VIKI53

Geste de savoir

Filtrer le spam grâce à AbuseIPDB

samedi 15 février 2025

Table des matières

Introduction

Il est difficile de nos jours de publier un formulaire de contact ou pour commenter un article de blog sans être spammé constamment. Les pots à miel n'étant pas très efficaces, j'ai récemment testé une méthode qui permet de filtrer plus efficacement le spam en utilisant une base de données collaborative d'adresses IP : AbuseIPDB.

> **Un anti-spam, et bien plus encore**
>
> Cette base de donnée n'est pas réservée qu'au signalement de spam : les cyberattaques sont également cataloguées. Une intégration native à fail2ban est d'ailleurs possible pour signaler automatiquement les tentatives d'intrusion sur votre serveur.

1. La collaboration au service de la sécurité

Même si les spammeurs ont aujourd'hui des outils sophistiqués pour automatiser le remplissage et l'envoi de formulaire, ils ont encore tendance à le faire depuis les mêmes machines pendant un moment. Et c'est justement grâce à cette faiblesse que l'on peut les repérer : en effet, ils changent rarement d'adresse IP.

Lorsqu'une adresse est signalée plusieurs fois comme source de spam (ou d'attaque), il suffit alors de la bloquer pour éviter d'en être soi-même victime.

C'est là qu'AbuseIPDB joue un rôle important : il s'agit d'une base de donnée collaborative où chaque inscrit peut à la fois signaler des attaques, mais aussi interroger la base pour obtenir à la fois une liste des signalements pour une adresse IP mais également obtenir un score (calculé en fonction de plusieurs facteurs, pour éviter les abus et rendre ce chiffre aussi objectif que possible) indiquant la probabilité qu'une adresse soit nuisible.

Non seulement le site permet de réaliser ces actions manuellement (pratique si vous voulez par exemple vérifier que votre propre adresse n'a pas été utilisée pas le passé par exemple), mais ils proposent surtout une API Web pour interroger leurs données directement depuis votre site ou application.

Pour éviter les abus le site demande à la fois d'être inscrit pour signaler une IP, mais il faudra générer une clé d' API pour l'utiliser.

Attention aux limites

Les API sont soumises à des limites d'accès quotidiennes, en fonction de votre abonnement. La version gratuite est suffisante pour un site perso, mais il faudra probablement passer à un abonnement payant si vous avez un trafic important.
Vous pouvez également vérifier votre nom de domaine et publier un lien vers votre profil pour augmenter gratuitement les limites appliqués à votre compte.

Une fois votre compte créé et une clé d' API générée, vous pouvez commencer à **implémenter un filtre anti-spam en quelques minutes** sur votre site !

À propos des exemples dans cet article

Mon site utilise le package Laravel officiel pour utiliser plus simplement ces API, mais les exemples partagés ici seront développés en PHP simple. Vous pouvez bien entendu les adapter à n'importe quel langage (Node.js, Python, Java…) pour implémenter les mêmes fonctionnalités dans votre projet.

2. Récupérer l'adresse IP du client

Lorsqu'un client fait appel à votre formulaire, son adresse IP est accessible via les informations envoyés au serveur.

Laravel propose un raccourci via `request()->ip()`, mais hors de ce framework il faudra probablement récupérer cette information manuellement (ou trouver le raccourci proposé par votre outil de choix) :

```php
$apiKey = 'Insérez votre clé ici';
$ip = $_SERVER['REMOTE_ADDR'];
```

Attention aux entêtes HTTP

Vous pouvez trouver d'autres bouts de code sur internet qui utilisent plutôt `$_SERVER['HTTP_CLIENT_IP']`, qui correspond à un entête HTTP. Celui-ci étant envoyé par le client il peut facilement être falsifié. Faites attention à utiliser une source d'information fiable lorsqu'il s'agit de sécuriser votre site.

3. Contrôler une adresse IP

Maintenant que vous avez récupéré l'adresse du client, vous pouvez interroger AbuseIPDB pour savoir si cette adresse a été utilisée pour des attaques récentes ? J'utiliserai ici `cURL` pour rendre le code le plus universel possible, mais vous pouvez également utiliser un client HTTP de votre choix, par exemple Guzzle, pour simplifier votre code.

```
$ch =
    curl_init('https://api.abuseipdb.com/api/v2/check?ipAddress='
    . urlencode($ip));
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, [
        'Key: ' . $apiKey,
        'Accept: application/json'
]);
$data = curl_exec($ch);
curl_close($ch);

$result = json_decode($data);
```

Avec le paquet pour Laravel, vous pouvez utiliser `AbuseIPDB::check($request->ip(), 30);` pour obtenir le même résultat.

La donnée qui va nous intéresser ici est contenue dans la clé `abuseConfidenceScore` de l'objet `data` retourné. Il s'agit d'un pourcentage de confiance, `100` représentant une certitude que l'adresse est nuisible.

Libre à vous d'utiliser le pourcentage que vous souhaitez comme limite pour bloquer une requête. J'utiliserai ici `75` pour bloquer les spammeurs potentiels.

```
if ($result->data->abuseConfidenceScore >= 75) {
        http_response_code(403);
        die('Spam détecté');
}
```

Avec Laravel pour pouvez renvoyer une erreur 403 et la page appropriée avec `abort(403, 'Spam détecté');`.

N'oubliez pas l'assurance qualité

Pensez à proposer une page d'erreur personnalisée pour expliquer à vos utilisateurs les raisons de ce filtrage. Les faux positifs sont vite arrivés, même avec une limite de filtrage haute.

4. Signaler un abus

Bien entendu, la même API permet également de signaler une adresse IP lorsque vous êtes victime d'une attaque.

Sur mon site par exemple j'enregistre l'adresse IP utilisée pour soumettre un formulaire public. Cela me permet ensuite de signaler les contrevenants qui n'auraient pas été directement filtré (par exemple lorsqu'ils utilisent une adresse considérée comme fiable).

Attention lorsque vous enregistrez une adresse IP. Comme il s'agit d'une donnée personnelle, le RGPD est très strict quant à son traitement.

Cette fois il suffit d'envoyer une requête POST au bon *endpoint* :

```
$ch = curl_init('https://api.abuseipdb.com/api/v2/report');
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_POSTFIELDS, http_build_query([
        'ipAddress' => $ip,
        'categories' => '10', // Cf.
            https://www.abuseipdb.com/categories pour choisir les
            bons identifiants, séparés par une virgule
        // Vous pouvez ajouter les paramètres facultatifs
            `comment` et `timestamp` si besoin
]));
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true);
curl_setopt($ch, CURLOPT_HTTPHEADER, [
        'Key: ' . $apiKey,
        'Accept: application/json'
]);
$data = curl_exec($ch);
curl_close($ch);
```

Cette requête renverra le score `abuseConfidenceScore` mis à jour, si cette donnée vous intéresse.

Avec le paquet Laravel, ce code peut être résumé par `AbuseIPDB::report($request->ip(), [10]);`.

Supprimer un signalement

Votre signalement étant lié à votre compte (via la clé d'API utilisée), vous pouvez le retrouver sur votre compte pour le supprimer en cas d'erreur (par exemple si vous avez signalé votre propre adresse pendant vos tests). Vous pouvez également effectuer cette action via l'API bien entendu.

5. Aller plus loin

N'hésitez pas à parcourir la documentation d'API d'AbuseIPDB pour en savoir plus sur les paramètres acceptés et découvrir d'autres usages possibles.

Vous pouvez par exemple l'utiliser pour lister les signalements faits sur votre adresse IP pour vous assurer que vous n'avez pas été signalé par quelqu'un d'autre (ce que j'ai fait sur l'interface d'administration de mon site).

Vous pouvez utiliser des intégrations ou plugins officiels dans les outils que vous avez déjà mis en place. Par exemple fail2ban peut signaler automatiquement les tentatives d'intrusion répétées en ajoutant quelques lignes à sa configuration.

6. Bonus : un middleware Laravel prêt à l'usage

Si comme moi vous utilisez Laravel pour votre projet, je vous partage mon middleware prêt à l'usage qui permet de filtrer et signaler les IP classées comme spam :

```php
<?php

namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Symfony\Component\HttpFoundation\Response;
use AbuseIPDB\Facades\AbuseIPDB;
use App\Enums\SpamType;

class FilterSpamIp
{
    public static $reportCategories = [
        'blog-comment' => [10, 12],
        'contact-form' => [10],
    ];

    public static $reportComments = [
        'blog-comment' => 'Blog comment spam attempt',
        'contact-form' => 'Contact form spam attempt',
    ];

    /**
     * Handle an incoming request.
     *
     * @param  \Closure(\Illuminate\Http\Request): (\Symfony\Componen
     */
```

```
28      public function handle(Request $request, Closure $next, string
            $type = SpamType::ContactForm): Response
29      {
30          $checkResponse = AbuseIPDB::check($request->ip(), 30);
31
32          if ($checkResponse->abuseConfidenceScore > 75) { //
                Plausible spam IP address
33              $reportCategories =
                    collect(self::$reportCategories)->get($type, [10]);
34              $reportComment =
                    collect(self::$reportComments)->get($type,
                    'Web spam attempt');
35
36              AbuseIPDB::report($request->ip(), $reportCategories,
                    $reportComment);
37
38              abort(403, 'Spam IP detected');
39          }
40
41          return $next($request);
42      }
43  }
```

Conclusion

La version originale de cet article est disponible sur mon blog.

Liste des abréviations

API Application Programming Interface. 2

RGPD Règlement général sur la protection des données. 4