# ModSecurity
# OWASP CRS 3.0

Open Source Web Application Firewall

# Introduction

ModSecurity is an Apache Module providing a Web Application Firewall by enforcing rules against requests and responses.

It has since been extended to work on other web servers such as Nginx.

The Core Rule Set from OWASP is what makes ModSecurity useful.

# Useless By Itself

When you install Mod Security on your web server, it does nothing until you start writing rules.

The Core Rule Set is a popular starting point providing a large range of general rules.

# OWASP

OWASP is an open community dedicated to enabling organizations to conceive, develop, acquire, operate, and maintain applications that can be trusted. All of the OWASP tools, documents, forums, and chapters are free and open to anyone interested in improving application security.

# Version 3.0

The current version is a major improvement.

The rules are better organized.

There are far fewer false positives.

There are custom exception sets for the two most popular CMS programs.

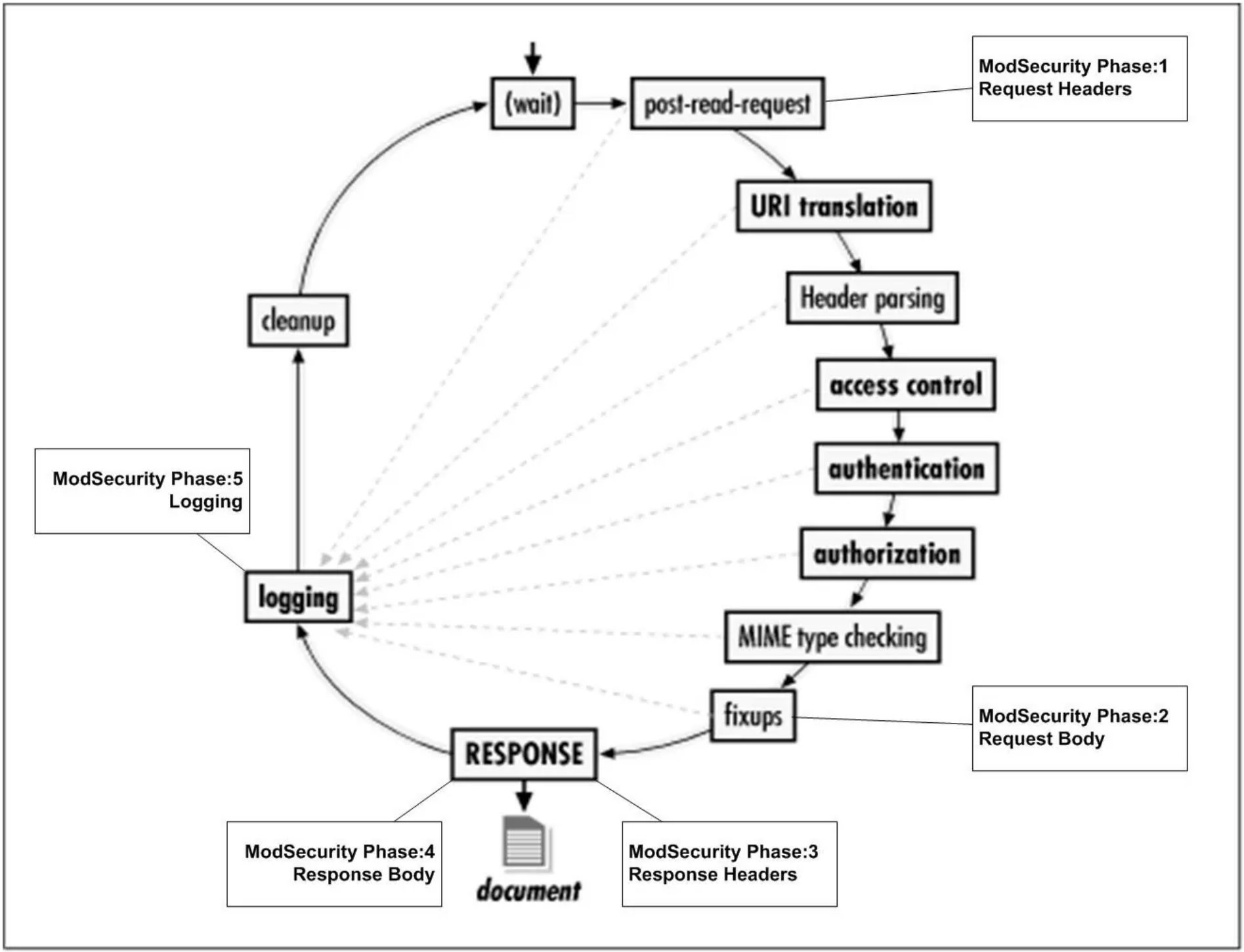
# Installing ModSecurity

- apt-get install libapache2-mod-security2
- cp modsecurity.conf-recommended 000-modsecurity.conf
- Add **SecRuleEngine On** to your virtual hosts
- Restart Apache

# Processing a Request

ModSecurity has 5 Phases

1) Request Header and Protocol Analysis
2) Body Analysis
3) Response Header Analysis
4) Response Body Analysis
5) Logging


(wait)
post-read-request
ModSecurity Phase:1
Request Headers
URI translation
Header parsing
access control
authentication
authorization
MIME type checking
fixups
ModSecurity Phase:2
Request Body
RESPONSE
document
ModSecurity Phase:3
Response Headers
ModSecurity Phase:4
Response Body
logging
ModSecurity Phase:5
Logging
cleanup

# The Phases

The preceding shows ModSecurity in relation to the apache request cycle.

**Phase I** has access to the headers this is where protocol enforcement rules occur, notice it is before URI translation which means it will also operate before rewrite.

For example the CRS sets up a list of allowed Request Types, disallowing for example DELETE by default.

# The Phases

**Phase II** has access to the body before the response is generated, this is where the data being submitted by the client can be looked at.

**Phase I** and **Phase II** are where most of the ModSec activity occurs, because they are where incoming data is coming from.

# The Phases

**Phase III** and **Phase IV** analyze the Response Header and Body respectively. If you're proxying an IIS server for example, the CRS has rules to prevent code leakage.

**Phase V** logs. There are a lot of logging options for ModSec. Enabling the Debug log will allow you to capture request data that cannot otherwise be captured.

# Performance

- ModSecurity can be configured to log when a phase or a rule takes too much time.
- Frequently the problem is scanning large form fields with complex regular expressions.
- The long running rules can be tuned or disabled

# An Example of a Rule

```
SecRule REQUEST_URI "(env\.cgi|info\.php)"
"Phase:1,\
  Id:990101,\
   tag:'prevent access to env.cgi and
info.php',\
   logdata:'attempt to access env or info
page',\
   log,block"
```

# Example Rule

**`SecRule REQUEST_URI "(env\.cgi|info\.php)"`**
**SecRule** is the keyword for defining a rule,
**REQUEST_URI** defines what the rule will look at and the quoted text is a regular expression to be run against it.

The rest of the rule is all enclosed in double quotes with line breaks using the \ terminator. If additional quoting is needed single quotes are used internally.

# Example Rule

```
"
   Phase:1,\
   Id:990101,\
   tag:'prevent access to env.cgi and
info.php',\
   logdata:'attempt to access env or info
page',\
   Log,\
   block\
"
```

# Core Rule Set

SQL Injection (SQLi)
Cross Site Scripting (XSS)
Local File Inclusion (LFI)
Remote File Inclusion (RFI)
Remote Code Execution (RCE)
PHP Code Injection
HTTP Protocol Violations
Session Fixation
Scanner Detection
Metadata/Error Leakages

# Organization

For upgraders the organization of the 3.0 ruleset is much better than the previous version.

The rules are grouped into files by attack types and also by vector so there are two files that are PHP specific, if you're not using PHP then you know you don't need them, whereas in the old ruleset you had to find and disable all of the PHP rules if you didn't want to waste time processing them on every request.

# Installing the CRS

Whereas ModSecurity is pretty stable and easier to install from repositories than to compile your own apache module. The CRS is likely to be out of date in your distribution and very easy to pull in from source.

```
cd /opt
git clone
    https://github.com/SpiderLabs/
    owasp-modsecurity-crs.git
```

# Modes

The CRS has two modes: Anomaly Scoring and Self-Contained. The default is Anomaly but beginners and simpler installations will prefer Self-Contained.

**Anomaly** assigns a score to every rule that triggers and decides based on the total score of the request whether to take an action.

**Self-Contained** takes an action whenever a rule triggers, usually block.

Copy the CRS-setup example and include it in your config to load immediately after modsecurity.conf.

Comment the default SecDefaultAction lines and replace with (each is a single line)

```
SecDefaultAction
 "phase:1,log,auditlog,deny,
  status:406"
SecDefaultAction
  "phase:2,log,auditlog,deny,
  status:406"
```

We've  told mod security that the default action for any rule is to deny.

We've also set a distinct status code and chosen an appropriate one not likely to be generated normally.

Popular Codes to use are
405 and 406 (Method Not Allowed and Not Acceptable)
Or to use a code that could be 'normal'
400 (Bad Request)  and 403 (Forbidden)

# Paranoia Level

The Paranoia Level is a global setting of 0 to 4.

The higher the Paranoia Level more and stricter rules are run.

Costs of Paranoia

- Longer Processing time for more rules
- More False Positives
- More time tuning to mitigate false positives.

# Links and References

- Mod Security HandBook
- https://modsecurity.org/
- https://modsecurity.org/crs/
- https://www.owasp.org/index.php/Category:OWASP_ModSecurity_Core_Rule_Set_Project