Securing Web Applications with ModSecurity

Emmanuel Bergmans
Sep, 2005
tech@i-logs.com

OWASP

The OWASP Foundation
http://www.owasp.org

Agenda

- Problem: web applications are not secure
- Web application firewall
- What is ModSecurity
- Total cost of ownership
  - (ModSecurity versus Commercial solution)
- Questions

Problem: web applications are not secure

- Everybody wants to be web developer
- Customers want features, focus is not set on security
- Web application security is young
- Development cycles of web applications are very short
- Lack of knowledge
- Easy access to the web applications (web browser)
- …

Problem: web applications are not secure (2)

- Things are changing but it is not possible nor feasible to achieve 100% security
- Intrusion is always possible

=>one of the solutions to increase security is to use a web application firewall

Web application firewall

- IDS/Firewall designed to understand HTTP protocol
- They can handle HTTPS traffic
- They are designed to make “intelligent” content filtering (including prevention)
- Selective policies according to url/website/…
- Two approaches
  - Network based
  - Web server based like ModSecurity

What is ModSecurity

- Concept
- Main features (stable version 1.8.7)
- Weakness
- Usefull products combination
- Product evolutions (devel version)
- SWOT analysis

Concept

- ModSecurity (http://www.modsecurity.org) is an open source intrusion, detection and prevention engine embedded into Apache webserver (as a module http://modules.apache.org/reference)
- ModSecurity has been written by Ivan Ristic
  - Author of the book « Apache Security » (http://www.apachesecurity.net/)
  - Founder of Thinking Stone, a web security company
  - He has made a presentation of ModSecurity at OWASP AppSec Europe (http://www.owasp.org/docroot/owasp/misc/OWASP_UK_2005_Presentations/AppSec2005-Ivan_Ristic-Web_Intrusion_Detection_w_ModSecurity.ppt)

Concept (2)

- As ModSecurity is embedded into Apache web server
  - You have access to any part of the request (including https, compressed files, …)
  - No practical impact on performance if you only activate ModSecurity for dynamic requests
  - No need to change network topology
  - But works only for one web server
  - But no information about compatibility with commercial modules (like the Zend Platform)

Concept (3)

- ModSecurity uses Apache features to propose different policies per container (Virtual Host/Location/File)
- ModSecurity is a rule-based Web IDS
  - Flexible rule system based on regular expressions
  - Rules may be related to any part of the HTTP request
  - Rules can be combined
- ModSecurity may act at 4 levels
  - Monitoring
  - Detection
  - Prevention
  - Auditing

Concept (4): Operation modes

- 3 kinds of operation modes:
  - Detect-only mode (detection/monitoring/auditing) – limitation: all implicit validations must be disabled (URL encoding check, unicode, cookie format, byte range)
  - Standard mode (detection/prevention/monitoring/auditing)
  - Quick fix for application vulnerabilities

Main features

- Request filtering
- Anti-evasion techniques
- Understanding the HTTP protocol
- POST payload analysis
- Script output analysis
- Audit logging
- HTTPS filtering
- Full integration with other Apache modules

Main Features (2) – inside the configuration

- Defining actions
  - You can define default actions list and actions per rules
  - Actions can be combined
  - Several built-in actions: pass, allow, deny, status, redirect, exec, log, nolog,skipnext, chain, pause
- Simple default action example
  - SecFilterDefaultAction "deny,log,status:500"

Main features (3) – inside the configuration

- Simple rule examples
  - Prevent SQL injection

SecFilter " DELETE[[:space:]]+FROM"

  - Prevent JavaScript injection

SecFilter "<script"

- Combined rules example
  - Restrict control panel access from a specific IP for user admin

SecFilterSelective ARG_login admin chain

SecFilterSelective REMOTE_ADDR " !^192.168.1.1$"

Main features (4) – tools and doc

- ModSecurity distribution also contains
  - A command line test tool
  - A ruleset converter utility: Snort to ModSecurity
  - A very good manual (see also http://www.modsecurity.org/documentation/ for all official versions of the documentation and external articles)

Features: Test tool

- Command line test tool included in the distribution
- 3 Components:
  - Server side, cgi-script (modsec-test.pl)
  - Test script (run-test.pl)
  - Test files

SQL injection test

- Test script content
  - GET /cgi-bin/modsec-test.pl?p=DELETE%20FRoM+users HTTP/1.0
- Launch the test script
  - run-test.pl 192.168.1.1 13-sql-injection.test

SQL Injection test

- Audit log trace

```
==================================
Request: 192.168.1.1 - - [25/Sep/2005:09:49:10 +0200] "GET /cgi-bin/modsec-test.pl?p=DELETE%20FRoM+users HTTP/1.0" 200 1048
Handler: cgi-script
----------------------------------------
GET /cgi-bin/modsec-test.pl?p=DELETE%20FRoM+users HTTP/1.0
Connection: Close
Host: test.test.be:80
User-Agent: mod_security regression test utility
mod_security-executed: /usr/local/bin/report-attack.pl

HTTP/1.0 200 OK
Connection: close
Content-Type: text/plain
==================================
```

SQL Injection test

- Mail alert
  - Subject:[MODSEC_ALERT] Report

  - Hostname: test.test.be
  - Date: 20050927 09:49:10

  - Alert message: Warning. Pattern match "delete[[:space:]]+from" at THE_REQUEST.
  - Attacker IP: 192.168.1.1
  - Virtual host: test.test.be:80
  - Requested URI: /cgi-bin/modsec-test.pl?p=DELETE%20FRoM+users
  - Request method: GET

  - All system ENV vars on alert:
  - DOCUMENT_ROOT=/dir/virtual/test
  - GATEWAY_INTERFACE=CGI/1.1
  - HTTP_CONNECTION=Close
  - HTTP_HOST=test.test.be:80
  - HTTP_MOD_SECURITY_ACTION=0
  - HTTP_MOD_SECURITY_EXECUTED=/usr/local//bin/report-attack.pl
  - HTTP_MOD_SECURITY_MESSAGE=Warning. Pattern match "delete[[:space:]]+from" at THE_REQUEST.
  - HTTP_USER_AGENT=mod_security regression test utility

SQL injection test

- PATH=/sbin:/bin:/usr/sbin:/usr/bin:/usr/local/sbin:/usr/local/bin:/usr/local/etc:/usr/X11R6/bin:/root/bin
- PATH_TRANSLATED=/usr/local/bin/report-attack.pl
- QUERY_STRING=p=DELETE%20FRoM+users
- REDIRECT_STATUS=302
- REMOTE_ADDR=192.168.1.1
- REMOTE_HOST=test.test.be
- REMOTE_PORT=2537
- REQUEST_METHOD=GET
- REQUEST_URI=/cgi-bin/modsec-test.pl?p=DELETE%20FRoM+users
- SCRIPT_FILENAME=/dir/virtual/cgi-bin/modsec-test.pl
- SCRIPT_NAME=/cgi-bin/modsec-test.pl
- SERVER_ADDR=192.168.1.1
- SERVER_ADMIN=webmaster@test.be
- SERVER_NAME=test.test.be
- SERVER_PORT=80
- SERVER_PROTOCOL=HTTP/1.0
- SERVER_SIGNATURE=
- SERVER_SOFTWARE=IIS/5.0

- Generated by report-attack.pl v0.1

Weakness

- ModSecurity offers most of the features of commercial solutions but:
  - No GUI
  - Only working with Apache web server
  - Requires good knowledge of Apache configuration
  - Requires good knowledge of regular expressions syntax
  - No out of the shelves solution for centralized logging and monitoring
  - No tool to manage rules in a pool of webservers running ModSecurity

Usefull products combination

- Increase ModSecurity efficiency through integration with other Open Source applications
  - Replicate ModSecurity configuration into a web farm with rsync through ssh (http://samba.anu.edu.au/rsync/)
  - Collect consolidated access log of all your webservers in one point for realtime or batch analysis with mod_log_spread (http://www.backhand.org/mod_log_spread/)
  - Use in combination with network IDS like Snort (http://www.snort.org/)
  - Protect java application (web services) with ModSecurity by using mod_jk2 and Tomcat (http://www.infosecwriters.com/text_resources/pdf/Defending-web-services.pdf)
  - Realtime update of firewall rules based on ModSecurity logging
  - Anti-virus filtering (see product evolutions)

Product evolutions

- New features in devel version
  - Integration with anti-virus like ClamAV (http://www.clamav.net/)
  - Audit logging improvement
  - Integration with httpd-guardian (http://www.apachesecurity.net/tools/)
  - New proxy action
  - ModSecurity activation/deactivation per request
  - …
- Java version (http://www.modsecurity.org/projects/modsecurity/java/index.html)
- Other external OpenSource development (GUI, monitoring console, …)

SWOT Analysis

Strenght No license fee Flexibility Embedded into web server	**Weakness** No user friendly tools One instance per server Only working with Apache
Opportunities Easy integration with other tools	**Threads** Breaking normal app workflow with false positive

Total cost of ownership

- Many factors may influence TCO of Web application firewall
  - Application complexity
  - Development cycles
  - Bandwidth/Number of hits/visits
  - Required security level
  - Number of servers
  - …
- Case study 1: one webserver
- Case study 2: webfarm with x servers

TCO – Case study 1: one webserver

	ModSecurity	Commercial App
Hard/Soft		
Hardware		+
License Fee		+
Installation		
Soft/hard Installation		+
Application workflow analysis	=	=
Policy Security Definition	=	=
Rules configuration	=	=
Monitoring configuration	+	
Custom application	+	
Documentation	=	=
Training	=	=
Maintenance		
Software/firmware upgrade	=	=
Rules updates	=	=
Customer support	=	=
Day 2 Day monitoring	=	=
Annual license fee		+

TCO – Case study 2: webfarm with x servers

	ModSecurity	**Commercial App**
Hard/Soft		
Hardware		+
License Fee		+
Installation		
Soft/hard Installation		+
Application workflow analysis	=	=
Policy Security Definition	=	=
Rules configuration	=	=
Monitoring configuration	+	
Custom application	+	
Documentation	=	=
Training	=	=
Maintenance		
Software/firmware upgrade	+	
Rules updates	+	=
Customer support	=	=
Day 2 Day monitoring	=	=
Annual license fee		+

That’s it…

- Any Questions?

Thank you!