mod_security...

...the web application firewall

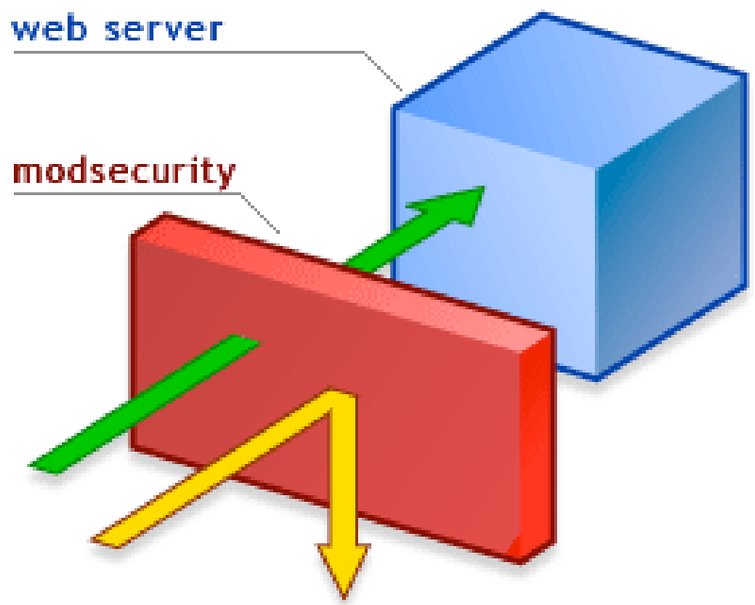
State

- **State of the “realworld”**
  - The world is going Web, companies must open their systems to their customers and partners.
  - Port 80 and 443(https) are used for everything now.
  - Web applications, web services.
  - Classic firewall architectures do not help any more.
  - Web development is rather often a mess.
  - Web applications are not secure.
  - Users want features; security is an afterthought.
  - Web servers do not provide the correct tools (e.g. auditing).
  - The awareness is rising but we have a long way to go...

Frédéric LAPLAGNE : f.laplagne@akersia.fr

mod_security : History and implementation

- **History**
  - Develop by Ivan Ristic and Thinking Stone (commercial support)
  - Begin of the project 2002
  - Last Stable Version : 1.8.7 (05.03.2005)



- **Implementation**
  - **Embed Into Web Server**
    - Inexpensive and easy since no changes to the network are required.
    - But works only for one web server.
    - No practical impact on performance.
  - **Apache-based Web Application Firewall**
    - It is a reverse proxy.
    - Easy to install and configure.
    - Created out of default and third-party modules:

```
mod_proxy       mod_proxy_html
mod_rewrite     mod_head
mod_security
```


Frédéric LAPLAGNE : f.laplagne@akersia.fr

mod_security : Features

- **Features**

- Audit logging.
- Provides access to any part of the request (request body included) and the response.
- Flexible regular expression-based rule engine.
- Rules can be combined.
- External logic can be invoked.
- Supports unlimited number of different policies (per virtual host, folder, even a single file).
- Supports file upload interception and real-time validation.
- Anti-evasion built in.
- Encoding validation built in.
- Buffer overflow protection.

Frédéric LAPLAGNE : f.laplagne@akersia.fr

mod_security : History and functionalities

- **Functionalities**
  - Intercepts HTTP(S) requests before they are fully processed by the web server
  - Intercepts the request body (e.g., the POST payload)
  - Intercepts, stores, and optionally validates uploaded files
  - Performs anti-evasion actions automatically
  - Performs request analysis by processing a set of rules defined in the configuration
  - Intercepts HTTP(S) responses before they are sent back to the client (Apache 2 only)
  - Takes one of the predefined actions or executes an external script when a request or a response fails analysis
  - Apache 2 advanced filtering API use by mod_security (more efficient in terms of memory consumption)
  - Possibility to “plug” modsecurity rules with third security software (SQUID ...).

mod_security : Compile

- **Compile**

  - Module mode : need to compile with apxs apache tool (`apxs -cia mod_security.c`)

  - Static compilation :

    1. Copy the file `mod_security.c` to `/src/modules/extra`
    2. Configure Apache distribution with two additional configuration options:

```
--activate-module=src/modules/extra/mod_security
--enable-module=securiy
```

  - The start or restart apache session.

Frédéric LAPLAGNE : f.laplagne@akersia.fr

mod_security : Basic configuration #1

- **Main configuration #1:**

```
# Enable mod_security
SecFilterEngine On

# Retrieve request payload
SecFilterScanPOST On

# Server identity masking
SecServerSignature " "

# Reasonable automatic validation defaults
SecFilterCheckURLEncoding On
SecFilterCheckUnicodeEncoding Off
```

- Enable mod_security, no process will be performed unless the module is explicity enable.
- Intercept resquest body (POST_PAYLOAD).
- Change the identity of the Apache web server you would have to go into the source code.
- Prevent bad URL encoding, it will reject invalid or missing hexadecimal numbers (eg. % XV). CheckUnicodeEncoding can detect three types of problems : invalid characters, missing bytes, and overlong characters.

mod_security : Basic configuration #2

- **Main configuration #2:**

```
# Accept almost all byte values
SecFilterForceByteRange 32 126
```

```
# Reject invalid requests with status 403
SecFilterDefaultAction deny,log,status:403
```

```
# Only record the relevant information
SecAuditEngine RelevantOnly
SecAuditLog /var/www/logs/audit_log
```

```
# Debug level
SecFilterDebugLog /var/logs/modsec_debug_log
SecFilterDebugLevel 0
```

- Force requests to consist only of bytes from a certain byte range. This can be useful to avoid stack overflow attacks. (for latin language we advise 32 168).

- Configures the default action list to handle invalid requests.

- Relevant requests are those requests that caused a filter match, and log path.

- Control datail and locate the debug log.

mod_security : Rules #1

- **SecFilter**

  - The **`SecFilter`** directive performs a broad search against the request parameters, as well as against the request body for POST requests:

    `SecFilter` ***`KEYWORD`***

  - If the keyword is detected, the rule will be triggered and will cause the default action list to be executed.

    e.g. **`SecFilter /tmp/`** will block **`http://www.site.net/tmp/script.sh`**

Regular expression :

- Wanted pattern "`1.1`" use `^1\.1$` else could be interpreted `101` or `1001.100`
- Pattern `!attack` causes a rule match if the searched string does not contain the pattern `attack`.

Frédéric LAPLAGNE : f.laplagne@akersia.fr

mod_security : Rules #2

- **SecFilterSelective**

  - The **`SecFilterSelective`** permit to design rules to certain parts of HTTP requests :

    `SecFilterSelective QUERY_STRING` ***`KEYWORD`***

  - With a selective rule you can examin more than one field at a time, you can separat multiple variable names with a pipe.

    e.g. **`SecFilterSelective ARG_authorized|COOKIE_authorized`** ***`KEYWORD`***

  - Exemple bellow : Look the ***`keyword`*** in teh parameter "**`authorized`**" and in the cookie "**`authorized`**".

Some Standard rule variables :

`REMOTE_ADDR`, **`REMOTE_HOST`**, **`REMOTE_USER`**, **`REQUEST_METHOD`** (e.g., `GET`, `POST`), **`QUERY_STRING`**, **`AUTH_TYPE`**, **`SERVER_PROTOCOL`**, **`THE_REQUEST`** (e.g., `GET /view.php?id=5 HTTP/1.0`), **`POST_PAYLOAD`**, **`COOKIES_VALUES`** ...

Frédéric LAPLAGNE : f.laplagne@akersia.fr

mod_security : Actions #1

- **SecFilterDefaultAction**
  - The **`SecFilterDefaultAction`** determines the default action list :

    e.g. **`SecFilterDefaultAction deny,log,status:403`**

  - In bellow's case : reject invalid requests with status 403, and we log it.

- **SecFilter : Override default action**
  - It is possible to override the default action list by supplying a list of actions to individual rules as the last (optional) parameter:

    e.g. **`SecFilter KEYWORD log,pass`**

  - In bellow's case : only log a warning message when the KEYWORD is found.

Frédéric LAPLAGNE : f.laplagne@akersia.fr

mod_security : Actions #2

- **Useful actions #1:**

  - **`allow`** Skip over the remaining rules and allow the request to be processed.
  - **`chain`** Chain the current rule with the one that follows. Process the next rule if the current rule matches (like logical AND).
  - **`deny`** Deny request processing..
  - **`id:`*`n`*** Assign a unique ID ***n*** to the rule. The ID will appear in the log. Useful when there are many rules designed to handle the same problem.
  - **`log`** Log the rule match. A message will go into the Apache error log and into the audit log.

Frédéric LAPLAGNE : f.laplagne@akersia.fr

mod_security : Actions #3

- **Useful actions #2:**

  - **`nolog`** Do not log the rule match.
  - **`pass`** Proceed to the next rule in spite of the current rule match.
  - **`redirect:`*`url`*** Redirection to the address specified by ***`url`*** when a request is denied.
  - **`skipnext:`*`n`*** On rule match skip the next ***`n`*** rules.
  - **`status:`*`n`*** Configure the status n to be used to deny the request.
  - **`exec:`*`filename`*** Execute the external script specified by ***`filename`*** on rule match.

Frédéric LAPLAGNE : f.laplagne@akersia.fr

mod_security : Starting rules

- **Enforce strict HTTP usage:**
  - Accept only valid protocol versions, helps fight HTTP fingerprinting

```
SecFilterSelective SERVER_PROTOCOL !^HTTP/(0\.9|1\.0|1\.1)$
```

  - Allow supported request methods only

```
SecFilterSelective REQUEST_METHOD !^(GET|HEAD|POST)$
```

  - Require HTTP_USER_AGENT and HTTP_HOST, no telnet use.

```
SecFilterSelective "HTTP_USER_AGENT|HTTP_HOST" "^$"
```

  - Require Content-Length to be provided with every POST request.

```
SecFilterSelective REQUEST_METHOD ^POST$ chain
SecFilterSelective HTTP_Content-Length ^$
```

- **Override rule for admin:**
  - Permit the Admin IP to skip the warning checks for all requests coming from it.

```
SecFilterSelective REMOTE_ADDR ^192.168.1.1$ allow
```


Frédéric LAPLAGNE : f.laplagne@akersia.fr

mod_security : Detecting Common Attacks #1

- **Database attacks #1:**

  - Database attacks are executed an SQL query or a part of it into request parameter.

Query	mod_security pattern (SecFilter)
`DELETE FROM users`	`SecFilter delete[[:space:]]+from`
`INSERT INTO users VALUES (1, 'admin')`	`SecFilter drop[[:space:]]+table`
`CREATE TABLE newusers`	`SecFilter create[[::space:]]+table`
`UPDATE users SET balance = 1000`	`SecFilter update.+set.+=`
`INSERT INTO users VALUES (1, 'admin')`	`SecFilter insert[[:space:]]+into.+values`
`SELECT username, balance FROM users`	`SecFilter select.+from`

  - Enhanced rules :

  - Attempt to modify the original query to always be true:

    Query : `SELECT * FROM users WHERE username = 'admin' and password = 'xxx' OR 1=1--'`

    mod_security rule: `SecFilter or.+1[[:space:]]*= [[:space:]]1`

mod_security : Detecting Common Attacks #2

- **Database attacks #2:**
  - Database attacks for MSSQL and MySQL

Attacks	mod_security pattern (SecFilter)	Type
`EXEC xp_cmdshell.`	`SecFilter exec.+xp_`	mssql
`EXEC sp_who.`	`SecFilter exec.+sp_`	mssql
`SELECT @@version.`	`SecFilter @@[[:alnum:]]+`	mssql
`SELECT * FROM '/tmp/users'.`	`SecFilter into[[:space:]]+outfile`	mysql
`LOAD DATA INFILE '/tmp/users' INTO TABLE users.`	`SecFilter load[[:space:]]+data`	mssql

mod_security : Detecting Common Attacks #3

- **Cross-site scripting (XSS) attacks :**
  - Definition : “Exploit where information from one context, where it is not trusted, can be inserted into another context, where it is.” (Wikipedia).

e.g. **`<object>...</object>`** Executes component when page is loaded (IE only).

e.g. **`<script>...</script> OR <script src="..."> </script>`** Executes code when page is loaded

e.g. **`<img src="javascript:...">`** Executes code when page is loaded (javascript)

  - Difficult to setup initial rules, because XSS have lot variant forms.

e.g. : **`<img src="javas&#99;ript:...">`**, **`<img src="javascript:...">`**

  - Basic XSS rules :

```
SecFilter "<( |\n)*script"
SecFilter "<(.|\n)+>"
SecFilter "<[[:space:]]*script"
```

... but must be redifine for your hown needs and obligations.

Frédéric LAPLAGNE : f.laplagne@akersia.fr

mod_security : Detecting Common Attacks #4

- **Command execution and file disclosure Unix :**
  - Prevent any query to system file or command access.

Description	mod_security pattern (SecFilter)
Common Unix commands	`SecFilter (uname|id|ls|cat|rm|kill|mail|su)`
Fragments of common Unix system path	`SecFilter (/home/|/var/|/boot/|/etc/|/bin/|/usr/|/tmp/)`
Directory backreference commonly used as part of file disclosure attacks (../../).	`SecFilter "\.\./"`

False positive things :

If **`SecFilter bin/`**
It will match also : **`http://www.site.net/cgi-bin/innocent.cgi`**

mod_security : Detecting Common Attacks #5

- **Command execution and file disclosure Windows :**
  - Prevent any query to system file or command access.

Description	mod_security pattern (SecFilter)
Common Windows extensions	`SecFilter "(\.cmd|\.bat|\.htw|\.ida|\.idq|\.htr|\.idc|\.printer|\.ini|\.pol|\.dat|\.cfg|\.idx|\.dll|\.inf|\.mdb|\.mde|\.msi|\.reg|\.scr|\.exe)"`
Fragments of common Windows system path	`SecFilter (c\:|/_vti_bin/|/_vti_cnf/|_vti_pvt/|/IISSAMPLES/|/MSOffice/|/system32/|/msadc/|/inetpub/|/winnt)`

Frédéric LAPLAGNE : f.laplagne@akersia.fr

mod_security : Complex Configuration #1

- **Add special rule for a location :**

- The mod_security configuration data can be placed into any Apache context.

e.g.:

```
SecFilterSelective ARG_a KEYWORD
<Location /test/>
    SecFilterSelective ARG_b KEYWORD
</Location>
```

e.g. bellow: The parent configuration will have only parameter “**a**” tested, while the requests that fall in the **/test/** location will have “**a**” and “**b**” tested.

mod_security : Complex Configuration #2

- **Replace rule in a location**

- The mod_security configuration data can be placed into any Apache context.

e.g.:

```
SecFilterSelective ARG_a KEYWORD
<Location /test/>
   SecFilterInheritance Off
   SecFilterSelective ARG_b KEYWORD
</Location>
```

e.g. bellow: Requests for the parent configuration will have only parameter "**a**" tested, while the requests that fall in the **/test/** location will have only parameter "**b**" tested

mod_security : Positive security model protection #1

- **Secure administration actions :**

  - mod_security permit to secure important administration operation.

```
<Location /user_view.php>
    # This script only accepts GET
    SecFilterSelective REQUEST_METHOD !^GET$
    # Accept only one parameter: id
    SecFilterSelective ARGS_NAMES !^id$
    # Parameter id is mandatory, and it must be
    # a number, 4-14 digits long
    SecFilterSelective ARG_id !^[[:digit:]]{4,14}$
</Location>
```


Frédéric LAPLAGNE : f.laplagne@akersia.fr

mod_security : Positive security model protection #2

- **Secure administration actions :**

  - mod_security permit to secure important administration operation.

```
<Location /user_add.php>
    # This script only accepts POST
    SecFilterSelective REQUEST_METHOD !^POST$
    # Accept three parameters: firstname, lastname, and email
    SecFilterSelective ARGS_NAMES !^(firstname|lastname|email)$
    # Parameter firstname is mandatory, and it must
    # contain text 1-64 characters long
    SecFilterSelective ARG_firstname !^[[:alnum:][:space:]]{1,64}$
    # Parameter lastname is mandatory, and it must
    # contain text 1-64 characters long
    SecFilterSelective ARG_lastname !^[ [:alnum:][:space:]]{1,64}$
    # Parameter email is optional, but if it is present
    # it must consist only of characters that are
    # allowed in an email address
    SecFilterSelective ARG_email !(^$|^[[:alnum:].@]{1,64}$)
</Location>
```

mod_security : Performance

- **Speed :**

  - Not significative speed difference.
  - Could cost 10% of apache speed in some case (only dynamic scan).

- **Memory consumption :**

  - mod_security stores it in memory. In most
  - Is not a big deal since most requests are small.
  - It can be a problem for parts of the web site where files are being uploaded.
  - Apache 2 you can modify **`SecUploadInMemoryLimit`** (default 64KB) to custom the buffer need by mod_security to scan upload files.

Frédéric LAPLAGNE : f.laplagne@akersia.fr

mod_security : informations sources

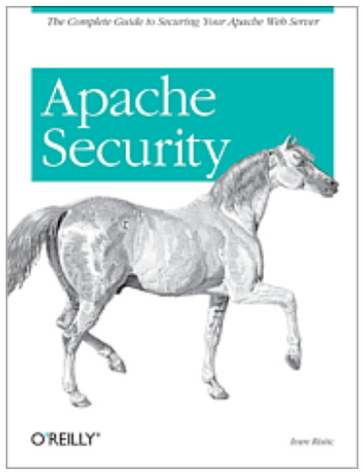
- **Web Sites**

  - **www.modsecurity.org**

    • Modsecurity home site, we can find useful information, software upgrade and online documentation.

  - **www.thinkingstone.com**

    • Corporate Web Site for modsecurity support.



- **Literature**

  - O'Reilly "**Apache Security**" (march 2005) ISBN: 0596007248 :

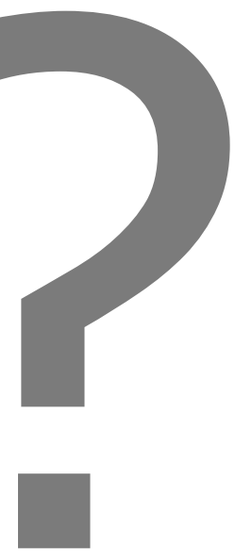
    • This guide, written by Ivan Ristic, arms readers with all the information they need to securely deplay WEB applications.

    • Topics covered include installation, server sharing, logging and monitoring, web applications, PHP and SSL/TLS, and more.

Most of those informations sources permit to compose this presentation.

Questions ?

Preguntas ?



Question ?

しつまん

質問

Frédéric LAPLAGNE : f.laplagne@akersia.fr

Merci !

Gracias !

Gràcies !

ありがとう

Thank you !