|  |
| --- |
| ModSecurity for NGINX  **Ikomet** |
| Following steps to install ModeSecurity in Ngnix. Currently we are using nginx version **1.18.0** |

# 1. Install Prerequisite Packages

|  |
| --- |
| InstallThe first step is to install the packages required to complete the remaining steps in this tutorial. Run the following command, which is appropriate for a freshly installed Ubuntu/Debian system. The required packages might be different for RHEL/CentOS/Oracle Linux. **$ apt-get install -y apt-utils autoconf automake build-essential git libcurl4-openssl-dev libgeoip-dev liblmdb-dev libpcre++-dev libtool libxml2-dev libyajl-dev pkgconf wget zlib1g-dev** |
|  |

# 2. Download and Compile the ModSecurity 3.0 Source Code

|  |
| --- |
| With the required prerequisite packages installed, the next step is to compile ModSecurity as an NGINX dynamic module. In ModSecurity 3.0’s new modular architecture, libmodsecurity is the core component which includes all rules and functionality. The second main component in the architecture is a connector that links libmodsecurity to the web server it is running with. There are separate connectors for NGINX, Apache HTTP Server, and IIS. We cover the NGINX connector in the next section.To compile libmodsecurity:Clone the GitHub repository: **$ git clone --depth 1 -b v3/master --single-branch https://github.com/SpiderLabs/ModSecurity** |
| Change to the ModSecurity directory and compile the source code: **$ cd ModSecurity**  **$ git submodule init**  **$ git submodule update**  **$ ./build.sh**  **$ ./configure**  **$ make**  **$ make install**  **$ cd ..**  The compilation takes about 15 minutes, depending on the processing power of your system.  Note: It’s safe to ignore messages like the following during the build process. Even when they appear, the compilation completes and creates a working object. |

# 3. Download the NGINX Connector for ModSecurity and Compile It as a Dynamic Module

|  |
| --- |
| Compile the ModSecurity connector for NGINX as a dynamic module for NGINX.  1. **Clone the GitHub repository:**   **$ git clone --depth 1** <https://github.com/SpiderLabs/ModSecurity-nginx.git>  **2. Determine which version of NGINX is running on the host where the ModSecurity module will be loaded**  **$ nginx -v**  **nginx version: nginx/1.18.0**    **3. Download the source code**  **$ wget http://nginx.org/download/nginx-1.18.0.tar.gz**  **$ tar zxvf nginx-1.13.1.tar.gz**    **4. Compile the dynamic module and copy it to the standard directory for modules:**  **$ cd nginx-1.18.0**  **$ export MODSECURITY\_INC="/home/iKomet/ModSecurity/headers/"**  **$ export MODSECURITY\_LIB="/home/iKomet/ModSecurity/src/.libs/"**  **$ ./configure --with-compat --add-dynamic-module=../ModSecurity-nginx**  **$ make modules**  **$ cp objs/ngx\_http\_modsecurity\_module.so /etc/nginx/modules**  **$ cd ..** |

# 4. Load the NGINX ModSecurity Connector Dynamic Module

|  |
| --- |
| Compile the ModSecurity connector for NGINX as a dynamic module for NGINX.  1. **Clone the GitHub repository:**   Add the following [load\_module](https://nginx.org/en/docs/ngx_core_module.html#load_module) directive to the main (top‑level) context in **/etc/nginx/nginx.conf.** It instructs NGINX to load the ModSecurity dynamic module when it processes the configuration:  **load\_module modules/ngx\_http\_modsecurity\_module.so;** |

# 5.Configure, Enable, and Test ModSecurity

|  |
| --- |
| **The final step is to enable and test ModSecurity.**  **Step : 1**  Set up the appropriate ModSecurity configuration file. Here we’re using the recommended ModSecurity configuration provided by TrustWave Spiderlabs, the corporate sponsors of ModSecurity.:  **$ mkdir /etc/nginx/modsec**  **$ wget -P /etc/nginx/modsec/ https://raw.githubusercontent.com/SpiderLabs/ModSecurity/v3/master/modsecurity.conf-recommended**  **$ mv /etc/nginx/modsec/modsecurity.conf-recommended /etc/nginx/modsec/modsecurity.conf**  **Step : 2**  To guarantee that ModSecurity can find the unicode.mapping file (distributed in the top‑level ModSecurity directory of the GitHub repo), copy it to /etc/nginx/modsec.  **$ cp ModSecurity/unicode.mapping /etc/nginx/modsec**  **Step : 3**  Change the SecRuleEngine directive in the configuration to change from the default “detection only” mode to actively dropping malicious traffic.  **$ sed -i 's/SecRuleEngine DetectionOnly/SecRuleEngine On/' /etc/nginx/modsec/modsecurity.conf**  **Step : 4**  Configure one or more rules. For the purposes of this blog we’re creating a single simple rule that drops a request in which the URL argument called testparam includes the string test in its value. Put the following text in /etc/nginx/modsec/main.conf:  **# From https://github.com/SpiderLabs/ModSecurity/blob/master/**  **# modsecurity.conf-recommended**  **#**  **# Edit to set SecRuleEngine On**  **Include "/etc/nginx/modsec/modsecurity.conf"**  **# Basic test rule**  **SecRule ARGS:testparam "@contains test" "id:1234,deny,status:403"**  In a production environment, you presumably would use rules that actually protect against malicious traffic, such as the free [OWASP core rule set](https://docs.nginx.com/nginx-waf/admin-guide/nginx-plus-modsecurity-waf-owasp-crs/).  **Step : 5**  Add the modsecurity and modsecurity\_rules\_file directives to the NGINX configuration to enable ModSecurity: |
| **server {**  **# ...**  **modsecurity on;**  **modsecurity\_rules\_file /etc/nginx/modsec/main.conf;**  **}**    **Step : 6**  Issue the following curl command. The 403 status code confirms that the rule is working.  **$ curl localhost?testparam=test**  **<html>**  **<head><title>403 Forbidden</title></head>**  **<body bgcolor="white">**  **<center><h1>403 Forbidden</h1></center>**  **<hr><center>nginx/1.13.1</center>**  **</body>**  **</html>** |