SECURING APACHE - MODSECURITY

The *Apache web server* is the undisputed backbone of the internet, the most prolific web server there is. It is well designed and very capable of many complex tasks. It is also heavily targeted. Port 80, the web traffic port, is always open on a company's firewall if they have a public web server, thus many attackers love using this port and attacking the servers using this port. A firewall is not always useful against these types of attacks, and while one can use an Intrusion Prevention System, they also are not specifically made for protecting your web server. This is where the *mod_security* module for Apache comes into play. This is a third-party module which you can add to your web server and which can be used as a web application firewall, greatly enhancing your servers chances of surviving an attack. Lets take a look..

What you need
You will need Apache and the development packages for it. And thats about it. Start by getting the module download (please note that the version could change)..

```
wget http://www.modsecurity.org/download/modsecurity-apache-1.9.2.tar.gz
```

Once you have that, it is the usual extraction process - ungzip it and untar it..

```
gzip -d ./modsecurity-apache-1.9.2.tar.gz
tar -xvf ./modsecurity-apache-1.9.2.tar
```

Go into the folder that has been created. You will see an *apache1* and an *apache2* folder. Depending on your version of apache you go into the relevant folder. Once there you can compile the module..

```
apxs -cia mod_security.c
```

You will of course need the *httpd-devel* package for this to work. Use your favorite method (*yum, up2date*, etc) to get this package and/or update it. Also some people may get this error (I know I did)..

```
mod_security.c: In function `sec_audit_logger_concurrent':
mod_security.c:5403: `APR_MD5_DIGESTSIZE' undeclared (first use in this
function)
mod_security.c:5403: (Each undeclared identifier is reported only once
mod_security.c:5403: for each function it appears in.)
apxs:Error: Command failed with rc=65536
```

Try this, it may work..

```
Edit modsecurity.c
#include <apr-0/apr_md5.h>
#include <apr-0/apr_user.h>

Then..
ln -s /usr/lib/libaprutil-0.so.0 /usr/lib/libaprutil-0.so
ln -s /usr/lib/libapr-0.so.0 /usr/lib/libapr-0.so

Compile..
apxs -cai -lapr-0 -laprutil-0 mod_security.c
```

Setup the Module
Now once the module has compiled, we need to check our *httpd.conf* file for the following lines (if they aren't there add them)..

```
For Apache 1.x:
```

```
AddModule mod_security.c

For Apache 2.x:
LoadModule security_module modules/mod_security.so
```

We will also use an *include* directive to specify a folder to store our configuration. Look for the following or add your own in the *httpd.conf* file..

```
Include conf.d/*.conf
```

Now in that file we will create a *modsecurity.conf* file into which we will put our module settings. Before we jump into that, go and take a look at the GotRoot website, it is a very bright bunch of people who have a very comprehensive set of rules and checks for the *mod_security* module, I would suggest downloading these rules and using them (remember the ruleset is different if you are using apache 1.x)...

```
mkdir /etc/modsecurity
wget http://www.gotroot.com/downloads/ftp/mod_security/apache2/apache2-
gotrootrules-latest.tar.gz
extract into /etc/modsecurity
```

Now that we have done that, lets get to editing that *modsecurity.conf* file which will be situated in the folder specified by the *Include* directive. Here is a sample configuration file, please adapt as needed..

```
<IfModule mod_security.c>

# Only inspect dynamic requests
# (YOU MUST TEST TO MAKE SURE IT WORKS AS EXPECTED)
#SecFilterEngine DynamicOnly

SecFilterEngine On

# Reject requests with status 500
SecFilterDefaultAction "deny,log,status:500"

# Some sane defaults
SecFilterScanPOST On
SecFilterCheckURLEncoding On
SecFilterCheckCookieFormat On
SecFilterCheckUnicodeEncoding Off
SecFilterNormalizeCookies On
# enable version 1 (RFC 2965) cookies
SecFilterCookieFormat 1

SecServerResponseToken Off

#If you want to scan the output, uncomment these
#SecFilterScanOutput On
#SecFilterOutputMimeTypes "(null) text/html text/plain"

# Accept almost all byte values
SecFilterForceByteRange 1 255

# Server masking is optional
#fake server banner -
SecServerSignature "NOYB"

#SecUploadDir /tmp
#SecUploadKeepFiles Off
```

```
# Only record the interesting stuff
SecAuditEngine RelevantOnly
SecAuditLog logs/audit_log

# You normally won't need debug logging
SecFilterDebugLevel 0
SecFilterDebugLog logs/modsec_debug_log

#And now, the rules
#Remove any of these Include lines you do not use or have rules for.

#First, add in your exclusion rules:
#These MUST come first!
Include /etc/modsecurity/exclude.conf

#Application protection rules
Include /etc/modsecurity/rules.conf

#Comment spam rules
Include /etc/modsecurity/blacklist.conf

#Bad hosts, bad proxies and other bad players
Include /etc/modsecurity/blacklist2.conf

#Bad clients, known bogus useragents and other signs of malware
Include /etc/modsecurity/useragents.conf

#Known bad software, rootkits and other malware
Include /etc/modsecurity/rootkits.conf

#Signatures to prevent proxying through your server
#only rule these rules if your server is NOT a proxy
Include /etc/modsecurity/proxy.conf

#Additional rules for Apache 2.x ONLY!  Do not add this line if you use
Apache 1.x
Include /etc/modsecurity/apache2-rules.conf
</IfModule>
```

Finally, restart your web server.

Final Words
This security module is a very useful add-on to securing your webserver, and can do a lot, so read up on the options and play around. But remember that one needs to remain vigilant for new attacks and methods to keep your web server secure. As always, have fun and learn.