基于nginx和modsecurity的WAF防火墙实现

第一部分 简单规则

引擎：使用nginx内置变量及正则表达式实现

作用范围：站点server字段

有部分规则重复进行注释处理，当modsecurity模块不可用时取消注释使其生效

1.过滤文件和路径

阻止 /~ 这种带有波浪线的路径

#阻止文件类型（扩展名、后缀）

.(bzr|cvs|git|svn)

.(bak|backup|bzr|cfg|conf|cvs|doc|docx|DS\_Store|ear|git|gitignore|hg|htaccess|htpasswd|ini|inc|jar|log|online|production|project|properties|pl|pm|py|pyc|pyo|sh|sql|svn|swp|war)$

#阻止常见windows文件格式

.(ade|adp|app|asa|ascx|ashx|asmx|asp|aspx|axd|bas|bat|cdx|cer|chm|class|cmd|com|config|cpl|crt|cs|csproj|csh|csr|dat|dbf|dll|dos|exe|fxp|hlp|hta|htr|htw|ida|idc|idq|ins|isp|its|jse|key|ksh|licx|lnk|mad|maf|mag|mam|maq|mar|mas|mat|mau|mav|maw|mda|mdb|mde|mdt|mdw|mdz|msc|msh|msh1|msh1xml|msh2|msh2xml|mshxml|msi|msp|mst|old|ops|pass|pcd|pdb|pif|pol|prf|prg|printer|pst|pwd|resources|resx|reg|rem|scf|scr|sct|shb|shs|shtm|shtml|soap|stm|sys|url|vb|vbe|vbs|vbproj|vsdisco|webinfo|xsd|xsx|ws|wsc|wsf|wsh)$

2.过滤http请求方法 仅允许GET HEAD POST OPTIONS

3.过滤用户代理

阻止各种机器人（robot），爬虫（spider），下载器，测试工具，注入工具，扫描器

BTWebClient|FlashGet|FreshDownload|JetCar|PycURL|wget

audit|BabyKrokodil|BBBike|httrack|httperf|harvest|hydra|netsparker|Nikto|owasp|parser

Alligator|Azureus|BackStreet Browser|BW-C-2.0|Charon|LWP::Simple

ApacheBench|GetRight|github|GrabNet|Havij|Jmeter|JoeDog|masscan|mail2000|TurnitinBot|WebBench

CPython|libwww|libwww-perl|python-httplib2|python-requests|Python-urllib

arachni|absinthe|bilbo|black widow|blackwidow|brutus|bsqlbf|cgichk|dirbuster|fimap|grabber|grendel-

scan|havij|hydra|jaascois|jbrofuzz|libwhisker|metis|n-stealth|netsparker|nasl|nmap|nse|nsauditor|nikto|nessus|Openvas|pmafin

d|paros|pangolin|sqlmap|sqlninja|sql power injector|webinspect|wifinder|w3af|whatweb|webtrends security analyzer|webshag|Win

Http

AhrefsBot|AltaVista|aiHitBot|BBScan|BLEXBot|CSS Certificate Spider|COMODO SSL Checker|Dataprovider|

electricmonk|eMusic|Exabot|FeedBurner|Feedskycrawler|ia\_archiver|ips-agent|NgSpider|panscient.com|Plukkie|SemrushBot|Seznam

Bot|spiderman|seoscanners.net|SafeDNSBot|scrapbot|SurveyBot|semanticbot|SiteExplorer|Scrapy|Uptimebot|Wotbox|YRSpider

4.过滤变量:强制规范特定类型变量，比如禁止数值变量传递文本字串

~~5.过滤SQL注入~~

~~6.过滤XSS跨域~~

7.过滤referer

只允许http://和https://开头的referer

8.防止快速DOS攻击规则

全局并发请求限制不区分内容，特定内容由modsecurity模块处理

~~每个客户端IP 100并发~~

~~每个服务器域名 2000并发~~

全局请求速率限制不区分内容，特定内容由modsecurity模块处理

~~每个客户端IP 600次/分钟~~

~~每个服务器域名 2000次/秒~~

全局限速规则

~~前100MB不限速，超过100MB后限速4KB/s~~

注： Pf(packet filter)防火墙： 系统底层限速，优先级，抗DOS攻击,内核代码完成同类工作比nginx更健壮更高效

第二部分 高级规则

引擎：由第三方模块modsecurity实现

作用范围：location 字段静态页面和动态页面

已明确定义的静态资源不做过滤

web服务器指纹伪装，迷惑入侵者

SecServerSignature "Apache/2.4.25 (HardenedBSD) PHP/7.1.1"

Modsecurity规则集详单

**REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf**

**规则排除**

**REQUEST-901-INITIALIZATION.conf**

**初始化**

**REQUEST-903.9001-DRUPAL-EXCLUSION-RULES.conf**

**DRUPAL排除规则**

**REQUEST-903.9002-WORDPRESS-EXCLUSION-RULES.conf**

**WORDPRESS排除规则**

**REQUEST-905-COMMON-EXCEPTIONS.conf**

**常见异常**

Exception for Apache SSL pinger

id:905100

Exception for Apache internal dummy connection

id:905110

**REQUEST-910-IP-REPUTATION.conf**

IP信誉

# -=[ IP Reputation Block Flag Check ]=-

#

# The first check we do is to see if the client IP address has already

# been blacklisted by rules from previous requests.

#

# If the rule matches, it will do a skipAfter and pick up processing

# at the end of the request phase for actual blocking.

id:910000

# -=[ GeoIP Checks ]=-

#

# This rule requires activating the SecGeoLookupDB directive

# in the crs-setup.conf file and specifying

# the list of blocked countries (tx.high\_risk\_country\_codes).

#

# This rule does a GeoIP resolution on the client IP address.

id:910100

# -=[ IP Reputation Checks ]=-

#

# ModSecurity Rules from Trustwave SpiderLabs: IP Blacklist Alert

# Ref: http://www.modsecurity.org/projects/commercial/rules/

#

# This rule checks the client IP address against a list of recent IPs captured

# from the SpiderLabs web honeypot systems (last 48 hours).

id:910110

# First check if we have already run an @rbl check for this IP by checking in IP collection.

# If we have, then skip doing another check.

id:910120

# Check Client IP against ProjectHoneypot's HTTP Blacklist

# Ref: http://www.projecthoneypot.org/httpbl\_api.php

#

# To use the blacklist, you must register for an HttpBL API Key

# and choose the traffic types to block. See section

# "Project Honey Pot HTTP Blacklist" in crs-setup.conf.

#

# Ref: https://github.com/SpiderLabs/ModSecurity/wiki/Reference-Manual#wiki-SecHttpBlKey

#

# Skip HttpBL checks if user has not defined one of the TX:block\_\* variables.

# This prevents error "Operator error: RBL httpBl called but no key defined: set SecHttpBlKey"

id:910130

id:910140

id:910150

id:910160

id:910170

id:910180

id:910190

**REQUEST-911-METHOD-ENFORCEMENT.conf**

方法限制

# -=[ Allowed Request Methods ]=-

#

# tx.allowed\_methods is defined in the crs-setup.conf file

id:911100

**REQUEST-912-DOS-PROTECTION.conf**

拒绝服务保护

# Anti-Automation rules to detect Denial of Service attacks.

#

# Description of mechanics:

# When a request hits a non-static resource (TX:STATIC\_EXTENSIONS), then a counter for the IP

# address is being raised (IP:DOS\_COUNTER). If the counter (IP:DOS\_COUNTER) hits a limit

# (TX:DOS\_COUNTER\_THRESHOLD), then a burst is identified (IP:DOS\_BURST\_COUNTER) and the

# counter (IP:DOS\_COUNTER) is reset. The burst counter expires within a timeout period

# (TX:DOS\_BURST\_TIME\_SLICE).

# If the burst counter (IP:DOS\_BURST\_COUNTER) is greater equal 2, then the blocking flag

# is being set (IP:DOS\_BLOCK). The blocking flag (IP:DOS\_BLOCK) expires within a timeout

# period (TX:DOS\_BLOCK\_TIMEOUT). All this counting happens in phase 5.

# There is a stricter sibling to this rule (912170) in paranoia level 2, where the

# burst counter check (IP:DOS\_BURST\_COUNTER) hits at greater equal 1.

#

# The blocking is done in phase 1: When the blocking flag is encountered (IP:DOS\_BLOCK),

# then the request is dropped without sending a response. If this happens, then a

# counter is # raised (IP:DOS\_BLOCK\_COUNTER).

# When an IP address is blocked for the first time, then the blocking is reported in a

# message and a flag (IP:DOS\_BLOCK\_FLAG) is set. This flag expires in 60 seconds.

# When an IP address is blocked and the flag (IP:DOS\_BLOCK\_FLAG) is set, then the

# blocking is not being reported (to prevent a flood of alerts). When the flag

# (IP:DOS\_BLOCK\_FLAG) has expired and a new request is being blocked, then the

# counter (IP:DOS\_BLOCK\_COUNTER) is being reset to 0 and the block is being treated

# as the first block (-> alert).

# In order to be able to display the counter (IP:DOS\_BLOCK\_COUNTER) and resetting

# it at the same time, we copy the counter (IP:DOS\_BLOCK\_COUNTER) into a different

# variable (TX:DOS\_BLOCK\_COUNTER), which is then displayed in turn.

#

# Variables:

# IP:DOS\_BLOCK Flag if an IP address should be blocked

# IP:DOS\_BLOCK\_COUNTER Counter of blocked requests

# IP:DOS\_BLOCK\_FLAG Flag keeping track of alert. Flag expires after 60 seconds.

# IP:DOS\_BURST\_COUNTER Burst counter

# IP:DOS\_COUNTER Request counter (static resources are ignored)

# TX:DOS\_BLOCK\_COUNTER Copy of IP:DOS\_BLOCK\_COUNTER (needed for display reasons)

# TX:DOS\_BLOCK\_TIMEOUT Period in seconds a blocked IP will be blocked

# TX:DOS\_COUNTER\_THRESHOLD Limit of requests, where a burst is identified

# TX:DOS\_BURST\_TIME\_SLICE Period in seconds when we will forget a burst

# TX:STATIC\_EXTENSIONS Paths which can be ignored with regards to DoS

#

# As a precondition for these rules, please set the following three variables:

# - TX:DOS\_BLOCK\_TIMEOUT

# - TX:DOS\_COUNTER\_THRESHOLD

# - TX:DOS\_BURST\_TIME\_SLICE

#

# And make sure that TX:STATIC\_EXTENSIONS is also set.

#

# -=[ Anti-Automation / DoS Protection : Block ]=-

#

#

# Block and track # of requests and log

#

id:912120

id:912130

id:912140

id:912150

**REQUEST-913-SCANNER-DETECTION.conf**

扫描器保护

# -=[ Vulnerability Scanner Checks ]=-

#

# These rules inspect the default User-Agent and Header values sent by

# various commercial and open source vuln scanners.

#

# The following rules contain User-Agent lists:

# 913100 - security scanners (data file scanners-user-agents.data)

# 913101 - scripting/generic HTTP clients (data file scripting-user-agents.data)

# 913102 - web crawlers/bots (data file crawlers-user-agents.data)

**REQUEST-920-PROTOCOL-ENFORCEMENT.conf**

协议限制

# Some protocol violations are common in application layer attacks.

# Validating HTTP requests eliminates a large number of application layer attacks.

#

# The purpose of this rules file is to enforce HTTP RFC requirements that state how

# the client is supposed to interact with the server.

# http://www.w3.org/Protocols/rfc2616/rfc2616-sec3.html

# Validate request line against the format specified in the HTTP RFC

#

# -=[ Rule Logic ]=-

#

# Uses rule negation against the regex for positive security. The regex specifies the proper

# construction of URI request lines such as:

#

# "http:" "//" host [ ":" port ] [ abs\_path [ "?" query ]]

#

# It also outlines proper construction for CONNECT, OPTIONS and GET requests.

#

# -=[ References ]=-

# http://www.w3.org/Protocols/rfc2616/rfc2616-sec3.html#sec3.2.1

# http://capec.mitre.org/data/definitions/272.html

id:920100

# Identify multipart/form-data name evasion attempts

#

# There are possible impedance mismatches between how

# ModSecurity interprets multipart file names and how

# a destination app server such as PHP might parse the

# Content-Disposition data:

#

# filename-parm := "filename" "=" value

#

# -=[ Rule Logic ]=-

# These rules check for the existence of the ' " ; = meta-characters in

# either the file or file name variables.

# HTML entities may lead to false positives, why they are allowed on PL1.

# Negative look behind assertions allow frequently used entities &\_;

#

# -=[ Targets, characters and html entities ]=-

#

# 920120: PL1 : FILES\_NAMES, FILES

# ['\";=] but allowed:

# &[aAoOuUyY]uml); &[aAeEiIoOuU]circ; &[eEiIoOuUyY]acute;

# &[aAeEiIoOuU]grave; &[cC]cedil; &[aAnNoO]tilde; &amp; &apos;

#

# 920121: PL2 : FILES\_NAMES, FILES

# ['\";=] : ' " ; = meta-characters

#

# -=[ References ]=-

# https://www.owasp.org/index.php/ModSecurity\_CRS\_RuleID-960000

# http://www.ietf.org/rfc/rfc2183.txt

#

id:920120

# Verify that we've correctly processed the request body.

#

# As a rule of thumb, when failing to process a request body

# you should reject the request (when deployed in blocking mode)

# or log a high-severity alert (when deployed in detection-only mode).

#

# -=[ Rule Logic ]=-

# Checks for the existence of the REQBODY\_ERROR variable that is created

# by the request body processor if it encounters errors.

#

# -=[ References ]=-

# https://sourceforge.net/apps/mediawiki/mod-security/index.php?title=Reference\_Manual#REQBODY\_ERROR

id:920130

# Strict Multipart Parsing Checks

#

# -=[ Rule Logic ]=-

# By default be strict with what we accept in the multipart/form-data

# request body. If the rule below proves to be too strict for your

# environment consider changing it to detection-only. You are encouraged

# \_not\_ to remove it altogether.

#

# -=[ References ]=-

# https://sourceforge.net/apps/mediawiki/mod-security/index.php?title=Reference\_Manual#MULTIPART\_STRICT\_ERROR

id:920140

# Accept only digits in content length

#

# -=[ Rule Logic ]=-

# This rule uses ModSecurity's rule negation against the regex meaning if the Content-Length header

# is NOT all digits, then it will match.

#

# -=[ References ]=-

# http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.13

id:920160

# Do not accept GET or HEAD requests with bodies

# HTTP standard allows GET requests to have a body but this

# feature is not used in real life. Attackers could try to force

# a request body on an unsuspecting web applications.

#

# -=[ Rule Logic ]=-

# This is a chained rule that first checks the Request Method. If it is a

# GET or HEAD method, then it checks for the existence of a Content-Length

# header. If the header exists and its payload is either not a 0 digit or not

# empty, then it will match.

#

# -=[ References ]=-

# http://www.w3.org/Protocols/rfc2616/rfc2616-sec4.html#sec4.3

id:920170

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

#

# -=[ Rule Logic ]=-

# This chained rule checks if the request method is POST, if so, it checks that a Content-Length

# header is also present.

#

# -=[ References ]=-

# http://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html#sec9.5

id:920180

# Range Header Checks

#

# 1. Range Header exists and begins with 0 - normal browsers don't do this.

# Automated programs and bots often do not obey the HTTP RFC

#

# -=[ Rule Logic ]=-

# This rule inspects the Range request header to see if it starts with 0.

#

# -=[ References ]=-

# http://www.bad-behavior.ioerror.us/documentation/how-it-works/

#

# 2. Per RFC 2616 -

# "If the last-byte-pos value is present, it MUST be greater than or equal to the first-byte-pos in that byte-range-spec,

# or the byte- range-spec is syntactically invalid."

# -=[ Rule Logic ]=-

# This rule compares the first and second byte ranges and flags when the first value is greater than the second.

#

# -=[ References ]=-

# http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html

# http://seclists.org/fulldisclosure/2011/Aug/175

id:920190

# Broken/Malicous clients often have duplicate or conflicting headers

# Automated programs and bots often do not obey the HTTP RFC

#

# -=[ Rule Logic ]=-

# This rule inspects the Connection header and looks for duplicates of the

# keep-alive and close options.

#

# -=[ References ]=-

# http://www.bad-behavior.ioerror.us/documentation/how-it-works/

id:920210

# Check URL encodings

#

# -=[ Rule Logic ]=-

# There are two different chained rules. We need to separate them as we are inspecting two

# different variables - REQUEST\_URI and REQUEST\_BODY. For REQUEST\_BODY, we only want to

# run the @validateUrlEncoding operator if the content-type is application/x-www-form-urlencoding.

#

# -=[ References ]=-

# http://www.ietf.org/rfc/rfc1738.txt

id:920220

id:920240

# Check UTF enconding

# We only want to apply this check if UTF-8 encoding is actually used by the site, otherwise

# it will result in false positives.

#

# -=[ Rule Logic ]=-

# This chained rule first checks to see if the admin has set the TX:CRS\_VALIDATE\_UTF8\_ENCODING

# variable in the crs-setup.conf file.

id:920250

# Disallow use of full-width unicode as decoding evasions my be possible.

#

# -=[ Rule Logic ]=-

# This rule looks for full-width encoding by looking for %u followed by 2 'f'

# characters and then 2 hex characters. It is a vulnerability that affected

# IIS circa 2007.

# The rule will trigger on %uXXXX formatted chars that are full or half

# width, as explained above. This %uXXXX format is passed as a raw parameter

# and is (seemingly only) accepted by IIS (5.0, 6.0, 7.0, and 8.0). Other

# webservers will only process unicode chars presented as hex UTF-8 bytes.

#

# -=[ References ]=-

# http://www.kb.cert.org/vuls/id/739224

# https://www.checkpoint.com/defense/advisories/public/2007/cpai-2007-201.html

# https://github.com/SpiderLabs/owasp-modsecurity-crs/issues/719

#

id:920260

# Restrict type of characters sent

#

# This is a rule with multiple stricter siblings that grows more

# restrictive in higher paranoia levels.

#

# -=[ Rule Logic ]=-

# This rule uses the @validateByteRange operator to restrict the request

# payloads.

#

# -=[ Targets and ASCII Ranges ]=-

#

# 920270: PL1 : REQUEST\_URI, REQUEST\_HEADERS, ARGS and ARGS\_NAMES

# ASCII 1-255 : Full ASCII range without null character

#

# 920271: PL2 : REQUEST\_URI, REQUEST\_HEADERS, ARGS and ARGS\_NAMES

# ASCII 9,10,13,32-126,128-255 : Full visible ASCII range, tab, newline

#

# 920272: PL3 : REQUEST\_URI, REQUEST\_HEADERS, ARGS, ARGS\_NAMES and REQUEST\_BODY

# ASCII 32-36,38-126 : Visible lower ASCII range without percent symbol

#

# 920273: PL4 : ARGS, ARGS\_NAMES and REQUEST\_BODY

# ASCII 38,44-46,48-58,61,65-90,95,97-122

# A-Z a-z 0-9 = - \_ . , : &

#

# 920274: PL4 : REQUEST\_HEADERS without User-Agent, Referer and Cookie

# ASCII 32,34,38,42-59,61,65-90,95,97-122

# A-Z a-z 0-9 = - \_ . , : & " \* + / SPACE

#

# REQUEST\_URI and REQUEST\_HEADERS User-Agent, Referer and Cookie are very hard

# to restrict beyond the limits in 920272.

#

# 920274 generally has few positives. However, it would detect rare attacks

# on Accept request headers and friends.

id:920270

# Do not accept requests without common headers.

# All normal web browsers include Host, User-Agent and Accept headers.

# Implies either an attacker or a legitimate automation client.

#

#

# Missing/Empty Host Header

#

# -=[ Rule Logic ]=-

# These rules will first check to see if a Host header is present.

# The second check is to see if a Host header exists but is empty.

id:920280

id:920290

# Empty Accept Header

#

# -=[ Rule Logic ]=-

# This rule checks if an Accept header exists, but has an empty value.

# This is only allowed in combination with the OPTIONS method.

# Additionally, there are some clients sending empty Accept headers.

# They are covered in another chained rule checking the User-Agent.

# This technique demands a separate rule to detect an empty

# Accept header if there is no user agent. This is checked via

# the separate rule 920311.

#

# Exclude some common broken clients sending empty Accept header:

# "Business/6.6.1.2 CFNetwork/758.5.3 Darwin/15.6.0" (CRS issue #515)

# "Entreprise/6.5.0.177 CFNetwork/758.4.3 Darwin/15.5.0" (CRS issue #366)

#

# -=[ References ]=-

# https://github.com/SpiderLabs/owasp-modsecurity-crs/issues/366

id:920310

# Empty User-Agent Header

#

# -=[ Rule Logic ]=-

# This rules will check to see if the User-Agent header is empty.

#

# Note that there is a second rule, 920320, which will check for

# the existence of the User-Agent header.

id:920330

# Missing Content-Type Header with Request Body

#

# -=[ Rule Logic]=-

# This rule will first check to see if the value of the Content-Length header is

# non-equal to 0. The chained rule is then checking the existence of the

# Content-Type header. The RFCs do not state there must be a

# Content-Type header. However, a request missing a Content-Header is a

# strong indication of a non-compliant browser.

#

# -=[ References ]=-

# http://httpwg.org/specs/rfc7231.html#header.content-type

id:920340

# Check that the host header is not an IP address

# This is not an HTTP RFC violation but it is indicative of automated client access.

# Many web-based worms propagate by scanning IP address blocks.

#

# -=[ Rule Logic ]=-

# This rule triggers if the Host header contains all digits (and possible port)

#

# -=[ References ]=-

# http://technet.microsoft.com/en-us/magazine/2005.01.hackerbasher.aspx

id:920350

# Maximum number of arguments in request limited

id:920380

## -- Arguments limits --

#

# Limit argument name length

id:920360

# Limit argument value length

id:920370

# Limit arguments total length

id:920390

# -- File upload limits --

#

# Individual file size is limited

id:920400

# Combined file size is limited

id:920410

# Restrict which content-types we accept

id:920420

# Restrict protocol versions

id:920430

# Restrict file extension

id:920440

# Restricted HTTP headers

#

# -=[ Rule Logic ]=-

# The use of certain headers is restricted. They are listed in the variable

# TX.restricted\_headers.

#

# The headers are transformed into lowercase before the match. In order to

# make sure that only complete header names are matching, the names in

# TX.restricted\_headers are wrapped in slashes. This guarantees that the

# header Range (-> /range/) is not matching the restricted header

# /content-range/ for example.

#

# This is a chained rule, where the first rule fills a set of variables of the

# form TX.header\_name\_<HEADER\_NAME>. The second rule is then executed for all

# variables of the form TX.header\_name\_<HEADER\_NAME>.

#

# As a consequence of the construction of the rule, the alert message and the

# alert data will not display the original header name Content-Range, but

# /content-range/ instead.

#

#

# -=[ References ]=-

# https://access.redhat.com/security/vulnerabilities/httpoxy (Header Proxy)

id:920450

# Check the number of range fields in the Range request header.

#

# An excessive number of Range request headers can be used to DoS a server.

# The original CVE proposed an arbitrary upper limit of 5 range fields.

#

# Several clients are known to request PDF fields with up to 34 range

# fields. Therefore the standard rule does not cover PDF files. This is

# performed in two separate (stricter) siblings of this rule.

#

# 920200: PL2: Limit of 5 range header fields for all filenames outside of PDFs

# 920201: PL2: Limit of 34 range header fields for PDFs

# 920202: PL4: Limit of 5 range header fields for PDFs

#

# -=[ References ]=-

# https://httpd.apache.org/security/CVE-2011-3192.txt

id:920200

# Missing Accept Header

#

# -=[ Rule Logic ]=-

# This rule generates a notice if the Accept header is missing

id:920300

# Missing User-Agent Header

#

# -=[ Rule Logic ]=-

# This rules will check to see if there is a User-Agent header or not.

id:920320

# -=[ Abnormal Character Escapes ]=-

#

# [ Rule Logic ]

# Consider the following payload: arg=cat+/e\tc/pa\ssw\d

# Here, \s and \d were only used to obfuscate the string passwd and a lot of

# parsers will silently ignore the non-necessary escapes. The case with \t is

# a bit different though, as \t is a natural escape for the TAB character,

# so we will avoid this (and \n, \r, etc.).

#

# This rule aims to detect non-necessary, abnormal esacpes. You could say it is

# a nice # way to forbid the backslash character where it is not needed.

#

# This is a new rule at paranoia level 4. We expect quite a few false positives

# for this rule and we will later evaluate if the rule makes any sense at all.

# The rule is redundant with 920273 and 920274 in PL4. But if the rule proofs

# to be useful and false positives remain at a reasonable level, then it might

# be shifted to PL3 in a future release, where it would be the only rule

# covering the backslash escape.

#

# The rule construct is overly complex due to the fact that matching the

# backslash character with \b did not work. \Q\\\E does match the backslash

# character though. This is thus the base of the rule. We forbid the backslash

# when followed by a list of basic ascii characters - unless the backslash

# is preceded by another backslash character, which is being checked via a

# negative look-behind construct. If that is the case, the backslash character

# is allowed.

id:920460

**REQUEST-921-PROTOCOL-ATTACK.conf**

**协议攻击**

# -=[ HTTP Request Smuggling ]=-

#

# [ Rule Logic ]

# This rule looks for a comma character in either the Content-Length or Transfer-Encoding

# request headers. This character would indicate that there were more than one request header

# with this same name. In these instances, Apache treats the data in a similar manner as

# multiple cookie values.

#

# [ References ]

# http://projects.webappsec.org/HTTP-Request-Smuggling

# http://article.gmane.org/gmane.comp.apache.mod-security.user/3299

id:921100

id:921110

# -=[ HTTP Response Splitting ]=-

#

# [ Rule Logic ]

# These rules look for Carriage Return (CR) %0d and Linefeed (LF) %0a characters.

# These characters may cause problems if the data is returned in a respones header and

# may be interpreted by an intermediary proxy server and treated as two separate

# responses.

#

# [ References ]

# http://projects.webappsec.org/HTTP-Response-Splitting

id:921120

id:921130

# -=[ HTTP Header Injection ]=-

#

# [ Rule Logic ]

# These rules look for Carriage Return (CR) %0d and Linefeed (LF) %0a characters,

# on their own or in combination with header field names.

# These characters may cause problems if the data is returned in a respones header

# and interpreted by the client.

# The rules are similar to rules defending against the HTTP Request Splitting and

# Request Smuggling rules.

#

# [ References ]

# https://en.wikipedia.org/wiki/HTTP\_header\_injection

id:921140

# Detect newlines in argument names.

# Checking for GET arguments has been moved to paranoia level 2 (921151)

# in order to mitigate possible false positives.

#

id:921150

id:921160

# Detect newlines in GET argument values.

# These may point to a HTTP header injection attack, but can also sometimes

# occur in benign query parameters.

#

# See also: rule 921140, 921150

# -=[ HTTP Parameter Polution ]=-

#

# [ Rule Logic ]

# These rules look for multiple parameters with the same name.

# 921170 counts the occurrences of the individual parameters.

# 921180 checks if any counter is > 1.

#

# One HPP attack vector is to try evade signature filters by distributing the

# attack payload across multiple parameters with the same name.

# This works as many security devices only apply signatures to individual

# parameter payloads, however the back-end web application may (in the case

# of ASP.NET) consolidate all of the payloads into one thus making the

# attack payload active.

#

# [ References ]

# http://tacticalwebappsec.blogspot.com/2009/05/http-parameter-pollution.html

# https://capec.mitre.org/data/definitions/460.html

id:921170

id:921180

**REQUEST-930-APPLICATION-ATTACK-LFI.conf**

**应用攻击-本地文件包含**

# -=[ Directory Traversal Attacks ]=-

#

# Ref: https://github.com/wireghoul/dotdotpwn

#

# [ Encoded /../ Payloads ]

id:930100

id:930110

# -=[ OS File Access ]=-

#

# Ref: https://github.com/lightos/Panoptic/blob/master/cases.xml

id:930120

# -=[ Restricted File Access ]=-

#

# Detects attempts to retrieve application source code, metadata,

# credentials and version control history possibly reachable in a web root.

id:930130

**REQUEST-931-APPLICATION-ATTACK-RFI.conf**

**应用攻击-远程文件包含**

# -=[ Rule Logic ]=-

# These rules look for common types of Remote File Inclusion (RFI) attack methods.

# - URL Contains an IP Address

# - The PHP "include()" Function

# - RFI Data Ends with Question Mark(s) (?)

# - RFI Host Doesn't Match Local Host

#

# -=[ References ]=-

# http://projects.webappsec.org/Remote-File-Inclusion

# http://tacticalwebappsec.blogspot.com/2009/06/generic-remote-file-inclusion-attack.html

id:931100

id:931110

id:931120

**REQUEST-932-APPLICATION-ATTACK-RCE.conf**

**应用攻击-RCE**

# [ Unix command injection ]

#

# This rule detects Unix command injections.

# A command injection takes a form such as:

#

# foo.jpg;uname -a

# foo.jpg||uname -a

#

# The vulnerability exists when an application executes a shell command

# without proper input escaping/validation.

#

# To prevent false positives, we look for a 'starting sequence' that

# precedes a command in shell syntax, such as: ; | & $( ` <( >(

#

# This rule is case-sensitive to prevent FP ("Cat" vs. "cat").

#

# An effort was made to combat evasions by shell quoting (e.g. 'ls',

# 'l'"s", \l\s are all valid). ModSecurity has a t:cmdLine

# transformation built-in to deal with this, but unfortunately, it

# replaces ';' characters and lowercases the payload, which is less

# useful for this case. However, emulating the transformation makes

# the regexp more complex.

#

# To rebuild the word list regexp:

# cd util/regexp-assemble

# cat regexp-932100.txt | ./regexp-cmdline.py unix | ./regexp-assemble.pl

id:932100

# Apache 2.2 requires configuration file lines to be under 8kB.

# Therefore, some remaining commands have been split off to a separate rule.

# For explanation of this rule, see rule 932100.

#

# To rebuild the word list regexp:

# cd util/regexp-assemble

# cat regexp-932105.txt | ./regexp-cmdline.py unix | ./regexp-assemble.pl

id:932105

# [ Windows command injection ]

#

# This rule detects Windows shell command injections.

# If you are not running Windows, it is safe to disable this rule.

id:932110

# [ Windows PowerShell, cmdlets and options ]

#

# Detect some common PowerShell commands, cmdlets and options.

# These commands should be relatively uncommon in normal text, but

# potentially useful for code injection.

#

# If you are not running Windows, it is safe to disable this rule.

#

# https://technet.microsoft.com/en-us/magazine/ff714569.aspx

# https://msdn.microsoft.com/en-us/powershell/scripting/core-powershell/console/powershell.exe-command-line-help

id:932120

# [ Unix shell expressions ]

#

# Detects the following patterns which are common in Unix shell scripts

# and oneliners:

#

# $(foo) Command substitution

# ${foo} Parameter expansion

# <(foo) Process substitution

# >(foo) Process substitution

# $((foo)) Arithmetic expansion

#

# Regexp generated from util/regexp-assemble/regexp-932130.data using Regexp::Assemble.

# See http://blog.modsecurity.org/2007/06/optimizing-regu.html for usage.

id:932130

# [ Windows FOR, IF commands ]

#

# This rule detects Windows command shell FOR and IF commands.

# If you are not running Windows, it is safe to disable this rule.

id:932140

# [ Unix direct remote command execution ]

#

# Detects Unix commands at the start of a parameter (direct RCE).

# Example: foo=wget%20www.example.com

id:932150

# [ Unix shell snippets ]

#

# Detect some common sequences found in shell commands and scripts.

#

# Some commands which were restricted in earlier rules due to FP,

# have been added here with their full path, in order to catch some

# cases where the full path is sent.

id:932160

# [ Shellshock vulnerability (CVE-2014-6271 and CVE-2014-7169) ]

#

# Detect exploitation of "Shellshock" GNU Bash RCE vulnerability.

#

# Based on ModSecurity rules created by Red Hat.

# Permission for use was granted by Martin Prpic <secalert@redhat.com>

#

# https://access.redhat.com/articles/1212303

id:932170

id:932171

**REQUEST-933-APPLICATION-ATTACK-PHP.conf**

**应用攻击-PHP**

# -=[ PHP Injection Attacks ]=-

#

# [ References ]

# http://rips-scanner.sourceforge.net/

# https://www.owasp.org/index.php/PHP\_Top\_5#P1:\_Remote\_Code\_Executionh

#

#

# [ PHP Open Tag Found ]

#

# Detects PHP open tags "<?" and "<?php".

# http://www.php.net/manual/en/language.basic-syntax.phptags.php

#

# Care is taken to avoid false positives in XML declarations "<?xml..."

#

# Also detects "[php]", "[/php]" and "[\php]" tags used by some applications

# to indicate PHP dynamic content.

id:933100

# [ PHP Script Uploads ]

#

# Block file uploads with PHP extensions (.php, .php5, .phtml etc).

#

# Many application contain Unrestricted File Upload vulnerabilities.

# https://www.owasp.org/index.php/Unrestricted\_File\_Upload

#

# Attackers may use such a vulnerability to achieve remote code execution

# by uploading a .php file. If the upload storage location is predictable

# and not adequately protected, the attacker may then request the uploaded

# .php file and have the code within it executed on the server.

#

# Also block files with just dot (.) characters after the extension:

# https://community.rapid7.com/community/metasploit/blog/2013/08/15/time-to-patch-joomla

#

# Some AJAX uploaders use the nonstandard request headers X-Filename,

# X\_Filename, or X-File-Name to transmit the file name to the server;

# scan these request headers as well as multipart/form-data file names.

id:933110

# [ PHP Configuration Directives ]

id:933120

# [ PHP Variables ]

id:933130

# [ PHP I/O Streams ]

#

# The "php://" syntax can be used to refer to various objects, such as local files (for LFI),

# remote urls (for RFI), or standard input/request body. Its occurrence indicates a possible attempt

# to either inject PHP code or exploit a file inclusion vulnerability in a PHP web app.

id:933140

# [ PHP Functions ]

#

# Detecting PHP function names is useful to block PHP code injection attacks.

# There are many PHP functions. We have to strike a balance between robust detection

# of PHP code in content, and the risk of false positives.

#

# The list of PHP functions is divided into four groups of varying attack/false positive risk.

# Four separate rules are used to detect these groups of functions:

#

# - Rule 933150: ~40 words highly common to PHP injection payloads and extremely rare in

# natural language or other contexts.

# Examples: 'base64\_decode', 'file\_get\_contents'.

# These words are detected as a match directly using @pmf.

# Function names are defined in php-function-names-933150.data

#

# - Rule 933160: ~220 words which are common in PHP code, but have a higher chance to cause

# false positives in natural language or other contexts.

# Examples: 'chr', 'eval'.

# To mitigate false positives, a regexp looks for PHP function syntax, e.g. 'eval()'.

# Regexp is generated from function names in util/regexp-assemble/regexp-933160.data

#

# - Rule 933151: ~1300 words of lesser importance. This includes most PHP functions and keywords.

# Examples: 'addslashes', 'array\_diff'.

# For performance reasons, the @pmf operator is used, and many functions from lesser

# used PHP extensions are removed.

# To mitigate false positives, we only match when the '(' character is also found.

# This rule only runs in paranoia level 2 or higher.

# Function names are defined in php-function-names-933151.data

#

# - Rule 933161: ~200 words with short or trivial names, possibly leading to false positives.

# Examples: 'abs', 'cos'.

# To mitigate false positives, a regexp matches on function syntax, e.g. 'abs()'.

# This rule only runs in paranoia level 3 or higher.

# Regexp is generated from function names in util/regexp-assemble/regexp-933161.data

#

#

# [ PHP Functions: High-Risk PHP Function Names ]

#

# Rule 933150 contains a small list of function names which are highly indicative of a PHP

# injection attack, for example 'base64\_decode'.

# We block these function names outright, without using a complex regexp or chain.

# This could make the detection a bit more robust against possible bypasses.

id:933150

id:933160

# [ PHP Object Injection ]

#

# PHP Object Injection is an application level vulnerability that could allow

# an attacker to perform different kinds of malicious attacks, such as

# Code Injection, SQL Injection, Path Traversal and Application Denial of Service,

# depending on the context.

#

# The vulnerability occurs when user-supplied input is not properly sanitized

# before being passed to the unserialize() PHP function. Since PHP allows object

# serialization, attackers could pass ad-hoc serialized strings to a vulnerable

# unserialize() call, resulting in an arbitrary PHP object(s) injection into the

# application scope.

#

# https://www.owasp.org/index.php/PHP\_Object\_Injection

id:933170

# [ PHP Functions: Variable Function Calls ]

#

# PHP 'variable functions' provide an alternate syntax for calling PHP functions.

# http://php.net/manual/en/functions.variable-functions.php

#

# An attacker may use variable function syntax to evade detection of function

# names during exploitation of a remote code execution vulnerability.

# An example to use the 'file\_get\_contents' function while evading rule 933150:

#

# $fn = 'file\_' . 'get\_' . 'contents';

# echo $fn('wp-co' . 'nfig.php');

id:933180

id:933151

**REQUEST-941-APPLICATION-ATTACK-XSS.conf**

**应用攻击-XSS跨域**

# -=[ Libinjection - XSS Detection ]=-

#

# Ref: https://libinjection.client9.com/

# Ref: https://speakerdeck.com/ngalbreath/libinjection-from-sqli-to-xss

#

# -=[ Targets ]=-

#

# 941100: PL1 : REQUEST\_COOKIES|!REQUEST\_COOKIES:/\_\_utm/|

# REQUEST\_COOKIES\_NAMES|REQUEST\_HEADERS:User-Agent|

# ARGS\_NAMES|ARGS|XML:/\*

#

# 941101: PL2 : REQUEST\_HEADERS:Referer

# -=[ XSS Filters - Category 1 ]=-

# http://xssplayground.net23.net/xssfilter.html

# script tag based XSS vectors, e.g., <script> alert(1)</script>

# -=[ XSS Filters - Category 2 ]=-

# XSS vectors making use of event handlers like onerror, onload etc, e.g., <body onload="alert(1)">

# -=[ XSS Filters - Category 3 ]=-

# -=[ XSS Filters - Category 4 ]=-

# -=[ NoScript XSS Filters ]=-

# Ref: http://noscript.net/

#

# [NoScript InjectionChecker] HTML injection

# [NoScript InjectionChecker] Attributes injection

# [Blacklist Keywords from Node-Validator]

# https://raw.github.com/chriso/node-validator/master/validator.js

# -=[ XSS Filters from IE ]=-

# Ref: http://blogs.technet.com/srd/archive/2008/08/18/ie-8-xss-filter-architecture-implementation.aspx

# Ref: http://xss.cx/examples/ie/internet-exploror-ie9-xss-filter-rules-example-regexp-mshtmldll.txt

**REQUEST-942-APPLICATION-ATTACK-SQLI.conf**

**应用攻击-SQL注入**

**REQUEST-943-APPLICATION-ATTACK-SESSION-FIXATION.conf**

**应用攻击-会话固定**

# Session fixation

#

# -=[ References ]=-

# http://projects.webappsec.org/Session-Fixation

# http://projects.webappsec.org/w/page/13246960/Session%20Fixation

# http://capec.mitre.org/data/definitions/61.html

id:943100

id:943110

id:943120

**REQUEST-949-BLOCKING-EVALUATION.conf**

评分阻断

**RESPONSE-950-DATA-LEAKAGES.conf**

数据泄密

# -=[ Directory Listing ]=-

id:950130

# -=[ The application is not available - 5xx level status code ]=-

id:950100

**RESPONSE-951-DATA-LEAKAGES-SQL.conf**

数据泄密-SQL

# -=[ SQL Error Leakages ]=-

#

# Ref: https://raw.github.com/sqlmapproject/sqlmap/master/xml/errors.xml

# Ref: https://github.com/Arachni/arachni/tree/master/modules/audit/sqli/patterns

#

id:951100

id:951130

id:951140

id:951150

id:951160

id:951170

id:951180

id:951190

**RESPONSE-952-DATA-LEAKAGES-JAVA.conf**

数据泄密-**JAVA**

# -=[ Java Source Code Leakages ]=-

id:952100

# -=[ Java Errors ]=-

id:952110

**RESPONSE-953-DATA-LEAKAGES-PHP.conf**

数据泄密**-PHP**

# -=[ PHP Error Message Leakage ]=-

id:953100

# -=[ PHP source code leakage ]=-

id:953110

# Detect the presence of the PHP open tag "<?" or "<?php" in output.

#

# To prevent false positives due to the short "<?" sequence, an attempt

# is made to stop alerts in binary output. This is done by detecting

# some common binary file format headers, such as gzip (\x1f\x8b\x08),

# png (IHDR), mp3 (ID3), movie formats et cetera.

id:953120

**RESPONSE-954-DATA-LEAKAGES-IIS.conf**

数据泄密**- IIS**

# IIS Errors leakage

id:954120

**RESPONSE-959-BLOCKING-EVALUATION.conf**

评分阻断

**RESPONSE-980-CORRELATION.conf**

关联

**RESPONSE-999-EXCLUSION-RULES-AFTER-CRS.conf**

**排除规则**

对开发人员要求

业务程序请求特征要更接近浏览器和真实人类访问请求，容易被安全软件和人员区分特征，不要与黑客工具，扫描器，机器人之类混同

1. App请求、API接口请求、定时任务或者其他程序调用

设置特定User-Agent 模拟浏览器，样例微信Mozilla/5.0 (Linux; Android 6.0; KNT-AL10 Build/HUAWEIKNT-AL10) AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/37.0.0.0 Mobile MQQBrowser/6.8 TBS/036849 Safari/537.36 MicroMessenger/6.3.27.880 NetType/WIFI Language/en

不要使用 curl，googlebot，apachebench之类user-agent ，不要用库函数默认user-agent例如Python-urllib，user-agent不能空

1. 设置访问超时和重试次数，请求失败报告错误不再发起请求，不要无限重试
2. 数据库和上游服务器开销巨大的操作，比如卡充值和查询剩余流量，人类正常请求不可能每秒100次，设置合理的时间间隔，过于频繁直接拒绝
3. 请求尽量使用GET方法，POST方法过滤性能开销过大，参数传递尽量使用GET，不能用GET情况使用POST
4. 根据规则拦截日志反馈，修改代码，最终完全开启规则也不会被误拦截。

参考：

1. <https://github.com/SpiderLabs/owasp-modsecurity-crs>

2017.04.18