Cyber Security

ModSecurity

Open Source Web Application Firewall

With the increasing threats and attacks on web applications, organizations require a more effective concept of web application security. Web Application Firewall (WAF) is such a concept that can be used to prevent various threats and attacks on web applications. WAF has the ability to filter packets, block malicious HTTP requests, and also do logging. The open-source WAFs are highly flexible and customizable. With full access to the source code, Open source WAF offers the freedom to WAF administrators, web administrators and developers to apply rules as per individual application and provides flexibility to customize and extend the tool itself to fit as per application requirements. ModSecurity is a popular open source Web Application Firewall.

ModSecurity is an open source, cross platform web application firewall (WAF) engine for Apache, IIS and Nginx. It provides protection from a range of attacks against web applications such as Cross Site Scripting (XSS), SQL Injection, Cross Site Request Forgery, Local File Inclusion, Path Traversal, Session Fixation, etc. and allows for HTTP traffic monitoring, logging and real-time analysis. ModSecurity excels at virtual patching contributed by its reliable blocking capabilities and the flexible rule language that can be adapted to any need.

Ratnaboli Ghorai Dinda
Scientist-G & HOG (Application Security)
ratnaboli@gov.in

R. K. Raina
Sr. Technical Director
rk.raina@nic.in

Rajeev Kumar Yadav
Scientist-B
yadav.rajeev@nic.in

ModSecurity gives access to the HTTP traffic stream in real time, along with the ability to inspect it. It can be deployed in embedded mode or in reverse proxy mode. ModSecurity excels at virtual patching because of its reliable blocking capabilities and the flexible rule language that can be adapted to any need. ModSecurity works with OWASP ModSecurity Core Rule Set (CRS), CRS is a set of generic attack detection rules for use with ModSecurity or compatible web application firewalls. The CRS aims to protect web applications from a wide range of attacks, including the OWASP Top Ten, with a minimum of false alerts. ModSecurity along with CRS provides protection against many common attack categories, including SQL Injection, Cross Site Scripting, Cross Site Request Forgery, Local File Inclusion, Open Redirect, Insufficient Session Expiration, Path Traversal, etc.

Features/Functionalities of ModSecurity

ModSecurity employs a variety of methods to protect websites. Following is a list of the most important usage scenarios for ModSecurity:

Real-time application security monitoring and access control

At its core, ModSecurity gives us access to the HTTP traffic stream in real time, along with the ability to inspect it. This is enough for real-time security monitoring. ModSecurity's persistent storage mechanism enables users to track system elements over time and perform event correlation. Users can block reliably, if they so wish, because ModSecurity uses full request and response buffering.

Virtual patching

Virtual patching is a concept that addresses vulnerability mitigation in a separate layer, in which you get to fix problems in applications without having to touch the applications themselves. Virtual patching is the quick development and short-term implementation of a security policy meant to prevent an exploit from occurring. The resulting impact of virtual patch is that, while the actual source code of the application itself has not been modified, the exploitation attempt does not succeed. ModSecurity excels at virtual patching because of its reliable blocking capabilities and the flexible rule language that can be adapted to any need. Virtual patching is, by far, the activity ModSecurity offers that requires the least investment, is the easiest to perform, and that most organizations can benefit from straight away.

Full HTTP traffic logging

Web servers traditionally do very little when it comes to logging for security purposes. They log very little by default, and even with a lot of tweaking we can't get all the data that we need. ModSecurity gives us the ability to log everything, including raw transaction data, which is essential for forensics. In addition, we get to choose which transactions are logged, which parts of a transaction are logged, and which parts are sanitized. As a bonus, this type of detailed logging is also helpful for application troubleshooting—not just security.

Web application hardening

One of important uses for ModSecurity is attack surface reduction, in which we can selectively narrow down the HTTP features we're willing to accept (e.g., request methods, request headers, content types, etc.). ModSecurity can assist users in enforcing many similar restrictions, either directly or through collaboration with other web server modules. For example, it's possible to fix many session management issues, as well as cross site request forgery vulnerabilities.

Deployment Options

ModSecurity supports two deployment options: embedded and reverse proxy deployment. Users can pick the most appropriate option based on their goals, requirements, and situation. There are advantages and disadvantages of both options:

Embedded

The embedded option is a great choice for those who already have their architecture laid out and don't want to change it. Embedded deployment is also the preferred option if we need to protect hundreds of web servers. In such situations, it is impractical to build a separate proxy-based security layer. Embedded ModSecurity not only does not introduce new points of failure, but also it scales seamlessly as the underlying web infrastructure scales. The main challenge of embedded deployment is that server resources are shared between the web server and ModSecurity.

Reverse proxy

Reverse proxies are effectively HTTP routers, designed to stand between web servers and their clients. When we install a dedicated reverse proxy web server and add ModSecurity to it, we get a "proper" network web application firewall, which we can use to protect any number of web servers on the same network. This mode gives us complete isolation from the systems (e.g. web servers/applications and databases) we are protecting. On the performance front, a standalone ModSecurity installation will have resources dedicated to it, which means that we will be able to do more (i.e., have more complex rules). The main disadvantage of this approach is the new point of failure, which will need to be addressed with a high-availability setup of two or more reverse proxies.

Transaction Lifecycle

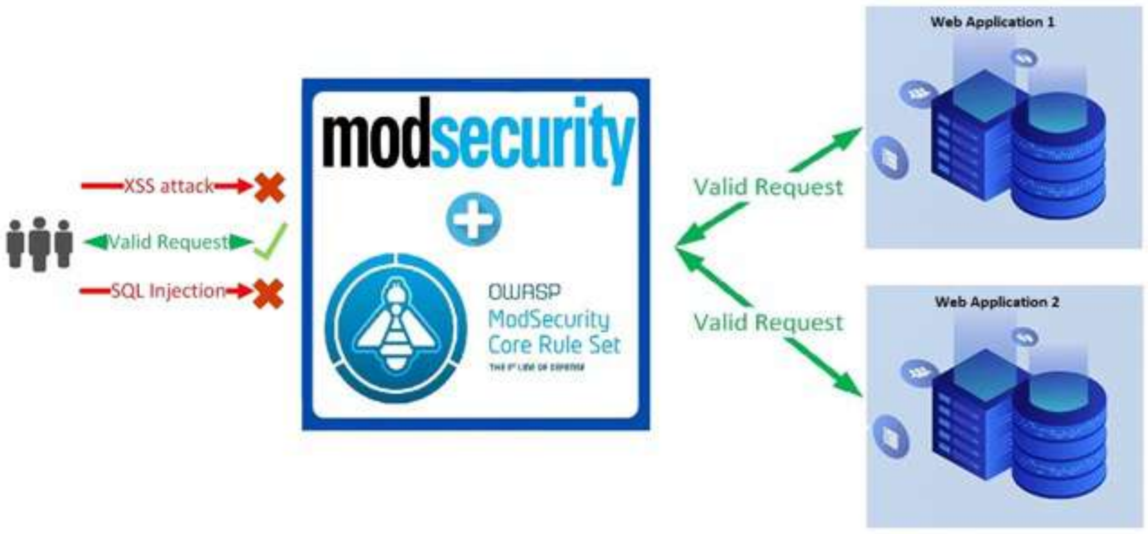
In ModSecurity, every transaction goes through five steps, or phases. In each of the phases, ModSecurity will do some work at the beginning (e.g., parse data that has become available), invoke the rules specified to work in that phase, and perhaps do a thing or two after the phase rules have finished.

Request headers

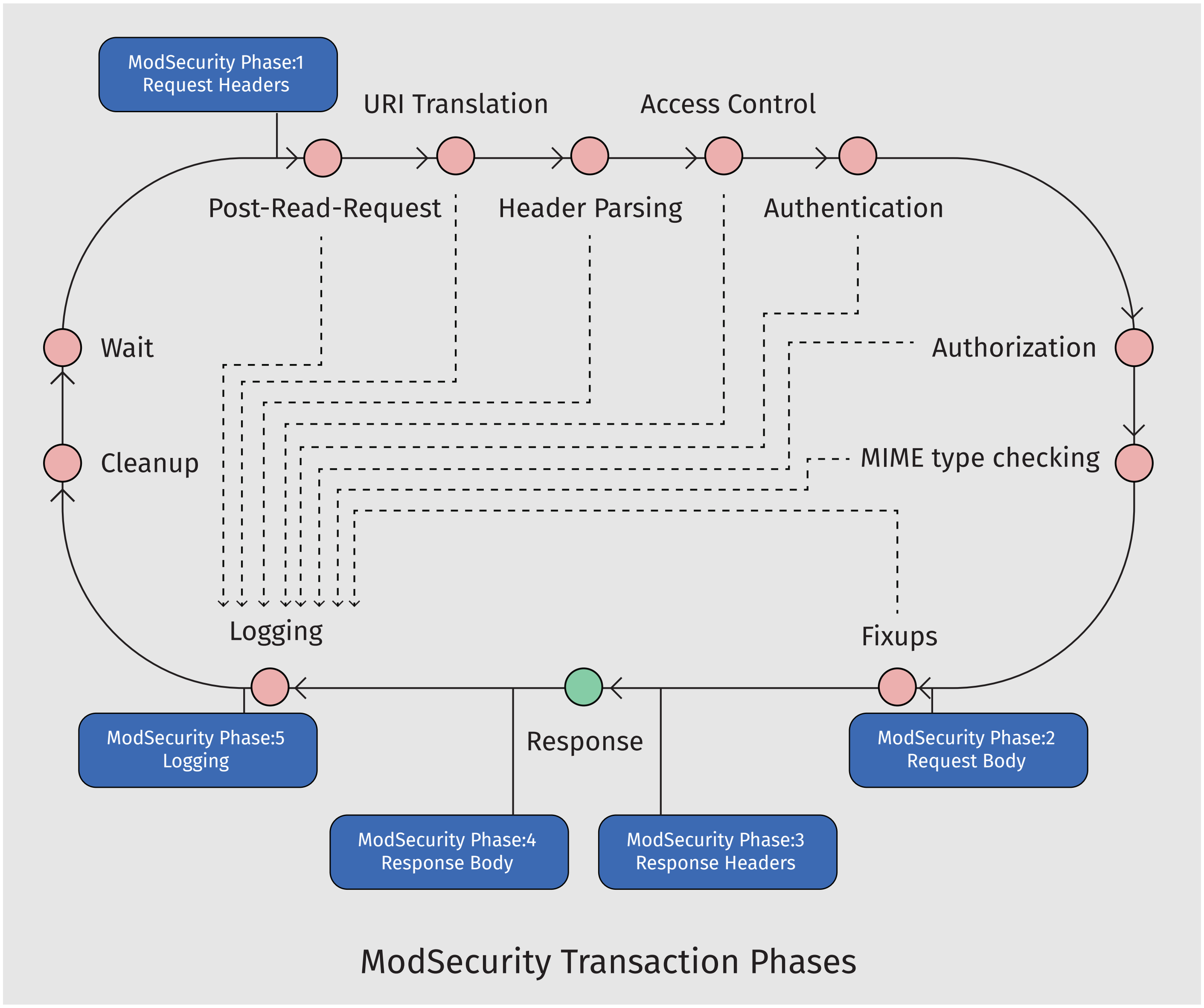
The request headers phase is the first entry point for ModSecurity. The principal purpose of this phase is to allow rule writers to assess a request before the costly request body processing is undertaken. Similarly, there is often a need to influence how ModSecurity will process a request body, and this phase is the place to do it. For example, ModSecurity will not parse an XML request body by default, but we can instruct it do so by placing the appropriate rules into phase 1.

Request body

The request body phase is the main request



▲ ModSecurity


ModSecurity Phase:1 Request Headers
URI Translation
Access Control
Post-Read-Request
Header Parsing
Authentication
Wait
Authorization
Cleanup
MIME type checking
Logging
Fixups
ModSecurity Phase:5 Logging
Response
ModSecurity Phase:2 Request Body
ModSecurity Phase:4 Response Body
ModSecurity Phase:3 Response Headers
ModSecurity Transaction Phases

analysis phase and takes place immediately after a complete request body has been received and processed. The rules in this phase have all the available request data at their disposal.

Response headers

The response headers phase takes place after response headers become available, but before a response body is read. The rules that need to decide whether to inspect a response body run in this phase.

Response body

The response body phase is the main response analysis phase. By the time this phase begins, the response body will have been read, with all its data available for the rules to make their decisions.

Logging

The logging phase is special in more ways than one. First, it's the only phase from which we cannot block. By the time this phase runs, the transaction will have finished, so there's little we can do but record the fact that it happened. Rules in this phase are run to control how logging is done.

Conclusion

ModSecurity is a very powerful and flexible WAF. It prevents web applications against a number of attacks such as SQL Injection, Cross Site Scripting, Cross Site Request Forgery, Local File Inclusion, Missing HTTP Only and Secure Flags on Sensitive Cookies, Improper Access Control, Sensitive Data Exposure and many more. Web application administrators can use ModSecurity as a defense against such web application vulnerability exploits. It gives us freedom to decide how to take advantage of the features available in it. This flexibility is a core element of ModSecurity's identity, and complements its open source structure. In fact, users can enjoy complete access to its source code, which empowers them to customize the tool to suit their unique needs.

For further information, please contact:
R. K. Raina
Sr. Technical Director
National Informatics Centre, A-Block
CGO Complex, Lodhi Road
New Delhi - 110003
Email: rk.raina@nic.in, Phone: 011-23405231