Apache mod_security

This is a module to enhance security in your apache web server, and therefore, enhancing the protection of tiki sites installed on apache against fairly common external attacks, based on some rules.

You need to invest some little time after its first installation in order to fine tune the rules to reduce the number of "false positives" detected.

- I.e., some actions from users in tiki sites which might raise an alert or trigger a defense action (blocking that page with an error code 500) by means of mod_security, because some text was written by users in some language that mod_security didn't expect.
- Example: "Měšťáček" (means something like "small town little simple man" in Czech language) use several accented characters in the same word, and mod_security might filter and block them with its default rules.

Installation

Version used in this description: modsecurity-apache_2.6.2

- Download the archive and place it in your work folder, unzip and untar it.
- Read the documentation for prerequisites and install the required bits, e.g. liblua5.1.so and libxml2.so.
- change to the folder and compile mod_security:

```
mkdir -p /work
cd /work
wget http://www.modsecurity.org/download/modsecurity-apache_2.6.2.tar.gz
gzip -d modsecurity-apache_2.6.2.tar.gz
tar xvf modsecurity-apache_2.6.2.tar
cd modsecurity-apache_2.6.2
./configure
make
make test
make install
```

Get the rules required by mod_security and unzip them. You can always get the latest rules by this, just make sure the changes you maybe do to the rules files when installed don't get lost on updating.

```
 cd tools
./rules-updater.pl -rhttp://www.modsecurity.org/autoupdate/repository/ -prules -Smodsecurity-crs
cd rules/modsecurity-crs
unzip *.zip
```

Copy the needed folders to the apache config folder.

```
mkdir -p /etc/apache2/mod_security
cp -R activated_rules /etc/apache2/mod_security/
cp -R base_rules /etc/apache2/mod_security/
```
Configuration

Now tell apache to use the module and configure it accordingly by editing httpd.conf.

```
<IfModule security2_module>
    Include /etc/apache2/mod_security/modsecurity_crs_10_config.conf
    Include /etc/apache2/mod_security/activated_rules/*.conf
</IfModule>
```

Create a required folder for mod_security working data.

```
mkdir -p /var/log/apache2/secdatadir
```

Configure mod_security and run it in test mode.

```
[...]
# enable test mode, logging only, NO blocking
SecRuleEngine DetectionOnly
# uncomment THIS line instead to fully enable mod_security
# SecRuleEngine On

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

# Log everything we know about a transaction.
SecAuditLogParts ABJDEHZ

SecAuditLogType Serial
SecAuditLog /var/log/apache2/modsec_audit.log
SecDataDir /var/log/apache2/secdatadir

# default action:
# * sleep 3000 milliseconds (3 seconds) to slow down brute force attacks
# * deny access to page (error 403)
```
# do not log to apache log
# but log to the audit log configured above
SecDefaultAction "phase:2,pause:3000,deny,nolog,auditlog"

Create links of all rule files you want to use into activated_rules/.

Rules are to be found in:

- /etc/apache2/mod_security/base_rules
- /etc/apache2/mod_security/experimental_rules
- /etc/apache2/mod_security/lua
- /etc/apache2/mod_security/optional_rules
- /etc/apache2/mod_security/slr_rules

Example:

```bash
cd /etc/apache2/mod_security/activated_rules/
ln -s ..../base_rules/modsecurity_crs_35_bad_robots.conf .
```

If there are datafile for special rules, link them, too:

```bash
ln -s ..../base_rules/modsecurity_35_bad_robots.data .
```

Create a custom error page

```
//forbidden/403.php
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en" id="page_1210">
<head><title>Tiki: Error 403</title></head>
<body style="margin: 3em 6em;">
<h1>Oops, sorry!</h1>
Your request got filtered out due to possible security issues.<br /><br />
1. You tried to access a page you are not allowed to.<br /><br />
2. One or more things in your request were suspicious (defective request header, invalid cookies, bad parameters, ...).

If you think you did nothing wrong<br /><br />
- try again with a different browser<br />
- avoid any evil characters inside the request url
```

All set up. Now test your config and then tell apache to reload config.

`phpinfo()` should tell you that `mod_security` is enabled and a new logfile should be available: `/var/log/apache2/modsec_audit.log`

Example error log entry of a blocked 'bad robot':

```
--25f26844-A--
[03/Nov/2011:12:25:35 +0100] Trj6LLwoNcsAABrw@7wAAAAh 41.82.184.111 2521 188.40.53.203 80
--25f26844-B--
GET / HTTP/1.1
Accept: text/html
Cache-Control: no-cache, no-cache
Proxy-Connection: Keep-Alive
User-Agent: Mozilla/4.0 (compatible; MSIE 5.0; Windows NT; DigExt; DTS Agent
Pragma: no-cache
Connection: close
Host: tiki.org

--25f26844-F--
HTTP/1.1 403 Forbidden
Vary: Accept-Encoding
Content-Length: 1677
Connection: close
Content-Type: text/html; charset=utf-8

--25f26844-H--
Message: Access denied with code 403 (phase 2). [file "modsecurity_crs_35_bad_robots.conf"] [line "27"] [id "990012"] [rev "2.2.2"] [msg "Rogue web site crawler"] [data "DTS Agent"] [severity "WARNING"] [tag "AUTOMATION/MALICIOUS"] [tag "WASCTC/WASC-21"] [tag "OWASP_TOP_10/A7"] [tag "PCI/6.5.10"]
Action: Intercepted (phase 2)
Stopwatch: 1320319532916353 2998180 (- - -)
Stopwatch2: 1320319532916353 2998180; combined=710, p1=232, p2=443, p3=0, p4=0, p5=34,
```
Now let it run for some time and then check for false positives. If you found one, grab the rule id from the log (here it is: id "990012") and the filename of the rule file: file %22modsecurity_crs_35_bad_robots.conf%22. Go to the rulefile and adjust the rule to exclude the false positive or disable the rule completely by commenting it out with a leading # character.

If you think the rules are good, adjust /etc/apache2/mod_security/modsecurity_crs_10_config.conf from

SecRuleEngine DetectionOnly

to

SecRuleEngine On

and let apache reload the configuration:

/etc/init.d/apache2 reload