WAF-FLE: *Deployment Guide*

Version 0.6.3

March, 2014

WAF-FLE Deployment Guide

Index

Introduction

This deployment guide will guide you in WAF-FLE installation and initial settings, the installation is really simple, but different needs can be achieved with different scenarios, as showed bellow. A detailed installation process is provided, and for basic and specific setup, you can use specific How-To for the most used operating system currently supported.

Deployment scenarios

WAF-FLE as a console for ModSecurity can be deployed in many ways, according to your needs, like: lab/small environment, large environment, huge volume of events, Etc.

In this guide, we have 3 scenarios, but you are not limited to they:

- Standalone/Same host as ModSecurity
- Distributed, WAF-FLE and database in same host
- Distributed, dedicated WAF-FLE, separated from database

1. Standalone or with ModSecurity on same host

For small or lab deployments, you can host WAF-FLE and ModSecurity in the same host, this need some additional and important cares, to not make ModSecurity to block events sent to WAF-FLE. This is specially important to avoid event amplification (ModSecurity send an event to WAF-FLE, and it blocks the event sent, and so on).
Bellow a simple diagram showing what you need and how this deployment is expected.

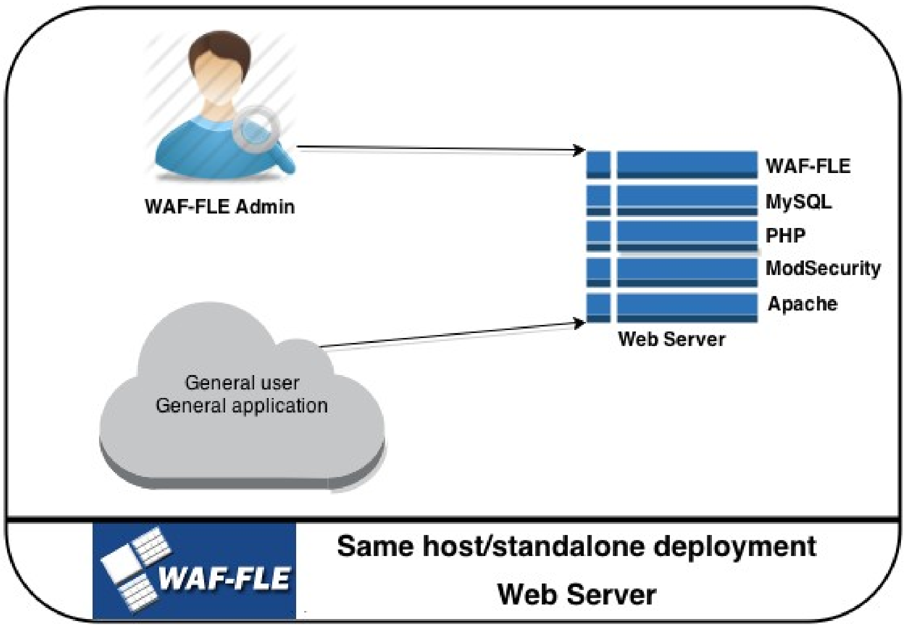


Figure 1: Standalone/Same host as ModSecurity

2. **Distributed, WAF-FLE and database in same host**

In an environment with many sensors, with a higher events volume, you can prefer or need to use a distributed deployment to consolidate events from all your ModSecurity hosts. This will require a dedicated box to WAF-FLE, having your events in a central point, allowing you to watch all events in a consolidated way.

This scenario will have better performance when compared to standalone, because in high event volume some event insertion or queries can be CPU and/or I/O bound, and is better to avoid share this resource with your application/web server.

You can even have events in a remote site (over a WAN or Internet), sent in real-time or scheduled over time (you can see more about this in *Log Feeder Configuration* section).

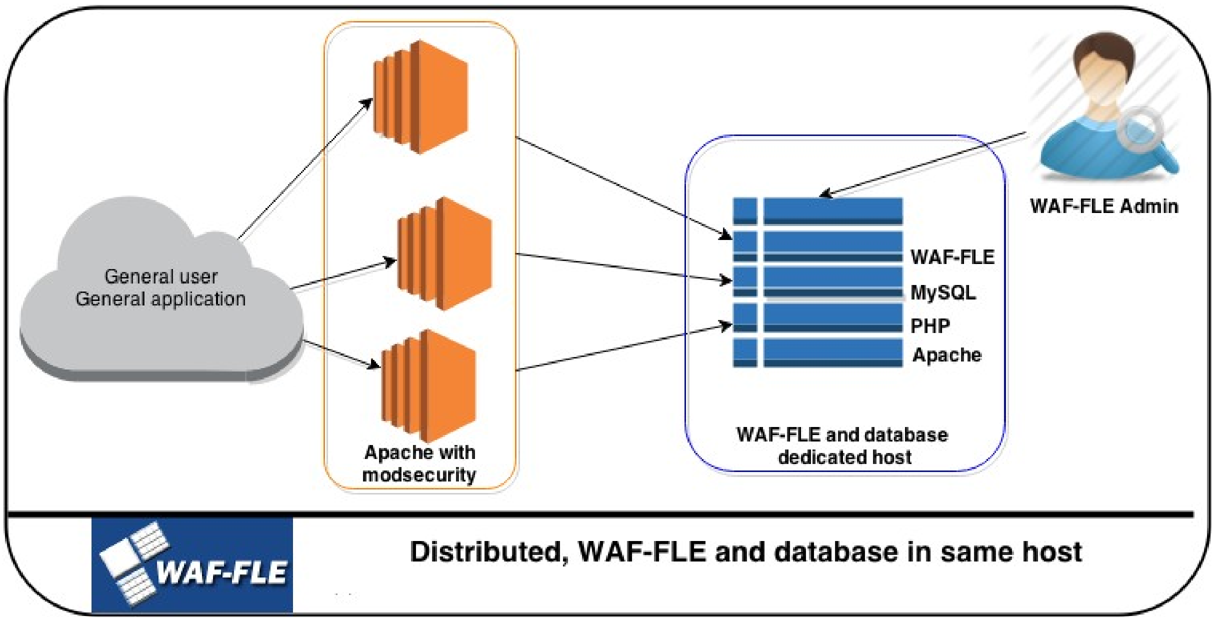
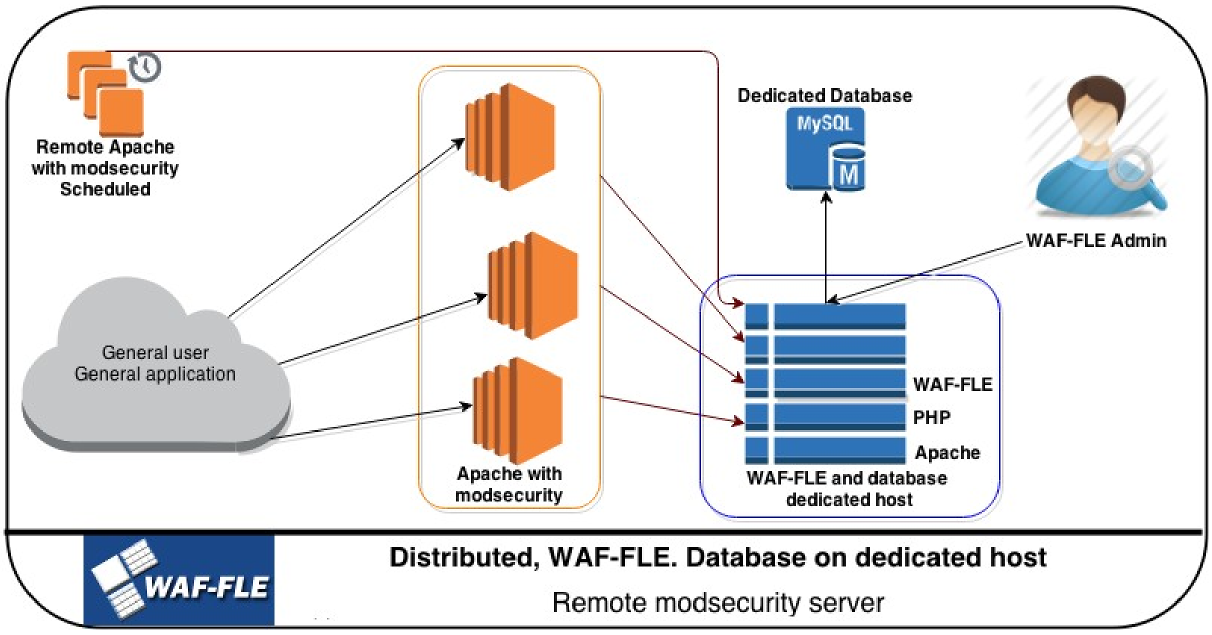


Figure 2: Distributed, WAF-FLE and database on same host

3. **Distributed, dedicated WAF-FLE, separated from database**

In a higher volume of events, you can need/prefer to keep the database in a separated and dedicated host, but for good performance be sure to keep the WAF-FLE and database server closer (typically a LAN latency will be OK).
If you need, you can have two WAF-FLE servers, one dedicated to receive events, other as a console for viewing the events.



Comparing deployment types

	#1 - Standalone	#2 - Distributed	#3 - Distributed, dedicated database
Simplicity	++	+	-
Performance	-	+	++
Scalability	-	+	++

TIP: If you are unsure about what is better, use scenario #2, that is a good balance of performance and resources.

Note: Regardless of your choice, keep security in mind. You can:

- Use SSL to transport events to WAF-FLE encrypted, as well to access and view events (optional);
- All event feed and console access is controlled by authentication (default);
- Event feed can be also restricted to one IP address, or network block (optional).

Requirements

To get the WAF-FLE working you need some components typically present in common *nix distributions. However, some times, you need to provide it from a third party package, or from the source. In specific how-to you'll find a step-by-step to main distributions used today.

- **Required**
  - Apache 2.x server
    - Apache mod-rewrite
  - PHP 5.3 or higher
    - PHP PDO Mysql extension
    - PHP GeoIP extension
  - MySQL 5.1 or later
- **Optional**
  - APC (alternative php cache), you can install, and can enable/disable in config.php
    TIP: You should consider keep APC enabled to improve WAF-FLE performance.

General Installation

WAF-FLE installation is getting simpler with time, in past you need to create yourself the database and check if all required elements was there. Starting with version 0.6.0-rc was introduced the setup script that will guide you in requirements, database creation and database permissions setup.

Before you start to install WAF-FLE, you need to install all required components, so use the typical commands of your system to do this. You can check a step-by-step for some OS's and distributions in Specific **How-To.**

Requirements for installation

1. Install Apache
   1.1. Enable mod-rewrite
2. Install MySQL (if you will use the database in the same server)
3. Install PHP
   3.1. Install php-pdo
   3.2. Install php-mysql
   3.3. Install php-apc
   3.4. Install php-geoip

<u>NOTE</u>: PHP-GeoIP module need some '*hack*' to be useful to WAF-FLE, once that module don't have a function to check Autonomous System Number (ASN). To solve this you need to follow the below steps (what include download MaxMind GeoIP Database), follow:

```
mkdir /usr/share/GeoIP/
cd /usr/share/GeoIP/

http://geolite.maxmind.com/download/geoip/database/GeoLiteCountry/GeoIP.dat.gz
http://geolite.maxmind.com/download/geoip/database/GeoLiteCity.dat.gz
http://geolite.maxmind.com/download/geoip/database/asnum/GeoIPASNum.dat.gz

gzip -d GeoIP.dat.gz
gzip -d GeoLiteCity.dat.gz
gzip -d GeoIPASNum.dat.gz
mv GeoLiteCity.dat GeoIPCity.dat
# To make php GeoIP extension work with ASNum database
cp GeoIPASNum.dat GeoIPISP.dat
```

WAF-FLE installation

1. Download last WAF-FLE tarball from http://waf-fle.org/download

2. Extract the WAF-FLE tarball in a directory like "/usr/local/", outside Apache web root (what will create a /usr/local/waf-fle directory). You can use other directory, but in this case you need to change Apache configuration to point to this new directory;

```
   cd /usr/local
   tar -zxvf /tmp/waf-fle-0.6.3.tar.gz
```

3. Change directory to /usr/local/waf-fle;

4. Copy the "extra/waf-fle.conf" (the WAF-FLE Apache configuration) to Apache configuration directory (normally /etc/httpd/conf.d or /etc/apache2/conf.d), and edit it to reflect your environment;

```
   cp extra/waf-fle.conf /etc/apache2/conf.d/
```

4.1. If you used a different directory for WAF-FLE, you'll need to edit *waf-fle.conf* (the Apache config), looking for alias and Directory directives, changing it to reflect the WAF-FLE location, as showed bellow;

```
alias /controller/ /usr/local/waf-fle/controller/
...
<Directory /usr/local/waf-fle/controller/>
...
alias /waf-fle /usr/local/waf-fle/dashboard/
...
<Directory /usr/local/waf-fle/dashboard/>
...
```

4.2. Check directory permissions for WAF-FLE in Apache. In some Apache installations (like in FreeBSD) the default configuration need to be changed to give permissions to WAF-FLE, in Apache 2.0 or 2.2 you should use "*Allow from all*", in Apache 2.4 and later use "*Require all granted*", this is explained in standard waf-fle.conf file, as showed bellow:

```
<Directory /usr/local/waf-fle/controller/>
...
   # On some installation, like FreeBSD you need to adjust the
   # 'Allow from' directive bellow
   # For Apache 2.0/2.2 use "Allow", uncomment the line below
   # Order allow,deny
   # Allow from all

   # For Apache 2.4 and later "Require", uncomment the line below
   # Require all granted

   AddType application/x-httpd-php .php
</Directory>

...
<Directory /usr/local/waf-fle/dashboard/>
...
   # On some installation, like FreeBSD you need to adjust the
   #'Allow from' directive bellow
   # For Apache 2.0/2.2 use "Allow", uncomment the line below
   # Order allow,deny
   # Allow from all

   # For Apache 2.4 and later "Require", uncomment the line below
   # Require all granted

   AddType application/x-httpd-php .php
</Directory>
```

4.3. If you prefer or has a dedicated server to WAF-FLE, you can make Apache redirect "/" to "/waf-fle/", as showed below. In *waf-fle.conf* this is commented by default:

```
# ATTENTION: If you deploy WAF-FLE as a dedicated virtual host/server
# you can uncomment the lines bellow to get a more simple access to
# web interface. You need mod_alias enabled to use this way.
#
<Location />
   RedirectMatch ^/$ /waf-fle/
</Location>
...
```

4.4. To add more security to your installation, consider enable SSL on your server, to make access encrypted for both event feed and for console access;

4.5. Reload Apache configuration.

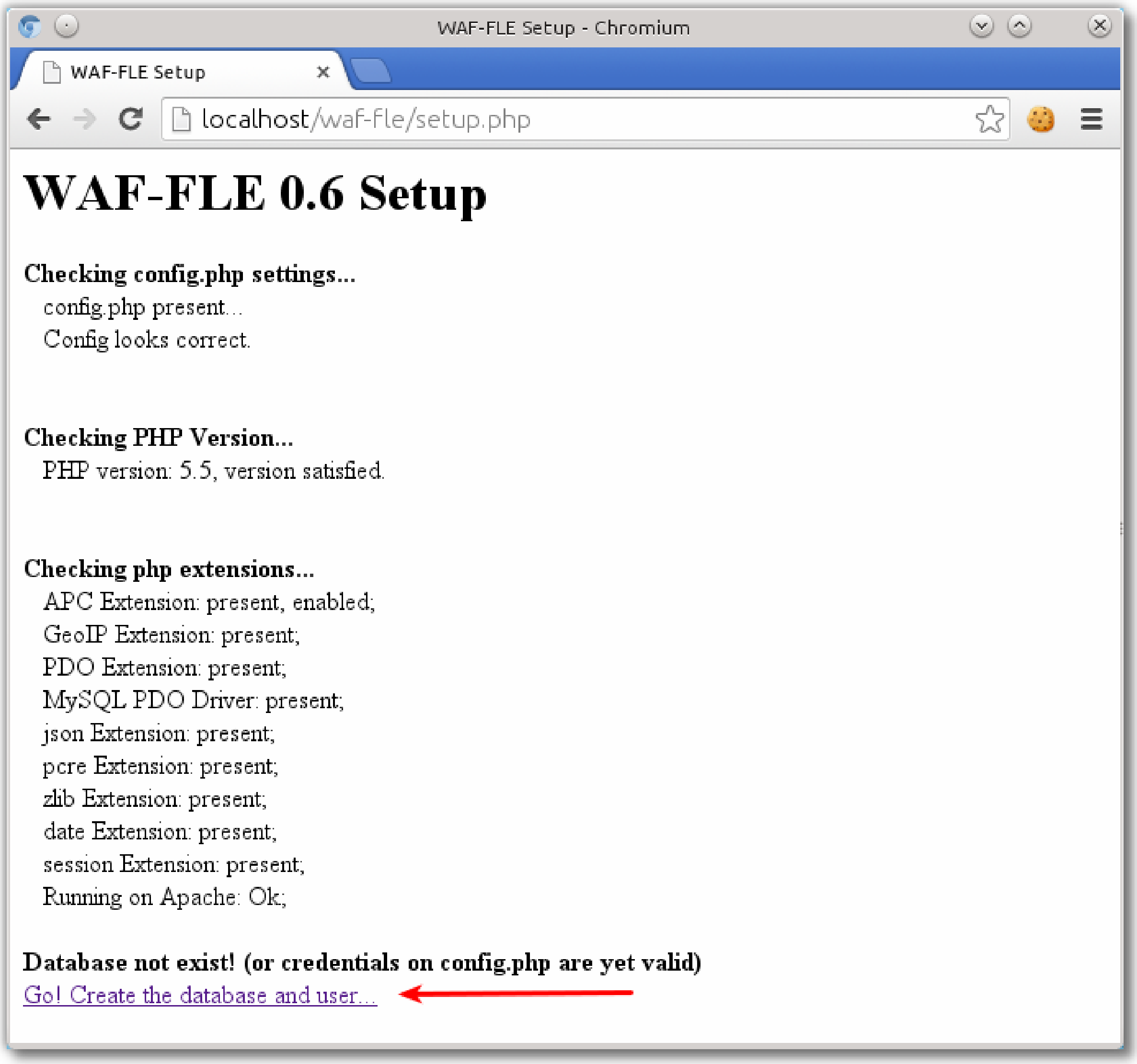
5. In WAF-FLE directory create your configuration file by copying config.php.example to config.php:

```
cp config.php.example config.php
```

6. Edit config.php file to define your database server, username, password and database name. The database and user permissions will be created by setup script. During setup, keep the $SETUP directive *true*.

```
$DB_HOST   = "localhost";
$DB_USER   = "waffle_user";
$DB_PASS   = "<FILL_User_Password>";
$DATABASE = "waffle";
...
$SETUP = true;
```

7. Point your browser to http://<your server>/waf-fle/ to start the setup script



WAF-FLE 0.6 Setup

Checking config.php settings...
config.php present...
Config looks correct.

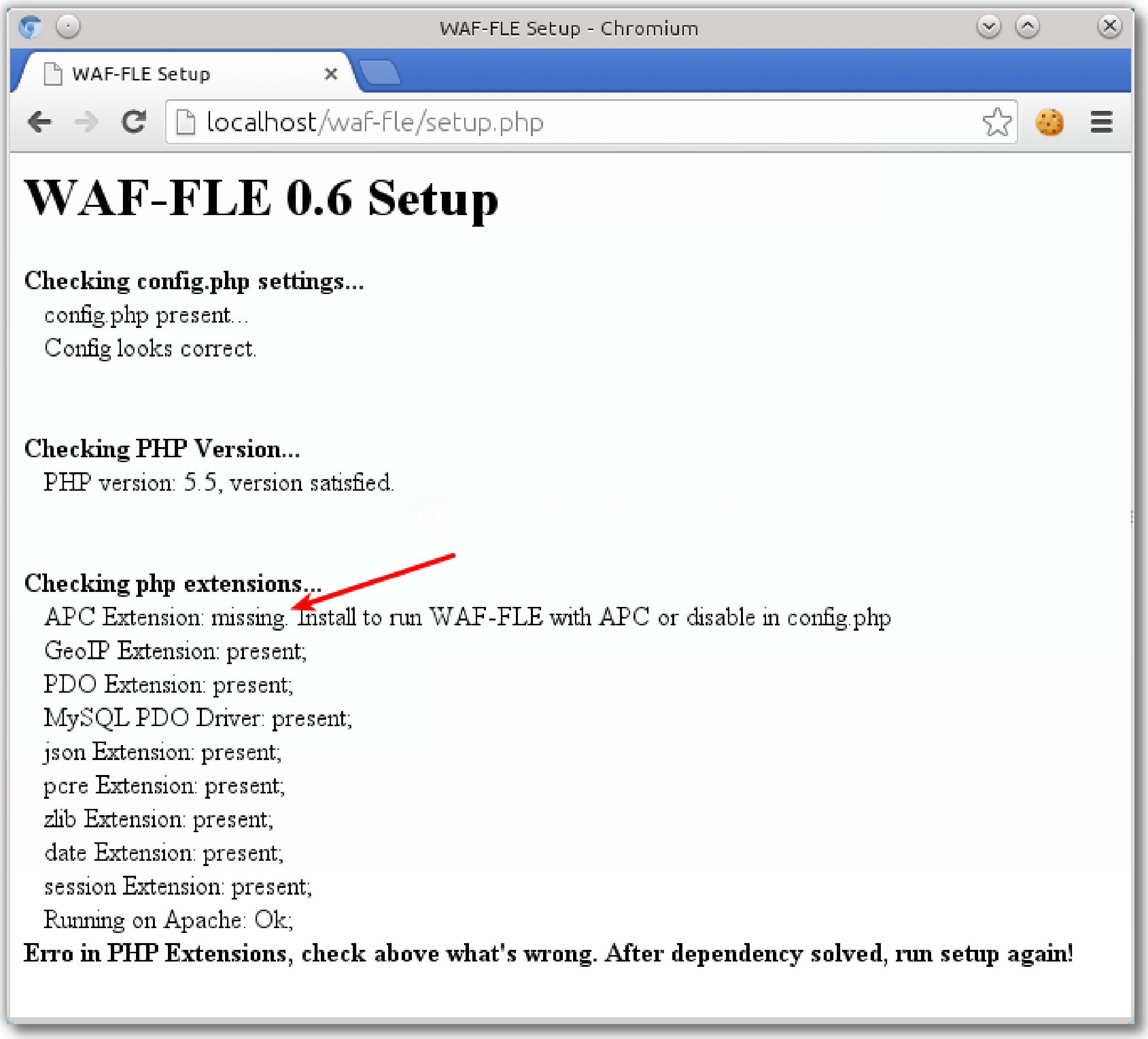
Checking PHP Version...
PHP version: 5.5, version satisfied.

Checking php extensions...
APC Extension: present, enabled;
GeoIP Extension: present;
PDO Extension: present;
MySQL PDO Driver: present;
json Extension: present;
pcre Extension: present;
zlib Extension: present;
date Extension: present;
session Extension: present;
Running on Apache: Ok;

Database not exist! (or credentials on config.php are yet valid)
Go! Create the database and user...

8. The setup script will first check if your server have all the required components installed and properly configured. The setup script will check if the database already exist (to avoid overwrite it), as showed below.

8.1. If some required components are missing you can get an error, as shown below:



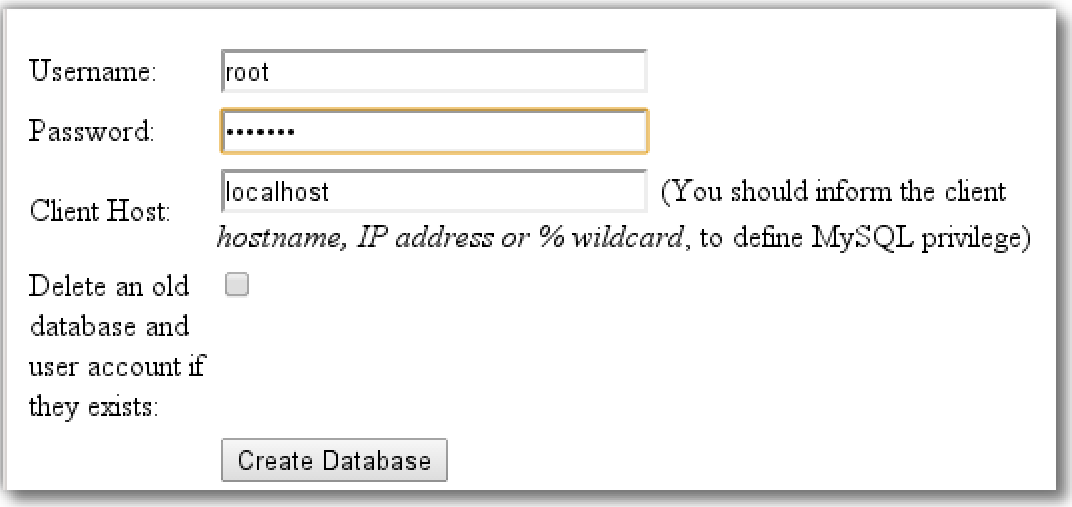
WAF-FLE 0.6 Setup

Checking config.php settings...
config.php present...
Config looks correct.

Checking PHP Version...
PHP version: 5.5, version satisfied.

Checking php extensions...
APC Extension: missing. Install to run WAF-FLE with APC or disable in config.php
GeoIP Extension: present;
PDO Extension: present;
MySQL PDO Driver: present;
json Extension: present;
pcre Extension: present;
zlib Extension: present;
date Extension: present;
session Extension: present;
Running on Apache: Ok;
Erro in PHP Extensions, check above what's wrong. After dependency solved, run setup again!

9. Go to next step by clicking in “*Go! Create the database and user...*”, and inform the credentials to access your MySQL database as administrator (Username, Password), the database hostname (that can be localhost for MySQL and WAF-FLE in same host, or other hostname for a MySQL and WAF-FLE in different hosts). Then click on “*Create Database*”



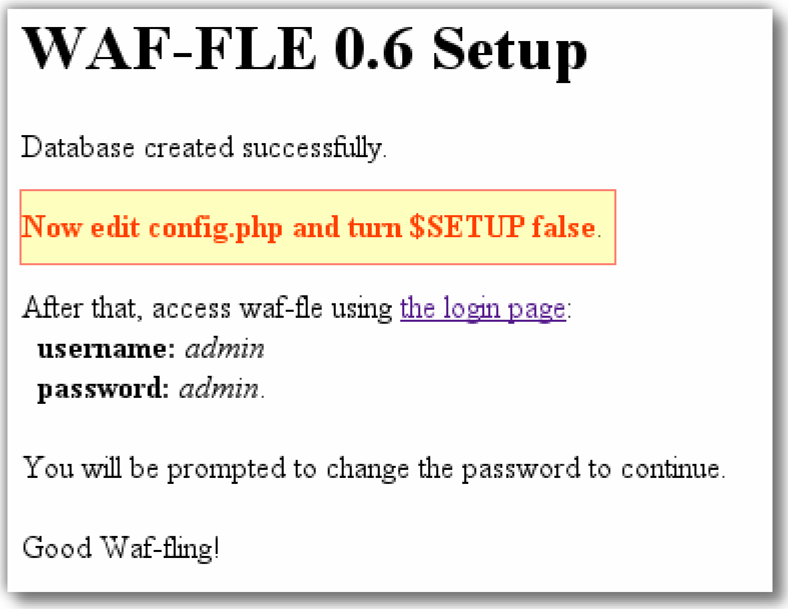
Username:	root
Password:	•••••••
Client Host:	localhost (You should inform the client *hostname, IP address or % wildcard*, to define MySQL privilege)
Delete an old database and user account if they exists:	☐

Create Database

10. If everything go well, you should get the message below, showing that the database was successfully created. Pay attention to emphasis "*Now edit config.php and turn $SETUP false*"

```
$SETUP = false;
```

10.1. After change $SETUP, you can click on "login page" to access WAF-FLE login.



WAF-FLE 0.6 Setup

Database created successfully.

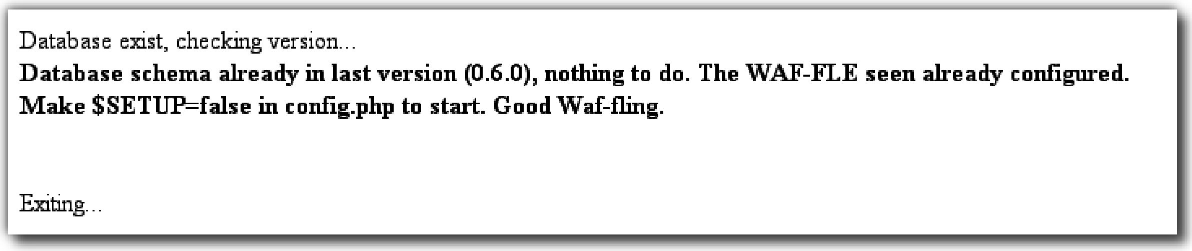
Now edit config.php and turn $SETUP false.

After that, access waf-fle using the login page:
username: *admin*
password: *admin*.

You will be prompted to change the password to continue.

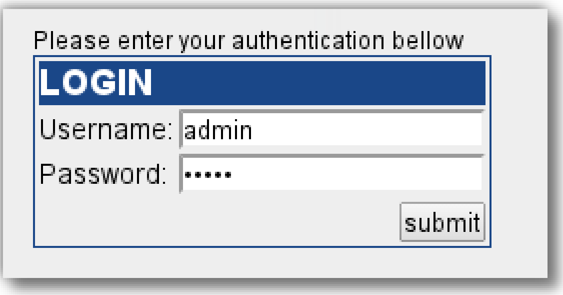
Good Waf-fling!

10.2. If you don't change $SETUP to false in config.php, WAF-FLE will redirect you to setup script again, and will show the error message below:



```
Database exist, checking version...
Database schema already in last version (0.6.0), nothing to do. The WAF-FLE seen already configured.
Make $SETUP=false in config.php to start. Good Waf-fling.


Exiting...
```

11. Login in WAF-FLE. (**Username** admin, **Password**: admin)

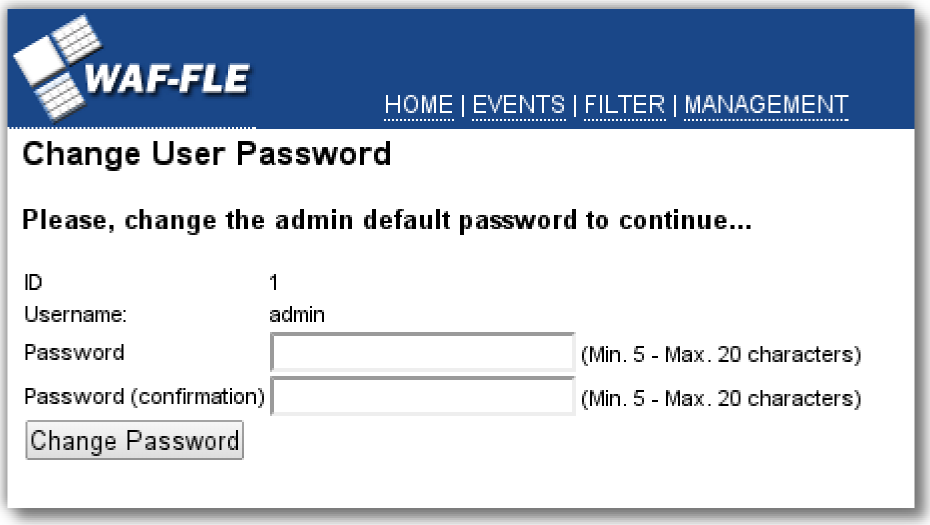


Please enter your authentication bellow

LOGIN

Username:	admin
Password:	•••••

12. You will be forced to change admin password, choose a strong password.



Change User Password

Please, change the admin default password to continue...

ID	1	
Username:	admin	
Password		(Min. 5 - Max. 20 characters)
Password (confirmation)		(Min. 5 - Max. 20 characters)

13. Go to *Management* menu, to setup your first sensor.

14. **Note to CRS Users**: If you run WAF-FLE in a machine with ModSecurity, using Core Rule Set (CRS), you can experience problems with some rules that run in phase 1. One example is rule "allowed method" (id 960032), other rules that inspect in phase 1 can cause problems too.
The exception made in waf-fle.conf don't work with these rules. This way, you

are advised to create and use a bypass rule, put it in proper order in your CRS structure.

For example, you can create a "modsecurity_crs_11_waffle.conf" file with rule below:

```
SecRule REQUEST_FILENAME '^/controller/$' \
"phase:1,msg:'Match',id:99999,nolog,noauditlog, \
allow,ctl:RuleEngine=On"
```

Don't worry, this rule turn ModSecurity engine On only to controller path (to avoid unwanted log in DetectionOnly mode), and keep other rules with your intentioned Engine status.

WAF-FLE upgrade

The upgrade process allow you to start to use a new version, enjoy new features and bug fixes. The process can be very simple or sometimes it can be more complex and slow. Pay attention to notes below in each version upgrade, this is explained too in README file of each release.

Attention: every time that a new version change database schema, your database file system must have more that 50% of free space to proceed with database modification. If you don't have this space, delete old events first, otherwise you will not be able to upgrade database schema, and the WAF-FLE version.

From version	**To version**	**Database schema changes**	**Notes and upgrade procedure**
0.6.0	0.6.3	No	**WAF-FLE**: Copy new files over old files. **Mlog2Waffle**: Copy the new files over the old files. **Process**: Very simple/fast upgrade
0.6.0-rcX	0.6.0	No	**WAF-FLE:** Just copy new files over old files. **Process**: Very simple/fast upgrade
0.5x	0.6.0-rc1	Yes	**WAF-FLE:** Copy new files over old files. **Mlog2Waffle**: first release of mlog2waffle. **Process:** More slow, because of many changes on database schema. To upgrade the schema and migrate the data you must edit the config.php and define all needed parameters, like database server, username and password, and turn the variable $SETUP to "*true*" in config.php. After that, you should point your browser to http(s)://webserver/waf-fle/setup.php, following the steps of setup script. If you already have lots of events in database, the migration can take a long time, be patient. The upgrade script will not compress old data, and will not process GeoIP data from source IP.

0.5	0.5.1	No	**WAF-FLE:** Just copy new files over old files. **Process**: Very simple/fast upgrade

Sensor Setup

While the WAF-FLE installation is a one time process, the sensor setup can be done many times, once for each new sensor. Follow the process below to each sensor on your network.
Note: you can use a sensor defined in WAF-FLE to aggregate a cluster of servers with ModSecurity, as if were one sensor, but with many servers, is up to you use one sensor for each server or one sensor for all servers of a cluster.
Note: Without define sensors in WAF-FLE you will not be able to receive any event, so this is a very important step.

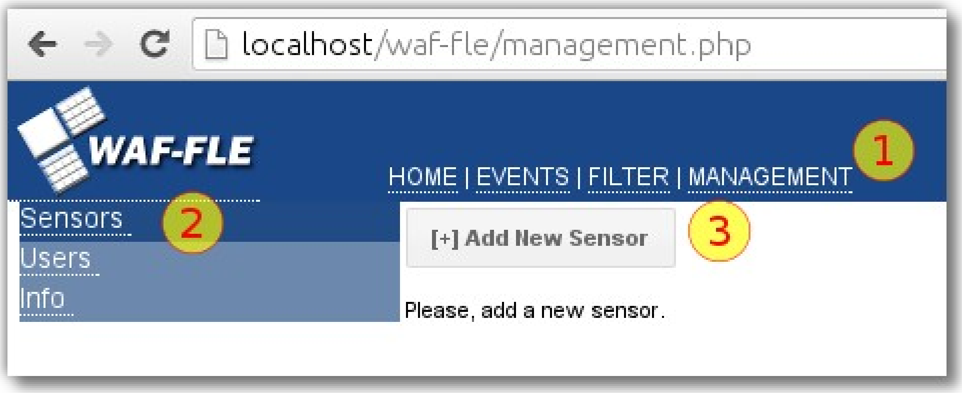
Sensor configuration is a two side setup, both parts will be discussed here.:

1. Sensor definition in WAF-FLE,
2. Events feeder, done in each box running ModSecurity.

Sensor Definition

To create a new sensor, follow the steps below:

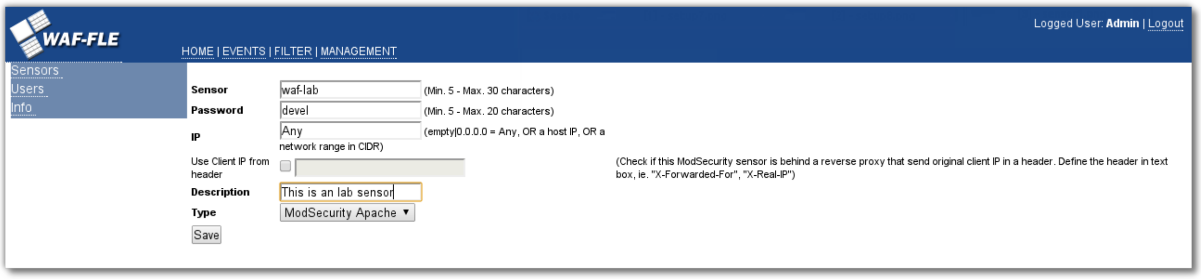
1. Go to *Management* menu
2. Click in *Sensors*
3. Click in Button "*Add New Sensor*"


localhost/waf-fle/management.php
WAF-FLE
HOME | EVENTS | FILTER | MANAGEMENT
1
Sensors
Users
Info
2
[+] Add New Sensor
3

Please, add a new sensor.

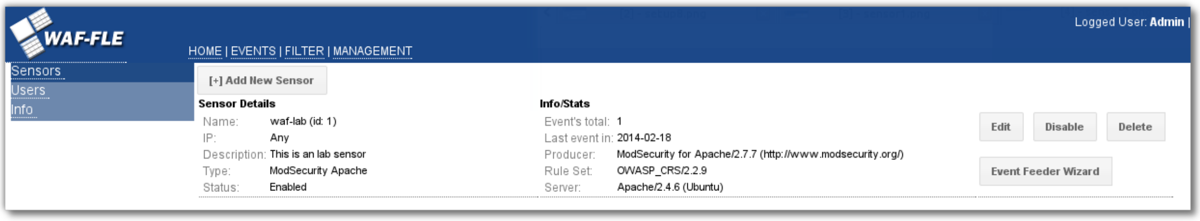
4. Fill the fields to identify this new sensor
    a. **Sensor**: is the sensor name, this is a required field, and is showed in events list and dashboard as the originator sensor for the event. It is also used as username for sensor authentication in event feed.
    Is required that sensor has a minimum of 5 characters and a maximum of 20.
    b. **Password**: is used for authentication in event feed. Is a required field; Is required that password has a minimum of 5 characters and a maximum of 20.

   c. **IP**: You must inform an IP address of the sensor, that can be:
      1. Any/0.0.0.0/Empty to accept events from any source address;
      2. IP address, to accept events only from one IP source address;
      3. Network, specified in CIDR notation (example, 192.168.1.0/24), where the sensor can be any address in that network;
   d. **Use Client IP from header**: Sometimes you need to use ModSecurity behind a Reverse Proxy (like Varnish, Nginx Etc), in this case can be useful (or even mandatory) to inform what HTTP Header used by reverse proxy to record remote client IP address. The typical is “X-Forwarded-For”, but you can use any other you need, just inform it in the field. This is an optional field.
   e. **Description**: An optional sensor description.
   f. **Type**: Which type of sensor is this. Currently only “ModSecurity Apache” is provided;
5. Click in save after fill all necessary fields;


WAF-FLE
Logged User: Admin | Logout
HOME | EVENTS | FILTER | MANAGEMENT
Sensors
Users
Info

Sensor	waf-lab	(Min. 5 - Max. 30 characters)
Password	devel	(Min. 5 - Max. 20 characters)
IP	Any	(empty|0.0.0.0 = Any, OR a host IP, OR a network range in CIDR)
Use Client IP from header	☐	(Check if this ModSecurity sensor is behind a reverse proxy that send original client IP in a header. Define the header in text box, ie. "X-Forwarded-For", "X-Real-IP")
Description	This is an lab sensor	
Type	ModSecurity Apache ▾	

6. After receive some event, the sensors have some useful information about events generated by this sensor, depicted below:
   a. **Event's total**: How many events this sensor has sent to (and still in) database;
   b. **Last event in**: When last event arrive;
   c. **Producer**: Which version of ModSecurity has sent last event;
   d. **Rule Set**: Which rule set generated the last event;
   e. **Server**: Which server is running on sensor
   f. **Status**: Inform if the sensor is enabled or disabled in WAF-FLE



Sensor Details		Info/Stats	
Name:	waf-lab (id: 1)	Event's total:	1
IP:	Any	Last event in:	2014-02-18
Description:	This is an lab sensor	Producer:	ModSecurity for Apache/2.7.7 (http://www.modsecurity.org/)
Type:	ModSecurity Apache	Rule Set:	OWASP_CRS/2.2.9
Status:	Enabled	Server:	Apache/2.4.6 (Ubuntu)

7. You can do additional operations in the interface:
   a. **Edit**: This allow you to change the sensor parameters, all parameters can be changed;
   b. **Disable**: This prevents new events to be accepted in WAF-FLE, preserving all events already in database. This is useful when you need to make some

maintenance in database;

c. **Delete**: This delete the sensor AND ALL ITS EVENTS;
d. **Event Feeder Wizard**: Create templates useful to configure sensors and his event's feeder in ModSecurity boxes. This will be detailed below.

Event Feeder Configuration

Event's feeder is located in sensor side, is an agent that will feed (push) ModSecurity's events (logs) to WAF-FLE. Currently we have two ways to accomplish this, first is by using *mlog2waffle* and the second *mlogc*. Each feeder has two ways to work, as noted below.

Mlog2waffle

Mlog2waffle is a multi-thread WAF-FLE component, written in Perl, to feed events from ModSecurity to WAF-FLE. It is a replacement for Mlogc. Mlog2waffle read event index file generated by the ModSecurity and send the events to WAF-FLE, in real time (using the "tail" mode) or periodically (in batch mode). It is not piped with ModSecurity logs, what avoid log feeder to disturb web server.

- **Features**:
  - Run in real time, following the "tail" of ModSecurity index log;
  - Run in scheduled way in crontab;
  - Support to send events with HTTPS (SSL/TLS);
  - Multi-thread support to speed boost;
  - HTTP Keep-alive to save resource and speed boost.
- **Requirements**:
  - Perl
  - libwww (6.0 or more recent for accept self-signed certificate)
  - File::Pid
  - File::Tail
  - LWP::UserAgent
- **Modes of operation:**
  - *Service daemon or tail mode*: means that ModSecurity log file will be written to disk, and mlog2waffle will read the log file, as soon as it is generated, processing all entries. Audit log is stored on disk, until each entry has been processed and sent to the WAF-FLE. This make the logs be sent in real time.

  - *Scheduled in crontab or batch mode*: means that the ModSecurity log file will be written to disk, and a scheduled task on crontab will run the mlog2waffle that will read and process the log file. Audit log is stored on disk until the mlog2waffle process each entry, and send it to WAF-FLE. This make logs be sent periodically (depending upon the frequency of crontab entry), but not immediately. Typically, the logs are sent each 5 minutes (once an hour, once a day. You choose).

Mlogc

Mlogc is multi-thread component of ModSecurity, written in C, to feed events from ModSecurity to consoles like WAF-FLE or others, it is the original tool to feed logs to a console. Mlogc read events generated by ModSecurity and send events to WAF-FLE or other console, in real time (using a "piped" mode) or periodically (in batch mode). The piped mode can disturb web server in case of mlogc bad behave, as many times mentioned in ModSecurity mailing-list and in issues opened in bug-tracking.

- **<u>Features</u>**:
  - Run in real time, piped with the ModSecurity log;
  - Run in scheduled way in crontab (using a Perl script to help);
  - Support send events with HTTPS (SSL/TLS);
  - Multi-thread support to speed boost;
- **<u>Requirements</u>**:
  - Same requirements to build ModSecurity, plus
  - libcurl
- **<u>Modes of operation:</u>**
  - *Piped mode*: means that the ModSecurity index log file will feed Mlogc directly and will not be written to disk. Audit log keep stored on disk, until the program process each entry and send to WAF-FLE. This make logs be sent as soon as it is generated, in real time.

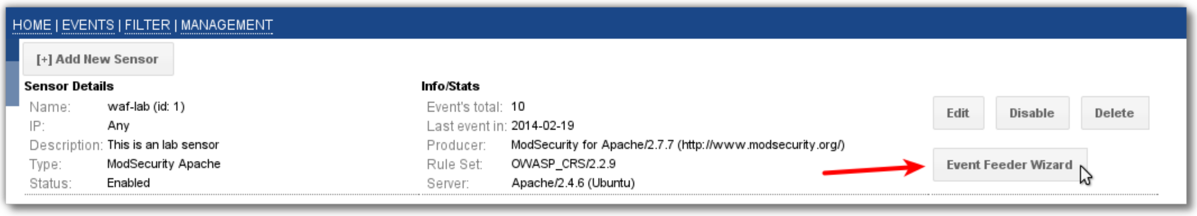
  - *Scheduled in crontab or batch mode*: means that the ModSecurity log file will be written to disk, and a scheduled task on crontab will read and process the log file. Audit log is stored on disk until the task process each entry, and send they to WAF-FLE. This make logs be sent periodically (depending upon the frequency of crontab entry), but not immediately. Typically, the logs are sent each 5 minutes.

Event Feeder Wizard

The Event Feeder Wizard will help you to configure your sensor, using mlog2waffle or mlogc, as service, batch or piped mode, providing all configuration files and changes needed to configure the feeder that you choose.
The wizard provide templates to configure both modsecurity.conf Log Directives, and mlog2waffle/mlogc configurations, as well directory creation and permissions.

Note: All files and commands shown in Wizard must be configured/executed on each sensor box that you install.

1. To access Event Feeder Wizard, go to Management/Sensors and click in Event Feeder Wizard button of the sensor that you want to configure.



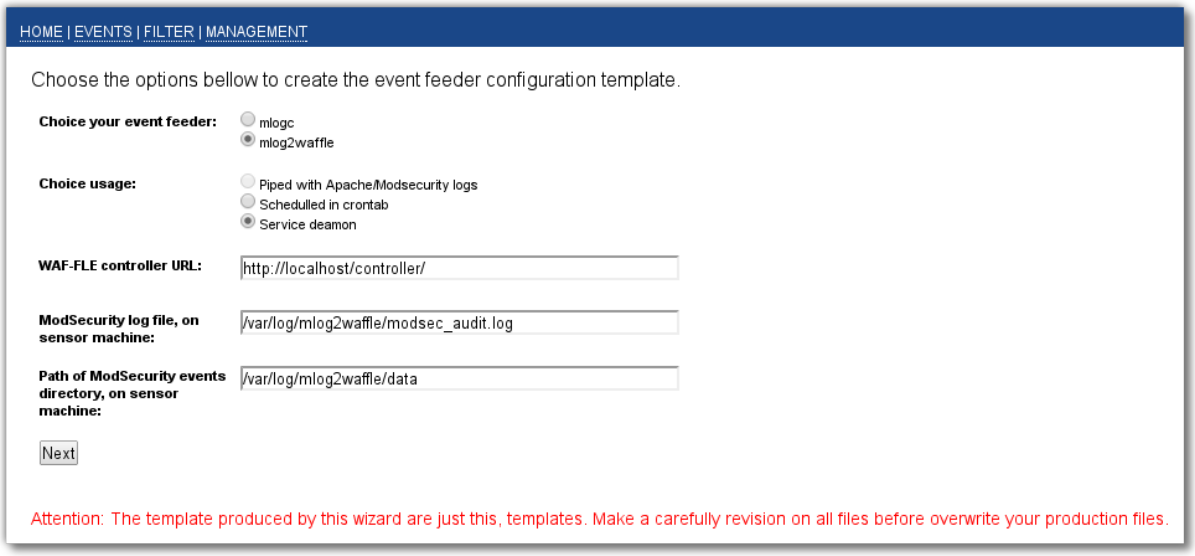
Sensor Details

Name:	waf-lab (id: 1)
IP:	Any
Description:	This is an lab sensor
Type:	ModSecurity Apache
Status:	Enabled

Info/Stats

Event's total:	10
Last event in:	2014-02-19
Producer:	ModSecurity for Apache/2.7.7 (http://www.modsecurity.org/)
Rule Set:	OWASP_CRS/2.2.9
Server:	Apache/2.4.6 (Ubuntu)

2. Make the appropriate selections:
   2.1. Select the event feeder of your preference (mlog2waffle or mlogc);
   2.2. Select the operation mode: service (available only with mlog2waffle), piped (available only with mlogc) or batch;
   2.3. Confirm or edit the controller URL. This URL is how the event feeder will send events to WAF-FLE, typically it has the form http://<WAF-FLE>/controller.
   2.4. Check the suggested ModSecurity log file (on sensor machine), and needed only on service or batch mode. This is defined in modsecurity.conf in SecAuditLog directive.
   2.5. Check the suggested path of ModSecurity events directory, on sensor machine. This is defined in m*odsecurity.conf* in *SecAuditLogStorageDir* directive.



Choose the options bellow to create the event feeder configuration template.

Choice your event feeder:	mlogc [✓ mlog2waffle]
Choice usage:	Piped with Apache/Modsecurity logs; Schedulled in crontab; [✓ Service deamon]
WAF-FLE controller URL:	http://localhost/controller/
ModSecurity log file, on sensor machine:	/var/log/mlog2waffle/modsec_audit.log
Path of ModSecurity events directory, on sensor machine:	/var/log/mlog2waffle/data

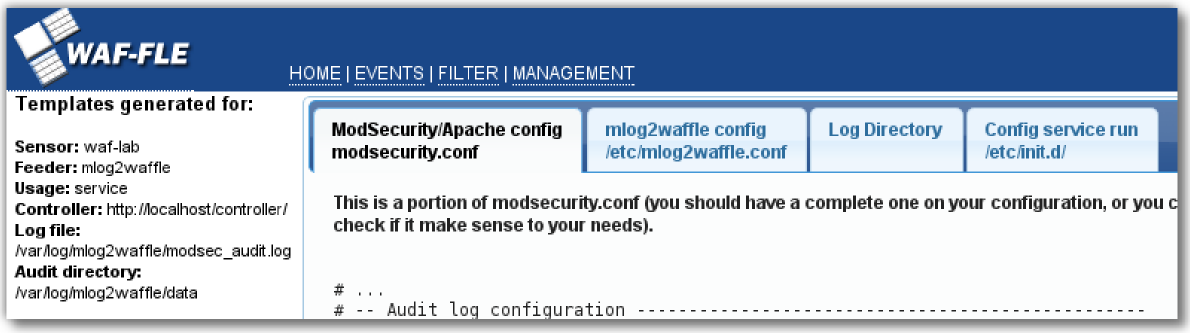
Attention: The template produced by this wizard are just this, templates. Make a carefully revision on all files before overwrite your production files.

3. Click <u>Next</u> to create file templates. This templates must be copied and sometimes edited by you in sensors machine;
4. Depending on your selection, you have some files to edit/create

	mlog2waffle	**mlogc**
Servic	- modsecurity.conf	

e	- /etc/mlog2waffle.conf - Directory for event data - Enable mlog2waffle init script	—
Batch	- modsecurity.conf - /etc/mlog2waffle.conf - Directory for event database - Add mlog2waffle to crontab	- modsecurity.conf - /etc/mlogc.conf - Directory for event data - Add push-mlogc.sh to crontab
Piped	—	- modsecurity.conf - /etc/mlogc.conf - Directory for event data

4.1. On the left side, you can review the choices you made:


WAF-FLE
HOME | EVENTS | FILTER | MANAGEMENT
ModSecurity/Apache config modsecurity.conf
mlog2waffle config /etc/mlog2waffle.conf
Log Directory
Config service run /etc/init.d/

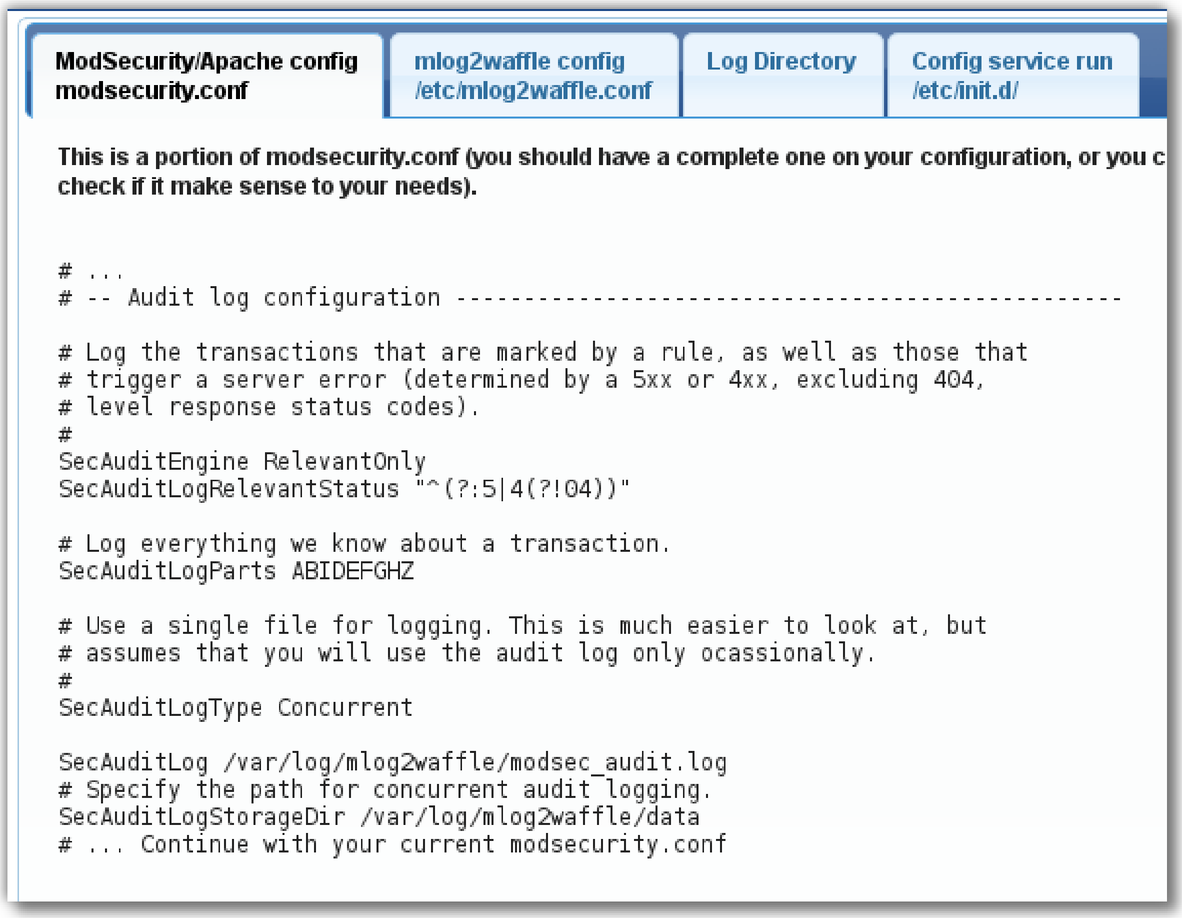
Templates generated for:

Sensor: waf-lab
Feeder: mlog2waffle
Usage: service
Controller: http://localhost/controller/
Log file: /var/log/mlog2waffle/modsec_audit.log
Audit directory: /var/log/mlog2waffle/data

This is a portion of modsecurity.conf (you should have a complete one on your configuration, or you c
check if it make sense to your needs).

```
# ...
# -- Audit log configuration -----------------------------------------------
```

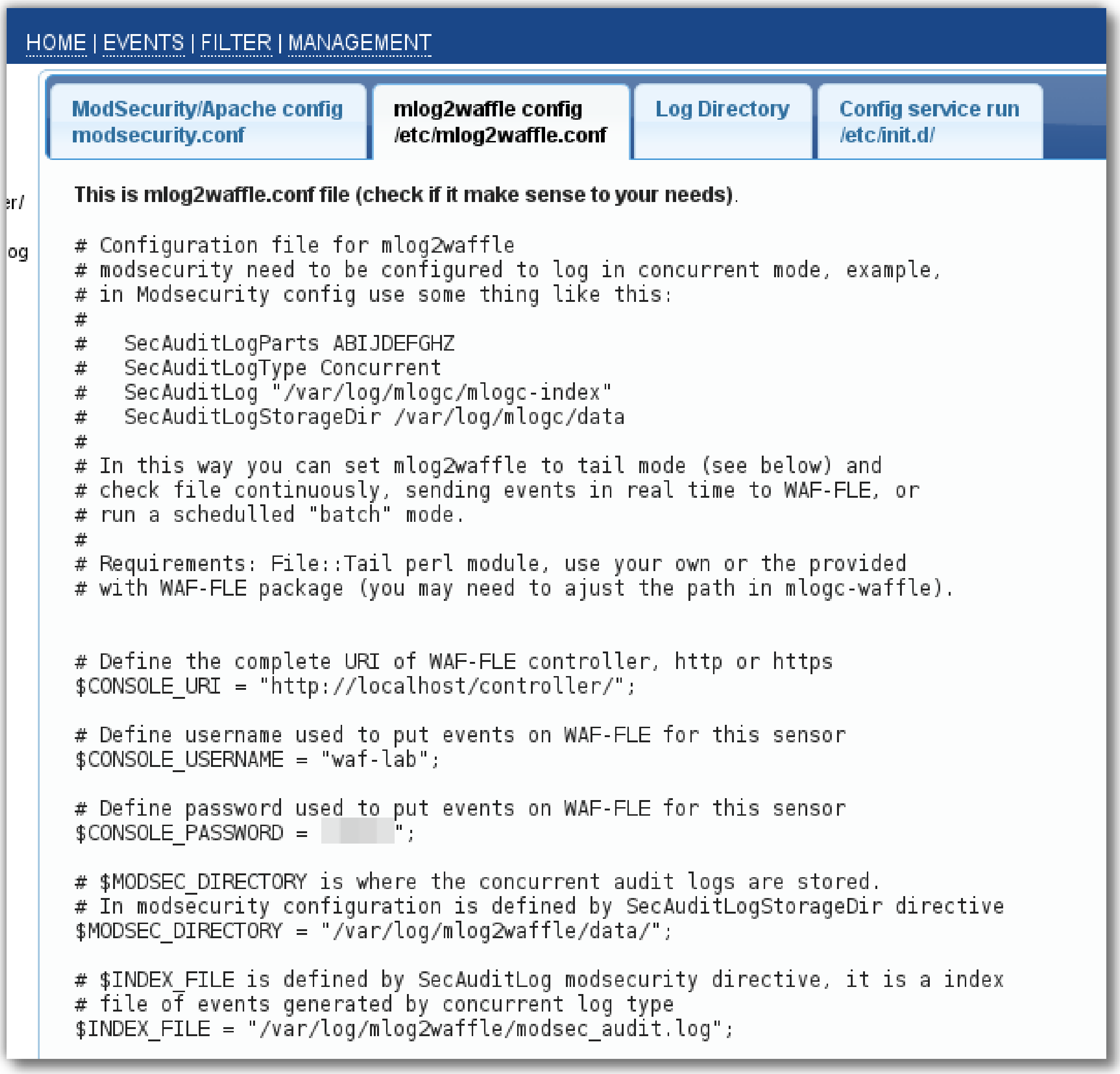
4.2. The first tab present what you need to change in modsecurity.conf file to use the log appropriately with the event feeder and the mode you chose. modsecurity.conf is a file that define basic configuration for ModSecurity, including how log will behave.



This is a portion of modsecurity.conf (you should have a complete one on your configuration, or you c
check if it make sense to your needs).

```
# ...
# -- Audit log configuration -------------------------------------------------

# Log the transactions that are marked by a rule, as well as those that
# trigger a server error (determined by a 5xx or 4xx, excluding 404,
# level response status codes).
#
SecAuditEngine RelevantOnly
SecAuditLogRelevantStatus "^(?:5|4(?!04))"

# Log everything we know about a transaction.
SecAuditLogParts ABIDEFGHZ

# Use a single file for logging. This is much easier to look at, but
# assumes that you will use the audit log only ocassionally.
#
SecAuditLogType Concurrent

SecAuditLog /var/log/mlog2waffle/modsec_audit.log
# Specify the path for concurrent audit logging.
SecAuditLogStorageDir /var/log/mlog2waffle/data
# ... Continue with your current modsecurity.conf
```

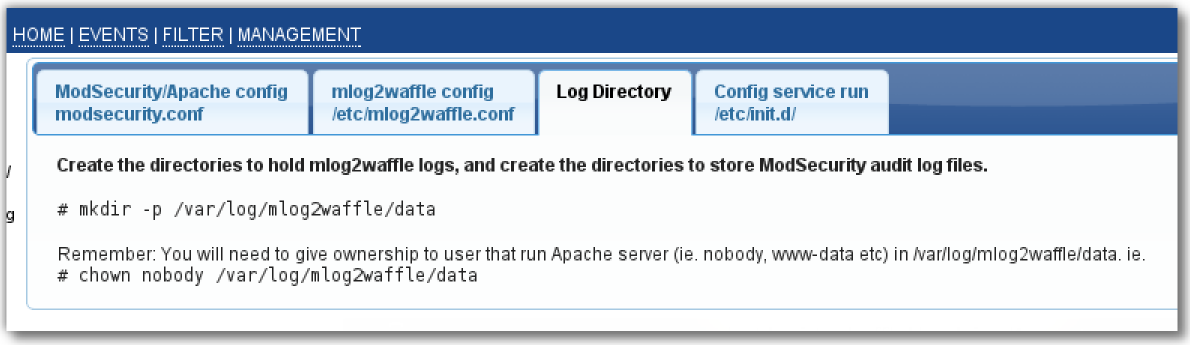
4.3. The second tab shows you mlog2waffle.conf or mlogc.conf template. This is a complete configuration file, you can replace your file with this new one, that include the WAF-FLE URL, username and password needed to authenticate the sensor in WAF-FLE.
Others details of mlog2waffle.conf or mlogc.conf are outside this deployment guide, but they are documented in default configuration file from both log feeders.


HOME | EVENTS | FILTER | MANAGEMENT
ModSecurity/Apache config modsecurity.conf
mlog2waffle config /etc/mlog2waffle.conf
Log Directory
Config service run /etc/init.d/

This is mlog2waffle.conf file (check if it make sense to your needs).

```
# Configuration file for mlog2waffle
# modsecurity need to be configured to log in concurrent mode, example,
# in Modsecurity config use some thing like this:
#
#   SecAuditLogParts ABIJDEFGHZ
#   SecAuditLogType Concurrent
#   SecAuditLog "/var/log/mlogc/mlogc-index"
#   SecAuditLogStorageDir /var/log/mlogc/data
#
# In this way you can set mlog2waffle to tail mode (see below) and
# check file continuously, sending events in real time to WAF-FLE, or
# run a schedulled "batch" mode.
#
# Requirements: File::Tail perl module, use your own or the provided
# with WAF-FLE package (you may need to ajust the path in mlogc-waffle).


# Define the complete URI of WAF-FLE controller, http or https
$CONSOLE_URI = "http://localhost/controller/";

# Define username used to put events on WAF-FLE for this sensor
$CONSOLE_USERNAME = "waf-lab";

# Define password used to put events on WAF-FLE for this sensor
$CONSOLE_PASSWORD = [redacted]";

# $MODSEC_DIRECTORY is where the concurrent audit logs are stored.
# In modsecurity configuration is defined by SecAuditLogStorageDir directive
$MODSEC_DIRECTORY = "/var/log/mlog2waffle/data/";

# $INDEX_FILE is defined by SecAuditLog modsecurity directive, it is a index
# file of events generated by concurrent log type
$INDEX_FILE = "/var/log/mlog2waffle/modsec_audit.log";
```

4.4. The third tab refer to directory needed to hold logs before they are feed to WAF-FLE.
Note: Pay attention to *chown* command, it need to be executed with appropriate user, that run Apache.


HOME | EVENTS | FILTER | MANAGEMENT
ModSecurity/Apache config modsecurity.conf
mlog2waffle config /etc/mlog2waffle.conf
Log Directory
Config service run /etc/init.d/

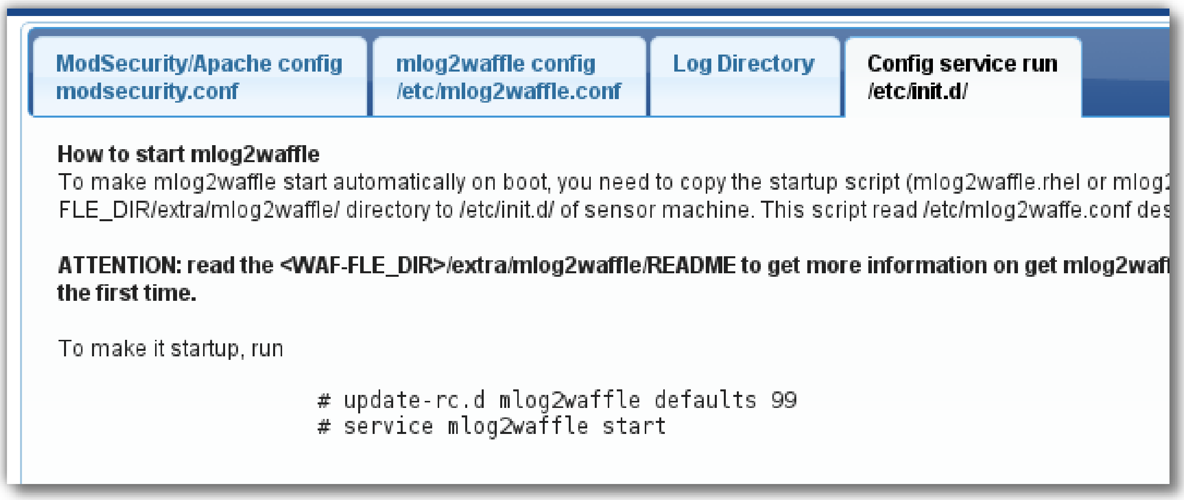
Create the directories to hold mlog2waffle logs, and create the directories to store ModSecurity audit log files.

```
# mkdir -p /var/log/mlog2waffle/data
```

Remember: You will need to give ownership to user that run Apache server (ie. nobody, www-data etc) in /var/log/mlog2waffle/data. ie.

```
# chown nobody /var/log/mlog2waffle/data
```

4.5. Fourth tab (if present), is the init script for mlog2waffle service, or crontab entries to run both mlog2waffle or mlogc in batch.


ModSecurity/Apache config modsecurity.conf
mlog2waffle config /etc/mlog2waffle.conf
Log Directory
Config service run /etc/init.d/

How to start mlog2waffle
To make mlog2waffle start automatically on boot, you need to copy the startup script (mlog2waffle.rhel or mlog2
FLE_DIR/extra/mlog2waffle/ directory to /etc/init.d/ of sensor machine. This script read /etc/mlog2waffe.conf des

**ATTENTION: read the <WAF-FLE_DIR>/extra/mlog2waffle/README to get more information on get mlog2waf
the first time.**

To make it startup, run

```
# update-rc.d mlog2waffle defaults 99
# service mlog2waffle start
```

5. After edit all files, you need:
   5.1. Reload Apache to make modsecurity.conf changes effective;
   5.2. For mlog2waffle service mode, start the service:
   (/etc/init.d/mlog2waffle start)
6. Test the system. Make a request that must be triggered by ModSecurity as an event, and check if the event come to WAF-FLE.

Configuring mlog2waffle.conf as a Service Daemon (tail mode)

As shown in *Event Feeder Wizard,* you can use the mlog2waffle in the ModSecurity sensor box as a service daemon (tail mode) using steps below, (use configuration from *Event Feeder Wizard* as your reference):

1. Configure the m*odsecurity.conf* to adjust log definitions:
   All configuration in the box is relevant to log configuration, but the bold part is fundamental. To understand more about ModSecurity logging check https://github.com/SpiderLabs/ModSecurity/wiki/Reference-Manual#wiki-secauditlog

```
# ...
SecAuditEngine RelevantOnly
SecAuditLogRelevantStatus "^(?:5|4(?!04))"

# Log everything we know about a transaction.
SecAuditLogParts ABIDEFGHZ

SecAuditLogType Concurrent
# Specify the log index
SecAuditLog /var/log/mlog2waffle/modsec_audit.log
# Specify the path for concurrent audit logging.
SecAuditLogStorageDir /var/log/mlog2waffle/data
# ...
```

2. Copy mlog2waffle files from your WAF-FLE box to the sensor box and install required packages (see more details in How-To below or in mlog2waffle README file).

```
cd waf-fle/extra/mlog2waffle
cp mlog2waffle /usr/sbin
cp mlog2waffle.conf /etc
# for RedHat based dist
cp mlog2waffle.rhel /etc/init.d/mlog2waffle
# for Debian/Ubuntu based dist
cp mlog2waffle.ubuntu /etc/init.d
```

3. Create ModSecurity log directories and give proper permission (location and permissions should be reviewed, because are system dependent). The ownership of /var/log/mlog2waffle/data need to be given to user running Apache (i.e. nobody, www-data, apache):

```
mkdir -p /var/log/mlog2waffle/data
chown -R nobody /var/log/mlog2waffle/data
```

4. Edit the configuration file (mlog2waffle.conf) and adjust to your needs. Highlighted the more relevant directives (others are commented in the file):

```
vi /etc/mlog2waffle.conf
```

```
# ...
# Define the complete URI of WAF-FLE controller, http or https
$CONSOLE_URI = "https://<host>/controller/";

# Define username used to put events on WAF-FLE for this sensor
$CONSOLE_USERNAME = "<sensor-name>";

# Define password used to put events on WAF-FLE for this sensor
$CONSOLE_PASSWORD = "<password>";

# $MODSEC_DIRECTORY is where the concurrent audit logs are stored.
#  In  modsecurity  configuration  is  defined  by  SecAuditLogStorageDir
# directive
$MODSEC_DIRECTORY = "/var/log/mlog2waffle/data/";

# $INDEX_FILE is defined by SecAuditLog modsecurity directive, it is
# a index file of events generated by concurrent log type
$INDEX_FILE = "/var/log/mlog2waffle/modsec_audit.log";
# ...
# Define the execution mode:
#  "tail":  for  run  continuously,  waiting  for  new  entries  on  log
# file;
$MODE = "tail";
# ...
```

5. Define the mlog2waffle automatic startup. Start it the first time:

```
ln -s /etc/init.d/mlog2waffle /etc/rc3.d/S99mlog2waffle
/usr/sbin/mlog2waffle
```

6. Reload Apache to make the changes in modsecurity.conf effective;

Configuring mlog2waffle.conf Scheduled in crontab (batch mode)

As shown in Event Feeder Wizard, you can use the mlog2waffle in ModSecurity sensor box in scheduled crontab mode (batch mode) using steps below (use Event Feeder Wizard as your reference):

1. Configure modsecurity.conf to adjust log definitions:
   All configuration in the box is relevant to log configuration, but the bold part is fundamental. To understand more about ModSecurity logging, check: https://github.com/SpiderLabs/ModSecurity/wiki/Reference-Manual#wiki-secau

ditlog

```
# ...
SecAuditEngine RelevantOnly
SecAuditLogRelevantStatus "^(?:5|4(?!04))"

# Log everything we know about a transaction.
SecAuditLogParts ABIDEFGHZ

SecAuditLogType Concurrent
# Specify the log index
SecAuditLog /var/log/mlog2waffle/modsec_audit.log
# Specify the path for concurrent audit logging.
SecAuditLogStorageDir /var/log/mlog2waffle/data
# ...
```

2. Copy mlog2waffle files from you WAF-FLE box to the sensor box and install required packages (see more details in How-To below or in mlog2waffle README file).

```
cd waf-fle/extra/mlog2waffle
cp mlog2waffle /usr/sbin
cp mlog2waffle.conf /etc
cp mlog2waffle.cron /etc/cron.d/mlog2waffle
```

3. Create ModSecurity log directories and give proper permissions (location and permissions should be reviewed, because are system dependent).

```
mkdir -p /var/log/mlog2waffle/data
chown -R nobody /var/log/mlog2waffle/data
```

4. Edit the configuration file (mlog2waffle.conf) and adjust to your needs. Highlighted the most relevant directives (others are commented in the file):

```
vi /etc/mlog2waffle.conf
```

```
# ...
# Define the complete URI of WAF-FLE controller, http or https
$CONSOLE_URI = "https://<host>/controller/";

# Define username used to put events on WAF-FLE for this sensor
$CONSOLE_USERNAME = "<sensor-name>";

# Define password used to put events on WAF-FLE for this sensor
$CONSOLE_PASSWORD = "<password>";

# $MODSEC_DIRECTORY is where the concurrent audit logs are stored.
#  In modsecurity configuration is defined by SecAuditLogStorageDir
# directive
$MODSEC_DIRECTORY = "/var/log/mlog2waffle/data/";

# $INDEX_FILE is defined by SecAuditLog modsecurity directive, it is
# a index file of events generated by concurrent log type
$INDEX_FILE = "/var/log/mlog2waffle/modsec_audit.log";
# ...
# Define the execution mode:
# "batch": for run and exit at end, but recording (offset file) the
# position in the last run, speeding up next execution. You can
#  schedulle the mlog2waffle in crontab to run periocally (for
# example, each 5min).
$MODE = "batch";
# ...
```

5. Edit the crontab entry (copied in step 2) to run the mlog2waffle with periodicity needed by you:

```
vi /etc/cron.d/mlog2waffle
```

```
PATH=/sbin:/usr/sbin:/bin:/usr/bin

# start mlog2waffle periodically, in this case 5 minutes
*/5 * * * * root mlog2waffle
```

6. Reload the Apache to make changes in modsecurity.conf effective;

Configuring mlogc Scheduled in crontab (batch mode)

As shown in Event Feeder Wizard, you can use the mlogc in ModSecurity sensor box in scheduled crontab mode (batch mode) using steps below, (use the Event Feeder Wizard as your reference):

1. Configure modsecurity.conf to adjust log definitions:
   All configuration in the box is relevant to log configuration, but the bold part is fundamental. To understand more about ModSecurity logging, check: https://github.com/SpiderLabs/ModSecurity/wiki/Reference-Manual#wiki-secauditlog

```
# ...
SecAuditEngine RelevantOnly
SecAuditLogRelevantStatus "^(?:5|4(?!04))"

# Log everything we know about a transaction.
SecAuditLogParts ABIDEFGHZ

SecAuditLogType Concurrent
# Specify the log index
SecAuditLog /var/log/mlogc/modsec_audit.log
# Specify the path for concurrent audit logging.
SecAuditLogStorageDir /var/log/mlogc/data
# ...
```

2. Check if the mlogc is installed on /usr/local/modsecurity/bin/, if you compiled ModSecurity on machine it should be there (i.e. /usr/local/modsecurity/bin/mlogc).

3. Create ModSecurity log directories and give proper permissions (location and permissions should be reviewed, because are system dependent). The ownership of /var/log/mlogc/data need to be given to user that run the Apache (i.e. nobody, www-data, apache):

```
mkdir -p /var/log/mlogc/data
chown nobody /var/log/mlogc/data
```

4. Edit the configuration file (mlogc.conf) and adjust to your needs. Highlighted the most relevant directives (others are commented in the file):

```
vi /etc/mlogc.conf
```

```
# Points to the root of the installation. All relative
# paths will be resolved with the help of this path.
CollectorRoot        "/var/log/mlogc"

# ModSecurity Console receiving URI. You can change the host
# and the port parts but leave everything else as is.
ConsoleURI           "http://<host>/controller/"

# Sensor credentials
SensorUsername       "<sensor-name>"
SensorPassword       "<password>"

# Base directory where the audit logs are stored.  This can be
  specified
# as a path relative to the CollectorRoot, or a full path.
LogStorageDir        "data"

# Transaction log will contain the information on all log collector
# activities that happen between checkpoints. The transaction log
# is used to recover data in case of a crash (or if Apache kills
# the process).
TransactionLog       "mlogc-transaction.log"

# The file where the pending audit log entry data is kept. This file
# is updated on every checkpoint.
QueuePath            "mlogc-queue.log"

# The location of the error log.
ErrorLog             "mlogc-error.log"

# ...
```

5. Check that the script mlogc-batch-load.pl (installed by default) is in /usr/local/modsecurity/bin/. It will read and will process the log entries, and will use mlogc to send it to the WAF-FLE. The mlogc-batch-load.pl come in ModSecurity source code.

6. Create a script, named push-mlogc.sh and put on /usr/local/sbin/. Use the lines bellow. Make the file executable.

```
vi /usr/local/modsecurity/bin/push-mlogc.sh
```

```
#!/bin/bash

# Check if a old execution still running, and kill it
Status=0;
while [ $Status -eq 0 ]; do
  PmlogcBatch=`/sbin/pidof -x /usr/local/modsecurity/bin/mlogc-batch-load.pl`
  PplStatus=$?
  Pmlogc=`/sbin/pidof -x /usr/sbin/mlogc`
  PmlogcStatus=$?

  if [ $PplStatus -eq 0 ]; then
     kill -9 $PmlogcBatch
     echo "Killing $PmlogcBatch"
  fi
  if [ $PmlogcStatus -eq 0 ]; then
     kill -9 $Pmlogc
     echo "Killing $Pmlogc"
  fi

  if [ $PplStatus -ne 0 -a $PmlogcStatus -ne 0 ]; then
     Status=1;
  fi
done

# Start mlogc push
echo "Sending logs to WAF-FLE";
date
/usr/local/modsecurity/bin/mlogc-batch-load.pl    /var/log/mlogc/data    \
/usr/local/modsecurity/bin/mlogc /etc/mlogc.conf

find  /var/log/mlogc/data -type d -empty -delete
```

```
chmod +x /usr/local/modsecurity/bin/push-mlogc.sh
```

7. Create a crontab entry to run push-mlogc.sh each 5 minutes;

```
PATH=/sbin:/usr/sbin:/bin:/usr/bin

# start push-mlogc periodically, in this case 5 minutes
*/5 * * * * root /usr/local/modsecurity/bin/push-mlogc.sh
```

8. Reload the Apache to make changes in modsecurity.conf effective;

Configuring mlogc Piped with Apache/ModSecurity log

As shown in Event Feeder Wizard, you can use the mlogc in ModSecurity sensor box in piped mode (that is the original mlogc configuration), using the steps below (use the Event Feeder Wizard as your reference):

1. Configure the modsecurity.conf to adjust log definitions:
   All configuration in the box is relevant to log configuration, but the bold part is fundamental. To understand more about the ModSecurity logging, check: https://github.com/SpiderLabs/ModSecurity/wiki/Reference-Manual#wiki-secauditlog

```
# ...
SecAuditEngine RelevantOnly
SecAuditLogRelevantStatus "^(?:5|4(?!04))"

# Log everything we know about a transaction.
SecAuditLogParts ABIDEFGHZ

SecAuditLogType Concurrent
# Specify the log index
SecAuditLog "|/usr/local/bin/mlogc /etc/mlogc.conf"
# Specify the path for concurrent audit logging.
SecAuditLogStorageDir /var/log/mlogc/data
# ...
```

2. Check if the mlogc is installed on /usr/local/modsecurity/bin/, if you compiled ModSecurity on machine it should be there (i.e. /usr/local/modsecurity/bin/mlogc).

3. Create ModSecurity log directories and give proper permissions (location and permissions should be reviewed, because are system dependent). The ownership of /var/log/mlogc/data need to be given to user that run the Apache (i.e. nobody, www-data, apache):

```
mkdir -p /var/log/mlogc/data
chown nobody /var/log/mlogc/data
```

4. Edit the configuration file (mlogc.conf) and adjust to your needs. Highlighted the most relevant directives (others are commented in the file):

```
vi /etc/mlogc.conf
```

```
# Points to the root of the installation. All relative
# paths will be resolved with the help of this path.
CollectorRoot       "/var/log/mlogc"

# ModSecurity Console receiving URI. You can change the host
# and the port parts but leave everything else as is.
ConsoleURI          "http://<host>/controller/"

# Sensor credentials
SensorUsername      "<sensor-name>"
SensorPassword      "<password>"

#  Base  directory  where  the  audit  logs  are  stored.   This  can  be
  specified
# as a path relative to the CollectorRoot, or a full path.
LogStorageDir       "data"

# Transaction log will contain the information on all log collector
# activities that happen between checkpoints. The transaction log
# is used to recover data in case of a crash (or if Apache kills
# the process).
TransactionLog      "mlogc-transaction.log"

# The file where the pending audit log entry data is kept. This file
# is updated on every checkpoint.
QueuePath           "mlogc-queue.log"

# The location of the error log.
ErrorLog            "mlogc-error.log"

# ...
```

5. Reload the Apache to make changes in modsecurity.conf effective;

Quick How-To

This How-To was created to help you to setup the WAF-FLE with popular distributions and operating systems more quickly. With time, others operating systems and distributions will be added. The focus is to provide the steps, in the OS/distribution specific commands and packages, to meet the requirements to run WAF-FLE in each of these systems.

NOTE: These instructions are basic, could be incomplete, and are just a kickoff to make a system able to run WAF-FLE.

Further explanation about the WAF-FLE installation is provided in the first part of this guide.

CentOS/RedHat 6.5

WAF-FLE Requirements

```
yum install httpd
yum install mysql-server
yum install php php-pdo php-mysql php-pecl-apc

#   Install   GeoIP   and   GeoIP   for   PHP[1],   download   from   EPEL
http://pkgs.org/download/GeoIP and http://pkgs.org/download/php-pecl-geoip
yum localinstall php-pecl-geoip-1.0.8-3.el6.x86_64.rpm
yum localinstall GeoIP-1.4.8-1.el6.x86_64.rpm

# After GeoIP install, download all MaxMind GeoIP Database, as follow:
cd /usr/share/GeoIP/
wget
http://geolite.maxmind.com/download/geoip/database/GeoLiteCountry/GeoIP.dat
.gz
wget http://geolite.maxmind.com/download/geoip/database/GeoLiteCity.dat.gz
wget
http://geolite.maxmind.com/download/geoip/database/asnum/GeoIPASNum.dat.gz

gzip -d GeoIP.dat.gz
gzip -d GeoLiteCity.dat.gz
gzip -d GeoIPASNum.dat.gz
mv GeoLiteCity.dat GeoIPCity.dat
# To make php GeoIP extension works with ASNum database
cp GeoIPASNum.dat GeoIPISP.dat

/etc/init.d/httpd start
# check if Apache is working properly
# check if IPTables rules allow your client connect to Apache
/etc/init.d/mysqld start
# define a password for root user in MySQL
/usr/bin/mysql_secure_installation
```

Follow the steps described in chapter: **WAF-FLE installation**

1 Instead of install from EPEL, you can follow the instruction from: http://blog.thecodingmachine.com/fr/content/installing-php-geolocalization-extension-centos

Debian 7 (wheezy) /Ubuntu 12.04 LTS (precise)

WAF-FLE Setup

```
apt-get install apache2
apt-get install mysql-server
apt-get install php5 php5-mysql php-apc php5-geoip

# After GeoIP install, download all MaxMind GeoIP Database, as follow:
cd /usr/share/GeoIP/
wget
http://geolite.maxmind.com/download/geoip/database/GeoLiteCountry/GeoIP.dat
.gz
wget http://geolite.maxmind.com/download/geoip/database/GeoLiteCity.dat.gz
wget
http://geolite.maxmind.com/download/geoip/database/asnum/GeoIPASNum.dat.gz

gzip -d GeoIP.dat.gz
gzip -d GeoLiteCity.dat.gz
gzip -d GeoIPASNum.dat.gz
mv GeoLiteCity.dat GeoIPCity.dat
# To make php GeoIP extension works with ASNum database
cp GeoIPASNum.dat GeoIPISP.dat

# enable mod_rewrite, needed to waf-fle
a2enmod rewrite
service apache2 restart

# check if Apache is working properly
# check if IPTables rules allow your client connect to Apache
```

Follow the steps described in chapter: **WAF-FLE installation**

FreeBSD 10

```
pkg install apache24
# To run apache www server from startup, add
# apache24_enable="yes" in your /etc/rc.conf.
# Uncomment the line
# "LoadModule rewrite_module libexec/apache24/mod_rewrite.so" in
# /usr/local/etc/apache24/httpd.conf

pkg install mysql55-server
# To run MySql from startup, add mysql_enable="YES" in your /etc/rc.conf.


# to install php you will need to use Ports[2]
# by now, you should use php 5.4 ( php5) as APC is not
# available in PHP 5.5

portsnap fetch
cd /usr/ports/lang/php55
make config
# select "Build Apache Module"
make install

cp /usr/local/etc/php.ini-production /usr/local/etc/php.ini

# edit /usr/local/etc/apache24/httpd.conf to include
AddType application/x-httpd-php .php
# and
<IfModule dir_module>
 DirectoryIndex index.php index.html
</IfModule>

cd /usr/ports/lang/php5-extensions
make config

# select the following extensions (keep the already selected as is)
      JSON
      PDO
      PDO_MYSQL
      ZLIB
      SESSION

make install

cd /usr/ports/www/pecl-APC/
make installation
vi /usr/local/etc/php/extensions.ini:

   apc.enabled=1
   apc.shm_size=32M # or other value appropriated to your setup
```

2 Steps adapted from http://fosskb.wordpress.com/2013/07/15/famp-installing-apache2-2-mysql-php-on-freebsd-9-1/

```
cd /usr/ports/net/pecl-geoip
make install
# After GeoIP install, download all MaxMind GeoIP Database, follow:
cd /usr/local/share/GeoIP

wget
http://geolite.maxmind.com/download/geoip/database/GeoLiteCountry/GeoIP.dat
.gz
wget http://geolite.maxmind.com/download/geoip/database/GeoLiteCity.dat.gz
wget
http://geolite.maxmind.com/download/geoip/database/asnum/GeoIPASNum.dat.gz

gzip -d GeoIP.dat.gz
gzip -d GeoLiteCity.dat.gz
gzip -d GeoIPASNum.dat.gz
mv GeoLiteCity.dat GeoIPCity.dat
# To make php GeoIP extension work with ASNum database
cp GeoIPASNum.dat GeoIPISP.dat

/usr/local/sbin/apachectl restart
```

Follow the steps described in chapter: **WAF-FLE installation**

Sizing

The sizing of WAF-FLE machines is not very demanding in resources, but this can vary with your needs.
The resources for WAF-FLE are directly dependent of events received/processed per second and the console usage. More events: more resources needed (CPU, memory and storage).

As a reference:

- For # 1 – Standalone (lab) machine: start with 1G of memory;
- For # 1 – Standalone (production): consider your application needs, plus extra memory. Remember that in this case you already has or will deploy ModSecurity too, that already has his own requirements.
- For #2 – Distributed: start with 4G, at least 2 (V)CPU Core, and sufficient storage to keep your events, and extra storage available to make MySQL maintenance. In a real server, consider to use a good RAID solution to don't make I/O a problem.
- For #3 - Distributed, dedicated database: In this case we are expecting much more event per second.
    - For WAF-FLE, start with: 4GB of memory, at least 4 (V)CPU Cores, and tune Apache to support sensors events concurrency;
    - For MySQL, start with 8GB of memory, and optimized storage, with sufficient space for your events.

How many disk space for database depend upon how many events you expect in a time frame. For example, consider an event (as stored in database, compressed) between 5kB to 10kB, with 10.000 events per sensor in a day. So you in get 300.000 events per month, per sensor. What make a 2.9GB per sensor per month.

Use the formula:

$$\text{Database Size (in GB)} = \frac{\text{event size (in KB)} * number\ of\ events * Days\ do\ keep * Number\ of\ Sensors}{1024 * 1024}$$

example:

$$2{,}86 = \frac{10 * 10000 * 30 * 1}{1024 * 1024}$$

Note:the size of event is variable, the sections of log recorded by ModSecurity, and the size of response body (if logged).

Remember: the amount of events is impacted by three factors:

1. Rules: If you make a rule that is triggered by anything, you will get many

events, doesn't matter if is in blocking mode or detection only.

2. Request triggering events: even very tuned rules, can produce many events if you get a bunch of events that trigger the rules. Scanners, worms, DoS, and other kinds of attacks are examples of this.
3. Audit log of Relevant Only/Relevant Status (SecAuditEngine RelevantOnly/ SecAuditLogRelevantStatus): if you configure a too wide Relevant status, you can receive thousands (or millions) of events in case an error in application.

MySQL Tunning

For growing installations, you can need to tune your MySQL installation, for what you can some more specialized references below:

Understanding caches and buffers of MySQL, a good explanation about it:
http://www.mysqlperformanceblog.com/2006/09/29/what-to-tune-in-mysql-server-after-installation/

For a more automatic review of your MySQl server you can use tools that make some advisory about your specific need, a good article summarizing great tools is:
http://www.askapache.com/mysql/performance-tuning-mysql.html

Percona Tools can help you to configure your my.cnf by ask questions:
https://tools.percona.com/wizard (free, but need registration).

The WAF-FLE user Fábio Miranda shared his tuning tip of MySQL, with great improve in response time (from a base about 70GB running on a VM with 6GB RAM):

In /etc/my.cnf :

```
#tuning
query_cache_size=64M
thread_cache_size=4
table_cache=256
key_buffer_size=1300M
```