



# Overview

CasWAF is an open-source Web Application Firewall (WAF) software developed by Go and React.

## CasWAF features

1. Front-end and back-end separate architecture, developed by Golang, CasWAF provides web-based managing UI and supports multiple languages(Chinese, English).
2. Databases. CasWAF supports mainstream databases: MySQL, PostgreSQL, SQL Server, etc.
3. Casdoor SSO. CasWAF uses [Casdoor](#) as the Identity Provider (IdP) for OAuth login.
4. Reverse proxy. CasWAF supports reverse proxy, which can be used as a reverse proxy server to protect the backend server.
5. OAuth proxy. CasWAF supports OAuth login, which can be integrated with the existing OAuth system such as [Casdoor](#).
6. Firewall. CasWAF uses [Coraza](#) as the firewall engine, which can protect the website from many common attack categories. Supports customized WAF rules.

# How it works

CasWAF has both reverse proxy and OAuth proxy functionalities. If you haven't configured OAuth for your website, it will function solely as a reverse proxy server.

## Reverse proxy

CasWAF appears externally as a reverse proxy server, providing an additional layer of security for your web servers and applications.

It sits between the users and web servers, acting as an intermediary, receiving requests from users, and forwarding them to the target web servers.

## OAuth proxy

CasWAF OAuth proxy acts as an Identity Provider (IdP) and collaborates with your application to perform authentication and authorization through the OAuth protocol.

When a user attempts to access a resource that requires authentication, CasWAF will redirect the user to the real Identity Provider (IdP).

Subsequently, the OAuth proxy will guide the user to the configured Identity Provider (e.g., Google, Facebook, or an internal authentication service within the company) for authentication.

In CasWAF, we use [Casdoor](#) as the Identity Provider (IdP). More information of Casdoor SSO can be found [here](#).

# Online Demo

Here is an online demo:

- Deployed site: <https://door.caswaf.com>

Global admin login:

- Username: `admin`
- Password: `123`

## Architecture

Caswaf contains 2 parts:

Name	Description	Language	Source code
Frontend	Web frontend UI for CasWAF	Javascript + React	<a href="https://github.com/casbin/caswaf/tree/master/web">https://github.com/casbin/caswaf/tree/master/web</a>
Backend	RESTful API backend for CAsWAF	Golang + Beego + MySQL	<a href="https://github.com/casbin/caswaf">https://github.com/casbin/caswaf</a>

[The Basics](#)[Core Concepts](#)

# Core Concepts

As CasWAF's administrator, you should get familiar with at least 2 core concepts:

[Site](#), [Cert](#).

## Site

In CasWAF, [Site](#) is representing the real applications or websites you wish to protect. Each Site is associated with specific domain names or IP addresses, and you can configure multiple Sites according to different needs to ensure comprehensive security protection for all your applications.

The Site class definition is shown as follows:

```
type Site struct {
    Owner      string `xorm:"varchar(100) notnull pk"
    json:"owner"`
    Name       string `xorm:"varchar(100) notnull pk" json:"name"`
    CreatedTime string `xorm:"varchar(100)" json:"createdTime"`
    UpdatedTime string `xorm:"varchar(100)" json:"updatedTime"`
    DisplayName string `xorm:"varchar(100)" json:"displayName"`

    Tag        string    `xorm:"varchar(100)" json:"tag"`
    Domain     string    `xorm:"varchar(100)" json:"domain"`
    OtherDomains []string `xorm:"varchar(500)"
    json:"otherDomains"`
    NeedRedirect bool      `json:"needRedirect"`
    EnableWaf   bool      `json:"enableWaf"`
    Waf         coraza.WAF `xorm:"- " json:"- "`
    Challenges  []string  `xorm:"mediumtext" json:"challenges" `
```

# Cert

In CasWAF, `Cert` is representing the certificates used for HTTPS authentication. By configuring `Cert`, CasWAF can establish secure encrypted connections between itself and the clients, ensuring the confidentiality and integrity of data and preventing information leakage and tampering attacks.

The `Cert` class definition is shown as follows:

```
type Cert struct {
    Owner      string `xorm:"varchar(100) notnull pk"
    json:"owner"`
    Name       string `xorm:"varchar(100) notnull pk" json:"name"`
    CreatedTime string `xorm:"varchar(100)" json:"createdTime"`
    DisplayName string `xorm:"varchar(100)" json:"displayName"`

    Type       string `xorm:"varchar(100)" json:"type"`
    CryptoAlgorithm string `xorm:"varchar(100)"
    json:"cryptoAlgorithm"`
    ExpireTime  string `xorm:"varchar(100)" json:"expireTime"`

    Certificate string `xorm:"mediumtext" json:"certificate"`
    PrivateKey  string `xorm:"mediumtext" json:"privateKey"`
}
```



# Installation

CasWAF uses Casdoor to manage members. So you need to create an organization and an application for CasWAF in a Casdoor instance.

## Requirements

### OS

All major operating systems including Windows, Linux and macOS are supported.

### Environments

- [Go 1.17+](#)
- [Node.js LTS \(16 or 14\)](#)
- [Yarn 1.x](#)

#### ⚠ 信息

We strongly suggest you use [Yarn 1.x](#) to run & build CasWAF frontend, using NPM might cause UI styling issues, see more details at: [casdoor#294](#)

#### ⚠ 警告

For Chinese users, in order to download the Go dependency packages successfully, you need to use a Go proxy by Configuring the GOPROXY environment variable. We strongly recommend: <https://goproxy.cn/>

# Download

The source code of CasWAF is hosted at GitHub: <https://github.com/casbin/caswaf>. Both the Go backend code and React frontend code are inside the single repository.

Name	Description	Language	Source code
Frontend	Web frontend UI for CasWAF	Javascript + React	<a href="https://github.com/casbin/caswaf/tree/master/web">https://github.com/casbin/caswaf/tree/master/web</a>
Backend	RESTful API backend for CAsWAF	Golang + Beego + MySQL	<a href="https://github.com/casbin/caswaf">https://github.com/casbin/caswaf</a>

CasWAF supports Go Modules. To download the code, you can just simply clone the code both via `go get` and `git`:

```
go get github.com/casbin/casdoor
go get github.com/casbin/caswaf
```

or

```
git clone https://github.com/casbin/casdoor
git clone https://github.com/casbin/caswaf
```



# Necessary configuration

## Set up database

CasWAF will store its users, nodes and topics information in a MySQL database named: `caswaf`. CasWAF will create it if not existed. The DB connection string can be specified at: <https://github.com/casbin/caswaf/blob/master/conf/app.conf>

```
dataSourceName = root:123@tcp(localhost:3306)/
```

CasWAF uses XORM to connect to DB, so all DBs supported by XORM can also be used.

## Configure Casdoor

### ! 信息

In order not to affect Docker users, we temporarily chose to embed the WAF rules into the binary, if you need to change the default rules (`conf/waf.conf`), please do so before compiling.

After creating an organization and an application for CasWAF in a Casdoor, you need to update `clientId`, `clientSecret`, `casdoorOrganization` and `casdoorApplication` in `app.conf` and `Conf.js` to change the configuration.

- Backend (`conf/app.conf`)

```
casdoorEndpoint = <Your Casdoor endpoint>
```

- Frontend (web/src/Conf.js)

```
serverUrl: "<Your Casdoor endpoint>"
clientId: "<Your Casdoor application's client ID>"
appName: "<Your Casdoor application name>"
organizationName: "<Your Casdoor organization name>"
```

More details about Casdoor configuration can be found at: [casdoor-sso](#)

## Run CasWAF

- Build backend of CasWAF
  - `go build github.com/casbin/caswaf`
- Build frontend of CasWAF
  - `yarn start`
- Now you can visit CasWAF configuration website at:
  - `http://localhost:16001/`

## Optional configuration

### Set up your WAF to enable some third-party login platform

CasWAF uses Casdoor to manage members. If you want to log in with oauth, you should see [casdoor oauth configuration](#).

### OSS, Mail, and SMS services

CasWAF uses Casdoor to upload files to cloud storage, send Emails and send SMSs. See Casdoor for more details.



The Basics



Casdoor SSO

# Casdoor SSO

## Introduction

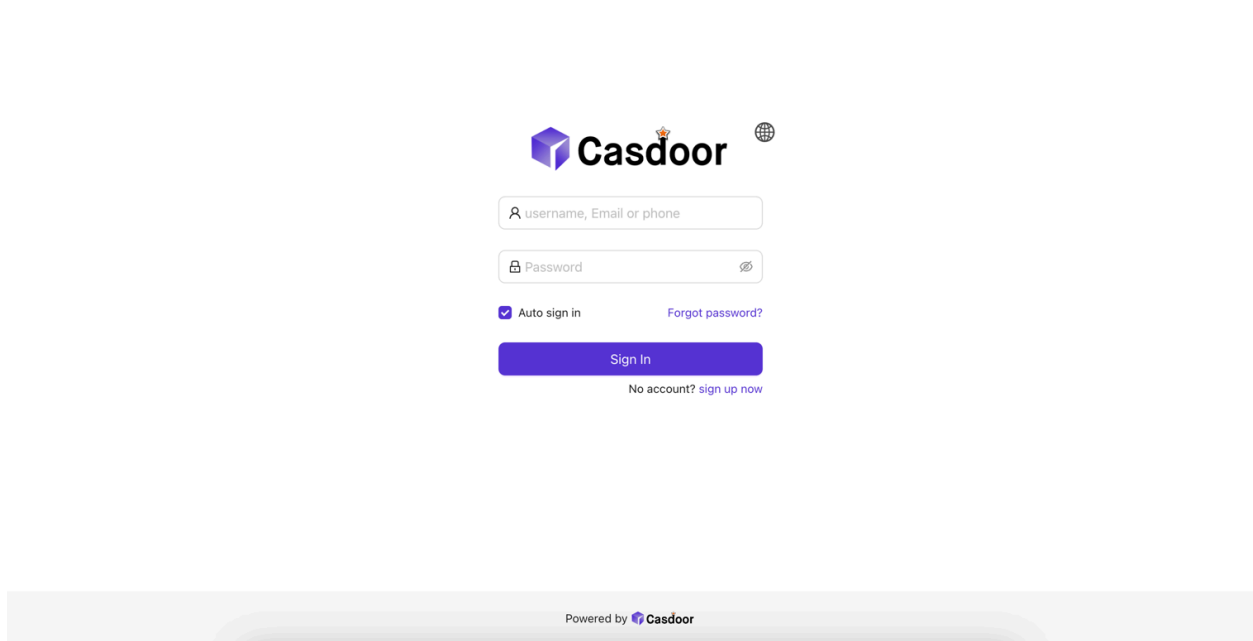
Casdoor is a UI-first centralized authentication / Single-Sign-On (SSO) platform based on OAuth 2.0 / OIDC. It provides OAuth 2.0 / OIDC based signup and login, as well as third-party login including GitHub, Google, QQ, WeChat, etc. CasWAF uses Casdoor as an SSO provider to manage users and permissions.

## Login to CasWAF

CasWAF supports logging in with Casdoor. You need to register CasWAF as a Casdoor application first.

About how to register CasWAF as a Casdoor application, please refer to [Casdoor documentation](#).

After registering CasWAF as a Casdoor application, you can log in to CasWAF with Casdoor.



After logging in, you can manage the site and certificate according to the instructions provided in the [Sites documentation](#) and [Cert documentation](#).

Home Sites Certs ...									admin Admin ▾
Sites <a href="#">Add</a>									
Name ▾	Display name ▾	Domain ▾	Host ▾	Public IP ▾	Node ▾	SSL mode ▾	SSL cert ▾	Casdoor app ▾	Action
<a href="#">site_y9wchu</a>	New Site - y9wchu	<a href="#">door.casdoor.com</a>	http://localhost:8000	8.131.81.162		HTTP			<a href="#">Edit</a> <a href="#">Delete</a>
< 1 >									

❗ TODO

add content



Sites

# Sites



## Site List

Learn how to configure your site in CasWAF.

[Sites](#)[Site List](#)

# Site List

`Site` is representing the real applications or websites you wish to protect.

This section will provide a detailed explanation of the properties and usage of `Site`.

## Site properties

- `Name`: The name of the site.
- `DisplayName`: The display name of the site.
- `Domain`: The domain of the site.
  - e.g. `blog.example.com`
- `Host`: The host of the site.
  - e.g. `localhost:8080`
- `Public IP (Optional)`: The public IP of the site (if available).
- `Node (Optional)`: The name of the host on which the site is deployed.
- `Rules`: The rules used in the site to handle requests. Users can select the rules from the dropdown list.
  - About how to add rules, please refer to [Rule List](#).
- `SSL mode`: The SSL mode of the site. It can be `HTTP` or `HTTPS and HTTP` or `HTTPS only`.
  - `HTTP`: The site is not using SSL. Users can access the site **only** via HTTP.
  - `HTTPS and HTTP`: The site is using SSL and HTTP. Users can access the site via **both** HTTP and HTTPS.
  - `HTTPS only`: The site is using SSL only. Users can access the site **only**

via HTTPS. If users access the site via HTTP, they will be redirected to HTTPS.

- **SSL cert**: The SSL certificate of the site. User can select the SSL certificate from the dropdown list.
  - About how to add SSL certificate, please refer to [SSL Certificates](#).
- **Casdoor app**: If your site needs OAuth login, you can select the Casdoor app from the dropdown list.
  - CasWAF uses Casdoor as the OAuth server. So you need to register your site as a Casdoor app first.
  - About how to add Casdoor app, please refer to [Casdoor SSO](#).

# Usage

## Manage sites

casbin

Home

Sites

Certs

...

Admin

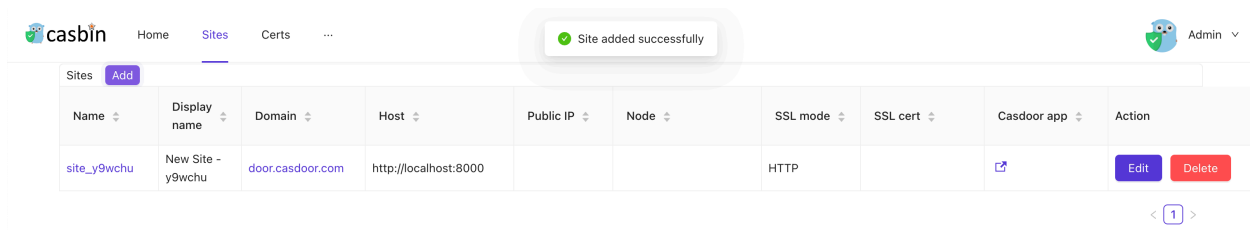
Sites

Add

Name	Display name	Domain	Host	Public IP	Node	SSL mode	SSL cert	Casdoor app	Action
<div><div></div><div>No Data</div></div>									

In the Sites page, you will see all sites you have created. You can create, edit, delete and view the site details. But now we don't have any sites.

# Add a site



Just click the **Add** button, you will create a site. The created Site will have some default information that you can modify.

## Edit site

After you create a site, you can click the **edit** button to edit the site.

**Edit Site** Save

Name:

Display name:

Domain:

Host:

Public IP:

Node:

SSL mode:

SSL cert:

Casdoor app:

Save

Each field's meaning is as described above, and you can freely modify them according to the actual situation of your website.

After you have completed the modifications, you only need to click the **Save** button to save your settings.





Certs

# Certs



## Cert List

Learn how to add your server's certificates in CasWAF.

# Cert List

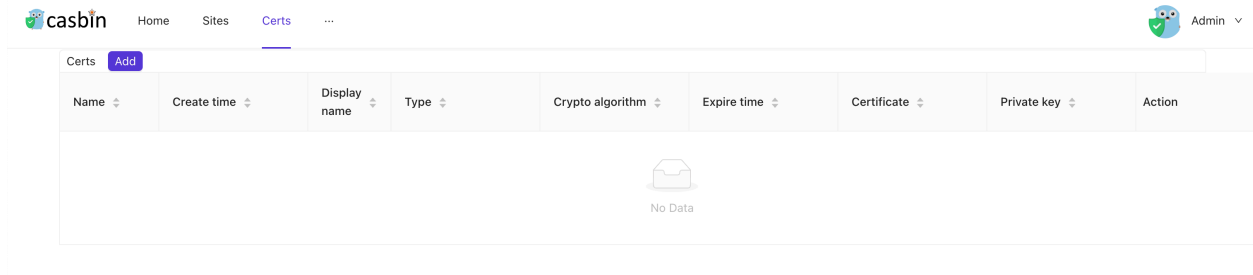
If your website supports HTTPS access, you need to configure an SSL certificate in CasWAF to enable it to establish a secure connection with your server as a reverse proxy.

This section will provide a detailed explanation of the properties and usage of `Cert`.

## Cert properties

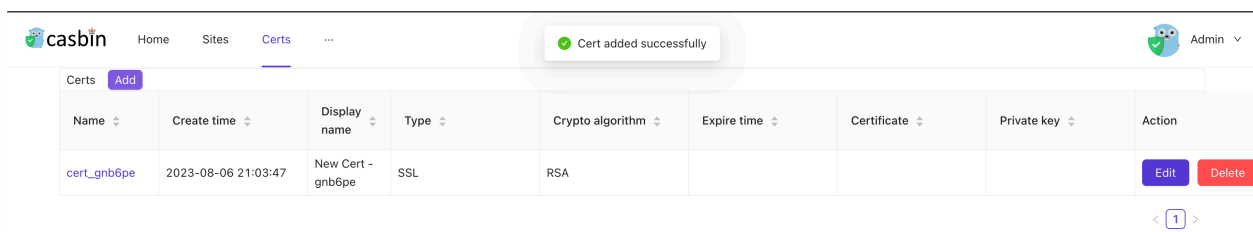
- `Name`: The name of the certificate.
- `Display name`: The display name of the certificate.
- `Type`: The type of the certificate. This field is usually set to `SSL` since SSL is the most commonly used option.
- `Crypto algorithm`: The crypto algorithm of the certificate. It can be set to `RSA` or `ECC` which is depend on algorithm your certificate used.
- `Expire time`: The expire time of the certificate. This field will be automatically filled in when you set certificate and private key.
- `Certificate`: The certificate content.
  - The certificate is a public key that has been authenticated by a Certificate Authority (CA).
- `Private key`: The private key content.
  - The private key is a secret key that is used to encrypt and decrypt data.

# Manage Certificates



In the Certs page, you will see all certs you have created. You can create, edit, delete and view the cert details. But now we don't have any certs.

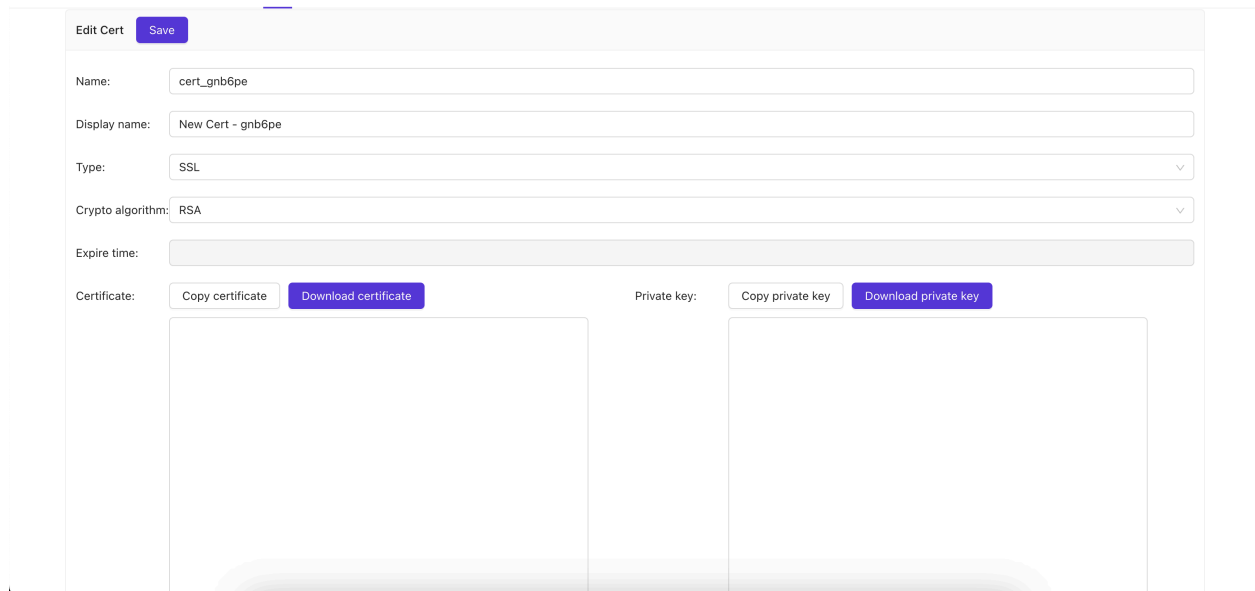
## Add a cert



Just click the **Add** button, you will create a cert. The created cert will have some default information that you can modify.

## Edit cert

After you create a site, you can click the **edit** button to edit the site.



The screenshot shows a web form titled "Edit Cert" with a "Save" button. The form contains the following fields and controls:

- Name:** A text input field containing "cert\_gnb6pe".
- Display name:** A text input field containing "New Cert - gnb6pe".
- Type:** A dropdown menu with "SSL" selected.
- Crypto algorithm:** A dropdown menu with "RSA" selected.
- Expire time:** A date and time picker field.
- Certificate:** A section with a "Copy certificate" button and a "Download certificate" button. Below these buttons is a large, empty rectangular box for the certificate content.
- Private key:** A section with a "Copy private key" button and a "Download private key" button. Below these buttons is a large, empty rectangular box for the private key content.

Each field's meaning is as described above, and you can freely modify them according to the actual situation of your website.

After you have completed the modifications, you only need to click the **Save** button to save your settings.



Rules

# Rules



## Rule List

Learn how to add your server's rule in CasWAF.



## IP Rule

Learn how to config IP rules in CasWAF.



## User-Agent Rule

Learn how to config User-Agent rules in CasWAF.



## WAF Rule

Learn how to config WAF rules in CasWAF.

# Rule List

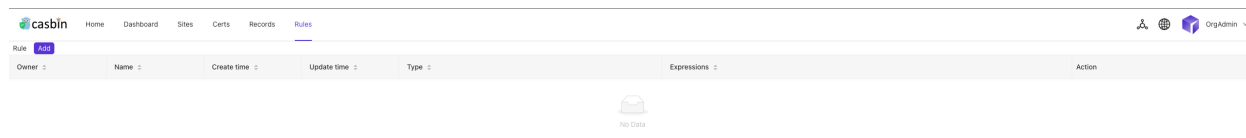
If you want your website under protection, you can add your rules in CasWAF.

This section will provide a detailed explanation of the properties and usage of Rule.

## Rule properties

- **Name**: The name of the rule.
- **Type**: The type of the rule. It can be set to **IP**, **User-Agent** and **WAF**.
- **Expressions**: The expressions contains the rule's conditions, including **operator** and **value**.
- **Action**: The action of the rule. It can be set to **Allow** or **Block**.
- **Reason**: The reason of the rule. When the rule is hitted and its action is **Block**, CasWAF would reply a 403 response with the reason.

## Manage Rules

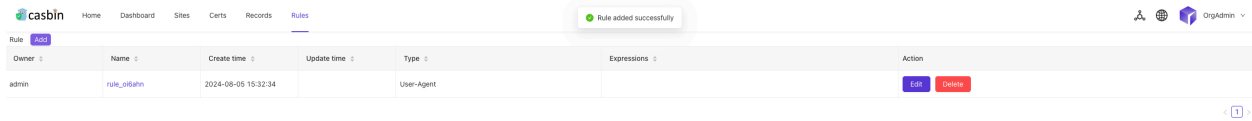


The screenshot shows the Casbin web interface. The top navigation bar includes the Casbin logo and links to Home, Dashboard, Sites, Certs, Records, and Rules (which is highlighted). On the right, there are icons for user management, a globe, and a dropdown menu for 'OrgAdmin'. Below the navigation bar, there is a 'Rule' section with a '+ Add' button. The main area contains a table with columns: Owner, Name, Create time, Update time, Type, Expressions, and Action. The table is currently empty, displaying a 'No Data' message with a folder icon.

Owner	Name	Create time	Update time	Type	Expressions	Action
No Data						

In the Rules page, you will see all rules you have created. You can create, edit, delete and view the rule details. But now we don't have any rules.

# Add a rule



Just click the **Add** button, you will create a rule. The created rule will have some default information that you can modify.

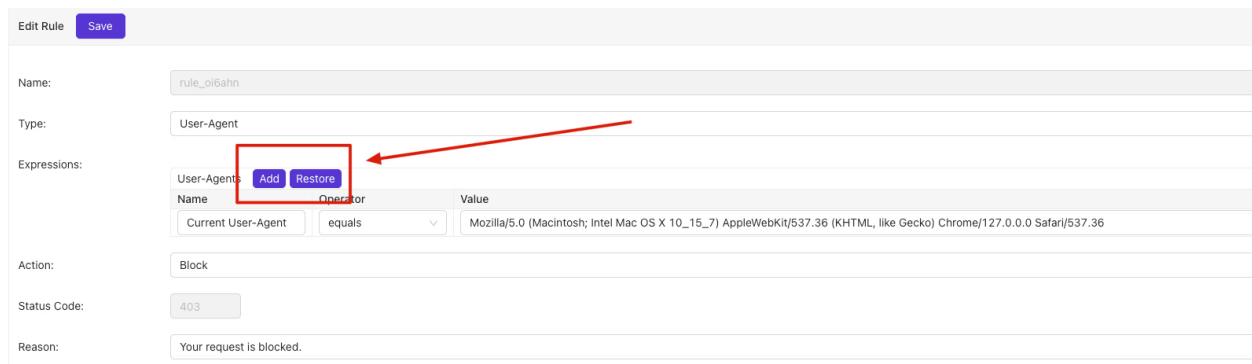
# Edit rule

After you create a rule, you can click the **edit** button to edit the rule.



Each field's meaning is as described above, and you can freely modify them according to the actual situation of your website.

# Expressions



Edit Rule [Save](#)

Name: rule\_o16ahn

Type: User-Agent

Expressions:

User-Agents [Add](#) [Restore](#)

Name	Operator	Value
Current User-Agent	equals	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36

Action: Block

Status Code: 403

Reason: Your request is blocked.

Click the [Add](#) button in the [Expressions](#) section, you can add a new expression to the rule.

Click the [Restore](#) button in the [Expressions](#) section, you can restore the default expressions of the rule.



# IP Rule

CasWAF provides a powerful IP rule feature to help you control the access of your website. You can add IP rules to allow or block specific IP addresses or IP ranges (CIDR only).

The screenshot shows the 'Edit Rule' interface in the Casbin console. The rule is named 'rule\_ipban' and is of type 'IP'. It contains two expressions: 'loopback' and 'lan cidr', both using the 'is in' operator. The values are '127.0.0.1' and '10.0.0.0/8 - 192.168.0.0/16' respectively. The action is set to 'Block' with a status code of '403' and a reason of 'Your request is blocked'.

Name	Operator	Value	Action
loopback	is in	127.0.0.1	Block
lan cidr	is in	10.0.0.0/8 - 192.168.0.0/16	

## IP Expression properties

- Name**: The name of the expression. It is used to identify the rule and have no effect on the rule itself.
- Operator**: The operator of the expression. It can be set to **is in** or **is not in**.
- Value**: The value of the expression. It can be set to IPv4 or IPv6 addresses or CIDR ranges (even mixed).

# User-Agent Rule

CasWAF provides a User-Agent rule feature to help you control the access of your website. You can add User-Agent rules to allow or block specific User-Agents.

The screenshot shows the 'Edit Rule' interface in the Casbin console. The 'Name' field is 'rule\_casbin'. The 'Type' is 'User-Agent'. Under 'Expressions', there is a table with columns 'Name', 'Operator', 'Value', and 'Action'. The first row has 'Current User-Agent' as the Name, 'equals' as the Operator, and a long Mozilla/5.0 string as the Value. The 'Action' is set to 'Block'. The 'Status Code' is '403' and the 'Reason' is 'Your request is blocked.' There are 'Save' buttons at the top and bottom of the form.

Name	Operator	Value	Action
Current User-Agent	equals	Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari/537.36	Block

## User-Agent Expression properties

- **Name**: The name of the expression. It is used to identify the rule and have no effect on the rule itself.
- **Operator**: The operator of the expression. It can be set to `equals`, `does not equals`, `contains`, `does not contain` and `regex match`.
- **Value**: The value of the expression. It can be set to any string. The default value is your current User-Agent. If you are using `regex match`, the value should be a regular expression.

# WAF Rule

CasWAF use coraza WAF as the WAF engine, you can follow the [coraza seclang documents](#) to write your WAF rules.

## WAF Expression properties

- `value`: The value of the rule. It should be a valid coraza WAF rule.

## Supported disruptive actions

Seclang action	CasWAF action	Status code
allow	Allow	200
block	Block	403
deny	Deny	403
drop	Drop	400

### ! 信息

We recommend that you have a good understanding of the coraza WAF rules before you write your own rules.