Building High-Security Apache with ModSecurity 3 & OWASP CRS

Target System: Ubuntu/Debian Linux **Software Versions:** Latest Apache2, ModSecurity v3 (Libmodsecurity), ModSecurity-Apache Connector, OWASP Core Rule Set (CRS).

Part 1: System Preparation & Apache Installation

First, we ensure the system is up to date and add a third-party repository (PPA) to get the absolute latest version of the Apache server, rather than the older version often found in default repositories.

Step 1.1: Update System & Add Repositories

```
# Update package lists and upgrade existing software
sudo apt update && sudo apt full-upgrade -y

# Add the repository for the latest stable Apache2 version
sudo apt-add-repository ppa:ondrej/apache2 -y

# Update the package list again to include the new repository
sudo apt update
```

Step 1.2: Install Apache

```
# Install Apache2 and the Apache development headers (required later for
compiling)
sudo apt install -y apache2 apache2-dev
```

Part 2: Compiling ModSecurity v3 (The Core Engine)

ModSecurity v3 is split into two parts: the **Library (Libmodsecurity)** and the **Connector**. We must build the library first. This acts as the "brain" that processes the security rules.

Step 2.1: Install Build Dependencies We install the compilers (`g++`, `make`) and libraries (`libcurl`, `libxml`, `pcre`) required to build ModSecurity from source code.

```
# Install git, build tools, and specific libraries for ModSecurity v3
sudo apt-get install -y git g++ apt-utils autoconf automake build-essential \
libcurl4-openssl-dev libgeoip-dev liblmdb-dev libpcre2-dev \
libtool libxml2-dev libyajl-dev pkgconf zlib1g-dev
```

Step 2.2: Clone and Compile ModSecurity

```
# Clone the repository including submodules (--recursive is critical)
git clone --recursive https://github.com/owasp-modsecurity/ModSecurity
ModSecurity

# Enter the directory
cd ModSecurity

# Generate the build scripts
./build.sh

# Configure the build environment
./configure

# Compile the source code (this may take a few minutes)
make

# Install the compiled library to your system
sudo make install

# Return to home directory
cd ..
```

Part 3: The Apache Connector

Now that the core engine is installed, we need the "bridge" that connects Apache to ModSecurity.

Step 3.1: Clone and Compile the Connector

```
# Clone the connector specifically for Apache
git clone https://github.com/owasp-modsecurity/ModSecurity-apache

cd ModSecurity-apache

# Initialize the build configuration
./autogen.sh

# Configure the build, pointing it to the ModSecurity library we just
installed
./configure --with-libmodsecurity=/usr/local

# Compile the connector
make

# Install the module (mod_security3.so) into Apache's module folder
sudo make install

cd ..
```

Part 4: Configuration & Integration

We now need to tell Apache to load the module and set up the directory structure for our security rules.

Step 4.1: Load the Module

```
# Create the ModSecurity configuration directory
sudo mkdir -p /etc/apache2/modsecurity.d

# Add the LoadModule line to the main Apache config file
echo "LoadModule security3_module /usr/lib/apache2/modules/mod_security3.so"
| sudo tee -a /etc/apache2/apache2.conf
```

Step 4.2: Setup Basic Configuration

```
# Copy the recommended configuration file from the source code to our config
directory
sudo cp ModSecurity/modsecurity.conf-recommended
/etc/apache2/modsecurity.d/modsecurity.conf

# Copy the unicode mapping file (required for proper text processing)
sudo cp ModSecurity/unicode.mapping /etc/apache2/modsecurity.d/

# Switch ModSecurity from "DetectionOnly" (logging only) to "On" (Active
Blocking)
sudo sed -i 's/SecRuleEngine DetectionOnly/SecRuleEngine On/'
/etc/apache2/modsecurity.d/modsecurity.conf
```

Part 5: Installing OWASP Core Rule Set (CRS)

ModSecurity is the engine; CRS is the database of "signatures" that detect attacks like SQL Injection and XSS.

Step 5.1: Clone the Ruleset

```
# Clone the latest CRS into the modsecurity directory
sudo git clone https://github.com/coreruleset/coreruleset
/etc/apache2/modsecurity.d/owasp-crs

# Create the CRS setup file from the example provided
sudo cp /etc/apache2/modsecurity.d/owasp-crs/crs-setup.conf.example
/etc/apache2/modsecurity.d/owasp-crs/crs-setup.conf
```

Step 5.2: Configure Custom Rules (Optional) You requested changing the default action to strictly deny (403 Forbidden).

```
# Use nano to edit the file
sudo nano /etc/apache2/modsecurity.d/owasp-crs/crs-setup.conf
```

Find the `SecDefaultAction` lines, comment out the "pass" versions, and uncomment or add the "deny" versions:

```
#SecDefaultAction "phase:1,log,auditlog,pass"
#SecDefaultAction "phase:2,log,auditlog,pass"

SecDefaultAction "phase:1,log,auditlog,deny,status:403"
SecDefaultAction "phase:2,log,auditlog,deny,status:403"
```

Step 5.3: Link Rules to Apache We create a main file that includes all other necessary config files.

```
sudo bash -c 'cat > /etc/apache2/modsecurity.d/modsec_rules.conf << EOL
Include "/etc/apache2/modsecurity.d/modsecurity.conf"
Include "/etc/apache2/modsecurity.d/owasp-crs/crs-setup.conf"
Include "/etc/apache2/modsecurity.d/owasp-crs/rules/*.conf"
EOL'
```

Part 6: Finalize and Test

Step 6.1: Enable ModSecurity in Apache Edit your Apache configuration (or a virtual host file) to turn the rules on.

```
sudo nano /etc/apache2/apache2.conf

# Include the virtual host configurations:
IncludeOptional sites-enabled/*.conf
LoadModule security3_module /usr/lib/apache2/modules/mod_security3.so

# Enable ModSecurity & link config
modsecurity on
modsecurity_rules_file /etc/apache2/modsecurity.d/modsec_rules.conf

# Include coreruleset & rules
#Include /etc/apache2/modsecurity.d/modsec_rules.conf
#Include /etc/apache2/rules/*.conf
```

Step 6.2: Restart Apache

```
# Check for syntax errors first
sudo apachectl -t

# If Syntax OK, restart the server
sudo systemctl restart apache2
```

Step 6.3: Verification

```
# 1. Test normal traffic (Should work, HTTP 200)
curl localhost

# 2. Test an attack (Should fail, HTTP 403 Forbidden)
curl "localhost?doc=/bin/ls"

# 3. Check the logs for the block
sudo tail -f /var/log/modsec_audit.log
```

Part 7: Excluding a rule

Step 7.1: How to Identify the Rule to Exclude

Before you can exclude a rule, you need its **ID**. You find this in your audit log:

```
sudo tail -f /var/log/modsec_audit.log
```

Look for a string like `[id "920350"]`. That number is what you need.

Step 7.2: Global Exclusions

If a specific rule is causing problems across your entire server, use the `SecRuleRemoveById` directive.

Add this to the bottom of your `/etc/apache2/modsecurity.d/modsec_rules.conf` file.

```
# Exclude a single rule globally
SecRuleRemoveById 920350

# Exclude multiple rules at once
SecRuleRemoveById 920350 941110 942100
```

7.3 Apply and Verify

```
# Check syntax for errors
sudo apachectl -t

# Restart Apache to apply changes
sudo systemctl restart apache2
```