Gods Web

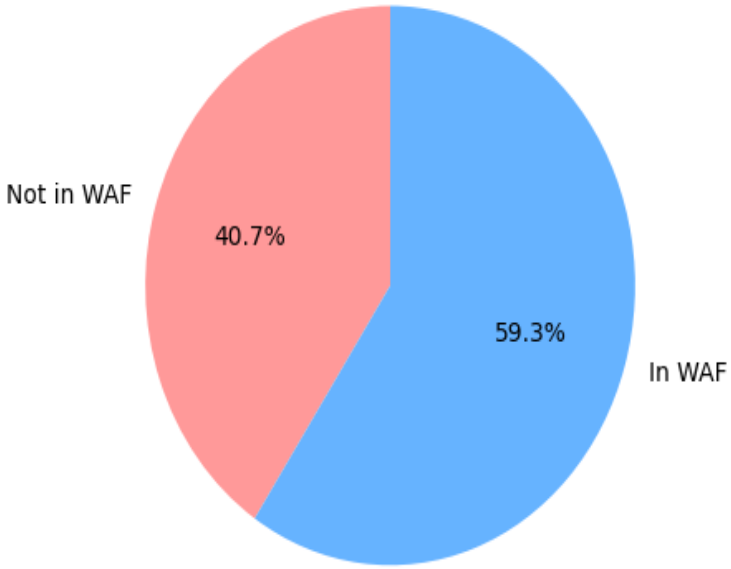
Generated on: 2023-03-09 18:43:38.348209

Overview

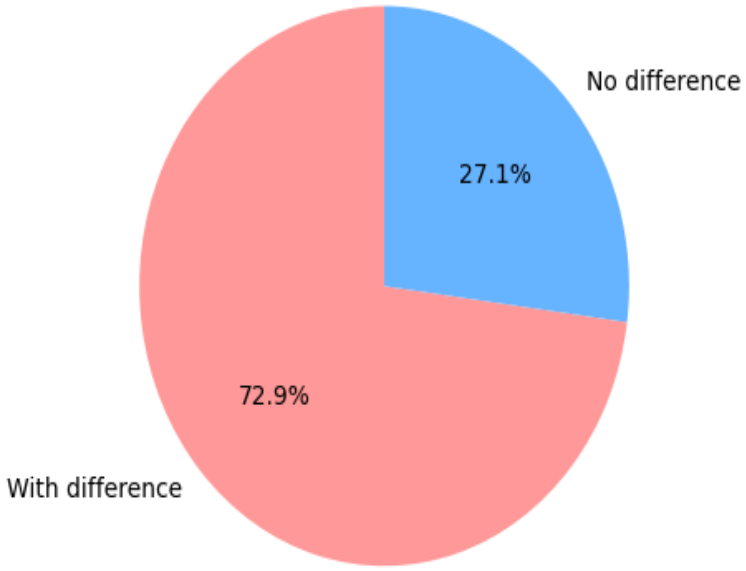
This report presents the results of God's Web, which is designed to evaluate the configuration of an OWASP Web Application Firewall (WAF) against the OWASP Core Rule Set (CRS) best practices. The tool assesses whether each CRS rule is present in the WAF configuration and ensures that the security levels of each rule cannot be lower than the CRS guidelines. The results of the audit tool are presented in the following sections, providing key findings and recommendations for improving the WAF configuration to align with CRS best practices.

Compliance

Of the 573 rules in the guideline, only 340 of them are deployed on the WAF. The following pie chart shows the distribution of WAF rules that are included in the guideline but are not present in the WAF, expressed as a percentage of the total number of WAF rules.

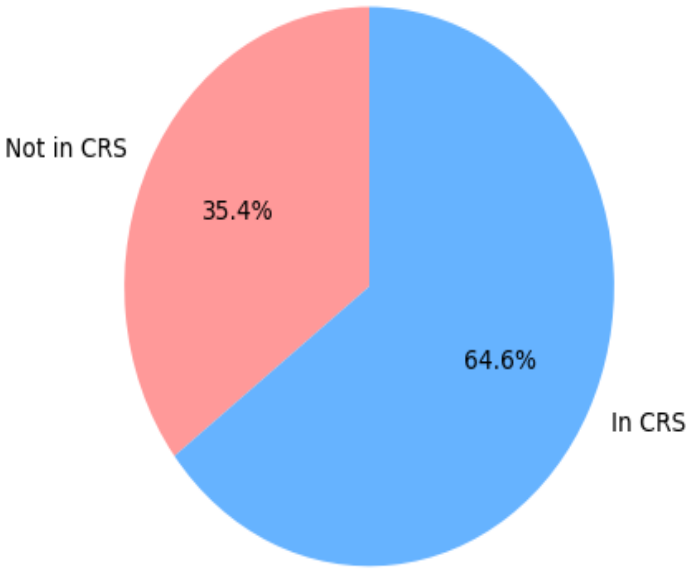


Of the 340 rules deplyed on the WAF, 248 of them have different configurations from the guideline. The following pie chart shows the distribution of WAF rules that are found to be different, expressed as a percentage of the total number of WAF rules.



WAF Breakdown

Of the 526rules on the WAF, only 340 of them are included in the ModSecurity Core Rule Set (CRS). The following pie chart shows the distribution of WAF rules that are included in the ModSecurity Core Rule Set (CRS) and custom rules, expressed as a percentage of the total number of WAF ru



Rule Variables

Variables in ModSecurity rule are used to define conditions that trigger specific actions, such as blocking or logging a request

If ModSecurity variables are set incorrectly, it can lead to unexpected behavior or errors in the rules processing. This can result in the rules not functioning as intended, or potentially blocking legitimate traffic.

The following rules have different variables configured:

Rule ID:933100
Description
PHP Injection Attack: PHP Open Tag Found
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?:<\?(?:[^x]|x[^m]|xm[^l]|xml[^\s]|xml$\|$)|<\?php|\[(?:/|\x5c)?php\])
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAM

ES|ARGS|XML:/* @rx (?:<\?(?:[^x]|x[^m]|xm[^l]|xml[^\s]|xml$|$)|<\?php|\[(?:\/|\\\\)?php\])

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:933110

Description

PHP Injection Attack: PHP Script File Upload Found

Configured Variable

SecRule FILES|REQUEST_HEADERSX-Filename|REQUEST_HEADERS:X_Filename|REQUEST_HEADERS:X.Filename|REQUEST_HEADERS:X-File-Name @rx .*\.ph(?:p\d*|tml|ar|ps|t|pt)\.*$

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a

leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule FILES|REQUEST_HEADERSX-Filename|REQUEST_HEADERS:X_Filename|REQUEST_HEADERS:X.Filename|REQUEST_HEADERS:X-File-Name @rx .*\.(?:php\d*|phtml)\.*$
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:933200
Description
PHP Injection Attack: Wrapper scheme detected
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?:bzip2|expect|glob|ogg|(?:ph|r)ar|ssh2(?:.(?:s(?:hell|(?:ft|c)p)|exec|tunnel))?|z(?:ip|lib))://
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n-

REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:zlib|glob|phar|ssh2|rar|ogg|expect|zip)://

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:933160

Description

PHP Injection Attack: High-Risk PHP Function Call Found

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|REQUEST_F

ILENAME|ARGS_NAMES|ARGS|XML:/* @rx (?i)\b(?[\]*(?:a(?:rray_(?:(?:diff|intersect)_u(?:assoc|key)|filter|map|reduce|u(?:diff|intersect)(?:_u?assoc)?)|ssert(?:_options)?)|b(?:(?:ase64_en|son_(?:de|en))code|zopen)|c(?:hr|onvert_uuencode|reate_function|url_(?:exec|file_create|init))|(?:debug_backtrac|json_(?:de|en)cod|tmpfil)e|e(?:rror_reporting|scapeshell(?:arg|cmd)|val|x(?:ec|if_(?:imagetype|read_data|t(?:agname|humbnail))))|f(?:i(?:le(?:(?:_exist|perm)s|(?:[acm]tim|inod)e|group)?|nfo_open)|open|(?:pu|unction_exis)ts|tp_(?:connec|ge|nb_(?:ge|pu)|pu)t|write)|g(?:et(?:_(?:c(?:fg_va|urrent_use)r|meta_tags)|(?:cw|lastmo)d|env|imagesize|my(?:[gpu]id|inode))|lob|z(?:compress|(?:(?:defla|wri)t|encod|fil)e|open|read))|h(?:(?:ash_(?:(?:hmac|update)_)?|ighlight_)file|e(?:ader_register_callback|x2bin)|tml(?:_entity_decode|entities|specialchars(?:_decode)?))|i(?:mage(?:2?wbmp|createfrom(?:gif|(?:jpe|pn)g|wbmp|x[bp]m)|g(?:d2?|if)|(?:jpe|pn)g|xbm)|ni_(?:get(?:_all)?|set)|ptcembed|s_(?:dir|(?:(?:execut|read|write?)ab|fi)le)|terator_apply)|m(?:b_(?:ereg(?:_(?:match|replace(?:_callback)?)|i(?:_replace)?)?|parse_str)|(?:d5|ove_uploaded)_file|ethod_exists|kdir|ysql_query)|o(?:b_(?:clean|end_(?:clean|flush)|flush|get_(?:c(?:lean|ontents)|flush)|start)|dbc_(?:connect|exec(?:ute)?|result(?:_all)?)|pendir)|p(?:a(?:rse_(?:ini_file|str)|ssthru)|g_(?:connect|(?:execut|prepar)e|query)|hp(?:_(?:strip_whitespac|unam)e|info|version)|o(?:pen|six_(?:get(?:(?:e[gu]|g)id|login|pwnam)|kill|mk(?:fifo|nod)|ttyname))|r(?:eg_(?:match(?:_all)?|replace(?:_callback(?:_array)?)?|split)|int_r|oc_(?:(?:clos|nic|terminat)e|get_status|open))|utenv)|r(?:awurl(?:de|en)code|e(?:ad(?:_exif_data|dir|(?:gz)?file)|(?:gister_(?:shutdown|tick)|name)_function)|unkit_(?:constant_(?:add|redefine)|(?:function|method)_(?:add|copy|re(?:defin|nam)e)))|s(?:e(?:ssion_s(?:et_save_handler|tart)|t(?:_(?:e(?:rror|xception)_handler|include_path|magic_quotes_runtime)|defaultstub))|h(?:a1_file|ow_source)|implexml_load_(?:file|string)|ocket_c(?:onnect|reate)|pl_autoload_register|qlite_(?:(?:(?:array|single|unbuffered)_)?query|create_(?:aggregate|function)|exec|p?open)|tr(?:eam_(?:context_create|socket_client)|ipc?slashes|rev)|ystem)|u(?:[ak]?sort|n(?:pack|serialize)|rl(?:de|en)code)|var_dump)(?:/(?:*.**/|/.*)|#.*[\s\v]|\)*[\]*\)?[\s\v]*\(.*\)

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a

ModSecurity operator that specifies a regular expression to match against the selected variables.\n-(?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|REQUEST_FILENAME|ARGS_NAMES|ARGS|XML:/* @rx (?i)\b(?:s(?:e(?:t(?:_(?:e(?:xception|rror)_handler|magic_quotes_runtime|include_path)|defaultstub)|ssion_s(?:et_save_handler|tart))|qlite_(?:(?:(?:unbuffered|single|array)_)?query|create_(?:aggregate|function)|p?open|exec)|tr(?:eam_(?:context_create|socket_client)|ipc?slashes|rev)|implexml_load_(?:string|file)|ocket_c(?:onnect|reate)|h(?:ow_sourc|a1_fil)e|pl_autoload_register|ystem)|p(?:r(?:eg_(?:replace(?:_callback(?:_array)?)?|match(?:_all)?|split)|oc_(?:(?:terminat|clos|nic)e|get_status|open)|int_r)|o(?:six_(?:get(?:(?:e[gu]|g)id|login|pwnam)|mk(?:fifo|nod)|tyname|kill)|pen)|hp(?:_(?:strip_whitespac|unam)e|version|info)|g_(?:(?:execut|prepar)e|connect|query)|a(?:rse_(?:ini_file|str)|ssthru)|utenv)|r(?:unkit_(?:function_(?:re(?:defin|nam)e|copy|add)|method_(?:re(?:defin|nam)e|copy|add)|constant_(?:redefine|add))|e(?:(?:gister_(?:shutdown|tick)|name)_function|ad(?:(?:gz)?file|_exif_data|dir))|awurl(?:de|en)code)|i(?:mage(?:createfrom(?:(?:jpe|pn)g|x[bp]m|wbmp|gif)|(?:jpe|pn)g|g(?:d2?|if)|2?wbmp|xbm)|s_(?:(?:(?:execut|write?|read)ab|fi)le|dir)|ni_(?:get(?:_all)?|set)|terator_apply|ptcembed)|g(?:et(?:_(?:c(?:urrent_use|fg_va)r|meta_tags)|my(?:[gpu]id|inode)|(?:lastmo|cw)d|imagesize|env)|z(?:(?:(?:defla|wri)t|encod|fil)e|compress|open|read)|lob)|a(?:rray_(?:u(?:intersect(?:_u?assoc)?|diff(?:_u?assoc)?)|intersect_u(?:assoc|key)|diff_u(?:assoc|key)|filter|reduce|map)|ssert(?:_options)?)|h(?:tml(?:specialchars(?:_decode)?|_entity_decode|entities)|(?:ash(?:_(?:update|hmac))?|ighlight)_file|e(?:ader_register_callback|x2bin))|f(?:i(?:le(?:(?:[acm]tim|inod)e|(?:_exist|perm)s|group)?|nfo_open)|tp_(?:nb_(?:ge|pu)|connec|ge|pu)t|(?:unction_exis|pu)ts|write|open)|o(?:b_(?:get_(?:c(?:ontents|lean)|flush)|end_(?:clean|flush)|clean|flush|start)|dbc_(?:result(?:_all)?|exec(?:ute)?|connect)|pendir)|m(?:b_(?:ereg(?:_(?:replace(?:_callback)?|match)|i(?:_replace)?)?|parse_str)|(?:ove_uploaded|d5)_file|ethod_exists|ysql_query|kdir)|e(?:x(?:if_(?:t(?:humbnail|agname)|imagetype|read_data)|ec)|scapeshell(?:arg|cmd)|rror_reporting|val)|c(?:url_(?:file_create|exec|init)|onvert_uuencode|reate_function|hr)|u(?:n(?:serialize|pack)|rl(?:de|en)code|[ak]?sort)|(?:json_(?:de|en)cod|debug_backtrac|tmpfil)e|b(?:(?:son_(?:de|en)|

ase64_en)code|zopen)|var_dump)(?:\s|/*.**/|//.*|#.*)*\(.*\)

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:933210

Description

PHP Injection Attack: Variable Function Call Found

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|REQUEST_FILENAME|ARGS_NAMES|ARGS|XML:/* @rx (?:\((?:.+\)(?:[\][\-0-9A-Z_a-z]+[\])?\(.+|[^\)]*string[^\)]*\)[\s\v\\--\.0-9A-\[\]_a-\{\}]+\([^\)]*)|(?:\[[0-9]+\]|\{[0-9]+\}|\$[^\(-\)\.-/;\x5c]+|[\][\-0-9A-Z\x5c_a-z]+[\])\(.+)\);

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n-

(?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|REQUEST_FILENAME|ARGS_NAMES|ARGS|XML:/* @rx (?:(?:\(|\[)[a-zA-Z0-9_.$\\[\](){}/*\s]+(?:\)|\])[0-9_.$\\[\](){}/*\s]*\([a-zA-Z0-9_.$\\[\](){}/*\s].*\)|\([\s]*string[\s]*\)[\s]*(?:\|))

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:933131

Description

PHP Injection Attack: Variables Found

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx AUTH_TYPE|HTTP_(?:ACCEPT(?:_(?:CHARSET|ENCODING|LANGUAGE))?|CONNECTION|(?:H

OS|USER_AGEN)T|KEEP_ALIVE|(?:REFERE|X_FORWARDED_FO)R)|ORIG_PATH_INFO|PATH_(?:INFO|TRANSLATED)|QUERY_STRING|REQUEST_URI

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?:HTTP_(?:ACCEPT(?:_(?:ENCODING|LANGUAGE|CHARSET))?|(?:X_FORWARDED_FO|REFERE)R|(?:USER_AGEN|HOS)T|CONNECTION|KEEP_ALIVE)|PATH_(?:TRANSLATED|INFO)|ORIG_PATH_INFO|QUERY_STRING|REQUEST_URI|AUTH_TYPE)

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a

leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:933161
Description
PHP Injection Attack: Low-Value PHP Function Call Found
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|REQUEST_FILENAME|ARGS_NAMES|ARGS|XML:/* @rx (?i)\b(?:a(?:bs|cosh?|r(?:ray|sort)|s(?:inh?|(?:o|se)rt)|tan[2h]?)|b(?:asename|indec)|c(?:eil|h(?:dir|eckdate|mod|o(?:p|wn)|root)|lose(?:dir|log)|o(?:(?:mpac|(?:nsta|u)n)t|py|sh?)|(?:ryp|urren)t)|d(?:ate|e(?:coct|fined?)|i(?:(?:skfreespac)?e|r(?:name)?)|(?:oubleva)?l)|e(?:a(?:ch|ster_da(?:te|ys))|cho|mpty|nd|r(?:egi?|ror_log)|x(?:(?:i|trac)t|p(?:lode)?))|f(?:close|eof|gets|ile(?:owner|pro|(?:siz|typ)e)|l(?:o(?:atval|ck|or)|ush)|(?:mo|rea)d|stat|t(?:ell|ok)|unction)|g(?:et(?:date|t(?:ext|ype))|mdate)|h(?:ash|e(?:ader(?:s_(?:lis|sen)t)?|brev)|ypot)|i(?:conv|(?:dat|mplod)e|n(?:(?:clud|vok)e|t(?:div|val))|s(?:_(?:a(?:rray)?|bool|(?:calla|dou)ble|f(?:inite|loat)|in(?:finite|t(?:eger)?)|l(?:ink|ong)|n(?:an|u(?:ll|meric))|object|re(?:al|source)|s(?:calar|tring))|set))|join|k(?:ey|sort)|l(?:(?:cfirs|sta)t|evenshtein|i(?:nk(?:info)?|st)|o(?:caltime|g(?:1[0p])?)|trim)|m(?:a(?:i[ln]|x)|b(?:ereg|split)|etaphone|hash|i(?:crotime|n)|y?sql)|n(?:atsor|ex)t|o(?:ctdec|penlog|rd)|p(?:a(?:ck|thinfo)|close|i|o[sw]|r(?:ev|intf?))|quotemeta|r(?:an(?:d|ge)|e(?:adlin[ek]|(?:cod|nam|quir)e|set|wind)|ound|sort|trim)|s(?:(?:candi|ubst)r|(?:e(?:rializ|ttyp)|huffl)e|i(?:milar_text|nh?|zeof)|leep|o(?:rt|undex)|p(?:liti?|rintf)|qrt|rand|t(?:at|r(?:coll|(?:le|sp)n))|y(?:mlink|slog))|t(?:a(?:int|nh?)|e(?:mpnam|xtdomain)|ime|ouch|rim)|u(?:cfirst|mask|n(?:iqid|link|(?:se|tain)t)|s(?:leep|ort))|virtual|wordwrap)(?:[\s\v]|/(?:*.**/|/.*)|#.*)*\(.*\)
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a

ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|REQUEST_FILENAME|ARGS_NAMES|ARGS|XML:/* @rx (?i)\b(?:i(?:s(?:_(?:in(?:t(?:eger)?|finite)|n(?:u(?:meric|ll)|an)|(?:calla|dou)ble|s(?:calar|tring)|f(?:inite|loat)|re(?:source|al)|l(?:ink|ong)|a(?:rray)?|object|bool)|set)|n(?:(?:clud|vok)e|t(?:div|val))|(?:mplod|dat)e|conv)|s(?:t(?:r(?:(?:(?:le|sp)n|coll)|at)|(?:e(?:rializ|ttyp)|huffl)e|i(?:milar_text|zeof|nh?)|p(?:liti?|rintf)|(?:candi|ubst)r|y(?:mlink|slog)|o(?:undex|rt)|leep|rand|qrt)|f(?:ile(?:(?:siz|typ)e|owner|pro)|l(?:o(?:atval|ck|or)|ush)|(?:rea|mo)d|t(?:ell|ok)|unction|close|gets|stat|eof)|c(?:h(?:o(?:wn|p)|eckdate|root|dir|mod)|o(?:(?:(?:nsta|u)n|mpac)t|sh?|py)|lose(?:dir|log)|(?:urren|ryp)t|eil)|e(?:x(?:(?:trac|i)t|p(?:lode)?)|a(?:ster_da(?:te|ys)|ch)|r(?:ror_log|egi?)|mpty|cho|nd)|l(?:o(?:g(?:1[0p])?|caltime)|i(?:nk(?:info)?|st)|(?:cfirs|sta)t|evenshtein|trim)|d(?:i(?:(?:skfreespac)?e|r(?:name)?)|e(?:fined?|coct)|(?:oubleva)?l|ate)|r(?:e(?:(?:quir|cod|nam)e|adlin[ek]|wind|set)|an(?:ge|d)|ound|sort|trim)|m(?:b(?:split|ereg)|i(?:crotime|n)|a(?:i[ln]|x)|etaphone|y?sql|hash)|u(?:n(?:(?:tain|se)t|iqid|link)|s(?:leep|ort)|cfirst|mask)|a(?:s(?:(?:se|o)rt|inh?)|r(?:sort|ray)|tan[2h]?|cosh?|bs)|t(?:e(?:xtdomain|mpnam)|a(?:int|nh?)|ouch|ime|rim)|h(?:e(?:ader(?:s_(?:is|sen)t)?|brev)|ypot|ash)|p(?:a(?:thinfo|ck)|r(?:intf?|ev)|close|o[sw]|i)|g(?:et(?:t(?:ext|ype)|date)|mdate)|o(?:penlog|ctdec|rd)|b(?:asename|indec)|n(?:atsor|ex)t|k(?:sort|ey)|quotemeta|wordwrap|virtual|join)(?:\s|/*.**/|//.*|#.*)*\(.*\)

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n-

(?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:953120
Description
PHP source code leakage
Configured Variable
SecRule RESPONSE_BODY @rx (?i)<\?(?=|php)?\s+
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule RESPONSE_BODY "@rx <\?(?!xml)"
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a

ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:934100
Description
Node.js Injection Attack 1/2
Configured Variable
SecRule REQUEST_FILENAME|REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx _(?:\$\$ND_FUNC\$\$_|_js_function)|(?:\beval|new[\s\v]+Function[\s\v]*)\(|String\.fromCharCode|function\(\)\{|this\.constructor|module\.exports=|\([\s\v]*[^0-9A-Z_a-z]child_process[^0-9A-Z_a-z][\s\v]*\)|process(?:\.(?:(?:a(?:ccess|ppendfile|rgv|vailability)|c(?:aveats|h(?:mod|own)|(?:los|opyfil)e|p|reate(?:read|write)stream)|ex(?:ec(?:file)?|ists)|f(?:ch(?:mod|own)|data(?:sync)?|s(?:tat|ync)|utimes)|inodes|l(?:chmod|ink|stat|utimes)|mkd(?:ir|temp)|open(?:dir)?|r(?:e(?:ad(?:dir|file|link|v)?|name)|m)|s(?:pawn(?:file)?|tat|ymlink)|truncate|u(?:n(?:link|watchfile)|times)|w(?:atchfile|rite(?:file|v)?))(?:sync)?(?:\.call)?\(|binding|constructor|env|global|main(?:Module)?|process|require)|\[[\`](?:(?:a(?:ccess|ppendfile|rgv|vailability)|c(?:aveats|h(?:mod|own)|(?:los|opyfil)e|p|reate(?:read|write)stream)|ex(?:ec(?:file)?|ists)|f(?:ch(?:mod|own)|data(?:sync)?|s(?:tat|ync)|utimes)|inodes|l(?:chmod|ink|stat|utimes)|mkd(?:ir|temp)|open(?:dir)?|r(?:e(?:ad(?:dir|file|link|v)?|name)|m)|s(?:pawn(?:file)?|tat|ymlink)|truncate|u(?:n(?:link|watchfile)|times)|w(?:atchfile|rite(?:file|v)?))(?:sync)?|binding|constructor|env|global|main(?:Module)?|process|require)[\`]\])|(?:binding|constructor|env|global|main(?:Module)?|process|require)\[|console(?:\.(?:debug|error|info|trace|warn)(?:\.call)?\(|\[[\`](?:debug|error|info|trace|warn)[\`]\])|require(?:\.(?:resolve(?:\.call)?\(|main|extensions|cache)|\[[\`](?:(?:resolv|cach)e|main|extensions)[\`]\])
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n-

REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?:(?:_(?:\$\$ND_FUNC\$\$_|_js_function)|(?:new\s+Function|\beval)\s*\(|String\s*\.\s*fromCharCode|function\s*\(\s*\)\s*{|this\.constructor)|module\.exports\s*=)
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942140
Description
SQL Injection Attack: Common DB Names Detected
Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)\b(?:d(?:atabas|b_nam)e[^0-9A-Z_a-z]*\(|(?:information_schema|m(?:aster\.\.sysdatabases|s(?:db|ys(?:ac(?:cess(?:objects|storage|xml)|es)|modules2?|(?:object|querie|relationship)s))|ysql\.db)|northwind|pg_(?:catalog|toast)|tempdb)\b|s(?:chema(?:_name\b|[^0-9A-Z_a-z]*\()|(?:qlite_(?:temp_)?master|ys(?:aux|\.database_name))\b))

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:\b(?:(?:m(?:s(?:ys(?:ac(?:cess(?:objects|storage|xml)|es)|(?:relationship|object|querie)s|modules2?)|db)|aster\.\.sysdatabases|ysql\.db)|pg_(?:catalog|toast)|information_schema|northwind|tempdb)\b|s(?:(?:ys(?:\.database_name|aux)|qlite(?:_temp)?_master)\b|chema(?:_name\b|\W*\())|d(?:atabas|b_nam)e\W*\())

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n-

REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942160
Description
Detects blind sqli tests using sleep() or benchmark()
Configured Variable
SecRule REQUEST_BASENAME|REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:sleep\(\s*?\d*?\s*?\)|benchmark\(.*?\.*?\))
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAM

ES|ARGS|XML:/* @rx (?i:sleep\(\s*?\d*?\s*?\)|benchmark\(.*?\.*?\))

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942170

Description

Detects SQL benchmark and sleep injection attempts including conditional queries

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)(?:select|;)[\s\v]+(?:benchmark|if|sleep)[\s\v]*?\([\s\v]*?\(?[\s\v]*?[0-9A-Z_a-z]+

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression

itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:(?:select|;)\s+(?:benchmark|sleep|if)\s*?\(\s*?\(?\s*?\w+)

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942190

Description

Detects MSSQL code execution and information gathering attempts

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)[\`](?:[\s\v]*![\s\v]*[\0-9A-Z_-z]|;?[\s\v]*(?:having|select|union\b[\s\v]*(?:all|(?:distin|sele)ct))\b[\s\v]*[^\s\v])|\b(?:(?:(?:c(?:onnection_id|urrent_user)|database|schema|user)[\s\v]*?|select.*?[0-9A-Z_a-z]?user)\(|exec(?:ute)?[\s\v]+master\.|from[^0-9A-Z_a-z]+information_schema[^0-9A-Z_a-z]|into[\s\v\+]+(?:dump|out)file[\s\v]*?[\`]|union(?:[\s\v]select[\s\v]@|[\s\v\(0-9A-Z_a-z]*?select))|[\s\v]*?exec(?:ute)?.

?[^0-9A-Z_a-z]xp_cmdshell|[^0-9A-Z_a-z]iif[\s\v]?\(

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:(?:[\`](?:;?\s*?(?:having|select|union)\b\s*?[^\s]|\s*?!\s*?[\`\w])|(?:c(?:onnection_id|urrent_user)|database)\s*?\([^\)]*?|u(?:nion(?:[\w(\s]*?select| select @)|ser\s*?\([^\)]*?)|s(?:chema\s*?\([^\)]*?|elect.*?\w?user\()|into[\s+]+(?:dump|out)file\s*?[\`]|\s*?exec(?:ute)?.*?\Wxp_cmdshell|from\W+information_schema\W|exec(?:ute)?\s+master\.|\wiif\s*?\())

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a

leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942220
Description
Looking for integer overflow attacks these are taken from skipfish except 2.2.2250738585072011e-308 is the \magic number\ crash
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx ^(?i:-0000023456|4294967295|4294967296|2147483648|2147483647|0000012345|-2147483648|-2147483649|0000023456|2.2250738585072007e-308|2.2250738585072011e-308|1e309)$
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx ^(?i:-0000023456|4294967295|4294967296|2147483648|2147483647|0000012345|-2147483648|-2147483649|0000023456|3.0.00738585072007e-308|1e309)$

Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942230
Description
Detects conditional SQL injection attempts
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)[\s\v\(-\)]case[\s\v]+when.*?then|\)[\s\v]*?like[\s\v]*?\(|select.*?having[\s\v]*?[^\s\v]+[\s\v]*?[^\s\v0-9A-Z_a-z]|if[\s\v]?\([0-9A-Z_a-z]+[\s\v]*?[<->~]
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression

itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:[\s()]case\s*?\(|\)\s*?like\s*?\(|having\s*?[^\s]+\s*?[^\w\s]|if\s?\([\d\w]\s*?[=<>~])
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942240
Description
Detects MySQL charset switch and MSSQL DoS attempts
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)alter[\s\v]*?[0-9A-Z_a-z]+.*?char(?:acter)?[\s\v]+set[\s\v]+[0-9A-Z_a-z]+|[\`](?:;*?[\s\v]*?waitfor[\s\v]+(?:time|delay)[\s\v]+[\`];.*?:[\s\v]*?goto)
Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:(?:[\`](?:;*?\s*?waitfor\s+(?:delay|time)\s+[\`]|;.*?:\s*?goto)|alter\s*?\w+.*?cha(?:racte)?r\s+set\s+\w+))

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942280

Description
Detects Postgres pg_sleep injection waitfor delay attacks and database shutdown attempts
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)select[\s\v]*?pg_sleep|waitfor[\s\v]*?delay[\s\v]?[\`]+[\s\v]?[0-9]|;[\s\v]*?shutdown[\s\v]*?(?:[#;\{]|/*|--)
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:(?:;\s*?shutdown\s*?(?:[#;]|/*|--|\{)|waitfor\s*?delay\s?[\`]+\s?\d|select\s*?pg_sleep))
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule

should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942290
Description
Finds basic MongoDB SQL injection attempts
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)\[?\$(?:n(?:e|in?|o[rt])|e(?:q|xists|lemMatch)|l(?:te?|ike)|mod|a(?:ll|nd)|(?:s(?:iz|lic)|wher)e|t(?:ype|ext)|x?or|div|between|regex|jsonSchema)\]?
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAM

ES|ARGS|XML:/* @rx
(?i:(?:\[\$(?:ne|eq|lte?|gte?|n?in|mod|all|size|exists|type|slice|x?or|div|like|between|and)\]))

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942320

Description

Detects MySQL and PostgreSQL stored procedure/function injections

Configured Variable

SecRule
REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx
(?i)create[\s\v]+(?:function|procedure)[\s\v]*?[0-9A-Z_a-z]+[\s\v]*?\([\s\v]*?\)[\s\v]*?-|d(?:eclare[^0-9A-Z_a-z]+[#@][\s\v]*?[0-9A-Z_a-z]+|iv[\s\v]*?\([\+\-]*[\s\v\.0-9]+[\+\-]*[\s\v\.0-9]+\))|exec[\s\v]*?\([\s\v]*?@|(?:lo_(?:impor|ge)t|procedure[\s\v]+analyse)[\s\v]*?\(|;[\s\v]*?(?:declare|open)[\s\v]+[\-0-9A-Z_a-z]+|::(?:b(?:igint|ool)|double[\s\v]+precision|int(?:eger)?|numeric|oid|real|(?:tex|smallin)t)

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAM

ES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:(?:create\s+(?:procedure|function)\s*?\w+\s*?\(\s*?\)\s*?-|;\s*?(?:declare|open)\s+[\w-]+|procedure\s+analyse\s*?\(|declare[^\w]+[@#]\s*?\w+|exec\s*?\(\s*?\@))

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942350

Description

Detects MySQL UDF injection and other data/structure manipulation attempts

Configured Variable

SecRule

REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)create[\s\v]+function[\s\v].+[\s\v]returns|;[\s\v]*?(?:alter|(?:(?:cre|trunc|upd)at|renam)e|d(?:e(?:lete|sc)|rop)|(?:inser|selec)t|load)\b[\s\v]*?[\(\[]?[0-9A-Z_a-z]{2}

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:(?:;\s*?(?:(?:(?:trunc|cre|upd)at|renam)e|(?:inser|selec)t|de(?:lete|sc)|alter|load)\b\s*?[\[(]?\w{2}|create\s+function\s+.+\s+returns))

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression

itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942360
Description
Detects concatenated basic SQL injection and SQLLFI attempts
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)\b(?:(?:alter|(?:(?:cre|trunc|upd)at|renam)e|de(?:lete|sc)|(?:inser|selec)t|load)[\s\v]+(?:char|group_concat|load_file)\b[\s\v]*\(?|end[\s\v]*?\);)|[\s\v\(]load_file[\s\v]*?\(|[\`][\s\v]+regexp[^0-9A-Z_a-z]|[\0-9A-Z_-z][\s\v]+as\b[\s\v]*[\0-9A-Z_-z]+[\s\v]*\bfrom|^[^A-Z_a-z]+[\s\v]*?(?:(?:(?:(?:cre|trunc)at|renam)e|d(?:e(?:lete|sc)|rop)|(?:inser|selec)t|load)[\s\v]+[0-9A-Z_a-z]+|u(?:pdate[\s\v]+[0-9A-Z_a-z]+|nion[\s\v]*(?:all|(?:sele|distin)ct)\b)|alter[\s\v]*(?:a(?:(?:ggregat|pplication[\s\v]*rol)e|s(?:sembl|ymmetric[\s\v]*key)|u(?:dit|thorization)|vailability[\s\v]*group)|b(?:roker[\s\v]*priority|ufferpool)|c(?:ertificate|luster|o(?:l(?:latio|um)|nversio)n|r(?:edential|yptographic[\s\v]*provider))|d(?:atabase|efault|i(?:mension|skgroup)|omain)|e(?:(?:ndpoi|ve)nt|xte(?:nsion|rnal))|f(?:lashback|oreign|u(?:lltext|nction))|hi(?:erarchy|stogram)|group|in(?:dex(?:type)?|memory|stance)|java|l(?:a(?:ngua|r)ge|ibrary|o(?:ckdown|g(?:file[\s\v]*group|in)))|m(?:a(?:s(?:k|ter[\s\v]*key)|terialized)|e(?:ssage[\s\v]*type|thod)|odule)|(?:nicknam|queu)e|o(?:perator|utline)|p(?:a(?:ckage|rtition)|ermission|ro(?:cedur|fil)e)|r(?:e(?:mot|sourc)e|o(?:l(?:e|lback)|ute))|s(?:chema|e(?:arch|curity|rv(?:er|ice)|quence|ssion)|y(?:mmetric[\s\v]*key|nonym)|togroup)|t(?:able(?:space)?|ext|hreshold|r(?:igger|usted)|ype)|us(?:age|er)|view|w(?:ork(?:load)?|rapper)|x(?:ml[\s\v]*schema|srobject))\b)
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule

should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:(?:^[\W\d]+\s*?(?:(?:alter\s*(?:a(?:(?:pplication\s*rol|ggregat)e|s(?:ymmetric\s*ke|sembl)y|u(?:thorization|dit)|vailability\s*group)|c(?:r(?:yptographic\s*provider|edential)|o(?:l(?:latio|um)|nversio)n|ertificate|luster)|s(?:e(?:rv(?:ice|er)|curity|quence|ssion|arch)|y(?:mmetric\s*key|nonym)|togroup|chema)|m(?:a(?:s(?:ter\s*key|k)|terialized)|e(?:ssage\s*type|thod)|odule)|l(?:o(?:g(?:file\s*group|in)|ckdown)|a(?:ngua|r)ge|ibrary)|t(?:(?:abl(?:espac)?|yp)e|r(?:igger|usted)|hreshold|ext)|p(?:a(?:rtition|ckage)|ro(?:cedur|fil)e|ermission)|d(?:i(?:mension|skgroup)|atabase|efault|omain)|r(?:o(?:l(?:lback|e)|ute)|e(?:sourc|mot)e)|f(?:u(?:lltext|nction)|lashback|oreign)|e(?:xte(?:nsion|rnal)|(?:ndpoi|ve)nt)|in(?:dex(?:type)?|memory|stance)|b(?:roker\s*priority|ufferpool)|x(?:ml\s*schema|srobject)|w(?:ork(?:load)?|rapper)|hi(?:erarchy|stogram)|o(?:perator|utline)|(?:nicknam|queu)e|us(?:age|er)|group|java|view)|u(?:nion\s*(?:(?:distin|sele)ct|all)|pdate)|(?:truncat|renam)e|(?:inser|selec)t|de(?:lete|sc)|load)\b|create\s+\w+)|(?:(?:(?:trunc|cre|upd)at|renam)e|(?:inser|selec)t|de(?:lete|sc)|alter|load)\s+(?:group_concat|load_file|char)\s?\(?|[\d\W]\s+as\b\s*[\`\w]+\s*\bfrom|[\s(]load_file\s*?\(|[\`]\s+regexp\W|end\s*?\);))

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression

itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942110
Description
SQL Injection Attack: Common Injection Testing Detected
Configured Variable
SecRule REQUEST_FILENAME|ARGS_NAMES|ARGS|XML/* @rx (?:^\s*[\`;]+|[\`]+\s*$)
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule ARGS_NAMES|ARGS|XML/* @rx (?:^\s*[\`;]+|[\`]+\s*$)
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n-

(?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942120
Description
SQL Injection Attack: SQL Operator Detected
Configured Variable
SecRule ARGS_NAMES|ARGS|REQUEST_FILENAME|XML/* @rx (?i)!=|&&||||>[=->]|<(?:<|=>?|>(?:[\s\v]+binary)?)|\b(?:(?:xor|r(?:egexp|like)|i(?:snull|like)|notnull)\b|collate(?:[^0-9A-Z_a-z]*?(?:U&)?[\`]|[^0-9A-Z_a-z]+(?:(?:binary|nocase|rtrim)\b|[0-9A-Z_a-z]*?_))|(?:likel(?:ihood|y)|unlikely)[\s\v]*\(\)|r(?:egexp|like)[\s\v]+binary|not[\s\v]+between[\s\v]+(?:0[\s\v]+and|(?:[^]*|\[^\]*\)[\s\v]+and[\s\v]+(?:[^]*|\[^\]*\))|is[\s\v]+null|like[\s\v]+(?:null|[0-9A-Z_a-z]+[\s\v]+escape\b)|(?:^|[^0-9A-Z_a-z])in[\s\v\+]*\([\s\v\0-9]+[^\(-\)]*\)|[!<->]{12}[\s\v]*all\b
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule ARGS_NAMES|ARGS|XML/* @rx (?i:(?:(?:^|\W)in[+\s]*\([\s\d\]+[^()]*\)|\b(?:r(?:egexp|like)|isnull|xor)\b|<(?:>(?:\s+binary)?|=>?|<)|r(?:egexp|like)\s+binary|not\s+between\s+0\s+and|(?:like|is)\s+null|>[=>]||||!=|&&))

Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942130
Description
SQL Injection Attack: SQL Boolean-based attack detected
Configured Variable
SecRule ARGS_NAMES|ARGS|XML/* @rx (?i)[\s\v\-\)`]*?\b([0-9A-Z_a-z]+)\b[\s\v\-\)`]*?(?:=|<=>|(?:sounds[\s\v]+)?like|glob|r(?:like|egexp))[\s\v\-\)`]*?\b([0-9A-Z_a-z]+)\b
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or

argument value, this rule would trigger and block the request.

Recommended Variable

SecRule ARGS_NAMES|ARGS|XML/* @rx (?i:[\s\`()]*?\b([\d\w]+)\b[\s\`()]*?(?:<(?:=(?:[\s\`()]*?(?!\b\1\b)[\d\w]+|>[\s\`()]*?(?:\b\1\b))|>?[\s\`()]*?(?!\b\1\b)[\d\w]+)|(?:not\s+(?:regexp|like)|is\s+not|>=?|!=|\^)[\s\`()]*?(?!\b\1\b)[\d\w]+|(?:(?:sounds\s+)?like|r(?:egexp|like)|=)[\s\`()]*?(?:\b\1\b)))

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942150

Description

SQL Injection Attack: SQL function name detected

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)\b(?:json(?:_[0-9A-Z_a-z]+)?|a(?:bs|(?:cos|sin)h?|tan[2h]?|vg)|c(?:eil(?:ing)?|h(?:a(?:nges|r(?:set)?)|r)|o(?:alesce|sh?|unt)|ast)|d(?:e(?:grees|fault)|a(?:te|y))|exp|f(?:loor(?:avg)?|ormat|ield)|g(?:lob|roup_concat)|h(?:ex|our)|i(?:f(?:null)?|if|n(?:str)?)|l(?:ast(?:_insert_rowid)?|ength|ike(?:l(?:ihood|y))?|n|o(?:ad_extension|g(?:10|2)?|wer(?:pi)?|cal)|trim)|m(?:ax|in(?:ute)?|o(?:d|nth))|n(?:ullif|ow)|p(?:i|ow(?:er)?|rintf|assword)|quote|r(?:a(?:dians|ndom(?:blob)?)|e(?:p(?:lace|eat)|verse)|ound|trim|ight)|s(?:i(?:gn|

nh?)|oundex|q(?:lite_(?:compileoption_(?:get|used)|offset|source_id|version)|rt)|u(?:bstr(?:ing)?|m)|econd|leep)|t(?:anh?|otal(?:_changes)?|r(?:im|unc)|ypeof|ime)|u(?:n(?:icode|likely)|(?:pp|s)er)|zeroblob|bin|v(?:alues|ersion)|week|year)[^0-9A-Z_a-z]*\(

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)\b(?:c(?:o(?:n(?:v(?:ert(?:_tz)?)?|cat(?:_ws)?|nection_id)|(?:mpres)?s|ercibility|(?:un)?t|llation|alesce)|ur(?:rent_(?:time(?:stamp)?|date|user)|(?:dat|tim)e)|h(?:ar(?:(?:acter)?_length|set)?|r)|iel(?:ing)?|ast|r32)|s(?:u(?:b(?:str(?:ing(?:_index)?)?|(?:dat|tim)e)|m)|t(?:d(?:dev_(?:sam|po)p)?|r(?:_to_date|cmp))|e(?:c(?:_to_time|ond)|ssion_user)|ys(?:tem_user|date)|ha[12]?|oundex|chema|ig?n|leep|pace|qrt)|i(?:s(?:_(?:ipv(?:4(?:_(?:compat|mapped))?|6)|n(?:ot(?:_null)?|ull)|(?:free|used)_lock)|null)|n(?:et(?:6_(?:aton|ntoa)|_(?:aton|ntoa))|s(?:ert|tr)|terval)?|f(?:null)?)|d(?:a(?:t(?:e(?:_(?:format|add|sub)|diff)?|abase)|y(?:of(?:month|week|year)|name)?)|e(?:(?:s_(?:de|en)cryp|faul)t|grees|code)|count|ump)|l(?:o(?:ca(?:l(?:timestamp)?|te)|g(?:10|2)?|ad_file|wer)|ast(?:_(?:inser_id|day))?|e(?:(?:as|f)t|ngth)|case|trim|pad|n)|u(?:n(?:compress(?:ed_length)?|ix_timestamp|hex)|tc_(?:time(?:stamp)?|date)|p(?:datexml|per)|uid(?:_short)?|case|ser)|t(?:ime(?:_(?:format|to_sec)|stamp(?:diff|add)?|diff)?|o(?:(?:second|day)s|_base64|n?char)|r(?:uncate|im)|an)|m(?:a(?:ke(?:_set|date)|ster_pos_wait|x)|i(?:(?:crosecon)?d|n(?:ute)?)|o(?:nth(?:name)?|d)|d5)|r(?:e(?:p(?:lace|eat)|lease_lock|verse)|a(?:wtohex|dians|nd)|o(?:w_co

unt|und)|ight|trim|pad)|f(?:i(?:eld(?:_in_set)?|nd_in_set)|rom_(?:unixtime|base64|days)|o(?:und_rows|rmat)|loor)|p(?:o(?:w(?:er)?|sition)|eriod_(?:diff|add)|rocedure_analyse|assword|g_sleep|i)|a(?:s(?:cii(?:str)?|in)|es_(?:de|en)crypt|dd(?:dat|tim)e|(?:co|b)s|tan2?|vg)|b(?:i(?:t_(?:length|count|x?or|and)|n(?:_to_num)?)|enchmark)|e(?:x(?:tract(?:value)?|p(?:ort_set)?)|nc(?:rypt|ode)|lt)|g(?:r(?:oup_conca|eates)t|et_(?:format|lock))|v(?:a(?:r(?:_(?:sam|po)p|iance)|lues)|ersion)|o(?:(?:ld_passwo)?rd|ct(?:et_length)?)|we(?:ek(?:ofyear|day)?|ight_string)|n(?:o(?:t_in|w)|ame_const|ullif)|h(?:ex(?:toraw)?|our)|qu(?:arter|ote)|year(?:week)?|xmltype)\W*\(

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942180

Description

Detects basic SQL authentication bypass attempts 1/3

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)(?:/*)+[\`]+[\s\v]?(?:--|[#\{]|/\*)?|[\`](?:[\s\v]*(?:(?:x?or|and|div|like|between)[\s\v\-0-9A-Z_a-z]+[\(-\)\+-\-<->][\s\v]*[\0-9`]||[!=\|](?:[\s\v -!\+\-0-9=]+.*?[\-\(`].*?|[\s\v -!0-9=]+.*?[0-9]+)$|(?:like|print)[^0-9A-Z_a-z]+[\-\(0-9A-Z_-z]|;)|(?:[<>~]+|[\s\v]*[^\s\v0-9A-Z_a-z]?=[\s\v]*|[^0-9A-Z_a-z]*?[\+=]+[^0-9A-Z_a-z]*?)[\`])|[0-9][\`][\s\v]+[\`][\s\v]+[0-9]|^admin[\s\v]*?[\`]|[\s\v\-\(`][\s

\v]*?glob[^0-9A-Z_a-z]+[\-\(0-9A-Z_-z]|[\s\v]is[\s\v]*?0[^0-9A-Z_a-z]|where[\s\v][\s\v-\.0-9A-Z_a-z]+[\s\v]=

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:[\`](?:\s*?(?:(?:between|x?or|and|div)[\w\s-]+\s*?[+<>=()-]\s*?[\d\`]|like(?:[\w\s-]+\s*?[+<>=()-]\s*?[\d\`]|\W+[\w\`(])|[!=|](?:[\d\s!=+-]+.*?[\`(].*?|[\d\s!=]+.*?\d+)$|[^\w\s]?=\s*?[\`])|(?:\W*?[+=]+\W*?|[<>~]+)[\`])|(?:/\*)+[\`]+\s?(?:\/*|--|\{|#)?|\d[\`]\s+[\`]\s+\d|where\s[\s\w\.-]+\s=|^admin\s*?[\`]|\sis\s*?0\W)

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a

leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942200
Description
Detects MySQL comment-/space-obfuscated injections and backtick termination
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|REQUEST_HEADERS:User-Agent|REQUEST_HEADERS:Referer|ARGS_NAMES|ARGS|XML:/* @rx (?i).*?[\\)0-9`-f][\`](?:[\`].*?[\`]|(?:\r?\n)?\z|[^\`]+)|[^0-9A-Z_a-z]select.+[^0-9A-Z_a-z]*?from|(?:alter|(?:(?:cre|trunc|upd)at|renam)e|d(?:e(?:lete|sc)|rop)|(?:inser|selec)t|load)[\s\v]*?\([\s\v]*?space[\s\v]*?\(
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:(?:(?:(?:(?:trunc|cre|upd)at|renam)e|(?:inser|selec)t|de(?:lete|sc)|alter|load)\s*?\(\s*?space\s*?\(|.*?[)\da-f\`][\`](?:[\`].*?[\`]|(?:\r?\n)?\z|[^\`]+)|\Wselect.+\W*?from))
Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942210
Description
Detects chained SQL injection attempts 1/2
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)(?:&&||||and|between|div|like|n(?:and|ot)|(?:xx?)?or)[\s\v\(]+[0-9A-Z_a-z]+[\s\v\)]*?[!\+=]+[\s\v0-9]*?[\-\)=`]|[0-9](?:[\s\v]*?(?:and|between|div|like|x?or)[\s\v]*?[0-9]+[\s\v]*?[\+\-]|[\s\v]+group[\s\v]+by.+\()|/[0-9A-Z_a-z]+;?[\s\v]+(?:and|between|div|having|like|x?or|select)[^0-9A-Z_a-z]|(?:[#;]|--)[\s\v]*?(?:alter|drop|(?:insert|update)[\s\v]*?[0-9A-Z_a-z]{2})|@.+=[\s\v]*?\([\s\v]*?select|[^0-9A-Z_a-z]SET[\s\v]*?@[0-9A-Z_a-z]+
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a

ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:(?:(?:n(?:and|ot)|(?:x?x)?or|between|\|\||like|and|div|&&)[\s(]+\w+[\s)]*?[!=+]+[\s\d]*?[\`=()]|\d(?:\s*?(?:between|like|x?or|and|div)\s*?\d+\s*?[\-+]|\s+group\s+by.+\()|\/\w+;?\s+(?:between|having|select|like|x?or|and|div)\W|--\s*?(?:(?:insert|update)\s*?\w{2}|alter|drop)|#\s*?(?:(?:insert|update)\s*?\w{2}|alter|drop)|;\s*?(?:(?:insert|update)\s*?\w{2}|alter|drop)|\@.+=\s*?\(\s*?select|[^\w]SET\s*?\@\w+))

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942260

Description

Detects basic SQL authentication bypass attempts 2/3

Configured Variable

SecRule

REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)[\`][\s\v]*?(?:(?:and|n(?:and|ot)|(?:xx?)?or|div|like|between|\|\||&&)[\s\v]+[\s\v0-9A-Z_a-z]+=[\s\v]*?[0-9A-Z_a-z]+[\s\v]*?having[\s\v]+|like[^0-9A-Z_a-z]*?[\0-9`])|[0-9A-Z_a-z][\s\v]+like[\s\v]+[\`]|like[\s\v]*?[\`]%|select[\s\v]+?[\s\v\-\)-\.0-9A-\[\]_-z]+from[\s\v]+

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:[\`]\s*?(?:(?:n(?:and|ot)|(?:x?x)?or|between|\|\||and|div|&&)\s+[\s\w]+=\s*?\w+\s*?having\s+|like(?:\s+[\s\w]+=\s*?\w+\s*?having\s+|\W*?[\`\d])|[^?\w\s=.;)(]++\s*?[(@\`]*?\s*?\w+\W+\w|\*\s*?\w+\W+[\`])|(?:union\s*?(?:distinct|[(!@]*?|all)?\s*?([[]*?\s*?select|select\s+?[\[\]()\s\w\.\`-]+from)\s+|\w\s+like\s+[\`]|find_in_set\s*?\(|like\s*?[\`]%)

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule

should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942300
Description
Detects MySQL comments conditions and ch(a)r injections
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)\)[\s\v]*?when[\s\v]*?[0-9]+[\s\v]*?then|[\`][\s\v]*?(?:[#\{]|--)|/*![\s\v]?[0-9]+|\b(?:b(?:inary[\s\v]*?\([\s\v]*?[0-9]|etween[\s\v]+[\s\v]*?[0-9A-Z_a-z]+\()|cha?r[\s\v]*?\([\s\v]*?[0-9]|(?:and|n(?:and|ot)|(?:xx?)?or|div|like|r(?:egexp|like))[\s\v]+[\s\v]*?[0-9A-Z_a-z]+\()|(?:||&&)[\s\v]+[\s\v]*?[0-9A-Z_a-z]+\(
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule

REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:(?:(?:n(?:and|ot)|(?:x?x)?or|between|\|\||like|and|div|&&)\s+\s*?\w+\(|\)\s*?when\s*?\d+\s*?then|[\`]\s*?(?:--|\{|#)|cha?r\s*?\(\s*?\d|\/*!\s?\d+))

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942310

Description

Detects chained SQL injection attempts 2/2

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)(?:\([\s\v]*?select[\s\v]*?[0-9A-Z_a-z]+|coalesce|order[\s\v]+by[\s\v]+if[0-9A-Z_a-z]*?)[\s\v]*?\(|*/from|\+[\s\v]*?[0-9]+[\s\v]*?\+[\s\v]*?@|[0-9A-Z_a-z][\`][\s\v]*?(?:(?:[\+\-=@\|]+[\s\v]+?)+|[\+\-=@\|]+)[\(0-9]|@@[0-9A-Z_a-z]+[\s\v]*?[^\s\v0-9A-Z_a-z]|[^0-9A-Z_a-z]!+[\`][0-9A-Z_a-z]|[\`](?:;[\s\v]*?(?:if|while|begin)|[\s\v0-9]+=[\s\v]*?[0-9])|[\s\v\(]+case[0-9]*?[^0-9A-Z_a-z].+[tw]hen[\s\v\(]

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web

application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:(?:[\`](?:;\s*?(?:begin|while|if)|[\s\d]+=\s*?\d|\s+and\s*?=\W)|(?:\(\s*?select\s*?\w+|order\s+by\s+if\w*?|coalesce)\s*?\(|\w[\`]\s*?(?:(?:[-+=|@]+\s+?)+|[-+=|@]+)[\d(]|[\s(]+case\d*?\W.+[tw]hen[\s(]|\+\s*?\d+\s*?\+\s*?\@|\@\@\w+\s*?[^\w\s]|\W!+[\`]\w|*\/from))

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942330

Description

Detects classic SQL injection probings 1/3

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)[\`][\s\v]*?(?:x?or|div|like|between|and)[\s\v]*?[\`]?[0-9]|\x5cx(?:2[37]|3d)|^(?:.?[\`]$|[\\x5c`]*?(?:[\0-9`]+|[^\`]+[\`])[\s\v]*?(?:and|n(?:and|ot)|(?:xx?)?or|div|like|between|\|\||&&)[\s\v]*?[\0-9A-Z_-z][!&\(-\)\+-\.@])|[^\s\v0-9A-Z_a-z][0-9A-Z_a-z]+[\s\v]*?[\-\|][\s\v]*?[\`][\s\v]*?[0-9A-Z_a-z]|@(?:[0-9A-Z_a-z]+[\s\v]+(?:and|x?or|div|like|between)[\s\v]*?[\0-9`]+|[\-0-9A-Z_a-z]+[\s\v](?:and|x?or|div|like|between)[\s\v]*?[^\s\v0-9A-Z_a-z])|[^\s\v0-:A-Z_a-z][\s\v]*?[0-9][^0-9A-Z_a-z]+[^\s\v0-9A-Z_a-z][\s\v]*?[\`].|[^0-9A-Z_a-z]information_schema|table_name[^0-9A-Z_a-z]

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:(?:^(?:[\`\\\\]*?(?:[^\`]+[\`]|[\d\`]+)\s*?(?:n(?:and|ot)|(?:x?x)?or|between|\|\||like|and|div|&&)\s*?[\w\`][+&!@().-]|.?[\`]$)|\@(?:[\w-]+\s(?:between|like|x?or|and|div)\s*?[^\w\s]|\w+\s+(?:between|like|x?or|and|div)\s*?[\`\d]+)|[\`]\s*?(?:between|like|x?or|and|div)\s*?[\`]?\d|[^\w\s:]\s*?\d\W+[^\w\s]\s*?[\`].|[^\w\s]\w+\s*?[|-]\s*?[\`]\s*?\w|\Winformation_schema|\\\\x(?:23|27|3d)|table_name\W))

Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942340
Description
Detects basic SQL authentication bypass attempts 3/3
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)in[\s\v]*?\(+[\s\v]*?select|(?:(?:(?i:N)?AND|(?i:X)?(?i:X)?OR|DIV|LIKE|BETWEEN|NOT)[\s\v]+|(?:||&&)[\s\v]*)[\s\v\+0-9A-Z_a-z]+(?:regexp[\s\v]*?\(|sounds[\s\v]+like[\s\v]*?[\`]|[0-9=]+x)|[\`](?:[\s\v]*?(?:[0-9][\s\v]*?(?:--|#)|is[\s\v]*?(?:[0-9].+[\`]?[0-9A-Z_a-z]|[\.0-9]+[\s\v]*?[^0-9A-Z_a-z].*?[\`]))|[%-&<->\^]+[0-9][\s\v]*?(?:=|x?or|div|like|between|and)|(?:[^0-9A-Z_a-z]+[\+\-0-9A-Z_a-z]+[\s\v]*?=[\s\v]*?[0-9][^0-9A-Z_a-z]+||?[\-0-9A-Z_a-z]{3}[^\s\v\.0-9A-Z_a-z]+)[\`]|[\s\v]*(?:(?:(?:(?i:N)?AND|(?i:X)?(?i:X)?OR|DIV|LIKE|BETWEEN|NOT)[\s\v]+|(?:||&&)[\s\v]*)(?:array[\s\v]*\[|[0-9A-Z_a-z]+(?:[\s\v]*!?~|[\s\v]+(?:not[\s\v]+)?similar[\s\v]+to[\s\v]+)|(?:tru|fals)e\b))|\bexcept[\s\v]+(?:select\b|values[\s\v]*?\(
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n-

REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:(?:[\`](?:\s*?(?:is\s*?(?:[\d.]+\s*?\W.*?[\`]|\d.+[\`]?\w)|\d\s*?(?:--|#))|(?:\W+[\w+-]+\s*?=\s*?\d\W+|\|?[\w-]{3}[^\w\s.]+)[\`]|[\%&<>^=]+\d\s*?(?:between|like|x?or|and|div|=))|(?i:n?and|x?x?or|div|like|between|not|\|\||\&\&)\s+[\s\w+]+(?:sounds\s+like\s*?[\`]|regexp\s*?\(|[=\d]+x)|in\s*?\(+\s*?select))

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942370

Description

Detects classic SQL injection probings 2/3

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|REQUEST_HEADERS:Referer|REQUEST_HEADERS:User-Agent|ARGS_NAMES|ARGS|XML:/* @rx (?i)[\`](?:[\s\v]*?(?:(?:\*.+(?:x?or|div|like|between|(?:an|i)d)[^0-9A-Z_a-z]*?[\`]|(?:x?or|div|like|between|and)[\s\v][^0-9]+[\-0-9A-Z_a-z]+.*?)[0-9]|[^\s\v0-9\?A-Z_a-z]+[\s\v]*?[^\s\v0-9A-Z_a-z]+[\s\v]*?[\`]|[^\s\v0-9A-Z_a-z]+[\s\v]*?[^A-Z_a-z].*?(?:#|--))|.*?\*[\s\v]*?[0-9])|\^[\`][%\(-\+\-<>][\-0-9A-Z_a-z]+[^\s\v0-9A-Z_a-z]+[\`][^]

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:[\`](?:\s*?(?:(?:\*.+(?:(?:an|i)d|between|like|x?or|div)\W*?[\`]|(?:between|like|x?or|and|div)\s[^\d]+[\w-]+.*?)\d|[^\w\s?]+\s*?[^\w\s]+\s*?[\`]|[^\w\s]+\s*?[\W\d].*?(?:--|#))|.*?\*\s*?\d)|[()\*<>%+-][\w-]+[^\w\s]+[\`][^]|\^[\`])

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n-

REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942380
Description
SQL Injection Attack
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)\b(?:having\b(?:[\s\v]+(?:[0-9]{110}|[^=]{110})[\s\v]*?[<->]| ?(?:[0-9]{110} ?[<->]+|[\][^=]{110}[\<-\?\[]+))|ex(?:ecute(?:\(|[\s\v]{15}[\$\.0-9A-Z_a-z]{15}[\s\v]{03})|ists[\s\v]*?\([\s\v]*?select\b)|(?:create[\s\v]+?table.{020}?|like[^0-9A-Z_a-z]*?char[^0-9A-Z_a-z]*?)\()|select.*?case|from.*?limit|order[\s\v]by|exists[\s\v](?:[\s\v]select|s(?:elect[^\s\v](?:if(?:null)?[\s\v]\(|top|concat)|ystem[\s\v]\()|\bhaving\b[\s\v]+[0-9]{110}|[^=]{110})
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a

leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?:\b(?:having\b ?(?:[\\][^=]{110}[\\ ?[=<>]+|\d{110} ?[=<>]+)|(?i:having)\b\s+(?:[^=]{110}|\d{110})\s*?[=<>])|exists\s(?:s(?:elect\S(?:if(?:null)?\s\(|concat|top)|ystem\s\()|\b(?i:having)\b\s+\d{110}|[^=]{110}|\sselect)|(?i:\bexecute\s{15}[\w\.$]{15}\s{03})|(?i:\bcreate\s+?table.{020}?\()|(?i:\blike\W*?char\W*?\()|(?i:select.*?case)|(?i:from.*?limit)|(?i:\bexecute\()|(?i:order\sby))

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942390

Description

SQL Injection Attack

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx

(?i)\b(?:or\b(?:[\s\v]?(?:[0-9]{110}|[\][^=]{110}[\])[\s\v]?[<->]+|[\s\v]+(?:[0-9]{110}|[^=]{110})(?:[\s\v]*?[<->])?)|xor\b[\s\v]+(?:[0-9]{110}|[^=]{110})(?:[\s\v]*?[<->])?)|[\s\v]+x?or[\s\v]+.{120}[!\+\-<->]

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?:\b(?:(?i:xor)\b\s+(?:[^=]{110}(?:\s*?[=<>])?|\d{110}(?:\s*?[=<>])?)|(?i:or)\b\s+(?:[^=]{110}(?:\s*?[=<>])?|\d{110}(?:\s*?[=<>])?))|(?i:\bor\b ?[\\][^=]{110}[\\] ?[=<>]+)|(?i:\s+xor\s+.{120}[+\-!<>=])|(?i:\s+or\s+.{120}[+\-!<>=])|(?i:\bor\b ?\d{110} ?[=<>]+))

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a

leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942400
Description
SQL Injection Attack
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)\band\b(?:[\s\v]+(?:[0-9]{110}[\s\v]*?[<->]|[^=]{110})| ?(?:[0-9]{110}|[\][^=]{110}[\]) ?[<->]+)
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:\band\b(?:\s+(?:[^=]{110}(?:\s*?[=<>])?|\d{110}(?:\s*?[=<>])?)| ?(?:[\\][^=]{110}[\\]|\d{110}) ?[=<>]+))
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe

regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942410
Description
SQL Injection Attack
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)\b(?:a(?:(?:b|co)s|dd(?:dat|tim)e|es_(?:de|en)crypt|s(?:in|cii(?:str)?)|tan2?|vg)|b(?:enchmark|i(?:n(?:_to_num)?|t_(?:and|count|length|x?or)))|c(?:ast|h(?:ar(?:(?:acter)?_length|set)?|r)|iel(?:ing)?|o(?:alesce|ercibility|(?:mpres)?s|n(?:cat(?:_ws)?|nection_id|v(?:ert(?:_tz)?)?)|(?:un)?t)|r32|ur(?:(?:dat|tim)e|rent_(?:date|time(?:stamp)?|user)))|d(?:a(?:t(?:abase|e(?:_(?:add|format|sub)|diff)?)|y(?:name|of(?:month|week|year))?)|count|e(?:code|(?:faul|s_(?:de|en)cryp)t|grees)|ump)|e(?:lt|nc(?:ode|rypt)|x(?:p(?:ort_set)?|tract(?:value)?))|f(?:i(?:eld(?:_in_set)?|nd_in_set)|loor|o(?:rmat|und_rows)|rom_(?:base64|days|unixtime))|g(?:et_(?:format|lock)|r(?:eates|oup_concat)|h(?:ex(?:toraw)?|our)|i(?:f(?:null)?|n(?:et6?_(?:aton|ntoa)|s(?:ert|tr)|terval)?|s(?:_(?:(?:free|used)_lock|ipv(?:4(?:_(?:compat|mapped))?|6)|n(?:ot(?:_null)?|ull))|null)?)|l(?:ast(?:_(?:day|insert_id))?|case|e(?:(?:as|f)t|ngth)|n|o(?:ad_file|ca(?:l(?:timestamp)?|te)|g(?:10|2)?|wer)|pad|trim)|m(?:a(?:ke(?:date|_set)|ster_pos_wait|x)|d5|i(?:(?:crosecon)?d|n(?:ute)?)|o(?:d|nth(?:name)?))|n(?:ame_const|o(?:t_in|w)|ullif)|o(?:ct(?:et_length)?|(?:ld_passwo)?rd)|p(?:assword|eriod_(?:add|diff)|g_sleep|i|o(?:sition|w(?:er)?)|rocedure_analyse)|qu(?:arter|ote)|r(?:a(?:dians|nd|wto(?:hex|nhex(?:toraw)?))|e(?:lease_lock|p(?:eat|lace)|verse)|ight|o(?:und|w_count)|pad|

trim)|s(?:chema|e(?:c(?:ond|_to_time)|ssion_user)|ha[1-2]?|ig?n|leep|oundex|pace|qrt|t(?:d(?:dev(?:_(?:po|sam)p)?)?|r(?:cmp|_to_date))|u(?:b(?:(?:dat|tim)e|str(?:ing(?:_index)?)?)|m)|ys(?:date|tem_user))|t(?:an|ime(?:diff|_(?:format|to_sec)|stamp(?:add|diff)?)?|o_(?:base64|n?char|(?:day|second)s)|r(?:im|uncate))|u(?:case|n(?:compress(?:ed_length)?|hex|ix_timestamp)|p(?:datexml|per)|ser|tc_(?:date|time(?:stamp)?)|uid(?:_short)?)|v(?:a(?:lues|r(?:iance|_(?:po|sam)p))|ersion)|we(?:ek(?:day|ofyear)?|ight_string)|xmltype|year(?:week)?)[^0-9A-Z_a-z]*?\(

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:\b(?:c(?:o(?:n(?:v(?:ert(?:_tz)?)?|cat(?:_ws)?|nection_id)|(?:mpres)?s|ercibility|(?:un)?t|alesce)|ur(?:rent_(?:time(?:stamp)?|date|user)|(?:dat|tim)e)|h(?:ar(?:(?:acter)?_length|set)?|r)|iel(?:ing)?|ast|r32)|s(?:t(?:d(?:dev(?:_(?:sam|po)p)?)?|r(?:_to_date|cmp))|u(?:b(?:str(?:ing(?:_index)?)?|(?:dat|tim)e)|m)|e(?:c(?:_to_time|ond)|ssion_user)|ys(?:tem_user|date)|ha[12]?|oundex|chema|ig?n|leep|pace|qrt)|i(?:s(?:_(?:ipv(?:4(?:_(?:compat|mapped))?|6)|n(?:ot(?:_null)?|ull)|(?:free|used)_lock)|null)?|n(?:et(?:6_(?:aton|ntoa)|_(?:aton|ntoa))|s(?:ert|tr)|terval)?|f(?:null)?)|d(?:a(?:t(?:e(?:_(?:format|add|sub)|diff)?|abase)|y(?:of(?:month|week|year)|name)?)|e(?:(?:s_(?:de|en)cryp|faul)t|grees|code)|count|ump)|l(?:o(?:ca(?:l(?:timestamp)?|te)|g(?:10|2)?|ad_file|wer)|ast(?:_(?:insert_id|day))?|e(?:(?:as|f)t|ngth)|case|trim|pad|n)|u(?:n(?:compress(?:ed_length)?|ix_timestamp|hex)|tc_(?:time(?:stamp)?|date)|p(?:datexml|per)

|uid(?:_short)?|case|ser)|r(?:a(?:wto(?:nhex(?:toraw)?|hex)|dians|nd)|e(?:p(?:lace|eat)|lease_lock|verse)|o(?:w_count|und)|ight|trim|pad)|t(?:ime(?:_(?:format|to_sec)|stamp(?:diff|add)?|diff)?|o_(?:(?:second|day)s|base64|n?char)|r(?:uncate|im)|an)|m(?:a(?:ke(?:_set|date)|ster_pos_wait|x)|i(?:(?:crosecon)?d|n(?:ute)?)|o(?:nth(?:name)?|d)|d5)|f(?:i(?:eld(?:_in_set)?|nd_in_set)|rom_(?:unixtime|base64|days)|o(?:und_rows|rmat)|loor)|p(?:o(?:w(?:er)?|sition)|eriod_(?:diff|add)|rocedure_analyse|assword|g_sleep|i)|a(?:s(?:cii(?:str)?|in)|es_(?:de|en)crypt|dd(?:dat|tim)e|(?:co|b)s|tan2?|vg)|b(?:i(?:t_(?:length|count|x?or|and)|n(?:_to_num)?)|enchmark)|e(?:x(?:tract(?:value)?|p(?:ort_set)?)|nc(?:rypt|ode)|lt)|g(?:r(?:oup_conca|eates)t|et_(?:format|lock))|v(?:a(?:r(?:_(?:sam|po)p|iance)|lues)|ersion)|o(?:(?:ld_passwo)?rd|ct(?:et_length)?)|we(?:ek(?:ofyear|day)?|ight_string)|n(?:o(?:t_in|w)|ame_const|ullif)|h(?:ex(?:toraw)?|our)|qu(?:arter|ote)|year(?:week)?|xmltype)\W*?\(

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942470

Description

SQL Injection Attack

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)autonomous_transaction|(?:current_use|n?varcha|tbcreato)r|db(?:a_users|ms_java)|open(?:owa_

util|query|rowset)|s(?:p_(?:(?:addextendedpro|sqlexe)c|execute(?:sql)?|help|is_srvrolemember|makewebtask|oacreate|p(?:assword|repare)|replwritetovarbin)|ql_(?:longvarchar|variant))|utl_(?:file|http)|xp_(?:availablemedia|(?:cmdshel|servicecontro)l|dirtree|e(?:numdsn|xecresultset)|filelist|loginconfig|makecab|ntsec(?:_enumdomains)?|reg(?:addmultistring|delete(?:key|value)|enum(?:key|value)s|re(?:ad|movemultistring)|write)|terminate(?:_process)?)

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:(?:xp_(?:reg(?:re(?:movemultistring|ad)|delete(?:value|key)|enum(?:value|key)s|addmultistring|write)|(?:servicecontro|cmdshel)l|e(?:xecresultset|numdsn)|ntsec(?:_enumdomains)?|terminate(?:_process)?|availablemedia|loginconfig|filelist|dirtree|makecab)|s(?:p_(?:(?:addextendedpro|sqlexe)c|p(?:assword|repare)|replwritetovarbin|is_srvrolemember|execute(?:sql)?|makewebtask|oacreate|help)|ql_(?:longvarchar|variant))|open(?:owa_util|rowset|query)|(?:n?varcha|tbcreato)r|autonomous_transaction|db(?:a_users|ms_java)|utl_(?:file|http)))

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n-

REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942480
Description
SQL Injection Attack
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|REQUEST_HEADERS|ARGS_NAMES|ARGS|XML:/* @rx (?i)\b(?:(?:d(?:bms_[0-9A-Z_a-z]+\.|elete\b[^0-9A-Z_a-z]*?\bfrom)|(?:group\b.*?\bby\b.{1100}?\bhav|overlay\b[^0-9A-Z_a-z]*?\(.*?\b[^0-9A-Z_a-z]*?plac)ing|in(?:ner\b[^0-9A-Z_a-z]*?\bjoin|sert\b[^0-9A-Z_a-z]*?\binto|to\b[^0-9A-Z_a-z]*?\b(?:dump|out)file)|load\b[^0-9A-Z_a-z]*?\bdata\b.*?\binfile|s(?:elect\b.{1100}?\b(?:(?:.*?\bdump\b.*|(?:count|length)\b.{1100}?)\bfrom|(?:data_typ|from\b.{1100}?\bwher)e|instr|to(?:_(?:cha|numbe)r|p\b.{1100}?\bfrom))|ys_context)|u(?:nion\b.{1100}?\bselect|tl_inaddr))\b|print\b[^0-9A-Z_a-z]*?@@)|(?:collation[^0-9A-Z_a-z]*?\(a|@@version|;[^0-9A-Z_a-z]*?\b(?:drop|shutdown))\b|(?:dbo|msdasql|s(?:a|qloledb))
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n-

(?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:(?:\b(?:(?:s(?:elect\b.{1100}?\b(?:(?:(?:length|count)\b.{1100}?|.*?\bdump\b.*)\bfrom|to(?:p\b.{1100}?\bfrom|_(?:numbe|cha)r)|(?:from\b.{1100}?\bwher|data_typ)e|instr)|ys_context)|in(?:to\b\W*?\b(?:dump|out)file|sert\b\W*?\binto|ner\b\W*?\bjoin)|u(?:nion\b.{1100}?\bselect|tl_inaddr)|group\b.*?\bby\b.{1100}?\bhaving|d(?:elete\b\W*?\bfrom|bms_\w+\.)|load\b\W*?\bdata\b.*?\binfile)\b|print\b\W*?\@\@)|(?:;\W*?\b(?:shutdown|drop)|collation\W*?\(a|\@\@version)\b|(?:s(?:qloledb|a)|msdasql|dbo)))

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942440

Description

SQL Comment Sequence Detected

Configured Variable

SecRule

REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx /*!?|*/|[;]--|--(?:[\s\v]|[^\-]*?-)|[^&\-]#.*?[\s\v]|;?\x00

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?:/*!?|*/|[;]--|--[\s\r\n\v\f]|--[^-]*?-|[^&-]#.*?[\s\r\n\v\f]|;?\\x00)

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or

argument value, this rule would trigger and block the request.

Rule ID:942510
Description
SQLi bypass attempt by ticks or backticks detected
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?:`(?:(?:[\w\s=_\-+{}()<@]){229}|(?:[A-Za-z0-9+/]{4})+(?:[A-Za-z0-9+/]{2}==|[A-Za-z0-9+/]{3}=)?)`)
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?:`((?:[\w\s=_\-+{}()<@]){229}|(?:[A-Za-z0-9+\/]{4})+(?:[A-Za-z0-9+\/]{2}==|[A-Za-z0-9+\/]{3}=)?)`)
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n-

REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942101
Description
SQL Injection Attack Detected via libinjection
Configured Variable
SecRule REQUEST_BASENAME|REQUEST_FILENAME "@detectSQLi"
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_BASENAME "@detectSQLi"
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web

application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942251
Description
Detects HAVING injections
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)\W+\d*?\s*?\bhaving\b\s*?[^\s\-]
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule

REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)\W+\d*?\s*?having\s*?[^\s\-]

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:942511

Description

SQLi bypass attempt by ticks detected

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?:(?:(?:[\w\s=_\-+{}()<@]){229}|(?:[A-Za-z0-9+/]{4})+(?:[A-Za-z0-9+/]{2}==|[A-Za-z0-9+/]{3}=)?))

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n-

(?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?:((?:[\w\s=_\-+{}()<@]){229}|(?:[A-Za-z0-9+\/]{4})+(?:[A-Za-z0-9+\/]{2}==|[A-Za-z0-9+\/]{3}=)?))

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:951100

Description

None

Configured Variable

SecRule RESPONSE_BODY "!@pmFromFile sql-errors.data"

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n-

REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule RESPONSE_BODY "@pmFromFile sql-errors.data"

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:951230

Description

mysql SQL Information Leakage

Configured Variable

SecRule RESPONSE_BODY @rx (?i)(?supplied argument is not a valid |SQL syntax.*)MySQL|Column count doesnt match(?: value count at row)?|mysql_fetch_array\(\)|on MySQL result index|You have an error in your SQL syntax(?:;| near)|MyS(?:QL server version for the right syntax to use|qlClient\.)|\[MySQL\]\[ODBC|(?:Table [^]+ doesnt exis|valid MySQL

resul)t|Warning.{110}mysql_(?:[\(-\)_a-z]{126})?|ERROR [0-9]{4} \([0-9a-z]{5}\):

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule RESPONSE_BODY @rx (?i)(?supplied argument is not a valid MySQL|Column count doesnt match value count at row|mysql_fetch_array\(\)|on MySQL result index|You have an error in your SQL syntax;|You have an error in your SQL syntax near|MySQL server version for the right syntax to use|\[MySQL\]\[ODBC|Column count doesnt match|Table [^]+ doesnt exist|SQL syntax.*MySQL|Warning.*mysql_.*|valid MySQL result|MySqlClient\.)

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:951240
Description
postgres SQL Information Leakage
Configured Variable
SecRule RESPONSE_BODY @rx (?i)P(?ostgreSQL(?: query failed:|.{120}ERROR)|G::[a-z]*Error)|pg_(?:query|exec)\(\) \[:|Warning.*\bpg_.*|valid PostgreSQL result|Npgsql\.|Supplied argument is not a valid PostgreSQL .*? resource|Unable to connect to PostgreSQL server
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule RESPONSE_BODY @rx (?iPostgreSQL query failed:|pg_query\(\) \[:|pg_exec\(\) \[:|PostgreSQL.*ERROR|Warning.*pg_.*|valid PostgreSQL result|Npgsql\.|PG::[a-zA-Z]*Error|Supplied argument is not a valid PostgreSQL .*? resource|Unable to connect to PostgreSQL server)
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule

should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:951260
Description
Sybase SQL Information Leakage
Configured Variable
SecRule RESPONSE_BODY @rx (?i)(?Sybase message:|Warning.{220}sybase|Sybase.*Server message.*)
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule RESPONSE_BODY @rx (?i)(?Sybase message:|Warning.*sybase.*|Sybase.*Server message.*)
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web

application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:954100
Description
Disclosure of IIS install location
Configured Variable
SecRule RESPONSE_BODY @rx [a-z]x5cinetpub\b
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule RESPONSE_BODY @rx [a-z]inetpub\b
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe

regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:954110
Description
Application Availability Error
Configured Variable
SecRule RESPONSE_BODY @rx (?Microsoft OLE DB Provider for SQL Server(?:.{120}?error 800(?:04005|40e31).{140}?Timeout expired|\(0x80040e31\) Timeout expired)| internal server error .*? part of the server has crashed or it has a configuration error\. |cannot connect to the server: timed out)
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable
SecRule RESPONSE_BODY @rx (?Microsoft OLE DB Provider for SQL Server(?:<\/font>.{120}?error 800(?:04005|40e31).{140}?Timeout expired| \(0x80040e31\) Timeout expired)|<h1>internal server error<\/h1>.*?<h2>part of the server has crashed or it has a configuration error\.<\/h2>|cannot connect to the server: timed out)
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:954120
Description
IIS Information Leakage
Configured Variable
SecRule RESPONSE_BODY "@pmFromFile iis-errors.data"
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n-

(?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule RESPONSE_BODY @rx (?b(?:A(?:DODB\.Command\b.{0100}?\b(?:Application uses a value of the wrong type for the current operation\b|error)| trappable error occurred in an external object\. The script cannot continue running\b)|Microsoft VBScript (?:compilation (?:\(0x8|error)|runtime (?:Error|\(0x8))\b|Object required: |error 800)|<b>Version Information:<\/b>(?: |\s)(?:Microsoft \.NET Framework|ASP\.NET) Version:|>error ASP\b|An Error Has Occurred|>Syntax error in string in query expression|\/[Ee]rror[Mm]essage\.aspx?\?[Ee]rror\b)

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:930100

Description

Path Traversal Attack (/../) or (/.../)

Configured Variable

SecRule REQUEST_URI_RAW|ARGS|REQUEST_HEADERS|!REQUEST_HEADERSReferer|FILES|XML:/*

@rx (?i)(?:[/\x5c]|%(?:2(?:f|5(?:2f|5c|c(?:1%259c|0%25af))|%46)|5c|c(?:0%(?:[2aq]f|5c|9v)|1%(?:[19p]c|8s|af))|(?:bg%q|(?:e|f(?:8%8)?0%8)0%80%a)f|u(?:221[5-6]|EFC8|F025|002f)|%3(?:2(?:%(?:%6|4)6|F)|5%%63)|1u)|0x(?:2f|5c))(?:\.(?:%0[0-1]|\?)?|\?\.?|%(?:2(?:(?:5(?:2|c0%25a))?e|%45)|c0(?:\.|%[25-6ae-f]e)|u(?:(?:ff0|002)e|2024)|%32(?:%(?:%6|4)5|E)|(?:e|f(?:(?:8|c%80)%8)?0%8)0%80%ae)|0x2e){23}(?:[/\x5c]|%(?:2(?:f|5(?:2f|5c|c(?:1%259c|0%25af))|%46)|5c|c(?:0%(?:[2aq]f|5c|9v)|1%(?:[19p]c|8s|af))|(?:bg%q|(?:e|f(?:8%8)?0%8)0%80%a)f|u(?:221[5-6]|EFC8|F025|002f)|%3(?:2(?:%(?:%6|4)6|F)|5%%63)|1u)|0x(?:2f|5c))

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_URI_RAW|ARGS|REQUEST_HEADERS|!REQUEST_HEADERSReferer|XML:/* @rx (?i)(?:\x5c|(?:%(?:c(?:0%(?:[2aq]f|5c|9v)|1%(?:[19p]c|8s|af))|2(?:5(?:c(?:0%25af|1%259c)|2f|5c)|%46|f)|(?:(?:f(?:8%8)?0%8|e)0%80%a|bg%q)f|%3(?:2(?:%(?:%6|4)6|F)|5%%63)|u(?:221[56]|002f|EFC8|F025)|1u|5c)|0x(?:2f|5c)|\/))(?:%(?:(?:(?:f(?:(?:c%80|8)%8)?0%8|e)0%80%ae|2(?:(?:5(?:c0%25a|2))?e|%45)|u(?:(?:002|ff0)e|2024)|%32(?:%(?:%6|4)5|E)|c0(?:%[256aef]e|\.))|\.(?:%0[01]|\?)?|\?\.?|0x2e){2}(?:\x5c|(?:%(?:c(?:0%(?:[2aq]f|5c|9v)|1%(?:[19p]c|8s|af))|2(?:5(?:c(?:0%25af|1%259c)|2f|5c)|%46|f)|(?:(?:f(?:8%8)?0%8|e)0%80%a|bg%q)f|%3(?:2(?:%(?:%6|4)6|F)|5%%63)|u(?:221[56]|002f|EFC8|F025)|1u|5c)|0x(?:2f|5c)|\/))

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:930110
Description
Path Traversal Attack (/../) or (/.../)
Configured Variable
SecRule REQUEST_URI|ARGS|REQUEST_HEADERS|!REQUEST_HEADERSReferer|FILES|XML:/* @rx (?:(?:^|[\x5c/;])\.{23}[\x5c/;]|[\x5c/;]\.{23}(?:[\x5c/;]|$))
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable
SecRule REQUEST_URI|ARGS|REQUEST_HEADERS|!REQUEST_HEADERSReferer|XML:/* @rx (?:^|[\\/])\.\.(?:[\\/]|$)
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941130
Description
XSS Filter - Category 3: Attribute Vector
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|REQUEST_HEADERS:User-Agent|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx (?i).(?:\b(?:x(?:link:href|html|mlns)|data:text/html|formaction|pattern\b.*?=)|!ENTITY[\s\v]+(?:%[\s\v]+)?[^\s\v]+[\s\v]+(?:SYSTEM|PUBLIC)|@import|;base64)\b
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule

should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|REQUEST_HEADERS:User-Agent|ARGS_NAMES|ARGS|XML:/* @rx (?i)[\s\S](?:!ENTITY\s+(?:\S+|%\s+\S+)\s+(?:PUBLIC|SYSTEM)|x(?:link:href|html|mlns)|data:text\/html|pattern\b.*?=|formaction|\@import|;base64)\b

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941140

Description

XSS Filter - Category 4: Javascript URI Vector

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|REQUEST_H

EADERS:User-Agent|REQUEST_HEADERS:Referer|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx (?i)[a-z]+=(?:[^:=]+:.+;)*?[^:=]+:url\(javascript

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|REQUEST_HEADERS:User-Agent|REQUEST_HEADERS:Referer|ARGS_NAMES|ARGS|XML:/* @rx (?i)(?:<(?:(?:apple|objec)t|isindex|embed|style|form|meta)\b[^>]*?>[\s\S]*?|(?:=|U\s*?R\s*?L\s*?\()\s*?[^>]*?\s*?S\s*?C\s*?R\s*?I\s*?P\s*?T\s*?:)

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or

argument value, this rule would trigger and block the request.

Rule ID:941160
Description
NoScript XSS InjectionChecker: HTML Injection
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|REQUEST_HEADERS:User-Agent|REQUEST_HEADERS:Referer|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx (?i)<[^0-9<>A-Z_a-z]*(?:[^\s\v\<>]*:)?[^0-9<>A-Z_a-z]*[^0-9A-Z_a-z]*?(?:s[^0-9A-Z_a-z]*?(?:c[^0-9A-Z_a-z]*?r[^0-9A-Z_a-z]*?i[^0-9A-Z_a-z]*?p[^0-9A-Z_a-z]*?t|t[^0-9A-Z_a-z]*?y[^0-9A-Z_a-z]*?l[^0-9A-Z_a-z]*?e|v[^0-9A-Z_a-z]*?g|e[^0-9A-Z_a-z]*?t[^0-9>A-Z_a-z])|f[^0-9A-Z_a-z]*?o[^0-9A-Z_a-z]*?r[^0-9A-Z_a-z]*?m|m[^0-9A-Z_a-z]*?(?:a[^0-9A-Z_a-z]*?r[^0-9A-Z_a-z]*?q[^0-9A-Z_a-z]*?u[^0-9A-Z_a-z]*?e[^0-9A-Z_a-z]*?e|e[^0-9A-Z_a-z]*?t[^0-9A-Z_a-z]*?a[^0-9>A-Z_a-z])|(?:l[^0-9A-Z_a-z]*?i[^0-9A-Z_a-z]*?n[^0-9A-Z_a-z]*?k|o[^0-9A-Z_a-z]*?b[^0-9A-Z_a-z]*?j[^0-9A-Z_a-z]*?e[^0-9A-Z_a-z]*?c[^0-9A-Z_a-z]*?t|e[^0-9A-Z_a-z]*?m[^0-9A-Z_a-z]*?b[^0-9A-Z_a-z]*?e[^0-9A-Z_a-z]*?d|a[^0-9A-Z_a-z]*?(?:p[^0-9A-Z_a-z]*?p[^0-9A-Z_a-z]*?l[^0-9A-Z_a-z]*?e[^0-9A-Z_a-z]*?t|u[^0-9A-Z_a-z]*?d[^0-9A-Z_a-z]*?i[^0-9A-Z_a-z]*?o|n[^0-9A-Z_a-z]*?i[^0-9A-Z_a-z]*?m[^0-9A-Z_a-z]*?a[^0-9A-Z_a-z]*?t[^0-9A-Z_a-z]*?e)|p[^0-9A-Z_a-z]*?a[^0-9A-Z_a-z]*?r[^0-9A-Z_a-z]*?a[^0-9A-Z_a-z]*?m|i?[^0-9A-Z_a-z]*?f[^0-9A-Z_a-z]*?r[^0-9A-Z_a-z]*?a[^0-9A-Z_a-z]*?m[^0-9A-Z_a-z]*?e|b[^0-9A-Z_a-z]*?(?:a[^0-9A-Z_a-z]*?s[^0-9A-Z_a-z]*?e|o[^0-9A-Z_a-z]*?d[^0-9A-Z_a-z]*?y|i[^0-9A-Z_a-z]*?n[^0-9A-Z_a-z]*?d[^0-9A-Z_a-z]*?i[^0-9A-Z_a-z]*?n[^0-9A-Z_a-z]*?g[^0-9A-Z_a-z]*?s)|i[^0-9A-Z_a-z]*?m[^0-9A-Z_a-z]*?a?[^0-9A-Z_a-z]*?g[^0-9A-Z_a-z]*?e?|v[^0-9A-Z_a-z]*?i[^0-9A-Z_a-z]*?d[^0-9A-Z_a-z]*?e[^0-9A-Z_a-z]*?o)[^0-9>A-Z_a-z])|(?:<[0-9A-Z_a-z].*[\s\v/]|[\](?:.*[\s\v/])?)(?:background|formaction|lowsrc|on(?:a(?:bort|ctivate|d(?:apteradded|dtrack)|fter(?:print|(?:scriptexecu|upda)te)|lerting|n(?:imation(?:end|iteration|start)|tennastatechange)|ppcommand|udio(?:end|process|start))|b(?:e(?:fore(?:(?:(?:de)?activa|scriptexecu)te|c(?:opy|ut)|editfocus|p(?:aste|rint)|u(?:nload|pdate))|gin(?:Event)?)|l(?:ocked|ur)|oun(?:ce|dary)|roadcast|usy)|c(?:a(?:(?:ch|llschang)ed|nplay(?:through)?|rdstatechange)|(?:ell|fstate)change|h(?:a(?:rging(?:time)?cha)?nge|ecking)|l(?:ick|ose)|o(?:m(?:mand(?:update)?|p(?:lete|osition(?:end|start|update)))|n(?:nect(?:ed|ing)|t(?:extmenu|rolselect))|py)|u(?:echange|t))|d(?:ata(?:(?:availabl|chang)e|err	

or|setc(?:hanged|omplete))|blclick|e(?:activate|livery(?:error|success)|vice(?:found|light|(?:mo|orienta)tion|proximity))|i(?:aling|s(?:abled|c(?:hargingtimechange|onnect(?:ed|ing))))|o(?:m(?:a(?:ctivate|ttrmodified)|(?:characterdata|subtree)modified|focus(?:in|out)|mousescroll|node(?:inserted(?:intodocument)?|removed(?:fromdocument)?))|wnloading)|r(?:ag(?:drop|e(?:n(?:d|ter)|xit)|(?:gestur|leav)e|over|start)|op)|urationchange)|e(?:mptied|n(?:abled|d(?:ed|Event)?|ter)|rror(?:update)?|xit)|f(?:ailed|i(?:lterchange|nish)|o(?:cus(?:in|out)?|rm(?:change|input)))|g(?:amepad(?:axismove|button(?:down|up)|(?:dis)?connected)|et)|h(?:ashchange|e(?:adphoneschange|l[dp])|olding)|i(?:cc(?:cardlockerror|infochange)|n(?:coming|put|valid))|key(?:down|press|up)|l(?:evelchange|o(?:ad(?:e(?:d(?:meta)?data|nd)|start)?|secapture)|y)|m(?:ark|essage|o(?:use(?:down|enter|(?:lea|mo)ve|o(?:ut|ver)|up|wheel)|ve(?:end|start)?|z(?:a(?:fterpaint|udioavailable)|(?:beforeresiz|orientationchang|t(?:apgestur|imechang))e|(?:edgeui(?:c(?:ancel|omplet)|start)e|network(?:down|up)loa)d|fullscreen(?:change|error)|m(?:agnifygesture(?:start|update)?|ouse(?:hittest|pixelscroll))|p(?:ointerlock(?:change|error)|resstapgesture)|rotategesture(?:start|update)?|s(?:crolledareachanged|wipegesture(?:end|start|update)?)))|no(?:match|update)|o(?:(?:bsolet|(?:ff|n)lin)e|pen|verflow(?:changed)?)|p(?:a(?:ge(?:hide|show)|int|(?:st|us)e)|lay(?:ing)?|op(?:state|up(?:hid(?:den|ing)|show(?:ing|n))))|ro(?:gress|pertychange))|r(?:atechange|e(?:adystatechange|ceived|movetrack|peat(?:Event)?|quest|s(?:et|ize|u(?:lt|m(?:e|ing)))|trieving)|ow(?:e(?:nter|xit)|s(?:delete|inserted)))|s(?:croll|e(?:ek(?:complete|ed|ing)|lect(?:start)?|n(?:ding|t)|t)|how|(?:ound|peech)(?:end|start)|t(?:a(?:lled|rt|t(?:echange|uschanged))|k(?:comma|sessione)nd|op)|u(?:bmit|ccess|spend)|vg(?:abort|error|(?:un)?load|resize|scroll|zoom))|t(?:ext|ime(?:out|update)|ouch(?:cancel|en(?:d|ter)|(?:lea|mo)ve|start)|ransition(?:cancel|end|run))|u(?:n(?:derflow|load)|p(?:dateready|gradeneeded)|s(?:erproximity|sdreceived))|v(?:ersion|o(?:ic|lum)e)change|w(?:a(?:it|rn)ing|heel)|zoom)|ping|s(?:rc|tyle))[\x08-\n\f-\r]*?=

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression

itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|REQUEST_HEADERS:User-Agent|REQUEST_HEADERS:Referer|ARGS_NAMES|ARGS|XML:/* @rx (?i:(?:<\w[\s\S]*[\s\/]|[\](?:[\s\S]*[\s\/])?)(?:on(?:d(?:e(?:vice(?:(?:orienta|mo)tion|proximity|found|light)|livery(?:success|error)|activate)|r(?:ag(?:e(?:n(?:ter|d)|xit)|(?:gestur|leav)e|start|drop|over)|op)|i(?:s(?:c(?:hargingtimechange|onnect(?:ing|ed))|abled)|aling)|ata(?:setc(?:omplete|hanged)|(?:availabl|chang)e|error)|urationchange|ownloading|blclick)|Moz(?:M(?:agnifyGesture(?:Update|Start)?|ouse(?:PixelScroll|Hittest))|S(?:wipeGesture(?:Update|Start|End)?|crolledAreaChanged)|(?:(?:Press)?TapGestur|BeforeResiz)e|EdgeUI(?:C(?:omplet|ancel)|Start)ed|RotateGesture(?:Update|Start)?|A(?:udioAvailable|fterPaint))|c(?:o(?:m(?:p(?:osition(?:update|start|end)|lete)|mand(?:update)?)|n(?:t(?:rolselect|extmenu)|nect(?:ing|ed))|py)|a(?:(?:llschang|ch)ed|nplay(?:through)?|rdstatechange)|h(?:(?:arging(?:time)?ch)?ange|ecking)|(?:fstate|ell)change|u(?:echange|t)|l(?:ick|ose))|s(?:t(?:a(?:t(?:uschanged|echange)|lled|rt)|k(?:sessione|comma)nd|op)|e(?:ek(?:complete|ing|ed)|(?:lec(?:tstar)?)?t|n(?:ding|t))|(?:peech|ound)(?:start|end)|u(?:ccess|spend|bmit)|croll|how)|m(?:o(?:z(?:(?:pointerlock|fullscreen)(?:change|error)|(?:orientation|time)change|network(?:down|up)load)|use(?:(?:lea|mo)ve|o(?:ver|ut)|enter|wheel|down|up)|ve(?:start|end)?)|essage|ark)|a(?:n(?:imation(?:iteration|start|end)|tennastatechange)|fter(?:(?:scriptexecu|upda)te|print)|udio(?:process|start|end)|d(?:apteradded|dtrack)|ctivate|lerting|bort)|b(?:e(?:fore(?:(?:(?:de)?activa|scriptexecu)te|u(?:nload|pdate)|p(?:aste|rint)|c(?:opy|ut)|editfocus)|gin(?:Event)?)|oun(?:dary|ce)|l(?:ocked|ur)|roadcast|usy)|DOM(?:Node(?:Inserted(?:IntoDocument)?|Removed(?:FromDocument)?)|(?:CharacterData|Subtree)Modified|A(?:ttrModified|ctivate)|Focus(?:Out|In)|MouseScroll)|r(?:e(?:s(?:u(?:m(?:ing|e)|lt)|ize|et)|adystatechange|pea(?:tEven)?t|movetrack|trieving|ceived)|ow(?:s(?:inserted|delete)|e(?:nter|xit))|atechange)|p(?:op(?:up(?:hid(?:den|ing)|show(?:ing|n))|state)|a(?:ge(?:hide|show)|(?:st|us)e|int)|ro(?:pertychange|gress)|lay(?:ing)?)|t(?:ouch(?:(?:lea|mo)ve|en(?:ter|d)|cancel|start)|ransition(?:cancel|end|run)|ime(?:update|out)|ext)|u(?:s(?:erproximity|sdreceived)|p(?:gradeneeded|dateready)|n(?:derflow|load))|f(?:o(?:rm(?:change|input)|cus(?:out|in)?)|i(?:lterchange|nish)|ailed)|l(?:o(?:ad(?:e(?:d(?:meta)?data|nd)|start)|secapture)|evelchange|y)|g(?:amepad(?:(?:dis)?connected|button(?:down|up)|axismove)|et)|e(?:n(?:d(?:Event|ed)?|abled|ter)|rror(?:update)

?|mptied|xit)|i(?:cc(?:cardlockerror|infochange)|n(?:coming|valid|put))|o(?:(?:(?:ff|n)lin|bsolet)e|verflow(?:changed)?|pen)|SVG(?:(?:Unl|L)oad|Resize|Scroll|Abort|Error|Zoom)|h(?:e(?:adphoneschange|l[dp])|ashchange|olding)|v(?:o(?:lum|ic)e|ersion)change|w(?:a(?:it|rn)ing|heel)|key(?:press|down|up)|(?:AppComman|Loa)d|no(?:update|match)|Request|zoom)|s(?:tyle|rc)|background|formaction|lowsrc|ping)[\s\x08]*?=|<[^\w<>]*(?:[^<>\\s]*:)?[^\w<>]*\W*?(?:(?:a\W*?(?:n\W*?i\W*?m\W*?a\W*?t\W*?e|p\W*?p\W*?l\W*?e\W*?t|u\W*?d\W*?i\W*?o)|b\W*?(?:i\W*?n\W*?d\W*?i\W*?n\W*?g\W*?s|a\W*?s\W*?e|o\W*?d\W*?y)|i?\W*?f\W*?r\W*?a\W*?m\W*?e|o\W*?b\W*?j\W*?e\W*?c\W*?t|i\W*?m\W*?a?\W*?g\W*?e?|e\W*?m\W*?b\W*?e\W*?d|p\W*?a\W*?r\W*?a\W*?m|v\W*?i\W*?d\W*?e\W*?o|l\W*?i\W*?n\W*?k)[^>\w]|s\W*?(?:c\W*?r\W*?i\W*?p\W*?t|t\W*?y\W*?l\W*?e|e\W*?t[^>\w]|v\W*?g)|m\W*?(?:a\W*?r\W*?q\W*?u\W*?e\W*?e|e\W*?t\W*?a[^>\w])|f\W*?o\W*?r\W*?m))

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941170

Description

NoScript XSS InjectionChecker: Attribute Injection

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|REQUEST_HEADERS:User-Agent|REQUEST_HEADERS:Referer|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx

(?i)(?:\W|^)(?:javascript:(?:[\s\S]+[=\x5c\(\[\.<]|[\s\S]*?(?:\bname\b|\x5c[ux]\d))|data:(?:(?:[a-z]\w+/\w[\w+-]+\w)?[;]|[\s\S]*?;[\s\S]*?\b(?:base64|charset=)|[\s\S]*?[\s\S]*?<[\s\S]*?\w[\s\S]*?>))|@\W*?i\W*?m\W*?p\W*?o\W*?r\W*?t\W*?(?:/*[\s\S]*?)?(?:[\]|\W*?u\W*?r\W*?l[\s\S]*?\()|[^-]*?-\W*?m\W*?o\W*?z\W*?-\W*?b\W*?i\W*?n\W*?d\W*?i\W*?n\W*?g[^:]*?:\W*?u\W*?r\W*?l[\s\S]*?\(

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|REQUEST_HEADERS:User-Agent|REQUEST_HEADERS:Referer|ARGS_NAMES|ARGS|XML:/* @rx (?i)(?:\W|^)(?:javascript:(?:[\s\S]+[=\\\(\[\.<]|[\s\S]*?(?:\bname\b|\\[ux]\d))|data:(?:(?:[a-z]\w+\/\w[\w+-]+\w)?[;]|[\s\S]*?;[\s\S]*?\b(?:base64|charset=)|[\s\S]*?[\s\S]*?<[\s\S]*?\w[\s\S]*?>))|@\W*?i\W*?m\W*?p\W*?o\W*?r\W*?t\W*?(?:\/*[\s\S]*?)?(?:[\]|\W*?u\W*?r\W*?l[\s\S]*?\()|\W*?-\W*?m\W*?o\W*?z\W*?-\W*?b\W*?i\W*?n\W*?d\W*?i\W*?n\W*?g[\s\S]*?:[\s\S]*?\W*?u\W*?r\W*?l[\s\S]*?\(

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a

ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941180
Description
Node-Validator Deny List Keywords
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @pm document.cookie document.domain document.write .parentnode .innerhtml window.location -moz-binding <!-- <![cdata[
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @pm document.cookie document.write .parentnode .innerhtml window.location -moz-binding <!-- --> <![cdata[

Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941190
Description
IE XSS Filters - Attack Detected
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx (?i:<style.*?>.*?(?:@[i\x5c]|(?:[:=]|&#x?0*(?:58|3A|61|3D);?).*?(?:[(\x5c]|&#x?0*(?:40|28|92|5C);?)))
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a

leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:<style.*?>.*?(?:@[i\\\\]|(?:[:=]|&#x?0*(?:58|3A|61|3D);?).*?(?:[(\\\\]|&#x?0*(?:40|28|92|5C);?)))

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941200

Description

IE XSS Filters - Attack Detected

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx (?i:<.*[:]?vmlframe.*?[\s/+]*?src[\s/+]*=)

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n-

REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:<.*[:]?vmlframe.*?[\s/+]*?src[\s/+]*=)

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941210

Description

IE XSS Filters - Attack Detected

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAM

ES|ARGS|REQUEST_FILENAME|XML:/* @rx (?i:(?:j|&#x?0*(?:74|4A|106|6A);?)(?:\t|\n|\r|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:a|&#x?0*(?:65|41|97|61);?)(?:\t|\n|\r|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:v|&#x?0*(?:86|56|118|76);?)(?:\t|\n|\r|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:a|&#x?0*(?:65|41|97|61);?)(?:\t|\n|\r|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:s|&#x?0*(?:83|53|115|73);?)(?:\t|\n|\r|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:c|&#x?0*(?:67|43|99|63);?)(?:\t|\n|\r|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:r|&#x?0*(?:82|52|114|72);?)(?:\t|\n|\r|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:i|&#x?0*(?:73|49|105|69);?)(?:\t|\n|\r|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:p|&#x?0*(?:80|50|112|70);?)(?:\t|\n|\r|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:t|&#x?0*(?:84|54|116|74);?)(?:\t|\n|\r|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?::|&(?:#x?0*(?:58|3A);?|colon;)).)

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:(?:j|&#x?0*(?:74|4A|106|6A);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:a|&#x?0*(?:65|41|97|61);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:v|&#x?0*(?:86|56|118|76);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:a|&#x?0*(?:65|41|97|61);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:s|&#x?0*(?:83|53|115|73);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:c|&#x?0*(?:67|43|99|63);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:r|&#x?0*(?:82|52|114|72);?)(?:\t|

&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:i|&#x?0*(?:73|49|105|69);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:p|&#x?0*(?:80|50|112|70);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:t|&#x?0*(?:84|54|116|74);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?::|&(?:#x?0*(?:58|3A);?|colon;)).)

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941220

Description

IE XSS Filters - Attack Detected

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx (?i:(?:v|&#x?0*(?:86|56|118|76);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:b|&#x?0*(?:66|42|98|62);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:s|&#x?0*(?:83|53|115|73);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:c|&#x?0*(?:67|43|99|63);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:r|&#x?0*(?:82|52|114|72);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:i|&#x?0*(?:73|49|105|69);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:p|&#x?0*(?:80|50|112|70);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:t|&#x?0*(?:84|54|116|74);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?::|&(?:#x?0*(?:58|3A);?|colon;)).)

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:(?:v|&#x?0*(?:86|56|118|76);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:b|&#x?0*(?:66|42|98|62);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:s|&#x?0*(?:83|53|115|73);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:c|&#x?0*(?:67|43|99|63);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:r|&#x?0*(?:82|52|114|72);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:i|&#x?0*(?:73|49|105|69);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:p|&#x?0*(?:80|50|112|70);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?:t|&#x?0*(?:84|54|116|74);?)(?:\t|&(?:#x?0*(?:9|13|10|A|D);?|tab;|newline;))*(?::|&(?:#x?0*(?:58|3A);?|colon;)).)

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n-

(?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941230
Description
IE XSS Filters - Attack Detected
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx (?i)<EMBED[\s/+].*?(?:src|type).*?=
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)<EMBED[\s/+].*?(?:src|type).*?=
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web

application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941240
Description
IE XSS Filters - Attack Detected
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx <[?]?import[\s/+\S]*?implementation[\s/+]*?=
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule

REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx <[?]?import[\s\/+\S]*?implementation[\s\/+]*?=

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941250

Description

IE XSS Filters - Attack Detected

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx (?i:<META[\s/+].*?http-equiv[\s/+]*=[\s/+]*[\`]?(?:(?:c|&#x?0*(?:67|43|99|63);?)|(?:r|&#x?0*(?:82|52|114|72);?)|(?:s|&#x?0*(?:83|53|115|73);?)))

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a

ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:<META[\s/+].*?http-equiv[\s/+]*=[\s/+]*[\`]?(?:(?:c|&#x?0*(?:67|43|99|63);?)|(?:r|&#x?0*(?:82|52|114|72);?)|(?:s|&#x?0*(?:83|53|115|73);?)))

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941260

Description

IE XSS Filters - Attack Detected

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx (?i:<META[\s/+].*?charset[\s/+]*=)

Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:<META[\s/+].*?charset[\s/+]*=)
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941270
Description

IE XSS Filters - Attack Detected
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx (?i)<LINK[\s/+].*?href[\s/+]*=
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)<LINK[\s/+].*?href[\s/+]*=
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a

leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941280
Description
IE XSS Filters - Attack Detected
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx (?i)<BASE[\s/+].*?href[\s/+]*=
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)<BASE[\s/+].*?href[\s/+]*=
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAM

ES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941290
Description
IE XSS Filters - Attack Detected
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx (?i)<APPLET[\s/+>]
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)<APPLET[\s/+>]

Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941300
Description
IE XSS Filters - Attack Detected
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx (?i)<OBJECT[\s/+].*?(?:type|codetype|classid|code|data)[\s/+]*=
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a

leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)<OBJECT[\s/+].*?(?:type|codetype|classid|code|data)[\s/+]*=
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941310
Description
US-ASCII Malformed Encoding XSS Filter - Attack Detected
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx (?:\xbc\s*/\s*[^\xbe>]*[\xbe>])|(?:<\s*/\s*[^\xbe]*\xbe)
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAM

ES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx \xbc[^\xbe>]*[\xbe>]|<[^\xbe]*\xbe

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941350

Description

UTF-7 Encoding IE XSS - Attack Detected

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx \+ADw-.*(?:\+AD4-|>)|<.*\+AD4-

Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx \+ADw-.*(?:\+AD4-|>)|<.*\+AD4-
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941360
Description

JSFuck / Hieroglyphy obfuscation detected
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx ![!+]\[\]
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx ![!+]\[\]
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a

leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941370
Description
JavaScript global variable found
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx (?:self|document|this|top|window)\s*(?:/*|[\[]).+?(?:\]|*/)
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS|XML:/* @rx (?:self|document|this|top|window)\s*(?:/*|[\[]).+?(?:\]|*/)
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAM

ES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941101
Description
XSS Attack Detected via libinjection
Configured Variable
SecRule REQUEST_FILENAME|REQUEST_HEADERSReferer @detectXSS
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_HEADERSReferer @detectXSS
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n-

REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941120
Description
XSS Filter - Category 2: Event Handler Vector
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|REQUEST_HEADERS:User-Agent|REQUEST_HEADERS:Referer|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx (?i)[\s\`;/0-9=\x0B\x09\x0C\x3B\x2C\x28\x3B]on[a-zA-Z]{325}[\s\x0B\x09\x0C\x3B\x2C\x28\x3B]*?=[^=]
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|REQUEST_HEADERS:User-Agent|REQUEST_HEADERS:Referer|ARGS_NAMES|ARGS|XML:/* @rx (?i)[\s\`;\/0-9=\x0B\x09\x0C\x3B\x2C\x28\x3B]on[a-zA-Z]+[\s\x0B\x09\x0C\x3B\x2C\x28\x3B]*?=
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941150
Description
XSS Filter - Category 5: Disallowed HTML Attributes
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|REQUEST_HEADERS:User-Agent|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx (?i)\b(?:s(?:tyle|rc)|href)\b[\s\S]*?=
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAM

ES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|REQUEST_HEADERS:User-Agent|ARGS_NAMES|ARGS|XML:/* @rx (?i)\b(?:s(?:tyle|rc)|href)\b[\s\S]*?=
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941320
Description
Possible XSS Attack Detected - HTML Tag Handler
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx

<(?:a|abbr|acronym|address|applet|area|audioscope|b|base|basefront|bdo|bgsound|big|blackface|blink|blockquote|body|bq|br|button|caption|center|cite|code|col|colgroup|comment|dd|del|dfn|dir|div|dl|dt|em|embed|fieldset|fn|font|form|frame|frameset|h1|head|hr|html|i|iframe|ilayer|img|input|ins|isindex|kdb|keygen|label|layer|legend|li|limittext|link|listing|map|marquee|menu|meta|multicol|nobr|noembed|noframes|noscript|nosmartquotes|object|ol|optgroup|option|p|param|plaintext|pre|q|rt|ruby|s|samp|script|select|server|shadow|sidebar|small|spacer|span|strike|strong|style|sub|sup|table|tbody|td|textarea|tfoot|th|thead|title|tr|tt|u|ul|var|wbr|xml|xmp)\W

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx <(?:a|abbr|acronym|address|applet|area|audioscope|b|base|basefront|bdo|bgsound|big|blackface|blink|blockquote|body|bq|br|button|caption|center|cite|code|col|colgroup|comment|dd|del|dfn|dir|div|dl|dt|em|embed|fieldset|fn|font|form|frame|frameset|h1|head|hr|html|i|iframe|ilayer|img|input|ins|isindex|kdb|keygen|label|layer|legend|li|limittext|link|listing|map|marquee|menu|meta|multicol|nobr|noembed|noframes|noscript|nosmartquotes|object|ol|optgroup|option|p|param|plaintext|pre|q|rt|ruby|s|samp|script|select|server|shadow|sidebar|small|spacer|span|strike|strong|style|sub|sup|table|tbody|td|textarea|tfoot|th|thead|title|tr|tt|u|ul|var|wbr|xml|xmp)\W

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941330
Description
IE XSS Filters - Attack Detected
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx (?i:[\][ ]*(?:[^a-z0-9~_:]|in).*?(?:(?:l|\x5cu006C)(?:o|\x5cu006F)(?:c|\x5cu0063)(?:a|\x5cu0061)(?:t|\x5cu0074)(?:i|\x5cu0069)(?:o|\x5cu006F)(?:n|\x5cu006E)|(?:n|\x5cu006E)(?:a|\x5cu0061)(?:m|\x5cu006D)(?:e|\x5cu0065)|(?:o|\x5cu006F)(?:n|\x5cu006E)(?:e|\x5cu0065)(?:r|\x5cu0072)(?:r|\x5cu0072)(?:o|\x5cu006F)(?:r|\x5cu0072)|(?:v|\x5cu0076)(?:a|\x5cu0061)(?:l|\x5cu006C)(?:u|\x5cu0075)(?:e|\x5cu0065)(?:O|\x5cu004F)(?:f|\x5cu0066)).*?=)
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule

should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i:[\][ ]*(?:[^a-z0-9~_:]|in).*?(?:(?:l|\\\\u006C)(?:o|\\\\u006F)(?:c|\\\\u0063)(?:a|\\\\u0061)(?:t|\\\\u0074)(?:i|\\\\u0069)(?:o|\\\\u006F)(?:n|\\\\u006E)|(?:n|\\\\u006E)(?:a|\\\\u0061)(?:m|\\\\u006D)(?:e|\\\\u0065)|(?:o|\\\\u006F)(?:n|\\\\u006E)(?:e|\\\\u0065)(?:r|\\\\u0072)(?:r|\\\\u0072)(?:o|\\\\u006F)(?:r|\\\\u0072)|(?:v|\\\\u0076)(?:a|\\\\u0061)(?:l|\\\\u006C)(?:u|\\\\u0075)(?:e|\\\\u0065)(?:O|\\\\u004F)(?:f|\\\\u0066)).*?=)

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:941340

Description

IE XSS Filters - Attack Detected

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx (?i)[\\][ ]*(?:[^a-z0-9~_:\ ]|in).+?[.].+?=

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|!REQUEST_COOKIES:/_pk_ref/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)[\\][ ]*(?:[^a-z0-9~_:\ ]|in).+?[.].+?=

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or

argument value, this rule would trigger and block the request.

Rule ID:941380
Description
AngularJS client side template injection detected
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|REQUEST_FILENAME|XML:/* @rx {{.*?}}
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx {{.*?}}
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule

should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:920100
Description
Invalid HTTP Request Line
Configured Variable
SecRule REQUEST_LINE !@rx (?i)^(?get /[^#\?]*(?:\?[^\s\v#]*)?(?:#[^\s\v]*)?|(?:connect (?:(?:[0-9]{13}\.){3}[0-9]{13}\.?(?::[0-9]+)?|[\--9A-Z_a-z]+:[0-9]+)|options *|[a-z]{310}[\s\v]+(?:[0-9A-Z_a-z]{37}?://[\--9A-Z_a-z]*(?::[0-9]+)?)?/[^#\?]*(?:\?[^\s\v#]*)?(?:#[^\s\v]*)?)[\s\v]+[\.-9A-Z_a-z]+)$
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_LINE !@rx ^(?i(?:[a-z]{310}\s+(?:\w{37}?://[\w\-\./]*(?::\d+)?)?/[^?#]*(?:\?[^#\s]*)?(?:#[\S]*)?|connect (?:\d{13}\.){3}\d{13}\.?(?::\d+)?|options *)\s+[\w\./]+|get /[^?#]*(?:\?[^#\s]*)?(?:#[\S]*)?)$

Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:920120
Description
Attempted multipart/form-data bypass
Configured Variable
SecRule FILES|FILES_NAMES !@rx (?i)^(?&(?:(?:[acegiln-or-suz]acut|[aeiou]grav|[ain-o]tild)e|[c-elnr-tz]caron|(?:[cgk-lnr-t]cedi|[aeiouy]um)l|[aceg-josuwy]circ|[au]ring|a(?:mp|pos)|nbsp|oslash);|[^\;=])*$
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or

argument value, this rule would trigger and block the request.
Recommended Variable
SecRule FILES_NAMES|FILES @rx (?<!&(?[aAoOuUyY]uml)|&(?:[aAeEiIoOuU]circ)|&(?:[eEiIoOuUyY]acute)|&(?:[aAeEiIoOuU]grave)|&(?:[cC]cedil)|&(?:[aAnNoO]tilde)|&(?:amp)|&(?:apos));|[\=]
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:920350
Description
Host header is a numeric IP address
Configured Variable
SecRule REQUEST_HEADERSHost @rx (?:^([\d.]+|\[[\da-f:]+\]|[\da-f:]+)(:[\d]+)?$)
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n-

(?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_HEADERSHost @rx ^[\d.:]+$

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:920470

Description

Illegal Content-Type header

Configured Variable

SecRule REQUEST_HEADERSContent-Type !@rx ^[\w/.+*-]+(?:\s?;\s?(?:action|boundary|charset|component|start(?:-info)?|type|version)\s?=\s?[\\w.()+/:=?<>@#*-]+)*$

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAM

ES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_HEADERSContent-Type !@rx ^[\w/.+-]+(?:\s?;\s?(?:action|boundary|charset|type|start(?:-info)?)\s?=\s?[\\w.()+/:=?<>@-]+)*$

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:920274

Description

Invalid character in request headers (outside of very strict set)

Configured Variable

SecRule REQUEST_HEADERS|!REQUEST_HEADERSUser-Agent|!REQUEST_HEADERS:Referer|!REQUEST_HEADERS:Cookie|!REQUEST_HEADERS:Sec-Fetch-User|!REQUEST_HEADERS:Sec-CH-UA|!REQUEST_HEADERS:Sec-CH-UA-Mobile @validateByteRange 32343842-596165-909597-122

Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_HEADERS|!REQUEST_HEADERSUser-Agent|!REQUEST_HEADERS:Referer|!REQUEST_HEADERS:Cookie|!REQUEST_HEADERS:Sec-Fetch-User @validateByteRange 32343842-596165-909597-122
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:920275

Description
Invalid character in request headers (outside of very strict set)
Configured Variable
SecRule REQUEST_HEADERSSec-Fetch-User|REQUEST_HEADERS:Sec-CH-UA-Mobile !@rx ^(?:\?[01])?$
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_HEADERSSec-Fetch-User @validateByteRange 32343842-59616365-909597-122
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or

argument value, this rule would trigger and block the request.

Rule ID:920460
Description
Abnormal character escapes in request
Configured Variable
SecRule REQUEST_URI|REQUEST_HEADERS|ARGS|ARGS_NAMES @rx (?^|[^\x5c])\x5c[cdeghijklmpqwxyz123456789]
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_URI|REQUEST_HEADERS|ARGS|ARGS_NAMES @rx (?^|[^\\\\])\\\\[cdeghijklmpqwxyz123456789]
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n-

(?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:921110
Description
HTTP Request Smuggling Attack
Configured Variable
SecRule ARGS_NAMES|ARGS|REQUEST_BODY|XML/* @rx (?:get|post|head|options|connect|put|delete|trace|track|patch|propfind|propatch|mkcol|copy|move|lock|unlock)\s+[^\s]+\s+http/\d
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule ARGS_NAMES|ARGS|REQUEST_BODY|XML/* @rx (?:get|post|head|options|connect|put|delete|trace|track|patch|propfind|propatch|mkcol|copy|move|lock|unlock)\s+(?:\/|\w)[^\s]*(?:\s+http\/\d|[\r\n])
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web

application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:901321
Description
None
Configured Variable
SecRule REQUEST_HEADERSUser-Agent @rx ^.*$
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecAction
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe

regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:901410
Description
None
Configured Variable
SecRule UNIQUE_ID "@rx ^[a-f]*([0-9])[a-f]*([0-9])"
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule UNIQUE_ID "@rx ^."
Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:901450
Description
Sampling: Disable the rule engine based on sampling_percentage %{TX.sampling_percentage} and random number %{TX.sampling_rnd100}
Configured Variable
SecMarker "END-SAMPLING"
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable

0
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:931100
Description
Possible Remote File Inclusion (RFI) Attack: URL Parameter using IP Address
Configured Variable
SecRule ARGS @rx ^(?ifile|ftps?|https?)://(?:\d{13}\.\d{13}\.\d{13}\.\d{13})
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable
SecRule ARGS @rx ^(?ifile|ftps?|https?):\/\/(?:\d{13}\.\d{13}\.\d{13}\.\d{13})
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:931110
Description
Possible Remote File Inclusion (RFI) Attack: Common RFI Vulnerable Parameter Name used w/URL Payload
Configured Variable
SecRule QUERY_STRING|REQUEST_BODY @rx (?i)(?binclude\s*\([^)]*|mosConfig_absolute_path|_CONF\[path\]|_SERVER\[DOCUMENT_ROOT\]|GALLERY_BASEDIR|path\[docroot\]|appserv_root|config\[root_dir\])=(?:file|ftps?|https?)://
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n-

(?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule QUERY_STRING|REQUEST_BODY @rx (?i)(?binclude\s*\([^)]*|mosConfig_absolute_path|_CONF\[path\]|_SERVER\[DOCUMENT_ROOT\]|GALLERY_BASEDIR|path\[docroot\]|appserv_root|config\[root_dir\])=(?:file|ftps?|https?):\/\/

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:931130

Description

Possible Remote File Inclusion (RFI) Attack: Off-Domain Reference/Link

Configured Variable

SecRule ARGS @rx (?i)(?(?:url|jar):)?(?:a(?:cap|f[ps]|ttachment)|b(?:eshare|itcoin|lob)|c(?:a(?:llto|p)|id|vs|ompress.(?:zlib|bzip2))|d(?:a(?:v|ta)|ict|n(?:s|tp))|e(?:d2k|xpect)|f(?:(?:ee)?d|i(?:le|nger|sh)|tps?)|g(?:it|o(?:pher)?|lob)|h(?:323|ttps?)|i(?:ax|cap|(?:ma|p)ps?|rc[6s]?)|ja(?:bbe)?r|l(?:dap[is]?|ocal_file)|m(?:a(?:ilto|ven)|ms|umble)|n(?:e(?:tdoc|ws)|fs|ntps?)|ogg|p(?:aparazzi|h(?:ar|p)|op(?:2|3s?)|r(?:es|oxy)|syc)|r(?:mi|sync|tm(?:f?p)?|ar)|s(?:3|ftp|ips?|m(?:[bs]|tps?)|n(?:ews|mp)|sh(?:2(?:.(?:s(?:hell|(?:ft|c)p)|exec|tunnel))?)?|vn

(?:\+ssh)?)|t(?:e(?:amspeak|lnet)|ftp|urns?)|u(?:dp|nreal|t2004)|v(?:entrilo|iew-source|nc)|w(?:ebcal|s s?)|x(?:mpp|ri)|zip)://(?:[^@]+@)?([^/]*)

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAM ES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule ARGS @rx ^(?ifile|ftps?|https?)://([^/]*).*$

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAM ES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:950130

Description

Directory Listing
Configured Variable
SecRule RESPONSE_BODY @rx (?<(?:TITLE>Index of.*?<H|title>Index of.*?<h)1>Index of|>\[To Parent Directory\]</[Aa]>)
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule RESPONSE_BODY @rx (?<(?:TITLE>Index of.*?<H|title>Index of.*?<h)1>Index of|>\[To Parent Directory\]<\/[Aa]>)
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:932115
Description
Remote Command Execution: Windows Command Injection
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)(?:t[\\^]*i[\\^]*m[\\^]*e|[\n\r;`\{]|||?|&&?)[\s\v]*[\s\v\-\(@]*(?:[\\.-9A-Z_a-z]+/|(?:[\\x5c\^]*[0-9A-Z_a-z][\\x5c\^]*:.*|[\\.-9A-Z\x5c\^-_a-z]*)\x5c)?[\\^]*(?:o[\\^]*(?:d[\\^]*b[\\^]*c[\\^]*(?:a[\\^]*d[\\^]*3[\\^]*2|c[\\^]*o[\\^]*n[\\^]*f)|p[\\^]*e[\\^]*n[\\^]*f[\\^]*i[\\^]*l[\\^]*e[\\^]*s)|p[\\^]*(?:a[\\^]*t[\\^]*h[\\^]*(?:[\s\v\.-/;-<>].*|p[\\^]*i[\\^]*n[\\^]*g)|e[\\^]*r[\\^]*(?:f[\\^]*m[\\^]*o[\\^]*n|l(?:[\\^]*(?:5|s[\\^]*h))?)|h[\\^]*p(?:[\\^]*[57])?|i[\\^]*n[\\^]*g|k[\\^]*g[\\^]*m[\\^]*g[\\^]*r|o[\\^]*(?:p[\\^]*d|r[\\^]*t[\\^]*q[\\^]*r[\\^]*y|w[\\^]*e[\\^]*r[\\^]*(?:c[\\^]*f[\\^]*g|s[\\^]*h[\\^]*e[\\^]*l[\\^]*l(?:[\\^]*_[\\^]*i[\\^]*s[\\^]*e)?))|r[\\^]*(?:i[\\^]*n[\\^]*t[\\^]*(?:[\s\v\.-/;-<>].*|b[\\^]*r[\\^]*m)|n[\\^]*(?:c[\\^]*n[\\^]*f[\\^]*g|m[\\^]*n[\\^]*g[\\^]*r)|o[\\^]*m[\\^]*p[\\^]*t)|s[\\^]*(?:e[\\^]*x[\\^]*e[\\^]*c|f[\\^]*i[\\^]*l[\\^]*e|g[\\^]*e[\\^]*t[\\^]*s[\\^]*i[\\^]*d|i[\\^]*n[\\^]*f[\\^]*o|k[\\^]*i[\\^]*l[\\^]*l|l[\\^]*(?:i[\\^]*s[\\^]*t|o[\\^]*g[\\^]*(?:g[\\^]*e[\\^]*d[\\^]*o[\\^]*n|l[\\^]*i[\\^]*s[\\^]*t))|p[\\^]*(?:a[\\^]*s[\\^]*s[\\^]*w[\\^]*d|i[\\^]*n[\\^]*g)|s[\\^]*(?:e[\\^]*r[\\^]*v[\\^]*i[\\^]*c[\\^]*e|h[\\^]*u[\\^]*t[\\^]*d[\\^]*o[\\^]*w[\\^]*n|u[\\^]*s[\\^]*p[\\^]*e[\\^]*n[\\^]*d))|u[\\^]*s[\\^]*h[\\^]*d|y[\\^]*t[\\^]*h[\\^]*o[\\^]*n(?:[\\^]*(?:2|3(?:[\\^]*m)?))?)|q[\\^]*(?:g[\\^]*r[\\^]*e[\\^]*p|p[\\^]*r[\\^]*o[\\^]*c[\\^]*e[\\^]*s[\\^]*s|u[\\^]*e[\\^]*r[\\^]*y[\\^]*[\s\v\.-/;-<>].*|w[\\^]*i[\\^]*n[\\^]*s[\\^]*t[\\^]*a)|r[\\^]*(?:a[\\^]*(?:r[\\^]*[\s\v\.-/;-<>].*|s[\\^]*(?:d[\\^]*i[\\^]*a[\\^]*l|p[\\^]*h[\\^]*o[\\^]*n[\\^]*e))|d[\\^]*[\s\v\.-/;-<>].*|e[\\^]*(?:c[\\^]*(?:d[\\^]*i[\\^]*s[\\^]*c|o[\\^]*v[\\^]*e[\\^]*r)|g[\\^]*(?:[\s\v\.-/;-<>].*|e[\\^]*d[\\^]*i[\\^]*t|i[\\^]*n[\\^]*i|s[\\^]*v[\\^]*r[\\^]*3[\\^]*2)|k[\\^]*e[\\^]*y[\\^]*w[\\^]*i[\\^]*z|(?:n[\\^]*(?:a[\\^]*m[\\^]*e[\\^]*)?|(?:p[\\^]*l[\\^]*a[\\^]*c[\\^]*e|s[\\^]*e[\\^]*t)[\\^]*)[\s\v\.-/;-<>].*)|m[\\^]*(?:(?:d[\\^]*i[\\^]*r[\\^]*)?[\s\v\.-/;-<>].*|t[\\^]*s[\\^]*h[\\^]*a[\\^]*r[\\^]*e)|o[\\^]*(?:b[\\^]*o[\\^]*c[\\^]*o[\\^]*p[\\^]*y|u[\\^]*t[\\^]*e[\\^]*[\s\v\.-/;-<>].*)|s[\\^]*(?:t[\\^]*r[\\^]*u[\\^]*i|y[\\^]*n[\\^]*c)|u[\\^]*(?:b[\\^]*y[\\^]*(?:1(?:[\\^]*[8-9])?|2[\\^]*[0-2])|n[\\^]*(?:a[\\^]*s|d[\\^]*l[\\^]*l[\\^]*3[\\^]*2)))|s[\\^]*(?:c[\\^]*(?:h[\\^]*t[\\^]*a[\\^]*s[\\^]*k[\\^]*s|l[\\^]*i[\\^]*s[\\^]*t)|e[\\^]*(?:c[\\^]*p[\\^]*o[\\^]*l|l[\\^]*e[\\^]*c[\\^]*t|t[\\^]*(?:(?:x[\\^]*)?[\s\v\.-/;-<>].*|l[\\^]*o[\\^]*c[\\^]*a[\\^]*l))|f[\\^]*c|h[\\^]*(?:a[\\^]*r[\\^]*e|e[\\^]*l[\\^]*l[\\^]*r[\\^]*u[\\^]*n[\\^]*a[\\^]*s|i[\\^]*f[\\^]*t|o[\\^]*(?:r[\\^]*t[\\^]*c[\\^]*u[\\^]*t|w[\\^]*(?:g[\\^]*r[\\^]*p|m[\\^]*b[\\^]*r)[\\^]*s)|r[\\^]*p[\\^]*u[\\^]*b[\\^]*w|u[\\^]*t[\\^]*d[\\^]*o[\\^]*w[\\^]*n)|i[\\^]*g[\\^]*v[\\^]*e[\\^]*r[\\^]*i[\\^]*f|l[\\^]*(?:e[\\^]*e[\\^]*p|m[\\^]*g[\\^]*r)|(?:o|t[\\^]*a)[\\^]*r[\\^]*t[\\^]*[\s\v\.-/;-<>].*|u[\\^]*b[\\^]*(?:i[	

\\^]*n[\\^]*a[\\^]*c[\\^]*l|s[\\^]*t)|v[\\^]*n|y[\\^]*s[\\^]*(?:d[\\^]*m|k[\\^]*e[\\^]*y|t[\\^]*e[\\^]*m[\\^]*(?:i[\\^]*n[\\^]*f[\\^]*o|p[\\^]*r[\\^]*o[\\^]*p[\\^]*e[\\^]*r[\\^]*t[\\^]*i[\\^]*e[\\^]*s[\\^]*(?:a[\\^]*d[\\^]*v[\\^]*a[\\^]*n[\\^]*c[\\^]*e[\\^]*d|d[\\^]*a[\\^]*t[\\^]*a[\\^]*e[\\^]*x[\\^]*e[\\^]*c[\\^]*u[\\^]*t[\\^]*i[\\^]*o[\\^]*n[\\^]*p[\\^]*r[\\^]*e[\\^]*v[\\^]*e[\\^]*n[\\^]*t[\\^]*i[\\^]*o[\\^]*n|(?:h[\\^]*a[\\^]*r[\\^]*d[\\^]*w[\\^]*a[\\^]*r|p[\\^]*e[\\^]*r[\\^]*f[\\^]*o[\\^]*r[\\^]*m[\\^]*a[\\^]*n[\\^]*c)[\\^]*e)))))|t[\\^]*(?:a[\\^]*(?:k[\\^]*e[\\^]*o[\\^]*w[\\^]*n|s[\\^]*k[\\^]*(?:k[\\^]*i[\\^]*l[\\^]*l|l[\\^]*i[\\^]*s[\\^]*t|m[\\^]*g[\\^]*r|s[\\^]*c[\\^]*h[\\^]*d))|(?:e[\\^]*l[\\^]*n[\\^]*e|i[\\^]*m[\\^]*e[\\^]*o[\\^]*u|l[\\^]*i[\\^]*s|p[\\^]*m[\\^]*i[\\^]*n[\\^]*i)[\\^]*t|r[\\^]*(?:a[\\^]*c[\\^]*e[\\^]*r[\\^]*t|e[\\^]*e)|s[\\^]*(?:d[\\^]*i[\\^]*s[\\^]*c[\\^]*o|s[\\^]*h[\\^]*u[\\^]*t[\\^]*d)[\\^]*n|y[\\^]*p[\\^]*e[\\^]*(?:[\s\v\.-/;-<>].*|p[\\^]*e[\\^]*r[\\^]*f))|u[\\^]*(?:n[\\^]*(?:r[\\^]*a[\\^]*r|z[\\^]*i[\\^]*p)|s[\\^]*(?:e[\\^]*r[\\^]*a[\\^]*c[\\^]*c[\\^]*o[\\^]*u[\\^]*n[\\^]*t[\\^]*c[\\^]*o[\\^]*n[\\^]*t[\\^]*r[\\^]*o[\\^]*l[\\^]*s[\\^]*e[\\^]*t[\\^]*t[\\^]*i[\\^]*n[\\^]*g[\\^]*s|r[\\^]*s[\\^]*t[\\^]*a[\\^]*t))|v[\\^]*(?:e[\\^]*r[\\^]*i[\\^]*f[\\^]*y|o[\\^]*l[\\^]*[\s\v\.-/;-<>].*)|w[\\^]*(?:a[\\^]*i[\\^]*t[\\^]*f[\\^]*o[\\^]*r|e[\\^]*v[\\^]*t[\\^]*u[\\^]*t[\\^]*i[\\^]*l|g[\\^]*e[\\^]*t|h[\\^]*o[\\^]*a[\\^]*m[\\^]*i|i[\\^]*n[\\^]*(?:d[\\^]*i[\\^]*f[\\^]*f|m[\\^]*s[\\^]*d[\\^]*p|r[\\^]*[ms]|v[\\^]*a[\\^]*r)|m[\\^]*i[\\^]*(?:c|m[\\^]*g[\\^]*m[\\^]*t)|s[\\^]*c[\\^]*(?:r[\\^]*i[\\^]*p[\\^]*t|u[\\^]*i)|u[\\^]*(?:a[\\^]*(?:p[\\^]*p|u[\\^]*c[\\^]*l[\\^]*t)|s[\\^]*a))|x[\\^]*c[\\^]*(?:a[\\^]*c[\\^]*l[\\^]*s|o[\\^]*p[\\^]*y)|z[\\^]*i[\\^]*p[\\^]*[\s\v\.-/;-<>].*)(?:\.[\\^]*[0-9A-Z_a-z]+)?\b

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?i)(?:;|\{|\|\|\||&|&&|\n|\r|`)\s*[\(@\\s]*(?:[\w\\./]+/|[\\\\\^]*\w[\\\\\^]*:.*\\\\|[\^\.\w

\/\\\\]*\\\\)?[\\^]*(?:s[\\^]*(?:y[\\^]*s[\\^]*(?:t[\\^]*e[\\^]*m[\\^]*(?:p[\\^]*r[\\^]*o[\\^]*p[\\^]*e[\\^]*r[\\^]*t[\\^]*i[\\^]*e[\\^]*s[\\^]*(?:d[\\^]*a[\\^]*t[\\^]*a[\\^]*e[\\^]*x[\\^]*e[\\^]*c[\\^]*u[\\^]*t[\\^]*i[\\^]*o[\\^]*n[\\^]*p[\\^]*r[\\^]*e[\\^]*v[\\^]*e[\\^]*n[\\^]*t[\\^]*i[\\^]*o[\\^]*n|(?:p[\\^]*e[\\^]*r[\\^]*f[\\^]*o[\\^]*r[\\^]*m[\\^]*a[\\^]*n[\\^]*c|h[\\^]*a[\\^]*r[\\^]*d[\\^]*w[\\^]*a[\\^]*r)[\\^]*e|a[\\^]*d[\\^]*v[\\^]*a[\\^]*n[\\^]*c[\\^]*e[\\^]*d)|i[\\^]*n[\\^]*f[\\^]*o)|k[\\^]*e[\\^]*y|d[\\^]*m)|h[\\^]*(?:o[\\^]*(?:w[\\^]*(?:g[\\^]*r[\\^]*p|m[\\^]*b[\\^]*r)[\\^]*s|r[\\^]*t[\\^]*c[\\^]*u[\\^]*t)|e[\\^]*l[\\^]*l[\\^]*r[\\^]*u[\\^]*n[\\^]*a[\\^]*s|u[\\^]*t[\\^]*d[\\^]*o[\\^]*w[\\^]*n|r[\\^]*p[\\^]*u[\\^]*b[\\^]*w|a[\\^]*r[\\^]*e|i[\\^]*f[\\^]*t)|e[\\^]*(?:t[\\^]*(?:(?:x[\\^]*)?(?:[\s;]|\.|/|<|>).*|l[\\^]*o[\\^]*c[\\^]*a[\\^]*l)|c[\\^]*p[\\^]*o[\\^]*l|l[\\^]*e[\\^]*c[\\^]*t)|c[\\^]*(?:h[\\^]*t[\\^]*a[\\^]*s[\\^]*k[\\^]*s|l[\\^]*i[\\^]*s[\\^]*t)|u[\\^]*b[\\^]*(?:i[\\^]*n[\\^]*a[\\^]*c[\\^]*l|s[\\^]*t)|t[\\^]*a[\\^]*r[\\^]*t[\\^]*(?:[\s;]|\.|/|<|>).*|i[\\^]*g[\\^]*v[\\^]*e[\\^]*r[\\^]*i[\\^]*f|l[\\^]*(?:e[\\^]*e[\\^]*p|m[\\^]*g[\\^]*r)|o[\\^]*r[\\^]*t|f[\\^]*c|v[\\^]*n)|p[\\^]*(?:s[\\^]*(?:s[\\^]*(?:h[\\^]*u[\\^]*t[\\^]*d[\\^]*o[\\^]*w[\\^]*n|e[\\^]*r[\\^]*v[\\^]*i[\\^]*c[\\^]*e|u[\\^]*s[\\^]*p[\\^]*e[\\^]*n[\\^]*d)|l[\\^]*(?:o[\\^]*g[\\^]*(?:g[\\^]*e[\\^]*d[\\^]*o[\\^]*n|l[\\^]*i[\\^]*s[\\^]*t)|i[\\^]*s[\\^]*t)|p[\\^]*(?:a[\\^]*s[\\^]*s[\\^]*w[\\^]*d|i[\\^]*n[\\^]*g)|g[\\^]*e[\\^]*t[\\^]*s[\\^]*i[\\^]*d|e[\\^]*x[\\^]*e[\\^]*c|f[\\^]*i[\\^]*l[\\^]*e|i[\\^]*n[\\^]*f[\\^]*o|k[\\^]*i[\\^]*l[\\^]*l)|o[\\^]*(?:w[\\^]*e[\\^]*r[\\^]*(?:s[\\^]*h[\\^]*e[\\^]*l[\\^]*l(?:[\\^]*_[\\^]*i[\\^]*s[\\^]*e)?|c[\\^]*f[\\^]*g)|r[\\^]*t[\\^]*q[\\^]*r[\\^]*y|p[\\^]*d)|r[\\^]*(?:i[\\^]*n[\\^]*t[\\^]*(?:(?:[\s;]|\.|/|<|>).*|b[\\^]*r[\\^]*m)|n[\\^]*(?:c[\\^]*n[\\^]*f[\\^]*g|m[\\^]*n[\\^]*g[\\^]*r)|o[\\^]*m[\\^]*p[\\^]*t)|a[\\^]*t[\\^]*h[\\^]*(?:p[\\^]*i[\\^]*n[\\^]*g|(?:[\s;]|\.|/|<|>).*)|e[\\^]*r[\\^]*(?:l(?:[\\^]*(?:s[\\^]*h|5))?|f[\\^]*m[\\^]*o[\\^]*n)|y[\\^]*t[\\^]*h[\\^]*o[\\^]*n(?:[\\^]*(?:3(?:[\\^]*m)?|2))?|k[\\^]*g[\\^]*m[\\^]*g[\\^]*r|h[\\^]*p(?:[\\^]*[57])?|u[\\^]*s[\\^]*h[\\^]*d|i[\\^]*n[\\^]*g)|r[\\^]*(?:e[\\^]*(?:(?:p[\\^]*l[\\^]*a[\\^]*c[\\^]*e|n(?:[\\^]*a[\\^]*m[\\^]*e)?|s[\\^]*e[\\^]*t)[\\^]*(?:[\s;]|\.|/|<|>).*|g[\\^]*(?:s[\\^]*v[\\^]*r[\\^]*3[\\^]*2|e[\\^]*d[\\^]*i[\\^]*t|(?:[\s;]|\.|/|<|>).*|i[\\^]*n[\\^]*i)|c[\\^]*(?:d[\\^]*i[\\^]*s[\\^]*c|o[\\^]*v[\\^]*e[\\^]*r)|k[\\^]*e[\\^]*y[\\^]*w[\\^]*i[\\^]*z)|u[\\^]*(?:n[\\^]*(?:d[\\^]*l[\\^]*l[\\^]*3[\\^]*2|a[\\^]*s)|b[\\^]*y[\\^]*(?:1(?:[\\^]*[89])?|2[\\^]*[012]))|a[\\^]*(?:s[\\^]*(?:p[\\^]*h[\\^]*o[\\^]*n[\\^]*e|d[\\^]*i[\\^]*a[\\^]*l)|r[\\^]*(?:[\s;]|\.|/|<|>).*)|m[\\^]*(?:(?:d[\\^]*i[\\^]*r[\\^]*)?(?:[\s;]|\.|/|<|>).*|t[\\^]*s[\\^]*h[\\^]*a[\\^]*r[\\^]*e)|o[\\^]*(?:u[\\^]*t[\\^]*e[\\^]*(?:[\s;]|\.|/|<|>).*|b[\\^]*o[\\^]*c[\\^]*o[\\^]*p[\\^]*y)|s[\\^]*(?:t[\\^]*r[\\^]*u[\\^]*i|y[\\^]*n[\\^]*c)|d[\\^]*(?:[\s;]|\.|/|<|>).*)|t[\\^]*(?:a[\\^]*(?:s[\\^]*k[\\^]*(?:k[\\^]*i[\\^]*l[\\^]*l|l[\\^]*i[\\^]*s[\\^]*t|s[\\^]*c[\\^]*h[\\^]*d|m[\\^]*g[\\^]*r)|k[\\^]*e[\\^]*o[\\^]*w[\\^]*n)|(?:i[\\^]*m[\\^]*e[\\^]*o[\\^]*u|p[\\^]*m[\\^]*i[\\^]*n[\\^]*i|e[\\^]*l[\\^]*n[\\^]*e|l[\\^]*i[\\^]*s)[\\^]*t|s[\\^]*(?:d[\\^]*i[\\^]*s[\\^]*c[\\^]*o|s[\\^]*h[\\^]*u[\\^]*t[\\^]*d)[\\^]*n|y[\\^]*p[\\^]*e[\\^]*(?:p[\\^]*e[\\^]*r[\\^]*f|(?:[\s;]|\.|/|<|>).*)|r[\\^]*(?:a[\\^]*c[\\^]*e[\\^]*r[\\^]*t|e[\\^]*e))|w[\\^]*(?:i[\\^]*n[\\^]*(?:d[\\^]*i[\\^]*f[\\^]*f|m[\\^]*s[\\^]*d[\\^]*p|v[\\^]*a[\\^]*r|r[\\^]*[ms])|u[\\^]*(?:a[\\^]*(?:u[\\^]*c[\\^]*l[\\^]*t|p[\\^]*p)|s[\\^]*a)|s[\\^]*c[\\^]*(?:r[\\^]*i[\\^]*p[\\^]*t|u[\\^]*i)|e[\\^]*v[\\^]*t[\\^]*u[\\^]*t[\\^]*i[\\^]*l|m[\\^]*i[\\^]*(?:m[\\^]*g[\\^]*m[\\^]*t|c)|a[\\^]*i[\\^]*t[\\^]*f[\\^]*o[\\^]*r|h[\\^]*o[\\^]*a[\\^]*m[\\^]*i|g[\\^]*e[\\^]*t)|u[\\^]*(?:s[\\^]*(?:e[\\^]*r[\\^]*

a[\\^]*c[\\^]*c[\\^]*o[\\^]*u[\\^]*n[\\^]*t[\\^]*c[\\^]*o[\\^]*n[\\^]*t[\\^]*r[\\^]*o[\\^]*l[\\^]*s[\\^]*e[\\^]*t[\\^]*t[\\^]*i[\\^]*n[\\^]*g[\\^]*s|r[\\^]*s[\\^]*t[\\^]*a[\\^]*t)|n[\\^]*(?:r[\\^]*a[\\^]*r|z[\\^]*i[\\^]*p))|q[\\^]*(?:u[\\^]*e[\\^]*r[\\^]*y[\\^]*(?:[\s;]|\.|/|<|>).*|p[\\^]*r[\\^]*o[\\^]*c[\\^]*e[\\^]*s[\\^]*s|w[\\^]*i[\\^]*n[\\^]*s[\\^]*t[\\^]*a|g[\\^]*r[\\^]*e[\\^]*p)|o[\\^]*(?:d[\\^]*b[\\^]*c[\\^]*(?:a[\\^]*d[\\^]*3[\\^]*2|c[\\^]*o[\\^]*n[\\^]*f)|p[\\^]*e[\\^]*n[\\^]*f[\\^]*i[\\^]*l[\\^]*e[\\^]*s)|v[\\^]*(?:o[\\^]*l[\\^]*(?:[\s;]|\.|/|<|>).*|e[\\^]*r[\\^]*i[\\^]*f[\\^]*y)|x[\\^]*c[\\^]*(?:a[\\^]*c[\\^]*l[\\^]*s|o[\\^]*p[\\^]*y)|z[\\^]*i[\\^]*p[\\^]*(?:[\s;]|\.|/|<|>).*)(?:\.[\\^]*\w+)?\b

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:932130

Description

Remote Command Execution: Unix Shell Expression Found

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx \$(?:\((?:.*|\(.*\))\)|\{.*\})|[<>]\(.*\)|/[0-9A-Z_a-z]*\[!?.+\]

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAM

ES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx (?:\$(?:\((?:\(.*\)|.*)\)|\{.*\})|[<>]\(.*\))

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:932140

Description

Remote Command Execution: Windows FOR/IF Command Found

Configured Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx \b(?:for(?:/[dflr].*)? %+[^]+ in\(.*\)[\s\v]?do|if(?:/i)?(?: not)?(?:

(?:e(?:xist|rrorlevel)|defined|cmdextversion)\b|[\(].*(?:\b(?:g(?:eq|tr)|equ|neq|l(?:eq|ss))\b|==)))

Configured Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Recommended Variable

SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx \b(?:if(?:/i)?(?: not)?(?: exist\b| defined\b| errorlevel\b| cmdextversion\b|(?:|\().*(?:\bgeq\b|\bequ\b|\bneq\b|\bleq\b|\bgtr\b|\blss\b|==))|for(?:/[dflr].*)? %+[^]+ in\(.*\)\s?do)

Recommended Regex Explanation

This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:932200
Description
RCE Bypass Technique
Configured Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|REQUEST_HEADERS:Referer|REQUEST_HEADERS:User-Agent|ARGS_NAMES|ARGS|XML:/* @rx (?:[*?`\x5c][^/\n]+/|\$[({\[#@!?*\-_$a-zA-Z0-9]|/[^/]+?[*?`\x5c])
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$\|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/* @rx ([*?`\\][^/\n]+/|\$[({\[#a-zA-Z0-9]|/[^/]+?[*?`\\])
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a

ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Rule ID:932190
Description
Remote Command Execution: Wildcard bypass technique attempt
Configured Variable
SecRule ARGS @rx /(?[?*]+[a-z/]+|[a-z/]+[?*]+)
Configured Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.
Recommended Variable
SecRule ARGS @rx (?/|\\\\)(?:[\?*]+[a-z/\\\\]+|[a-z/\\\\]+[\?*]+)
Recommended Regex Explanation
This expression is used to detect potential PHP code injection attacks in web requests. \n\nThe regex expression consists of several parts: \n- SecRule: This is a directive in ModSecurity, a web application firewall, that specifies a rule to be applied to incoming requests.\n- REQUEST_COOKIES|!REQUEST_COOKIES/__utm/|REQUEST_COOKIES_NAMES|ARGS_NAMES|ARGS|XML:/*: These are variables that represent different parts of the request that the rule

should be applied to. In this case, it includes cookies, arguments, and XML data.\n- @rx: This is a ModSecurity operator that specifies a regular expression to match against the selected variables.\n- (?:<\\?(?:[^x]|x[^m]|xm[^l]|xml[^\\s]|xml$|$)|<\\?php|\\[(?:/|\\x5c)?php\\]): This is the regular expression itself, which matches common PHP opening tags, including short tags and the opening tag with a leading square bracket.\n\nFor example, if a request contains a PHP opening tag in a cookie or argument value, this rule would trigger and block the request.

Version

As of 2023-03-09 18:43:39.902797, the latest version of ModSecurity Core Rule Set (CRS) is: 4.0.0. The following rules are not using the latest version. Please check the latest version from https://github.com/coreruleset/coreruleset

Rule ID	Current Version	Latest Version
933100	3.3.0	4.0.0
933110	3.3.0	4.0.0
933120	3.3.0	4.0.0
933130	3.3.0	4.0.0
933140	3.3.0	4.0.0
933200	3.3.0	4.0.0
933150	3.3.0	4.0.0
933160	3.3.0	4.0.0
933170	3.3.0	4.0.0
933180	3.3.0	4.0.0
933210	3.3.0	4.0.0
933151	3.3.0	4.0.0
933131	3.3.0	4.0.0
933161	3.3.0	4.0.0
933111	3.3.0	4.0.0
933190	3.3.0	4.0.0
953100	3.3.0	4.0.0

953110	3.3.0	4.0.0
953120	3.3.0	4.0.0
934100	3.3.0	4.0.0
942100	3.3.0	4.0.0
942140	3.3.0	4.0.0
942160	3.3.0	4.0.0
942170	3.3.0	4.0.0
942190	3.3.0	4.0.0
942220	3.3.0	4.0.0
942230	3.3.0	4.0.0
942240	3.3.0	4.0.0
942250	3.3.0	4.0.0
942270	3.3.0	4.0.0
942280	3.3.0	4.0.0
942290	3.3.0	4.0.0
942320	3.3.0	4.0.0
942350	3.3.0	4.0.0
942360	3.3.0	4.0.0
942500	3.3.0	4.0.0
942110	3.3.0	4.0.0
942120	3.3.0	4.0.0
942130	3.3.0	4.0.0
942150	3.3.0	4.0.0
942180	3.3.0	4.0.0
942200	3.3.0	4.0.0
942210	3.3.0	4.0.0
942260	3.3.0	4.0.0
942300	3.3.0	4.0.0
942310	3.3.0	4.0.0
942330	3.3.0	4.0.0
942340	3.3.0	4.0.0

942361	3.3.0	4.0.0
942370	3.3.0	4.0.0
942380	3.3.0	4.0.0
942390	3.3.0	4.0.0
942400	3.3.0	4.0.0
942410	3.3.0	4.0.0
942470	3.3.0	4.0.0
942480	3.3.0	4.0.0
942430	3.3.0	4.0.0
942440	3.3.0	4.0.0
942450	3.3.0	4.0.0
942510	3.3.0	4.0.0
942101	3.3.0	4.0.0
942251	3.3.0	4.0.0
942490	3.3.0	4.0.0
942420	3.3.0	4.0.0
942431	3.3.0	4.0.0
942460	3.3.0	4.0.0
942511	3.3.0	4.0.0
942421	3.3.0	4.0.0
942432	3.3.0	4.0.0
905100	3.3.0	4.0.0
905110	3.3.0	4.0.0
951100	3.3.0	4.0.0
951110	3.3.0	4.0.0
951120	3.3.0	4.0.0
951130	3.3.0	4.0.0
951140	3.3.0	4.0.0
951150	3.3.0	4.0.0
951160	3.3.0	4.0.0
951170	3.3.0	4.0.0

951180	3.3.0	4.0.0
951190	3.3.0	4.0.0
951200	3.3.0	4.0.0
951210	3.3.0	4.0.0
951220	3.3.0	4.0.0
951230	3.3.0	4.0.0
951240	3.3.0	4.0.0
951250	3.3.0	4.0.0
951260	3.3.0	4.0.0
943100	3.3.0	4.0.0
943110	3.3.0	4.0.0
943120	3.3.0	4.0.0
944110	3.3.0	4.0.0
944120	3.3.0	4.0.0
944130	3.3.0	4.0.0
944210	3.3.0	4.0.0
944240	3.3.0	4.0.0
944250	3.3.0	4.0.0
954100	3.3.0	4.0.0
954110	3.3.0	4.0.0
954120	3.3.0	4.0.0
954130	3.3.0	4.0.0
930100	3.3.0	4.0.0
930110	3.3.0	4.0.0
930120	3.3.0	4.0.0
930130	3.3.0	4.0.0
941100	3.3.0	4.0.0
941110	3.3.0	4.0.0
941130	3.3.0	4.0.0
941140	3.3.0	4.0.0
941160	3.3.0	4.0.0

941170	3.3.0	4.0.0
941180	3.3.0	4.0.0
941190	3.3.0	4.0.0
941200	3.3.0	4.0.0
941210	3.3.0	4.0.0
941220	3.3.0	4.0.0
941230	3.3.0	4.0.0
941240	3.3.0	4.0.0
941250	3.3.0	4.0.0
941260	3.3.0	4.0.0
941270	3.3.0	4.0.0
941280	3.3.0	4.0.0
941290	3.3.0	4.0.0
941300	3.3.0	4.0.0
941310	3.3.0	4.0.0
941350	3.3.0	4.0.0
941360	3.3.0	4.0.0
941370	3.3.0	4.0.0
941101	3.3.0	4.0.0
941120	3.3.0	4.0.0
941150	3.3.0	4.0.0
941320	3.3.0	4.0.0
941330	3.3.0	4.0.0
941340	3.3.0	4.0.0
941380	3.3.0	4.0.0
949110	3.3.0	4.0.0
920100	3.3.0	4.0.0
920120	3.3.0	4.0.0
920160	3.3.0	4.0.0
920170	3.3.0	4.0.0
920171	3.3.0	4.0.0

920180	3.3.0	4.0.0
920181	3.3.0	4.0.0
920190	3.3.0	4.0.0
920210	3.3.0	4.0.0
920220	3.3.0	4.0.0
920240	3.3.0	4.0.0
920250	3.3.0	4.0.0
920260	3.3.0	4.0.0
920270	3.3.0	4.0.0
920280	3.3.0	4.0.0
920290	3.3.0	4.0.0
920310	3.3.0	4.0.0
920311	3.3.0	4.0.0
920330	3.3.0	4.0.0
920340	3.3.0	4.0.0
920350	3.3.0	4.0.0
920380	3.3.0	4.0.0
920360	3.3.0	4.0.0
920370	3.3.0	4.0.0
920390	3.3.0	4.0.0
920400	3.3.0	4.0.0
920410	3.3.0	4.0.0
920470	3.3.0	4.0.0
920420	3.3.0	4.0.0
920480	3.3.0	4.0.0
920430	3.3.0	4.0.0
920440	3.3.0	4.0.0
920500	3.3.0	4.0.0
920450	3.3.0	4.0.0
920200	3.3.0	4.0.0
920201	3.3.0	4.0.0

920230	3.3.0	4.0.0
920271	3.3.0	4.0.0
920320	3.3.0	4.0.0
920121	3.3.0	4.0.0
920341	3.3.0	4.0.0
920272	3.3.0	4.0.0
920300	3.3.0	4.0.0
920490	3.3.0	4.0.0
920510	3.3.0	4.0.0
920202	3.3.0	4.0.0
920273	3.3.0	4.0.0
920274	3.3.0	4.0.0
920275	3.3.0	4.0.0
920460	3.3.0	4.0.0
959100	3.3.0	4.0.0
952100	3.3.0	4.0.0
952110	3.3.0	4.0.0
921110	3.3.0	4.0.0
921120	3.3.0	4.0.0
921130	3.3.0	4.0.0
921140	3.3.0	4.0.0
921150	3.3.0	4.0.0
921160	3.3.0	4.0.0
921190	3.3.0	4.0.0
921200	3.3.0	4.0.0
921151	3.3.0	4.0.0
921170	3.3.0	4.0.0
921180	3.3.0	4.0.0
911100	3.3.0	4.0.0
901001	3.3.0	4.0.0
901100	3.3.0	4.0.0

901110	3.3.0	4.0.0
901120	3.3.0	4.0.0
901125	3.3.0	4.0.0
901130	3.3.0	4.0.0
901140	3.3.0	4.0.0
901141	3.3.0	4.0.0
901142	3.3.0	4.0.0
901143	3.3.0	4.0.0
901160	3.3.0	4.0.0
901162	3.3.0	4.0.0
901168	3.3.0	4.0.0
901163	3.3.0	4.0.0
901164	3.3.0	4.0.0
901165	3.3.0	4.0.0
901167	3.3.0	4.0.0
901200	3.3.0	4.0.0
901321	3.3.0	4.0.0
901340	3.3.0	4.0.0
901350	3.3.0	4.0.0
901400	3.3.0	4.0.0
901410	3.3.0	4.0.0
901450	3.3.0	4.0.0
901500	3.3.0	4.0.0
931100	3.3.0	4.0.0
931110	3.3.0	4.0.0
931120	3.3.0	4.0.0
931130	3.3.0	4.0.0
950130	3.3.0	4.0.0
950140	3.3.0	4.0.0
950100	3.3.0	4.0.0
932115	3.3.0	4.0.0

932120	3.3.0	4.0.0
932130	3.3.0	4.0.0
932140	3.3.0	4.0.0
932160	3.3.0	4.0.0
932170	3.3.0	4.0.0
932171	3.3.0	4.0.0
932180	3.3.0	4.0.0
932200	3.3.0	4.0.0
932190	3.3.0	4.0.0
913100	3.3.0	4.0.0
913110	3.3.0	4.0.0
913120	3.3.0	4.0.0
913101	3.3.0	4.0.0
913102	3.3.0	4.0.0

Severity

The severity level of a rule in ModSecurity CRS affects the score that is assigned to a particular event or anomaly detected by that rule. Each severity level has a different weight or impact on the overall score that is calculated for a particular request.

The severity levels are as follows:

CRITICAL (level 5)
Indicates that the anomaly detected by the rule is very severe and requires immediate attention. A request that triggers such a rule would be assigned a high score, which would indicate that it is likely an attack
ERROR (Level 4)
Indicates that the anomaly is serious and could result in a security breach if not addressed. A request that triggers such a rule would be assigned a high score, which would indicate that it is potentially malicious.
WARNING (Level 3)
Indicates that the anomaly is of moderate severity and could potentially lead to a security issue. A request that triggers such a rule would be assigned a lower score than a critical or error-level rule.
NOTICE (Level 2)
Indicates that the anomaly is of low severity and may not necessarily indicate an attack or security issue. A request that triggers such a rule would be assigned a low score.

It is important to configure the WAF rules based on the severity of the application's security needs. If the configured WAF rules have a lower severity than the OWASP CRS Guideline rules, it may result in a higher risk of successful attacks.

The following rules have different severity

Rule ID	Configured Severity	Recommended Severity
949110	CRITICAL	None

Action

In ModSecurity, "pass", "deny", and "block" are actions that can be taken by a rule when a request or response matches that rule.

The various actions are as follows:

pass
Rule will be skipped and the request/response will be allowed to continue through the WAF without being blocked or flagged as an anomaly.
deny
Request will be blocked, and the client will receive a response indicating that their request was denied.
block
similar to "deny" in that it also blocks the request, but it also generates an event that can be logged and alerts the WAF administrator to the attempted attack.

These actions are usually associated with the severity level of a rule, with higher severity rules being more likely to "deny" or "block" a request. The specific actions taken by a rule depend on the configuration of the WAF, including the desired level of protection, the sensitivity of the protected application, and the likelihood of false positives.
Having current configured rules to have lower restrictive actions such as "pass" when it should be a higher restrictive action such as "deny" can leave the system vulnerable to attacks.

The following rules have different actions:

Rule ID	Configured Action	Recommended Action
901001	pass	deny

Scoring

The OWASP CRS anomaly scoring system is derived by combining two factors: severity and paranoia level, with severity carrying a higher weight than paranoia level.

Severity
Is determined by the type of attack and the potential impact it could have on the system.
Paranoia
Reflects the likelihood that the rule could generate false positives or block legitimate traffic.

The aim should be to achieve a score as close as possible to the score of the OWASP CRS Guideline. This indicates that the WAF configuration is aligned with the best practices outlined in the guideline.
However, having a higher severity or paranoia level does not necessarily mean a rule is more secure. It means that the rule is more likely to detect and potentially block an attack that matches the rule's criteria. A rule with a high anomaly score is more likely to detect more sophisticated attacks, but it also increases the risk of false positives.
It is important to note that the anomaly scoring is just one aspect of WAF configuration management. Other factors such as the accuracy of the rules, false positives, and false negatives should also be considered in determining the effectiveness of the WAF. Please consider these factors with your organization's security objectives

Obtained Score: 609.5 / 862.9
Your average weighted score is lower than guideline by a significant amount. This suggests that your WAF may not be following the security posture recommended by the guideline and is not adequately protecting the web application against attacks. It is important to investigate the reasons for the low weighted score and take appropriate actions to improve the security of the web application. This may involve reviewing and updating the WAF rule sets according to the severity and paranoia levels stated in the guideline.

Rule Violations

This section outlines the rule violations detected by the WAF. In ModSecurity, rule violations are incidents when a request matches a defined rule or set of rules that are designed to detect and prevent malicious activity. When a rule is triggered, it generates a log entry in the ModSecurity audit log file, which contains information about the request, the rule that was triggered, and other relevant details. To view more information, please refer to the ModSecurity audit log file at /var/log/apache2/modsec_audit.log

It is important to keep track of rule violations in ModSecurity audit logs because it can help to detect and prevent potential security threats to your web application. By reviewing the log entries, you can identify patterns of suspicious activity or attacks and take appropriate measures to protect your application from those threats.

These are the violations found, shown with the rules triggered and number of hits.

Rule Violations:

Message: Warning. Pattern match "^[\\d.:]+$" at REQUEST_HEADERS:Host. [file "/etc/modsecurity/rules/REQUEST-920-PROTOCOL-ENFORCEMENT.conf"] [line "735"] [id "920350"] [msg "Host header is a numeric IP address"] [data "20.187.92.114"] [severity "WARNING"] [ver "OWASP_CRS/3.3.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-protocol"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/210/272"] [tag "PCI/6.5.10"]

Count: 14

Message: Warning. Pattern match "^[\\d.:]+$" at REQUEST_HEADERS:Host. [file "/etc/modsecurity/rules/REQUEST-920-PROTOCOL-ENFORCEMENT.conf"] [line "735"] [id "920350"] [msg "Host header is a numeric IP address"] [data "20.187.92.114:80"] [severity "WARNING"] [ver "OWASP_CRS/3.3.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-protocol"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/210/272"] [tag "PCI/6.5.10"]

Count: 4

Message: Warning. Matched phrase "zgrab" at REQUEST_HEADERS:User-Agent. [file

"/etc/modsecurity/rules/REQUEST-913-SCANNER-DETECTION.conf"] [line "54"] [id "913100"] [msg "Found User-Agent associated with security scanner"] [data "Matched Data: zgrab found within REQUEST_HEADERS:User-Agent: mozilla/5.0 zgrab/0.x"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-reputation-scanner"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/118/224/541/310"] [tag "PCI/6.5.10"]
Message: Warning. Pattern match "^[\\d.:]+$" at REQUEST_HEADERS:Host. [file "/etc/modsecurity/rules/REQUEST-920-PROTOCOL-ENFORCEMENT.conf"] [line "735"] [id "920350"] [msg "Host header is a numeric IP address"] [data "20.187.92.114"] [severity "WARNING"] [ver "OWASP_CRS/3.3.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-protocol"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/210/272"] [tag "PCI/6.5.10"]
Message: Access denied with code 403 (phase 2). Operator GE matched 5 at TX:anomaly_score. [file "/etc/modsecurity/rules/REQUEST-949-BLOCKING-EVALUATION.conf"] [line "93"] [id "949110"] [msg "Inbound Anomaly Score Exceeded (Total Score: 8)"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-generic"]
Message: Warning. Operator GE matched 5 at TX:inbound_anomaly_score. [file "/etc/modsecurity/rules/RESPONSE-980-CORRELATION.conf"] [line "91"] [id "980130"] [msg "Inbound Anomaly Score Exceeded (Total Inbound Score: 8 - SQLI=0,XSS=0,RFI=0,LFI=0,RCE=0,PHPI=0,HTTP=0,SESS=0): individual paranoia level scores: 8, 0, 0, 0"] [ver "OWASP_CRS/3.3.0"] [tag "event-correlation"]
Count: 2

Message: Warning. Pattern match "^[\\d.:]+$" at REQUEST_HEADERS:Host. [file "/etc/modsecurity/rules/REQUEST-920-PROTOCOL-ENFORCEMENT.conf"] [line "735"] [id "920350"] [msg "Host header is a numeric IP address"] [data "5.188.210.227"] [severity "WARNING"] [ver "OWASP_CRS/3.3.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-protocol"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/210/272"] [tag "PCI/6.5.10"]
Count: 1

Message: Warning. Pattern match "^[\\d.:]+$" at REQUEST_HEADERS:Host. [file "/etc/modsecurity/rules/REQUEST-920-PROTOCOL-ENFORCEMENT.conf"] [line "735"] [id "920350"] [msg "Host header is a numeric IP address"] [data "20.187.92.114"] [severity "WARNING"] [ver "OWASP_CRS/3.3.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-protocol"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/210/272"] [tag "PCI/6.5.10"]
Message: Warning. Matched phrase "/.env" at REQUEST_FILENAME. [file "/etc/modsecurity/rules/REQUEST-930-APPLICATION-ATTACK-LFI.conf"] [line "125"] [id "930130"] [msg "Restricted File Access Attempt"] [data "Matched Data: /.env found within REQUEST_FILENAME: /.env"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-lfi"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/255/153/126"] [tag "PCI/6.5.4"]
Message: Access denied with code 403 (phase 2). Operator GE matched 5 at TX:anomaly_score. [file "/etc/modsecurity/rules/REQUEST-949-BLOCKING-EVALUATION.conf"] [line "93"] [id "949110"] [msg "Inbound Anomaly Score Exceeded (Total Score: 8)"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-generic"]
Message: Warning. Operator GE matched 5 at TX:inbound_anomaly_score. [file "/etc/modsecurity/rules/RESPONSE-980-CORRELATION.conf"] [line "91"] [id "980130"] [msg "Inbound Anomaly Score Exceeded (Total Inbound Score: 8 - SQLI=0,XSS=0,RFI=0,LFI=5,RCE=0,PHPI=0,HTTP=0,SESS=0): individual paranoia level scores: 8, 0, 0, 0"] [ver "OWASP_CRS/3.3.0"] [tag "event-correlation"]
Count: 2

Message: Warning. Matched phrase "masscan" at REQUEST_HEADERS:User-Agent. [file "/etc/modsecurity/rules/REQUEST-913-SCANNER-DETECTION.conf"] [line "54"] [id "913100"] [msg "Found User-Agent associated with security scanner"] [data "Matched Data: masscan found within REQUEST_HEADERS:User-Agent: masscan/1.3 (https://github.com/robertdavidgraham/masscan)"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-reputation-scanner"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/118/224/541/310"] [tag "PCI/6.5.10"]

Message: Access denied with code 403 (phase 2). Operator GE matched 5 at TX:anomaly_score. [file "/etc/modsecurity/rules/REQUEST-949-BLOCKING-EVALUATION.conf"] [line "93"] [id "949110"] [msg "Inbound Anomaly Score Exceeded (Total Score: 5)"] [severity "CRITICAL"] [ver "OWASP_CRS/3.3.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-generic"]
Message: Warning. Operator GE matched 5 at TX:inbound_anomaly_score. [file "/etc/modsecurity/rules/RESPONSE-980-CORRELATION.conf"] [line "91"] [id "980130"] [msg "Inbound Anomaly Score Exceeded (Total Inbound Score: 5 - SQLI=0,XSS=0,RFI=0,LFI=0,RCE=0,PHPI=0,HTTP=0,SESS=0): individual paranoia level scores: 5, 0, 0, 0"] [ver "OWASP_CRS/3.3.0"] [tag "event-correlation"]
Count: 2

Message: Warning. Pattern match "^[\\d.:]+$" at REQUEST_HEADERS:HOST. [file "/etc/modsecurity/rules/REQUEST-920-PROTOCOL-ENFORCEMENT.conf"] [line "735"] [id "920350"] [msg "Host header is a numeric IP address"] [data "114.92.187.20"] [severity "WARNING"] [ver "OWASP_CRS/3.3.0"] [tag "application-multi"] [tag "language-multi"] [tag "platform-multi"] [tag "attack-protocol"] [tag "paranoia-level/1"] [tag "OWASP_CRS"] [tag "capec/1000/210/272"] [tag "PCI/6.5.10"]
Count: 1