Step-by-Step Guide: Integrating Coraza WAF Plugin with Traefik on Docker Swarm

Prerequisites

- A working Docker Swarm cluster.
- Traefik configured on the `management_net` overlay network.
- Basic knowledge of Traefik's static and dynamic configuration files.

Part 1: Adding the Coraza WAF Plugin to Traefik

We will integrate the Coraza WAF plugin into Traefik to block access to a specific path (`/admin`) and log denied requests.

Step 1: Modify the `static.toml` Configuration

The first step is to enable the Coraza WAF plugin in the Traefik static configuration (`static.toml` file). This file defines the essential settings for Traefik and is loaded at startup.

```
[experimental.plugins]
  [experimental.plugins.coraza]
    moduleName = "github.com/jcchavezs/coraza-http-wasm-traefik"
    version = "v0.2.2"
```

> This enables the Coraza WAF plugin for Traefik.

Step 2: Configure Middleware in the `dynamic.toml`

Next, define the Coraza WAF middleware in the `dynamic.toml` file. This middleware will block access to `/admin` and log the event.

```
[http.middlewares]
  [http.middlewares.coraza-waf.plugin.coraza]
    directives = [
      "SecRuleEngine On",
      "SecDebugLog /dev/stdout",
      "SecDebugLogLevel 9",
      "SecRule REQUEST_URI \"@streq /admin\" \"id:101,phase:1,log,deny,status:403\""
    ]
```

- **`SecRuleEngine On`**: Activates the WAF engine.
- **`SecRule REQUEST_URI "@streq /admin"`**: This checks if the request URI matches `/admin`.
- **Action**: If it matches, the WAF logs the attempt and denies access with a `403 Forbidden` response.

Step 3: Deploy the Middleware on Docker Swarm

Now, let's create a `docker-compose.yml` file to deploy Traefik and its services in Docker Swarm, with 1 replica running on the `management_net` network. With the Static & Dynamic Configs in the

Glustermount.

This is an Example on how to Implement the Middleware into an Example Service called "whoami".

```
whoami:
  image: traefik/whoami
  networks:
    - management_net
  deploy:
    replicas: 1
    labels:
      - "traefik.http.routers.whoami.rule=Host(`whoami.aeoneros.com`)"
      - "traefik.http.middlewares.coraza-waf.plugin.coraza.directives"
```

Deploy the stack to Docker Swarm with the following command:

```
docker stack deploy -c docker-compose.yml waf_stack
```

> **This will deploy Whoami as a Service in Docker Swarm with the Coraza WAF middleware applied.**

Part 2: Adding OWASP Core Rule Set (CRS) to Coraza Middleware

Coraza doesn't include the OWASP CRS by default, but you can manually integrate the CRS to bolster security. Let’s walk through how to download, customize, and apply the CRS to the Coraza WAF.

Step 1: Download the Core Rule Set

Start by downloading the OWASP CRS from its official repository. This rule set provides security rules to protect against a wide range of common threats, including XSS, SQLi, and more.

Clone the repository:

```
git clone https://github.com/coreruleset/coreruleset.git
```

Step 2: Integrate the CRS into Coraza

Next, integrate the CRS into Coraza by modifying the `dynamic.toml` file to load the CRS rules.

Update the `dynamic.toml` to include the CRS rule files:

```
[http.middlewares]
  [http.middlewares.coraza-waf-crs.plugin.coraza]
    directives = [
      "Include /etc/modsecurity.d/coreruleset/crs-setup.conf",
      "Include /etc/modsecurity.d/coreruleset/rules/*.conf"
    ]
```

This configuration tells Coraza to load the Core Rule Set. The `crs-setup.conf` file is used for basic CRS configuration, and the `rules/*.conf` files contain the individual rule sets.

Step 3: Add Custom Rules

You can further enhance security by adding custom rules to your WAF configuration. For instance, you might want to protect your application against SQL injection attempts.

Add a custom SQL injection detection rule in the `dynamic.toml` file:

```
[http.middlewares]
  [http.middlewares.coraza-waf-custom.plugin.coraza]
    directives = [
      "Include /etc/modsecurity.d/custom_rules.conf",
      "SecRule ARGS \"@rx select.*from.*\" \"id:102,phase:2,log,deny,status:403,msg:'SQL Injection Attempt'\""
    ]
```

This rule will inspect the request arguments (query parameters) for SQL injection patterns and block the request if it detects a match.

Additional Examples: Core Rule Set Enhancements

1. Blocking SQL Injection

Add this rule to block SQL injection attempts in URL parameters:

```
SecRule ARGS "@rx select.*from.*" "id:103,phase:2,log,deny,status:403,msg:'SQL Injection Attempt'"
```

2. Enabling Rate Limiting

To prevent brute-force attacks or excessive requests, you can implement rate limiting using ModSecurity:

```
SecAction "id:104,phase:1,pass,nolog,initcol:ip=%{REMOTE_ADDR},expirevar:ip.counter=60"
SecRule IP:COUNTER "@gt 100" "id:105,phase:1,deny,status:429,msg:'Too Many Requests'"
```

This rule limits clients to 100 requests within a 60-second period.

Conclusion

Integrating Coraza WAF with Traefik is an excellent way to secure your web applications from common threats. By following this guide, you've successfully added Coraza to your Traefik setup, integrated the OWASP Core Rule Set, and customized rules to meet your security needs. With proper monitoring, troubleshooting, and performance considerations in place, you can deploy this WAF solution confidently in production environments.

Revision #5
Created 13 October 2024 15:15:21 by aeoneros
Updated 13 October 2024 15:30:45 by aeoneros