Tightening PHP Security on Apache 2.2 with ModSecurity2 on Debian Lenny Linux

Author : admin

In this article you'll learn how I easily installed and configured the ModSecurity 2 on a Debian Lenny system.
First let me give you a few introductionary words to modsecurity, what is it and why it's a good idea to install and use it on your Apache Webserver.

ModSecurity is an Apache module that provides intrusion detection and prevention for web applications. It aims at shielding web applications from known and unknown attacks, such as SQL injection attacks, cross-site scripting, path traversal attacks, etc.

As you can see from ModSecurityâ€™s description itâ€™s a priceless module add on to Apache that is able to protect your PHP Applications and Apache server from a huge number of hacker attacks undertook against your Online Web Application or Webserver.
The only thing I donâ€™t like about this module is that it is actually a 3rd party module (e.g. not officially part of Apache). Some time ago I remember there was even an exploit for one of the versions of the module.
So in some cases the ModSecurity could also pose a security risk, so beware!
However if you know what you'rre doing and you keep a regular track of security news on some major security websites, that shouldnâ€™t be a concern for you.
Now let'ss proceed to the install of the ModSecurity module itself.
The install is a piece of cake on Debian though you'll be required to use the Debian Lenny backports

Here is the install of the module step by step:

1. First add the gpg key of the backports repository to your install

debian-server:~# gpg --keyserver pgp.mit.edu --recv-keys C514AF8E4BA401C3
another possible way to add the repository as the website describes is through the command
debian-server:~# wget -O - http://backports.org/debian/archive.key | apt-key add -

2. Install the libapache-mod-security package from the backports Debian Lenny repository

debian-server~:~# apt-get -t lenny-backports install libapache2-mod-security2

Now as a last step of the install ModSeccurity install procedure you have to add some configuration directives to Apache and restart the server afterwards.

â€“ Open your /etc/apache2/apache2.conf and place in it the following configurations

```
# Basic configuration options
SecRuleEngine On
SecRequestBodyAccess On
SecResponseBodyAccess Off

# Handling of file uploads
# TODO Choose a folder private to Apache.
# SecUploadDir /opt/apache-frontend/tmp/
SecUploadKeepFiles Off

# Debug log
SecDebugLog /var/log/apache2/modsec_debug.log
SecDebugLogLevel 0

# Serial audit log
SecAuditEngine RelevantOnly
SecAuditLogRelevantStatus ^5
SecAuditLogParts ABIFHZ
SecAuditLogType Serial
SecAuditLog /var/log/apache2/modsec_audit.log

# Maximum request body size we will
# accept for buffering
SecRequestBodyLimit 131072

# Store up to 128 KB in memory SecRequestBodyInMemoryLimit 131072
# Buffer response bodies of up to # 512 KB in length SecResponseBodyLimit 524288
```

The ModSecurity2 module would be properly installed and configured as an Apache module.

3.All left is to restart Apache in order the new module and configurations to take effect.

debian-server:~# /etc/init.d/apache restart

Donâ€™t forget to check the apache conf file for errors before restarting the Apache with the above command for that to happen issue the command:
debian-server:~# apache2ctl -t

If all is fine you should get as an output:

Syntax OK

4. Next to find out if the Apache ModSecurity2 module is enabled and already used by Apache as a mean of protection you,
you might want to check if the log files **modsec_audit.log** and **modsec_debug.log** files has grown and doesfeed a new content.
If theyâ€™re growing and you see messages concerning the operation of the ModSecurity2 Apache module thatâ€™s a sure sign all is fine.
5. **As we have the Mod Security Apache module configured on our Debian Server, now we will need to apply some ModSecurity** Core Rules .
In short **ModSecurity Core Rules are** some critical protection rules against attacks across almost every web architecture.
Another really neat thing about Core Rules (CRS) for ModSecurity is that they are written with a performance in mind.
So enabling this filter rules wonâ€™t be a too heavy load for your Apache server.

Here is how to install the core rules:

6. **Download latest ModSecurity Code Rules**

Download them from the following Code Rule url
At the time of writting this article the latest code rules are version modsecurity-crs_2.0.6.tar.gz

To download and install this rules issue some commands like:

debian-server:~# wget http://sourceforge.net/projects/mod-security/files/modsecurity-crs/0-CURRENT/modsecurity-crs_2.0.6.tar.gz/download
debian-server:~# cp -rpf ~/modsecurity-crs_2.0.6.tar.gz /etc/apache2/
debian-server:~# cd /etc/apache2/; tar -zxvvf modsecurity-crs_2.0.6.tar.gz

Besides physically storing the unarchived modsecirity-crs in your **/etc/apache2** itâ€™s also necessery to add to your Apache **Ifmodule mod_security.c** block of code the following two lines:

Include /etc/apache2/modsecurity-crs_2.0.6/*.conf
Include /etc/apache2/modsecurity-crs_2.0.6/base_rules/*.conf

Thus ultimately the configuration concerning ModSecurity in your Apache Server configuration should look like the following:

```
# Basic configuration options
SecRuleEngine On
SecRequestBodyAccess On
SecResponseBodyAccess Off

# Handling of file uploads
# TODO Choose a folder private to Apache.
```

```
# SecUploadDir /opt/apache-frontend/tmp/
SecUploadKeepFiles Off

  # Debug log
SecDebugLog /var/log/apache2/modsec_debug.log
SecDebugLogLevel 0

  # Serial audit log
SecAuditEngine RelevantOnly
SecAuditLogRelevantStatus ^5
SecAuditLogParts ABIFHZ
SecAuditLogType Serial
SecAuditLog /var/log/apache2/modsec_audit.log

  # Maximum request body size we will
# accept for buffering
SecRequestBodyLimit 131072

  # Store up to 128 KB in memory
SecRequestBodyInMemoryLimit 131072
SecRequestBodyInMemoryLimit 131072

  # Buffer response bodies of up to
# 512 KB in length
SecResponseBodyLimit 524288
Include /etc/apache2/modsecurity-crs_2.0.6/*.conf
Include /etc/apache2/modsecurity-crs_2.0.6/base_rules/*.conf
```

Once again you have to check if everything is fine with Apache configurations with:

```
debian-server:~# apache2ctl -t
```

If itâ€™s showing once again an **OK** status. Then youâ€™re ready to restart the Webserver.

```
debian-server:~# /etc/init.d/apache2 restart
```

One example goodness of setting up the ModSecurity + the Core rule sets are that after the above described installationis fully functional.

ModSecurity will be able to track if somebody tries to execute PHP Shell on your server . ModSecurity will catch, log and block (forbid) requests to r99.txt, r59, safe0ver and possibly other hacked modifications of the php shell script

Thatâ€™s it! Now Enjoy your tightened Apache Security and Hopefully catch the script kiddie trying to h4x0r yoU :)