Installing ModSecurity on FreeBSD

21 Feb 2014

ModSecurity is a Web Application Firewall (WAF) that can block generic as well as specific threats. Off-the-shelf firewall appliances are becoming increasingly common. ModSecurity has the advantage over these, that you can write and customize firewall rules yourself. Combined with knowledge of your web application and insight into detailed audit logs, ModSecurity allows you to block attacks much more aggressively than a vendor appliance can. I will walk you through installing and configuring ModSecurity on FreeBSD.

Initial installation

Assuming you are using the new pkgng package manager, installing Apache is easy.

```
pkg install apache22
echo 'apache22_enable=YES' >> /etc/rc.conf
service apache22 start
```

Install the ModSecurity package:

```
pkg install www/mod_security
```

Getting the Core Rule Set

ModSecurity requires firewall rule definitions. Most people use the OWASP ModSecurity Core Rule Set (CRS). The easiest way to track the OWASP CRS repository right now is to use Git. Let's make a directory for all our ModSecurity related stuff, and clone the CRS repository under it.

```
pkg install git
mkdir -p /usr/local/etc/modsecurity
cd /usr/local/etc/modsecurity
git clone https://github.com/SpiderLabs/owasp-modsecurity-crs crs
```

Creating the ModSecurity configuration

Copy the default ModSecurity config file, and fetch a necessary file which is currently not included in the port:

```
cp /usr/local/etc/modsecurity.conf-example modsecurity.conf
fetch https://raw.github.com/SpiderLabs/ModSecurity/master/unicode.mapping
cp crs/modsecurity_crs_10_setup.conf.example modsecurity_crs_10_setup.conf
```

Now we create an Apache configuration snippet in Apache's `modules.d` directory. It loads the ModSecurity module, and includes the configurations and CRS:

```
cat << EOF > /usr/local/etc/apache22/modules.d/000_modsecurity.conf
# Load ModSecurity
# Comment out the next line to temporarily disable ModSecurity:
LoadModule security2_module libexec/apache22/mod_security2.so

<IfModule security2_module>
    # Include ModSecurity configuration
    Include etc/modsecurity/modsecurity.conf
```

```
    # Include OWASP Core Rule Set (CRS) configuration and base rules
    Include etc/modsecurity/modsecurity_crs_10_setup.conf
    Include etc/modsecurity/crs/base_rules/*.conf

    # Add custom configuration and CRS exceptions here. Example:
    # SecRuleRemoveById 960015
</IfModule>
EOF
```

Starting ModSecurity

When the configuration is all set, simply restart Apache, and confirm that ModSecurity is loaded by checking Apache's log file:

```
service apache22 restart
tail /var/log/httpd-error.log
```

Hopefully, the log will show something like this:

```
ModSecurity for Apache/2.7.7 (http://www.modsecurity.org/) configured.
ModSecurity: APR compiled version="1.4.8"; loaded version="1.4.8"
ModSecurity: PCRE compiled version="8.34 "; loaded version="8.34 2013-12-15"
ModSecurity: LIBXML compiled version="2.8.0"
```

Configuring blocking mode

Now that ModSecurity is active, try making a suspicious request to your web server, for instance browse to a URL http://www.example.com/?foo=/etc/passwd. The CRS has a rule against this type of request. After browsing to the URL, you should now see this request logged in `/var/log/modsec_audit.log`.

You'll notice that the request succeeds, and the response is sent to the browser normally. The reason is that ModSecurity runs in `DetectionOnly` mode by default, in order to prevent downtime from misconfiguration or heavy-handed blocking. You can enable blocking mode simply by editing `modsecurity.conf` and changing the following line:

```
SecRuleEngine On
```

Again, restart Apache. Now, make the same suspicious request to your web server. You should now see a *"403 Forbidden"* error!

In practice, it's probably best to keep `SecRuleEngine DetectionOnly` for some time, while your users exercise the web applications. Meanwhile, you should keep an eye on `/var/log/modsec_audit.log` to see what is being blocked. If there are any false positives, you need to mitigate this by writing custom exceptions.

Maintenance

An essential resource for working with ModSecurity is the ModSecurity Handbook by Ivan Ristić. ModSecurity exposes quite some internals, and it's good to scan this book before you start writing custom rules and exceptions.

You probably want to keep the CRS updated from time to time. You can do this with Git:

```
cd /usr/local/etc/modsecurity/crs
git pull
```

https://lifeforms.nl/20140221/install-modsecurity-freebsd

FreeBSD Install mod_security For The Apache HTTPD Server

in Categories Apache, FreeBSD, Security, UNIX last updated May 1, 2009

Q. mod_security supplies an array of request filtering and other security features to the Apache HTTP Server. How do I install mod_security under FreeBSD operating systems?

A. ModSecurity is an open source web application firewall that runs as an Apache module, and version 2.0 offers many new features and improvements.

It provides protection from a range of attacks against web applications and allows for HTTP traffic monitoring and real-time analysis with no changes to existing infrastructure. Some of the features include:

=> Parallel text matching
=> Geo IP resolution
=> Credit card number detection
=> Support for content injection
=> Automated rule updates
=> scripting as well as many others.

FreeBSD install mod_security

Type the following command to update ports tree:

```
# portsnap fetch update
```

Under FreeBSD 7, mod_security can be installed by typing the following commands:

```
# cd /usr/ports/www/mod_security
# make install clean
```

Configure mod_security

The modsecurity 2 Core Rules have been installed in `/usr/local/etc/apache22/Includes/mod_security2/`

By default it run in “DetectionOnly” mode as not to disturb operatings of working websites and Apache. First change directory to /usr/local/etc/apache22/Includes/mod_security2/:

```
# cd /usr/local/etc/apache22/Includes/mod_security2/
```

Now, open the ModSecuirty core rule set file – modsecurity_crs_10_config.conf, enter:

```
# vi modsecurity_crs_10_config.conf
```

The file is well documented so just customize it according to your requirements. Open httpd.conf file located at /usr/local/etc/apache22 and make sure following line exists:

```
LoadFile /usr/local/lib/libxml2.so
LoadModule security2_module libexec/apache22/mod_security2.so
```

Finally, restart the apache:

```
# /usr/local/etc/rc.d/apache22 restart
```

Monitoring mod_security log files

By default logs are written to following two files:

- /var/log/httpd-modsec2_audit.log
- /var/log/httpd-modsec2_debug.log
- /var/log/httpd-error.log or virtual domain error.log file

You can detect attacks by viewing these two files using grep or tail:

```
tail -f /var/log/httpd-modsec2_audit.log
grep cmd.exe /var/log/httpd-modsec2_audit.log
tail -f /home/httpd/example.com/logs/error.log
```

Once everything started to working perfectly open modsecurity_crs_10_config.conf file and set SecRuleEngine to On:

```
SecRuleEngine On
```

Restart apache:

```
# /usr/local/etc/rc.d/apache22 restart
```

https://www.cyberciti.biz/faq/freebsd-install-configure-mod_security/

FreeBSD and Apache ModSecurity

Apache ModSecurity:

ModSecurity performs real-time web application monitoring, logging and access control. It can also reduce Apache's attack surface by narrowing down HTTP features the server is willing to accept.

Installing ModSecurity:

```
root@bsd220:/ # pkg install ap24-mod_security
```

Configure ap24-mod_security:

To enable mod_security in Apache edit the following file: /usr/local/etc/apache24/modules.d/280_mod_security.conf

```
vim: set filetype=apache:
##
## module file for mod_security
##
## PROVIDE: mod_security2
## REQUIRE: mod_unique_id

##
## To enable ModSecurity in Apache, enable the modules
##  mod_unique_id (in httpd.conf) and
##  mod_security2 in this config file
##
## Additionally, load configuration and rules with an Include line from
##  /usr/local/etc/modsecurity/*.conf
##
## Most users will use the signatures from the OWASP Core Rule Set (CRS).
## For configuration instructions, see
/usr/local/share/doc/mod_security2/README.
##

## apache modules for mod_security
LoadModule unique_id_module libexec/apache24/mod_unique_id.so
LoadModule security2_module libexec/apache24/mod_security2.so
Include /usr/local/etc/modsecurity/*.conf
```

Restart Apache:

```
root@bsd220:/ # apachectl restart

Performing sanity check on apache24 configuration:
Syntax OK
Stopping apache24.
Waiting for PIDS: 10082.
Performing sanity check on apache24 configuration:
Syntax OK
Starting apache24.
root@bsd220:/ #
```

Verify ModSecurity is loaded by checking the Apache error log file:

```
root@bsd220:/ # tail -f /var/log/http-error.log

ModSecurity for Apache/2.9.1 (http://www.modsecurity.org/) configured.
ModSecurity: APR compiled version="1.5.2"; loaded version="1.5.2"
ModSecurity: PCRE compiled version="8.39 "; loaded version="8.39 2016-06-14"
ModSecurity: YAJL compiled version="2.1.0"
ModSecurity: LIBXML compiled version="2.9.4"
```

Configuring the Core Rule Set:

ModSecurity requires firewall rule definitions. Most people use the OWASP ModSecurity Core Rule Set (CRS). The easiest way to track the OWASP CRS repository right now is to use Git. Let’s make a directory for all our ModSecurity related stuff, and clone the CRS repository under it.

First, install devel/git, clone the CRS Repository, and then copy the crs.conf file:

```
pkg install git
cd /usr/local/etc/modsecurity
git clone https://github.com/SpiderLabs/owasp-modsecurity-crs

cp owasp-modsecurity crs/modsecurity_crs_10_setup.conf.example \ crs.conf
```

Activate CRS Base Rules by adding the following to /usr/local/etc/apache24/modules.d/280_mod_security.conf:

```
vim: set filetype=apache:
##
## module file for mod_security
##
## PROVIDE: mod_security2
## REQUIRE: mod_unique_id

##
## To enable ModSecurity in Apache, enable the modules
##  mod_unique_id (in httpd.conf) and
##  mod_security2 in this config file
##
## Additionally, load configuration and rules with an Include line from
##  /usr/local/etc/modsecurity/*.conf
##
## Most users will use the signatures from the OWASP Core Rule Set (CRS).
## For configuration instructions, see
/usr/local/share/doc/mod_security2/README.
##

## apache modules for mod_security
LoadModule unique_id_module libexec/apache24/mod_unique_id.so
LoadModule security2_module libexec/apache24/mod_security2.so
Include /usr/local/etc/modsecurity/*.conf
Include /usr/local/etc/modsecurity/owasp-modsecurity-crs/base_rules/*.conf
```

Configuring Blocking Mode:

To enable ModSecurity blocking mode edit the following file /usr/local/etc/modsecurity/modsecurity.conf and change the following:

```
SecRuleEngine On
```

Next, restart Apache:

```
root@bsd220:/ # apachectl restart
```

At this point, it is wise to check the functionality of your web sites to ensure ModSecurity is not preventing any legitimate actions. If there are any false positives, custom exceptions/rules written can mitigate the offending behavior.

Additionally, monitor https-error.log and modsec_audit.log in /var/log to view any potential false positives.

Log Rotation:

As with any service running on a server, it's imperative to monitor the log files. Also, proper log file rotation is essential to managing the amount of data collected. FreeBSD's log manager – newsyslog is designed to rotate various log files for the operating systems. Newsyslog allows one to include a file to pull in additional entries. Please review the man page for newsyslog(8).

First, create the directory /usr/local/etc/newsyslog.conf.d/:

```
root@bsd220:/ # mkdir /usr/local/etc/newsyslog.conf.d
```

Create /usr/local/etc/newsyslog.conf.d/httpdlog.conf

```
root@bsd220:/  # touch /usr/local/etc/newsyslog.conf.d/httpdlog.conf
```

Add the following to /usr/local/etc/newsyslog.conf.d/httpdlog.conf:

```
/var/log/httpd-access.log               600  7     *    @T12   B
/var/run/httpd.pid 30

/var/log/httpd-error.log                600  7     *    @T12   B
/var/run/httpd.pid 30

/var/log/httpd-ssl_request.log          600  7     *    @T12   B
/var/run/httpd.pid 30

/var/log/modsec_audit.log               600  7     *    @T12   B
/var/run/httpd.pid 30
```

Next, restart newsyslog:

```
root@bsd220:/ # service newsyslog restart
```

Maintenance:

You probably want to keep the CRS updated from time to time. You can do this with Git:

```
root@bsd220: / # cd /usr/local/etc/modsecurity/owasp-modsecurity-crs
root@bsd220: / # git pull
root@bsd220: / # apachectl restart
```

Summary:

ModSecurity is a powerful application layer firewall for the venerable Apache web server. It provides real-time application security monitoring and access control as well as full HTTP traffic logging. With the OWASP ModSecurity Core Rule Set (CRS), it is ready to use right out-of-the-box. CRS also provides default rule sets for both Drupal and WordPress which comes in quite handy for site using these popular frameworks. Placing ModSecurity in monitoring mode is an excellent resource for System Administrators to monitor traffic and make corrective actions.

References:

ModSecurity's Web Site

OWASP ModSecurity Core Rule Set

https://loga.us/2017/02/22/freebsd-apache-modsecurity/

https://www.freshports.org/www/mod_security

Mod_Security on FreeBSD

Mod_Security is a application level firewall that helps to defend againist common website attacks. We are demonstrating how to install and do a basic configuration of Mod Security on FreeBSD with the Apache webserver.

Before we begin, these are the software versions and locations used throughout the installation tutorial.

- Mod Security Version: 2.2.6
- Mod Security rule set 2.5.25
- Rule Folder Location: /usr/local/etc/apache22/Includes/modsecurity2

Step 1: Install the mod_security port
Using portmaster: portmaster www/modsecurity2
The portmaster installation will install and configure the needed modules for Apache and provide you with a default mod_security.conf file.

Step 2: Download ruleset
Mod_security rules come in two varieties. The core rules set, as provided by OWASP, and the commercial rules sets are provided by Trustwave SpiderLabs. Here, we are going to concentrate on the core rules set. For commercial support and additional rules, we recomend that you contact Trustwave SpiderLabs directly.

The latest copy of the rules are stored in Github at https://github.com/SpiderLabs/owasp-modsecurity-crs . Additional information as well as links to direct downloads can be found at https://www.owasp.org/index.php/Category:OWASP_ModSecurity_Core_Rule_Set...

While following this tutorial, use ruleset 2.5.5. The downloaded rules should be placed in /usr/local/etc/apache22/Includes/modsecurity2 folder.

Step 3: Configure modsecurity.conf
A sample modsecurity.conf file is available at /usr/local/etc/modsecurity.conf-example. This file controls the core aspects of how mod _security works. By default, it has mod_Security set to "detection only" mode. The following file system locations should be defined.

1. SecTmpDir
2. SecDataDir
3. SecUploadDir

Copy the file to your rule location. You can leave the rest of the file at default for now, but it is important that the entire file is read and configured before activating blocking mode.

Step 4: Configure modsecurity_crs_10_setup.conf
The mod security ruleset download has an example file: modsecurity_crs_10_setup.conf.example file. Copy the file to modsecurity_crs_10_setup.conf and edit according to your needs.

Step 5: Configure Apache
The Apache webserver will need to be configured in order to load the mod_security modules and define additional configuration. Edit the httpd.conf file and add the following:

```
LoadModule security2_module libexec/apache22/mod_security2.so
<IfModule security2_module>
Include /usr/local/etc/apache22/Includes/modsecurity2/*.conf
Include /usr/local/etc/apache22/Includes/modsecurity2/activated_rules/*.conf
</IfModule>
```

Step 6: Activate Rules
In order to activate the rules, they will need to be copied or symlinked from the 'base_rules' folder to the 'activated_rules' folder. Notice that some rules require other rules in order to work properly. It is recommended that you read each rule carefully before activating it.

You will need to restart the Apache webserver each time you change or add rules.

Step 7: Confirm that mod_security is working
There are two log files that will shed light on the operation of Mod_Security.

- /var/log/modsec_audit.log
- /var/log/httpd-error.log

An example entry from httpd-error.log:
[Mon Mar 25 16:31:27 2013] [error] [client <client IP>] ModSecurity: Access denied with code 403 (phase 2). Match of "within %{tx.allowed_methods}" against "REQUEST_METHOD" required. [file "/usr/local/etc/apache22/Includes/modsecurity2/activated_rules/modsecurity_crs_30_http_policy.conf"] [line "31"] [id "960032"] [msg "Method is not allowed by policy"] [data "\\x16\\x03\\x01"] [severity "CRITICAL"] [tag "POLICY/METHOD_NOT_ALLOWED"] [tag

"WASCTC/WASC-15"] [tag "OWASP_TOP_10/A6"] [tag "OWASP_AppSensor/RE1"] [tag "PCI/12.1"] [hostname] [uri "/"] [unique_id "UVB739gRKvoAAJFHBHMAAADS"]

An example entry from modsec_audit.log:
Message: Access denied with code 403 (phase 2). Pattern match "(?:\\b(?:(?:n(?:et(?:\\b\\W+?\\blocalgroup|\\.exe)|(?:map|c)\\.exe)|t(?:racer(?:oute|t)|elnet\\.exe|clsh8?|ftp)|(?:w(?:guest|sh)|rcmd|ftp)\\.exe|echo\\b\\W*?\\by+)\\b|c(?:md(?:(?:\\.exe|32)\\b|\\b\\W*?\\/c)|d(?:\\b\\W*?[\\/]|\\W*?\\.\\.)|hmod.{0,40}?\\ ..." at ARGS:_message. [file "/usr/local/etc/apache22/Includes/modsecurity2/activated_rules/modsecurity_crs_40_generic_attacks.conf"] [line "197"] [id "950006"] [rev "2.2.5"] [msg "System Command Injection"] [data "chmod +x"] [severity "CRITICAL"] [tag "WEB_ATTACK/COMMAND_INJECTION"] [tag "WASCTC/WASC-31"] [tag "OWASP_TOP_10/A1"] [tag "PCI/6.5.2"]

Action: Intercepted (phase 2)
Stopwatch: 1364229727373433 2062 (- - -)
Stopwatch2: 1364229727373433 2062; combined=1543, p1=125, p2=1400, p3=0, p4=0, p5=17, sr=16, sw=1, l=0, gc=0
Response-Body-Transformed: Dechunked
Producer: ModSecurity for Apache/2.6.6 (http://www.modsecurity.org/); OWASP_CRS/2.2.5.
Server: Apache

Note: The [id "number"] is the exact rule id that is being triggered. For example, [id "960032"] or [id "950006"]. This can be used to whitelist websites from being affected.

Creating a rule exemption per website using virtual hosts
A exemption may need to be provided to a website from a specfic rule. This can be done by editing the vitual host definition and adding the SecRuleRemoveById <number>.

Virtual host example:

```
<Directory "/path/to/website/documentroot">
        SecRuleRemoveById 960032
</Directory>
```

Activating Blocking mode
In order for mod_security to effectively stop attacks, it will need to be placed into "blocking" mode. This can be done by editing the mod_security.conf file and changing the following.

Remove: - SecRuleEngine DetectionOnly
Add: SecRuleEngine on

https://www.knthost.com/apache/modsecurity-freebsd