installed by adding the AtomiCorp yum repository:

```
# wget -q -O - https://updates.atomicorp.com/installers/atomic | sh
```

Next use `yum` to install the specific packages. For an OSSEC server run:

```
# yum install ossec-hids ossec-hids-server
```

And for an agent run:

```
# yum install ossec-hids ossec-hids-agent
```

Deb Installation

Atomicorp provides *.deb* installation packages as well.

Install the apt-get repository key:

```
# wget -q -O - https://www.atomicorp.com/RPM-GPG-KEY.atomicorp.txt  | sudo apt-key␣
↪add -
```

Add the repository for Debian and Ubuntu:

```
# wget -q -O - https://updates.atomicorp.com/installers/atomic | sudo bash
```

Update the repository:

Install OSSEC HIDS server/manager:

```
# apt-get install ossec-hids-server
```

Or install OSSEC HIDS agent:

```
# apt-get install ossec-hids-agent
```

FreeBSD Ports

TBD

2.1.6 Windows Agent Installation

Note: OSSEC only supports Windows systems as agents, and they will require an OSSEC server to function.

OSSEC Windows executable

Download the executable named Agent Windows from https://ossec.net/downloads.html. Run through the install wizard with all defaults. The Ossec Agent Manager should launch when the installation completes. The IP address of the server and the agent key can be pasted into the OSSEC Agent Manager.

The OSSEC service can be restarted via the Manage tab.

Chocolatey

Chocolatey is a package manager for Windows. It offers an OSSEC package.

Chocolatey installation instructions

Warning: Some more content should go here.

2.1.7 Compiling OSSEC for a Binary Installation

Warning: This installation method should be revisited, it may not work.

In the past OSSEC was often compiled on each system it was installed on. This is not the only method of installation however. OSSEC can be compiled on one system and copied to a destination system. This method of installation still requires GNU make on the target system, but not the compilers or development packages.

Note: OSSEC has very limited cross compiling facilities. Windows binaries can be built on Linux systems, but binaries for other systems should be built on a system of the same OS and CPU platform.

Compiling OSSEC for install on a second server

First download the OSSEC package corresponding to the version you want to install and extract it on the build system.

Enter in the source directory of the downloaded package, and configure OSSEC to build the `agent` version. The `make` commands should compile the correct binaries.

```
$ cd ossec-hids-*/src
$ make TARGET=agent
```

Modify *ossec-hids-*/etc/preloaded-vars.conf* to set *BINARY_INSTALL* to yes.

```
$ echo "USER_BINARYINSTALL=\"y\"" >> ossec-hids*/etc/preloaded-vars.conf
```

Finally create an OSSEC package.

```
$ tar -cvf ossec-binary.tar ossec-hids*
```

Installation of the binary OSSEC package

On the target system (that does not have a C compiler) download your ossec-binary.tar created in the steps above.

Complete the installation by unarchiving the binary package and running *sudo ./install.sh*.

```
$ tar xfv ossec-binary.tar
$ cd ossec-*
$ sudo ./install.sh
```

After following the installation prompts the install will be complete.

2.1.8 Compiling OSSEC 3.x with MinGW

OSSEC's Windows agent is compiled on Linux using MinGW

Requirements

- MinGW
- nsis
- OpenSSL/LibreSSL (optional)

New in version 3.3.

- PCRE2 source tree in *src/external*

Compilation

Change directory to the src directory:

```
$ cd ossec-hids-*/src
```

Download and extract the pcre2 source:

```
$ wget https://ftp.pcre.org/pub/pcre/pcre2-10.32.tar.gz
$ tar xzf pcre2-10.32.tar.gz -C src/external
```

Run `make TARGET=winagent`:

```
$ make TARGET=winagent
```

This should produce a good amount of compilation output that ends with:

```
Output: "ossec-win32-agent.exe"
Install: 7 pages (448 bytes), 3 sections (3144 bytes), 769 instructions (21532 bytes),
↪ 318 strings (32350 bytes), 1 language table (346 bytes).
Uninstall: 5 pages (320 bytes),
1 section (1048 bytes), 350 instructions (9800 bytes), 184 strings (3360 bytes), 1␣
↪language table (290 bytes).
Datablock optimizer saved 100205 bytes (~8.1%).

Using zlib compression.

EXE header size:               57856 / 56320 bytes
Install code:                  14832 / 58196 bytes
Install data:                1045670 / 3116385 bytes
Uninstall code+data:           21058 / 21474 bytes
CRC (0x239C5E6F):                  4 / 4 bytes

Total size:                  1139420 / 3252379 bytes (35.0%)
make settings
make[1]: Entering directory `/home/ddp/src/projects/git/github/ddpbsd/ossec-hids/src'

General settings:
    TARGET:            winagent
    V:
    DEBUG:
    DEBUGAD
    PREFIX:            /var/ossec
    MAXAGENTS:         2048
    DATABASE:
    ONEWAY:            no
    CLEANFULL:         no
User settings:
    OSSEC_GROUP:       ossec
    OSSEC_USER:        ossec
    OSSEC_USER_MAIL:   ossecm
    OSSEC_USER_REM:    ossecr
Lua settings:
    LUA_PLAT:          posix
USE settings:
    USE_ZEROMQ:        no
    USE_GEOIP:         no
    USE_PRELUDE:       no
    USE_OPENSSL:       no
    USE_INOTIFY:       no
Mysql settings:
    includes:
    libs:
Pgsql settings:
    includes:
```

(continues on next page)

(continued from previous page)

```
    libs:
Defines:
    -DMAX_AGENTS=2048 -DOSSECHIDS -DDEFAULTDIR="/var/ossec" -DUSER="ossec" -DREMUSER=
↪"ossecr" -DGROUPGLOBAL="ossec" -DMAILUSER="ossecm" -DLinux
 Compiler:
    CFLAGS           -O2 -DMAX_AGENTS=2048 -DOSSECHIDS -DDEFAULTDIR="/var/ossec" -
↪DUSER="ossec" -DREMUSER="ossecr" -DGROUPGLOBAL="ossec" -DMAILUSER="ossecm" -DLinux -
↪Wall -Wextra -I./ -I./headers/
    LDFLAGS          -lm
    CC              gcc
    MAKE            make
make[1]: Leaving directory `/home/ddp/src/projects/git/github/ddpbsd/ossec-hids/src'

Done building winagent
```

The final output will be saved to `./win32/ossec-win32-agent.exe`.

2.2 Agent Management

On the OSSEC management server an agent can be added in multiple ways. The first method is adding an agent with the *manage_agents* utility. The second is using *ossec-authd*. This is a process on the ossec management server that receives key requests, adds an agent, and sends the key to the new agent.

2.2.1 manage_agents

The *manage_agents* utility is run on both the OSSEC management server and the OSSEC agent. On the management server it will add an agent and export a key to be imported on the agent. On the agent the *manage_agents* utility will import a key for authenticating the agent to the management server.

manage_agents provides both a menu based interface and a command line interface. When run without any arguments, the menu is presented.

If an agent is added with a specific IP address, it must be unique. Duplicate IP addresses will cause issues with agents connecting to the server.

2.2.2 Adding an agent

The menu interface for *manage_agents*

```
# /var/ossec/bin/manage_agents

****************************************
* OSSEC HIDS v3.2.0 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:
```

Typing *a* or *A* will provide prompts for information about the agent.

When adding an agent, *manage_agents* will prompt for information about the agent. This information includes an ID, name, and IP address. Each entry should be unique. Duplicating information will cause issues with agents connecting to the server.

The ID is a number, starting with 1. This ID is used by OSSEC for configuration.

The name of the agent can also be used in a number of configuration options.

The IP address should be the IP address the OSSEC server will see from the agent. Instead of a specific IP address a CIDR address can be used for DHCP ranges or *any* can be used if the IP is not known (dhcp from an ISP).

After the first agent is added, the OSSEC server processes should be restarted.

The key that is extracted is a secret key, and should be kept safe. If this key is disclosed, a malicious user could impersonate the agent.

2.2.3 ossec-authd

Instead of adding agents manually, *ossec-authd* can be used. *ossec-authd* is a daemon that listens for TLS connections, adds the connecting machine as an agent, and replies with a new key. On the agent, the *agent-auth* program is run to communicate with *ossec-authd*.

Support for this requires the OpenSSL development libraries.

2.2.4 Adding an agent example

```
OSSEC-SERVER # /var/ossec/bin/manage_agents


****************************************
* OSSEC HIDS v3.2.0 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: A

- Adding a new agent (use '\q' to return to the main menu).
  Please provide the following:
   * A name for the new agent: test
   * The IP Address of the new agent: 10.10.10.10
   * An ID for the new agent[1025]:
Agent information:
   ID:1025
   Name:test
   IP Address:10.10.10.10

Confirm adding it?(y/n): y
Agent added with ID 1025
```

After the agent is added, its key can be extracted with *E* at the menu.

```
OSSEC-AGENT # /var/ossec/bin/manage_agents

****************************************
* OSSEC HIDS v3.2.0 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: e

Available agents:
   ID: 010, Name: public, IP: 192.168.17.12
   ID: 1024, Name: junction.example.com, IP: any
   ID: 1025, Name: test, IP: 10.10.10.10
Provide the ID of the agent to extract the key (or '\q' to quit): 1025

Agent key information for '1025' is:
MTAyNSB0ZXN0IDEwLjEwLjEwLjEwIDQ3ZDFkOGFiMzU5OWRiNDkyNTU4NjkzOGNiYTY4YTM5NmMwNmIwZmJkOTY3YWVjMmUzNzU4

** Press ENTER to return to the main menu.
```

This key can be added to an agent to associate it with a manager.

2.2.5 Adding an agent with authd example

2.3 Agentless Management

OSSEC has the capability to monitor systems which cannot accept an agent. This is done via the *ossec-agentlessd* daemon for some checks, and syslog for log monitoring.

2.3.1 agentlessd

ossec-agentlessd performs active checks on devices that do not have OSSEC agents installed on them. These checks can include checking configurations for modification and running checksums on files on the system.

Active agentless monitoring is disabled by default. It can be enabled with the following command:

```
# /var/ossec/bin/ossec-control enable agentless
```

This does not start the process, only enables it. It can be started manually by restarting the OSSEC processes:

```
# /var/ossec/bin/ossec-control restart
```

Or by starting the process manually:

```
# /var/ossec/bin/ossec-agentlessd
```

2.3.2 Adding Hosts

Hosts can be added via the *register_host.sh* script.

The user, host, and password to access the host can be set with this script.

```
# /var/ossec/agentless/register_host.sh
/var/ossec/agentless/register_host.sh options:
     add <user@host> [<passwd>] (<additional_pass>)
     list (passwords)
```

These hosts must also be defined in the *ossec.conf* to define which checks are performed on them.

More information can be found in the *ossec.conf* configuration documentation. The agentless section provides details.

The default checks bundled in OSSEC are:

```
main.exp
ssh.exp
ssh_asa-fwsmconfig_diff
ssh_foundry_diff
ssh_generic_diff
ssh_integrity_check_bsd
ssh_integrity_check_linux
ssh_nopass.exp
ssh_pixconfig_diff
sshlogin.exp
su.exp
```

2.4 Log Monitoring

2.4.1 What is log monitoring

OSSEC can monitor log messages in real-time, comparing them to a set of pre-defined rules. These rules can trigger alerts to notify analysts or administrators of a possible issue to be investigated.

These log messages can currently come from log files on the system, commands run by OSSEC on the system and via syslog from networked devices.

2.4.2 log files

OSSEC can monitor text based log files from syslog on all systems, and *EventLog* or *EventChannel* log formats on Windows based systems.

The *localfile* option can be used to monitor a log file. See the syntax for *localfile* for more information.

2.4.3 commands

OSSEC is able to run commands on a system and monitor the output. The command will be run periodically, with a admin configurable frequency. The command output can be monitored for content directly, or for changes between runs.

See the syntax for *command* for more information..

2.4.4 syslog

OSSEC is able to receive syslog messages from systems that cannot have an agent installed on them. This insludes many network devices like routers, firewalls, and switches.

See the *remote* documentation for more information.

2.5 Decoders and Rules

2.5.1 Decoders

In OSSEC decoders are an attempt to parse a log message, extracting important information for use elsewhere. Information like user names or IP addresses can be passed to rules for comparison or differentiated from other information in the final log. This can make importing the log messages into a SIEM or log management system easier.

decoder

Each decoder starts by labeling it as a decoder and giving it a name. The *name* must be unique.

```
<decoder name="widget-processor">
  ...
</decoder>
```

parent

A decoder may build on the parsing done by a previous decoder. This is defined by setting a *parent* for the second decoder, using the parent's name.

```
<decoder name="widget-auth">
  <parent>widget-processor</parent>
  ...
</decoder>
```

accumulate

A number of programs will log a single event over multiple messages, using a unique identifier to link them together. The *accumulate* option will allow OSSEC to track events over multiple log messages based on a "decoded ID" within the log message. The *accumulate* option requires an *id* field as seen in the example below.

```
 <decoder name="widget-transaction">
   <parent>widget-processor</parent>
   <regex>ID: (\d+) </regex>
   <order>id</order>
   <accumulate />
</decoder>
```

program_name

This compares the defined value to the *program_name* the pre-decoder assigns to a log message. This uses the simple regex syntax.

program_name_pcre2

This compares the defined value to the pre-decoded *program_name* using the pcre2 syntax.

prematch

A search string that must match for the decoder to continue evaluation. This uses the simple regex syntax.

prematch_pcre2

A search string that must match for the decoder to continue evaluation. This option uses the pcre2 syntax.

regex

An OSSEC regex formatted search string. Parts of the log message may be extracted into fields named in *order*.

pcre2

A pcre2 regex formatted search string. Parts of the log message may be extracted into fields namd in *order*.

order

This option names the fields created by pcre2 or *regex*. The field names are comma separated.

Historically there were only a few field names available for the *order* option, but as of version 3.3 any names may be used.

The historical *order* list:

- location - where the log came from (only on FTS)
- 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 - dst 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)
- extra_data - Any extra data

The following fields may be used in *active response* configurations:

- *user*
- *srcip*
- *filename* (syscheck file name?)

```
<decoder name="widget-reg">
  <pcre2>widget id: (\S+), name: (\w+)$</pcre2>
  <order>widget_id, widget_name</order>
</decoder>
```

fts

OSSEC is able to create alerts based on *First Time Seen* events. This option enables the *fts* tracking for the specified fields.

```
<decoder name="widget-fts">
  <fts>name, user, location</fts>
```

type

Categorize log messages into different groups. This could include firewall, windows, ids, web-log, etc.

2.5.2 Rules

Rules compare log messsages to a set of pre-defined conditions. The comparisons can happen on the entire log message, or on fields defined in decoders.

rule

Each rule begins by defining certain settings

- *level*

This defines the severity of the rule. Valid levels are 0-16.

- *id*

A unique identification number for the rule.

- *maxsize*

Specifies the maximum size of the event. The valid range is 1-99999

- *frequency*

Specifies the number of times the rule must have matched before firing. The number that triggers the rule is actually 2 more than this setting.

Note: More information about how frequency is counted can be found in this thread.

- *timeframe*

The timeframe in seconds. This option is intended to be used with the frequency option.

- *ignore*

The time (in seconds) to ignore this rule after firing it (to avoid floods).

- *overwrite*

Used to supercede an OSSEC rule with local changes. This is useful to change the level or other options of rules included with OSSEC.

- *noalert*

Prevent the rule from triggering an alert. Further rule checks will not happen, except for rules specifically using this rule in an <*if_sid*> configuraiton.

match

A simple string comparison.

regex

This option uses the OSSEC *regex* syntax for comparisons.

pcre2

The *pcre2* option utlizes OSSEC's pcre2 support. Refer to the pcre2 page for information on the syntax.

decoded_as

Define a decoder that must be matched for the rule comparison to continue.

category

The decoded category to match (ids, syslog, firewall, web-log, squid or windows).

srcip

Any IP address or CIDR block to be compared to an IP decoded as srcip. Use "!" to negate it.

dstip

Any IP address or CIDR block to be compared to an IP decoded as dstip. Use "!" to negate it.

extra_data

Any string that is decoded into the `extra_data` field.

user

Any username (decoded as the username).

program_name

Program name is decoded from syslog process name.

hostname

Any hostname (decoded as the syslog hostname) or log file.

time

Time that the event was generated. Any time range can be defined, in the format of *hh:mm-hh:mm*. AM/PM can also be used: *<time>6 am - 6 pm</time>*

weekday

Specify a week day that the event was generated. Multiple entries can be separated by commas.

id

Any ID (decoded as the ID).

url

Any string decoded into the *url* field.

if_sid

Matches if the rule ID has matched. This is used to create children to other rules.

if_group

Matches if the group has matched before. This can be used to create children of other rules.

if_level

Matches if the level has matched before.

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*.

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*.

same_id

Specifies that the decoded id must be the same. This option is used in conjunction with *frequency* and *timeframe*.

same_source_ip

Specifies that the decoded source ip must be the same. This option is used in conjunction with *frequency* and *timeframe*.

same_source_port

Specifies that the decoded source port must be the same. This option is used in conjunction with *frequency* and *timeframe*.

same_dst_port

Specifies that the decoded destination port must be the same. This option is used in conjunction with *frequency* and *timeframe*.

same_location

Specifies that the location must be the same. This option is used in conjunction with *frequency* and *timeframe*.

same_user

Specifies that the decoded user must be the same. This option is used in conjunction with *frequency* an *timeframe*.

description

The rule description.

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.

- *field*

  Field that is used as the key to look up in the CDB file:

  - Value: srcip
  - Value: srcport
  - Value: dstip
  - Value: dstport
  - Value: extra_data
  - Value: user
  - Value: url
  - Value: id
  - Value: hostname
  - Value: program_name
  - Value: status
  - Value: action

- *lookup*

  This is the type of lookup that is preformed:

  - Value: match_key
    - Positive key match: field is the key to search within the cdb and will match if they key is present.
    - This is the default if no lookup is specified.
  - Value: not_match_key
    - Negative key match: field is the key to search and will match if it *IS NOT* present in the database.
  - Value: match_key_value
    - Key and Value Match: field is searched for in the cdb and if found the value will be compared with regex from attribute check_value.

    Note: This feature is not yet complete.
  - Value: address_match_key
    - Positive key match: field is an IP address and the key to search within the cdb and will match if they key is present.
  - Value: not_address_match_key
    - Negative key match: field is an IP address the key to search and will match if it *IS NOT* present in the database.

- Value: address_match_key_value
  - Key and Value Match: field is an IP address searched for in the cdb and if found the value will be compared with regex from attribute check_value.

Note: This feature is not yet complete.

- *check_value*
  - regex pattern for matching on the value pulled out of the cdb when using lookup types: address_match_key_value, match_key_value

Path to the CDB file to be used for lookup from the OSSEC directory. This file must also be included in the ossec.conf file.

Example:

```
<rule id="100000" level="7">
      <list lookup="match_key" field="srcip">path/to/list/file</list>
      <description>Checking srcip against cdb list file</description>
</rule>
```

info

Extra information may be added to an alert using *info*. The type must be specified using one of the following options:

- *type*
  - Value: text

    This is the default when no type is selected. Just used for additional information about the alert/event.
  - Value: link

    Link to more information about the alert/event.
  - Value: cve

    The CVE Number related to this alert/event.
  - Value: ovsdb

    The osvdb id related to this alert/event.

Example:

```
<rule id="502" level="3">
    <if_sid>500</if_sid>
    <options>alert_by_email</options>
    <match>Ossec started</match>
    <description>Ossec server started.</description>
    <info type="link">http://ossec.net/wiki/Rule:205</info>
    <info type="cve">2009-1002</info>
    <info type="osvdb"> 61509</info>
    <info type="text">Internal Why we are running this run in our␣
↪company</info>
    <info>Type text is the default</info>
</rule>
```

options

Additional rule options

- *alert_by_email*

  Always alert by email.

  – *Example:* <options>alert_by_email</options>

- *no_email_alert*

  Never alert by email.

  – *Example:* <options>no_email_alert</options>

- *no_log*

  Do not log this alert.

  – *Example:* <options>no_log</options>

check_diff

Used to determine when the output of a command changes.

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.

2.6 Regular Expression Syntax

Currently OSSEC supports three regex syntaxes: *OS_Regex*, *OS_Match*, and *pcre2*. *OS_Regex/regex* is an OSSEC specific subset of regular expressions. This syntax is considered legacy, and is being replaced by the *pcre2* variant. *pcre2* is provided by the pcre2 library. It is much more complete and more widely tested than OS_Regex. *pcre2* will be the preferred syntax moving forward. *OS_Match/sregex* is an even smaller subset of regular expressions. It provides some very basic mechanisms used in matching, and is widely available inside OSSEC.

2.6.1 pcre2

Information on the syntax for pcre2 can be found in the pcre documentation.

2.6.2 OS_Regex/regex

Warning: This syntax is legacy, please use the *pcre2* syntax for anything new.

Fast and simple library for regular expressions in C.

This library is designed to be simple, but support the most common regular expressions. It was designed with intrusion detection systems in mind, where having all options is not crucial, but speed is.

Supported expressions:

```
\w  ->  A-Z, a-z, 0-9, '-', '@' characters
\d  ->  0-9 characters
\s  ->  For spaces " "
\t  ->  For tabs.
\p  ->  ()*+,-.:;<=>?[]!"'#$%&|{} (punctuation characters)
\W  ->  For anything not \w
\D  ->  For anything not \d
\S  ->  For anything not \s
\.  ->  For anything
```

Modifiers:

```
+   ->  To match one or more times (eg \w+ or \d+)
*   ->  To match zero or more times (eg \w* or \p*)
```

Special Characters:

```
^ -> To specify the beginning of the text.
$ -> To specify the end of the text.
| -> To create an "OR" between multiple patterns.
```

Characters Escaping

To utilize the following characters they must be escaped:

```
$ -> \$
( -> \(
) -> \)
\ -> \\
| -> \|
```

2.6.3 OS_Match/sregex

Faster than the OS_Regex/regex, but only supports simple string matching and the following special characters.

Special Characters:

```
^ -> To specify the beginning of the text.
$ -> To specify the end of the text.
| -> To create an "OR" between multiple patterns.
```

CHAPTER 3

Configuration

3.1 ossec.conf

The configuration for OSSEC is mostly held in *ossec.conf*. It is written in loose XML, and consists of a number of sections.

3.1.1 global

The <*global*> section is valid on Server and Local installations only.

email_notification

enable or disable email alerting.

Default: no

Allowed: yes/no

Enable email_notification

```
<email_notification>yes</email_notification>
```

Disable email_notification

```
<email_notification>no</email_notification>
```

email_to

Email address alerts are sent to.

Allowed: Any valid email address.

Set an email address

```
<email_to>security@example.com</email_to>
```

email_from

Email address the alert emails will come from.

Allowed: Any valid email address.

Set an email address

```
<email_from>ossec@example.com</email_from>
```

reply_to

New in version 3.0.

Email "Reply-to" for alert emails.

Allowed: Any valid email address.

Set an email address

```
<reply_to>team@example.com</reply_to>
```

smtp_server

Hostname or IP address alert emails will be sent to.

Allowed: IP address or hostname of an smtp server.

Set the SMTP server to an IP address

```
<smtp_server>10.0.0.25</smtp_server>
```

Set the SMTP server to a hostname

```
<smtp_server>mail.example.com</smtp_server>
```

email_maxperhour

The maximum number of emails that can be sent per hour. Emails in excess of this will be queued for later distribution.

Default: 12

Allowed: 1-9999

Note: At the end of the hour any queued emails will be sent together. This is true whether email grouping is enabled or disabled.

Set the maximum numer of emails per hour

```
<email_maxperhour>700<email_maxperhour>
```

custom_alert_output

Specifies a custom format for alerts written to the *alerts.log* file.

```
Variables:
"$TIMESTAMP"    -       The time the event was processed by OSSEC.
"$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.
```

Note: Some OSSEC daemons rely on the standard alerts log format to functon properly. Using a custom log format may prevent *ossec-maild* or others from working.

stats

Alerting level for the events generated by the statistical analysis.

Default: 8

Allowed: Any level from 0-16

Note: XXX I actually have no idea what this does, and the description makes no sense.

logall

If enabled, *ossec-analysisd* will log all messages it receives to the *archives.log* file. The log messages will be prefixed with OSSEC meta-data before being written.

Default: no

Allowed: yes/no

Turn on the logall option

```
<logall>yes</logall>
```

memory_size

Sets the memory size for event correlation.

Default: 1024

Allowed: Any size from 16 to 5096

Note: XXX Here is another one I don't understand.

allow_list

List of IP addresses that should never be blocked by active response. One host should be specified per instance of *allow_list*. Multiple *allow_list* options can be specified.

Allowed: Any IP address or netblock

Valid on:* Server and Local

Allow an IP address

```
<allow_list>192.168.1.100</allow_list>
```

Allow an IP block

```
<allow_list>10.0.0.0/24</allow_list>
```

host_information

Alerting level for events generated by the host change monitor.

Default: 8

Allowed: Any level from 0-16

jsonout_output

New in version 2.9.0.

Enable or disable alert logging in a json format. Alerts will be saved to *alerts.json*.

Default: no

Allowed: yes/no

Valid on:* Server and Local

prelude_output

Deprecated since version 3.4.

Enables or disables output to Prelude-IDS.

Default: no

Allowed: yes/no

Warning: Support for this is rarely tested, and may not work. Consider it deprecated.

zeromq_output

Enable ZeroMQ output of alerts.

Default: no

Allowed: yes/no

Valid on: Server and Local

zeromq_url

The URI for the ZeroMQ publisher socket.

Allowed: URI as specified by the format used by the ZeroMQ project.

Note: The format of this URI is specified by the ZeroMQ project. XXX Find the specification and link to it.

Examples:

Listen on localhost, tcp port 11111:

```
<zeromq_uri>tcp://localhost:11111/</zeromq_uri>
```

Listen on tcp port 21212 on all IP addresses assigned to eth0:

```
<zeromq_uri>tcp://eth0:21212/</zeromq_uri>
```

Listen on the Unix Domain Socket */alerts-zmq*:

```
<zeromq_uri>ipc:///alerts-zmq</zeromq_uri>
```

geoip_db_path

The full path to hte GeoIP IPv4 database file.

Example:

```
<geoip_db_path>/etc/GeoListeCity.dat</geoip_db_path>
```

geoip6_db_path

The full path to hte GeoIP IPv6 database file.

Example:

```
<geoip6_db_path>/etc/GeoListeCity.dat</geoip6_db_path>
```

md5_whitelist

New in version 3.0.

Defines an SQLite database for white listed MD5 hashes. The path should begin at the root of the OSSEC installation (*/var/ossec* by default)..

Example:

```
<md5_whitelist>/rules/lists/md5whitelist.db</md5_whitelist>
```

3.1.2 client

Settings defining how an OSSEC agent interacts with the OSSEC management server. The *<client>* section is valid on Agents only.

server-ip

Specifies the IP address of the OSSEC management server.

Allowed: Any valid IP address

server-hostname

Specifies the hostname of the OSSEC management server.

Allowed: Any valid hostname

port

Specifies the port used by the OSSEC management server.

Default: 1514/udp

Allowed: Any port number

config-profile

Specifies the profile to be used by the agent. The profiles are defined in the *agent.conf* on the OSSEC management server. Multiple profiles can be specified, separated by a comma and a space.

Allowed: Valid profile name

notify_time

Specifies the time in seconds between messages sent to the OSSEC management server.

Note: XXX More details needed.

time-reconnect

Time in seconds until a reconnection attempt is made. This should be set to a number greater than *notify_time*.

Note: XXX More details needed.

3.1.3 remote

Settings defining the configuration of *ossec-remoted* on the OSSEC management server.

connection

Specifies the type of connection *ossec-remoted* will accept. Two types of connections are accepted:

- secure Messages from agents are encrypted and authenticated. Uses UDP.
- syslog Messages from devices are not encrypted or authenticated. Can use UDP or TCP

Default: secure

Allowed: secure or syslog

port

The port utilized by *ossec-remoted*.

Default: 1514 for secure, 514 for syslog

Allowed: Any port number

protocol

The protocol used by *ossec-remoted* for syslog messages.

Default: udp

Allowed: udp or tcp

allowed-ips

A list of IP addresses that are permitted to send syslog messages to *ossec-remoted*. Each instance of *allowed-ips* can specify one IP address. Multiple instances are permitted.

Allowed: Any IP address or network

Note: If using the syslog connection type, at least one IP address must be specified.

deny-ips

A list of IP addresses that are not permitted to send syslog messages to *ossec-remoted.* Each instance of *deny-ips* can specify one IP address. Multiple instances are permitted.

*Allowed:** Any IP address or network

local_ip

The local IP address *ossec-remoted* will listen on.

Default: All interfaces

Allowed: Any local IP address

ipv6

The local IPv6 address *ossec-remoted* will listen on.

Default: None

Allowed: Any local IPv6 address

3.1.4 syscheck

Settings controlling the file integrity monitoring features in OSSEC. Most settings should be configured on the system they apply to, but settings that are only valid on the management server or local installs have been marked as such.

directories

The *<directories>* option specifies which directories *ossec-syscheckd* will monitor. Multiple directories can be specified per instance, separated with a comma. Windows drive letters without directories are not valid, at a minimum . should be included (*D:.*). These settings are local to the system they are configured on.

Default: */etc,/usr/bin,/usr/sbin,/bin,/sbin*

Attributes:

- **check_all:**

  The directories are monitored for changes in file hash, size, owner, group, and permissions. This can be over ridden by setting each specific option to *no*. The current specific checks are: *check_md5sum*, *check_sha1sum*, *check_size*, *check_owner*, *chek_group*, and *check_perm*.

  Value: yes/no

- **check_sum:**

  Check the md5 and sha1 hashes of the monitored files.

  Value: yes

- **check_sha1sum:**

  Check the sha1 hash.

  Value: yes

- **check_md5sum:**

  Check the md5 hash.

  Value: yes

- **check_size:**

  Check the size.

  Value: yes

- **check_owner:**

  Check the owner.

  Value: yes

- **check_group:**

  Check the group.

  Value: yes

- **check_perm:**

  Check the permissions.

  Value: yes

- **realtime:**

  Enable realtime monitoring of files on Linux and Windows systems.

  Value: yes

- **report_changes:**

  Report changes to a file with diffs of the changes made. The diffs could include sensitive information. This option is limited to text files only.

  Value: yes

- **restrict:**

  A string that will limit the checks to files containing the specified string in the file name.

  Value: Any directory or file name (but not a path)

- **no_recurse:**

  New in version 3.2.

  Do not recurse into the defined directory. *yes* unintuitively disables recursion.

  Value: yes

ignore

List of files or directories to be ignored. One entry per instance of *ignore*. Multiple instances can be defined.

Attributes:

- **type:**

This a simple regex pattern to filter out files.

Value: sregex

Allowed: Any directory or file name.

frequency

The frequency at which full system scans will be performed (in seconds). Frequency should be no lower than 300 seconds, and larger for larger groups of files.

Default: 21600

Allowed: Time in seconds between scans.

nodiff

New in version 3.0.

List of files where track changes will be disabled. This option can be used to not send diffs for sensitive files.

Attributes:

type: This is a simple regex pattern to filter out files from the diff function.

Value: sregex

scantime

Time to run the scans (can be in the formats of 21pm, 8:30, 12am, etc).

Allowed: Time to run the scan

Note: This may delay the initialization of realtime scans.

scan_day

Day of the week to run the scans.

Allowed: Day of the week

auto_ignore

After 3 changes to a file, further changes to a file will be ignored. Setting *auto_ignore* to *no* will disable this.

Default: yes

Allowed: yes/no

Valid on: Server and Local installations

alert_new_files

By default syscheck will not trigger an alert on file creation. Setting *alert_new_files* to *yes* will make it do so.

Default: no

Allowed: yes/no

Valid on: Server and Local installations

Note: New files will only be detected on a full scan, this option does not affect realtime operations.

scan_on_start

Controls whether *ossec-syscheckd* will perform a full scan when it is started.

Default: yes

Allowed: yes/no

windows_registry

Specifies entries to be monitored in the Windows registry. One entry per instance of this options. Multiple instances can be used.

Allowed: Any registry path

Note: New entries will not trigger alerts, only changes to existing entries.

registry_ignore

List of registry entries to be ignore. One entry per instance of this option. Multiple instances can be specified.

Allowed: Any registry path

prefilter_cmd

Command to run to prevent prelinking from creating false positives.

Example:

```
<prefilter_cmd>/usr/sbin/prelink -y</prefilter_cmd>
```

Note: This option can negatively impact performance. The configured command will be run for each file checked.

skip_nfs

New in version 2.9.0.

This option will prevent *ossec-syscheckd* from scanning network mounted filesystems. This option is only valid on Linux, FreeBSD, and OpenBSD (added in v3.3) systems. Currently *skip_nfs* will abort checks running on files stored on CIFS and NFS mount points.

Default: no

Allowed: yes/no

3.1.5 rootcheck

Settings controlling the rootcheck functionality in OSSEC.

base_directory

The base path prepended to a number of options. Usually the installation path for OSSEC. It will modify the folling options:

- rootkit_files
- rootkit_trojans
- windows_malware
- windows_audit
- windows_apps
- systems_audit

Allowed: Directory path

Default: */var/ossec*

rootkit_files

Specifies the location of the rootkit files database.

Default: */etc/shared/rootkit_files.txt*

Allowed: A file with the rootkit files signatures

rootkit_trojans

Specifies the location of the rootkit trojans database.

Default: */etc/shared/rootkit_trojans.txt*

Allowed: A file with the trojans signatures

windows_audit

Specifies the location of the Windows audit database.

Default: *./shared/win_audit_rcl.txt*

Allowed: A file with the Windows audit signatures

system_audit

Specifies the location of the system audit database. One entry per instance of this option. This option can be specified multiple times.

Default: */etc/shared/system_audit_rcl.txt*, */etc/shared/cis_debian_linux_rcl.txt*, */etc/shared/cis_rhel_linux_rcl.txt*, */etc/shared/cis_rhel5_linux_rcl.txt*

Allowed: A file with system audit signatures

windows_apps

Specifies the location of the Windows application audit database.

Default: *./shared/win_applications_rcl.txt*

Allowed: A file with Windows application signatures

windows_malware

Specifies the location of the Windows malware database.

Default: *./shared/win_malware_rcl.txt*

Allowed: A file with Windows malware audit database.

scanall

Can force rootcheck to scan the entire system. This may trigger false positives.

Default: no

Allowed: yes/no

Note: XXX More information needed.

frequency

Frequency (in seconds) between rootcheck scans.

Default: 36000

Allowed: Time in seconds

disabled

Disable the rootcheck functionality.

Default: no

Allowed: yes/no

check_dev

Enable or disable checking for files in */dev*.

Default: yes

Allowed: yes/no

check_files

Enable or disable checks based on the rootkit files.

Default: yes

Allowed: yes/no

Note: XXX Need to research.

check_if

Enable or disable checking the network interfaces.

Default: yes

Allowed: yes/no

check_pids

Enable or disable checking process IDs.

Default: yes

Allowed: yes/no

check_ports

Enable or disable checking network ports.

Default: yes

Allowed: yes/no

check_sys

Enable or disable checking the filesystem.

Default: yes

Allowed: yes/no

Note: XXX More info needed

check_trojans

Enable or disable checking for trojans.

Default: yes

Allowed: yes/no

check_unixaudit

Enable or disable checking for unix issues.

Default: yes

Allowed: yes/no

check_winapps

Enable or disable checking Windows applications.

Default: yes

Allowed: yes/no

check_winaudit

Enable or disable checking Windows audit.

Default: yes

Allowed: yes/no

check_winmalware

Enable or disable checking Windows malware.

Default: yes

Allowed: yes/no

skip_nfs

New in version 2.9.0.

If enabled, this options prevents rootcheck from scanning network filesystems. Currently works on Linux, FreeBSD, and OpenBSD (support added in v3.3). If enabled, it will abort checks running on CIFS and NFS mounts.

Default: no

Allowed: yes/no

3.1.6 localfile

Settings specifying the location and format of logs for ingestion.

location

Specifies the location of a log file for ingestion.

Default: Multiple

Allowed: Path to a log file

log_format

Specifies the format of the log file. OSSEC assumes the logs are in the default format and have not been customized.

Allowed:

- syslog

  syslog is used for plain text files with one log message per line. The log messages do not have to be in a syslog format.

- snort-full

  Snort's full text output format.

- snort-fast

  Snort's fast text output format.

- squid

  Squid's default log format.

- iis

  Microsoft IIS log format.

- eventlog

  Microsoft Windows eventlog format.

- eventchannel

  New in version 2.8.

  Microsoft Windows eventlog format, using the EventApi. This should allow OSSEC to monitor both "Windows" eventlogs and the more recent "Applications and Services" logs.

- mysql_log

  MySQL's log format.

- postgresql_log

  Postgresql's log format.

- nmapg

  Nmap's grepable log format.

- apache

  Apache's default log format.

- command

  This log format will run a command (as root). Each line of the output will be treated as a separate log.

  Warning: Cannot be specified in the *agent.conf*

- full_command

  This log format rill run a command as root, and treat the entire output as a single log message.

  Warning: Cannot be specified in the *agent.conf*

- djb-multilog

  Daniel J. Bernstein's multilog output.

- multi-line

  This format type is for log messages consisting of multiple lines. The number of lines used per message should be the same, and the number of lines should be specified in the format:

  ```
  <log_format>multi-line: 3</log_format>
  ```

- multi-line_indented

  This log format accepts logs spanning multiple lines with subsequent lines beginning with either a space or tab.

  ```
  <log_format>multi-line_indented</log_format>
  ```

command

The command to be run when using ithe *command* or *full_command log_format*.

Allowed: Any command with arguments

command_alias

An alias for a command to help with identification. This alias will replace the command in the log output.

Example:

The following alias:

```
<alias>usb-check</alias>
```

would change the output from:

```
ossec: output: 'reg QUERY HKLM\SYSTEM\CurrentControlSet\Enum\USBSTOR
↪':
```

to:

```
ossec: output: 'usb-check':
```

Allowed: Any string

check_diff

Store the output of an event in an internal database, and compare new output to the previous. If the output has changed an alert will be triggered.

only-future-events

Only read log messages starting from the time the log is opened. By default *ossec-logcollector* will read events from the beginning of the log or from the message it last read.

Allowed: yes

Note: This only applies to *eventchannel* log sources.

query

Specifies an *XPATH* query following the event schema in order to filter the events OSSEC will process. See Microsoft's documentation for more details.

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>
```

Note: This only applies to *eventchannel* log sources.

3.1.7 rules

Settings specifying rule locations. These settings are only valid on a Server or Local installation.

include

Load a single rule file. The rule files are stored in *rules/*, relative to the install location of OSSEC.

Allowed: File name of a rule file.

rule

Load a single rule file.

Note: This is an alias for *include*.

rule_dir

Specifies a directory containing rule files. The rule files will be loaded alphebetically.

Attributes:

- **pattern:** is a regex match string used to filter the files in the directory.

  Default: Regex *_rules.xml$* is used if no other pattern is specified.

Allowed: Path to a directory of rule files, relative to the OSSEC installation location.

decoder

Specifies the path to a decoder file to be used by *ossec-analysisd*. If no decoders are specified in the *ossec.conf* the default *etc/decoder.xml* and *etc/local_decoder.xml* are used. If a decoder is specified with *decoder* or *decoder_dir* the default *decoder.xml* and *local_decoder.xml* will not be used.

Allowed: Path to a decoder file relative to OSSEC's install location.

decoder_dir

Specifies the path to a directory containing decoder files. Files will be loaded in alphebetical order.

Attributes:

- **pattern:** is a regex match string used to filter the files in the directory.

  Default: Regex *.xml$* is used if no other pattern is specified.

Allowed: Path to a directory of rule files, relative to the OSSEC installation location.

list

Specifies a single cdb reference for inclusion by other rules. File extensions should not be included.

Example:

For a cdb list named *blocked_hosts.txt* use:

```
<list>rules/lists/blocked_hosts</list>
```

3.1.8 command

Definitions for commands available to the active response system.

name

Specifies the name of the command.

executable

Specifies the executable to be run. The executable must be a file (with exec permissions) inside *active-response/bin* relative to the OSSEC install location.

expect

Specifies the information gathered from a log via decoding to send to the command. Common options are *srcip* and *username*.

timeout_allowed

Specifies if the command supports a timeout. After the timeout period the command will attempt to reverse changes it has made (delete firewall rules or entries in *hosts.deny*).

3.1.9 active-response

disabled

Disables active response if set to *yes*. Active response defaults to enabled on Unix-like systems and disabled on Windows. Setting *disabled* to *yes* on an OSSEC management server will disable all active response. Disabling it on an agent will only disable it for that agent.

command

Specifies the command to be run when the active response is triggered.

location

Specifies where the active response will be run.

Available options:

- local: the agent that generated the event
- server: the OSSEC management server
- defined-agent: on a specific agent
- all: all agents

agent_id

Specifies the agent ID to run the active response on when using *defined-agent* as the *location*.

level

The active response will be executed on any event of the specified level or higher.

rules_group

The active response will be executed on any alert with the specified group. Multiple groups can be defined, separated with the pipe (|) character.

rules_id

The active response will be executed on any alert with the specified rule ID. Multiple IDs can be specified separated by a comma.

timeout

Specifies the amount of time, in seconds, before an active response will attempt to reverse its actions. For example IPs can be unblocked or entries removed from the *hosts.deny* file.

repeated_offenders

A comma separated list or increasing timeouts (in minutes) for repeat offenders. There can be a maximum of 5 entries.

Valid on: Agent and Local

3.1.10 alerts

Settings related to alert logging and notifications.

The <*alerts*> section is valid on Server and Local installations only.

email_alert_level

Minimum alert level for email notification.

Default: 7

Allowed: Any level from 1-16

Note: This is the minimm level for an alert to trigger an email. This will over ride granular email alert levels. Individual rules can over ride this with the *alert_by_email* option.

log_alert_level

Minimum alert level to record an alert in *alerts.log*.

Default: 1

Allowed: Any level from 1-16

use_geoip

Enable or disable GeoIP lookups.

Default: no

Allowed: yes/no

3.1.11 email_alerts

Settings for which alerts should trigger emails.

email_to

The recipient address of alert emails.

Allowed: A valid email address

level

The minimum alert level for an email.

Allowed: Any alert level 0-16

Note: *level* should be set to or above teh *email_alert_level* in the *alerts* section of the configuration.

group

Specifies a group that an alert must match for an email to be sent. Multiple groups can be specified with a pipe (|) character.

Allowed: One or more groups

event_location

An alert must match this *event_location* for an email to be sent. Only one *event_location* may be specified, the last entry will be used.

Allowed: Any single agent name, hostname, ip address, or log file

format

Specifies the format of the email.

Options:

- full: normal emails

rule_id

Option to send granular emails based on the rule id.

Allowed: One or more rule IDs, separated by a comma and a space.

do_not_delay

Option to send the email immediately.

Example:

```
<do_not_delay />
```

do_not_group

Option to not group alerts in an email.

Example:

```
<do_not_group />
```

3.1.12 syslog_output

OSSEC is able to forward alerts via syslog in a number of formats. These options are for use on a Server or Local installation.

server

Specifies the IP address of the syslog server.

port

Specifies the port of the syslog server.

level

Specifies the minimum alert level for alerts to be forwarded.

group

Alerts belonging to the configured group will be forwarded. Multiple groups can be specified, separated with the pipe (|) character.

rule_id

Alerts matching the configured rule ID will be forwarded.

location

Alerts from this location will be forwarded.

use_fqdn

By default OSSEC truncates the hostname at the first period (.) when generating syslog messages. Setting this option to *yes* will force it to use the full hostname of the server.

Default: no

Allowed: yes/no

format

ossec-csyslogd can send alerts in multiple formats. The default format is a standard syslog message.

Available options:

- **default:** Default syslog messages
- **CEF:** ArcSight Common Event Format
- **json:** JSON
- **splunk:** A format for use with Splunk

3.1.13 database_output

OSSEC can store alerts in MySQL or PostgreSQL databases. These options are for use on a Server or Local installation.

Note: OSSEC must be compiled with database support for *ossec-dbd* to function.

hostname

IP address of the database server.

username

The username for accessing the database.

password

The password for the user connecting to the database.

database

The database where alerts will be stored.

type

The type of database: MySQL or PostgreSQL.

3.1.14 agentless

Configures scripts to be run against systems where an agent cannot be installed.

type

The type of check to be run on the agentless system.

Available options:

- **ssh_integrity_check_bsd:** Perform a file integrity check via ssh on BSD
- **ssh_integrity_check_linux:** Perform a file integrity check via ssh on Linux
- **ssh_generic_diff:** Run a command and compare the output to previous command runs
- **ssh_pixconfig_diff:** Compare the current running Cisco PIX configuration to previous invocations

frequency

Specifies the time in seconds between each check.

host

Specifies the username and host to be checked.

Example:

```
<host>root@linux.server.example.com</host>
```

state

Specifies whether the checks are periodic or periodic_diff.

Available options:

- **periodic:** The output from the script is processes by the OSSEC management server
- **periodic_diff:** The output of the script is compared to previous invocations

arguments

Specifies the arguments passed to the script.

3.1.15 reports

Configuration for automated daily reports. This is only valid on Server and Local installations.

group

Filter by group or category.

categories

Filter by group or category. This is the same as the group option above.

rule

Filter on a specific Rule ID.

level

Filter by minimum rule level.

location

Filter by the log location or agent name.

srcip

Filter by the source ip recorded in an alert.

user

Filter by the user name in an alert. This will match both *srcuser* and *dstuser*.

title

Specify the title of the report. This is a required option.

email_to

The email address the completed report is sent to. This is a required option.

showlogs

If *showlogs* is set to *yes* the relevant logs will be included in the report. This is set to *no* by default.

3.2 agent.conf

The file */var/ossec/etc/shared/agent.conf* can be used for centralized configuration. The *agent.conf* file (and the contents of */var/ossec/etc/shared*) will be pushed from the OSSEC server to the agents.

Note: The server itself does not read the configuration in the *agent.conf*. This file is only for use by the agents.

Many of the configuration options in the *ossec.conf* can be placed in teh agent.conf. If the *agent.conf* is modified, the agent will have to be restarted for the changes to take effect.

3.2.1 Syntax

Each block of configuration should be surrounded by <*agent_config*>. There can be multiple <*agent_config*> blocks in the *agent.conf*.

```
<agent_config>

  <!-- Configuration goes here -->

</agent_config>
```

Each block can be restricted with 3 options. The options are *os* to limit by operating system, *name* to limit by the agent name, and *profile* to use agent profiles. Every agent that matches the restriction will apply the configuration block. Every agent will apply a configuration block that does not contain a restriction.

3.2.2 os

The general operating system can be specified with *os*. Typical options include *Windows* and *Linux*.

```
<agent_config os="Windows">
```

3.2.3 name

An agent name can be specified with *name*. This is the name the agent is given when it is added with manage_agents or authd.

```
<agent_config name="agent007">
```

3.2.4 profile

A profile is a group of similar agents that can agent can subscribe to in its *ossec.conf*.

```
<agent_config profile="webservers">
```

3.2.5 Minimal ossec.conf

At a minimum, the *<client>* section is necessary. Without this, the agent does not know how to communicate with the server.

Minimal *ossec.conf* example:

```
<ossec_config>
  <client>
    <server-hostname>ossec.example.com</server-hostname>
  </client>
```

3.2.6 Settings that will not work

By default the OSSEC agent will not run commands from an *agent.conf*. Any *<localfile>* options that require *command* or *full_command* should be configured in the agent's *ossec.conf*.

Warning: There are other configuration options that will not work in the *agent.conf*, but there is not an up to date list of which options these are. Any help making this list would be appreciated.

3.2.7 Examples

Adding a syscheck configuration to Linux agents

The directories option to monitor */var/www/html* on Linux agents can be added to the *agent.conf*.

```
<agent_config os="Linux">
  <directories check_all="yes" realtime="yes">/var/www/html</directories>
</agent_config>
```

CHAPTER 4

Documentation

4.1 Current documentation

This documentation is mostly a re-write of the original documentation. It is currently a work in progress, and incomplete.

4.2 Legacy documentation

The original OSSEC documentation is included under the Legacy Documentation. As parts of it are replicated in the new repository, the old parts will be deleted.

4.3 Helping out

If you are interested in assisting with the documentation, please jump right in. The current repository for the documentation is on github. Feel free to submit pull requests and open issues.

Issues and pull requests can be opened for everything from spelling mistakes to helping a section make more sense. What makes sense to an OSSEC veteran migth not make sense to someone using the software for the first time.

CHAPTER 5

Legacy Documentation

5.1 Legacy Documentation

Warning: This documentation is no longer maintained, and will be removed as the non-legacy documentation is updated.

5.1.1 Manual & FAQ

Manual

Getting started with OSSEC

OSSEC is a platform to monitor and control your systems. It mixes together all the aspects of HIDS (host-based intrusion detection), log monitoring, and Security Incident Management (SIM)/Security Information and Event Management (SIEM) together in a simple, powerful, and open source solution.

Key Benefits

Compliance Requirements

OSSEC helps customers meet specific compliance requirements such as PCI and HIPAA. It lets customers detect and alert on unauthorized file system modifications and malicious behavior embedded in the log files of commercial products as well as custom applications. For PCI, it covers the sections of file integrity monitoring (PCI 11.5, 10.5), log inspection and monitoring (section 10), and policy enforcement/checking.

Multi platform

OSSEC lets customers implement a comprehensive host based intrusion detection system with fine grained application/server specific policies across multiple platforms such as Linux, Solaris, Windows, and Mac OS X.

Real-time and Configurable Alerts

OSSEC lets customers configure incidents they want to be alerted on, and lets them focus on raising the priority of critical incidents over the regular noise on any system. Integration with smtp, sms, and syslog allows customers to be on top of alerts by sending them to e-mail enabled devices. Active response options to block an attack immediately are also available.

Integration with current infrastructure

OSSEC will integrate with current investments from customers such as SIM/SEM (Security Incident Management/Security Events Management) products for centralized reporting and correlation of events.

Centralized management

OSSEC provides a simplified centralized management server to manage policies across multiple operating systems. Additionally, it also lets customers define server specific overrides for finer grained policies.

Agent and agentless monitoring

OSSEC offers the flexibility of agent based and agentless monitoring of systems and networking components such as routers and firewalls. Agentless monitoring lets customers who have restrictions on software being installed on systems (such as FDA approved systems or appliances) meet security and compliance needs.

Key Features

File Integrity checking

There is one thing in common to any attack to your networks and computers: they change your systems in some way. The goal of file integrity checking (or FIM - file integrity monitoring) is to detect these changes and alert you when they happen. It can be an attack, or a misuse by an employee or even a typo by an admin, any file, directory or registry change will be alerted to you.

Covers PCI DSS sections 11.5 and 10.5.5.

Log Monitoring

Your operating system wants to speak to you, but do you know how to listen? Every operating system, application, and device on your network generate logs (events) to let you know what is happening. OSSEC collects, analyzes and correlates these logs to let you know if something suspicious is happening (attack, misuse, errors, etc). Do you want to know when an application is installed on your client box? Or when someone changes a rule in your firewall? By monitoring your logs, OSSEC will notify you.

This should cover PCI DSS section 10.

Rootkit detection

Criminal hackers want to hide their actions, but using rootkit detection you can be notified when the system is modified in a way common to rootkits.

Active response

Active response allows OSSEC to take immediate action when specified alerts are triggered. This may prevent an incident from spreading before an administrator can take action.

OSSEC Architecture

OSSEC is composed of multiple pieces. It has a central manager for monitoring and receiving information from agents, syslog, databases, and from agentless devices.

Manager (or Server)

The manager is the central piece of the OSSEC deployment. It stores the file integrity checking databases, the logs, events, and system auditing entries. All the rules, decoders, and major configuration options are stored centrally in the manager; making it easy to administer even a large number of agents.

Agents connect to the server on port 1514/udp. Communication to this port must be allowed for agents to communicate with the server.

Note: The manager may be called the OSSEC server, or even just server in this documentation.

Agents

The agent is a small program, or collection of programs, installed on the systems to be monitored. The agent will collect information and forward it to the manager for analysis and correlation. Some information is collected in real time, others periodically. It has a very small memory and CPU footprint by default, not affecting the system's usage.

Agent security: It runs with a low privilege user (generally created during the installation) and inside a chroot jail isolated from the system. Most of the agent configuration can be pushed from the manager.

Note: OSSEC can only be installed as an agent on Microsoft Windows platforms. These systems will require an OSSEC server, running on Linux or another unix-like system.

Agentless

For systems that an agent cannot be installed on, the agentless support may allow integrity checks to be performed. Agentless scans can be used to monitor firewalls, routers, and even Unix systems.

Virtualization/VMware

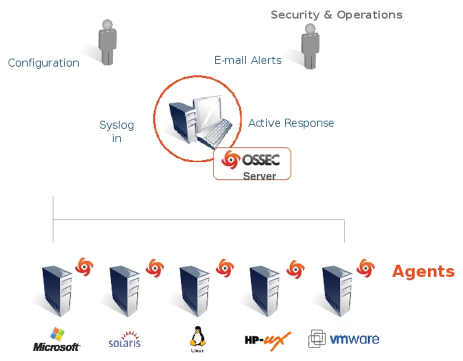
OSSEC allows you to install the agent on the guest operating systems. It may also be installed inside some versions of VMWare ESX, but this may cause support issues. With the agent installed inside VMware ESX you can get alerts about when a VM guest is being installed, removed, started, etc. It also monitors logins, logouts and errors inside the ESX server. In addition to that, OSSEC performs the Center for Internet Security (CIS) checks for VMware, alerting if there is any insecure configuration option enabled or any other issue.

Firewalls, switches and routers

OSSEC can receive and analyze syslog events from a large variety of firewalls, switches and routers. It supports all Cisco routers, Cisco PIX, Cisco FWSM, Cisco ASA, Juniper Routers, Netscreen firewall, Checkpoint and many others.

Architecture

This diagram shows the central manager receiving events from the agents and system logs from remote devices. When something is detected, active responses can be executed and the admin is notified.



Supported Systems

OSSEC supports the following operating systems and log formats.

Operating Systems

The following operating systems are supported by the OSSEC agent:

- GNU/Linux (all distributions, including RHEL, Ubuntu, Slackware, Debian, etc)
- Windows XP, 2003, Vista, 2008, 2012
- VMWare ESX 3.0,3.5 (including CIS checks)
- FreeBSD (all current versions)
- OpenBSD (all current versions)
- NetBSD (all current versions)

- Solaris 2.7, 2.8, 2.9 and 10
- AIX 5.2 and 5.3
- Mac OS X 10.x
- HP-UX 11

Devices supported via Syslog

These systems/devices are also supported via remote syslog:

- Cisco PIX, ASA and FWSM (all versions)
- Cisco IOS routers (all versions)
- Juniper Netscreen (all versions)
- SonicWall firewall (all versions)
- Checkpoint firewall (all versions)
- Cisco IOS IDS/IPS module (all versions)
- Sourcefire (Snort) IDS/IPS (all versions)
- Dragon NIDS (all versions)
- Checkpoint Smart Defense (all versions)
- McAfee VirusScan Enterprise (v8 and v8.5)
- Bluecoat proxy (all versions)
- Cisco VPN concentrators (all versions)
- VMWare ESXi 4.x

Devices and Operating Systems via Agentless

Using OSSEC agentless options, the following systems are also supported (for log analysis and file integrity checking):

- Cisco PIX, ASA and FWSM (all versions)
- Cisco IOS routers (all versions)
- Juniper Netscreen (all versions)
- SonicWall firewall (all versions)
- Checkpoint firewall (all versions)
- All operating systems specified in the "operating systems" section

Installation

The best installation tutorial is available in the OSSEC book.

Installations requirements

For UNIX systems, OSSEC only requires gnu make, gcc, and libc. OpenSSL is a suggested, but optional, prerequisite. However, you always have the option to pre-compile it on one system and move the binaries to the final box.

PCRE2

New in version 3.3.

PCRE2 has been added to version 3.3. The build system can either use the system's PCRE2 libraries, or the necessary bits can be built as part of the installation process.

The default build process expects the *pcre2-10.32* source to be installed in *src/external*:

```
$ cd ossec-hids-*/src
$ wget https://ftp.pcre.org/pub/pcre/pcre2-10.32.tar.gz
$ tar xzf pcre2-10.32.tar.gz -C src/external
```

To use the system's PCRE2, set the *PCRE2_SYSTEM* variable to yes:

```
# cd ossec-hids-*
# PCRE2_SYSTEM=yes ./install.sh
```

zlib

zlib is included with OSSEC in *src/external/zlib-1.2.11*. In previous versions this included version was used by default during the build process, but this changed to using the system zlib. Ensure the correct zlib development packages are installed.

To use the included version of zlib, simply set *ZLIB_SYSTEM* to *no*:

```
# cd ossec-hids-*
# ZLIB_SYSTEM=no ./install.sh
```

Ubuntu

On Ubuntu you will need the *build-essential* package in order to compile and install OSSEC.

To install the package run the following command.

```
# apt-get install build-essential zlib1g-dev
```

To use the system's pcre2 libraries, install the libpcre2 development package:

```
# apt-get install libpcre2-dev
```

If database support is needed *mysql-dev* or *postgresql-dev* should be installed. Run the following command to install these packages.

```
# apt-get install mysql-dev postgresql-dev
```

To use the SQLite features, the *libsqlite3-dev* package is necessary.

New in version 3.0.

```
# apt-get install libsqlite3-dev
```

RedHat

RedHat should have most of the packages needed by default. The zlib development package should be installed:

```
# yum install zlib-devel
```

To use the system's pcre2 libraries, add the pcre2 development package:

```
# yum install pcre2-devel
```

If database support is needed the package mysql-devel and/or postgresql-devel will need to be installed.

```
# yum install mysql-devel postgresql-devel
```

To use the SQLite features, the *sqlite-devel* package is necessary.

New in version 3.0.

```
# yum install sqlite-devel
```

OpenSuse

The zlib development package should be installed:

```
# zypper install zlib-devel
```

To use the system's pcre2 libraries, add the pcre2 development package:

```
# zypper install pcre2-devel
```

If database support is needed the package mysql-devel and/or postgresql-devel will need to be installed.

```
# zypper install postgresql-devel mysql-devel
```

Debian

Warning: The Debian instructions are probably out of date. Contributions updating this section would be appreciated.

Debian has replaced bash with dash, and this may cause issues during installation. Dash does not appear to support all of the features available in other shells, and may display an error when trying to set the server's IP address on an agent system. The error can be ignored, but the server ip address will need to be set.

Do this by making sure something like the following information is in the agent's ossec.conf:

```
<ossec_config>
  <client>
    <server-ip>SERVER'S IP</server-ip>
  </client>
```

This can also be avoided by using bash to run `install.sh`:

```
# bash ./install.sh
```

Manager/Agent Installation

Installation of OSSEC HIDS is very simple, the `install.sh` shell script automating most of it. There are a few questions to be answered before the installation will occur, one of the most important being which type of installation is desired. It is important to choose the correct installation type: server, agent, local, or hybrid. More information on them can be found on the OSSEC Architecture page.

Note: In the following installation the commands follow the `#`. Everything else is either comments or output.

1. Download the latest version and verify its checksum.

   Note: On some systems, the command md5, sha1, or wget may not exist. Try md5sum, sha1sum or lynx respectively instead.

   Warning: wget may not be able to pull files from the OSSEC site. Use the `-U` flag to add a UserAgent, or obtain the checksum file by some other manner.

   ```
   # wget -U ossec https://bintray.com/artifact/download/ossec/ossec-hids/ossec-
   ↪hids-2.8.3.tar.gz
   # wget -U ossec https://raw.githubusercontent.com/ossec/ossec-docs/master/
   ↪docs/whatsnew/checksums/2.8.3/ossec-hids-2.8.3.tar.gz.sha256
   # cat ossec-hids-2.8.3.tar.gz.sha256
   SHA256 (ossec-hids-2.8.3.tar.gz) =␣
   ↪917989e23330d18b0d900e8722392cdbe4f17364a547508742c0fd005a1df7dd
   # sha256sum -c  ossec-hids-2.8.3.tar.gz.sha256 ossec-hids-2.8.3.tar.gz
   (SHA256) ossec-hids-2.8.3.tar.gz: OK
   ```

2. Extract the compressed package and run the `install.sh` script. It will guide you through the installation and compile the source (not shown).

   ```
   # tar -zxvf ossec-hids-*.tar.gz (or gunzip -d; tar -xvf)
   # cd ossec-hids-*
   # ./install.sh
   ```

3. The OSSEC manager listens on UDP port 1514. Any firewall sbetween the agents and the manager will need to allow this traffic.

4. The server, agent, and hybrid installations will require additional configuration. More information can be found on the Managing the agents page.

5. Start OSSEC HIDS by running the following command:

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

Manual Installation

OSSEC can also be installed in a more manual fashion. No modifications will be made to the *ossec.conf* file, so it will have to be configured after installation. The *ossec*, *ossecm* and *ossecr* users will still be created automatically.

After the source tarball is downloaded and extracted:

```
cd ossec-hids-*/src
make TARGET=<server|local|agent>
make install
```

Build options can still be passed to *make* (*USE_ZEROMQ*, *USE_GEOIP*, etc.).

Windows Agent Installation

Note: OSSEC only supports Windows systems as agents, and they will require an OSSEC server to function.

Step 1: Opening the Agent Manager menu

The first step of this process is to get into the Agent Manager menu. From the ossec server, type the following command:

```
/var/ossec/bin/manage_agents
```

The menu should look like this:

```
****************************************
* OSSEC HIDS v2.5-SNP-100809 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:
```

Step 2: Adding an Agent

Adding an agent is the first thing that needs to be done. Choose action "A".

The agent manager first asks for a name. This can be named anything.

```
Adding a new agent (use '\q' to return to the main menu).
Please provide the following:   * A name for the new agent:
```

Next, it asks for the IP address of the windows client.

```
The IP Address of the new agent:
```

After that, it asks for a unique ID to assign to the client. The ID must be all numerical with a maximum of eight digits. The agent manager also suggests ID's for new agents.

```
An ID for the new agent[001]:
```

Lastly, it asks for confirmation of all the information provided. Then it appends all of the agent information to /var/ossec/etc/client.keys and returns to the main menu.

Step 3: Extracting a Key

Now, the client key needs to be extracted. From the main menu, choose action "E". A list of agents will be displayed:

```
Available agents:   ID: 001, Name: agent1, IP: 10.10.50.2
Provide the ID of the agent to extract the key (or '\q' to quit):
```

Enter the full ID of the agent to extract the key for. It will display the entire key. Copy that to the clipboard, for it will be needed later.

```
Agent key information for '001'␣
↪is:MDAyIGFnZW50MSAxOTIuMTY4LjIuMC8yNCBlNmY3N2RiMTdmMTJjZGRmZjg5YzA4ZDk5m
```

Step 4: The Windows Side

Next up, download the executable named Agent Windows from https://ossec.github.io/downloads.html. Run through the install wizard with all defaults. It should launch the Ossec Agent Manager when it's done. The Ossec Agent Manager looks like this:

Enter the IP address of your ossec server in the first text field, and enter the extracted key that was copied to the clipboard earlier to the second textfield. Finally, click on the manage tab and hit restart.

Package Installation

The OSSEC project has made RPM and deb packages available. Links to the packages can be found on the OSSEC download page

RPM Installation

OSSEC's RPMs are made available by AtomiCorp.

The RPMs can be installed by adding the AtomiCorp yum repository:

```
# wget -q -O - https://updates.atomicorp.com/installers/atomic | sh
```

Next use `yum` to install the specific packages. For an OSSEC server run:

```
# yum install ossec-hids ossec-hids-server
```

And for an agent run:

```
# yum install ossec-hids ossec-hids-agent
```

Deb Installation

OSSEC's deb packages are available in the Wazuh repository.

Install the apt-get repository key:

```
# apt-key adv --fetch-keys http://ossec.wazuh.com/repos/apt/conf/ossec-key.gpg.key
```

Add the repository for Debian (available distributions are Sid, Jessie and Wheezy):

```
# echo 'deb http://ossec.wazuh.com/repos/apt/debian wheezy main' >> /etc/apt/sources.
↪list
```

Or add the repository for Ubuntu (available distributions are Precise, Trusty and Utopic):

```
# echo 'deb http://ossec.wazuh.com/repos/apt/ubuntu precise main' >> /etc/apt/sources.
↪list
```

Update the repository:

Install OSSEC HIDS server/manager:

```
# apt-get install ossec-hids
```

Or install OSSEC HIDS agent:

```
# apt-get install ossec-hids-agent
```

Compiling OSSEC for a Binary Installation

OSSEC is typically compiled on each system it is installed on, but this may not always be easy. To help in these cases there are a few methods of binary installation available. OSSEC can be compiled on one system, and copied to the destination systems. This installation method still requires GNU make on the target system.

There are also RPM and deb packages available for some systems.

Note: OSSEC has very limited cross compiling facilities. Windows binaries can be built on Linux systems, but binaries for other systems should be built on a system of the same OS and CPU platform.

Compiling OSSEC for install on a second server

First download the OSSEC package corresponding to the version you want to install and unpack it (on the system with a compiler).

```
# wget -U ossec http://www.ossec.net/files/ossec-hids-2.8.1.tar.gz
# tar -zxvf ossec-hids-latest.tar.gz
```

Enter in the source directory of the downloaded package, and configure OSSEC to build the `agent` version. The `make` commands should compile the correct binaries.

```
# cd ossec-*/src
# make setagent
# make all
# make build
```

Modify ossec-hids-*/etc/preloaded-vars.conf to set BINARY_INSTALL to yes.

```
# echo "USER_BINARYINSTALL=\"y\"" >> ossec-hids*/etc/preloaded-vars.conf
```

Finally create an OSSEC package.

```
# tar -cvf ossec-binary.tar ossec-hids*
```

Installation of the binary OSSEC package

On the target system (that does not have a C compiler) download your ossec-binary.tar created in the steps above.

Complete the installation by unarchiving the binary package and running ./install.sh.

```
# tar xfv ossec-binary.tar
# cd ossec-*
# ./install.sh
```

After following the installation prompts the install will be complete.

Server Virtual Appliance Installation

Overview:

The OSSEC virtual appliance is a virtual system in the Open Virtualized Format (OVF). It contains an OSSEC 2.7 server installation and the WebUI (0.8 Beta).

Accounts and passwords:

The default password for all accounts on the system is `_0ssec_`. The username from the WebUI is `user`, and for phpMyAdmin it is `root`.

Convert OVF to a VMWare image:

Some VMWare desktop environments may not support the OVF images natively, for those systems VMWare created the ovftool. Download the ovftool from VMWare's site (registration required).

Convert the file using the following procedure:

```
# tar zxvf ossec_virtual_appliance.tar.gz
# cd ossec_virtual_appliance
# ovftool ossec.ovf ossec.vmx
```

Unattended Source Installation

OSSEC has the capability to be compiled and installed without the interactivity of `install.sh`. The install script can collect the answers to its questions from the `etc/preloaded-vars.conf` configuration file.

Most of the questions asked by the installer are present in the configuration file, along with the default answers. Uncommenting each variable will allow the script to know the answer. Any changes from the default install should be made in the configuration file.

Note: If `USER_NO_STOP="y"` is not set, install.sh may prompt for confirmation.

Example preloaded-vars.conf:

```
# preloaded-vars.conf, Daniel B. Cid (dcid @ ossec.net).
#
# Use this file to customize your installations.
# It will make the install.sh script pre-load some
# specific options to make it run automatically
# or with less questions.

# PLEASE NOTE:
# When we use "n" or "y" in here, it should be changed
# to "n" or "y" in the language your are doing the
# installation. For example, in portuguese it would
# be "s" or "n".


# USER_LANGUAGE defines to language to be used.
# It can be "en", "br", "tr", "it", "de" or "pl".
# In case of an invalid language, it will default
# to English "en"
#USER_LANGUAGE="en"     # For english
#USER_LANGUAGE="br"     # For portuguese


# If USER_NO_STOP is set to anything, the confirmation
# messages are not going to be asked.
#USER_NO_STOP="y"


# USER_INSTALL_TYPE defines the installation type to
# be used during install. It can only be "local",
# "agent" or "server".
#USER_INSTALL_TYPE="local"
#USER_INSTALL_TYPE="agent"
#USER_INSTALL_TYPE="server"


# USER_DIR defines the location to install ossec
#USER_DIR="/var/ossec"


# If USER_DELETE_DIR is set to "y", the directory
# to install OSSEC will be removed if present.
#USER_DELETE_DIR="y"
```

(continues on next page)

(continued from previous page)

```
# If USER_ENABLE_ACTIVE_RESPONSE is set to "n",
# active response will be disabled.
#USER_ENABLE_ACTIVE_RESPONSE="y"


# If USER_ENABLE_SYSCHECK is set to "y",
# syscheck will be enabled. Set to "n" to
# disable it.
#USER_ENABLE_SYSCHECK="y"


# If USER_ENABLE_ROOTCHECK is set to "y",
# rootcheck will be enabled. Set to "n" to
# disable it.
#USER_ENABLE_ROOTCHECK="y"


# If USER_UPDATE is set to anything, the update
# installation will be done.
#USER_UPDATE="y"

# If USER_UPDATE_RULES is set to anything, the
# rules will also be updated.
#USER_UPDATE_RULES="y"

# If USER_BINARYINSTALL is set, the installation
# is not going to compile the code, but use the
# binaries from ./bin/
#USER_BINARYINSTALL="x"


### Agent Installation variables. ###

# Specifies the IP address or hostname of the
# ossec server. Only used on agent installations.
# Choose only one, not both.
# USER_AGENT_SERVER_IP="1.2.3.4"
# USER_AGENT_SERVER_NAME


# USER_AGENT_CONFIG_PROFILE specifies the agent's config profile
# name. This is used to create agent.conf configuration profiles
# for this particular profile name. Only used on agent installations.
# Can be any string. E.g. LinuxDBServer or WindowsDomainController
#USER_AGENT_CONFIG_PROFILE="generic"



### Server/Local Installation variables. ###

# USER_ENABLE_EMAIL enables or disables email alerting.
#USER_ENABLE_EMAIL="y"

# USER_EMAIL_ADDRESS defines the destination e-mail of the alerts.
#USER_EMAIL_ADDRESS="dcid@test.ossec.net"
```

(continues on next page)

(continued from previous page)

```
# USER_EMAIL_SMTP defines the SMTP server to send the e-mails.
#USER_EMAIL_SMTP="test.ossec.net"


# USER_ENABLE_SYSLOG enables or disables remote syslog.
#USER_ENABLE_SYSLOG="y"


# USER_ENABLE_FIREWALL_RESPONSE enables or disables
# the firewall response.
#USER_ENABLE_FIREWALL_RESPONSE="y"


# Enable PF firewall (OpenBSD and FreeBSD only)
#USER_ENABLE_PF="y"


# PF table to use (OpenBSD and FreeBSD only).
#USER_PF_TABLE="ossec_fwtable"


# USER_WHITE_LIST is a list of IPs or networks
# that are going to be set to never be blocked.
#USER_WHITE_LIST="192.168.2.1 192.168.1.0/24"


#### exit ? ###
```

Compiling the OSSEC Windows Agent on Windows

Warning: As of 2.9 this is no longer supported. The Windows agent can be built on Linux systems. Patches to update the Windows compilation support are very welcome.

Note: Originally posted Compiling the OSSEC Windows Agent on Windows by mstarks, duplicated here with permission.

Most people that use the OSSEC Windows agent download a pre-compiled copy from the OSSEC site. While that is a good option for many individual users, it may not suit those with more specific needs and/or those in enterprise environments. Users who fall into those categories could benefit from customizing the agent and maintaining internal builds in order to suit their individual needs.

There are already instructions on how to compile the Windows agent on Linux, but ironically the process doesn't work so well on Windows. I had a need to make this work on Windows, so I thought I would share the process with you.

Requirements:

- The Nullsoft Scriptable Install System (NSIS)
- The Minimalist GNU for Windows (MinGW) compiler
- My batch file. Simply rename from gen_win.txt to gen_win.cmd.

- 7-Zip for Windows
- The public domain Unix2DOS utility
- The latest OSSEC for Unix/Linux (this contains the Windows source code)

Here are the steps:

1. Download and install the required programs. Be sure to pay special attention to the steps for properly installing and configuring MinGW, particularly the part about modifying the PATH environment variable.
2. Next, we.re going to extract OSSEC using 7-Zip. To do so, simply right-click on the file and select 7-Zip, extract to "folder name.tar," where folder name is the name of the package. This decompresses the archive. Navigate within that folder and repeat this step to untar the archive. At this point, you should see all of the files in the package.
3. Place gen_win.txt in the `src\win32` folder and rename the extension to .cmd.
4. Download Unix2DOS and place it in the `src\win32` folder
5. Open a command prompt. Navigate to `src\win32`, make any desired customizations, and execute gen_win.cmd. This should gather all of the required files and place them in `src\win-pkg`.
6. Next, we compile the Windows agent by navigating to `src\win-pkg` and executing make.bat (I assume you have the chops to know how to change directories :)).
7. Now we have all of the files we need but no way to effectively install it. To generate the installer, simply execute the NSIS compiler like so: `"c:\Program Files\NSIS\makensis.exe" ossec-installer.nsi`

If you see no errors and a binary named ossec-win32-agent.exe, everything was successful. Congratulations, you now have a custom-made version of OSSEC! OSSEC's Windows agent is compiled using MinGW

Note: This documentation assumes you have MinGW installed, and it is usable. An example of installing MinGW on CentOS is available here.

It has always been a pain to generate snapshots for Windows because it required me to open up my Windows VM (slow), push the code there, compile, etc. Well, until this week when I started to play with MinGW cross-compilation feature to completely build an Windows agent from Linux.

How it works? First, you need to install MinGW and nsis (to build the installer). For OpenSSL support, an OpenSSL MinGW package will also be necessary.

After that, download the source and generate the Windows package directory (replace 2.6 with the latest version or download the latest source from the github repository):

Note: On some systems, the command md5, sha1, or wget may not exist. Try md5sum, sha1sum or lynx respectively instead.

Warning: wget may not be able to pull files from the OSSEC site. Use the `-U` flag to add a UserAgent, or obtain the checksum file by some other manner.

```
# wget -U ossec https://bintray.com/artifact/download/ossec/ossec-hids/ossec-
↪hids-2.8.3.tar.gz
# wget -U ossec https://raw.githubusercontent.com/ossec/ossec-docs/master/
↪docs/whatsnew/checksums/2.8.3/ossec-hids-2.8.3.tar.gz.sha256
```

(continues on next page)

(continued from previous page)

```
# cat ossec-hids-2.8.3.tar.gz.sha256
SHA256 (ossec-hids-2.8.3.tar.gz) =␣
↪917989e23330d18b0d900e8722392cdbe4f17364a547508742c0fd005a1df7dd
# sha256sum -c  ossec-hids-2.8.3.tar.gz.sha256 ossec-hids-2.8.3.tar.gz
(SHA256) ossec-hids-2.8.3.tar.gz: OK
```

Compiling OSSEC 2.9 with MinGW:

Change directory to the src directory:

```
# cd ossec-hids-2.9.0/src
```

Now run `make TARGET=winagent`:

```
# make TARGET=winagent
```

This should produce a good amount of compilation output that ends with:

```
Output: "ossec-win32-agent.exe"
Install: 7 pages (448 bytes), 3 sections (3144 bytes), 769 instructions (21532 bytes),
↪ 318 strings (32350 bytes), 1 language table (346 bytes).
Uninstall: 5 pages (320 bytes),
1 section (1048 bytes), 350 instructions (9800 bytes), 184 strings (3360 bytes), 1␣
↪language table (290 bytes).
Datablock optimizer saved 100205 bytes (~8.1%).

Using zlib compression.

EXE header size:               57856 / 56320 bytes
Install code:                  14832 / 58196 bytes
Install data:                1045670 / 3116385 bytes
Uninstall code+data:           21058 / 21474 bytes
CRC (0x239C5E6F):                  4 / 4 bytes

Total size:                  1139420 / 3252379 bytes (35.0%)
make settings
make[1]: Entering directory `/home/ddp/src/projects/git/github/ddpbsd/ossec-hids/src'

General settings:
    TARGET:           winagent
    V:
    DEBUG:
    DEBUGAD
    PREFIX:           /var/ossec
    MAXAGENTS:        2048
    DATABASE:
    ONEWAY:           no
    CLEANFULL:        no
User settings:
    OSSEC_GROUP:      ossec
    OSSEC_USER:       ossec
    OSSEC_USER_MAIL:  ossecm
    OSSEC_USER_REM:   ossecr
Lua settings:
    LUA_PLAT:         posix
```

(continues on next page)

(continued from previous page)

```
USE settings:
    USE_ZEROMQ:       no
    USE_GEOIP:        no
    USE_PRELUDE:      no
    USE_OPENSSL:      no
    USE_INOTIFY:      no
Mysql settings:
    includes:
    libs:
Pgsql settings:
    includes:
    libs:
Defines:
    -DMAX_AGENTS=2048 -DOSSECHIDS -DDEFAULTDIR="/var/ossec" -DUSER="ossec" -DREMUSER=
↪"ossecr" -DGROUPGLOBAL="ossec" -DMAILUSER="ossecm" -DLinux
 Compiler:
    CFLAGS           -O2 -DMAX_AGENTS=2048 -DOSSECHIDS -DDEFAULTDIR="/var/ossec" -
↪DUSER="ossec" -DREMUSER="ossecr" -DGROUPGLOBAL="ossec" -DMAILUSER="ossecm" -DLinux -
↪Wall -Wextra -I./ -I./headers/
    LDFLAGS          -lm
    CC              gcc
    MAKE            make
make[1]: Leaving directory `/home/ddp/src/projects/git/github/ddpbsd/ossec-hids/src'

Done building winagent
```

The final output will be saved to `./win32/ossec-win32-agent.exe`.

Compiling OSSEC 2.8.x with MinGW:

Generate the Windows package directory:

```
# cd ossec-hids-2.8.3/src/win32
# ./gen-win.sh
```

Now, you will have the win-pkg directory under src. Just go there and run make.sh. Your Windows agent package should be created in a few minutes:

Note: The `make.sh` script may require modification depending on the Linux distribution used.

```
# cd ../win-pkg
# sh ./make.sh
```

You will see the following in the screen:

```
Making windows agent
rootcheck/win-common.c: In function "__os_winreg_querykey":
rootcheck/win-common.c:279: warning: pointer targets in passing argument 7 of
↪"RegEnumValueA" differ in signedness
win-registry.c: In function "os_winreg_querykey":
...

Output: "ossec-win32-agent.exe"
```

(continues on next page)

(continued from previous page)

```
Install: 7 pages (448 bytes), 3 sections (3144 bytes), 379 instructions (10612 bytes),
↪ 247 strings (42580 bytes), 1 language table (346 bytes).
Uninstall: 5 pages (320 bytes),
1 section (1048 bytes), 301 instructions (8428 bytes), 166 strings (2646 bytes), 1␣
↪language table (290 bytes).
Datablock optimizer saved 8371 bytes (~0.7%).
```

Which means that your agent executable ossec-win32-agent.exe has been created properly.

Integration and Deployment with cfengine

I recently required a larger deployment of OSSEC-HIDS without too much manual intervention. Almost every OSSEC-HIDS tutorial I've across says this is possible, yet I was unable to find a tutorial demonstrating it. So, in the spirit of open source, I'm contributing a brief overview.

Prerequisites:

In order to facilitate the key request, I chose to generate a file with the relevant information and copy it back to my cfmaster server. I developed the following tutorial to demonstrate a cfengine copy back scenario: Copy Back with cfengine.

Configuring the cfengine clients:

I added a group to my `cfagent.conf` for my ossec server named: `hg_ossec_server` (host group). I then created an `ossec-hids.cf` containing the following:

- control

My control sections sets up the variables I'll be using in the rest of the file.

```
control:
  any::
    ossec_key_dir = (/usr/local/cfkeys/ossec)
    ossec_req_dir = ( $(util_updir)/ossec )
```

- package

I'm using yum to automatically install OSSEC-HIDS from my local RPM Repository.

```
packages:
  !hg_ossec_server::
      ossec-hids                  action=install
      ossec-hids-client           action=install
```

- links

The Links section just links ossec-agent.conf to ossec.conf on the clients.

```
links:
  !hg_ossec_server::
    /var/ossec/etc/ossec.conf -> /var/ossec/etc/ossec-agent.conf
```

- copy

I manage the `ossec-agent.conf` in cfengine, because my cfengine configurations are all stored in a subversion repository. The first stanza in copy just pushes the most recent copy of the `ossec-agent.conf` file to my network, setting the dynamic class `dc_restart_ossec` if the copy occurs.

```
copy:
  !hg_ossec_server::
      $(distribute)/ossec-agent.conf      dest=/var/ossec/etc/ossec-agent.conf
                                          server=$(policyhost)
                                          mode=640
                                          group=ossec
                                          type=sum
                                          define=dc_restart_ossec
```

This second stanza in the `copy` section copies a file from our ossec key directory to the client.keys file on the client. This copy only happens if the two files are different. It also sets `dc_restart_ossec` if the copy occurs.

```
$(ossec_key_dir)/$(host).ossec    dest=/var/ossec/etc/client.keys
                                  server=$(policyhost)
                                  mode=640
                                  group=ossec
                                  type=sum
                                  define=dc_restart_ossec
```

- processes

My processes block checks to ensure that OSSEC-HIDS is running the correct daemons.

```
processes:
  !hg_ossec_server::
      "ossec-agentd" elsedefine=dc_restart_ossec
  ``hg_ossec_server``::
      "ossec-remoted" elsedefine=dc_restart_ossec
```

- shellcommands

This section is where the certificate request occurs through some devious mechanisms I designed for no other reason than to amuse myself. Hopefully, it amuses others as well. The first thing it does is issue a command that echo's the client eth0 ipv4 address to a file named ''host.ossec" in the ossec request directory I defined. The `hg_ossec_server` class will use this to generate a cert to place in the aforementioned copy block.

```
shellcommands:
  !hg_ossec_server::
      "/usr/bin/ssh util@$(policyhost) -i $(util_privkey) 'echo $(global.ipv4[eth0]) >
↪ $(ossec_req_dir)/$(host).ossec'"
```

The last statement checks to see if anyone defined `dc_restart_ossec`, and restart ossec-hids if it was defined.

```
dc_restart_ossec::
    "/sbin/service ossec-hids restart"
```

Ok, so who cares?

Well, now, our clients are setup to install, configure, and run OSSEC-HIDS as well as issuing a request for their certificate. However, the certificate directory on the server is empty and so none of them will actually run. This is a problem.

Configuring the OSSEC Server w/cfengine

The cfengine part of this was a pain for me because of the order of the actions I had defined and the extent of work I had done incorrectly in the past. I could have figured out an interesting way to handle this, but I didn't want to scrap my entire cfengine config and start from scratch. So I created a perl script that allowed me to use the `manage_agents` script without interaction. It does require the `Expect.pm` & `Regexp::Common` from CPAN, but is otherwise stock Perl 5.8.x. I also wrote a shell script wrapper to handle running the perl script and culminating the results. I saved these two scripts in `/root/security`, so if you put them elsewhere, make sure to update the shell script wrapper.

The scripts for managing keys can be downloaded here

The cfengine bit was really simple, it just had to call my wrapper shell script and set the class. I did this with a control block:

```
control:
 hg_ossec_server::
   AddClasses = ( ExecResult(/root/security/ossec-scan.sh) )
```

The combination of the two scripts and this one line in the cfengine configuration handle creating, removing, and exporting the keys, as well as configuring the `dc_restart_ossec` class if there have been changes.

OSSEC Updates

Updating OSSEC is as easy as it can get. Just download the latest package and follow the installation instructions as usual. It will detect that you already have it installed and ask:

```
- You already have OSSEC installed. Do you want to update it? (y/n): y
```

Just answer `yes` to this question and the script will update the OSSEC binaries. `local_rules.xml` and `local_decoder.xml` will not be modified during this upgrade.

The script will also prompt for an answer to the following question:

```
- Do you want to update the rules? (y/n): y
```

Answering `yes` to this question updates the `<rules>` section of the system's ossec.conf.

Agents

There are two types of agents within OSSEC: installable agents and agentless agents. Installable agents are installed on hosts, and they report back to a central OSSEC server via the OSSEC encrypted message protocol. Agentless agents require no installation on remote hosts. They are processes initiated from the OSSEC manager, which gather information from remote systems, and use any RPC method (e.g. ssh, snmp rdp, wmi).

Agent

Communication between agents and the OSSEC server

Communication between agents and the OSSEC server generally occurs on port 1514/udp in secure mode. If using the syslog mode for *ossec-remoted*, then port 514 is the default (both UDP and TCP are supported). These ports are configurable in the remote section of the ossec.conf

The secure connection method is generally preferred over syslog. Also, an outside syslog daemon (like rsyslog or syslog-ng) may be used instead of the syslog support in *ossec-remoted*.

Managing Agents

To add an agent to an OSSEC manager with *manage_agents* you need to follow the steps below.

1. Run manage_agents on the OSSEC server.
2. Add an agent.
3. Extract the key for the agent.
4. Copy that key to the agent.
5. Run manage_agents on the agent.
6. Import the key copied from the manager.
7. Restart the manager's OSSEC processes.
8. Start the agent.

manage_agents on the OSSEC server

The server version of manage_agents provides an interface to:

- add an OSSEC agent to the OSSEC server
- extract the key for an agent already added to the OSSEC server
- remove an agent from the OSSEC server
- list all agents already added to the OSSEC server.

Running manage_agents and start screen

`manage_agents` should be run as a user with the appropriate privileges (e.g. root).

Run `manage_agents`:

```
# /var/ossec/bin/manage_agents
```

The manage_agents menu:

```
****************************************
* OSSEC HIDS v2.5-SNP-100809 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:
```

Typing the appropriate letter and hitting enter will initiate that function.

Adding an agent

To add an agent type `a` in the start screen:

```
Choose your action: A,E,L,R or Q: a
```

You are then prompted to provide a name for the new agent. This can be the hostname or another string to identify the system. In this example the agent name will be agent1.

```
- Adding a new agent (use '\q' to return to the main menu).
  Please provide the following:
   * A name for the new agent: agent1
```

After that you have to specify the IP address for the agent. This can either be a single IP address (e.g. 192.168.1.25), a range of IPs (e.g. 192.168.2.0/24), or `any`. Using a network range or `any` is preferable when the IP of the agent may change frequently (DHCP), or multiple systems will appear to come from the same IP address (NAT).

```
* The IP Address of the new agent: 192.168.2.0/24
```

Warning: If you use a specific IP address it **must** be unique. Duplicate IP addresses will cause issues. Multiple systems can use the same IP range or `any`.

The last information you will be asked for is the ID you want to assign to the agent. *manage_agents* will suggest a value for the ID. This value should be the lowest positive number that is not already assigned to another agent. The ID 000 is assigned to the OSSEC server. To accept the suggestion, simply press ENTER. To choose another value, type it in and press ENTER.

```
* An ID for the new agent[001]:
```

As the final step in creating an agent, you have to confirm adding the agent:

```
Agent information:
   ID:002
   Name:agent1
   IP Address:192.168.2.0/24

Confirm adding it?(y/n): y
Agent added.
```

After that *manage_agents* appends the agent information to /var/ossec/etc/client.keys and goes back to the start screen.

Warning: If this is the first agent added to this server, the server's OSSEC processes should be restarted using `/var/ossec/bin/ossec-control restart`.

Extracting the key for an agent

After adding an agent, a key is created. This key must be copied to the agent. To extract the key, use the `e` option in the manage_agents start screen. You will be given a list of all agents on the server. To extract the key for an agent, simply type in the agent ID. It is important to note that you have to enter all digits of the ID.

```
Choose your action: A,E,L,R or Q: e

Available agents:
   ID: 001, Name: agent1, IP: 192.168.2.0/24
Provide the ID of the agent to extract the key (or '\q' to quit): 001
```

(continues on next page)

(continued from previous page)

```

Agent key information for '001' is:
MDAyIGFnZW50MSAxOTIuMTY4LjIuMC8yNCBlNmY3N2RiMTdmMTJjZGRmZjg5YzA4ZDk5m

** Press ENTER to return to the main menu.
```

The key is encoded in the string (shortened for this example) `MDAyIGFnZW50MSAxOTIuMTY4LjIuMC8yNCBlNmY3N2RiMTdmMTJ` and includes information about the agent. This string can be added to the agent through the agent version of `manage_agents`.

Removing an agent

If you want to remove an OSSEC agent from the server, use the `r` option in the *manage_agents* start screen. You will be given a list of all agents already added to the server. To remove an agent, simply type in the ID of the agent, press enter, and finally confirm the deletion. It is important to note that you have to enter all digits of the ID.

```
Choose your action: A,E,L,R or Q: r

Available agents:
   ID: 001, Name: agent1, IP: 192.168.2.0/24
Provide the ID of the agent to be removed (or '\q' to quit): 001
Confirm deleting it?(y/n): y
Agent '001' removed.
```

`manage_agents` then invalidates the agent information in `/var/ossec/etc/client.keys`. Only the values for ID and the key are kept to avoid conflicts when adding agents. The deleted agent can no longer communicate with the OSSEC server.

manage_agents on OSSEC agents

The agent version provides an interface for importing authentication keys.

```
****************************************
* OSSEC HIDS v2.5-SNP-100809 Agent manager.     *
* The following options are available: *
****************************************
   (I)mport key from the server (I).
   (Q)uit.
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): [key extracted via manage_agents on the server]

Agent information:
   ID:001
   Name:agent1
   IP Address:192.168.2.0/24

Confirm adding it?(y/n): y
Added.
** Press ENTER to return to the main menu.
```

For the changes to be in effect you have to restart the server and start the agent.

Agent systems behind NAT or with dynamic IPs (DHCP)

If you want to install the agent on systems without a static IP address or behind a NAT device, you need to configure the agent using a CIDR address or the ip address of `any`.

DHCP Example

To add an agent that can receive any IP address in the 192.168.2.0/24 network, just provide the IP address of the agent as 192.168.2.0/24. Example (taken from manage_agents):

```
Please provide the following:
* A name for the new agent: test
* The IP Address of the new agent: 192.168.2.0/24
```

NAT Example

The same applies to systems behind a NAT device. The OSSEC server will see all agents behind the NAT as if they have the same IP address.

For example, you have systems 192.168.1.2, 192.168.1.3 and 192.168.1.4 behind a nat server that connects to network 10.1.1.0/24 with the ossec server on it.

In this case, you need to config the agents as if their IP was 10.1.1.0/24, because this is the IP that the server is seeing (not their original IP). Using `any` instead of an IP address or range is also a valid option, allowing the agent to connect from any IP address.

On the *manage_agents* tool, add each one of those agents on the server using the following format:

```
Please provide the following:
* A name for the new agent: agent-1
* The IP Address of the new agent: 10.1.1.0/24
```

```
Please provide the following:
* A name for the new agent: agent-2
* The IP Address of the new agent: any
```

Note: Make sure to use one separate key for each agent.

Adding an agent with ossec-authd

It is possible to add a key to a system via an automated method. ossec-authd and agent-auth provide this functionality.

ossec-authd

`ossec-authd` will run on the server adding agents and distributing authentication keys.

Warning: There is currently no authentication, so any host that can connect to the port ossec-authd listens to can obtain an OSSEC agent key. It is recommended that the OSSEC manager's firewall be used to help limit connections.

Run ossec-authd, listening on port 1515:

```
/var/ossec/bin/ossec-authd -p 1515
```

agent-auth

`agent-auth` will connect to an ossec-authd instance to receive, and install an agent key.

Run agent-auth connecting to the manager on IP 192.168.1.12 port 1515:

```
/var/ossec/bin/agent-auth -m 192.168.1.12 -p 1515
```

Centralized agent configuration

If you ever wanted to be able to configure your agents remotely, you will be happy to know that starting on version 2.1 you will be able to do so. We allow centralized configuration for file integrity checking (syscheckd), rootkit detection (rootcheck) and log analysis.

This is how it works.

Create agent configuration

First Create the file /var/ossec/etc/shared/agent.conf.

Inside the file you can configure the agent just as you would normally at ossec.conf

```
<agent_config>
    <localfile>
        <location>/var/log/my.log</location>
        <log_format>syslog</log_format>
    </localfile>
</agent_config>
```

But you have a few more options. You can restrict the config by agent name, operating system, or profile:

```
<agent_config name="agent1">
    <localfile>
        <location>/var/log/my.log</location>
        <log_format>syslog</log_format>
    </localfile>
</agent_config>

<agent_config os="Linux">
    <localfile>
        <location>/var/log/my.log2</location>
        <log_format>syslog</log_format>
    </localfile>
</agent_config>
```

(continues on next page)

(continued from previous page)

```
<agent_config os="Windows">
    <localfile>
        <location>C:\myapp\my.log</location>
        <log_format>syslog</log_format>
    </localfile>
</agent_config>
```

And only the proper agent will read them, giving us great granularity to push the configuration to all your agents.

After you configured, the manager will push it to the agents. Note that it can take a while for it to complete (since the manager caches the shared files and only re-reads them every few hours). If you restart the manager the configuration will be pushed much quicker.

Restart the agent

Once the configuration file is pushed, you can run the command *agent_control* to see if the agent received the config and restart the agent remotely.

```
# md5sum /var/ossec/etc/shared/agent.conf
MD5 (/var/ossec/etc/shared/agent.conf) = ee1882236893df851bd9e4842007e7e7
# /var/ossec/bin/agent_control -i 200

OSSEC HIDS agent_control. Agent information:
Agent ID: 200
Agent Name: ourhome
IP address: 192.168.0.0/16
Status: Active

Operating system: Linux ourhome 2.6.24-23-generic #1 SMP Mon Jan 26 00..
Client version: OSSEC HIDS v2.1 / ee1882236893df851bd9e4842007e7e7
Last keep alive: Tue Jun 30 08:29:17 2009

Syscheck last started at: Tue Jun 30 04:29:32 2009
Rootcheck last started at: Tue Jun 30 06:03:08 2009
```

When the agent received the configuration, the "Client Version" field will have the md5sum of the agent.conf file.

Note: Linux systems generally use `md5sum`, but other systems may use `md5` as the name of the application to check the hash of the file.

To restart the agent:

```
# /var/ossec/bin/agent_control -R 200 (where 200 is the agent id)

OSSEC HIDS agent_control: Restarting agent: 200
```

Agentless

Agentless Monitoring

Agentless monitoring allows you to run integrity checking on systems without an agent installed (including routers, firewalls, switches and even Linux/BSD systems). It can be executed just like our normal file integrity checking (alerting of checksum changes) or doing diffs and showing exactly what has changed.

Agentless configuration options

Check _manual-agentless-scripts for more information.

Getting started with agentless

After you installed OSSEC, you need to enable the agentless monitoring:

```
# /var/ossec/bin/ossec-control enable agentless
```

And provide the SSH authentication to the host you want to access. For Cisco devices (PIX, routers, etc), you need to provide an additional parameter for the enable password. The same thing applies if you want to add support for "su", it must be the additional parameter. In this example, I am adding a Linux box (example.net) and a PIX firewall (pix.fw.local):

```
# /var/ossec/agentless/register_host.sh add root@example.net mypass1
  *Host root@example.netl added.
# /var/ossec/agentless/register_host.sh add pix@pix.fw.local pixpass enablepass
  *Host pix@pix.fw.local added.

# /var/ossec/agentless/register_host.sh list
  *Available hosts:
pix@pix.fw.local
root@example.net
```

Note: `register_host.sh` is a shell script, special characters may need to be escaped to not be interpreted by the shell.

If you want to use public key authentication instead of passwords, you need to provide NOPASS as the password and create the public key:

```
# sudo -u ossec ssh-keygen
```

It will create the public keys inside /var/ossec/.ssh . After that, just scp the public key to the remote box and your password less connection should work.

Configuring agentless

Once you have added all your systems, you need to configure OSSEC to monitor them. By default, we have 4 agentless types (but we plan to add more soon):

- ssh_integrity_check_bsd
- ssh_integrity_check_linux
- ssh_generic_diff
- ssh_pixconfig_diff

For the first two, you give a list of directories in the configuration and OSSEC will do the integrity checking of them on the remote box. On the ssh_generic_diff, you give a set of commands to run on the remote box and OSSEC will alert when the output of them changes. The ssh_pixconfig_diff will alert when a Cisco PIX/router configuration changes.

So, for my first system (root@example.net), I will monitor the /bin, /etc and /sbin directories every 10 hours (if I was using the ssh_integrity_check_bsd, the argument would be the directories as well):

```
<agentless>
    <type>ssh_integrity_check_linux</type>
    <frequency>36000</frequency>
    <host>root@example.net</host>
    <state>periodic</state>
    <arguments>/bin /etc/ /sbin</arguments>
</agentless>
```

For my PIX, the configuration looks like:

```
<agentless>
    <type>ssh_pixconfig_diff</type>
    <frequency>36000</frequency>
    <host>pix@pix.fw.local</host>
    <state>periodic_diff</state>
</agentless>
```

And just to exemplify the ssh_generic_diff I will also monitor ls -la /etc; cat /etc/passwd on the root@example.net. Note that if you want to monitor any network firewall or switch, you can use the ssh_generic_diff and just specify the commands in the arguments option. To use "su", you need to set the value "use_su" before the hostname (eg: <host>use_su root@example.net</host>).

```
<agentless>
    <type>ssh_generic_diff</type>
    <frequency>36000</frequency>
    <host>root@example.net</host>
    <state>periodic_diff</state>
    <arguments>ls -la /etc; cat /etc/passwd</arguments>
</agentless>
```

Running the completed setup

Once the configuration is completed, you can restart OSSEC. You should see something like "Started ossec-agentlessd" in the output. Before each agentless connection is started, OSSEC will do a configuration check to make sure everything is fine. Look at /var/ossec/logs/ossec.log for any error. If you see:

```
2008/12/12 15:20:06 ossec-agentlessd: ERROR: Expect command not found (or bad␣
↪arguments) for 'ssh_integrity_check_bsd'.
2008/12/12 15:20:06 ossec-agentlessd: ERROR: Test failed for 'ssh_integrity_check_bsd
↪' (127). Ignoring.'
```

It means that you don't have the expect library installed on the server (it is not necessary to install anything on the agentless systems to monitor). On Ubuntu you can do the following to install:

```
# apt-get install expect
```

After installing expect, you can restart OSSEC and you should see:

```
2008/12/12 15:24:12 ossec-agentlessd: INFO: Test passed for 'ssh_integrity_check_bsd'.
↪'
```

When it connects to the remote system, you will also see:

```
2008/12/12 15:25:19 ossec-agentlessd: INFO: ssh_integrity_check_bsd: root@example.
↪net: Starting.
2008/12/12 15:25:46 ossec-agentlessd: INFO: ssh_integrity_check_bsd: root@example.
↪net: Finished.
```

Alerts

These are some of the alerts you will get:

For the ssh_generic_diff:

```
OSSEC HIDS Notification.
2008 Dec 12 01:58:30

Received From: (ssh_generic_diff) root@example.net->agentless
Rule: 555 fired (level 7) -> "Integrity checksum for agentless device changed."
Portion of the log(s):

ossec: agentless: Change detected:
35c35
< -rw-r-r- 1 root wheel 34 Dec 10 03:55 hosts.deny
--
> -rw-r-r- 1 root wheel 34 Dec 11 18:23 hosts.deny
-END OF NOTIFICATION
```

For the PIX:

```
OSSEC HIDS Notification.
2008 Dec 01 15:48:03

Received From: (ssh_pixconfig_diff) pix@pix.fw.local->agentless
Rule: 555 fired (level 7) -> "Integrity checksum for agentless device changed."
Portion of the log(s):

ossec: agentless: Change detected:
48c48
< fixup protocol ftp 21
--
> no fixup protocol ftp 21
100c100
< ssh timeout 30
--
> ssh timeout 50
More changes..

-END OF NOTIFICATION
```

Contents

Writing Agentless Scripts

All scripts that work with OSSEC agentless security monitoring use stdout for communication and reporting to the OSSEC server. This makes writing scripts for OSSEC simple as you do not need to do anything more then print or echo to stdout. The format of the output does need to meet the OSSEC specification, but that is a very simple thing to do.

Agentless Script Types

Before we move to the specification details I need to explain that OSSEC agentless runs to different types of scripts. Namely the following:

- **periodic_diff**
  - Scripts output data to the OSSEC agentless process that will then be compared to past runs and if there are differences an OSSEC alert will be generated.
- **periodic**
  - Scripts output controlled messages to the OSSEC agentless process that will then be processed accordingly.

Periodic diff Specification

The output for periodic_diff is very simple, any and all output after the agentless command `STORE: now` and before the next OSSEC Command will be stored and compared for differences. This type of script is mostly used for hardware devices such as Cisco IOS, Juniper JunOS, and other products.

Scripts that use the periodic_diff make use of the following commands:

- **INFO:**
  - The string following INFO will be logged to /var/ossec/logs/ossec.log by OSSEC for debugging.
- **ERROR:**
  - Error needs to be reported. The string following this command is forwarded to the OSSEC manager, and the OSSEC process closes down the script.
- **STORE:**
  - All the lines that follows this command will be added stored and compared to previous runs of the script

Here is an example of a periodic_diff script that comes with OSSEC. (Please note with all agentless scripts you must be in the root of the OSSEC install for them to function correctly.)

```
obsd46#( cd /var/ossec && ./agentless/ssh_pixconfig_diff cisco@172.17.0.1 'show␣
↪hardware' )
spawn ssh -c des cisco@172.17.0.1
No valid ciphers for protocol version 2 given, using defaults.
Password:
```

(continues on next page)

(continued from previous page)

```
a.zfw.tss>INFO: Starting.
enable
Password:
a.zfw.tss#ok on enable pass

STORE: now
no pager
             ^
% Invalid input detected at '^' marker.

a.zfw.tss#term len 0
a.zfw.tss#terminal pager 0
                     ^
% Invalid input detected at '^' marker.

a.zfw.tss#show version | grep -v Configuration last| up
                         ^
% Invalid input detected at '^' marker.

a.zfw.tss#show running-config
Building configuration...


Current configuration : 14631 bytes
!
version 12.4

[................SNIP CONFIG.................]

a.zfw.tss#show hardware
Cisco IOS Software, 3800 Software (C3845-ADVENTERPRISEK9-M), Version 12.4(24)T1,␣
↪RELEASE SOFTWARE (fc3)
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2009 by Cisco Systems, Inc.
Compiled Fri 19-Jun-09 19:21 by prod_rel_team

ROM: System Bootstrap, Version 12.3(11r)T2, RELEASE SOFTWARE (fc1)

a.zfw.tss uptime is 1 week, 5 days, 7 hours, 29 minutes
System returned to ROM by reload at 13:34:26 UTC Thu Oct 22 2009
System image file is "flash:c3845-adventerprisek9-mz.124-24.T1.bin"


This product contains cryptographic features and is subject to United
States and local country laws governing import, export, transfer and
use. Delivery of Cisco cryptographic products does not imply
third-party authority to import, export, distribute or use encryption.
Importers, exporters, distributors and users are responsible for
compliance with U.S. and local country laws. By using this product you
agree to comply with applicable laws and regulations. If you are unable
to comply with U.S. and local laws, return this product immediately.

A summary of U.S. laws governing Cisco cryptographic products may be found at:
http://www.cisco.com/wwl/export/crypto/tool/stqrg.html

If you require further assistance please contact us by sending email to
export@cisco.com.
```

(continues on next page)

(continued from previous page)

```
Cisco 3845 (revision 1.0) with 1007615K/40960K bytes of memory.
Processor board ID FTX1043A2CR
2 Gigabit Ethernet interfaces
1 ATM interface
1 Virtual Private Network (VPN) Module
4 CEM T1/E1 ports
DRAM configuration is 64 bits wide with parity enabled.
479K bytes of NVRAM.
492015K bytes of USB Flash usbflash0 (Read/Write)
62720K bytes of ATA System CompactFlash (Read/Write)

Configuration register is 0x2102


a.zfw.tss#exit
Connection to 172.17.0.1 closed by remote host.
Connection to 172.17.0.1 closed.

INFO: Finished.
```

In this example above the script would store the contents between `STORE: now` and `INFO: Finished.`. If this is the first time that OSSEC agentless has run this command no alerts would be generated and the contents would have been saved for later comparisons. If OSSEC agentless has a stored copy from a previous execution it will compare the files and if there are any differences it will generate an alert.

Periodic Specification

The periodic specification has more options and gives more control to the script writer on what actions OSSEC will take. Once again stdout is used for communication so script writing is easy.

- **INFO:**
  - The string following INFO will be logged to /var/ossec/logs/ossec.log by OSSEC for debugging.
- **ERROR:**
  - Error needs to be reported. The string following this command is forwarded to the OSSEC manager, and the OSSEC process closes down the script.
- **FWD:**
  - The string following FWD is a colon delimited list of stats on a given file.
- **LOG:**
  - The string following LOG: will be passed into ossec-analysisd and processed like all other log messages.

Example of real FWD: command.

```
FWD:␣
↪19419:600:0:0:fb30de5b02029950ae05885a3d407c8c:017cd6118cdc166ee8eba8af1b7fdad6763203d3␣
↪./.bash_history
```

The Fields break down in to the following:

- FWD:

  - The OSSEC Command
- 19419
  - Total size of file, in bytes
- 600
  - Access rights of file in octal
- 0
  - User ID of file owner
- 0
  - Group ID of file owner
- fb30de5b02029950ae05885a3d407c8c
  - MD5 Hash of file
- 017cd6118cdc166ee8eba8af1b7fdad6763203d3
  - SHA1 Hash of file
- ./.bash_history
  - Path and name of file

Using this format OSSEC can store the information about a file and then in the future run compare that they are the same. If for some reason they are not the same an alert will be generated. Here is an example of a password change on a linux system:

```
OSSEC HIDS Notification.
2009 Sep 21 15:19:00

Received From: (ssh_integrity_check_linux) root@172.17.20.20->syscheck
Rule: 550 fired (level 7) -> "Integrity checksum changed."
Portion of the log(s):

Integrity checksum changed for: '/etc/shadow'
Old md5sum was: '0d92e12c92f3edcf9d8876ea57c5f677'
New md5sum is : '2bd51b61dea17c5682fb2c0cf4f92c63'
Old sha1sum was: '2270c03a920ef8dd50e11cefdef046a8660f7a29'
New sha1sum is : 'd9518ea9022b10d07f81925c6d7f2abb4364b548'

--END OF NOTIFICATION
```

Agentless Script: ssh_integrity_check_linux

Now that we have an understanding of how agentless scripts communicate with the parent OSSEC process, let's move on to a working example. The OSSEC supplied script `ssh_integrity_check_linux` is a great place to start, so lets open it up and see what is going on.

```
obsd46# cat /var/ossec/agentless/ssh_integrity_check_linux
 #!/usr/bin/env expect

 # @(#) $Id: ssh_integrity_check_linux,v 1.11 2009/06/24 17:06:21 dcid Exp $
 # Agentless monitoring
 #
```

(continues on next page)

(continued from previous page)

```
 # Copyright (C) 2009 Trend Micro Inc.
 # All rights reserved.
 #
 # This program is a free software; you can redistribute it
 # and/or modify it under the terms of the GNU General Public
 # License (version 3) as published by the FSF - Free Software
 # Foundation.


 # Main script.
source "agentless/main.exp"


 # SSHing to the box and passing the directories to check.
if [catch {
    spawn ssh $hostname
} loc_error] {
    send_user "ERROR: Opening connection: $loc_error.\n"
    exit 1;
}


source $sshsrc
source $susrc

set timeout 600
send "echo \"INFO: Starting.\"; for i in `find $args 2>/dev/null`;do tail \$i >/dev/
↪null 2>&1 &&
md5=`md5sum \$i | cut -d \" \" -f 1` && sha1=`sha1sum \$i | cut -d \" \" -f
 1` && echo FWD: `stat --printf \"%s:%a:%u:%g\" \$i`:\$md5:\$sha1 \$i; done; exit\r"
send "exit\r"

expect {
    timeout {
        send_user "ERROR: Timeout while running commands on host: $hostname .\n"
        exit 1;
    }
    eof {
        send_user "\nINFO: Finished.\n"
        exit 0;
    }
}

exit 0;
```

The comments in the script hints to what is going on, but everything up to and including set timeout 600 is related to setting up the expect functions and code for handling the ssh subprocess and connecting to the remote host. I am not going to spend any time with this section, I am just going to make use of it.

The meat of what is getting processed on the remote end all happens in two lines.

```
send "echo \"INFO: Starting.\"; for i in `find $args 2>/dev/null`;do tail \$i >/dev/
↪null 2>&1 &&
md5=`md5sum \$i | cut -d \" \" -f 1` && sha1=`sha1sum \$i | cut -d \" \" -f
 1` && echo FWD: `stat --printf \"%s:%a:%u:%g\" \$i`:\$md5:\$sha1 \$i; done; exit\r"
```

Let’s break this down to see what is happening.

The send command pushes the following string to the ssh subprocess which gets run on the remote end of the connection. Before the script is sent to the remote host expect internally processes the string. This includes searching for variables and removing any control characters.

The control characters are first taken into account, and in the case of our example all escaped special characters are processed. “, r, and $ would be replaced with “, “carriage return“, and & respectively. The reason the escape characters are needed so that they will not interfere with expects own string processing and control. We will need to handle control characters in this way when we begin writing our own script.

While special characters were being handled by expect it also looked for variables to replace, in this case it will find $args and replace it with what ever arguments were passed to the script by the OSSEC agentless process. If we specified the following in `/var/ossec/etc/ossec.conf` the $args variable would be replaced with `/bin /etc /sbin`.

```
<agentless>
    <type>ssh_integrity_check_linux</type>
    <frequency>3600</frequency>
    <host>root@172.17.20.20</host>
    <state>periodic</state>
    <arguments>/bin /etc /sbin</arguments>
</agentless>
```

Back to the commands that get run. Once expect has completed replacement we are left with this command.

```
echo "INFO: Starting."; for i in `find /bin /etc /sbin 2>/dev/null`;do tail $i >/dev/
↪null 2>&1 &&
md5=`md5sum $i | cut -d " " -f 1` && sha1=`sha1sum $i | cut -d " " -f
 1` && echo FWD: `stat --printf "%s:%a:%u:%g" $i`:$md5:$sha1 $i; done; exit
    exit
```

This script then goes and uses the Unix find command to locate all files in the specified path (from the arguments passed) and generates an OSSEC FWD: command for each one and prints it to stdout. Making use of the commands stat, md5sum, and sha1sum to generate the data needed. Here is an example of the output checking.

```
spawn ssh root@172.17.20.20
Last login: Wed Nov  4 11:32:51 2009 from 172.17.20.131^M
[linux26 ~]#
INFO: Started.
echo "INFO: Starting."; for i in `find {/bin /etc /sbin} 2>/dev/null`;do tail $i >/
↪dev/null 2>&1 &&
md5=`md5sum $i | cut -d " " -f 1` && sha1=`sh a1sum $i | cut -d " " -f
 1` && echo FWD: `stat --printf "%s:%a:%u:%g" $i`:$md5:$sha1 $i; done; exit
INFO: Starting.
FWD:␣
↪833:644:0:0:4148adea745af5121963f6b731b60013:60877a6f6981b16c0d53d32bcd3f07d41cfb5bd4␣
↪/etc/modprobe.d/
glib2.sh
[...........SNIP............]
FWD:␣
↪1696:644:0:0:c2bd306b205ad9e81fb02ce6b225d384:5244d65815cb228a4fac7bc4c1c7774508fb7505␣
↪/etc/nsswitch.conf
FWD:␣
↪85179:644:0:0:8db574225cd1068b47e77ceccd96f8ff:b5ef6183b35ee9d1b66ed2cefe98003c5bd99192␣
↪/etc/sensors.conf
FWD:␣
↪49:644:0:0:52c3df2f1edf30ca3db82174be3a68d2:1934648f2429b70b1f729d343a6956fb0ea73136␣
↪/etc/php.d/imap.ini
FWD:␣
↪873:644:0:0:04559d1fe27ecd079b69df8b319f937e:e5cab1bf1f9e4bc4386309f4e00a8b7be3e543a2␣
↪/etc/php.d/memcache.ini
```

(continues on next page)

(continued from previous page)

```
FWD:␣
↪59:644:0:0:94636ba6c4bac9d8d49d9de1a513ae0c:41d5164a2c6e332e40edf55c59a2d0df8a260964␣
↪/etc/php.d/pdo_mysql.ini
FWD:␣
↪49:644:0:0:917dbbafbfaaa20f660063d627123dae:0e829d4ffc69f58dc258510b4b8452412e31ccc5␣
↪/etc/php.d/json.ini
FWD:␣
↪0:644:0:0:d41d8cd98f00b204e9800998ecf8427e:da39a3ee5e6b4b0d3255bfef95601890afd80709␣
↪/etc/wvdial.conf
logout
Connection to 172.17.20.20 closed.

INFO: Finished.
```

Modifying to make own Agentless Script: ssh_dmz_linux

Using the built in OSSEC agentless scripts are great, but sometimes we need more focused scanning and checking. So let's modify the ssh_integrity_check_linux for our environment.

The goals for this new script will be to watch for changes to files based on the following criteria:

- All setuid and setgid files
- All files related to authentication (including .htaccess and ssh files)
- All application specific files (apache, ssh)

Finding all setuid and setgid files

Let's first start by identifying a method to locate all files with their setuid or setgid bits enabled. To do this we will ssh to the host 172.17.20.20 and use find to locate the files.

```
obsd46# sudo -u ossec ssh root@172.17.20.20
[linux26 ~]# find / -type f \( -perm -4000 -o -perm -2000 \)
/sbin/umount.nfs
/sbin/netreport
/sbin/unix_chkpwd
/sbin/mount.nfs
/sbin/pam_timestamp_check
/sbin/mount.nfs4
/sbin/umount.nfs4
/bin/ping6
/bin/su
/bin/umount
/bin/ping
/bin/mount
/lib/dbus-1/dbus-daemon-launch-helper
/usr/libexec/openssh/ssh-keysign
/usr/libexec/utempter/utempter
/usr/sbin/usernetctl
/usr/sbin/postqueue
/usr/sbin/userhelper
/usr/sbin/userisdnctl
/usr/sbin/postdrop
/usr/sbin/suexec
/usr/bin/chsh
/usr/bin/chfn
```

(continues on next page)

(continued from previous page)

```
/usr/bin/sudo
/usr/bin/locate
/usr/bin/wall
/usr/bin/sudoedit
/usr/bin/gpasswd
/usr/bin/lockfile
/usr/bin/newgrp
/usr/bin/write
/usr/bin/screen
/usr/bin/passwd
/usr/bin/chage
/usr/bin/sperl5.8.8
/usr/bin/crontab
/usr/bin/ssh-agent
```

Finding all files related to authentication and applications specific files

Finding all files with setuid and setgid was simple, but finding all files related to authentication is more involved. This of course will vary from system to system, but this should be good starting point.

```
obsd46# sudo -u ossec ssh root@172.17.20.20
[linux26 ~]# find / \( -name ".ssh" -o -name "ssh" -o -name "sshd" -o -name "httpd" -
↪o -name ".htaccess"
-o -name "pam.d" \) -exec find {} \;
/var/www/html/admin/modules/framework/var/www/html/admin/modules/.htaccess
/etc/httpd
/etc/httpd/conf
/etc/httpd/conf.d
/etc/httpd/conf.d/php.conf
/etc/httpd/conf.d/proxy_ajp.conf
/etc/httpd/conf.d/README
/etc/httpd/conf.d/ssl.conf
/etc/httpd/conf.d/welcome.conf
/etc/httpd/conf/httpd.conf
/etc/httpd/conf/magic
/etc/httpd/logs
/etc/httpd/modules
/etc/httpd/run
/etc/logrotate.d/httpd
/etc/pam.d
/etc/pam.d/authconfig
[...................SNIP PAM Files.....................]
/etc/pam.d/system-config-network-cmd
/etc/pam.d/vsftpd
/etc/rc.d/init.d/httpd
/etc/rc.d/init.d/sshd
/etc/ssh
/etc/ssh/ssh_config
/etc/ssh/sshd_config
/etc/ssh/ssh_host_dsa_key
/etc/ssh/ssh_host_dsa_key.pub
/etc/ssh/ssh_host_key
/etc/ssh/ssh_host_key.pub
/etc/ssh/ssh_host_rsa_key
/etc/ssh/ssh_host_rsa_key.pub
/etc/sysconfig/httpd
/root/.ssh
```

(continues on next page)

(continued from previous page)

```
/root/.ssh/authorized_keys
/usr/bin/ssh
/usr/lib/httpd
/usr/lib/httpd/modules
/usr/lib/httpd/modules/libphp5.so
[..................SNIP Apache modules...............]

/usr/lib/httpd/modules/mod_vhost_alias.so
/usr/sbin/httpd
/usr/sbin/sshd
/usr/src/tbm-pbxconfig-5.5.1/amp_conf/htdocs/admin/modules/framework/htdocs/admin/
↪modules/.htaccess
/usr/src/tbm-pbxconfig-5.5.1/amp_conf/htdocs/admin/modules/.htaccess
/var/empty/sshd
/var/empty/sshd/etc
/var/empty/sshd/etc/localtime
/var/www/html/admin/modules/framework/var/www/html/admin/modules/.htaccess
/var/www/html/admin/modules/.htaccess
```

Merging finds

Now we have two basic find methods that identify the files we want to monitor for changes, but our finds were a little greedy so we should create a way to strip out unwanted files from the list. As this is a unix system egrep is the king for finding or removing items from a list. To simplify things we can use egrep with the -v command line argument which tells egrep NOT to print any matching items.

Just to make sure that we do not end up double processing files we can make use of the sort command with -u argument to remove any duplicates.

Here is how we would put together both finds, egrep, and sort to locate and filter what is needed.

```
( find / -type f \( -perm -4000 -o -perm -2000 \) && \find / \( -name ".ssh" -o -name
↪"ssh" -o -name "sshd"
-o -name "httpd" -o -name ".htaccess" -o -name "pam.d" \) -exec find {} \; ) 2>/dev/
↪null | egrep
-v "known_hosts|moduli|var\/log|var\/lock" | sort -u
```

The above command we have found all files and paths that we would like to monitor, but this still needs to be integrated into a script on the OSSEC server.

Creating ssh_dmz_linux

We don't want to make changes to ssh_integrity_check_linux directly so we will need to make a copy.

```
obsd46# (cd /var/ossec/agentless && cp ssh_integrity_check_linux ssh_dmz_linux)
```

Integrating our new command line into the script we must pay close attention to special characters that expect will process. Due to this we will need to escape all / and " by proceeding them with . Once we are done escaping we just insert our new line in place of find $args 2>/dev/null in our new file.

Here is what the completed script will look like.

```
obsd56# cat /var/ossec/agentless/ssh_dmz_linux
 #!/usr/bin/env expect

 # @(#) $Id: ssh_integrity_check_linux,v 1.11 2009/06/24 17:06:21 dcid Exp $
 # Agentless monitoring
 #
```

(continues on next page)

(continued from previous page)

```
 # Copyright (C) 2009 Trend Micro Inc.
 # All rights reserved.
 #
 # This program is a free software; you can redistribute it
 # and/or modify it under the terms of the GNU General Public
 # License (version 3) as published by the FSF - Free Software
 # Foundation.


 # Main script.
source "agentless/main.exp"


 # SSHing to the box and passing the directories to check.
if [catch {
    spawn ssh $hostname
} loc_error] {
    send_user "ERROR: Opening connection: $loc_error.\n"
    exit 1;
}


source $sshsrc
source $susrc

set timeout 600
send "echo \"INFO: Starting.\"; for i in `(find / \\( -name \".ssh\" -o -name \"ssh\"␣
↪-o -name \"sshd\"
-o -name \"httpd\" -o -name \".htaccess\" -o -name \"pam.d\" \\) -exec find {} \\; &&␣
↪find / -type f
\\( -perm -4000 -o -perm -2000 \\); ) 2>/dev/null | egrep -v \"known_
↪hosts|moduli|var\\/log|var\\/lock\" | sort -u`;
do tail \$i >/dev/null 2>&1 && md5=`md5sum \$i | cut -d \" \" -f 1` && sha1=`sha1sum \
↪$i | cut -d \" \"
-f 1` && echo FWD: `stat --printf \"%s:%a:%u:%g\" \$i`:\$md5:\$sha1 \$i; done; exit\r"
send "exit\r"

expect {
    timeout {
        send_user "ERROR: Timeout while running commands on host: $hostname .\n"
        exit 1;
    }
    eof {
        send_user "\nINFO: Finished.\n"
        exit 0;
    }
}

exit 0;
```

Testing

Before we add this new script to OSSEC configuration we need to test it.

```
obsd46# (cd /var/ossec && sudo -u ossec ./agentless/ssh_dmz_linux root@172.17.20.20 )

ERROR: ssh_integrity_check <hostname> <arguments>
```

Due to not making use of the of the $arg variable in the way that ssh_integrity_check_linux wants use too, this caused this the problem above. Solving this problem would require making changes to files that will affect other built in scripts. So a quick solution is to just pass anything as an argument to the script. This will have no effect on our script as we do not make use of the $arg variable.

```
obsd46# (cd /var/ossec && sudo -u ossec ./agentless/ssh_dmz_linux root@172.17.20.20␣
↪NOTUSED)
spawn ssh root@172.17.20.20
Last login: Wed Nov  4 13:46:32 2009 from 172.17.20.131^M
[linux26 ~]#
INFO: Started.
echo "INFO: Starting."; for i in `(find / \( -name ".ssh" -o -name "ssh" -o -name
↪"sshd" -o -name "httpd"
-o -name ".htaccess" -o -name "pam.d" \)  -exec find {} \; && find / -type f \( -perm␣
↪-4000 -o -perm -2000
\); ) 2>/dev/null | egrep -v "known_hosts|moduli|var\/log|var\/lock"`;do tail $i >/
↪dev/null 2>&1 &&
 md5=`md5s ^Mum $i | cut -d " " -f 1` && sha1=`sha1sum $i | cut -d " " -f 1` && echo␣
↪FWD: `stat --printf
"%s:%a:%u:%g" $i`:$md5:$sha1 $i; done; exit
INFO: Starting.
FWD:␣
↪14:775:100:101:3bc0a3e92f8170084dd102eda9a474b1:25a1783a3c6bdd9745ec245ec1bfa0414ee05d23␣
↪/var/www/html/admin/modules/.htaccessmodules/.htaccess
FWD:␣
↪3519:644:0:0:e4ca381035a34b7a852184cc0dd89baa:6e43d0b5a46ed5ba78da5c7e9dcf319b27d769e7␣
↪/var/empty/sshd/etc/localtime
FWD:␣
↪560:644:0:0:58370830ecfa056421ad21aff9c18905:d115bb5aeefaab97c53fbbd5df84ebcb9170d796␣
↪/etc/httpd/conf.d/php.conf
[...................SNIP............................]
FWD:␣
↪392:644:0:0:e92bea7e9d70a9ecdc61edd7c0a2f59a:d77b61dac010c60589b4d8a2039e3b8a5bed18b2␣
↪/etc/httpd/conf.d/README
FWD:␣
↪70888:4711:0:0:9046bd13339e7ef22266067b633e601a:3fc41029ddb14fe4ed613f479fa9e89c944f04dd␣
↪/usr/bin/sperl5.8.8
FWD:␣
↪315416:6755:0:0:4c63a9709fb7f0f97c30aa29d204859c:c379efa658de72866b8f6de5767906ff78d127b0␣
↪/usr/bin/crontab
FWD:␣
↪88964:2755:0:99:baf3ebef6377d6ef42858776c33621b0:62394bf57d18c3fd49adeb39a1da61661cabc3c8␣
↪/usr/bin/ssh-agent
logout
Connection to 172.17.20.20 closed.

INFO: Finished.
```

Log monitoring/analysis

Log Analysis (or log inspection) is done inside OSSEC by the logcollector and analysisd processes. The first one collects the events and the second one analyzes (decodes, filters and classifies) them.

It is done in real time, so as soon as an event is written OSSEC will process them. OSSEC can read events from internal log files, from the Windows event log and also receive them directly via remote syslog.

What is log analysis?

Inside OSSEC we call log analysis a LIDS, or log-based intrusion detection. The goal is to detect attacks, misuse or system errors using the logs.

LIDS - Log-based intrusion detection or security log analysis are the processes or techniques used to detect attacks on a specific network, system or application using logs as the primary source of information. It is also very useful to detect software misuse, policy violations and other forms of inappropriate activities.

Quick Facts

- How often are logs monitored?
  - In real time.
- Where are the events analyzed?
  - In the manager.
- How long are they stored?
  - For as long as your policy dictates (it is user configurable).
- Where does this help me with compliance?
  - (PCI DSS, etc) It helps with the whole section 10 (log monitoring) of PCI.
- How much CPU does it use?
  - On the agent, it uses very little CPU/memory since it just read the events and forwards them to the manager.
  - On the manager, it depends on the number of events per second (EPS).
- How does it deal with false positives?
  - False positives can be eliminated using local rules.

Configuration Options

These options should be specified locally in each agent's ossec.conf file or the share agent.conf. Inside the `<localfile>` element, you can have the following options.

Monitoring logs

With in OSSEC there are two major methods for monitoring logs: file and process. Each method has its own page and examples.

Process Monitoring

Overview

We love logs. Inside OSSEC we treat everything as if it is a log and parse it appropriately with our rules. However, some information is not available in log files but we still want to monitor it. To solve that gap, we added the ability to monitor the output of commands via OSSEC, and treat the output of those commands just like they were log files.

Configuration examples

Disk space utilization (df -h) example

For example, if you wanted to monitor the disk space utilization, you would need to setup a cron job to dump the output of `df -h` to a log file (maybe /var/log/df.log) and configure OSSEC to look at it.

As of OSSEC version 2.3 you can monitor commands directly in OSSEC following configuration (in /var/ossec/etc/ossec.conf):

```
<localfile>
    <log_format>command</log_format>
    <command>df -h</command>
</localfile>
```

Since we already have a sample rule for df -h included with OSSEC you would see the following when any partition reached 100%:

```
** Alert 1257451341.28290: mail - ossec,low_diskspace,
2009 Nov 05 16:02:21 (home-ubuntu) 192.168.0.0->df -h

Rule: 531 (level 7) -> "Partition usage reached 100% (disk space monitor)."
Src IP: (none)
User: (none)
ossec: output: 'df -h': /dev/sdb1 24G 12G 11G 100% /var/backup
```

Load average (uptime) Example

Another example, if you want to monitor the load average, you can configure OSSEC to monitor the "uptime" command and alert when it is higher than 2, for example:

```
<localfile>
    <log_format>command</log_format>
    <command>uptime</command>
</localfile>
```

And in the rule (in /var/ossec/rules/local_rules.xml):

```
<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>
```

There are lots of possibilities with this feature. If you have ideas for commands to monitor and rules, please comment.

Alerting when output of a command changes

If you want to create alerts when a log or the output of a command changes, take a look at the new <check_diff /> option in the rules (available on the latest snapshot).

To demonstrate with an example, we will create a rule to alert when there is a new port open in listening mode on our server.

First, we configure OSSEC to run the `netstat -tan |grep LISTEN` command by adding the following to ossec.conf:

```
<localfile>
    <log_format>full_command</log_format>
    <command>netstat -tan |grep LISTEN|grep -v 127.0.0.1</command>
</localfile>
```

After that, I add a rule to alert when its output changes:

```
<rule id="140123" level="7">
    <if_sid>530</if_sid>
    <match>ossec: output: 'netstat -tan |grep LISTEN</match>
    <check_diff />
    <description>Listened ports have changed.</description>
</rule>
```

Note that we use the `<check_diff />` option. The first time it receives the event, it will store in an internal database. Every time it receives the same event, it will compare against what we have store and only alert if the output changes.

In our example, after configuring OSSEC, I started netcat to listen on port 23456 and that's the alert I got:

```
OSSEC HIDS Notification.
2010 Mar 11 19:56:30

Received From: XYZ->netstat -tan |grep LISTEN|grep -v 127.0.0.1
Rule: 140123 fired (level 7) -> "Listened ports have changed."
Portion of the log(s):

ossec: output: 'netstat -tan |grep LISTEN|grep -v 127.0.0.1':
tcp4       0      0 *.23456          *.*               LISTEN
tcp4       0      0 *.3306           *.*               LISTEN
tcp4       0      0 *.25             *.*               LISTEN
Previous output:
ossec: output: 'netstat -tan |grep LISTEN|grep -v 127.0.0.1':
tcp4       0      0 *.3306           *.*               LISTEN
tcp4       0      0 *.25             *.*               LISTEN
```

Detecting USB Storage Usage

Xavier Mertens wrote a very interesting article on Detecting USB Storage Usage with OSSEC. He used our policy auditing module for that, but I think USB monitoring can be done in a much easier way with our new **:xml:‘check_diff‘** feature.

To get started, first configure your Windows agents to monitor the USBSTOR registry entry using the reg command:

```
<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 a local rule for that command:

```
<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>
```

Now after a few minutes you will see a directory at `/var/ossec/queue/diff/[agent_name]/[rule_id]` with the current snapshot of this command. Once someone adds a new USB device you will get this alert:

```
** Alert 1268687754.35062: mail  - local,syslog,
2010 Mar 15 18:15:54 (xx-netbook) any->reg QUERY␣
↪HKLMSYSTEMCurrentControlSetEnumUSBSTOR
Rule: 140125 (level 7) -> 'New USB device connected'
Src IP: (none)
User: (none)
ossec: output: 'reg QUERY HKLMSYSTEMCurrentControlSetEnumUSBSTOR':! REG.EXE VERSION 3.
↪0

HKEY_LOCAL_MACHINESYSTEMCurrentControlSetEnumUSBSTOR
HKEY_LOCAL_MACHINESYSTEMCurrentControlSetEnumUSBSTORDisk&Ven_&Prod_USB_Flash_Memory&
↪Rev_5.00
HKEY_LOCAL_MACHINESYSTEMCurrentControlSetEnumUSBSTORDisk&Ven_Generic&Prod_Flash_Disk&
↪Rev_8.0
HKEY_LOCAL_MACHINESYSTEMCurrentControlSetEnumUSBSTORDisk&Ven_Hitachi&Prod_
↪HTS543225L9A300&Rev_
HKEY_LOCAL_MACHINESYSTEMCurrentControlSetEnumUSBSTORDisk&Ven_LEXAR&Prod_JD_FIREFLY&
↪Rev_1100
HKEY_LOCAL_MACHINESYSTEMCurrentControlSetEnumUSBSTORDisk&Ven_SAMSUNG&Prod_HM160JC&Rev_
↪0000
HKEY_LOCAL_MACHINESYSTEMCurrentControlSetEnumUSBSTORDisk&Ven_Sony&Prod_DSC&Rev_1.00
HKEY_LOCAL_MACHINESYSTEMCurrentControlSetEnumUSBSTORDisk&Ven_TomTom&Prod_ONE_XXL_IQ_
↪Rts
HKEY_LOCAL_MACHINESYSTEMCurrentControlSetEnumUSBSTORDisk&Ven_USB_2.0&Prod_USB_Flash_
↪Drive&Rev_0.00

Previous output:

ossec: output: 'reg QUERY HKLMSYSTEMCurrentControlSetEnumUSBSTOR':
! REG.EXE VERSION 3.0
HKEY_LOCAL_MACHINESYSTEMCurrentControlSetEnumUSBSTOR
HKEY_LOCAL_MACHINESYSTEMCurrentControlSetEnumUSBSTORDisk&Ven_&Prod_USB_Flash_Memory&
↪Rev_5.00
HKEY_LOCAL_MACHINESYSTEMCurrentControlSetEnumUSBSTORDisk&Ven_Generic&Prod_Flash_Disk&
↪Rev_8.07
HKEY_LOCAL_MACHINESYSTEMCurrentControlSetEnumUSBSTORDisk&Ven_Hitachi&Prod_
↪HTS543225L9A300&Rev_
HKEY_LOCAL_ACHINESYSTEMCurrentControlSetEnumUSBSTORDisk&Ven_SAMSUNG&Prod_HM160JC&Rev_
↪0000
HKEY_LOCAL_MACHINESYSTEMCurrentControlSetEnumUSBSTORDisk&Ven_Sony&Prod_DSC&Rev_1.00
HKEY_LOCAL_MACHINESYSTEMCurrentControlSetEnumUSBSTORDisk&Ven_TomTom&Prod_ONE_XXL_IQ_
↪Rts
HKEY_LOCAL_MACHINESYSTEMCurrentControlSetEnumUSBSTORDisk&Ven_USB_2.0&Prod_USB_Flash_
↪Drive&Rev_0.00
```

File Monitoring

Overview

OSSEC has a process named `ossec-logcollector` that monitors the configured log files for new events. When new log messages arrive, it forwards them to other processes for analysis or transport to an OSSEC server.

Configuration

The configuration for ossec-logcollector exists in `/var/ossec/etc/ossec.conf` in the `<ossec_config>` section. The syntax can be found in the localfile syntax page

Configuration examples

Simple example

Configuring a log file to be monitored is simple. Just provide the name of the file to be monitored and the format:

```
<localfile>
    <location>/var/log/messages</location>
    <log_format>syslog</log_format>
</localfile>
```

Windows EventLog Example

To monitor a Windows event log, you need to provide the format as "eventlog" and the location is the name of the event log. Example:

```
<localfile>
    <location>Security</location>
    <log_format>eventlog</log_format>
</localfile>
```

Windows EventChannel Example

To monitor a Windows event log on Windows Vista or later, you have the possibility to use the "eventchannel" log format. The location is the name of the event log. This is the only way to monitor Applications and Services logs. If the file name contains a "%4", replace it with "/". Example:

```
<localfile>
    <location>Microsoft-Windows-PrintService/Operational</location>
    <log_format>eventchannel</log_format>
</localfile>
```

Multiple Files Example

To check multiple files, OSSEC 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>
```

Date Based Example

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 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:

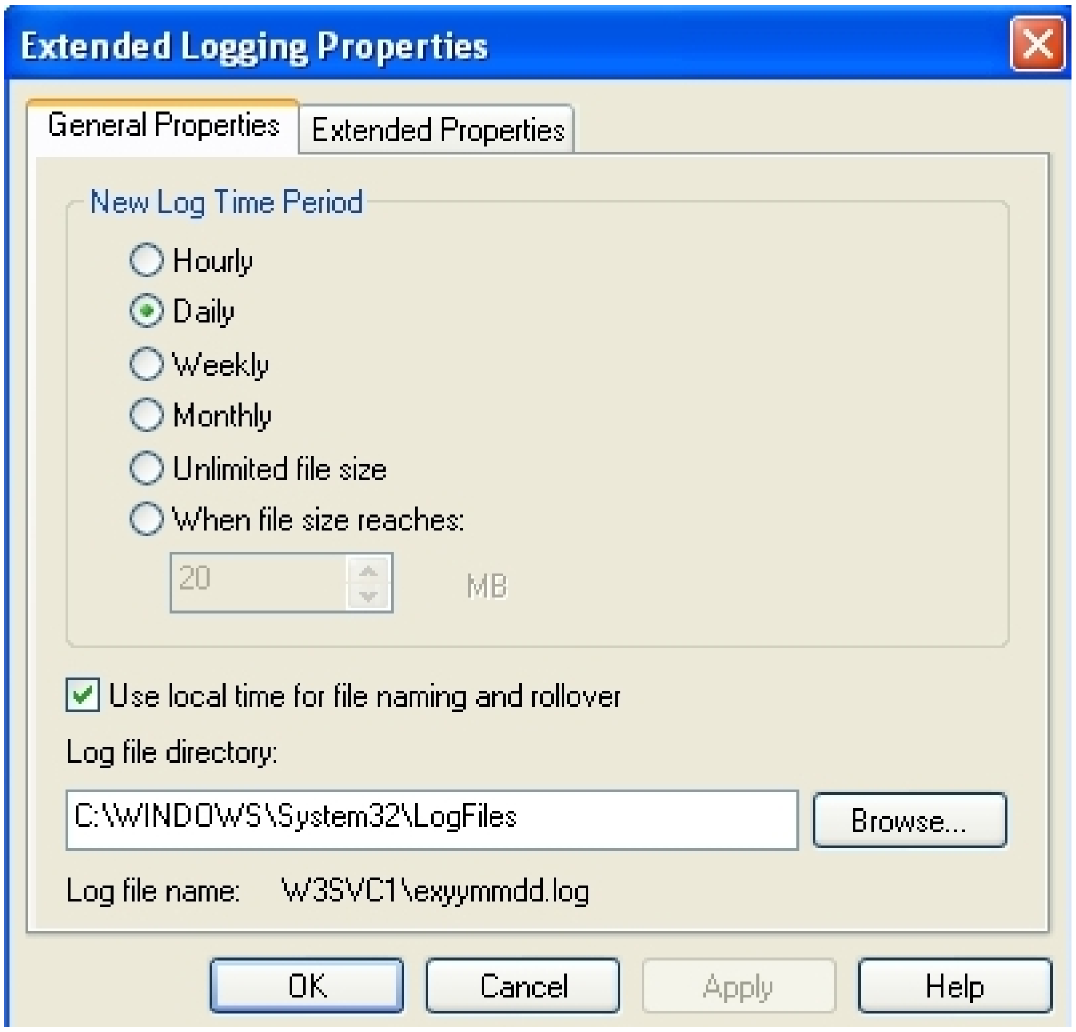
```
<localfile>
    <location>C:\Windows\app\log-%y-%m-%d.log</location>
    <log_format>syslog</log_format>
</localfile>
```

Warning: Wildcards cannot be combined with the date based format.

IIS Logs Example

Support for IIS (5 and 6) is available for the NCSA format (web only) and the W3C extended format (for Web, FTP and SMTP). By default, the installation scripts will attempt to configure OSSEC to monitor the first virtual hosts for web (W3SVC1 to W3SVC254), ftp (MSFTPSVC1 to MSFTPSVC254) and smtp (SMTPSVC1 to SMTPSVC254). To monitor any other file you need to add a new entry manually.

In addition to that, make sure to set the log time period to daily.



New Log Time Period

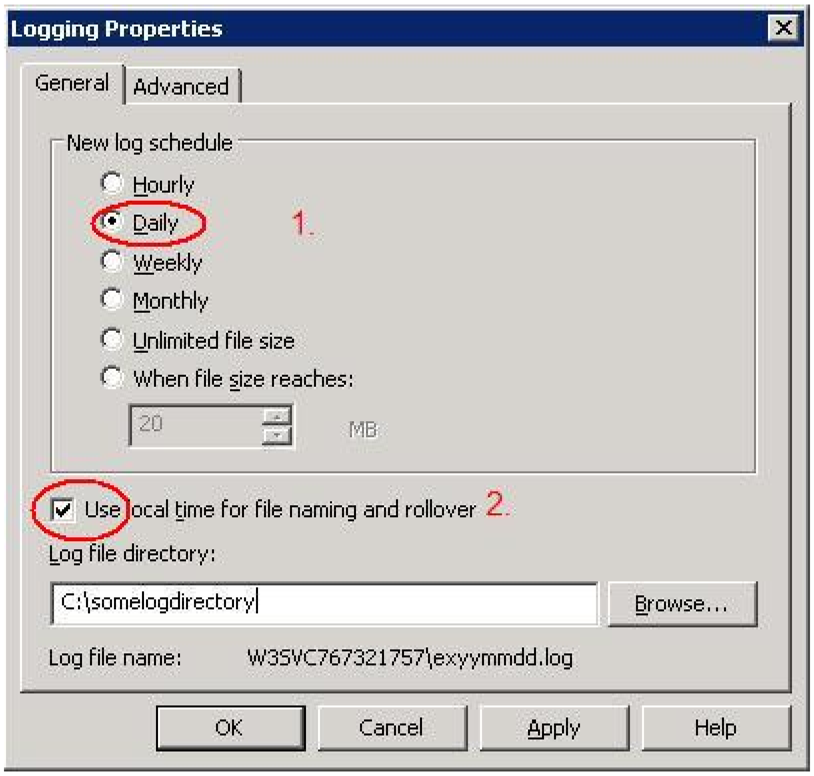
Hourly [✓ Daily] Weekly Monthly Unlimited file size When file size reaches: 20 MB

[✓] Use local time for file naming and rollover

Log file directory: C:\WINDOWS\System32\LogFiles

Log file name: W3SVC1\exyymmdd.log

And using the local time for file naming and rollover.


Logging Properties

General | Advanced

New log schedule

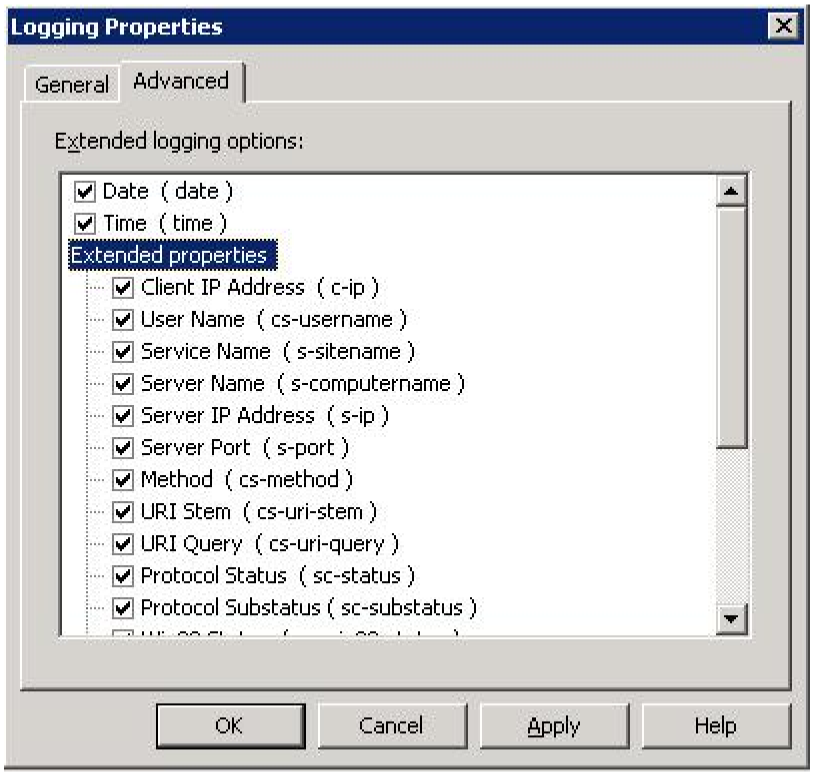
- ○ Hourly
- ● Daily 1.
- ○ Weekly
- ○ Monthly
- ○ Unlimited file size
- ○ When file size reaches: 20 MB

☑ Use local time for file naming and rollover 2.

Log file directory: C:\somelogdirectory

Log file name: W3SVC767321757\exyymmdd.log

In the extended logging properties, configure it to log the Date, Time and all the extended properties.


Logging Properties

General | Advanced

Extended logging options:

- ☑ Date (date)
- ☑ Time (time)
- Extended properties
  - ☑ Client IP Address (c-ip)
  - ☑ User Name (cs-username)
  - ☑ Service Name (s-sitename)
  - ☑ Server Name (s-computername)
  - ☑ Server IP Address (s-ip)
  - ☑ Server Port (s-port)
  - ☑ Method (cs-method)
  - ☑ URI Stem (cs-uri-stem)
  - ☑ URI Query (cs-uri-query)
  - ☑ Protocol Status (sc-status)
  - ☑ Protocol Substatus (sc-substatus)

The following is an example of configuration to monitor the virtual server 2 of IIS web

```
<localfile>
    <location>%WinDir%\System32\LogFiles\W3SVC3\ex%y%m%d.log</location>
    <log_format>iis</log_format>
</localfile>
```

Syscheck

Syscheck is the name of the integrity checking process inside OSSEC. It runs periodically to check if any configured file (or registry entry on Windows) has changed.

Why Integrity checking?

This is the explanation from the OSSEC book:

> There are multiple types of attacks and many attack vectors, but there is one thing unique about all of them: they leave traces and always change the system in some way. From viruses that modify a few files, to kernel-level rootkits that alters the kernel, there is always some change in the integrity of the system.
>
> Integrity checking is an essential part of intrusion detection, that detects changes in the integrity of the system. OSSEC does that by looking for changes in the MD5/SHA1 checksums of the key files in the system and on the Windows registry.
>
> The way it works is that the agent scans the system every few hours (user defined) and send all the checksums to the server. The server stores the checksums and look for modifications on them. An alert is sent if anything changes.

Quick facts

- How often does it run?
  - By default every 6 hours, but the frequency or time/day are configurable.
- Where is the database stored?
  - On the manager in `/var/ossec/queue/syscheck`.
- How does it help with compliance? (PCI DSS, etc)
  - It helps with sections 11.5 (install FIM software) and 10.5 (integrity checking of log files) of PCI.
- How much CPU does it use?
  - The scans are performed slowly to avoid using too much CPU/memory.
- How are false positives handled?
  - Files can be ignored manually in the configuration or using rules. By default when a file has changed 3 times further changes are automatically ignored.

Realtime options

`ossec-syscheckd` is able to check file integrity in near realtime on Windows and modern Linux distros. Windows comes with support out of the box, but on Linux systems inotify packages may need to be installed. Check for inotify dev packages, and possibly an inotify-tools package.

Configuration options

These configuration options can be specified in each agent's ossec.conf file, except for the `auto_ignore` and `alert_new_file` which apply to manager and local installs. The `ignore` option applies to all agents if specified on the manager.

`directories`

Use this option to add or remove directories to be monitored (they must be comma separated). All files and subdirectories will also be monitored. Drive letters without directories are not valid. At a minimum the '.' should be included (`D:\.`). This should be set on the system you wish to monitor (or in the agent.conf if appropriate).

Default: /etc,/usr/bin,/usr/sbin,/bin,/sbin

Attributes:

- **realtime**: Value=yes
  - This will enable realtime/continuous monitoring on Linux (using the inotify system calls) and Windows systems.
- **report_changes**: Value=yes
  - Report diffs of file changes. This is limited to text files at this time.

  Note: This option is only available on Unix-like systems.
- **check_all**: Value=yes
  - All the following check_* options are used together unless a specific option is explicitly overridden with "no".
- **check_sum**: Value=yes
  - Check the md5 and sha1 hashes of the of the files will be checked.

    This is the same as using both check_sha1sum="yes" and check_md5sum="yes"
- **check_sha1sum**: Value=yes
  - When used only the sha1 hash of the files will be checked.
- **check_md5sum**: Value=yes
  - The md5 hash of the files will be checked.
- **check_size**: Value=yes
  - The size of the files will be checked.
- **check_owner**: Value=yes
  - Check the owner of the files selected.
- **check_group**: Value=yes
  - Check the group owner of the files/directories selected.
- **check_perm**: Value=yes
  - Check the UNIX permission of the files/directories selected. On windows this will only check the POSIX permissions.
- **restrict**: Value=string

- A string that will limit checks to files containing that string in the file name.

  Allowed: Any directory or file name (but not a path)

- **no_recurse**: Value=no

  New in version 3.2.

  - Do not recurse into the defined directory.

  Allowed: yes/no

`ignore`

List of files or directories to be ignored (one entry per element). The files and directories are still checked, but the results are ignored.

Default: /etc/mtab

Attributes:

- **type**: Value=sregex
  - This is a simple regex pattern to filter out files so alerts are not generated.

Allowed: Any directory or file name

`nodiff`

New in version 3.0.

List of files to not attach a diff. The files are still checked, but no diff is computed. This allows to monitor sensitive files like private key or database configuration without leaking sensitive data through alerts.

Attributes:

- **type**: Value=sregex
  - This is a simple regex pattern to filter out files so alerts are not generated.

Allowed: Any directory or file name

`frequency`

Frequency that the syscheck is going to be executed (in seconds).

The default is 6 hours or 21600 seconds

Default: 21600

Allowed: Time in seconds

`scan_time`

Time to run the scans (can be in the formats of 21pm, 8:30, 12am, etc).

Allowed: Time to run scan

Note: This may delay the initialization of realtime scans.

`scan_day`

Day of the week to run the scans (can be in the format of sunday, saturday, monday, etc)

Allowed: Day of the week

`auto_ignore`

Specifies if syscheck will ignore files that change too often (after the third change)

Default: yes

Allowed: yes/no

Valid: server, local

`alert_new_files`

Specifies if syscheck should alert on new files created.

Default: no

Allowed: yes/no

Valid: server, local

Note: New files will only be detected on a full scan, this option does not work in realtime.

`scan_on_start`

Specifies if syscheck should do the first scan as soon as it is started.

Default: yes

Allowed: yes/no

`windows_registry`

Use this option to add Windows registry entries to be monitored (Windows-only).

Default: HKEY_LOCAL_MACHINESoftware

Allowed: Any registry entry (one per element)

Note: New entries will not trigger alerts, only changes to existing entries.

`registry_ignore`

List of registry entries to be ignored.

Default: ..CryptographyRNG

Allowed: Any registry entry (one per element)

`prefilter_cmd`

Command to run to prevent prelinking from creating false positives.

Example:

```
<prefilter_cmd>/usr/sbin/prelink -y</prefilter_cmd>
```

Note: This option can potentially impact performance negatively. The configured command will be run for each and every file checked.

`skip_nfs`

New in version 2.9.0.

Specifies if syscheck should scan network mounted filesystems. Works on Linux and FreeBSD. Currently skip_nfs will abort checks running against CIFS or NFS mounts.

Default: no

Allowed: yes/no

Warning: This option was added in OSSEC 2.9.

Configuration Examples

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.

Example:

```
<syscheck>
    <directories check_all="yes">/etc,/usr/bin,/usr/sbin</directories>
    <directories check_all="yes">/root/users.txt,/bsd,/root/db.html</directories>
</syscheck>
```

Files and directories can be ignored using the `ignore` option (or `registry_ignore` for Windows registry entries):

```
<syscheck>
    <ignore>/etc/random-seed</ignore>
    <ignore>/root/dir</ignore>
    <ignore type="sregex">.log$|.tmp</ignore>
</syscheck>
```

The `type` attribute can be set to sregex to specify a *Regular Expression Syntax* in the ignore option.

```
<syscheck>
    <ignore type="sregex">^/opt/application/log</ignore>
</syscheck>
```

A local rule can be used to modify the severity for changes to specific files or directories:

```
<rule id="100345" level="12">
    <if_matched_group>syscheck</if_matched_group>
    <match>/var/www/htdocs</match>
    <description>Changes to /var/www/htdocs - Critical file!</description>
</rule>
```

In the above example, a rule was created to alert with high severity (12) for changes to the files in the htdocs directory.

Real time Monitoring

OSSEC supports realtime (continuous) file integrity monitoring on Linux (support was added kernel version 2.6.13) and Windows systems.

The configuration is very simple. In the `<directories>` option where you specify what directories to monitor, adding `realtime="yes"` will enable it. For example:

```
<syscheck>
    <directories realtime="yes" check_all="yes">/etc,/usr/bin,/usr/sbin</directories>
    <directories check_all="yes">/bin,/sbin</directories>
</syscheck>
```

In this case, the directories /etc, /usr/bin and /usr/sbin will be monitored in real time. The same applies to Windows too.

Warning: The real time monitoring will not start immediately. First ossec-syscheckd needs to scan the file system and add each sub-directory to the realtime queue. It can take a while for this to finish (wait for the log "ossec-syscheckd: INFO: Starting real time file monitoring").

Note: Real time only works with directories, not individual files. So you can monitor the /etc or C:\program files directory, but not an individual file like /etc/file.txt.

Note: Both rootcheck and syscheck runs on the same thread, so when rootcheck is running, the inotify events would get queued until it finishes.

Report Changes

OSSEC supports sending diffs when changes are made to text files on Linux and unix systems.

Configuring syscheck to show diffs is simple, add `report_changes="yes"` to the `<directories` option. For example:

```
<syscheck>
    <directories report_changes="yes" check_all="yes">/etc</directories>
    <directories check_all="yes">/bin,/sbin</directories>
</syscheck>
```

Note: Report Changes can only work with text files, and the changes are stored on the agent inside `/var/ossec/queue/diff/local/dir/file`.

If OSSEC has not been compiled with libmagic support, report_changes will copy any file designated, e.g. mp3, iso, executable, /chroot/dev/urandom (which would fill your hard drive). So unless libmagic is used, be very carefull on which directory you enable report_changes.

MD5 whitelist database

ossec-analysisd can query an sqlite database for known-good md5 hashes. The database should contain the hashes, file names, and an optional date.

This feature uses code from Xavier Mertens.

Configure the database in *ossec.conf*:

```
</global>
  <md5_whitelist>/rules/lists/md5whitelist.db</md5_whitelist>
</global>
```

Schema:

```
CREATE TABLE files (
    md5sum VARCHAR(32),
    file VARCHAR(256),
    time DATETIME
```

(continues on next page)

(continued from previous page)

```
);
CREATE UNIQUE INDEX files_idx ON files(md5sum);
```

Syscheck: FAQ

How to force an immediate syscheck scan?

Run agent control tool to perform a integrity checking immediately (option -a to run on all the agents and -u to specify an agent id)

```
# /var/ossec/bin/agent_control -r -a
# /var/ossec/bin/agent_control -r -u <agent_id>
```

For more information see the *agent_control* documentation.

How to tell syscheck not to scan the system when OSSEC starts?

Set the option <scan_on_start> to "no" on ossec.conf

How to ignore a file that changes too often?

Set the file/directory name in the <ignore> option or create a simple local rule.

The following one will ignore files /etc/a and /etc/b and the directory /etc/dir for agents mswin1 and ubuntu-dns:

```
<rule id="100345" level="0" >
    <if_group>syscheck</if_group>
    <description>Changes ignored.</description>
    <match>/etc/a|/etc/b|/etc/dir</match>
    <hostname>mswin1|ubuntu-dns</hostname>
</rule>
```

Why does OSSEC still scan a file even though it's been ignored?

No idea. So if there are some directories you do not want scanned at all, make sure they are not included in a `<directories>` configuration.

How to know when the syscheck scan ran?

Use the agent_control tool on the manager, to see this information.

More information see the *agent_control* documentation.

How to get detailed reporting on the changes?

Use the syscheck_control tool on the manager or the web ui for that.

More information see the *syscheck_control* documentation.

Syscheck not sending any file data to the server?

With ossec 1.3 and Fedora you may run into this problem:

You have named files you'd like ossec to monitor so you add:

```
<ossec_config>
    <syscheck>
        <directories check_all="yes">/var/named</directories>
```

to ossec.conf on the client. Fedora – at least as of version 7 – runs named in a chroot jail under /var/named/chroot. However, part of that chroot jail includes /var/named/chroot/proc. The contents of that directory are purely ephemeral; there is no value to checking their integrity. And, at least in ossec 1.3, your syscheck may stall trying to read those files.

The symptom is a syscheck database on the server that never grows beyond a file or two per restart of the client. The log monitoring continues to work, so you know it's not a communication issue, and you will often see a slight increase in syscheck database file size after the client has restarted (in one case about 20 minutes after). But the database will never be completely built; there will only be a couple files listed in database.

The solution is to add an ignore clause to ossec.conf on the client:

```
<ossec_config>
    <syscheck>
        <ignore>/var/named/chroot/proc</ignore>
```

Why aren't new files creating an alert?

By default OSSEC does not alert on new files. To enable this functionality, <alert_new_files> must be set to yes inside the <syscheck> section of the manager's ossec.conf. Also, the rule to alert on new files (rule 554) is set to level 0 by default. The alert level will need to be raised in order to see the alert. Alerting on new files does not work in realtime, a full scan will be necessary to detect them.

Add the following to local_rules.xml:

```
<rule id="554" level="10" overwrite="yes">
  <category>ossec</category>
  <decoded_as>syscheck_new_entry</decoded_as>
  <description>File added to the system.</description>
  <group>syscheck,</group>
</rule>
```

The `<alert_new_files>` entry should look something like this:

```
<syscheck>
  <frequency>7200</frequency>
  <alert_new_files>yes</alert_new_files>
  <directories check_all="yes">/etc,/bin,/sbin</directories>
</syscheck>
```

Can OSSEC include information on who changed a file in the alert?

In short, no. OSSEC does not track this information. You could use your OS's auditing facilities to track this information, and create a rule to alert when an appropriate log is created.

How do I stop syscheck alerts during system updates?

There is no easy way to do this, but there are work-arounds. Stop the OSSEC processes on the manager, and run `/var/ossec/bin/syscheck_control -u AGENT_ID`. This will clear the syscheck database for the agent, and the next time syscheck runs it will create a new baseline. Next, start the OSSEC processes on the manager. Once the system update is complete, run a syscheck scan on that agent. The database will be populated with new values, and should not trigger "file modified" alarms.

Rootcheck Manual

Rootcheck

OSSEC HIDS will perform rootkit detection on every system where the agent is installed. The rootcheck (rootkit detection engine) will be executed every X minutes (user specified - by default every 2 hours) to detect any possible rootkit installed. Used with the log analysis and the integrity checking engine, it will become a very powerful monitoring solution.

Checks that rootcheck performs

1. Read the rootkit_files.txt which contains a database of rootkits and files commonly used by them. It will try to stats, fopen and opendir each specified file. We use all these system calls because some kernel-level rootkits hide files from some system calls. The more system calls we try, the better the detection. This method is more like an anti-virus rule that needs to be updated constantly. The chances of false-positives are small, but false negatives can be produced by modifying the rootkits.
2. Read the rootkit_trojans.txt which contains a database of signatures of files trojaned by rootkits. This technique of modifying binaries with trojaned versions was commonly used by most of the popular rootkits available. This detection method will not find any kernel level rootkit or any unknown rootkit.
3. Scan the /dev directory looking for anomalies. The /dev should only have device files and the Makedev script. A lot of rootkits use the /dev to hide files. This technique can detect even non-public rootkits.

4. Scan the whole filesystem looking for unusual files and permission problems. Files owned by root, with write permission to others are very dangerous, and the rootkit detection will look for them. Suid files, hidden directories and files will also be inspected.
5. Look for the presence of hidden processes. We use getsid() and kill() to check if any pid is being used or not. If the pid is being used, but "ps" can't see it, it is the indication of kernel-level rootkit or a trojaned version of "ps". We also verify that the output of kill and getsid are the same.
6. Look for the presence of hidden ports. We use bind() to check every tcp and udp port on the system. If we can't bind to the port (it's being used), but netstat does not show it, we probably have a rootkit installed
7. Scan all interfaces on the system and look for the ones with "promisc" mode enabled. If the interface is in promiscuous mode, the output of "ifconfig" should show that. If not, we probably have a rootkit installed.

Configuration options

These configuration options can be specified in each agent's ossec.conf, except `auto_ignore` and `alert_new_file` which are manager side options. If the `ignore` option is specified on the manager the setting becomes global for all agents.

Understanding the Unix policy auditing on OSSEC

OSSEC's policy monitor allows you to verify that all your systems conform to a set of policies regarding configuration settings and applications usage. They are configured centrally on the ossec server and pushed down to the agents. It also checks if a system in in compliance with the CIS Security Benchmarks and VMware security hardening guidelines.

The following systems are tested for the CIS and VMware guidelines:

- Debian and Ubuntu
- Red Hat and Fedora
- Red Hat Enterprise Linux 5
- VMWare ESX 3.0 and 3.5

Receiving Audit and Application alerts via Email

By default, both the policy auditing and application checks are logged as level `3`, so you will not receive any e-mail alerts with the original configuration.

If you wish to receive e-mail alerts for any (or both of the two) types of events, you need to create local rules with a higher severity or with the `alert_by_email` option set.

Example1: Sending e-mail for every Audit event

Add to your local_rules.xml the following:

```
<pre>
  <rule id="512" level="9" overwrite="yes">
    <if_sid>510</if_sid>
    <match>^System Audit</match>
    <description>System Audit event.</description>
    <group>rootcheck,</group>
```

(continues on next page)

(continued from previous page)

```
  </rule>
</pre>
```

Listing entries per agent

To control the policy database, use the ‘‘rootcheck_control’‘ tool.

This page was originally authored by Daniel Cid for the OSSEC wiki.

Rules and Decoders

Testing OSSEC rules/decoders

The first problem most people have when troubleshooting OSSEC or trying to write new rules and decoders is how to test them. In the past, this would require manually restarting OSSEC or creating a testing installation. As of version 1.6, there is a tool to simplify this task (ossec-logtest).

Testing using ossec-logtest

The tool *ossec-logtest* is installed into `/var/ossec/bin`. It will read the current rules and decoder (from `/var/ossec`) and accept log input from stdin:

```
# /var/ossec/bin/ossec-logtest
2008/07/04 09:57:28 ossec-testrule: INFO: Started (pid: 12683).
ossec-testrule: Type one log per line.

Jul 4 09:42:16 enigma sshd[11990]: Accepted password for dcid from 192.168.2.10 port␣
↪35259 ssh2

**Phase 1: Completed pre-decoding.
full event: "Jul 4 09:42:16 enigma sshd[11990]: Accepted password for dcid from 192.
↪168.2.10 port 35259 ssh2"
hostname: "enigma"
program_name: "sshd"
log: "accepted password for dcid from 192.168.2.10 port 35259 ssh2"

**Phase 2: Completed decoding.
decoder: 'sshd'
dstuser: `dcid'
srcip: `192.168.2.10

**Phase 3: Completed filtering (rules).
Rule id: `10100
Level: `4
Description: `First time user logged in.'
**Alert to be generated.
```

In the above example, we provided an authentication success log and ossec-logtest showed us how it would be decoded, what information was extracted and which rule fired. In the next example, we can see how it would extract a user logoff message from Windows:

```
# /var/ossec/bin/ossec-logtest
2008/07/04 09:57:28 ossec-testrule: INFO: Started (pid: 12683).
ossec-testrule: Type one log per line.

WinEvtLog: Security: AUDIT_SUCCESS(538): Security: lac: OSSEC-HM: OSSEC-HM: User␣
↪Logoff: User Name: lac Domain: OSSEC-HM Logon ID: (0x0,0xF784D5) Logon Type: 2

**Phase 1: Completed pre-decoding.
full event: 'WinEvtLog: Security: AUDIT_SUCCESS(538): Security: lac: OSSEC-HM: OSSEC-
↪HM: User Logoff: User Name: lac Domain: OSSEC-HM Logon ID: (0x0,0xF784D5) Logon␣
↪Type: 2
hostname: 'enigma'
program_name: '(null)'
log: 'WinEvtLog: Security: AUDIT_SUCCESS(538): Security: lac: OSSEC-HM: OSSEC-HM:␣
↪User Logoff: User Name: lac Domain: OSSEC-HM Logon ID: (0x0,0xF784D5) Logon Type: 2

**Phase 2: Completed decoding.
decoder: 'windows'
status: 'AUDIT_SUCCESS'
id: '538
extra_data: 'Security'
dstuser: 'lac'
system_name: 'OSSEC-HM'

**Phase 3: Completed filtering (rules).
Rule id: '18149
Level: '3
Description: 'Windows User Logoff.'
**Alert to be generated.
```

In addition to the information above, ossec-logtest -v can be used to follow the log through the rule path:

```
# /var/ossec/bin/ossec-logtest -v
2008/07/04 10:05:43 ossec-testrule: INFO: Started (pid: 23007).
ossec-testrule: Type one log per line.

Jul 4 10:05:30 enigma sshd[27588]: Failed password for invalid user test2 from 127.0.
↪0.1 port 19130 ssh2

**Phase 1: Completed pre-decoding.
full event: 'Jul 4 10:05:30 enigma sshd[27588]: Failed password for invalid user␣
↪test2 from 127.0.0.1 port 19130 ssh2
hostname: 'enigma'
program_name: 'sshd'
log: 'Failed password for invalid user test2 from 127.0.0.1 port 19130 ssh2

**Phase 2: Completed decoding.
decoder: 'sshd'
srcip: '127.0.0.1

**Rule debugging:
Trying rule: 1 - Generic template for all syslog rules.
*Rule 1 matched.
*Trying child rules.
Trying rule: 5500 - Grouping of the pam_unix rules.
Trying rule: 5700 - SSHD messages grouped.
*Rule 5700 matched.
```

(continues on next page)

(continued from previous page)

```
*Trying child rules.
Trying rule: 5709 - Useless SSHD message without an user/ip.
Trying rule: 5711 - Useless SSHD message without a user/ip.
Trying rule: 5707 - OpenSSH challenge-response exploit.
Trying rule: 5701 - Possible attack on the ssh server (or version gathering).
Trying rule: 5706 - SSH insecure connection attempt (scan).
Trying rule: 5713 - Corrupted bytes on SSHD.
Trying rule: 5702 - Reverse lookup error (bad ISP or attack).
Trying rule: 5710 - Attempt to login using a non-existent user
*Rule 5710 matched.
*Trying child rules.
Trying rule: 5712 - SSHD brute force trying to get access to the system.

**Phase 3: Completed filtering (rules).
Rule id: '5710
Level: '5
Description: 'Attempt to login using a non-existent user'
**Alert to be generated.
```

CDB List lookups from within Rules

Allow for CDB lookups from within rules in OSSEC (ossec-analysisd) of all possible fields.

Use cases

Anything that has a large number of items. Some examples:

- named with recursive logs checking the www.malwaredomains.com list for suspicious domains
- lists of approved users by server
- mstark (on irc) originally came up with suggestion for approved software based on a md5 list
- IP address lookups - there are a large number of lists of suspicious or known bad IP addresses to match inside of ossec rules

Syntax for Lists

Rules

A rule would use the following syntax to look up a key within a CDB database.

Positive key match

This example is a search for the key within the `rules/cdb_record_file` and will match if they key is present:

```
<list field="program_name" lookup="match_key">rules/records</list>
```

The `lookup="match_key"` is the default and can be left out as in this example:

```
<list field="program_name">rules/records</list>
```

Negative key match

This example is a search for the key stored in field attribute and will match if it *IS NOT* present in the database:

```
<list field="program_name" lookup="not_match_key">rules/records</list>
```

Key and Value match

This example is a search for a 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="program_name" lookup="match_key_value" check_value="^reject">rules/
↪records</list>
```

Positive IP address match

This example is a search for the IP address stored in the field attribute and will match if it *IS* present in the database.

```
<list field="srcip" lookup="address_match_key">rules/records</list>
```

Negative IP address match

This example is a search for the IP address stored in the field attribute and will match if it *IS NOT* present in the database.

```
<list field="srcip" lookup="not_address_match_key">rules/records</list>
```

Key and Value Address Match

This example is a search for a 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="srcip" lookup="address_match_key_value" check_value="^reject">rules/
↪records</list>
```

ossec.conf

Each list will need to be defined and told to be available using the ossec.conf file. Using the following syntax:

```
<ossec_config>
    <rules>
        <list>rules/records</list>
```

Commands

CDB files must be compiled before they can be used. *ossec-makelists* is used to compile lists.

The command *ossec-makelists* will process and compile all lists if the master text rules have been changed. Basically logic is as follows:

- Read ossec.conf for all lists
- Check the mtime of each list and compare it to the mtime of the compiled .cdb file
- if mtime is newer create new database file ending in .tmp
- use atomic rename to change the .tmp to .cdb. This will invalidate all mmap pages currently in use by ossec-analysisd and will cause them to be reloaded with the new data as needed

List text file format

Creating cdb lists the following file format is specified:

```
key1:value
key2:value
key3:diff value
```

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
10.1.1.1    10.1.1.1/32      10.1.1.1
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
```

Due to address lookups being based on the class boundary extra scripts are suggested for creating lists that need fine control. Example of IP address list file:

```
192.168.: RFC 1918 Address space
172.16.:RFC 1918 Address space
172.17.:RFC 1918 Address space
172.18.:RFC 1918 Address space
172.19.:RFC 1918 Address space
172.20.:RFC 1918 Address space
172.21.:RFC 1918 Address space
172.22.:RFC 1918 Address space
172.23.:RFC 1918 Address space
172.24.:RFC 1918 Address space
172.25.:RFC 1918 Address space
172.26.:RFC 1918 Address space
172.27.:RFC 1918 Address space
172.28.:RFC 1918 Address space
172.29.:RFC 1918 Address space
172.30.:RFC 1918 Address space
172.31.:RFC 1918 Address space
10.:RFC 1918 Address space
```

Note: Previous versions of this page page originally was created by @j_hen on her blog http://jentalkstoomuch.blogspot.com/2010/09/writing-custom-ossec-rules-for-your.html Some content may be the same, but examples have been updated.

Note: In the xml based examples, any text between `<!--` and `-->` are comments. In the console based examples, anything after `#` may be an example. For more information on OSSEC's non-standard regular expression (regex) syntax, refer to the regex page.

Create Custom decoder and rules

One of the main features of OSSEC is monitoring system and application logs. Many popular services have logs and decoders, but there are hundreds that are not covered. Custom applications and services will also not be covered. Adding decoders and rules for services is generally very easy.

Adding a File to be Monitored

Adding a log file to the configuration for monitoring is simple. In the system's ossec.conf add an entry like this:

```
<localfile>
  <log_format>syslog</log_format>
  <location>/path/to/log/file</location>
</localfile>
```

`syslog` is a generic format, consisting of a singular line of text appended to the log file. There are other formats available, they are detailed on the localfile syntax page.

Note: Additional examples can be found here. More detailed syntax can be found here.

After adding a localfile entry, the OSSEC processes must be restarted.

Create a Custom Decoder

The following log messages will be used for most of the examples in this section:

```
2013-11-01T10:01:04.600374-04:00 arrakis ossec-exampled[9123]: test connection from␣
↪192.168.1.1 via test-protocol1
2013-11-01T10:01:05.600494-04:00 arrakis ossec-exampled[9123]: successful␣
↪authentication for user test-user from 192.168.1.1 via test-protocol1
```

The first log message is broken down as follows:

- 2013-11-01T10:01:04.600374-04:00 - timestamp from rsyslog
- arrakis - hostname of the system
- ossec-exampled - daemon creating the log
- [9123] - process ID of the ossec-exampled instance
- test connection from 192.168.1.1 via test-protocol1 - log message

ossec-logtest will be used to test the custom decoder and any custom rules.

Custom decoders are added to the `local_decoder.xml` file, typically found in `/var/ossec/etc` on a standard installation. The basic syntax is listed here, but this page is not well documented at the moment.

Using ossec-logtest on this sample rule results in the following output:

```
# /var/ossec/bin/ossec-logtest
2013/11/01 10:39:07 ossec-testrule: INFO: Reading local decoder file.
2013/11/01 10:39:07 ossec-testrule: INFO: Started (pid: 32109).
ossec-testrule: Type one log per line.

2013-11-01T10:01:04.600374-04:00 arrakis ossec-exampled[9123]: test connection from␣
↪192.168.1.1 via test-protocol1


**Phase 1: Completed pre-decoding.
       full event: '2013-11-01T10:01:04.600374-04:00 arrakis ossec-exampled[9123]:␣
↪test connection from 192.168.1.1 via test-protocol1'
       hostname: 'arrakis'
       program_name: 'ossec-exampled'
       log: 'test connection from 192.168.1.1 via test-protocol1'

**Phase 2: Completed decoding.
       No decoder matched.
```

There is not a lot of output here because OSSEC does not understand this log. Creating a decoder for this log message will provide OSSEC much more information.

Phase 1 "pre-decodes" some information. The hostname is the system that generated the log message, program_name is the name of the application that created the log, and log is the rest of the log message.

The following is a very basic decoder for `ossec-exampled`:

```
<decoder name="ossec-exampled">
  <program_name>ossec-exampled</program_name>
</decoder>
```

This decoder simply looks for any log messages generated by `ossec-exampled`. Using a very generic decoder like this can allow an OSSEC user to create more specific child decoders for services with less consistent log messages.

Here is the ossec-logtest output after adding this decoder:

```
# /var/ossec/bin/ossec-logtest
2013/11/01 10:52:09 ossec-testrule: INFO: Reading local decoder file.
2013/11/01 10:52:09 ossec-testrule: INFO: Started (pid: 25151).
ossec-testrule: Type one log per line.

2013-11-01T10:01:04.600374-04:00 arrakis ossec-exampled[9123]: test connection from␣
↪192.168.1.1 via test-protocol1


**Phase 1: Completed pre-decoding.
       full event: '2013-11-01T10:01:04.600374-04:00 arrakis ossec-exampled[9123]:␣
↪test connection from 192.168.1.1 via test-protocol1'
       hostname: 'arrakis'
       program_name: 'ossec-exampled'
       log: 'test connection from 192.168.1.1 via test-protocol1'

**Phase 2: Completed decoding.
       decoder: 'ossec-exampled'
```

Phase 2 now correctly identifies this log message as coming from ossec-exampled. There is still some very important information in the log message that should be decoded, namely the source IP and `test-protocol1`. To decode these a child decoder will be added. It will set the `ossec-exampled` decoder as a parent, and use `prematch` to limit its use to the correct log message.

```
<decoder name="ossec-exampled-test-connection">
  <parent>ossec-exampled</parent>
  <prematch offset="after_parent">^test connection </prematch> <!-- offset="after_
↪parent" makes OSSEC ignore anything matched by the parent decoder and before -->
  <regex offset="after_prematch">^from (\S+) via (\S+)$</regex> <!-- offset="after_
↪prematch" makes OSSEC ignore anything matched by the prematch and earlier-->
  <order>srcip, protocol</order>
</decoder>
```

Breaking this down piece by piece:

- `<decoder name="ossec-exampled-test-connection">` - Declaring this to be a decoder and giving it a name.
- `<parent>ossec-exampled</parent>` - This decoder will only be checked if `ossec-exampled` also matched.
- `<prematch offset="after_parent">^test connection </prematch>` - If a log message does not contain the data in the prematch, it will not use that decoder. Setting the offset tells OSSEC to only look at data after the parent (ossec-exampled[9123]: in this case), in an effort to speed up matches.
- `<regex offset="after_prematch">^from (\S+) via (\S+)$</regex>` - The regex line can be used to pull data out of the log message for use in rules. In this instance the first `\S+` matches the IP address, and the second matches the protocol. Anything between the parenthesis will be able to be used in rules.
- `<order>srcip, protocol</order>` - Defines what the entries in the regex line are labeled as. The IP address will be labeled as srcip, and the protocol by proto.

ossec-logtest output after adding this decoder:

```
# /var/ossec/bin/ossec-logtest
2013/11/01 11:03:25 ossec-testrule: INFO: Reading local decoder file.
2013/11/01 11:03:25 ossec-testrule: INFO: Started (pid: 6290).
ossec-testrule: Type one log per line.

2013-11-01T10:01:04.600374-04:00 arrakis ossec-exampled[9123]: test connection from
↪192.168.1.1 via test-protocol1


**Phase 1: Completed pre-decoding.
       full event: '2013-11-01T10:01:04.600374-04:00 arrakis ossec-exampled[9123]:
↪test connection from 192.168.1.1 via test-protocol1'
       hostname: 'arrakis'
       program_name: 'ossec-exampled'
       log: 'test connection from 192.168.1.1 via test-protocol1'

**Phase 2: Completed decoding.
       decoder: 'ossec-exampled'
       srcip: '192.168.1.1'
       proto: 'test-protocol1'
```

Note: The `decoder` will be labeled as the parent decoder, not the child. It's common to think a child decoder doesn't work because the parent decoder's name is listed, but that may not be a problem.

Now that the first sample log message is decoded, how does the second message fare? `ossec-logtest` output:

```
2013-11-01T10:01:05.600494-04:00 arrakis ossec-exampled[9123]: successful␣
↪authentication for user test-user from 192.168.1.1 via test-protocol1


**Phase 1: Completed pre-decoding.
       full event: '2013-11-01T10:01:05.600494-04:00 arrakis ossec-exampled[9123]:␣
↪successful authentication for user test-user from 192.168.1.1 via test-protocol1'
       hostname: 'arrakis'
       program_name: 'ossec-exampled'
       log: 'successful authentication for user test-user from 192.168.1.1 via test-
↪protocol1'

**Phase 2: Completed decoding.
       decoder: 'ossec-exampled'
```

The decoded fields added in `ossec-exampled-test-connection` do not get decoded in this log message. This is expected because the `prematch` does not match. In this log message there are 4 fields that would be useful: status (successful), srcuser, srcip, and protocol. Adding a decoder for this should also be simple:

```
<decoder name="ossec-exampled-auth">
  <parent>ossec-exampled</parent>
  <prematch offset="after_parent"> authentication </prematch>
  <regex offset="after_parent">^(\S+) authentication for user (\S+) from (\S+) via␣
↪(\S+)$</regex> <!-- Using after_parent here because after_prematch would eliminate␣
↪the possibility of matching the status (successful) -->
  <order>status, srcuser, srcip, protocol</order>
</decoder>
```

`ossec-logtest` output:

```
2013-11-01T10:01:05.600494-04:00 arrakis ossec-exampled[9123]: successful␣
↪authentication for user test-user from 192.168.1.1 via test-protocol1


**Phase 1: Completed pre-decoding.
       full event: '2013-11-01T10:01:05.600494-04:00 arrakis ossec-exampled[9123]:␣
↪successful authentication for user test-user from 192.168.1.1 via test-protocol1'
       hostname: 'arrakis'
       program_name: 'ossec-exampled'
       log: 'successful authentication for user test-user from 192.168.1.1 via test-
↪protocol1'

**Phase 2: Completed decoding.
       decoder: 'ossec-exampled'
       status: 'successful'
       srcuser: 'test-user'
       srcip: '192.168.1.1'
       proto: 'test-protocol1'
```

Now the useful fields have been extracted for this log message as well. Double checking the original log message, to make sure there were no regressions:

```
2013-11-01T10:01:04.600374-04:00 arrakis ossec-exampled[9123]: test connection from␣
↪192.168.1.1 via test-protocol1


**Phase 1: Completed pre-decoding.
```

(continues on next page)

(continued from previous page)

```
       full event: '2013-11-01T10:01:04.600374-04:00 arrakis ossec-exampled[9123]:␣
↪test connection from 192.168.1.1 via test-protocol1'
       hostname: 'arrakis'
       program_name: 'ossec-exampled'
       log: 'test connection from 192.168.1.1 via test-protocol1'

**Phase 2: Completed decoding.
       decoder: 'ossec-exampled'
       srcip: '192.168.1.1'
       proto: 'test-protocol1'
```

Historical

Previous versions of this page page originally was created by @j_hen on her blog http://jentalkstoomuch.blogspot.com/2010/09/writing-custom-ossec-rules-for-your.html This blog no longer exists (at least at that location).

Directory path loading of rules and decoders

To allow whole directories of files to be loaded as decoders, lists, or rules by ossec-anaylistd.

Use case

Greatly simplifies working with decoders as there can be as many files as needed. Also will make packaging of rules and decoders a simple unzip/untar and restart operation. This will also greatly reduce the amount of code needed to manage the upgrade scripts of ossec.

Details

Syntax for OSSEC

All Directory loading is done in alphabetical form. This is much like init.d where the use of numeric prefixes on file names can affect the order of loading. Example of file names and the order they would be loaded:

1. 00_sshd_rules.xml
2. 01_local_sshd_rules.xml
3. 99_shun_rules.xml

Directory loading

The basic format for selection of rules file is as follows. This will load all files in the rules dir that match the regex `_rules.xml$`.

```
<ossec_config>
    <rules>
        <rule_dir pattern="_rules.xml">rules</rule_dir>
```

The pattern is optional and defaults to _rules.xml for rules loading so this could be written as:

```
<ossec_config>
    <rules>
        <rule_dir>rules</rule_dir>
```

Order of the directives in ossec.conf is still respected, and duplicate files will not be loaded. In the following example 00_setup_rules.xml is always loaded first, and will *NOT* be loaded a second time by the rule_dir directive.

```
<ossec_config>
    <rules>
        <include>rules/00_setup_rules.xml</include>
        <rule_dir>rules</rule_dir>
```

For full details on all the Syntax see **:xml:‘rule_dir‘** and **:xml:‘decoder_dir‘**

Compete Examples of syntax

This is an example where the decoders and rules have been broken out into subdirectories.

- rules/
  - 00_rules_config.xml
  - 50_apache_rules.xml
  - 50_arpwatch_rules.xml
  - plugins/
    - 50_wimax_rules.xml
    - 50_wimax_decoders.xml
- etc/
  - decoder.xml
  - local_decoder.xml

```
<ossec_config>
    <rules>
        <decoder>etc/decoder.xml</decoder>
        <decoder_dir>rules/plugins</decoder_dir>

        <rule>rules/rules/00_rules_config.xml</rule>
        <rule_dir pattern=".xml$">rules/</rule_dir>
        <rule_dir>rules/plugins</rule_dir>
    </rules>
</ossec_config>
```

Rules Classification

The rules are classified in multiple levels. From the lowest (00) to the maximum level 16. Some levels are not used right now. Other levels can be added between them or after them.

The rules will be read from the highest to the lowest level.

00 - Ignored - No action taken. Used to avoid false positives. These rules are scanned before all the others. They include events with no security relevance.

01 - None -

02 - System low priority notification - System notification or status messages. They have no security relevance.

03 - Successful/Authorized events - They include successful login attempts, firewall allow events, etc.

04 - System low priority error - Errors related to bad configurations or unused devices/applications. They have no security relevance and are usually caused by default installations or software testing.

05 - User generated error - They include missed passwords, denied actions, etc. By itself they have no security relevance.

06 - Low relevance attack - They indicate a worm or a virus that have no affect to the system (like code red for apache servers, etc). They also include frequently IDS events and frequently errors.

07 - "Bad word" matching. They include words like "bad", "error", etc. These events are most of the time unclassified and may have some security relevance.

08 - First time seen - Include first time seen events. First time an IDS event is fired or the first time an user logged in. If you just started using OSSEC HIDS these messages will probably be frequently. After a while they should go away, It also includes security relevant actions (like the starting of a sniffer or something like that).

09 - Error from invalid source - Include attempts to login as an unknown user or from an invalid source. May have security relevance (specially if repeated). They also include errors regarding the "admin" (root) account.

10 - Multiple user generated errors - They include multiple bad passwords, multiple failed logins, etc. They may indicate an attack or may just be that a user just forgot his credentials.

11 - Integrity checking warning - They include messages regarding the modification of binaries or the presence of rootkits (by rootcheck). If you just modified your system configuration you should be fine regarding the "syscheck" messages. They may indicate a successful attack. Also included IDS events that will be ignored (high number of repetitions).

12 - High importancy event - They include error or warning messages from the system, kernel, etc. They may indicate an attack against a specific application.

13 - Unusual error (high importance) - Most of the times it matches a common attack pattern.

14 - High importance security event. Most of the times done with correlation and it indicates an attack.

15 - Severe attack - No chances of false positives. Immediate attention is necessary.

Rules Group

We can specify groups for specific rules. It's used for active response reasons and for correlation.

We currently use the following groups:

- invalid_login
- authentication_success
- authentication_failed
- connection_attempt
- attacks
- adduser
- sshd
- ids
- firewall

- squid
- apache
- syslog

Output and Alert options

Contents:

Sending alerts via syslog

Syslog output allows an OSSEC manager to send the OSSEC alerts to one or more syslog servers. Because OSSEC only sends the alerts via syslog, these options are for server or local installations only.

OSSEC also supports sending alerts via cef, json, and to Splunk.

Configuration options

These configurations options require a server or local installation.

Enabling Syslog output

An OSSEC server can be configured to send the alerts via syslog. In this example all alerts are sent to 192.168.4.1, and alerts of level 10 and above are also sent to 10.1.1.1:

```
<ossec_config>
  ...

  <syslog_output>
    <server>192.168.4.1</server>
  </syslog_output>

  <syslog_output>
    <level>10</level>
    <server>10.1.1.1</server>
  </syslog_output>

  ...
</ossec_config>
```

After this change is made, the client-syslog process should be enabled:

```
# /var/ossec/bin/ossec-control enable client-syslog
```

And finally restart the OSSEC processes:

```
# /var/ossec/bin/ossec-control restart
```

ossec-csyslog should start along with the other OSSEC processes:

```
Starting OSSEC HIDS v2.8 (by Trend Micro Inc.)...
...
Started ossec-csyslogd...
...
```

And in the logs:

```
# tail -n 1000 /var/ossec/logs/ossec.log | grep csyslog
2008/07/25 12:55:16 ossec-csyslogd: INFO: Started (pid: 19412).
2008/07/25 12:55:16 ossec-csyslogd: INFO: Forwarding alerts via syslog to: '192.168.4.
↪1:514.
2008/07/25 12:55:16 ossec-csyslogd: INFO: Forwarding alerts via syslog to: '10.1.1.
↪1:514.
```

Here is an example of what the listening syslog daemon should receive (every log separated by level, rule, location and the actual event that generated it):

```
Jul 25 12:17:41 enigma ossec: Alert Level: 3; Rule: 5715 - SSHD authentication␣
↪success.; Location: (jul) 192.168.2.0->/var/log/messages;
srcip: 192.168.2.190; user: root; Jul 25 13:26:24 slacker sshd[20440]: Accepted␣
↪password for root from 192.168.2.190 port 49737 ssh2
```

Send all alerts to 10.10.10.125:

```
<syslog_output>
  <server>10.10.10.125</server>
</syslog_output>
```

Send all alerts to 10.10.10.126 in CEF:

```
<syslog_output>
  <server>10.10.10.126</server>
  <format>cef</format>
</syslog_output>
```

Send all alerts level 6 and above to 10.10.10.127 on port 515:

```
<syslog_output>
  <server>10.10.10.127</server>
  <port>515</port>
  <level>6</level>
</syslog_output>
```

Sending alerts via E-Mail

There are currently three types of email alerts:

- Single Notification E-Mail addresses
- Granular Notifications to any number of E-mail addresses

- Daily E-mail Reports

Warning: Single E-Mail Notification must be setup before Granular Notification will work.

Alerts to a single E-Mail Address

In order to send notifications to a single address three items need to be setup within ossec.conf

Global E-Mail address destination

The destination email address and mail host should be configured inside the <global> section of the /var/ossec/etc/ossec.conf.

```
<ossec_config>
    <global>
        <email_notification>yes</email_notification>
        <email_to>me@example.com</email_to>
        <smtp_server>mx.example.com..</smtp_server>
        <email_from>ossec@example.com</email_from>
```

Full details on all the options are available at *ossec.conf: Global options*

Note: If the *smtp_server* entry contains a hostname, */etc/resolv.conf* will probably have to be copied to OSSEC's *etc* directory (*/var/ossec/etc* by default).

Set the alert levels that will send notifications

The minimum email_alert_level can be set inside the <alerts> section of the /var/ossec/etc/ossec.conf file.

```
<ossec_config>
    <alerts>
        <email_alert_level>10</email_alert_level>
```

Full details on all the options are available at *ossec.conf: Alerts Options*

Restart OSSEC to complete the changes

OSSEC needs to be restarted for the change to take effect.

```
# /var/ossec/bin/ossec-control restart
```

Granular E-Mail alerts to many E-Mail addresses

OSSEC allows very granular options for the e-mail alerting and its format (full or SMS).

Note: Note that there must be at least one <email_to> recipient mentioned in the <global> section of the configuration or no emails will be sent at all.

Example 1: Group alerts

If you want to e-mail *xx@y.z* for every event in the group syslog you can add the following to ossec

```
<email_alerts>
    <email_to>xx@y.z</email_to>
    <group>syslog</group>
</email_alerts>
```

Example 2: Message Format

To e-mail (in the SMS format) *aa@y.z* for every event with severity higher than 10

Note: Note that the SMS format is not grouped, so the e-mail is sent immediately).

```
<email_alerts>
    <email_to>aa@y.z</email_to>
    <level>10</level>
    <format>sms</format>
</email_alerts>
```

Example 3: Email based on Rule ID's

To e-mail *bb@y.z* for every event from rule 123 or rule 124 (without grouping):

```
<email_alerts>
    <email_to>bb@y.z</email_to>
    <rule_id>123, 124</rule_id>
    <do_not_delay />
    <do_not_group />
</email_alerts>
```

Example 4: Email based on severity and agent

To e-mail *cc@y.z* for every event with severity higher than 12, from agent qwert or agt1, without any delay (immediately):=====

```
<email_alerts>
    <email_to>cc@y.z</email_to>
    <level>12</level>
    <event_location>qwerty|agt1</event_location>
    <do_not_delay />
</email_alerts>
```

Example 5: Multiple granular options together

You can have as many granular options as you want. In this example, we want the following:

- Email cc@y.z for every alert from agents qwerty and agt1
- Email john@y.z for every alert from agent secsys, lowsys and aixsys
- Email mike@y.z for every alert from /log/secure (from any agent)
- Email l@y.z for every alert from 192.168.0.0/24 network
- Email boss@y.z for every alert above level 10.

```
<ossec_config>
    <email_alerts>
        <email_to>cc@y.z</email_to>
        <event_location>qwerty|agt1</event_location>
    </email_alerts>

    <email_alerts>
        <email_to>john@y.z</email_to>
        <event_location>secsys|lowsys|aixsys</event_location>
    </email_alerts>

    <email_alerts>
        <email_to>mike@y.z</email_to>
        <event_location>/log/secure$</event_location>
    </email_alerts>

    <email_alerts>
        <email_to>l@y.z</email_to>
        <event_location>192.168.</event_location>
    </email_alerts>

    <email_alerts>
        <email_to>boss@y.z</email_to>
        <level>12</level>
    </email_alerts>
</ossec_config>
```

Daily E-Mail Reports

Daily E-Mail reports are summaries of the OSSEC alerts for the day.

Configuration options

All of these configuration options should be specified in the /var/ossec/etc/ossec.conf.

Receive a summary of all authentication success alerts

The following example will send a daily report of all authentication_success alerts, sorted by the related field srcip.

```
<ossec_config>
    <reports>
        <category>authentication_success</category>
        <user type="relation">srcip</user>
        <title>Daily report: Successful logins</title>
        <email_to>me@example.com</email_to>
```

Receive summary of all File integrity monitoring alerts

The following example will send a report of all events related to syscheck.

```
<ossec_config>
    <reports>
        <category>syscheck</category>
        <title>Daily report: File changes</title>
        <email_to>me@example.com</email_to>
```

Storing alerts as JSON

Note: This feature first appeared in OSSEC 2.9.

Sometimes you want to easily consume OSSEC alerts in other programs. With the json output, you can write alerts as a newline separated json file which other programs can easily consume.

For example, you can pair OSSEC with logstash-forwarder to effortlessly export your alerts to logstash, elasticsearch, and kibana (ELK).

This can provide the simplest method of exporting the entire alert message to other programs without any limitations or dependencies. For example the syslog output can write json, but is limited to the maximum syslog message size and excludes the full_log information like integrity checking diffs. You also do not need syslog, zeromq or any other dependencies.

Configuration

Turning it on or off is easy as setting a single configuration parameter in the ossec.conf. These configurations options require a server or local installation.

Enabling json output

An OSSEC server can be configured to write the alerts in json format.

File: /var/ossec/etc/ossec.conf

```
<ossec_config>
  <global>
    <jsonout_output>yes</jsonout_output>
    ...
  </global>
  ...
</ossec_config>
```

After this change is made, the alerts are written to alerts.json side-by-side with the legacy alerts.log file.

```
# tail /var/ossec/logs/alerts/alerts.json
```

You will still have the legacy alerts.log and any custom log formats you've created will remain. The files are md5 and sha1 checksummed and compressed once daily (just like the legacy alerts.log).

That's it.

Sending output to a Database

OSSEC supports MySQL and PostgreSQL database outputs.

Configuration options

These configurations options can be specified in the server or local install ossec.conf file.

Enabling Database Support

You must have the MySQL or PgSQL Client libraries installed on the OSSEC server. Typically something like

```
Ubuntu
# apt install mysql-server libmysqld-dev
  or
# apt install postgresql libpq-dev

RedHat / CentOS
# yum install mysql-devel
  or
# yum install postgresql-devel
```

You then need to set the DATABASE environment variable and run the "./install.sh" script, to compile OSSEC with the appropriate database support.

```
# DATABASE=mysql ./install.sh
  or
# DATABASE=pgsql ./install.sh
```

Enable Database output in the configuration

After installation is complete database support needs to be enabled. The following command will enable the database daemon on the next restart.

```
# /var/ossec/bin/ossec-control enable database
```

Database Specific Setup

Configuring MySQL

Database Setup

Create a database, setup the database user, and add the schema (located in the `src/os_dbd` directory of the distribution) with the following commands.

```
# mysql -u root -p

mysql> create database ossec;

mysql> grant INSERT,SELECT,UPDATE,CREATE,DELETE,EXECUTE on ossec.* to ossecuser@
↪<ossec ip>;
Query OK, 0 rows affected (0.00 sec)

mysql> set password for ossecuser@<ossec ip>=PASSWORD('ossecpass');
Query OK, 0 rows affected (0.00 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)

mysql> quit

# mysql -u root -p ossec < mysql.schema
```

OSSEC Setup

In order for ossec to output alerts and other data into the database the /var/ossec/etc/ossec.conf will need to have a <database_output> section added.

```
<ossec_config>
    <database_output>
        <hostname>192.168.2.30</hostname>
        <username>ossecuser</username>
        <password>ossecpass</password>
        <database>ossec</database>
        <type>mysql</type>
    </database_output>
</ossec_config>
```

The values will need to be corrected for your installations hostname, mysql user, password, and database.

Complete MySQL Output

All that is left is to enable the database daemon and restart ossec for the changes to take effect.

```
# /var/ossec/bin/ossec-control enable database
# /var/ossec/bin/ossec-control restart
```

Configuring PgSQL

Database Setup

Create a user for OSSEC within PgSQL

```
$ sudo -u postgres createuser -D -A -P ossec_user
Enter password for new role:
Enter it again:
Shall the new role be allowed to create more new roles? (y/n) n
CREATE ROLE
```

Create a database for OSSEC

```
$ sudo -u postgres createdb -O ossec_user ossecdb
CREATE DATABASE
```

Create the necessary tables from the PostgreSQL schema located in the `src/os_dbd` directory of the distribution.

```
$ psql -h 127.0.0.1 -U ossec_user -d ossecdb -f postgresql.schema
```

OSSEC Setup

In order for ossec to output alerts and other data into the database the /var/ossec/etc/ossec.conf will need to be updated and a <database_output> section will need to be added.

```
<ossec_config>
    <database_output>
        <hostname>192.168.2.30</hostname>
        <username>ossecuser</username>
        <password>ossecpass</password>
        <database>ossec</database>
        <type>postgresql</type>
    </database_output>
</ossec_config>
```

The values will need to be corrected for your installation's hostname, postgresql user, password, and database.

Complete PgSQL Output

All that is left is to enable the database daemon and restart ossec for the changes to take effect.

```
# /var/ossec/bin/ossec-control enable database
# /var/ossec/bin/ossec-control restart
```

Sending output to prelude

Prelude is a Hybrid IDS that uses IDMEF to receive alert information from external devices. If you are a Prelude user and wish to send your OSSEC alerts to Prelude, do the following:

Enabling Prelude Support

Note: You must have the Prelude libraries installed on the OSSEC server.

Before you run the "./install.sh" script execute the following to compile OSSEC with prelude support.

```
# cd ossec-hids-*
# cd src; make setprelude; cd ..
# ./install.sh
```

Enable Prelude output in the configuration

Just add the following entry to your ossec.conf inside the <global> section.

```
<prelude_output>yes</prelude_output>
```

Prelude extra options

You can define your own profile and set the log level from which you can send alerts to prelude with those parameters. Once again in the <global> section.

```
<prelude_profile>MyOssecProfile</prelude_profile>
<prelude_log_level>6</prelude_log_level>
```

Overview:

OSSEC includes a number of ways to send alerts to other systems or applications. Syslog, email, and sending the alerts to an SQL database are the typical methods. These output methods send only alerts, not full log data. Since the agents do not generate alerts, these options are server side only.

Active Response

The Active Response feature within OSSEC can run applications on an agent or server in response to certain triggers. These triggers can be specific alerts, alert levels, or rule groups.

The active response framework is also what allows an OSSEC administrator to start a syscheck scan or restart OSSEC on a remote agent.

Creating Customized Active Responses

OSSEC by default comes with a few active response scripts, but if you ever need to expand them, this tutorial can be of help.

As always, learning via examples is easier and faster. We will write a simple active response script to e-mail the alert to a specific address.

Creating the command

The first thing we need to do is to create a new “command” entry in the ossec config.

```
<command>
    <name>mail-test</name>
    <executable>mail-test.sh</executable>
```

(continues on next page)

(continued from previous page)

```
    <timeout_allowed>no</timeout_allowed>
    <expect />
</command>
```

Since our script does not need a timeout, we set it to no. We also don't expect any input (like srcip or username), so we leave the "expect" tag empty. In the executable tag, we specify the name of the script to be executed (it must be inside /var/ossec/active-response/bin/).

Note: If you do need a srcip or username, just add it, eg: <expect>srcip</expect>

Configure the Active response

Next, we need to configure ossec to run the active response. In my case, I want to run it on the ossec server (so I choose location server) and every time the rule 1002 is fired (see rules_id 1002). You can also specify the level or different locations.

```
<active-response>
    <command>mail-test</command>
    <location>server</location>
    <rules_id>1002</rules_id>
</active-response>
```

Create active response script

With that done, we can create the active response script. The mail-test.sh must be inside the /var/ossec/active-response/bin/ with the execution permissions set.

What are the arguments are passed to the script?

1. action (delete or add)
2. user name (or - if not set)
3. src ip (or - if not set)
4. Alert id (uniq for every alert)
5. Rule id
6. Agent name/host
7. Filename

```
#!/bin/sh
# E-mails an alert - copy at /var/ossec/active-response/bin/mail-test.sh
# Change e-mail ADDRESSS
# Author: Daniel Cid

MAILADDRESS="xx@ossec.net"
ACTION=$1
USER=$2
IP=$3
ALERTID=$4
RULEID=$5
```

(continues on next page)

(continued from previous page)

```
LOCAL=`dirname $0`;
cd $LOCAL
cd ../
PWD=`pwd`


# Logging the call
echo "`date` $0 $1 $2 $3 $4 $5 $6 $7 $8" >> ${PWD}/../logs/active-responses.log


# Getting alert time
ALERTTIME=`echo "$ALERTID" | cut -d  "." -f 1`

# Getting end of alert
ALERTLAST=`echo "$ALERTID" | cut -d  "." -f 2`

# Getting full alert
grep -A 10 "$ALERTTIME" ${PWD}/../logs/alerts/alerts.log | grep -v ".$ALERTLAST: " -A␣
↪10 | mail $MAILADDRESS -s "OSSEC Alert"
```

Restart OSSEC and test

After the configuration is done, you can restart OSSEC and test the configuration. For thee above example, I can run the logger command to similar a segmentation fault message.

```
# /var/ossec/bin/ossec-control restart
# logger "Segmentation Fault"
```

You should get in the /var/ossec/logs/active-response.log, the following:

```
Fri Jul 27 23:48:31 BRT 2007 /var/ossec/active-response/bin/mail-test.sh add - -␣
↪1185590911.25916 1002 /var/log/messages
```

And in your e-mail:

```
from: root <root@xx.org>
to: xx@ossec.net
date: Jul 27,27 2007 11:48 PM
subject: OSSEC Alert

** Alert 1185590911.25661: mailsl  - syslog,errors,
2007 Jul 27 23:48:31 xx->/var/log/messages
Rule: 1002 (level 7) -> 'Unknown problem somewhere in the system.'
Src IP: (none)
User: (none)
Jul 27 23:48:30 xx dcid: Segmentation Fault 123
```

UNIX: Active Response Configuration

The Active response configuration is divided into two parts. In the first one you configure the commands you want to execute. In the second one, you bind the commands to rules or events.

Commands Configuration

In the commands configuration you create new "commands" to be used as responses. You can have as many commands as you want. Each one should be inside their own "command" element. For further information please see the examples.

```
<command>
    <name>The name (A-Za-Z0-9)</name>
    <executable>The command to execute (A-Za-z0-9.-)</executable>
    <expect>Comma separated list of arguments (A-Za-z0-9)</expect>
    <timeout_allowed>yes/no</timeout_allowed>
</command>
```

- **name**: Used to link the command to the response.
- **executable**: It must be a file (with exec permissions) inside "/var/ossec/active-response/bin".

  You don't need to provide the whole path.
- **expect**: The arguments this command is expecting (options are srcip and username).
- **timeout_allowed**: Specifies if this command supports timeout.

Responses Configuration

In the active-response configuration, you bind the commands (created) to events. You can have as many responses as you want. Each one should be inside their own "active-response" element. For further information please see the ' <../../syntax/head_ossec_config.active-response.html#example-active-response-con figurations>'_.

```
<active-response>
    <disabled>Completely disables active response if "yes"</disabled>
    <command>The name of any command already created</command>
    <location>Location to execute the command</location>
    <agent_id>ID of an agent (when using a defined agent)</agent_id>
    <level>The lower level to execute it (0-9)</level>
    <rules_id>Comma separated list of rules id (0-9)</rules_id>
    <rules_group>Comma separated list of groups (A-Za-z0-9)</rules_group>
    <timeout>Time to block</timeout>
</active-response>
```

- **disabled**: Disables the active response capabilities if set to yes. If this is set, active response will not work.
- **command**: Used to link the response to the command
- **location**: Where the command should be executed. You have four options:
  - **local**: on the agent that generated the event
  - **server**: on the OSSEC server
  - **defined-agent**: on a specific agent (when using this option, you need to set the agent_id to use)
  - **all**: or everywhere.
- **agent_id**: The ID of the agent to execute the response (when defined-agent is set).
- **level**: The response will be executed on any event with this level or higher.
- **timeout**: How long until the reverse command is executed (IP unblocked, for example).

Active Response Tools

By default, the ossec hids comes with the following pre-configured active-response tools:

- **host-deny.sh**: Adds an IP to the /etc/hosts.deny file (most Unix systems).
- **firewall-drop.sh** (iptables): Adds an IP to the iptables deny list (Linux 2.4 and 2.6).
- **firewall-drop.sh** (ipfilter): Adds an IP to the ipfilter deny list (FreeBSD, NetBSD and Solaris).
- **firewall-drop.sh** (ipfw): Adds an IP to the ipfw deny table (FreeBSD).

  Note: On IPFW we use the table 1 to add the IPs to be blocked. We also set this table as deny in the beginning of the firewall list. If you use the table 1 for anything else, please change the script to use a different table id.

- **firewall-drop.sh** (ipsec): Adds an IP to the ipsec drop table (AIX).
- **pf.sh** (pf): Adds an IP to a pre-configured pf deny table (OpenBSD and FreeBSD).

  Note: On PF, you need to create a table in your config and deny all the traffic to it. Add the following lines at the beginning of your rules and reload pf (pfctl -F all && pfctl -f /etc/pf.conf): table <ossec_fwtable> persist #ossec_fwtable

  block in quick from <ossec_fwtable> to any block out quick from any to <ossec_fwtable>

- **firewalld-drop.sh** (firewalld): Adds a rich-rule to block an IP to firewalld (Linux with firewalld enabled).

  Note: You must manually enable this script in ossec.conf if you have firewalld enabled. The script will add (and remove) a rich-rule that drops all incoming communication from the supplied srcip.

Windows: Active Response Configuration

To start, you need to enable active response on Windows (disabled by default). To do that, just add the following to the agent's ossec.conf:

```
<active-response>
    <disabled>no</disabled>
</active-response>
```

After that, you need to go to the manager and specify when to run the response. Adding the following to ossec.conf will enable the responses for alerts above level 6:

```
<command>
    <name>win_nullroute</name>
    <executable>route-null.cmd</executable>
    <expect>srcip</expect>
    <timeout_allowed>yes</timeout_allowed>
</command>

<active-response>
    <command>win_nullroute</command>
    <location>local</location>
```

(continues on next page)

(continued from previous page)

```
    <level>6</level>
    <timeout>600</timeout>
</active-response>
```

With the configuration completed (and the manager restarted), you can test the active response by running the agent-control script (in this case, I am running it on agent id 185 to block ip 2.3.4.5):

```
# /var/ossec/bin/agent_control -L

OSSEC HIDS agent_control. Available active responses:

Response name: host-deny600, command: host-deny.sh
Response name: firewall-drop600, command: firewall-drop.sh
Response name: win_nullroute600, command: route-null.cmd

# /var/ossec/bin/agent_control -b 2.3.4.5 -f win_nullroute600 -u 185

OSSEC HIDS agent_control: Running active response "win_nullroute600 "n: 185
```

And looking at the agent you should see the new entry in the route table:

```
C:\>route print
..
Active Routes:
Network Destination Netmask Gateway Interface Metric
2.3.4.5 255.255.255.255 x.y.z x.y.z 1
..
```

If you run into any issues, look at the ossec.log file (on the agent) for any entry for ossec-execd. If you enabled it correctly, you will see:

```
2008/08/20 11:53:49 ossec-execd: INFO: Started (pid: 3896).
```

Understanding Active Response with FreeBSD

There are currently 3 options for firewalls in FreeBSD: IPF, IPFW, and PF. Each is configured differently on FreeBSD. OSSEC will attempt to check for IPFW and then PF, falling back to IPF if neither of these was found at the time of installation.

How does OSSEC know which firewall is being used?

The OSSEC install script will check `rc.conf` to determine which firewall is currently active. It first greps for `firewall_enable="YES"`, and enables IPFW if this is found. IPFW support is enabled by copying the `ipfw.sh` to `/var/ossec/active-response/bin/firewall-drop.sh`. The installation script will then look for ‘‘pf_enable=”YES”‘ in the rc.conf, and will enable PF instead if this is found. The script for pf is pf.sh. If neither of these is found, the default firewall-drop.sh script will be installed. This script will use attempt to use IPF to block IPs.

How do I change which script is used by an agent?

Copy the appropriate script from the OSSEC source to `/var/ossec/active-response/bin/firewall-drop.sh` on the agent.

Frequently asked questions

Agents: FAQ

Why can't agent IDs be re-used?

When looking at historical alerts you don't want to associate alerts from one system to be attributed to another, especially if the those alerts are from an unrelated and retired system.

ossec-logcollector(PID): ERROR: Unable to open file '/queue/ossec/.agent_info'

Ensure there is a <server-ip> configured in the agent's /var/ossec/etc/ossec.conf, and that the IP is correct.

The OSSEC agent is unable to resolve hostnames from /etc/hosts

By default OSSEC chroots many of its daemons to */var/ossec*. When this happens the */etc/hosts* file is unreadable. To resolve this issue, copy */etc/hosts* to */var/ossec/etc/*. A hardlink to */etc/hosts* can be used if the system is does not have a separate */var/* partition.

Using a hostname for the server does not work.

ossec-agentd chroots to */var/ossec* by default. In order to resolve hostnames, the agent will need some support. It could be as simple as copying */etc/resolv.conf* to */var/ossec/etc/resolv.conf*.

Alerts: FAQ

How do you monitor for usb storage?

The first step is to configure the agents to check a registry entry with the `reg` command:

```
<agent_config os="Windows">
  <localfile>
    <log_format>full_command</log_format>
    <command>reg QUERY HKLM\SYSTEM\CurrentControlSet\Enum\USBSTOR</command>
    <alias>usb-check</alias>
  </localfile>
</agent_config>
```

Next create a local rule for that command:

```
<rule id="140125" level="7">
  <if_sid>530</if_sid>
  <match>ossec: output: 'usb-check':</match>
  <check_diff />
  <description>New USB device connected</description>
</rule>
```

When a USB drive is inserted into a Windows machine, an alert will not be triggered. The alert will contain a diff of the registry entry before the USB device was inserted and after.

Originally from: 'http://dcid.me/2010/03/detecting-usb-storage-usage-with-ossec/'

Additional data from: 'http://blog.rootshell.be/2010/03/15/detecting-usb-storage-usage-with-ossec/'

Why do I see alerts for agent2 in an email about agent1?

When an email is being prepared alerts will be grouped together. The only real criteria for grouping alerts together is the timeframe. To prevent alerts from being grouped together you can set `maild.groupping` to 0 in `/var/ossec/etc/internal_options.conf`. If this is set, alerts will be sent out individually. By default OSSEC will only send 12 emails per hour. To increase this limit, modify or add the <email_maxperhour> setting in the `<global>` section of the `ossec.conf`. (see: email_maxperhour .)

```
<global>
  <email_maxperhour>100</email_maxperhour>
```

Alerts for different sensors are appearing in the same email, how do I stop this from happening?

Read the previous FAQ entry.

How do I ignore rule 1002?

Rule 1002 is a catch-all rule. It looks for keywords that are generally considered "bad." It also means there is not currently a rule that deals with the log message. It is configured to always send an email when it's triggered, and many users have found it annoying. The best thing to do when you encounter something that triggers rule 1002 is write a rule. Contributing the logs and/or rules back to the project is also encouraged. Unless the application creating the log is an internal application, someone else may find the rule useful.

I set the <email_alert_level> to 10, why do I keep seeing rules with lower levels?

Some rules have an option set to force OSSEC into sending an alert email. This option is `<options>alert_by_email</options>`. One of these rules is 1002. To ignore these rules you will have to create a rule to specifically ignore it, or overwrite the rule without the `alert_by_email` option.

Why are all of my Windows alerts showing up as rule 1002?

This is a known issue when using an older version on the OSSEC manager and newer versions on the agents. The manager should never be an older version than the agents. Using the same version is ideal, but when that is not possible, the manager should be the newest version.

I keep getting log messages that start with `--MARK`, what do I do?

Example:

```
OSSEC HIDS Notification.
2014 Sep 21 08:36:11

Received From: (services01.qrios.com) 10.10.0.01->ossec-keepalive
Rule: 1002 fired (level 2) -> "Unknown problem somewhere in the system."
Portion of the log(s):

--MARK--: cB82L!#'zr%lQfGUE))-7Q#Tj4fp+KG=@Wbq^wXiN)7L;ha!JB0kA_mJT5g-j[v_
↪R@TAbk-,/fHEnHjerroRgAIh?OLWJ6LIL/q[Wg-cl#H#n/&(3NDr$@#v8r*l;!b*qru
↪$@YxEE4Zak=(wEqN0JuDlLo!*HtlXKF(3.kQ&wj&+aklF%YNsA&*#Mef)xq'qd@)P+Dz0[jP]70
↪%Q6vqbfbv?fA)D?#bc?_R+6y.i+MdXUzhucx9V#MV($-3uk4!ja!MocJx;D%P=We0Y^DoV&
↪r+fha$rmRA1v$^y4U4cD1'H5T?OF1R3(Hq'H.YcO'3soM/(P2_A@w7K^6G2C=z2#.
↪W2[24=VBXrvVe!5;eKotCM[8W!hE_CB;/!Vk1k'sCov_H[u'(=no*VEH$
↪'@1vVU7zj*I7s0RHD)w@ipX?&@y)X50Q'w#OyN$+$pW?xW_0JYFRwK/g3wIuKc!D#Q*eMg'79/
↪oaURi.j].))JIQ6&!k(O]ZN#lHATidRwRTvhQFQ]6DFiBT;fltbg(OALDi/LlPqkcL5bypK?
↪axVByOGJp+.P(@[p)'j
```

This is a keep alive message, sent from an OSSEC agent to the manager. This prevents the manager from marking the agent as disconnected. These log messages should be filtered out, but sometimes one slips through (this one has the string `erroR` which may be the cause). These should be safe to ignore.

Installation: FAQ

Chown errors during installation on AIX:

AIX does not accept `/bin/false` as a valid shell by default. It must be added to the SHELLS line in `/etc/security/login.cfg` before running `install.sh`.

The Windows GUI is asking me for a key, where do I get it?

The Windows version of OSSEC is agent only, it cannot work without a server. The key can be obtained from the server using *manage_agents*.

Miscellaneous: FAQ

What are the github issues intended to be used for?

Due to the technical nature of OSSEC, it can be difficult, to troubleshoot an issue. Because of the time and effort involved, the mailing list is the preferred method of seeking help. Github issues are great for tracking discovered bugs, but are more difficult for communicating when troubleshooting.

OSSEC: FAQ

Can an OSSEC manager have more than 256 agents?

By default OSSEC limits the number of agents to 256 per manager. This limitation is set in the code, but can be modified at compile time. Depending on the event load, a manager running on modern hardware can handle many more agents. Some users have more than 1000 agents on a single manager. To change the maximum number of agents, cd into the src directory and run the following command:

```
make setmaxagents
```

You should be prompted for the number of agents to allow.

One issue you may face after changing this setting is the number of files allowed to be open for a single user. The users `ossec` and `ossecr` both open at least 1 file (syscheck database and rids file) per agent. Raising this limit is operating system specific.

Some Linux distributions support a `/etc/security/limits.conf`. Set the limits to be at least a few files above what the max agents is set to.

```
ossec              soft    nofile           2048
ossec              hard    nofile           2048
ossecr             soft    nofile           2048
ossecr             hard    nofile           2048
```

Where are OSSEC's logs stored?

On OSSEC server and local installs there are several classes of OSSEC logs. There are the logs created by the OSSEC daemons, the log messages from the agents, and the alerts. Agent installs do not have logs from other agents or alerts, but do have logs created by the OSSEC processes.

All logs are stored in subdirectories of `/var/ossec/logs`. OSSEC's log messages are stored in `/var/ossec/logs/ossec.log`.

Log messages from the agents are not stored by default. After analysis they are deleted unless the <logall> option is included in the manager's ossec.conf. If set all log messages sent to the manager are stored in `/var/ossec/logs/archives/archives.log` and rotated daily.

Alerts are stored in `/var/ossec/logs/alerts/alerts.log`, and rotated daily.

Where can I view the logs sent to an OSSEC manager (or on a local install)?

OSSEC does not store the logs sent to it by default. If a log does not trigger an alert it is discarded, and logs that do trigger alerts are stored with the alerts in `/var/ossec/logs/alerts`.

The `<log-all>` option can be added to the `<global>` section (see: *ossec.conf: Global options*) of the manager's ossec.conf. The manager's OSSEC processes should be restarted. The raw logs will then be saved to files, organized by date, in `/var/ossec/logs/archives`.

The headers attached to these log messages are in the format of `"YYYY Month dd HH:MM:ss agent_name->/path/to/log/file "`.

```
2011 Aug 04 00:00:01 server->/var/log/local7 Aug  4 00:00:26 server␣
↪named[29909]: client 192.168.1.7#39323: query: fake.example.net IN AAAA +
```

Can OSSEC's logs be saved to a different directory?

As a protection mechanism, OSSEC chroots most of its processes to the install directory (typically `/var/ossec`). Due to this chroot, logs must be saved to a location under `/var/ossec`. OSSEC does rotate its logs, but will not be able to move them from `/var/ossec`.

Be sure to allocate enough space to `/var/ossec`.

I'm getting an error when starting OSSEC: "OSSEC analysisd: Testing rules failed. Configuration error. Exiting." Why?

There was a small bug in the ossec-control script that was not caught in time for 2.6. The error comes from the script trying to run ossec-logtest from the wrong directory. The solution is to change the line where ossec-logtest is running to look like this:

```
echo | ${DIR}/bin/ossec-logtest > /dev/null 2>&1;
```

The rules aren't on my agents, they're only on the server!

That's not a question. Also, that's the way it is. Only the server has the rules. Agents do not get a copy of the rules.

Do the rules get pushed to the agents automatically?

The rules only exist on the manager. All analysis is done on the manager. Agents do not send alerts to the manager, they only send the raw logs.

How can I get ossec.log to rotate daily?

Currently OSSEC does not rotate the `ossec.log`, use logrotate.d or newsyslog to rotate it for now.

OSSEC-WUI: FAQ

Why does the OSSE-WUI appear to be dead?

Because it is. No one has worked on it for quite a while. There may be some ongoing work with it, but as of this writing it is considered a dead project.

Why does the src ip field contain strange information instead of an IP?

Users who have installed OSSEC-WUI 0.3, have not applied the necessary patches, and are using OSSEC 2.6 or later may see alerts like the following:

```
2013 Feb 02 10:48:42 Rule Id: 2901 level: 3
Location: ubuntu->/var/log/dpkg.log
Src IP: 02 10:48:41 install libapr1 <none> 1.4.6-1
New dpkg (Debian Package) requested to install.
** Alert 1359830922.3553: - syslog,dpkg,
2013 Feb 02 10:48:42 ubuntu->/var/log/dpkg.log
Rule: 2901 (level 3) -> 'New dpkg (Debian Package) requested to install.'
2013-02-02 10:48:41 install libaprutil1 <none> 1.3.12+dfsg-3
```

The alert format changed in 2.6, and since OSSEC-WUI is essentially abandonware it was not updated to handle the changes. A number of users have provided patches to correct the issues, and the OSSEC team is planning on releasing an updated WUI containing these patches. You can find a patched version of the OSSEC-WUI at a *github repository <https://github.com/ossec/ossec-wui>*_.

Syscheck: FAQ

How to force an immediate syscheck scan?

Run agent control tool to perform a integrity checking immediately (option -a to run on all the agents and -u to specify an agent id)

```
# /var/ossec/bin/agent_control -r -a
# /var/ossec/bin/agent_control -r -u <agent_id>
```

For more information see the *agent_control* documentation.

How to tell syscheck not to scan the system when OSSEC starts?

Set the option <scan_on_start> to "no" on ossec.conf

How to ignore a file that changes too often?

Set the file/directory name in the <ignore> option or create a simple local rule.

The following one will ignore files /etc/a and /etc/b and the directory /etc/dir for agents mswin1 and ubuntu-dns:

```
<rule id="100345" level="0" >
    <if_group>syscheck</if_group>
    <description>Changes ignored.</description>
    <match>/etc/a|/etc/b|/etc/dir</match>
    <hostname>mswin1|ubuntu-dns</hostname>
</rule>
```

Why does OSSEC still scan a file even though it's been ignored?

No idea. So if there are some directories you do not want scanned at all, make sure they are not included in a `<directories>` configuration.

How to know when the syscheck scan ran?

Use the agent_control tool on the manager, to see this information.

More information see the *agent_control* documentation.

How to get detailed reporting on the changes?

Use the syscheck_control tool on the manager or the web ui for that.

More information see the *syscheck_control* documentation.

Syscheck not sending any file data to the server?

With ossec 1.3 and Fedora you may run into this problem:

You have named files you'd like ossec to monitor so you add:

```
<ossec_config>
    <syscheck>
        <directories check_all="yes">/var/named</directories>
```

to ossec.conf on the client. Fedora – at least as of version 7 – runs named in a chroot jail under /var/named/chroot. However, part of that chroot jail includes /var/named/chroot/proc. The contents of that directory are purely ephemeral; there is no value to checking their integrity. And, at least in ossec 1.3, your syscheck may stall trying to read those files.

The symptom is a syscheck database on the server that never grows beyond a file or two per restart of the client. The log monitoring continues to work, so you know it's not a communication issue, and you will often see a slight increase in syscheck database file size after the client has restarted (in one case about 20 minutes after). But the database will never be completely built; there will only be a couple files listed in database.

The solution is to add an ignore clause to ossec.conf on the client:

```
<ossec_config>
    <syscheck>
        <ignore>/var/named/chroot/proc</ignore>
```

Why aren't new files creating an alert?

By default OSSEC does not alert on new files. To enable this functionality, <alert_new_files> must be set to yes inside the <syscheck> section of the manager's ossec.conf. Also, the rule to alert on new files (rule 554) is set to level 0 by default. The alert level will need to be raised in order to see the alert. Alerting on new files does not work in realtime, a full scan will be necessary to detect them.

Add the following to local_rules.xml:

```
<rule id="554" level="10" overwrite="yes">
  <category>ossec</category>
  <decoded_as>syscheck_new_entry</decoded_as>
  <description>File added to the system.</description>
  <group>syscheck,</group>
</rule>
```

The `<alert_new_files>` entry should look something like this:

```
<syscheck>
  <frequency>7200</frequency>
  <alert_new_files>yes</alert_new_files>
  <directories check_all="yes">/etc,/bin,/sbin</directories>
</syscheck>
```

Can OSSEC include information on who changed a file in the alert?

In short, no. OSSEC does not track this information. You could use your OS's auditing facilities to track this information, and create a rule to alert when an appropriate log is created.

How do I stop syscheck alerts during system updates?

There is no easy way to do this, but there are work-arounds. Stop the OSSEC processes on the manager, and run `/var/ossec/bin/syscheck_control -u AGENT_ID`. This will clear the syscheck database for the agent, and the next time syscheck runs it will create a new baseline. Next, start the OSSEC processes on the manager. Once the system update is complete, run a syscheck scan on that agent. The database will be populated with new values, and should not trigger "file modified" alarms.

When the unexpected happens: FAQ

How do I troubleshoot ossec?

If you are having problems with ossec, the first thing to do is to look at your logs.

- Unix/Linux: The logs will be at /var/ossec/logs/ossec.log
- Windows: The logs are at C:Program Filesossec-agentossec.log.

If by looking at them, you can't find out the error, we suggest you to send an e-mail to one of our mailing lists with the following information:

- OSSEC version number.

  Run the following to get the version installation.

  ```
  # /var/ossec/bin/ossec-analysisd -V
  ```

- Content of /etc/ossec-init.conf
- Content of /var/ossec/etc/ossec.conf or (or C:Program Filesossec-agentossec.log if Windows)
- Content of /var/ossec/logs/ossec.log
- Operating system name/version (uname -a if Unix)
- Any other relevant information.

How to debug ossec?

Warning: Only read this section if you tried to troubleshoot ossec already, but didn't have lucky solving your problem. Most of the users will never need to enable debugging, since it can significantly hurt performance.

Debug Logging

You can also enable debugging mode on ossec to extract more data about what is going on. To do so, you will need to modify the file /var/ossec/etc/internal_options.conf (or C:\Program Files\ossec-agent\internal_options.conf on Windows) and change the debug level from the default "0" to "1" or "2".

For example, if you wish to debug your windows agent, just change the option windows.debug from 0 to 2. Bellow is the list of all the debug options:

```
# Debug options.
# Debug 0 -> no debug
# Debug 1 -> first level of debug
# Debug 2 -> full debugging

# Windows debug (used by the windows agent)
windows.debug=0

# Syscheck (local, server and unix agent)
syscheck.debug=0

# Remoted (server debug)
remoted.debug=0

# Analysisd (server or local)
```

(continues on next page)

(continued from previous page)

```
analysisd.debug=0

# Log collector (server, local or unix agent)
logcollector.debug=0

# Unix agentd
agent.debug=0
```

If this is on an OSSEC server you can enable debug by running:

```
# /var/ossec/bin/ossec-control enable debug
```

Enable debug mode and restart the OSSEC processes to view more verbose logs.

Getting more log data

If you are up to editing the source and recompiling, you can use the verbose() function to add entries to the log. This has been helpful on at least one occasion to help pinpoint where a problem was occurring. Something along these lines should work (at least in 1.3):

```
verbose("MyName: inside the_file.c the_function() %s ..", the_string);
```

- If you tag all your extra logs with something, MyName, in this example, they stand out better.
- If you need to get information from several source files, including the file name the_file.c, in this example is helpful.
- You will almost surely want information from more than one function, including the name, the_function() will show which function sent the log.
- Finally, you can include a variable string with the printf format specifier %s in the log entry and the_string is the name of the string variable to send to the log.

With some calls to verbose, recompile and replace the stock binary with your edited one. Restart ossec and tail the log.

The communication between my agent and the server is not working. What to do?

There are multiple reasons for it to happen. First, you should look at your agent and server logs to see what they say. If you don't know where they are, go to our Troubleshooting page for more information.

In addition to that, follow the step by step at the end, if you need to add/re-add the authentication keys.

There is a firewall between the agent and the server.

If you have the following message on the agent log:

```
2007/04/19 12:42:54 ossec-agentd(4101): Waiting for server reply (not␣
↪started).
2007/04/19 12:43:10 ossec-agentd(4101): Waiting for server reply (not␣
↪started).
2007/04/19 12:43:41 ossec-agentd(4101): Waiting for server reply (not␣
↪started).
2007/04/19 12:44:27 ossec-agentd(4101): Waiting for server reply (not␣
↪started).
```

And nothing on the server log, you probably have a firewall between the two devices. Make sure to open port 1514 UDP between them (keeping state –the agent connects to the server and expects a reply back).

Note: The way the agent/server communication works is that the agent starts a connection to the server using any random high port. So, the only port that OSSEC opens is in the server side (port 1514 UDP). It works similar to DNS, where the DNS client connects to UDP port 53 and expects a reply back.

Wrong authentication keys configured (you imported a key from a different agent).

If that's the case, you would be getting logs similar to the above on the agent and the following on the server (see also Errors:1403):

```
2007/05/23 09:27:35 ossec-remoted(1403): Incorrectly formated message from
↪'xxx.xxx.xxx.xxx'.
2007/05/23 09:27:35 ossec-remoted(1403): Incorrectly formated message from
↪'xxx.xxx.xxx.xxx'.''
```

The IP address you configured the agent is different from what the server is seeing.

Same as above (see also see Errors:1403).

Step by Step – adding the authentication keys

For most of the errors (except the firewall issue), removing and re-adding the authentication keys fix the problem. Do the following if you are having issues:

1. 'Stop the server and the agent.'
   - Make sure they are really stopped (ps on Unix or sc query ossecsvc on Windows)
2. Run the manage-agents tool on the server and remove the agent.
3. Still on the server, add the agent using manage-agents. Make sure the IP is correct.
4. Start the server.
5. Run manage-agents on the agent and import the newly generated key.
6. Start the agent.

If after that, it still doesn't work, contact our mailing list for help.

What does "1403 - Incorrectly formated message" means?

It means that the server (or agent) wasn't able to decrypt the message from the other side of the connection. See *The communication between my agent and the server is not working. What to do?*

The main reasons for this to happen are:

- Wrong authentication keys configured (you imported a key from a different agent).
- The IP address you configured the agent is different from what the server is seeing.

How to fix it:

- Check if you imported the right authentication keys into the agent.
- Check if the IP address is correctly.
- You can also try to remove the agent (using manage_agents), add it back again and re-import the keys into the agent. Make sure to restart the server (first) and then the agent after that.

What does "1210 - Queue not accessible?" mean?

Check queue/ossec/queue

If you have logs similar to the following in `/var/ossec/queue/ossec/queue`:

```
2008/04/29 15:40:39 ossec-syscheckd(1210): ERROR: Queue '/var/ossec/queue/ossec/queue
↪' not accessible: 'Connection refused'.
2008/04/29 15:40:39 ossec-rootcheck(1210): ERROR: Queue '/var/ossec/queue/ossec/queue
↪' not accessible: 'Connection refused'.
2008/04/29 15:40:45 ossec-logcollector(1210): ERROR: Queue '/var/ossec/queue/ossec/
↪queue' not accessible: 'Connection refused'.
2008/04/29 15:40:45 ossec-logcollector(1211): ERROR: Unable to access queue: '/var/
↪ossec/queue/ossec/queue'. Giving up..
2008/04/29 15:41:00 ossec-syscheckd(1210): ERROR: Queue '/var/ossec/queue/ossec/queue
↪' not accessible: 'Connection refused'.
2008/04/29 15:41:00 ossec-rootcheck(1211): ERROR: Unable to access queue: '/var/ossec/
↪queue/ossec/queue'. Giving up..
```

It means that *ossec-analysisd* is not running for some reason.

The main reasons for this to happen are:

- *ossec-analysisd* didn't start properly. Look at the logs for any error from it.
- *ossec-analysisd* didn't start at all. There is a bug in the init scripts that during system reboot, it may not start if the PID is already in use (we are working to fix it).
- *ossec-analysisd* cannot access `/queue/fts/fts-queue`. Look for the error message `ossec-analysisd(1103): ERROR: Unable to open file '/queue/fts/fts-queue'`. This can be fixed by ensuring that the ossec user owns `/var/ossec/queue/fts/fts-queue`.

How to fix it:

Stop OSSEC and start it back again:

```
# /var/ossec/bin/ossec-control stop
(you can also check at /var/ossec/var/run that there is not PID file in there)
# /var/ossec/bin/ossec-control start
```

If there is any configuration error, fix it.

Check queue/alerts/ar

If you have logs similar to the following in `/var/ossec/queue/alerts/ar`:

```
2009/02/17 12:03:04 ossec-analysisd(1210): ERROR: Queue '/queue/alerts/ar' not␣
↪accessible: 'Connection refused'.
2009/02/17 12:03:04 ossec-analysisd(1301): ERROR: Unable to connect to active␣
↪response queue.
```

It means that there is nothing listening on the other end of the socket the *ossec-analysisd* deamon would want to write to. This can happen in an ossec server installation. The deamon that should be listening on this socket is *ossec-remoted*.

How to fix it:

Add an OSSEC client (agent) with the *manage_agents* utility on both agent and server. Then restart OSSEC. *ossec-remoted* should now be listening on the socket.

Remote commands are not accepted from the manager. Ignoring it on the agent.conf

This error message is caused by `command` or `full_command` log types in the agent.conf. Originally OSSEC supported running commands from the agent.conf by default. This was later changed as a security precaution due to the commands being run as root. When a command is encountered on an agent in the agent.conf this error will be produced and the agent may not fully start. This error may also accompany the above error message:

```
ERROR: Configuration error at '/var/ossec-agent/etc/shared/agent.conf'. Exiting.
```

Errors when dealing with multiple agents

When you have hundreds (or even thousands) of agents, OSSEC may not work properly by default. There are a few changes that you will need to do:

Increase maximum number of allowed agents

To increase the number of agents, before you install (or update OSSEC), just do:

```
#cd src; make setmaxagents (it will ask how many do you want.. )

Specify maximum number of agents: 2048 (to increase to 2048)
Maximum number of agents set to 20.

#cd ..; ./install.sh
```

Increase your system's limits

Most systems have limits regarding the maximum number of files you can have. A few commands you should try are (to increase to 2048):

```
# ulimit -n 2048
# sysctl -w kern.maxfiles=2048
```

Fixing Duplicate Errors

Ossec agents and server keep a counter of each message sent and received in files in …/ossec/queue/rids. This is a technique to prevent replay attacks. If the counters between agent and server don't match you'll see errors like this in the agents ossec.log file:

```
2007/10/24 11:19:21 ossec-agentd: Duplicate error:  global: 12, local: 3456, saved␣
↪global: 78, saved local: 91011
2007/10/24 11:19:21 ossec-agentd(<pid>): Duplicated counter for '<host name>'.
2007/10/24 11:19:21 ossec-agentd(<pid>): Problem receiving message from www.xxx.yyy.
↪zzz.
```

This normally happens when you restore the ossec files from a backup or you reinstall server or agents without performing an upgrade, this can also be caused by duplicate agent ID's. The fix for this problem is:

1. On every agent:

1. stop ossec
2. go to: …/ossec/queue/rids (or ossec-agent/rids on Windows) and remove every file in there.

2. Go to the server:

1. Stop ossec

2. Remove the rids file with the same name as the agent id that is reporting errors.
3. Restart the server
4. Restart the agents.

To avoid this problem from ever happening again, make sure to:

- Always use the update option (when updating). Do not remove and reinstall the ossec server, unless you plan to do the same for all agents.
- Do not re-use the same agent key between multiple agents or the same agent key after you remove/re-install an agent. If you use the "update" options everything should just work.

Agent won't connect to the manager or the agent always shows never connected

The following log messages may appear in the `ossec.log` file on an agent when it is having issues connecting to a manager:

```
2011/11/13 18:05:13 ossec-agent: WARN: Process locked. Waiting for permission...
2011/11/13 18:05:24 ossec-agent(4101): WARN: Waiting for server reply (not started).␣
↪Tried: '10.10.134.241'.
2011/11/13 18:05:26 ossec-agent: INFO: Trying to connect to server (10.10.134.
↪241:1514).
2011/11/13 18:05:26 ossec-agent: INFO: Using IPv4 for: 10.10.134.241 .
2011/11/13 18:05:47 ossec-agent(4101): WARN: Waiting for server reply (not started).␣
↪Tried: '10.10.134.241'.
```

If the agent's packets are making it to the manager, the manager will also include error messages in its `ossec.log` related to that agent. Some possible issues:

- The agent may not be using the correct IP address. Some systems with multiple IP addresses may not choose the correct one to communicate with the OSSEC manager. Using `any` or a CIDR address (192.168.1.0/24) for the agent may be one solution, and adjusting the system's route settings is another.
- Every agent must be using a unique key. If 2 agents look like they're coming from the same IP (possibly from a NAT gateway), then `any` or the CIDR address should be used to identify them on the manager.
- There may be a firewall blocking the OSSEC traffic, udp 1514 should be allowed to and from the manager.
- UAC may be blocking the OSSEC service from communicating with the manager on Windows 7.

I am seeing high CPU utilization on a Windows agent

Some OSSEC HIDS users who have deployed the Windows agent have experienced situations where the windows OSSEC agent causes high CPU utilization. In some cases, this may be due to syscheck having to do integrity checking on a large number of files and the frequency with which this is done. The high CPU utilization could also take place when the OSSEC agent has to analyze Windows Event logs with very large numbers of generated events.

A clue to what may be happening are alerts like these:

```
OSSEC HIDS Notification.
2006 Oct 24 03:18:07

Received From: (ACME-5) 10.23.54.40->WinEvtLog
Rule: 11 fired (level 8) -> "Excessive number of events (above normal)."
Portion of the log(s):

```

(continues on next page)

(continued from previous page)

```
The average number of logs between 3:00 and 4:00 is 268689. We reached 270690.



--END OF NOTIFICATION
```

The above alert indicates the condition where a large number of events are being generated in the Windows event logs. In Windows, setting the Windows audit policy to Audit Object Access or Audit Process Tracking can cause the generation of many event log entries. This gives the OSSEC agent much more work to do in log analysis, and thus causes the consumption of much more CPU cycles. To reduce the CPU utilization in this case, the solution is to disable auditing of object access and/or process tracking. Typically, these audit settings aren't required except for debugging purposes, or situations in which you absolutely have to track everything.

My /etc/hosts.deny file is blank after install 2.8.1!

There was a bug introduced to the host-deny.sh script that would empty the file. It has been fixed for 2.9. Some variable declarations in the script have a space between the variable name, the =, and the value. Removing these spaces allows the script to work as planned. If you are using a system that is still using tcpwrappers, either use the current host-deny.sh, or remove the spaces from the script before installation.

5.1.2 Development

Build, compile, and not much more

install.sh

Makefile

OSSEC is using (starting with 2.9) a single Makefile to build the binaries for each installation type. The `Makefile` uses features of GNU make, and it is a requirement of the build process. The make system attempts to perform all necessary tasks in a single run. The type of installation must be specified when executed because there is no state stored between executions.

Settings

All changes to the makefile that take external input should be reported via the settings build step. This allows troubleshooting and review of the environment, and the hope is that some new features will become discoverable for other developers.

```
% make DATABASE=mysql TARGET=server USE_ZEROMQ=1 USE_GEOIP=1 settings

General settings:
    TARGET:          server
    V:
    DEBUG:
    DEBUGAD
    PREFIX:          /var/ossec
    MAXAGENTS:       2048
    DATABASE:        mysql
User settings:
```

(continues on next page)

(continued from previous page)

```
    OSSEC_GROUP:     ossec
    OSSEC_USER:      ossec
    OSSEC_USER_MAIL: ossecm
    OSSEC_USER_REM:  ossecr
Lua settings:
    LUA_PLAT:       macosx
USE settings:
    USE_ZEROMQ:     1
    USE_GEOIP:      1
    USE_PRELUDE:    0
Mysql settings:
    includes:       -I/usr/include/mysql/
    libs:           -L/usr/lib/mysql -lmysqlclient
Pgsql settings:
    includes:
    libs:
Defines:
    -DMAX_AGENTS=2048 -DOSSECHIDS -DDarwin -DHIGHFIRST -DZEROMQ_OUTPUT  -DGEOIP -
↪DUMYSQL
Compiler:
    CFLAGS          -DMAX_AGENTS=2048 -DOSSECHIDS -DDarwin -DHIGHFIRST -DZEROMQ_
↪OUTPUT  -DGEOIP -DUMYSQL  -Wall -Wextra -O2 -I./ -I./headers/
    LDFLAGS         -lzmq -lczmq -lGeoIP -L/usr/lib/mysql -lmysqlclient
    CC              cc
    MAKE            /Applications/Xcode.app/Contents/Developer/usr/bin/make
```

Options / Variables

TARGET

Defines the type of installation to build.

Allowed: server/agent/hybrid/local

V

`V` can toggle verbose building, and will instruct `make` to display the full output without color.

Applies to Target: all

Default: 0

Allowed: 0/1

PREFIX

`PREFIX` is the absolute path OSSEC will be installed to.

> **Warning:** Please note that paths with SPACES and tabs in them are not supported and may cause compilation or runtime issues.

Applies to Target: all

Default: /var/ossec

Allowed: All valid paths

ZLIB_SYSTEM

New in version 3.0: `ZLIB_SYSTEM` can enable OSSEC to use a pre-installed zlib instead of the bundled version.

Applies to Target: all

Default: no

Allowed: yes/no

`LUA_ENABLE`

New in version 3.0: `LUA_ENABLE` can enable or disable the bundled lua support.

Applies to Target: all

Default: yes

Allowed: yes/no

`MAXAGENTS`

OSSEC is compiled with a maximum number of agents on the server/hybrid TARGETS. `MAXAGENTS` allows users to select values expected for their environment.

Applies to Target: server/hybrid

Default: 2048

Allowed: [2 - 65000]

`DEBUG`

`DEBUG` enables debugging symbols in all compiled programs.

Applies to Target: all

Default: 0

Allowed: 0/1

`DEBUGAD`

`DEBUGAD` enables extra debugging logging in ossec-analysisd.

Applies to Target: server/hybrid

Default: 0

Allowed: 0/1

`OSSEC_USER`

Default: ossec

`OSSEC_GROUP`

Default: ossec

`OSSEC_USER_MAIL`

Default: ossecm

`OSSEC_USER_REM`

Default: ossecr

`LUA_PLAT`

`LUA_PLAT` determines the platform that LUA will be compiled for.

This is autogenerated for each install.

Applies to Target: all

`USE_GEOIP`

`USE_GEOIP` enables support for MAX Mind GeoIP looks on output.

Applies to Target: server/hybrid/local

Default: 0

Allowed: 0/1

`USE_PRELUDE`
`USE_PRELUDE` enables support for Prelude-IDS alert output.

Applies to Target: server/hybrid/local

Default: 0

Allowed: 0/1

`USE_ZEROMQ`
`USE_ZEROMQ` enables support for zeromq output.

Applies to Target: server/hybrid/local

Default: 0

Allowed: 0/1

`USE_SQLITE`
`USE_SQLITE` enables support for sqlite3 databases.

Spplies to Target: server/hybrid/local

Default: 0

Allowed: 0/1

`DATABASE`
The `DATABASE` variable selects the database enviromnet to enable.

Applies to Target: server/hybrid/local

Allowed: mysql/pgsql

test-rules:

OSSEC includes the facilities to test rules in bulk. These checks should ensure there are no regressions after changes have been made. `ossec-logtest -U` is used to test the outcome of rules. If `ossec-logtest` exits with any code but 0 it is considered a failure.

Requirements:

This feature currently requires python, a copy of the OSSEC source code, and the tools required to build OSSEC.

Note: This requirement may change in the future. There is interest in utilizing the embedded lua.

How to run:

Running the current tests is as simple as running `make test-rules` in the `src` directory.

```
[ddpbsd@ossec-build:ossec-hids/src] $ more /tmp/ttt
( cd ../contrib/ossec-testing && sudo python runtests.py)
- [ File = ./tests/named.ini ] ---------
.....
- [ File = ./tests/sshd.ini ] ---------
...........
- [ File = ./tests/apparmor.ini ] ---------
.....
- [ File = ./tests/cimserver.ini ] ---------
..
- [ File = ./tests/cisco_ios.ini ] ---------
.....
- [ File = ./tests/firewalld.ini ] ---------
..
- [ File = ./tests/netscreen.ini ] ---------
...
- [ File = ./tests/ossec.ini ] ---------
....
- [ File = ./tests/pam.ini ] ---------
.....
- [ File = ./tests/rsh.ini ] ---------
..
- [ File = ./tests/samba.ini ] ---------
....
- [ File = ./tests/su.ini ] ---------
.....
- [ File = ./tests/sudo.ini ] ---------
..
- [ File = ./tests/syslog.ini ] ---------
..
- [ File = ./tests/systemd.ini ] ---------
.
- [ File = ./tests/unbound.ini ] ---------
```

File format:

The rule checks are configured in `ini` files. These files are in the `contrib/ossec-testing/tests` directory. Each file contains the configuration to check the rules for one program, for example `named.ini` contains checks for the named rules.

Example:

```
[su: failed ]
log 1 pass = Apr 27 15:22:23 niban su[2921936]: failed: ttyq4 changing from ldap to␣
↪root
rule = 5302
alert = 9
decoder = su

[su: bad pass]
log 1 pass = Apr 27 15:22:23 niban su[234]: BAD SU ger to fwmaster on /dev/ttyp0
rule = 5301
alert = 5
decoder = su

[su: pam - auth fail]
```

(continues on next page)

(continued from previous page)

```
log 1 fail = Apr 27 15:22:23 niban su(pam_unix)[23164]: authentication failure;␣
↪logname= uid=1342 euid=0 tty= ruser=dcid rhost=  user=osaudit
log 2 fail = Apr 27 15:22:23 niban su(pam_unix)[2298]: authentication failure;␣
↪logname= uid=1342 euid=0 tty= ruser=dcid rhost=  user=root
rule = 5503
alert = 5
decoder = su

[su: work]
log 1 pass = Apr 22 17:51:51 enigma su: dcid to root on /dev/ttyp1
rule = 5303
alert = 3
decoder = su
```

Each entry has a number of configuration elements:

- `[su:  - bad pass]` - This is a heading, it usually contains the rule description.
- `log 1 pass =` - This is the first log, and the `ossec-logtest` should pass. Noting failures is also possible by using `fail` instead of `pass`.
- `Apr 27 15:22:23 niban su[234]:  BAD SU ger to fwmaster on /dev/ttyp0` - This is the log message to be checked.
- `rule = 5301` - This is the expected rule, if the log message triggers a different rule the test will return a failure.
- `alert = 5` - This is the expected rule level. A failure to match this level will result in a failure.
- `decoder = su` - This is the expected decoder. A failure to match this decoder will result in a failure.

The above entry can be verified by running the log message through ossec-logtest:

```
 [ddpbsd@ossec-build:/var/ossec] $ sudo /var/ossec/bin/ossec-logtest
 2014/10/31 10:37:38 ossec-testrule: INFO: Reading local decoder file.
 2014/10/31 10:37:38 ossec-testrule: INFO: Started (pid: 25545).
 ossec-testrule: Type one log per line.

 Apr 27 15:22:23 niban su[234]: BAD SU ger to fwmaster on /dev/ttyp0


**Phase 1: Completed pre-decoding.
        full event: 'Apr 27 15:22:23 niban su[234]: BAD SU ger to fwmaster on /dev/
↪ttyp0'
        hostname: 'niban'
        program_name: 'su'
        log: 'BAD SU ger to fwmaster on /dev/ttyp0'

 **Phase 2: Completed decoding.
        decoder: 'su'
        srcuser: 'ger'
        dstuser: 'fwmaster'

 **Phase 3: Completed filtering (rules).
        Rule id: '5301'
        Level: '5'
        Description: 'User missed the password to change UID (user id).'
 **Alert to be generated.
```

The rule triggered is indeed 5301, it is a level 5 alert, and the decoder was su.

Here is an example of a planned failure:

```
[su: pam - auth fail]
log 1 fail = Apr 27 15:22:23 niban su(pam_unix)[23164]: authentication failure;␣
↪logname= uid=1342 euid=0 tty= ruser=dcid rhost=  user=osaudit
rule = 5503
alert = 5
decoder = su
```

Running the above log message through ossec-logtest:

```
Apr 27 15:22:23 niban su(pam_unix)[23164]: authentication failure; logname= uid=1342␣
↪euid=0 tty= ruser=dcid rhost=  user=osaudit

**Phase 1: Completed pre-decoding.
       full event: 'Apr 27 15:22:23 niban su(pam_unix)[23164]: authentication failure;
↪ logname= uid=1342 euid=0 tty= ruser=dcid rhost=  user=osaudit'
       hostname: 'niban'
       program_name: 'su(pam_unix)'
       log: 'authentication failure; logname= uid=1342 euid=0 tty= ruser=dcid rhost= ␣
↪user=osaudit'

**Phase 2: Completed decoding.
       decoder: 'pam'
       dstuser: 'dcid'

**Phase 3: Completed filtering (rules).
       Rule id: '5503'
       Level: '5'
       Description: 'User login failed.'
**Alert to be generated.
```

The rule and level are correct, but the decoder is not. This triggers the expected failure.

An unexpected failure will produce output like the following:

```
( cd ../contrib/ossec-testing && sudo python runtests.py)
- [ File = ./tests/named.ini ] ---------

------------------------------------------------------------
Failed: Exit code = 0
        Alert     = 0
        Rule      = 12108
        Decoder   = named
        Section   = Query cache denied
        line name = log 1 fail

2014/10/31 10:49:47 ossec-testrule: INFO: Reading local decoder file.
2014/10/31 10:49:47 ossec-testrule: INFO: Started (pid: 16533).
ossec-testrule: Type one log per line.



**Phase 1: Completed pre-decoding.
       full event: 'Aug 29 15:33:13 ns3 named[464]: client 217.148.39.3#1036: query␣
↪(cache) denied'
       hostname: 'ns3'
       program_name: 'named'
```

(continues on next page)

(continued from previous page)

```
       log: 'client 217.148.39.3#1036: query (cache) denied'

**Phase 2: Completed decoding.
       decoder: 'named'
       srcip: '217.148.39.3'

**Phase 3: Completed filtering (rules).
       Rule id: '12108'
       Level: '0'
       Description: 'Query cache denied (probably config error).'
       Info - Link: 'http://www.reedmedia.net/misc/dns/errors.html'
0

....
- [ File = ./tests/sshd.ini ] ---------
...........
```

The named rule failed because it was expecting a failure in decoding, but did not trigger one.

oRFC:

oRFC: 1 The Collective Code Construction Contract (C4)

The Collective Code Construction Contract (C4) is an evolution of the github.com Fork + Pull Model, aimed at providing an optimal collaboration model for free software projects. This is revision 1 of the C4 specification.

Name	C4.1
Editor	Jeremy Rossi <jeremy at jeremyrossi dot com>
State	Accepted
Origin	http://rfc.zeromq.org/spec:22

Language

The key words “MUST”, “MUST NOT”, “REQUIRED”, “SHALL”, “SHALL NOT”, “SHOULD”, “SHOULD NOT”, “RECOMMENDED”, “MAY”, and “OPTIONAL” in this document are to be interpreted as described in RFC 2119[#f1]_.

Goals

C4 is meant to provide a reusable optimal collaboration model for open source software projects. It has these specific goals:

- To maximize the scale of the community around a project, by reducing the friction for new Contributors and creating a scaled participation model with strong positive feedbacks;
- To relieve dependencies on key individuals by separating different skill sets so that there is a larger pool of competence in any required domain;
- To allow the project to develop faster and more accurately, by increasing the diversity of the decision making process;

- To support the natural life cycle of project versions from experimental through to stable, by allowing safe experimentation, rapid failure, and isolation of stable code;
- To reduce the internal complexity of project repositories, thus making it easier for Contributors to participate and reducing the scope for error;
- To enforce collective ownership of the project, which increases economic incentive to Contributors and reduces the risk of hijack by hostile entities.

Design

Preliminaries

- The project SHALL use the git distributed revision control system.
- The project SHALL be hosted on github.com or equivalent, herein called the "Platform".
- The project SHALL use the Platform issue tracker.
- The project SHOULD have clearly documented guidelines for code style.
- A "Contributor" is a person who wishes to provide a patch, being a set of commits that solve some clearly identified problem.
- A "Maintainer" is a person who merge patches to the project. Maintainers are not developers; their job is to enforce process.
- Contributors SHALL NOT have commit access to the repository unless they are also Maintainers.
- Maintainers SHALL have commit access to the repository.
- Everyone, without distinction or discrimination, SHALL have an equal right to become a Contributor under the terms of this contract.

Licensing and Ownership

- The project SHALL use the GPLv2 or a variant thereof (LGPL, AGPL).
- All contributions to the project source code ("patches") SHALL use the same license as the project.
- All patches are owned by their authors. There SHALL NOT be any copyright assignment process.
- The copyrights in the project SHALL be owned collectively by all its Contributors.
- Each Contributor SHALL be responsible for identifying themselves in the project Contributor list.

Patch Requirements

- Maintainers and Contributors MUST have a Platform account and SHOULD use their real names or a well-known alias.
- A patch SHOULD be a minimal and accurate answer to exactly one identified and agreed problem.
- A patch MUST adhere to the code style guidelines of the project if these are defined.
- A patch MUST adhere to the "Evolution of Public Contracts" guidelines defined below.
- A patch SHALL NOT include non-trivial code from other projects unless the Contributor is the original author of that code.

- A patch MUST compile cleanly and pass project self-tests on at least the principle target platform.
- A patch commit message SHOULD consist of a single short (less than 50 character) line summarizing the change, optionally followed by a blank line and then a more thorough description.
- A “Correct Patch” is one that satisfies the above requirements.

Development Process

- Change on the project SHALL be governed by the pattern of accurately identifying problems and applying minimal, accurate solutions to these problems.
- To initiate changes, a user SHOULD log an issue on the project Platform issue tracker.
- The user SHOULD write the issue by describing the problem they face or observe.
- The user SHOULD seek consensus on the accuracy of their observation, and the value of solving the problem.
- Users SHALL NOT log feature requests, ideas, suggestions, or any solutions to problems that are not explicitly documented and provable.
- Thus, the release history of the project SHALL be a list of meaningful issues logged and solved.
- To work on an issue, a Contributor SHALL fork the project repository and then work on their forked repository.
- To submit a patch, a Contributor SHALL create a Platform pull request back to the project.
- A Contributor SHALL NOT commit changes directly to the project.
- To discuss a patch, people MAY comment on the Platform pull request, on the commit, or elsewhere.
- To accept or reject a patch, a Maintainer SHALL use the Platform interface.
- Maintainers SHALL NOT accept their own patches.
- Maintainers SHALL NOT make value judgments on correct patches.
- Maintainers SHALL merge correct patches rapidly.
- The Contributor MAY tag an issue as “Ready” after making a pull request for the issue.
- The user who created an issue SHOULD close the issue after checking the patch is successful.
- Maintainers SHOULD ask for improvements to incorrect patches and SHOULD reject incorrect patches if the Contributor does not respond constructively.
- Any Contributor who has value judgments on a correct patch SHOULD express these via their own patches.
- Maintainers MAY commit changes to non-source documentation directly to the project.

Creating Stable Releases

- The project SHALL have one branch (“master”) that always holds the latest in-progress version and SHOULD always build.
- The project SHALL NOT use topic branches for any reason. Personal forks MAY use topic branches.
- To make a stable release someone SHALL fork the repository by copying it and thus become maintainer of this repository.
- Forking a project for stabilization MAY be done unilaterally and without agreement of project maintainers.
- A stabilization project SHOULD be maintained by the same process as the main project.

- A patch to a stabilization project declared “stable” SHALL be accompanied by a reproducible test case.

Evolution of Public Contracts

- All Public Contracts (APIs or protocols) SHOULD be documented.
- All Public Contracts SHOULD have space for extensibility and experimentation.
- A patch that modifies a stable Public Contract SHOULD not break existing applications unless there is overriding consensus on the value of doing this.
- A patch that introduces new features to a Public Contract SHOULD do so using new names.
- Old names SHOULD be deprecated in a systematic fashion by marking new names as “experimental” until they are stable, then marking the old names as “deprecated”.
- When sufficient time has passed, old deprecated names SHOULD be marked “legacy” and eventually removed.
- Old names SHALL NOT be reused by new features.
- When old names are removed, their implementations MUST provoke an exception (assertion) if used by applications.

Project Administration

- The project founders SHALL act as Administrators to manage the set of project Maintainers.
- The Administrators SHALL ensure their own succession over time by promoting the most effective Maintainers.
- A new Contributor who makes a correct patch SHALL be invited to become a Maintainer.
- Administrators MAY remove Maintainers who are inactive for an extended period of time, or who repeatedly fail to apply this process accurately.

Further Reading

- Argyris’ Models 1 and 2 - the goals of C4.1 are consistent with Argyris’ Model 2.
- Toyota Kata - covering the Improvement Kata (fixing problems one at a time) and the Coaching Kata (helping others to learn the Improvement Kata).

References

License

oRFC: 2 Coding Style Guide

OSSEC Coding Style Guide

Name	OSSEC Style Guide
Editor	• Jeremy Rossi <jeremy at jeremyrossi dot com> • Christian Göttsche <cgzones at googlemail dot com>
State	Draft
Origin	• http://zeromq.org/docs:style • https://www.kernel.org/doc/Documentation/CodingStyle

Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119[1].

Goals

The OSSEC Style Guide is meant to provide framework and guide of formating code contributor to OSSEC. The overall goals are:

- Maximize Readability of code in OSSEC;
- Reduction in the number of bugs by removing ambiguity in code and logic flow;
- Be as minimally invasive as possible while achieving the stated goals;
- Trust the contributor.

Code Style

Formatter

- Artistic Style should be used to format the source code.
- Every code style should be achieved by a astyle command argument.

Indentation

- 4 spaces shall be used per indentation level. *–indent=spaces=4*
- Switch and case blocks shall be indented. *–indent-switches*
- Preprocessor conditional statements shall be indented to the same level as the source code. *–indent-preproc-cond*

[1] "Key words for use in RFCs to Indicate Requirement Levels" - http://tools.ietf.org/html/rfc2119

File endings

- Every files shall end with a newline.
- Every files shall have linux like line endings (\n). *–lineend=linux*

Breaking long lines and strings

Placing Braces and Spaces

- Braces shall be placed according to the stroustrup style. *–style=stroustrup*
- Operators shall be padded by a space. *–pad-oper*
- Every branch of conditional statements shall be surrounded by brackets. *–add-brackets*
- Pointer and reference operators shall be attached to the variable name. *–align-pointer=name*

Typedefs & Struct

Naming

- Variables
- Functions
- Typedefs
- Structs

References

5.1.3 Reference

Syntax and Options

Regular Expression Syntax

Currently OSSEC supports two regex syntaxes:

- OS_Regex or regex
- OS_Match or sregex

OS_Regex/regex Syntax

Fast and simple library for regular expressions in C.

This library is designed to be simple, but support the most common regular expressions. It was designed with intrusion detection systems in mind, where having all options is not crucial, but speed is.

Supported expressions:

```
\w  ->  A-Z, a-z, 0-9, '-', '@' characters
\d  ->  0-9 characters
\s  ->  For spaces " "
\t  ->  For tabs.
\p  ->  ()*+,-.:;<=>?[]!"'#$%&|{} (punctuation characters)
\W  ->  For anything not \w
\D  ->  For anything not \d
\S  ->  For anything not \s
\.  ->  For anything
```

Modifiers:

```
+   ->  To match one or more times (eg \w+ or \d+)
*   ->  To match zero or more times (eg \w* or \p*)
```

Special Characters:

```
^ -> To specify the beginning of the text.
$ -> To specify the end of the text.
| -> To create an "OR" between multiple patterns.
```

Characters Escaping

To utilize the following characters they must be escaped:

```
$ -> \$
( -> \(
) -> \)
\ -> \\
| -> \|
```

OS_Match/sregex Syntax

Faster than the OS_Regex/regex, but only supports simple string matching and the following special characters.

Special Characters:

```
^ -> To specify the beginning of the text.
$ -> To specify the end of the text.
| -> To create an "OR" between multiple patterns.
```

Log Analysis Syntax: Rules and Decoders

Rules Syntax

Overview

Options

Decoders Syntax

Overview

Options

ossec.conf: syntax and options

ossec.conf: Active Response Options

Overview

Supported types

Most active-response options are available in the the following installation types:

- server
- local

The `disabled` option is available on all installation types.

Configuration pieces

There are two pieces to an active-response configuration. The first is the `<command>` section. This details the command to be run, and the options it will use. There can be any number of command options.

The second is the `<active-response>` section. This section defines when the command will be run.

Location

All active-response options must be configured in the /var/ossec/etc/ossec.conf and used within the <ossec_config> tag.

XML excerpt to show location:

```
<ossec_config>
    <command>
        <!--
        Command options here
        -->
    </command>
    <active-response>
        <!--
        active-response options here
        -->
    </active-response>
</ossec_config>
```

Command Options

Active-response Options

Example active response configurations:

Command: Restart OSSEC on *nix systems:

This command can be used to restart the OSSEC processes. It's commonly used to automatically restart agent processes when an `agent.conf` is modified. Since no parameters are necessary the `<expect>` is empty.

```
<command>
  <name>restart-ossec</name>
  <executable>restart-ossec.sh</executable>
  <expect></expect>
</command>
```

Active-Response: Restart the OSSEC processes:

This active response will restart the OSSEC processes using the `restart-ossec` command above. It is runs when rule `510010` is triggered, and it runs on the system where the rule was triggered.

```
<active-response>
  <command>restart-ossec</command>
  <location>local</location>
  <rules_id>510010</rules_id>
</active-response>
```

Here is an example rule checking for changes to the `agent.conf`.

```
<rule id="510011" level="10">
  <if_sid>550</if_sid>
  <match>/var/ossec/etc/shared/agent.conf</match>
  <description>agent.conf has been modified</description>
</rule>
```

Command: Block an IP with pf.sh:

`pf.sh` adds an ip (`srcip`) to an `ossec_fwtable` packet filter table. Information on pf tables can be found here.

```
<command>
  <name>pf-block</name>
  <executable>pf.sh</executable>
  <expect>srcip</expect>
</command>
```

This is the minimum configuration necessary to utilize `pf.sh`:

```
table <ossec_fwtable> persist #ossec_fwtable
block in log quick from <ossec_fwtable>
```

Active-Response: Block an IP with pf:

This active-response blocks an IP triggering an `authentication_failed` or `authentication_failures` alert. This active-response will run on agent `001` only.

```
<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>
```

Warning: This may trigger on a single authentication failure.

Command: Run the makelists.sh script:

The `makelists.sh` script runs `/var/ossec/bin/ossec-makelists` to update cdb lists. This command can be triggered by changes in configured cdb lists.

```
<command>
  <name>makelists</name>
  <executable>makelists.sh</executable>
  <expect>hostname</expect>
</command>
```

Active-Response: Update cdb lists:

This active-response will run the `makelists` command to update the cdb lists. This active-response should run only on the OSSEC server since agents do not have cdb lists.

```
<active-response>
  <command>makelists</command>
  <location>server</location>
  <rules_id>510011</rules_id>
</active-response>
```

Rule 510011: This example rule looks for changes to `/var/ossec/lists/blocked.txt` based on syscheck alerts.

```
<rule id="510011" level="10">
  <if_sid>550</if_sid>
  <match>/var/ossec/lists/blocked.txt</match>
  <description>blocked.txt has been modified</description>
</rule>
```

Command: firewall-drop:

This is a command to run the `firewall-drop.sh` script to block the `srcip`.

```
<command>
  <name>firewall-drop</command>
  <executable>firewall-drop.sh</executable>
  <expect>srcip</expect>
</command>
```

Active-Response: Block a srcip:

This active-response will use the `firewall-drop` command to block an IP address that has triggered an `authentication_failed` or `authentication_failures` alert. It will run on all agents, and has a timeout of 600 seconds. It also uses the `repeated_offenders` option blocking an IP for 30 minutes on the second infraction, 60 minutes on the third, etc.

```
<active-response>
  <command>firewall-block</command>
  <location>all</location>
  <rules_group>authentication_failed,authentication_failures</rules_group>
  <timeout>600</timeout>
  <repeated_offenders>30,60,120</repeated_offenders>
</active-response>
```

Warning: This may trigger on a single authentication failure.

ossec.conf: Agentless Options

Overview

Supported types

Agentless options are available in the the following installation types:

- server
- local

Location

All agentless options must be configured in the /var/ossec/etc/ossec.conf and used within the <ossec_config> tag.

XML excerpt to show location:

```
<ossec_config>
    <agentless>
        <!--
        agentless options here
        -->
    </agentless>
</ossec_config>
```

Options

ossec.conf: Alerts Options

Overview

Supported types

Alerts options are available in the the following installation types:

- server
- local

Location

All alerts options must be configured in the /var/ossec/etc/ossec.conf and used within the <ossec_config> tag.

XML excerpt to show location:

```
<ossec_config>
    <alerts>
        <!--
        alerts options here
        -->
    </alerts>
</ossec_config>
```

Options

ossec.conf: Client Options

Overview

Supported types

client options are available in the the following installation types:

- agent

Location

All client options must be configured in the /var/ossec/etc/ossec.conf and used within the <ossec_config> tag.

XML excerpt to show location:

```
<ossec_config>
    <client>
        <!--
        client options here
        -->
    </client>
</ossec_config>
```

Options

ossec.conf: Database Output options

Overview

Supported types

Database Output options are available in the the following installation types:

- server
- local

Location

All database_output options must be configured in the /var/ossec/etc/ossec.conf and used within the <ossec_config> tag.

XML excerpt to show location:

```
<ossec_config>
    <database_output>
        <!--
        Database Output options here
        -->
    </database_output>
</ossec_config>
```

Options

ossec.conf: Granular Email options

Overview

Supported types

Global options are available in the the following installation types:

- server
- local

Notes

Global email configuration is necessary to use the granular email options.

Location

All global options must be configured in the /var/ossec/etc/ossec.conf and used within the <ossec_config> tag.

XML excerpt to show location:

```
<ossec_config>
    <email_alerts>
        <!--
        Email_alerts options here
        -->
    </email_alerts>
</ossec_config>
```

Options

Examples

Example email alerts configurations:

Global Configuration:

```
<global>
  <email_notification>yes</email_notification>
  <email_to>admin@example.com</email_to>
  <smtp_server>127.0.0.1</smtp_server>
  <email_from>ossecm@example.com</email_from>
</global>
```

Global Configuration with a larger maximum emails per hour:

```
<global>
  <email_notification>yes</email_notification>
  <email_to>admin@example.com</email_to>
  <smtp_server>127.0.0.1</smtp_server>
  <email_from>ossecm@example.com</email_from>
  <email_maxperhour>100</email_maxperhour>
</global>
```

Granular Email alert: Level 12 and above:

```
<email_alerts>
  <email_to>other_admin@example.com</email_to>
  <level>12</level>
</email_alerts>
```

Syscheck alerts to syscheck admin address:

```
<email_alerts>
  <email_to>syscheck-admin@example.com</email_to>
  <group>syscheck</group>
</email_alerts>
```

Level 15 alerts from agent007 without delay or grouping:

```
<email_alerts>
  <email_to>bond@example.com</email_to>
  <event_location>agent007</event_location>
  <level>15</level>
  <do_not_delay />
  <do_not_group />
</email_alerts>
```

ossec.conf: Global options

Overview

Supported types

Global options are available in the the following installation types:

- server
- local

Location

All global options must be configured in the /var/ossec/etc/ossec.conf and used within the <ossec_config> tag.

XML excerpt to show location:

```
<ossec_config>
    <global>
        <!--
        Global options here
        -->
    </global>
</ossec_config>
```

Options

Note: If the *smtp_server* entry contains a hostname, */etc/resolv.conf* will probably have to be copied to OSSEC's *etc* directory (*/var/ossec/etc* by default).

Warning: This feature first appeared in OSSEC 2.9.

ossec.conf: Localfile options

Overview

Supported types

Localfile options are available in the the following installation types:

- server
- local

Location

All localfile options must be configured in the /var/ossec/etc/ossec.conf or /var/ossec/etc/shared/agent.conf and used within the <ossec_config> or <agent_config> tags.

XML excerpt to show location:

```
<ossec_config>
    <localfile>
        <!--
        Localfile options here
        -->
    </localfile>
</ossec_config>
```

Options

ossec.conf: Remote Options

Overview

Supported types

remote options are available in the the following installation types:

- server

Location

All remote options must be configured in the /var/ossec/etc/ossec.conf and used within the <ossec_config> tag.

XML excerpt to show location:

```
<ossec_config>
    <remote>
        <!--
        remote options here
        -->
    </remote>
</ossec_config>
```

Options

ossec.conf: Reports options

Overview

Supported types

Reports options are available in the the following installation types:

- server
- local

Location

All reports options must be configured in the /var/ossec/etc/ossec.conf and used within the <ossec_config> tag.

XML excerpt to show location:

```
<ossec_config>
    <reports>
        <!--
        Reports options here
        -->
    </reports>
</ossec_config>
```

Options

ossec.conf: Rootcheck options

Overview

Supported types

rootcheck options are available in the the following installation types:

- server
- local
- agent

Location

All rootcheck options must be configured in the /var/ossec/etc/ossec.conf or /var/ossec/etc/shared/agents.conf and used within the <ossec_config> tag.

XML excerpt to show location if part of ossec.conf:

```
<ossec_config>
    <rootcheck>
        <!--
        rootcheck options here
        -->
    </rootcheck>
</ossec_config>
```

XML excerpt to the Location if part of agent.conf

```
<agent_config>
    <rootcheck>
        <!--
        rootcheck options here
        -->
    </rootcheck>
</agent_config>
```

Options

ossec.conf: Rules options

Overview

Supported types

Rules options are available in the the following installation types:

- server
- local

Location

All global options must be configured in the /var/ossec/etc/ossec.conf and used within the <ossec_config> tag.

XML excerpt to show location:

```
<ossec_config>
    <rules>
        <!--
        Rules options here
        -->
    </rules>
</ossec_config>
```

Options

ossec.conf: Syscheck Options

Overview

Supported types

Syscheck options are available in the the following installation types:

- server
- local
- agent

Location

All global options must be configured in the /var/ossec/etc/ossec.conf and used within the <ossec_config> tag.

XML excerpt to show location:

```
<ossec_config>
    <syscheck>
        <!--
        Syscheck options here
        -->
    </syscheck>
</ossec_config>
```

Options

directories

Use this option to add or remove directories to be monitored (they must be comma separated). All files and subdirectories will also be monitored. Drive letters without directories are not valid. At a minimum the '.' should be included (`D:\.`). This should be set on the system you wish to monitor (or in the agent.conf if appropriate).

Default: /etc,/usr/bin,/usr/sbin,/bin,/sbin

Attributes:

- **realtime**: Value=yes
  - This will enable realtime/continuous monitoring on Linux (using the inotify system calls) and Windows systems.
- **report_changes**: Value=yes
  - Report diffs of file changes. This is limited to text files at this time.

  Note: This option is only available on Unix-like systems.

- **check_all**: Value=yes
  - All the following check_* options are used together unless a specific option is explicitly overridden with "no".
- **check_sum**: Value=yes
  - Check the md5 and sha1 hashes of the of the files will be checked.

    This is the same as using both check_sha1sum="yes" and check_md5sum="yes"

- **check_sha1sum**: Value=yes
  - When used only the sha1 hash of the files will be checked.
- **check_md5sum**: Value=yes
  - The md5 hash of the files will be checked.
- **check_size**: Value=yes
  - The size of the files will be checked.
- **check_owner**: Value=yes
  - Check the owner of the files selected.
- **check_group**: Value=yes
  - Check the group owner of the files/directories selected.
- **check_perm**: Value=yes
  - Check the UNIX permission of the files/directories selected. On windows this will only check the POSIX permissions.
- **restrict**: Value=string
  - A string that will limit checks to files containing that string in the file name.

  Allowed: Any directory or file name (but not a path)
- **no_recurse**: Value=no

  New in version 3.2.
  - Do not recurse into the defined directory.

  Allowed: yes/no

`ignore`

List of files or directories to be ignored (one entry per element). The files and directories are still checked, but the results are ignored.

Default: /etc/mtab

Attributes:

- **type**: Value=sregex
  - This is a simple regex pattern to filter out files so alerts are not generated.

Allowed: Any directory or file name

`nodiff`

New in version 3.0.

List of files to not attach a diff. The files are still checked, but no diff is computed. This allows to monitor sensitive files like private key or database configuration without leaking sensitive data through alerts.

Attributes:

- **type**: Value=sregex
  - This is a simple regex pattern to filter out files so alerts are not generated.

Allowed: Any directory or file name

`frequency`
Frequency that the syscheck is going to be executed (in seconds).

The default is 6 hours or 21600 seconds

Default: 21600

Allowed: Time in seconds

`scan_time`
Time to run the scans (can be in the formats of 21pm, 8:30, 12am, etc).

Allowed: Time to run scan

Note: This may delay the initialization of realtime scans.

`scan_day`
Day of the week to run the scans (can be in the format of sunday, saturday, monday, etc)

Allowed: Day of the week

`auto_ignore`
Specifies if syscheck will ignore files that change too often (after the third change)

Default: yes

Allowed: yes/no

Valid: server, local

`alert_new_files`
Specifies if syscheck should alert on new files created.

Default: no

Allowed: yes/no

Valid: server, local

Note: New files will only be detected on a full scan, this option does not work in realtime.

`scan_on_start`
Specifies if syscheck should do the first scan as soon as it is started.

Default: yes

Allowed: yes/no

`windows_registry`
Use this option to add Windows registry entries to be monitored (Windows-only).

Default: HKEY_LOCAL_MACHINESoftware

Allowed: Any registry entry (one per element)

Note: New entries will not trigger alerts, only changes to existing entries.

`registry_ignore`
List of registry entries to be ignored.

Default: ..CryptographyRNG

Allowed: Any registry entry (one per element)

prefilter_cmd

Command to run to prevent prelinking from creating false positives.

Example:

```
<prefilter_cmd>/usr/sbin/prelink -y</prefilter_cmd>
```

Note: This option can potentially impact performance negatively. The configured command will be run for each and every file checked.

skip_nfs

New in version 2.9.0.

Specifies if syscheck should scan network mounted filesystems. Works on Linux and FreeBSD. Currently skip_nfs will abort checks running against CIFS or NFS mounts.

Default: no

Allowed: yes/no

Warning: This option was added in OSSEC 2.9.

ossec.conf: Syslog Output options

Overview

Supported types

Syslog Output options are available in the the following installation types:

- server
- local

Location

All syslog_output options must be configured in the /var/ossec/etc/ossec.conf and used within the <ossec_config> tag.

XML excerpt to show location:

```
<ossec_config>
    <syslog_output>
        <!--
        Syslog Output options here
        -->
    </syslog_output>
</ossec_config>
```

Options

agent.conf

Overview

More information on using the agent.conf can be found here

Supported types

The agent.conf is valid on the server install only.

Location

The `agent.conf` exists in `/var/ossec/etc/shared`. It should be readable by the ossec user.

```
-r-xr-x---  1 root  ossec  10908 Aug 12 16:06 /var/ossec/etc/shared/agent.conf
```

XML excerpt to show location:

```
<agent_config>
    ...
</agent_config>
```

Options

internal_options.conf: syntax and options

The */var/ossec/etc/internal_options.conf* file includes several options not present int he *ossec.conf*. */var/ossec/etc/local_internal_options.conf* can be used for changes to the defaults, and this file will not be overwritten during an upgrade.

internal_options.conf: analysisd

internal_options.conf: agent

internal_options.conf: dbd

internal_options.conf: logcollector

internal_options.conf: maild

internal_options.conf: monitord

internal_options.conf: remoted

internal_options.conf: syscheck

internal_options.conf: windows

Output Formats

OSSEC alert log samples

Example alert.log messages:

```
** Alert 1510376401.0: - syslog,errors,
2017 Nov 11 00:00:01 ix->/var/log/messages
Rule: 1005 (level 5) -> 'Syslogd restarted.'
Nov 11 00:00:01 ix syslogd[72090]: restart

** Alert 1510376417.172: - syslog,smtpd,
2017 Nov 11 00:00:17 (junction) 192.168.17.17->/var/log/maillog
Rule: 53508 (level 5) -> 'Server TLS certificate verification failed.'
Nov 11 00:00:16 junction smtpd[86532]: smtp-out: Server certificate verification␣
↪failed on session 99fc1afc58067419

** Alert 1510376428.465: - syslog,sudo
2017 Nov 11 00:00:28 ubnt->/var/log/syslog-ng/messages
Rule: 5402 (level 3) -> 'Successful sudo to ROOT executed'
User: root
Nov  5 15:35:03 ubnt sudo:     root : TTY=unknown ; PWD=/ ; USER=root ; COMMAND=/usr/
↪bin/vtysh.pl -c show ip route summary json

** Alert 1510376428.758: - pam,syslog,authentication_success,
2017 Nov 11 00:00:28 ubnt->/var/log/syslog-ng/messages
Rule: 5501 (level 3) -> 'Login session opened.'
Nov  5 15:35:03 ubnt sudo: pam_unix(sudo:session): session opened for user root by␣
↪(uid=0)
```

Sample alerts.json messages:

```
{"rule":{"level":5,"comment":"Syslogd restarted.","sidid":1005,"group":"syslog,errors,
↪"},"id":"1510376401.0","TimeStamp":1510376401000,"location":"/var/log/messages",
↪"full_log":"Nov 11 00:00:01 ix syslogd[72090]: restart","hostname":"ix","program_
↪name":"syslogd"}
{"rule":{"level":5,"comment":"Server TLS certificate verification failed.","sidid
↪":53508,"group":"syslog,smtpd,"},"id":"1510376417.172","TimeStamp":1510376417000,
↪"decoder":"smtpd","location":"(junction) 192.168.17.17->/var/log/maillog","full_log
↪":"Nov 11 00:00:16 junction smtpd[86532]: smtp-out: Server certificate verification␣
↪failed on session 99fc1afc58067419","hostname":"(junction) 192.168.17.17->/var/log/
↪maillog","program_name":"smtpd"}
{"rule":{"level":3,"comment":"Successful sudo to ROOT executed","sidid":5402,"group":
↪"syslog,sudo"},"id":"1510376428.465","TimeStamp":1510376428000,"decoder":"sudo",
↪"srcuser":"root","dstuser":"root","location":"/var/log/syslog-ng/messages","full_log
↪":"Nov  5 15:35:03 ubnt sudo:     root : TTY=unknown ; PWD=/ ; USER=root ; COMMAND=/
↪usr/bin/vtysh.pl -c show ip route summary json","url":"/","status":"/usr/bin/vtysh.
↪pl -c show ip route summary json","hostname":"ubnt","program_name":"sudo"}
{"rule":{"level":3,"comment":"Login session opened.","sidid":5501,"group":"pam,syslog,
↪authentication_success,"},"id":"1510376428.758","TimeStamp":1510376428000,"decoder":
↪"pam","location":"/var/log/syslog-ng/messages","full_log":"Nov  5 15:35:03 ubnt␣
↪sudo: pam_unix(sudo:session): session opened for user root by (uid=0)","hostname":
↪"ubnt","program_name":"sudo"}
```

(continues on next page)

(continued from previous page)

```
{"rule":{"level":3,"comment":"Login session closed.","sidid":5502,"group":"pam,syslog,
↪"},"id":"1510376430.1015","TimeStamp":1510376430000,"decoder":"pam","location":"/
↪var/log/syslog-ng/messages","full_log":"Nov  5 15:35:04 ubnt sudo: pam_
↪unix(sudo:session): session closed for user root","hostname":"ubnt","program_name":
↪"sudo"}
{"rule":{"level":3,"comment":"Successful sudo to ROOT executed","sidid":5402,"group":
↪"syslog,sudo"},"id":"1510376490.1239","TimeStamp":1510376490000,"deco
der":"sudo","srcuser":"root","dstuser":"root","location":"/var/log/syslog-ng/messages
↪","full_log":"Nov  5 15:36:04 ubnt sudo:     root : TTY=unknown ; PWD=/ ; USER=root␣
↪; COMMAND=/usr/bin/vtysh.pl -c show ip route summary json","url":"/","status":"/usr/
↪bin/vtysh.pl -c show ip route summary json","hostname":"ubnt","program_name":"sudo"}
{"rule":{"level":3,"comment":"Login session opened.","sidid":5501,"group":"pam,syslog,
↪authentication_success,"},"id":"1510376490.1533","TimeStamp":1510376490000,"decoder
↪":"pam","location":"/var/log/syslog-ng/messages","full_log":"Nov  5 15:36:04 ubnt␣
↪sudo: pam_unix(sudo:session): session opened for user root by (uid=0)","hostname":
↪"ubnt","program_name":"sudo"}
{"rule":{"level":3,"comment":"Login session closed.","sidid":5502,"group":"pam,syslog,
↪"},"id":"1510376490.1791","TimeStamp":1510376490000,"decoder":"pam","location":"/
↪var/log/syslog-ng/messages","full_log":"Nov  5 15:36:05 ubnt sudo: pam_
↪unix(sudo:session): session closed for user root","hostname":"ubnt","program_name":
↪"sudo"}
{"rule":{"level":3,"comment":"Successful sudo to ROOT executed","sidid":5402,"group":
↪"syslog,sudo"},"id":"1510376550.2015","TimeStamp":1510376550000,"decoder":"sudo",
↪"srcuser":"root","dstuser":"root","location":"/var/log/syslog-ng/messages","full_log
↪":"Nov  5 15:37:05 ubnt sudo:     root : TTY=unknown ; PWD=/ ; USER=root ; COMMAND=/
↪usr/bin/vtysh.pl -c show ip route summary json","url":"/","status":"/usr/bin/vtysh.
↪pl -c show ip route summary json","hostname":"ubnt","program_name":"sudo"}
```

JSON Format

At this time we have one alert JSON formatted messages.

Also see manual/output/json-alert-log-output.

```
{
  "rule":1000,
  "level":1,
  "comment":"This is a comment",
  "sidid":1111,
  "cve":"cve-1001-1000",
  "action":"drop",
  "srcip":"10.1.1.1",
  "srcport":"1000",
  "srcuser":"root",
  "dstip":"10.2.2.2",
  "dstport":"2000",
  "dstuser":"root",
  "location":"/var/log/auth.log",
  "full_log":"This is the full log message",
  "file":{
    "path":"/etc/",
    "md5_before":"XXXXXXXXXXXXXXXXXXX",
    "md5_after":"YYYYYYYYYYYYYYYYYYYY",
    "sha1_before":"XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX",
    "sha1_after":"YYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY",
```

(continues on next page)

(continued from previous page)

```
    "owner_before":"root",
    "owner_after":"nobody",
    "gowner_before":"root",
    "gowner_after":"nodody",
    "perm_before":660,
    "perm_after":777,
  }
}
```

cef log format:

Man pages

agent-auth

The agent-auth program is the client application used with *ossec-authd* to automatically add agents to an OSSEC 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.

agent-auth argument options

-A `<agent_name>`
: Agent name to be used. **Default** hostname

-D
: Directory where OSSEC is installed. **Default** /var/ossec

-d
: Execute agent-auth in debug mode. This option can be used multiple times to increase the verbosity of the debug messages.

-g `<group>`
: Run as `group`.

-h
: Display the help message

-k `<path>`
: Full path to the agent key.

-m `<manager_ip>`
: IP address of the manager.

-p `<port>`
: Port ossec-authd is running on.

  Default 1515

-V
: Display OSSEC Version and license information.

-v `<path>`

Full path to the CA certificate used to verify the server.

Note: This option was added after 2.8.1.

-x `<path>`
Full path to the agent certificate.

Note: The following options are only necessary if verification of server or client certificates is desired. See *Optional Server Authentication* and *Optional Client Authentication* below.

Note: This option was added after 2.8.1.

-x `</path/to/certificate>`
Load the PEM encoded certificate that will be presented to *ossec-authd* during establishment of the SSL connection.

Note: This option was added after 2.8.1.

-k `</path/to/private_key>`
Load the certificate's corresponding PEM encoded private key.

Note: This option was added after 2.8.1.

-v `</path/to/CA_certificate>`
Load the PEM encoded CA Certificate that will be used to verify *ossec-authd* if desired. If this option is used then *ossec-authd* must present a valid certificate signed by this CA.

Note: This option was added after 2.8.1.

Optional Server Authentication

`agent-auth` can verify that the server it's connecting to presents a valid X.509 certificate when requesting a key. This is optional and is only useful if hosts in your environment have access to the root certificate of the CA that signed the certificate presented by *ossec-authd*. If server certificate verification is desired then the relevant CA certificate must be loaded with the -v option, then if the server does not present a valid certificate the agent will not be allocated a key.

A certificate presented by the server may be found to be invalid for the following reasons:

- It was not signed by the specified CA.
- It doesn't contain the IP address or hostname given with the -m option in the subject's common name field or a subject alternative name extension field.
- It is expired.

While server authentication is optional it is highly recommended that it be used if possible when running ossec-authd and agent-auth.

Optional Client Authentication

`agent-auth` can present its own certificate to the server for verification. This is mandatory if *ossec-authd* was run with the -v option and optional otherwise. This is only useful if hosts in your environment are assigned certificates when they're provisioned (or at some point before being added to OSSEC). Use the -x and -k options to load a certificate and private key.

agent-auth example usage

Example: Adding an agent with a hostname

```
# /var/ossec/bin/agent-auth -m 192.168.1.1 -p 1515 -A example-agent
INFO: Connected to 192.168.1.1:1515
INFO: Using agent name as: melancia
INFO: Send request to manager. Waiting for reply.
INFO: Received response with agent key
INFO: Valid key created. Finished.
INFO: Connection closed.
```

Example: Adding an agent and verifying the certificate presented by ossec-authd

```
# /var/ossec/bin/agent-auth -m ossec-manager.localdomain -p 1515 -v /etc/pki/CA/certs/
↪internal_CA.cert
INFO: Connected to 192.168.1.1:1515
INFO: Verifying manager's certificate
INFO: Using agent name as: melancia
...
```

Example: Adding an agent and presenting a certificate to ossec-authd

```
# /var/ossec/bin/agent-auth -m ossec-manager.localdomain -p 1515 -x /var/ossec/etc/
↪client.cert -k /var/ossec/etc/client.key
INFO: Connected to 192.168.1.1:1515
INFO: Using agent name as: melancia
...
```

agent_control

The agent_control tool allows you to query and get information from any agent you have configured on your server and it also allows you to restart (run now) the syscheck/rootcheck scan on any agent.

Enabling active response will be necessary to start scans remotely and possibly other functions.

agent_control argument options

-h
Display the help message

-l
: List available (active or not) agents

-lc
: List active agents

-i `<agent_id>`
: Extracts information from an agent

-R `<agent_id>`
: Restarts the OSSEC processes on the agent

> **Note:** Requires active response to be enabled.

-r
: Run the integrity/rootcheck checking on agents. Must be utilized with `agent_control -a` or `agent_control -u`

> **Note:** Requires active response to be enabled.

-a
: Utilizes all agents.

-u `<agent_id>`
: <agent_id> that will perform the requested action.

agent_control example usage

Example 1: Listing all active agents

The first interesting argument is `agent_control -lc`, to list the connected (active agents). To list all of them, use `agent_control -l` only.

```
# /var/ossec/bin/agent_control -lc
OSSEC HIDS agent_control. List of available agents:
ID: 000, Name: enigma.ossec.net (server), IP: 127.0.0.1, Active/Local
ID: 002, Name: winhome, IP: 192.168.2.190, Active
ID: 005, Name: jul, IP: 192.168.2.0/24, Active
ID: 165, Name: esqueleto2, IP: 192.168.2.99, Active
ID: 174, Name: lili3win, IP: 192.168.2.0/24, Active
```

Example 2: Querying information from agent 002

To query an agent, just use the `agent_control -i` option followed by the agent id.

```
# /var/ossec/bin/agent_control -i 002

OSSEC HIDS agent_control. Agent information:
Agent ID: 002
Agent Name: winhome
IP address: 192.168.2.190
Status: Active
```

(continues on next page)

(continued from previous page)

```
Operating system: Microsoft Windows XP Professional (Build 2600)
Client version: OSSEC HIDS v1.5-SNP-080412
Last keep alive: Fri Apr 25 14:33:03 2008

Syscheck last started at: Fri Apr 25 05:07:13 2008
Rootcheck last started at: Fri Apr 25 09:04:12 2008
```

Example 3: Executing syscheck and rootcheck scan immediately

To execute the syscheck/rootcheck scan immediately, use the *agent_control -r* option followed by the *agent_control -u* with the agent id.

```
# /var/ossec/bin/agent_control -r -u 000

OSSEC HIDS agent_control: Restarting Syscheck/Rootcheck locally.
```

clear_stats

Clear the events stats

clear_stats argument options

-h
Print and display a help message of all available options to clear_stats

-a
Clear all the stats (averages).

-d
Clear the daily averages.

-w
Clear the weekly averages.

list_agents

OSSEC HIDS list_agents: List available agents.

list_agents is only available on OSSEC servers or local mode installations. It can be used to retrieve

- a list of all OSSEC agents that successfully connected to the server in the past
- a list of all OSSEC agents currently connected to the server
- a list of all OSSEC agents that were connected to the server in the past but are currently not connected.

If an agent was added via the *manage_agents* tool but has not yet been connected to the server, it will not show up in the output of list_agents.

list_agents argument options

-h
Display the help message.

-a
List all agents.

-c
List the connected (active) agents.

-n
List the not connected (active) agents.

manage_agents

manage_agents is available in two versions:

- a version for OSSEC server installations
- a version for OSSEC agent installations

The purpose of manage_agents is to provide an easy-to-use interface to handle authentication keys for OSSEC agents. These authentication keys are required for secure (encrypted and authenticated) communication between the OSSEC server and its affiliated agent instances.

manage_agents argument options

-V
Display OSSEC Version.

-h
Display the help message.

-l
List available agents.

-e `<agent_id>`
Extracts key for an agent (Manager only).

-r `<agent_id>`
Remove an agent (Manager only).

-i `<key>`
Import authentication key (Agent only).

-f `<file>`
Generate clients in bulk from <file> (Manager only). The file is a comma delimited file containing the IP addresses and agent names to be added. This file should be located within `/var/ossec`, and referenced by its path relative to `/var/ossec`.

Example:

```
# cat /var/ossec/k
192.168.1.2,host02
192.168.1.3,host03

# /var/ossec/bin/manage_agents -f /k
```

(continues on next page)

(continued from previous page)

```
Bulk load file: /k
Opening: [/k]
Agent information:
   ID:002
   Name:host02
   IP Address:192.168.1.2

Agent added.
Agent information:
   ID:003
   Name:host03
   IP Address:192.168.1.3

Agent added.
```

Usage

The OSSEC manual goes into details on usage of this command at *Managing Agents*

ossec-agentd

`ossec-agentd` is the client side daemon that communicates with the server. It runs as ossec and is chrooted to `/var/ossec` by default.

ossec-agentd argument options

-c `<config>`
  Run `ossec-agentd` using <config> as the configuration file.

  Default: /var/ossec/etc/ossec.conf

-D `<dir>`
  Chroot to <dir>.

  Default: /var/ossec

-d
  Run in debug mode. This option can be used multiple times to increase the verbosity of the debug messages.

-f
  Run `ossec-agentd` in the foreground.

-g `<group>`
  Run `ossec-agentd` as <group>.

-h
  Display the help message.

-t
  Test configuration.

-u `<user>`
  Run `ossec-agentd` as <user>.

  Default: ossecm

-V
Version and license information.

ossec-agentlessd

ossec-agentlessd argument options

-c `<config>`
Read the configuration from file `<config>`.

-D `<dir>`
chroot to `<dir>`.

-d
Execute ossec-agentlessd in debug mode. This option can be used multiple times to increase the verbosity of the debug messages.

-f
Run ossec-agentlessd in the foreground.

-g `<group>`
Run as `group`.

-h
Display a help message.

-t
Test the configuration.

-u
Run as `user`.

-V
Display OSSEC Version and license information.

ossec-analysisd

`ossec-analysisd` receives the log messages and compares them to the rules. It will create alerts when a log message matches an applicable rule.

ossec-analysisd argument options

-c `<config>`
Configuration file ossec-analysisd should use.

-D `<dir>`
Chroot to `<dir>`.

-d
Execute ossec-analysisd in debug mode. This can be used more than once to increase the verbosity of the debug messages.

-f
Run ossec-agentlessd in the foreground.

-g `<group>`
Run as `group`.

-h
Display a help message.

-t
Test the configuration.

-u
Run as `user`.

-V
Display the version and license information.

ossec-authd

The ossec-authd daemon will automatically add an agent to an OSSEC manager and provide the key to the agent. The *agent-auth* application is the client application used with ossec-authd. *ossec-authd* will create 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.

ossec-authd argument options

-D `<dir>`
chroot to `<dir>`.

-d
Execute ossec-authd in debug mode. This option can be used multiple times to increase the verbosity of the debug messages.

-g `<group>`
Run as `group`.

-h
Display a help message.

-i
Add agents with a specific IP address instead of using `any`.

-k `<path>`
Full path to the server key.

-p `<port>`
Listen on port.

Default 1515

-t
Test the configuration.

-V
Display OSSEC Version and license information.

-v `<path>`
Full path to the CA certificate used to verify the clients.

Note: This option is available in OSSEC 2.9.

-x <path>
Full path to the server certificate.

Note: This option is available in OSSEC 2.9.

Creating SSL keys

`ossec-authd` requires SSL keys to run. This process will create the necessary keys in `/var/ossec/etc` and allow `ossec-authd` to start:

```
# openssl genrsa -out /var/ossec/etc/sslmanager.key 2048
# openssl req -new -x509 -key /var/ossec/etc/sslmanager.key -out /var/ossec/etc/
↪sslmanager.cert -days 365
```

Without the key `ossec-authd` will give the following error:

```
[user@ossec-manager] :; sudo /var/ossec/bin/ossec-authd
2012/04/18 11:05:01 ossec-authd: INFO: Started (pid: 20669).
2012/04/18 11:05:01 ossec-authd: ERROR: Unable to read certificate file (not found): /
↪var/ossec/etc/sslmanager.cert
2012/04/18 11:05:01 ossec-authd: ERROR: SSL error. Exiting.
```

If the default locations of /var/ossec/etc/sslmanager.cert and /var/ossec/etc/sslmanager.key are not suitable then the -x and -k options can be used to specify alternative locations.

Optional Client Authentication

`ossec-authd` can verify that connecting agents present a valid X.509 certificate when requesting a key. This is optional and is only useful if hosts in your environment are assigned certificates when they're provisioned (or at some point before being added to OSSEC). If agent certificate verification is desired then the relevant CA certificate must be loaded with the -v option. This will cause `ossec-authd` to verify that agents present a valid certificate when requesting a key. If an agent does not present a certificate or presents an invalid certificate then the agent will not be allocated a key.

A certificate presented by an agent may be found to be invalid for the following reasons:

- It was not signed by the specified CA.
- It is expired.

ossec-authd example usage

Example: Running ossec-authd

```
# /var/ossec/bin/ossec-authd -p 1515 >/dev/null 2>&1 &
```

And the logs when an agent is added:

```
2011/01/19 15:04:40 ossec-authd: INFO: New connection from 192.168.10.5
2011/01/19 15:04:41 ossec-authd: INFO: Received request for a new agent (example-
↪agent) from: 192.168.10.5
2011/01/19 15:04:41 ossec-authd: INFO: Agent key generated for example-agent␣
↪(requested by 192.168.10.5)
2011/01/19 15:04:41 ossec-authd: INFO: Agent key created for example-agent (requested␣
↪by 192.168.10.5)
```

Example: Running ossec-authd with client authentication

```
# /var/ossec/bin/ossec-authd -v /var/ossec/etc/CA.cert -d
```

If debug output is enabled then "Peer verification requested" will be displayed when starting.

```
2014/06/07 17:04:56 ossec-authd: DEBUG: Starting ...
2014/06/07 17:04:56 ossec-authd: INFO: Started (pid: 2043).
2014/06/07 17:04:56 ossec-authd: DEBUG: Peer verification requested.
2014/06/07 17:04:56 ossec-authd: DEBUG: Returning CTX for server.
2014/06/07 17:04:56 ossec-authd: DEBUG: Going into listening mode.
2014/06/07 17:04:58 ossec-authd: INFO: New connection from 192.168.10.5
2014/06/07 17:04:58 ossec-authd: INFO: Received request for a new agent (example-
↪agent) from: 192.168.10.5
2014/06/07 17:04:58 ossec-authd: INFO: Agent key generated for example-agent␣
↪(requested by 192.168.10.5)
2014/06/07 17:04:58 ossec-authd: INFO: Agent key created for example-agent (requested␣
↪by 192.168.10.5)
2014/06/07 17:04:58 ossec-authd: DEBUG: Process 2044 exited
```

ossec-control

`ossec-control` is a script to start, stop, configure, or check on the status of OSSEC processes. `ossc-control` can enable or disable client-syslog, database logging, agentless configurations, and debug mode.

ossec-control argument options

start Start the OSSEC processes.

stop Stop the OSSEC processes.

restart Restart the OSSEC processes.

reload Restart all OSSEC processes except `ossec-execd`. This allows an agent to reload without losing active response status.

Note: This is only available on an OSSEC agent.

status Determine which OSSEC processes are running.

enable Enable OSSEC functionality.

database Enable the `ossec-dbd` daemon for logging to a database.

Available: Server and local installs only.

Note: Database support must be compiled in at install time.

client-syslog Enable `ossec-csyslogd` for logging to remote syslog.

Available: Server and local installs only.

agentless Enable `ossec-agentlessd` for running commands on systems without OSSEC agents.

Available: Server and local installs only.

debug Run all OSSEC daemons in debug mode.

disable Disable OSSEC functionality.

database Disable the `ossec-dbd` daemon for logging to a database.

Available: Server and local installs only.

Note: Database support must be compiled in at install time.

client-syslog

Disable `ossec-csyslogd` for logging to remote syslog.

Available: Server and local installs only.

agentless Disable `ossec-agentlessd` for running commands on systems without OSSEC agents.

Available: Server and local installs only.

debug Turn off debug mode.

ossec-control example usage

Example: Running ossec-control

```
# /var/ossec/bin/ossec-control

Usage: /var/ossec/bin/ossec-control {start|stop|restart|status|enable|disable}
```

ossec-csyslogd

`ossec-csyslogd` is a daemon that forwards the OSSEC alerts via syslog. Configuration is done in the `<syslog_output>` section of the ossec.conf. (see *ossec.conf: Syslog Output options*)

ossec-csyslogd argument options

-c `<config>`
Run `ossec-csyslogd` using <config> as the configuration file.

Default: /var/ossec/etc/ossec.conf

-D `<dir>`
Chroot to <dir>.

Default: /var/ossec

-d
Execute ossec-csyslogd in debug mode. This option can be used multiple times to increase the verbosity of the debug messages.

-f
Run `ossec-csyslogd` in the foreground.

-g `<group>`
Run `ossec-csyslogd` as <group>.

-h
Display the help message.

-t
Test configuration.

-u `<user>`
Run `ossec-csyslogd` as <user>.

Default: ossecm

-V
Version and license information.

ossec-dbd

The `ossec-dbd` daemon inserts the alert logs into a database, either postgresql or mysql. `ossec-dbd` is configured in ossec.conf. (see *ossec.conf: Database Output options*)

ossec-dbd argument options

-c `<config>`
Run `ossec-dbd` using <config> as the configuration file.

Default: /var/ossec/etc/ossec.conf

-D `<dir>`
Chroot to <dir>.

Default: /var/ossec

-d
Execute ossec-dbd in debug mode. This option can be used multiple times to increase the verbosity of the debug messages.

-f
Run `ossec-dbd` in the foreground.

-g `<group>`
Run `ossec-dbd` as <group>.

-h
Display the help message.

-t
: Test configuration.

-u `<user>`
: Run `ossec-dbd` as <user>.

  Default: ossecm

-V
: Version and license information.

ossec-execd

`ossec-execd` executes active responses by running the configured scripts. `ossec-execd` is configured in the ossec.conf. (see *ossec.conf: Active Response Options*)

ossec-execd argument options

-c `<config>`
: Run `ossec-execd` using <config> as the configuration file.

  Default: /var/ossec/etc/ossec.conf

-d
: Execute ossec-execd in debug mode. This option can be used multiple times to increase the verbosity of the debug messages.

-f
: Run `ossec-execd` in the foreground.

-g
: Run as `group`.

-h
: Display the help message.

-t
: Test configuration.

-V
: Version and license information.

ossec-logcollector

The `ossec-logcollector` daemon monitors configured files and commands for new log messages. `ossec-logcollector` is configured in ossec.conf. (see *ossec.conf: Localfile options*)

ossec-logcollector argument options

-c `<config>`
: Run `ossec-logcollector` using <config> as the configuration file.

  Default: /var/ossec/etc/ossec.conf

-d
: Execute ossec-logcollector in debug mode. This option can be used multiple times to increase the verbosity of the debug messages.

-f
: Run `ossec-logcollector` in the foreground.

-h
: Display the help message.

-t
: Test configuration.

-V
: Version and license information.

ossec-logtest

ossec-logtest is the single most useful tool when working with ossec. This tool allows oneself to test and verify log files in the exact same way that `ossec-anaylistd` does.

Something ossec-logtest can help with:

- Writing rules (Debugging your custom rules)
- Troubleshooting false positives or false negatives

ossec-logtest accepts standard input for all log to test.

osssec-logtest argument options

-a
: Analyze of input lines as if they are live events.

-c `<config>`
: <config> is the path and filename to load in place of the default /var/ossec/etc/ossec.conf.

-D `<dir>`
: This is the path that ossec-logtest will chroot to before it completes loading all rules, decoders, and lists and processing standard input.

-d
: Print debug output to the terminal. This option can be used multiple times to increase the verbosity of the debug messages.

-h
: Print the help message to the console.

-t
: Test configuration. This will print file details on the ossec-anaylistd rules, 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 with an exit status other then zero unless the last line tested matches the arguments passed.

Note: This only works for the last, line so passing many lines into ossec-logtest with this argument may not provide the desired results.

Note: This ossec-logtest code requires access to all ossec configuration files. This is a bug and will be corrected. It is documented in issue #‘61 <https://bitbucket.org/jbcheng/ossec-hids/issue/61/ossec-logtest-must-be-used-with-a-full>‘_

```
% echo "Aug 29 15:33:13 ns3 named[464]: client 217.148.39.3#1036: query (cache)␣
↪denied" | sudo /var/ossec/bin/ossec-logtest -U 12108:0:named 2>&1 > /dev/null
% echo $?
0
% echo "Aug 29 15:33:13 ns3 XXXXXX[464]: client 217.148.39.3#1036: query (cache)␣
↪denied" | sudo /var/ossec/bin/ossec-logtest -U 12108:0:named 2>&1 > /dev/null
% echo $?
3
```

-V

Print the Version and license message for OSSEC and ossec-logtest.

-v

Full output of all details and matches.

Note: This the key argument to troubleshoot a rule, decoder problem.

Note: This is argument was incorrectly displayed as running in the foreground in all version before version 2.5

Caveats

Some log formats will be processed differently than they appear in the log file. MySQL log files for instance will have *MySQL log:* prepended to the log message before analysis. If using ossec-logtest to test MySQL logs, please add this string to the beginning.

Example:

Given the following MySQL log message:

```
130218 12:07:52 [Warning] Unsafe statement written to the binary log using statement␣
↪format since BINLOG_FORMAT = STATEMENT. The statement is unsafe because it uses a␣
↪LIMIT clause. This is unsafe because the set of rows included cannot be predicted.␣
↪Statement: DELETE FROM `emailQueue` WHERE `emailQueueID` = '12207' LIMIT 1
```

The message that should be pasted into ossec-logtest is:

```
MySQL log: 130218 12:07:52 [Warning] Unsafe statement written to the binary log using␣
↪statement format since BINLOG_FORMAT = STATEMENT. The statement is unsafe because␣
↪it uses a LIMIT clause. This is unsafe because the set of rows included cannot be␣
↪predicted. Statement: DELETE FROM `emailQueue` WHERE `emailQueueID` = '12207' LIMIT␣
↪1
```

ossec-logtest example usage

Example 1: Testing standard rules

```
# echo "Aug 29 15:33:13 ns3 named[464]: client 217.148.39.3#1036: query (cache) denied
↪" | /var/ossec/bin/ossec-logtest -f
2010/08/10 06:57:06 ossec-testrule: INFO: Reading decoder file loadables/decoders/00_
↪decoders.xml.
2010/08/10 06:57:06 ossec-testrule: INFO: Reading decoder file loadables/decoders/50_
↪named.xml.
2010/08/10 06:57:06 ossec-testrule: INFO: Reading decoder file loadables/decoders/50_
↪pam.xml.
2010/08/10 06:57:06 ossec-testrule: INFO: Reading decoder file loadables/decoders/50_
↪sshd.xml.
2010/08/10 06:57:06 ossec-testrule: INFO: Reading loading the lists file: 'loadables/
↪lists/rfc1918-privateaddresses'
2010/08/10 06:57:06 ossec-testrule: INFO: Started (pid: 78828).
ossec-testrule: Type one log per line.



**Phase 1: Completed pre-decoding.
       full event: 'Aug 29 15:33:13 ns3 named[464]: client 217.148.39.3#1036: query␣
↪(cache) denied'
       hostname: 'ns3'
       program_name: 'named'
       log: 'client 217.148.39.3#1036: query (cache) denied'

**Phase 2: Completed decoding.
       decoder: 'named'
       srcip: '217.148.39.3'

**Rule debugging:
    Trying rule: 1 - Generic template for all syslog rules.
       *Rule 1 matched.
       *Trying child rules.
    Trying rule: 30100 - Apache messages grouped.
    Trying rule: 7200 - Grouping of the arpwatch rules.
    Trying rule: 6200 - Asterisk messages grouped.
    Trying rule: 9600 - cimserver messages grouped.
    Trying rule: 4700 - Grouping of Cisco IOS rules.
    Trying rule: 3900 - Grouping for the courier rules.
    Trying rule: 9700 - Dovecot Messages Grouped.
    Trying rule: 11100 - Grouping for the ftpd rules.
    Trying rule: 9300 - Grouping for the Horde imp 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: 6300 - Grouping for the MS-DHCP rules.
    Trying rule: 6350 - Grouping for the MS-DHCP rules.
    Trying rule: 11500 - Grouping for the Microsoft ftp rules.
    Trying rule: 50100 - MySQL messages grouped.
    Trying rule: 12100 - Grouping of the named rules
       *Rule 12100 matched.
       *Trying child rules.
    Trying rule: 12107 - DNS update using RFC2136 Dynamic protocol.
    Trying rule: 12101 - Invalid DNS packet. Possibility of attack.
    Trying rule: 12109 - Named fatal error. DNS service going down.
    Trying rule: 12102 - Failed attempt to perform a zone transfer.
```

(continues on next page)

(continued from previous page)

```
    Trying rule: 12103 - DNS update denied. Generally mis-configuration.
    Trying rule: 12104 - Log permission misconfiguration in Named.
    Trying rule: 12105 - Unexpected error while resolving domain.
    Trying rule: 12106 - DNS configuration error.
    Trying rule: 12108 - Query cache denied (maybe config error).
       *Rule 12108 matched.

**Phase 3: Completed filtering (rules).
       Rule id: '12108'
       Level: '4'
       Description: 'Query cache denied (maybe config error).'
       Info - Link: 'http://www.reedmedia.net/misc/dns/errors.html'
**Alert to be generated.
```

Example 2: Using OSSEC for the forensic analysis of log files

If you have one old log file that you want to check or if you are doing a forensics analysis of a box and wants to check the logs with OSSEC, we now have a solution too.

Let's say you have a file /var/log/secure that you want to analyze with OSSEC. You need to use the ossec-logtest tool with the "-a" flag to reproduce the alerts:

```
# cat /var/log/secure | /var/ossec/bin/ossec-logtest -a

** Alert 1264788284.11: - syslog,sshd,authentication_success,
2010 Jan 29 14:04:44 enigma->stdin
Rule: 5715 (level 3) -> 'SSHD authentication success.'
Src IP: a.b.2.15
User: dcid
Jan 15 10:25:01 enigma sshd[17594]: Accepted password for dcid from a.b.2.15 port␣
↪47526 ssh2

** Alert 1264788284.12: - syslog,sshd,authentication_success,
2010 Jan 29 14:04:44 enigma->stdin
Rule: 5715 (level 3) -> 'SSHD authentication success.'
Src IP: 127.0.0.1
User: dcid
Jan 15 11:19:20 enigma sshd[18853]: Accepted publickey for dcid from 127.0.0.1 port␣
↪6725 ssh2
```

You will get the alerts just like you would at /var/ossec/logs/alerts.log. The benefit now is that you can pipe this output to ossec-reported to get a better view of what is going on:

```
# cat /var/log/secure | /var/ossec/bin/ossec-logtest -a |/var/ossec/bin/ossec-reported
Report completed. ==
--------------------------------
->Processed alerts: 522
->Post-filtering alerts: 522

Top entries for 'Source ip':
--------------------------------
89.200.169.170 |41 |
127.0.0.1 |33 |
83.170.106.142 |20 |
204.232.206.109 |16 |
```

(continues on next page)

(continued from previous page)

```
..

Top entries for 'Username':
--------------------------------
root |247 |

Top entries for 'Level':
--------------------------------
Severity 5 |406 |
Severity 3 |41 |
Severity 10 |32 |

Top entries for 'Group':
--------------------------------
syslog |522 |
sshd |509 |
authentication_failed |369 |
invalid_login |146 |

Top entries for 'Rule':
--------------------------------
5716 - SSHD authentication failed. |223 |
5710 - Attempt to login using a non-existent.. |146 |
5715 - SSHD authentication success. |41 |
5702 - Reverse lookup error (bad ISP or atta.. |37 |
```

To get a report of all brute force attacks (for example) that scanned my box:

```
# cat /var/log/secure | /var/ossec/bin/ossec-logtest -a |/var/ossec/bin/ossec-
↪reported -f group authentication_failures

Report completed. ==
--------------------------------
->Processed alerts: 522
->Post-filtering alerts: 25

Top entries for 'Source ip':
--------------------------------
83.170.106.142 |2 |
89.200.169.170 |2 |
114.255.100.163 |1 |
117.135.138.183 |1 |
124.205.62.36 |1 |
173.45.108.230 |1 |
200.182.99.59 |1 |
202.63.160.50 |1 |
210.21.225.202 |1 |
211.151.64.220 |1 |
213.229.70.12 |1 |
218.30.19.48 |1 |
221.12.12.3 |1 |
59.3.239.114 |1 |
61.168.227.12 |1 |
61.233.42.47 |1 |
67.43.61.80 |1 |
72.52.75.228 |1 |
77.245.148.196 |1 |
```

(continues on next page)

(continued from previous page)

```
79.125.35.214 |1 |
85.21.83.170 |1 |
92.240.75.6 |1 |
94.198.49.185 |1 |

Top entries for 'Username':
--------------------------------
root |24 |

Top entries for 'Level':
--------------------------------
Severity 10 |25 |

Top entries for 'Group':
--------------------------------
authentication_failures |25 |
sshd |25 |
syslog |25 |

Top entries for 'Location':
--------------------------------
enigma->stdin |25 |

Top entries for 'Rule':
--------------------------------
5720 - Multiple SSHD authentication failures. |24 |
5712 - SSHD brute force trying to get access.. |1 |
```

ossec-maild

The `ossec-maild` daemon sends OSSEC alerts via email. `ossec-maild` is started by ossec-control. Configuration for ossec-maild is handled in the ossec.conf. (see *ossec.conf: Global options*)

ossec-maild argument options

-c `<config>`
Run `ossec-maild` using <config> as the configuration file.

Default: /var/ossec/etc/ossec.conf

-D `<dir>`
Chroot to <dir>.

Default: /var/ossec

-d
Execute ossec-maild in debug mode. This option can be used multiple times to increase the verbosity of the debug messages.

-f
Run `ossec-maild` in the foreground.

-g `<group>`
Run `ossec-maild` as <group>.

-h
    Display the help message.

-t
    Test configuration.

-u `<user>`
    Run `ossec-maild` as <user>.

    Default: ossecm

-V
    Version and license information.

ossec-makelists

The `ossec-makelists` utility to compile cdb databases. `ossec-makelists` will scan ossec.conf for database files, check the mtime, and recompile all out of date databases.

See *CDB List lookups from within Rules* for more information.

ossec-makelists argument options

-c `<config>`
    Run with configuration file of <config>.

    Default /var/ossec/etc/ossec.conf

-d
    Execute ossec-makelists in debug mode. This option can be used multiple times to increase the verbosity of the debug messages.

-F
    Force the rebuild of all configured databases.

-g `<group>`
    Run as <group>.

-h
    Display the help message.

-t
    Test the configuration.

-u `<user>`
    Run as <user>.

-V
    Display the version and license information.

ossec-makelists example usage

Example: Running ossec-makelists and an update is necessary

```
# /var/ossec/bin/ossec-makelists
 * File lists/blocked.txt.cdb need to be updated
```

Example: Running ossec-makelists when no update is necessary

```
# /var/ossec/bin/ossec-makelists
 * File lists/blocked.txt.cdb does not need to be compiled
```

ossec-monitord

The `ossec-monitord` daemon monitors agent connectivity and compress daily log files. `ossec-monitord` is configured in ossec.conf. (see *ossec.conf: Localfile options*)

ossec-monitord argument options

-c `<config>`
: Run `ossec-monitord` using <config> as the configuration file.

  Default: /var/ossec/etc/ossec.conf

-D `<dir>`
: Chroot to <dir>.

  Default: /var/ossec

-d
: Execute ossec-monitord in debug mode. This option can be used multiple times to increase the verbosity of the debug messages.

-f
: Run `ossec-monitord` in the foreground.

-g `<group>`
: Run `ossec-monitord` as <group>.

-h
: Display the help message.

-t
: Test configuration.

-u `<user>`
: Run `ossec-monitord` as <user>.

  Default: ossecm

-V
: Version and license information.

ossec-regex

`ossec-regex` is a simple program that will validate a regex expression.a The pattern should be enclosed in single quotes to help prevent any strange interactions with the shell.

The syntax for `ossec-regex` is simple: `/var/ossec/bin/ossec-regex '<pattern>'` It then reads strings from stdin and outputs matches to stdout. `+OSRegex_Execute` and `+OS_Regex` are printed if a match is successful.

Example 1: A simple digit match: ^^^^^^^^^^^^^~^^^^^^^^^^^^^^^^^^

```
# /var/ossec/bin/ossec-regex '^\d\d\d'
333
+OSRegex_Execute: 333
+OS_Regex        : 333
f44
222
+OSRegex_Execute: 222
+OS_Regex        : 222
```

ossec-remoted

`ossec-remoted` is the server side daemon that communicates with the agents. It can listen to port 1514/udp (for OSSEC communications) and/or 514 (for syslog). It runs as ossecr and is chrooted to `/var/ossec` by default. `ossec-remoted` is configured in the <remote> section of ossec.conf. (see *ossec.conf: Remote Options*)

ossec-remoted argument options

-c `<config>`
Run `ossec-remoted` using <config> as the configuration file.

Default: /var/ossec/etc/ossec.conf

-D `<dir>`
Chroot to <dir>.

Default: /var/ossec

-d
Execute ossec-remoted in debug mode. This can be used more than once to increase the verbosity of the debug messages.

-f
Run ossec-remoted in the foreground.

-g `<group>`
Run `ossec-remoted` as <group>.

-h
Display the help message.

-t
Test configuration.

-u `<user>`
Run `ossec-remoted` as <user>.

Default: ossecm

-V
Version and license information.

ossec-reportd

`ossec-reportd` is a program to create reports from OSSEC alerts. `ossec-reportd` 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. End the ossec-reportd with `2>&1` to redirect stderr to stdout. `more` or `less` can be easily used after the stderr redirect.

ossec-reportd argument options

-D `<dir>`
: chroot to `<dir>`.

-d
: Execute ossec-reportd in debug mode. This option can be used multiple times to increase the verbosity of the debug messages.

-f `<filter> <value>`
: Filter the results.

Note: Allowed filters: group, rule, level, location, user, srcip, and 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 OSSEC Version and license information.

ossec-reportd example usage

Example 1: Show Successful Logins

```
# cat /var/ossec/logs/alerts/alerts.log | /var/ossec/bin/ossec-reportd -f group␣
↪authentication_success
```

Example 2: Show Alerts Level 10 and Greater

```
# cat /var/ossec/logs/alerts/alerts.log | /var/ossec/bin/ossec-reportd -f level 10
```

Example 3: Show the srcip for all users

```
# cat /var/ossec/logs/alerts/alerts.log | /var/ossec/bin/ossec-reportd -f group␣
↪authentication -r user srcip
```

Example 4: Show Changed files as reported by Syscheck

```
# cat /var/ossec/logs/alerts/alerts.log | /var/ossec/bin/ossec-reportd -f group␣
↪syscheck -r location filename
```

Example output

```
# cat /var/ossec/logs/alerts/alerts.log | /var/ossec/bin/ossec-reportd 2>&1 | more
2011/07/11 21:01:36 ossec-reportd: INFO: Started (pid: 1444).
2011/07/11 21:01:41 ossec-reportd: INFO: Report completed. Creating output...

Report completed. ==
------------------------------------------------
->Processed alerts: 17
->Post-filtering alerts: 17
->First alert: 2011 Jul 11 00:00:46
->Last alert: 2011 Jul 11 00:16:52


Top entries for 'Username':
------------------------------------------------
_nrpe                                           |6        |
SYSTEM                                          |2        |


Top entries for 'Level':
------------------------------------------------
Severity 3                                      |13       |
Severity 2                                      |4        |


Top entries for 'Group':
------------------------------------------------
syslog                                          |10       |
sudo                                            |6        |
dropbearrecon                                   |4        |
ossec                                           |4        |
sshd                                            |4        |
authentication_success                          |2        |
windows                                         |2        |
clamd                                           |1        |
freshclam                                       |1        |
virus                                           |1        |


Top entries for 'Location':
------------------------------------------------
ix->/var/log/secure                             |4        |
ix->ossec-logcollector                          |3        |
(vistapc) 192.168.17.0->WinEvtLog               |2        |
buffalo1->/var/log/secure                       |2        |
buffalo2->/var/log/secure                       |2        |
(junction) 192.168.17.17->/var/log/secure       |1        |
(junction) 192.168.17.17->ossec-logcollector    |1        |
ix->/var/log/local6                             |1        |
```

(continues on next page)

(continued from previous page)

```
junction->/var/log/secure                            |1       |


Top entries for 'Rule':
------------------------------------------------
5402 - Successful sudo to ROOT executed          |6       |
51006 - Client exited before authentication.     |4       |
591 - Log file rotated.                          |4       |
18107 - Windows Logon Success.                   |2       |
52507 - ClamAV database update                   |1       |
```

ossec-syscheckd

The `ossec-syscheckd` daemon checks configured files for changes to the checksums, permissions or ownership. `ossec-syscheckd` is started by ossec-control. Configuration for ossec-syscheckd is handled in the ossec.conf.

See *Syscheck* for more detailed configuration information.

ossec-syscheckd argument options

-c `<config>`
Run `ossec-syscheckd` using <config> as the configuration file.

Default: /var/ossec/etc/ossec.conf

-d
Execute ossec-syscheckd in debug mode. This can be used more than once to increase the verbosity of the debug messages.

-f
Run `ossec-syscheckd` in the foreground.

-h
Display the help message.

-t
Test configuration.

-V
Version and license information.

rootcheck_control

The rootcheck_control tool allows you to manage the policy monitoring and system auditing database that is stored on the server (manager) side. You can list anomalies detected by the rootcheck functionality, categorized into resolved and outstanding issues. Moreover you can find out when ossec-rootcheck was run the last time.

rootcheck_control argument options

-h
Display the help message.

-l
List available agents.

-lc
List only currently connected agents.

-u <id>
Updates (clear) the database for the agent.

-u all
Updates (clear) the database for all agents.

-i <agent_id>
Prints database for the agent.

-r
Used with -i, prints all the resolved issues.

-q
Used with -i, prints all the outstanding issues.

-L
Used with -i, prints the last scan.

-s
Changes the output to CSV (comma delimited).

rootcheck_control example usage

Example 1: Getting a list of system auditing/policy monitoring events

To get a list of all auditing/policy monitoring events for a specific agent, you can run `rootcheck_control -i`. To retrieve the agent id you can use any of the following commands:

- *rootcheck_control -l*,
- *agent_control -l*
- *syscheck_control -l*
- *syscheck_update -l*
- *manage_agents -l*

```
# /var/ossec/bin/rootcheck_control -i 002

Policy and auditing events for agent 'ossecagent (002) - 192.168.1.86':

Resolved events:

2010 Jun 15 13:01:22 (first time detected: 2009 Dec 10 18:48:43)
System Audit: System Audit: CIS - Debian Linux 8.8 - GRUB Password not set. File: /
↪boot/grub/menu.lst. Reference: http://www.ossec.net/wiki/index.php/CIS_DebianLinux .


Outstanding events:

2010 Jun 17 17:34:37 (first time detected: 2009 Dec 10 18:48:43)
System Audit: System Audit: CIS - Testing against the CIS Debian Linux Benchmark v1.0.
↪ File: /etc/debian_version. Reference: http://www.ossec.net/wiki/index.php/CIS_
↪DebianLinux .
```

(continues on next page)

(continued from previous page)

```
2010 Jun 17 17:34:37 (first time detected: 2009 Dec 10 18:48:43)
System Audit: System Audit: CIS - Debian Linux 1.4 - Robust partition scheme - /tmp␣
↪is not on its own partition. File: /etc/fstab. Reference: http://www.ossec.net/wiki/
↪index.php/CIS_DebianLinux .

2010 Jun 17 17:34:37 (first time detected: 2009 Dec 10 18:48:43)
System Audit: System Audit: CIS - Debian Linux 2.3 - SSH Configuration - Root login␣
↪allowed. File: /etc/ssh/sshd_config. Reference: http://www.ossec.net/wiki/index.php/
↪CIS_DebianLinux .
```

As you can see the detected events are shown in two categories, resolved events and outstanding event. To only show resolved events, run `rootcheck_control -ri`. To only show outstanding events, run `rootcheck_control -qi`. To only show the results of the last scan and time of that scan, run `rootcheck_control -Li`.

To gain that kind of information for the OSSEC server, run `rootcheck_control -i 000`.

Example 2: Clearing the system auditing/policy database

To clear the system auditing/policy monitoring database for a certain agent run the following command:

```
# /var/ossec/bin/rootcheck_control -u 002

** Policy and auditing database updated.
```

To clear the database for all agents and the server run the following command:

```
# /var/ossec/bin/rootcheck_control -u all

** Policy and auditing database updated.
```

The next time rootcheck is run, the database will be populated again.

syscheck_control

syscheck_control provides an interface for managing and viewing the integrity checking database.

syscheck_control argument options

-h
Display the help message.

-l
List available agents.

-lc
List only currently connected agents.

-u `<agent_id>`
Updates (clear) the database for the agent.

-u `all`
Updates (clear) the database for all agents.

-i `<agent_id>`
Prints database for the agent.

-r `-i`
List modified registry entries for the agent (Windows only).

-f `<file>`
Used with -i. Prints information about a modified file.

-z
Used with -f, zeroes the auto-ignore counter.

-d
Used with -f, ignores that file.

-s
Changes the output to CSV (comma delimited).

syscheck_control example usage

Example 1: Getting a list of modified files for an agent

To retrieve information about files that were monitored by OSSEC and modified after OSSEC was deployed, run `syscheck_control -i`.

```
# /var/ossec/bin/syscheck_control -i 002

Integrity changes for agent 'ossec-agent (002) - 192.168.1.86':

Changes for 2009 Dec 21:
2009 Dec 21 13:52:40,0 - /etc/authorization
2009 Dec 21 13:52:42,0 - /etc/cups/printers.conf
2009 Dec 21 13:52:42,0 - /etc/cups/printers.conf.O
2009 Dec 21 13:52:58,0 - /etc/postfix/main.cf.default

Changes for 2010 Jan 04:
2010 Jan 04 10:13:58,0 - /etc/authorization

Changes for 2010 Jan 06:
2010 Jan 06 09:45:43,0 - /etc/postfix/main.cf.default

Changes for 2010 Jan 18:
2010 Jan 18 09:18:51,0 - /etc/cups/printers.conf
2010 Jan 18 09:18:51,0 - /etc/cups/printers.conf.O

Changes for 2010 Feb 23:
2010 Feb 23 09:17:22,2 - /etc/cups/printers.conf
2010 Feb 23 09:17:22,2 - /etc/cups/printers.conf.O

Changes for 2010 Mar 24:
2010 Mar 24 08:42:52,3 - /etc/cups/printers.conf
2010 Mar 24 08:42:52,3 - /etc/cups/printers.conf.O
```

As you can see this command provides an overview about file modifications.

Example 2: Getting more detailed information about a modified file

If you need to get more detailed information about a file that was modified you can use syscheck_control to view

- the time stamp when the file was added to the syscheck database
- the integrity checking values when the file was added to the syscheck database
- the time stamps when OSSEC detected a modification
- the integrity checking values for every time OSSEC detected a modification.

The integrity checking values include

- how often the file has changed
- file size
- file permissions
- owner and group id of the file
- MD5 and SHA1 hashes of the file.

To retrieve this information, run `syscheck_control -i`:

```
# /var/ossec/bin/syscheck_control -i 002 -f /etc/authorization

Integrity changes for agent 'ossec-agent (002) - 192.168.1.86':
Detailed information for entries matching: '/etc/authorization'

2009 Dec 21 13:52:40,0 - /etc/authorization
File added to the database.
Integrity checking values:
   Size: 27771
   Perm: rw-r--r--
   Uid:  0
   Gid:  0
   Md5:  dd62912576ae05d611d7469be809cf1d
   Sha1: 530df0283df52f0152b9e7ce1a518119b06ceebc

2010 Jan 04 10:13:58,0 - /etc/authorization
File changed. - 1st time modified.
Integrity checking values:
   Size: >28050
   Perm: rw-r--r--
   Uid:  0
   Gid:  0
   Md5:  >50da55def41bcede7d42ac5ee8fe12c9
   Sha1: >97f4b2b48a97321a3e245221e0ea4353cf4fa8ef
```

Example 3: Clearing the syscheck database

To clear the syscheck database for a certain agent run the following command:

```
# /var/ossec/bin/syscheck_control -u 002

** Integrity check database updated.
```

`syscheck_control -i 002` will now show that no modified files for that agent are in the database:

```
# /var/ossec/bin/syscheck_control -i 002

Integrity changes for agent 'ossec-agent (002) - 192.168.1.86':

** No entries found.
```

To clear the database for all agents and the server run the following command:

```
# /var/ossec/bin/syscheck_control -u all

** Integrity check database updated.
```

The next time syscheck is run, the database will be populated again.

syscheck_update

syscheck_update: Updates the integrity check database. This means that all information about files that were added to the integrity check database will be dismissed and leave an empty database which will be populated again the next time the syscheck daemon runs on agents or the server.

It does the same thing as `syscheck_control -u`` (cf. :ref:`syscheck_control`).

syscheck_update argument options

-h
    Display the help message.

-l
    List available agents.

-a
    Updates the database for all agents.

-u `<agent_id>`
    Updates the database for the agent.

-u `local`
    Updates the local database.

util.sh

The `util.sh` shell script can add a file to be monitored by `ossec-logcollector`. It can also add a full_command to check for changes to a website, or for changes to the name server of a domain.

A blogpost from Daniel Cid (for 3WoO) introduced this utility.

util.sh argument options

addfile `<filename> [<format>]`
    Add a file to be monitored by `ossec-logtest`. A `localfile` will be added to the ossec.conf.

addsite `<domain>`
    Monitor a website for changes. A `full_command` will be added to the `ossec.conf` using lynx to dump the initial page. A rule can be written to monitor this output for changes.

Note: Requires lynx.

Warning: This may not be useful on pages with dynamic content.

adddns <domain>
Monitor the name server of a domain for changes. A `full_command` will be added to the ossec.conf using host

Note: Requites the `host` command.

util.sh example usage

Example: Running util.sh

Running the following command:

```
# /var/ossec/bin/util.sh adddns ossec.net
```

will add the following to that system's `ossec.conf`:

```
<ossec_config>
   <localfile>
     <log_format>full_command</log_format>
     <command>host -W 5 -t NS ossec.net; host -W 5 -t A ossec.net | sort</command>
   </localfile>
 </ossec_config>
```

verify-agent-conf

`verify-agent-conf` verifies the OSSEC agent.conf configuration. It exits silently if the configuration is correct.

verify-agent-conf example usage

Example 1: Running verify-agent-conf on a working agent.conf

```
# /var/ossec/bin/verify-agent-conf
#
```

Example 2: Running verify-agent-conf on a non-working agent.conf

```
# /var/ossec/bin/verify-agent-conf
2011/07/12 21:22:07 ossec-config(1226): ERROR: Error reading XML file '/var/ossec/etc/
↪shared/agent.conf': XML ERR: Bad formed XML. Element not opened (line 13).
```

Examples

Contents:

Examples

Granular Email Examples

Example 1: Group alerts

If you want to e-mail *xx@y.z* for every event in the group syslog you can add the following to ossec

```
<email_alerts>
    <email_to>xx@y.z</email_to>
    <group>syslog</group>
</email_alerts>
```

Example 2: Message Format

To e-mail (in the SMS format) *aa@y.z* for every event with severity higher than 10

Note: Note that the SMS format is not grouped, so the e-mail is sent immediately).

```
<email_alerts>
    <email_to>aa@y.z</email_to>
    <level>10</level>
    <format>sms</format>
</email_alerts>
```

Example 3: Email based on Rule ID's

To e-mail *bb@y.z* for every event from rule 123 or rule 124 (without grouping):

```
<email_alerts>
    <email_to>bb@y.z</email_to>
    <rule_id>123, 124</rule_id>
    <do_not_delay />
    <do_not_group />
</email_alerts>
```

Example 4: Email based on severity and agent

To e-mail *cc@y.z* for every event with severity higher than 12, from agent qwert or agt1, without any delay (immediately):=====

```
<email_alerts>
    <email_to>cc@y.z</email_to>
    <level>12</level>
```

(continues on next page)

(continued from previous page)

```
    <event_location>qwerty|agt1</event_location>
    <do_not_delay />
</email_alerts>
```

Example 5: Multiple granular options together

You can have as many granular options as you want. In this example, we want the following:

- Email cc@y.z for every alert from agents qwerty and agt1
- Email john@y.z for every alert from agent secsys, lowsys and aixsys
- Email mike@y.z for every alert from /log/secure (from any agent)
- Email l@y.z for every alert from 192.168.0.0/24 network
- Email boss@y.z for every alert above level 10.

```
<ossec_config>
    <email_alerts>
        <email_to>cc@y.z</email_to>
        <event_location>qwerty|agt1</event_location>
    </email_alerts>

    <email_alerts>
        <email_to>john@y.z</email_to>
        <event_location>secsys|lowsys|aixsys</event_location>
    </email_alerts>

    <email_alerts>
        <email_to>mike@y.z</email_to>
        <event_location>/log/secure$</event_location>
    </email_alerts>

    <email_alerts>
        <email_to>l@y.z</email_to>
        <event_location>192.168.</event_location>
    </email_alerts>

    <email_alerts>
        <email_to>boss@y.z</email_to>
        <level>12</level>
    </email_alerts>
</ossec_config>
```

Report Output Examples

Receive a summary of all authentication success alerts

The following example will send a daily report of all authentication_success alerts, sorted by the related field srcip.

```
<ossec_config>
    <reports>
        <category>authentication_success</category>
```

(continues on next page)

(continued from previous page)

```
        <user type="relation">srcip</user>
        <title>Daily report: Successful logins</title>
        <email_to>me@example.com</email_to>
```

Receive summary of all File integrity monitoring alerts

The following example will send a report of all events related to syscheck.

```
<ossec_config>
    <reports>
        <category>syscheck</category>
        <title>Daily report: File changes</title>
        <email_to>me@example.com</email_to>
```

Syslog Output Examples

Send all alerts to 10.10.10.125:

```
<syslog_output>
  <server>10.10.10.125</server>
</syslog_output>
```

Send all alerts to 10.10.10.126 in CEF:

```
<syslog_output>
  <server>10.10.10.126</server>
  <format>cef</format>
</syslog_output>
```

Send all alerts level 6 and above to 10.10.10.127 on port 515:

```
<syslog_output>
  <server>10.10.10.127</server>
  <port>515</port>
  <level>6</level>
</syslog_output>
```

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Index