International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-3S, February 2019

# Designing Security Cheat sheet for Mod Security Firewall tool

**Sonti Likitha, Korvi Raja Sekhar, Pasumarthy Sudeep**

***Abstract:*** ***Web Application Firewall (WAF) is the security tool that acts as a shield for web applications and web application servers from various classes of attacks. WAF acts as a tool/scanner/interface between the server and web applications that provide inclusive protection by validating the constraints (restrictions) specified using 'Sec Rules' which are executed when that particular application is protecting by WAF. Providing protection to applications is one of the key aspects, as WAF can protect against a number of Application Layer Security threats which are usually not protected by numerous typical network layer tools like IDS, IPS & other categories of firewalls. Web applications can be easily attacked by hackers even with the presence of normal firewalls. This is due to the limitation that a normal firewall installed for network layer protection does not work for application layer security issues. Security cheat sheets are very popular for community developed by OWASP. They provide first hand information for developer/designer/analyst/administrator/any other who is part of security system. This paper addresses the cheat sheet for ModSecurity web application firewall tool, which will be helpful for customizing new rules and also helps in designing the documentation part similar to Readme Text file of the tool.***

***Index Terms: Application Security, Web Application, Web-Application Firewall, ModSecurity, Cheat sheet.***

## INTRODUCTION

In this era of globalization, with the proliferation of Internet users, many applications have been designed and deployed to provide more and more services. The web application has become esteem for the organizations. So, the web servers are becoming the predominant target for the cyber-attacks. Almost 70% of web attacks are triumphant because due to inadequate amount of awareness and knowledge on attack defense techniques. There is a need to secure web applications from information leakage and different kinds of cyber attacks. Generally, all web applications use HTTP protocol for client and server communication, it is the path from where attacks come from. So this increases the acceptance of usage of web application firewall.

**Manuscript published on 28 February 2019.**
* Correspondence Author (s)
**Sonti Likitha,** Security Analyst, eSF Labs Ltd., Tadepalli, Guntur District Andhra Pradesh. India (likhitha.sonti@esflabs.com)
**Dr. Korvi RajaSekhar,** Professor, Department of CSE, Koneru Lakshmaiah Education Foundation (Deemed to be University), Vaddeswaram, Guntur District, AP, India (rajasekhar_cse@kluniversity.in)
**Pasumarthy Sudeep,** M.Tech, 2nd year Student, KLEF, Green Fields, Vaddeswaram, Guntur District, AP, India (Sudeep.pasumarthy@gmail.com)



**Firewall:** A firewall is a network security device, protects the system which is used for tracking all the incoming and outgoing network traffic and determines whether to allow or block the request according to the predefined rules.

**Web Application:** Web application is similar to an application program that is deployed on a remote server and can be accessed from anywhere in web browser making use of the internet.

**Web Application Firewall (WAF):** A web application firewall is a tool which is used to detect, monitor and filter all the data packets that are traveling to and fro through a web application [2].

Depending upon the requirement they can be deployed as a Network-based firewall, Host-based firewall, and Cloud-based firewall. The WAF inspects each and every data packet and compares them with the rules and analyzes the packet at layer-7 application logic and filters out potentially harmful traffic. Some of the vulnerabilities that can be mitigated by using this WAF's are SQL Injections, Cross Site Scripting, and Cross-Site Request Forgery.

**Working of Web application Firewall:**

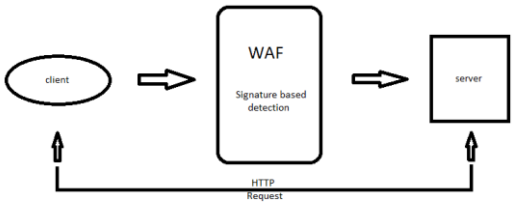


**Figure:1.0 represents the working of the web application firewall.**

When every a request is made by the client in the browser it will communicate with web server in the form of HTTP Request. Attackers generally try to exploit by making use of this communication. To mitigate this organizations use tools like WAF for secure communication. When a request is made, first it passes through the WAF, WAF will compare the request with all its rules, which are already configured and decide whether to forward or block the request. If the request is genuine it allows the request to the server. Once the request reaches the server it will respond back by sending back the requested information. WAF will also look into that response request that is sent by the server. If any sensitive information is passing through it, immediately it will block that response request from server. The figure 1.0 depicts the function of the web application firewall.

Web Application Firewall is generally classified into three types:

**Network-based firewall:** Network-based firewall is generally a hardware-based firewall which is configured near to the application, where every incoming and outgoing traffic passes through it and blocks any malicious or dangerous traffic that has been deducted.

**Host-based firewall:** Host-based firewall is easy to deploy as its code can be directly integrated into the application code itself. This firewall can be customized easily and the implementation includes low cost when compared to network based firewall and cloud based firewall.

**Cloud-based firewall:** Cloud-based firewall can be deployed easily by taking the subscription and changing the DNS. It will just divert the traffic to the firewall and filters the traffic. The most challenging thing here for companies is diverting the traffic through third-party providers.

Here are some of the popular web attacks explained:

**Injection:** SQL, LDAP, and OS injections are the different types of injection flaws that occur when an attacker sends untrusted data through an interpreter as a command or query. If that malicious command or query executes, it will allow the attacker to access that particular data present in that database.

**Cross Site Scripting (XSS):** XSS is a flaw in an application when it takes untrusted data without proper validation through the web browser. This flaw allows attackers to inject some malicious scripts in the victim's browser by which he can hijack user sessions, redirects the users to malicious websites and deface websites.

**Cross-Site Request Forgery (CSRF):** This is an attack which forces users those who were already logged-on. The attacker sends a forged HTTP request, which includes the victim's session cookie and authentication information. Once the victim clicks on that link, the attacker can gain access to that account.

Here we are developing a check sheet that offers tips to battle with most kind of web application attacks.

**Cheat sheet:** A cheat sheet is a documented paper which acts as a reference tool that provides simple, brief instructions for accomplishing a specific task. This is a resource which consists of the technical information about ModSecurity firewall. So, that it helps both the seasoned users and people who are getting started with web application firewall configuration & deployment.

## MOD SECURITY FIREWALL

In order to protect, detect and prevent web applications against web application attacks, we generally prefer a web application firewall. ModSecurity is one of its kind. ModSecurity firewall [3, 9] is one of the tool which is used to stop/limit different types of web application attacks. It is a cross-platform tool which works with three major web server platforms i.e Microsoft IIS, Apache and Nginx.

SecRule is the rule configuration language provided by this platform. It acts as a real-time web application monitoring, logging and access control for every HTTP request and response.

ModSecurity is commonly placed before the application to defense against various types of vulnerabilities using the OWASP ModSecurity Core Rule Set (CRS) [7]. Core Rule Set is an open source ModSecurity rule written in SecRule language which is one of the OWASP projects.

Open Web Application Security Project (OWASP) is an online community where a group of security researches work together to produce methodologies and tools in the field of the web application. It publishes top attacks with the name of OWASP Top 10 [8] which will be updated every four years.

To detect threats, the ModSecurity engine is installed within the web server or as a reverse proxy in front of a web application. This is the reason for which it can be able to monitor all the incoming and outgoing HTTP requests. According to the rule, the rule configuration engine will decide how the communication should be managed which may include the ability to pass, drop, redirect, execute the script and more.

As the web application was configured with OWASP ModSecurity CRS V3 [11] running in reverse proxy mode, the attacker directly can't access the server whereas every request he made will redirect to the firewall first.

The main aim of CRS is to protect web applications from wide range of attacks, which includes OWASP TOP Ten. This provides protection against many common attacks.

## SECRULES SPECIFICATION

For reading every request and analyzing it whether to allow or deny the request we need a rule. ModSecurity also has a rule language for writing a rule which is called "SecRule". For writing a new SecRule we need to follow the syntax. The syntax is "SecRule VARIABLES OPERATOR [TRANSFORMATION FUNCTIONS, ACTIONS]". By using the above syntax we can develop our own SecRule for ModSecurity.

**SecRule:** SecRule means it indicates that this is a new rule.

**Variable:** Variable is used to identify every part of the HTTP transaction that deals with the rule. It will abstract every transaction and makes it accessible through Variables. The key thing about the variable is it contains special characters of any byte value. The main characteristic of ModSecurity is to pre-process the transaction data and makes it easy for the rules to access on the logic of perception. There are nearly 77 variables in ModSecurity.

**Operator:** Operator is used for analyzing the variable. Here we generally go with regular expressions; however there are many other operators that can also be used. Operator always starts with @ character. It is always placed at the starting of the second SecRule token. Some of the string matching operators that are commonly used are @rx, @pm, @endswith, @within, @contains. @beginswith.

**Transformation Function:** A transformation function is specified for every rule which instructs ModSecurity to check how a variable will change before the analysis process is done.


*Published By:*
*Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP)*

**Action:** Action specifies what to do if the rule matches. This is the trickiest part of ModSecurity. They make it possible to react to the events, and this is what the actual thing which makes the advanced features possible. There are seven categories in the action field. They are allow, block, deny, drop, pass, proxy and redirect.

**Regular Expression:** Regular expression is the pattern matching method used in programming. These provide an adaptable and brief means to peer strings to text. Regular expressions are used for data validation for search engines, syntax emphasizing the systems.

OWASP and Comodo [15] are two organizations that are providing free rule sets which can be used by ModSecurity.

There are few things that need to know before installing and configuring the ModSecurity firewall. There are lot of options to build a rule, but few are mandatory to follow. They are like explained below.

- Every rule should contain a VARIABLE, an OPERATOR and ACTION.
- If no OPERATOR is taken then @rx works as the default operator.
- For ACTION, action id is required. However SecDefaultAction consists of several actions like phase, log, auditlog, pass. Phase:2 acts as a default phase, when no phase is included.
- Every rule will have a disruptive ACTION. This tells the rule what to do when the transaction is initiated. If there is no disruptive action then pass is the default action.
- Transformations are optional but these are used to prevent the rule from being bypassed.

## RESULTS & DISCUSSIONS

Generally a security cheat sheet must contain important elements: problem statement, different mechanism/solutions for the problem (s), description of the solution(s) in terms of steps required for each one described. The main motive behind designing the cheat sheet is to give a detailed understanding of the tool for any naive person.

This paper talks about the cheat sheet for ModSecurity firewall tool where it will brief all the key information that is probably to be on a test. We are dividing the sheet into three important components Wiz: Component 1 - Basic information about the tool, Component 2 - Syntax of Security rules including customization against new rules to be defined for other attacks which are not available in the ModSecurity database and the third one is Component 3- Example scenario in terms of sample rules.

Component 1 is described in Table - I which will give all the information of the key terminology. This table is divided into two parts. The 1st part deals with the basic terms and the 2nd part describes key terminologies so that any person with non-technical knowledge can also understand it easily. This table explains what ModSecurity is? What are the general attacks that are performed on layer-7? How this can be installed and configured? How rules work? How rules and the rule syntax can be defined?

**TABLE - I**

| Basic Terms | Description |
|---|---|
| ModSecurity | ModSecurity is an open-source cross-platform web application firewall. |
| Layer-7 Attacks | SQL Injection, Cross-site-scripting, Brute Force, Path Traversal, Cross-site request forgery, Denial of service. |
| ModSecurity Installation & Configuration | Windows: Can be installed using IIS MSI installer.<br>Linux: Can install and configure the ModSecurity through a command line. |
| RuleSet | ModSecurity makes use of OWASP CRS (Core Rule Set) as it is generic attack detection. |
| Top Rules | Application Attacks: XSS, SQLI, LFI, RFI, RCE, File-Detection, DOS-Protection,<br>Exclusion rules: Wordpress, CPANEL.<br>Data leakages: SQL, JAVA, PHP, IIS. |
| Rules Working | Firewall compares each and every request with the rules and determines if it should be allowed or denied. |
| Customize New Rule | SecRule Variables Operators [Transformation function, Action] |
| Definitions | SecRule: Defines that this is a new rule.<br>Variables: Interact with each HTTP transaction and make it available for the rules.<br>Operators: Specify how a variable should be analyzed<br>Transformation Function: This specifies how a variable should be changed for the analysis.<br>Action: Defines what to be done if the rule matches. |

Component 2 shown in Table - II describes all the other alternatives to create our own rule. This table only concentrates on the rule customization. This explains Directives, Variables, Operators and Actions. Whenever we want to create a new rule these four terms play a key role. Whenever we want to create a rule we will pick one from Directives, can select two or more from Variables, an Operator and an Action. The combination of all these can make a successful SecRule.

Table – II plays a crucial role in the development of a new SecRule. Successful rule making can only be done if we know the problem statement. So that we can pick up the right components and can place them at right place to stop that particular attack.


Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

**TABLE - II**

Rule Customization

Rule Syntax: *SecRule Variables Operators [Transformation function, Action]*

Directives: *SecAction*
*SecDefaultAction*
*SecMarker*
*SecRule*
*SecRuleInheritance*
*SecRuleUpdateActionById*
*SecRuleRemoveById*
*SecRuleRemoveByMsg*
*SecRuleScript*

Rule language is executed by using directives. Here we have 9 directives, by using these directives we can make rules. SecRule is considered as the main directive where we can create a new rule. Every Directive has its own importance.

Variables: *ARGS*
*ARGS_NAMES*
*ARGS_GET*
*ARGS_POST*
*REMOTE_USER*
*REQUEST_BODY*
*REQUEST_COOKIES*
*REQUEST_FILENAME*
*REQUEST_HEADER*
*REQUEST_METHOD*
*REQUEST_PROTOCOL*
*REQUEST_URI*
*RESPONSE_BODY*
*RESPONSE_HEADER*
*RESPONSE_HEADER_NAMES*
*RESPONSE_PROTOCOL*
*RESPONSE_STATUS*
*RESPONSE_CONTENT_LENGTH*

There are about 77 variables in the recent version of ModSecurity, here are some of the mostly used variables. These variables are used to identify the HTTP transactions that are passing through the request. ModSecurity will extract all the information from the request and make it available for the rule in the form of variables. So that rules will work on it.

Operators: *@rx*
*@pm*
*@eq*
*@ge*
*@gt*
*@le*
*@lt*
*@contains*
*@endswith*
*@beginswith*

Operators work with the transformed variable, how it should be analyzed. Regular Expressions are commonly used operators. ModSecurity supports with other operators too. All Operators starts with '@' character. @rx and @pm are most commonly used Operators, because of their versatility (@rx) and speed (@pm).

Actions: *Allow*
*deny*
*block*
*drop*
*Pass*
*proxy*
*status*
*pause*
*skip*
*chain*
*msg*
*id*
*rev*
*severity*
*Log*

Action defines what it should be done when the rule matches.

Component 3 described in Table - III shows us the example and working rule. The example rule and working rule both are given on cross-site scripting attack. Example rule is explained briefly so that a normal user can understand it easily.

The below Figure 1.1 represents one of the working rule used to protect the web applications from cross-site scripting attack. Here in this particular example we have shown you how ModSecurity Firewall is configured with OWASP Core Rule SET (CRS). Whenever ModSecurity finds a request with a script tag, this rule will be activated.

**TABLE - III**

**Example rule with explanation:**
SecRule ARGS|REQUEST_HEADERS "@rx (?i)(<script[^>]*>[\s\S]*?<\/script[^>]*>" id:201, msg: 'XSS Attack', severity:critical, deny, status:404

**DIRECTIVE:**
SecRule – Make a new rule.

**VARIABLES:**
ARGS - Request Parameters.
REQUEST_HEADERS - All of the request headers.

**OPERATOR:**
"@rx (?i)(<script[^>]*>[\s\S]*?<\/script[^>]*>" - Performs a regular expression match pattern provided as parameter.

**ACTION:**
id, msg, severity, deny, status - These all are the actions that will be performed if the pattern matches.

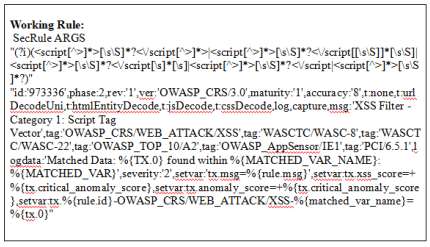
**Working Rule:**
SecRule ARGS
"(?i)(<script[^>]*>[\s\S]*?<\/script[^>]*>|<script[^>]*>[\s\S]*?<\/script[[\s\S]]*[\s\S]|<script[^>]*>[\s\S]*?<\/script[\s]*[\s]|<script[^>]*>[\s\S]*?<\/script\<script[^>]*>[\s\S]*?)"
"id:'973336',phase:2,rev:'1',ver:'OWASP_CRS/3.0',maturity:'1',accuracy:'8',t:none,t:urlDecodeUni,t:htmlEntityDecode,t:jsDecode,t:cssDecode,log,capture,msg:'XSS Filter - Category 1: Script Tag Vector',tag:'OWASP_CRS/WEB_ATTACK/XSS',tag:'WASCTC/WASC-8',tag:'WASCTC/WASC-22',tag:'OWASP_TOP_10/A2',tag:'OWASP_AppSensor/IE1',tag:'PCI/6.5.1',logdata:'Matched Data: %{TX.0} found within %{MATCHED_VAR_NAME}: %{MATCHED_VAR}',severity:'2',setvar:tx.msg=%{rule.msg}',setvar:tx.xss_score=+%{tx.critical_anomaly_score},setvar:tx.anomaly_score=+%{tx.critical_anomaly_score},setvar:tx.%{rule.id}-OWASP_CRS/WEB_ATTACK/XSS-%{matched_var_name}=%{tx.0}"

**Figure:1.1 Represent one of the working rules.**


Published By:
Blue Eyes Intelligence Engineering
and Sciences Publication (BEIESP)

## ACKNOWLEDGEMENT

This work is supported by the eSF Labs Ltd as part of the project/consultancy work taken by Department of Computer Science and Engineering of KLEF with I am very thankful to all my colleagues at eSF Labs Ltd. for their continuous support and helping me in completing the project.

## CONCLUSION

The mentioned cheat sheet provides a complete awareness of ModSecurity web application firewall (WAF) tool. We have provided all the requirements for configuring, installing and deploying ModSecurity WAF. This paper covers the importance of web application firewall, some of the web application attacks including cross site scripting, SQL injection, Brute force, DoS and File Detection. The cheat sheet will give the sequence of steps required to execute any attack and corresponding mitigation is shown with Sec Rule. With the help of this cheat sheet, a normal user can be able to configure and write new rules on his/her own depending up on the usage.

## REFERENCES

1. Jatesh Jagraj Singh, Hamman Samuel, Pavol Zavarsky, “Impact of Paranoia Levels on the Effectiveness of the ModSecurity Web Application Firewall,” IEEE Xplore: 28 May 2018.
2. Asrul H. Yaacob, Nazrul M. Ahmad, Nurul N. Ahmad, Mardeni Roslee, “Moving towards Positive Security Model for Web Application Firewall ” International Journal of Computer and Information Engineering, Vol:6, No:12, 2012.
3. “what ModSecurity can do?” ModSecurity.
4. Online “ModSecurity Rule Writing” KEMP Technologies.
5. Ryan C. Barnett, Shreeraj Shah “Securing Web Services with ModSecurity” 9 June 2005, Oreilly.
6. Ivan Ristic “ModSecurity Handbook” 30 jan 2010.
7. Online “OWASP Core Rule Set Project” OWASP.
8. “OWASP TOP10 2017” OPEN WEB APPLICATION SECURITY PROJECT.
9. “ModSecurity WAF” SpiderLabs/ModSecurit, GitHub.
10. Online “Defense against web application attacks” OWASP
11. Online “New Features in CRS 3” ModSecurity.
12. Ryan Barnett, Greg Wroblewski “ModSecurity as Universal Cross-platform Web Protection Tool” media.blackhat.
13. Online “OWASP Cheat Sheet” OWASP_Cheat_Sheet_Series.
14. Sam Hobbs “Example Whitelisting Rules for Apache ModSecurity and the OWASP Core Rule Set” 22 sep 2015.
15. Online “comodo web application firewall” waf.comodo.
16. Alexander Endraca, Bryan King, George Nodalo, Maricone Sta. Maria, and Isaac Sabas “Web Application firewall (WAF)” ijeeee, vol. 3, 6 Dec 2013.