Coraza - Web Application Firewall

Overview

Overview

What is OWASP Coroza WAF?

Introduction

In today's security landscape, web applications are vulnerable to a variety of threats such as SQL injection (SQLi), Cross-Site Scripting (XSS), and brute-force attacks. A Web Application Firewall (WAF) is a crucial defense mechanism, filtering and monitoring HTTP traffic to prevent such threats. OWASP Coraza WAF is an open-source WAF solution, highly performant and designed to provide robust protection for modern web applications.

What is OWASP?

The **Open Web Application Security Project (OWASP)** is a non-profit, open-source foundation dedicated to improving the security of software. OWASP provides freely available resources, such as documentation, tools, and community support, to help developers and organizations secure their applications. Among its most notable contributions is the **OWASP Top 10**, a list of the most critical security risks to web applications. Another significant offering is the **OWASP Core Rule Set (CRS)**, a set of attack detection rules that can be used to protect web applications from various types of attacks.

What is ModSecurity and its Relation to Coraza WAF?

ModSecurity is one of the most widely used open-source WAF engines, initially developed as a module for Apache HTTP Server and now supporting other platforms. It helps secure web applications by filtering HTTP requests, using rulesets such as the OWASP CRS. ModSecurity can detect and block attacks like SQLi, XSS, and Local File Inclusion (LFI).

Coraza WAF builds upon the principles of ModSecurity but aims to offer a more modern, lightweight, and flexible approach. While ModSecurity has become a standard for many years, **Coraza WAF** introduces new performance optimizations and extensibility, ensuring high throughput for large-scale applications with minimal latency. Coraza is compatible with OWASP CRS, allowing it to offer similar attack prevention capabilities while being more adaptable to modern infrastructures like containers and cloud-native environments.

What is Coraza WAF?

Coraza WAF is an open-source, high-performance Web Application Firewall designed to protect web applications from common vulnerabilities and attacks. Built with extensibility and performance in mind, Coraza provides a modern alternative to legacy WAF solutions like ModSecurity. It offers features such as customizable rules, integration with OWASP CRS, and flexible deployment options in cloud-native environments.

Key Features:

- **Open-Source**: Coraza is fully open-source under the Apache 2 license, encouraging community-driven development and contribution.
- **Security**: It is designed to enforce security policies using either the OWASP CRS or custom rule sets, providing comprehensive protection against common attack vectors like SQLi and XSS.
- **High Performance**: Coraza is optimized for performance, making it suitable for a range of applications, from small blogs to high-traffic websites, without introducing significant latency.
- **Extensibility**: Coraza’s modular design allows for easy extension through custom audit loggers, persistence engines, and additional functionalities.
- **Integrations**: Although Coraza is primarily a WAF library, it supports numerous integrations, making it deployable as a reverse proxy, containerized service, or in traditional server setups.

Benefits of Adding WAF Middleware to Traefik

By integrating Coraza WAF into Traefik, you can significantly enhance the security posture of your applications. Some of the benefits include:

- **Enhanced Security**: Protect applications from common attack vectors such as SQLi, XSS, and brute-force attempts.
- **Centralized Management**: Apply security policies across all services managed by Traefik from a single location, simplifying administration.
- **Flexibility**: Modify or remove security rules without affecting the underlying infrastructure or requiring service downtime.
- **Customization**: Coraza allows you to create custom security rules that are tailored specifically to your application’s environment.
- **OWASP CRS Support**: Coraza can integrate the OWASP Core Rule Set to provide out-of-the-box protection against common vulnerabilities.

Conclusion

Coraza WAF, with its strong integration capabilities, performance, and extensibility, offers a modern approach to web application security. When paired with Traefik, it provides a powerful combination of reverse proxying and security enforcement, making it an excellent choice for safeguarding web applications in both traditional and cloud-native environments.

Overview

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.

Troubleshooting

Monitoring and Troubleshooting Coraza WAF

For users deploying Coraza WAF in production environments, **monitoring and troubleshooting** are essential for ensuring optimal security and performance.

Monitoring WAF Logs

Coraza's WAF rules can be monitored through log files. Logs can be directed to standard output (`/dev/stdout`) to view in real time, or you can configure log files for long-term monitoring.

To monitor logs, ensure you have the following settings in your `dynamic.toml`:

```
[http.middlewares]
  [http.middlewares.coraza-waf-logging.plugin.coraza]
    directives = [
      "SecDebugLog /dev/stdout",
      "SecDebugLogLevel 9"
    ]
```

Use `docker logs` to view the WAF activity logs:

```
docker logs $(docker ps -qf name=traefik)
```

Performance Considerations

Introducing a WAF may add latency to your application due to the extra processing required to inspect HTTP requests. To monitor performance, you can use tools like Prometheus and Grafana to gather metrics on request processing time and WAF performance.

Troubleshooting Issues

When troubleshooting WAF-related issues:

- Check the **debug logs** for detailed information on blocked requests.
- Use **whitelisting** techniques to avoid false positives. For example, you can disable specific rules for known safe traffic by using the `SecRuleRemoveById` directive.