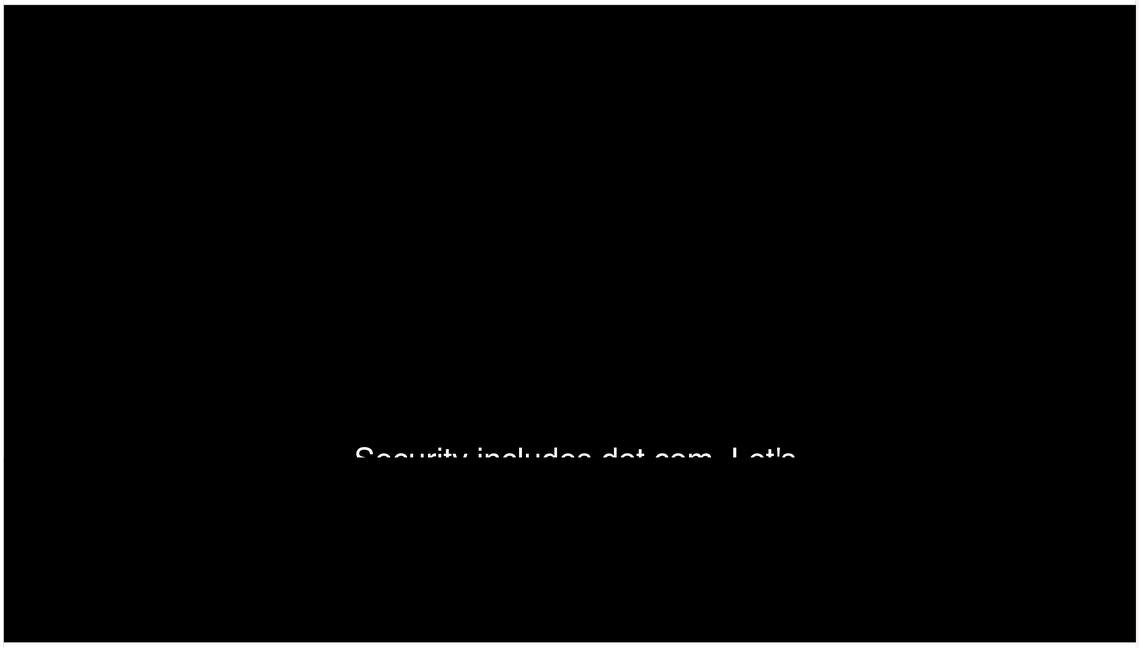
← NGINX Web Server Deep Dive

Lecture: Utilizing ModSecurity WAF



We've installed the `ModSecurity` module for NGINX, but we haven't done much with it yet. In this lesson, we'll pull in the OWASP ModSecurity Core Rule Set (CRS) and examine what this professionally sourced set of security rules provides for us.

Note: the commands in this video are run as the `root` user.

Documentation For This Video

- `ModSecurity`
- `ModSecurity-nginx`
- OWASP ModSecurity Core Rule Set

Installing OWASP ModSecurity Core Rule Set

The OWASP ModSecurity Core Rule Set is a collection of rules for ModSecurity that provide protection from many common attacks. These rules are created by the "Open Web Application Security Project" (OWASP) organization that is entirely focused on software security. Security and cryptography are two topics that are best left to those that study them extensively, and with that in mind, we're not going to create our own ModSecurity rule set. Our first step is to install the current version of the OWASP CRS using `git`:

```
[root] $ cd /etc/nginx/modsecurity
[root] $ git clone https://github.com/SpiderLabs/owasp-modsecurity-crs.git
[root] $ cd owasp-modsecurity-crs
```

This repository includes quite a lot, but most of what we're interested in is going to be in the `rules` directory. Before moving forward, we want to copy the files that include `.example` in the name to not include that:

```
[root] $ cp crs-setup.conf{.example,}
[root] $ cp rules/REQUEST-900-EXCLUSION-RULES-BEFORE-CRS.conf{.example,}
[root] $ cp rules/RESPONSE-999-EXCLUSION-RULES-AFTER-CRS.conf{.example,}
```

As of right now, those two rule files only contain documentation explaining how they work, and we're not going to put anything into them (but it's a good idea to read them). The important thing to note here is that we named them so that they aren't managed as part of the git repository, and if we put overrides in them, we can safely update the rule set using `git pull` later on without losing our customizations. With this modification made, we're ready to create our full rules file for use with NGINX.

Building and Using Our Full ModSecurity Rules File

It's best practice to create a ModSecurity configuration file that simply includes the parts that we want from both the ModSecurity default configuration file that we have and the specialized rule sets from OWASP. We need to create this file so that we have a single file that we can utilize with the `modsecurity_rules_file` directive. We're going to put this file directly in the `/etc/nginx/modsecurity` directory, and we'll call it `modsecurity_includes.conf`. The `REQUEST` and `RESPONSE` configuration files are numbered so that we know the order include them into our own custom configuration, and we want to include the base configuration from the `modsecurity.conf` as the very first thing followed by the initial CRS configuration (`crs-setup.conf`). Let's create this file now:

/etc/nginx/modsecurity/modsecurity_includes.conf

```
include modsecurity.conf
include owasp-modsecurity-crs/crs-setup.conf
```

To easily include the `REQUEST` and `RESPONSE` lines, let's write a bash loop to append to this file:

```
[root] $ for f in $(ls -1 owasp-modsecurity-crs/rules/ | grep -E "^(RESPONSE|REQUEST)-.*\.conf$"); do \
  echo "include owasp-modsecurity-crs/rules/${f}" >> modsecurity_includes.conf; done
```

With this file created, we can not use it with the `modsecurity_rules_file` and if we want to make customizations we can comment out includes that we don't need. Let's add ModSecurity for our WordPress site now:

/etc/nginx/conf.d/blog.example.com.conf (partial)

```
    root /var/www/blog.example.com;
    index index.php;

modsecurity on;
modsecurity_rules_file /etc/nginx/modsecurity/modsecurity_includes.conf;

access_log syslog:server=unix:/dev/log vhost;</code></pre>
```

Now we can validate the configuration and reload NGINX.

Testing The WAF

The last thing that we need to do is make sure that the WAF is actually doing something. We're going to simulate a cross-site scripting attack to make sure that ModSecurity catches it. First, let's start tailing the ModSecurity audit log:

```
[root] $ tail -f /var/log/nginx/modsecurity_audit.log
```