DOI 10.37882/2223-2966.2023.06.27

РАЗРАБОТКА АЛГОРИТМА ДЛЯ ПРОТИВОДЕЙСТВИЯ СПАМУ В БЕСЕДАХ ВКОНТАКТЕ

Пиманов Андрей Евгеньевич
Технический институт (филиал) федерального государственного автономного образовательного учреждения высшего образования «Северо-Восточный федеральный университет имени М.К. Аммосова», г. Нерюнгри
epimanov15@gmail.com

Аннотация. Социальная сеть ВКонтакте известна среди лиц, занимающихся спам рассылкой, как источник условно-бесплатного трафика. Причиной этому стали два фактора. Во-первых, это отличная замена профессиональному таргетингу. Во-вторых, администрация социальной сети при борьбе со спамом обходится лишь полумерами, замораживая страницы, от которых он исходит. Таким образом социальная сеть не борется ни с источником спама, ни с его последствиями.

На текущий момент пользователи, находящиеся в групповых чатах с открытым доступом для всех желающих, наиболее часто сталкиваются с раздражающей рассылкой. В связи с чем необходимы эффективные методы защиты от спама для решения данной проблемы.

Ключевые слова: скрипт, алгоритм, спам, VK API, PHP, ВКонтакте.

DEVELOPMENT OF AN ALGORITHM TO COUNTER SPAM IN VKONTAKTE CONVERSATIONS

A. Pimanov

Summary: The VKontakte social network is known among people engaged in spam mailing as a source of shareware traffic. The reason for this was two factors. Firstly, it is a great substitute for professional targeting. Secondly, the administration of the social network in the fight against spam does only half measures, freezing the pages from which, it comes. Thus, the social network does not fight either the source of spam or its consequences.

At the moment, users who are in group chats with open access for everyone are most often faced with annoying mailing lists. Therefore, effective anti-spam methods are needed to solve this problem.

Keywords: script, algorithm, spam, VK API, PHP, VKontakte.

Для создания алгоритма основным средством взаимодействия с ВКонтакте является VK API. VK API — это интерфейс, который позволяет получать информацию из базы данных vk.com с помощью http-запросов к специальному серверу. В качестве исполнителя разработанного алгоритма будет выступать не страница пользователя, а страница сообщества, так как администрация ВКонтакте приветствует размещение различных чат-ботов в сообществах. Именно по этой причине оттуда была убрана капча для подтверждения того, что все манипуляции выполняются не роботом.

На текущий момент во ВКонтакте присутствуют пользовательские решения данной проблемы в виде чат-ботов, но они обладают рядом недостатков. Во-первых, они приносят не всегда желаемый функционал в беседы (упоминания, игры, статистику, модерацию и т. д.). Во-вторых, эффективность противодействия спаму у каждого подобного решения различна, что всегда приводит к возникновению прорех, через которые спам начинает просачиваться. Главной такой прорехой является ограничение на количество запросов к VK API, в связи с чем, ботам с большой аудиторией приходится находить компромиссы между эффективностью и работоспособностью остального функционала.

Ситуация становится значительно сложнее в связи с текущим устройством VK API. Актуальная на данный момент версия 5.131, как и её предшественники имеют ограничения во всех методах, которые могли бы помочь бороться со спамом. Например, в беседу приходит сообщение, которое можно отметить, как спам, обнаружив в нём соответствующие ключевые слова или ссылки. В данном случае необходимо просто удалить сообщение и исключить пользователя написавшего его с помощью методов messages.delete и messages.removeChatUser соответственно.

Однако есть и другой пример, который не является редкостью. В случае второго варианта идентифицировать спам крайне затруднительно. В беседу также приходит сообщение, но в отличии от первого случая оно не содержит никаких признаков спама и может обладать смысловой нагрузкой. Сразу или в течении нескольких минут автор сообщения редактирует его, добавляя в него ссылки и прочие атрибуты. Таким образом образуется направление, в рамках которого стоит провести работу по обнаружению спама. Первый способ — это отследить сообщение по событию его редактирования, которое можно включить во вкладке “Работа с API” сообщества. Однако, как описывалось ранее всё гораздо сложнее. Данное событие приходит только в личных сообщениях, следовательно, данный способ не подходит. Его отслеживание могло бы стать самым простым и эффективным решением данной задачи. Второй способ — это получение N-ого количества последних сообщений через советующий метод, но данный способ также не подходит из-за ограничений VK API. При вызове данного метода, результат, который он вернёт не будет содержать необходимой информации. Весь необходимый набор данных метод вернёт только в двух случаях: беседа создана сообществом, от имени которого будет

исполняться алгоритм; история сообщений запрашивается из личных сообщений. У методов используемых в двух данных случаях, существуют аналоги, но каждый из таких аналогов повторяет недостатки предыдущих методов. Доподлинно не известна причина существования подобных ограничений, однако наиболее вероятный повод их появления — сохранение конфиденциальности пользователей.

Третий способ — это получение сообщения по его идентификатору. В отличии от предыдущего способа, запрашивать каждое сообщение достаточно трудозатратно с точки зрения вычислительных ресурсов, а главным ограничением выступит лимит на количество запросов. Лимит на количество запросов в данном случае можно обойти, работая в связке с методом execute. Метод выполняет переданный ему VKScript-код, в котором может происходить вызов других API-методов с сохранением и обработкой промежуточных результатов. Язык похож на JavaScript или ActionScript (предполагается совместимость с ECMAScript). Этот подход может быть эффективным для решения проблемы, но в данном материале он не будет рассмотрен подробнее из-за излишнего функционала. Решения поставленной задачи можно достичь, не прибегая к использованию языка программирования VKScript. Это более труднореализуемо, но привносит в алгоритм свои значимые преимущества.

Для реализации проекта был выбран скриптовой язык общего назначения PHP. Благодаря ему взаимодействие с социальной сетью максимально упрощено.

Для хранения параметров взаимодействия с VK API спроектирован класс Config, хранящий соответствующие константы и размещённый в файле config.php. Таким образом, конфигурационный файл будет выглядеть так:

Листинг 1

```
<?php

class Config
{
 // Параметры взаимодействия с VK API
const VERSION = 5.131;
const TOKEN = 'токен сообщества';
const KEY = 'строка подтверждения';
const GROUP_ID = 'идентификатор сообщества';
}
```

Для взаимодействия с VK API спроектирован класс VK и размещён в файле vk.php. Класс VK работает с API в формате Callback API, как и весь последующий код. Существует и другой подход, носящий название LongPoll API. Различие между ними заключается в том, что LongPoll получает информацию о произошедших событиях обращаясь к VK API самостоятельно, а Callback API ожидает пока эта же информация придёт к нему от VK API в виде соответствующего запроса. Для высоконагруженных ботов лучше использовать Callback API. Для простых проектов лучше подойдёт LongPoll API, так как он более дружелюбен к начинающим разработчикам. Класс VK будет использовать константы, заложенные ранее в классе Config, для этого подключим его с помощью require_once. В класс будет включено публичное поле data хранящее в себе информацию о пришедшем событии и следующие публичные методы: call — дополняет входящий в него набор данных, параметрами для аутентификации, собирает и отправляет полученный JSON запрос на сервера ВКонтакте; messages_send — объединяет аргументы необходимые методу VK API используемого для отправки сообщения и после отдаёт их методу call класса VK. Одного метода messages_send для решения поставленной задачи недостаточно, поэтому подобные ему методы, дублирующие его структуру, будут описаны по мере необходимости их включения в алгоритм. Помимо перечисленных методов, класс VK содержит конструктор, выполняющий достаточно важную функцию. При инициализации объекта класса, приходящие данные от ВКонтакте будут записаны в публичное поле data, что позволит в процессе разработки алгоритма оперативно обращаться к этим данным. Ещё одна важная функция конструктора — подтверждение адреса сервера, которое нужно производить единоразово при его подключении к рассылке событий. Таким образом, файл vk.php будет выглядеть следующим образом:

Листинг 2

```
<?php

require_once 'config.php';

class VK
{
public $data;

public function __construct()
 {
$this->data = json_decode(file_get_contents('php://input'),
true);
 if ($this->data['type'] == 'confirmation') exit($this->key);
 else print('ok');
}

 // Отправка JSON
 public function call($method, $params = [])
 {
$params['access_token'] = Config::TOKEN;
$params['v'] = Config::VERSION;
$url = 'https://api.vk.com/method/' . $method . '?' . http_
build_query($params);
$curl = curl_init($url);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
```

```
 $json = curl_exec($curl);
 curl_close($curl);
 return json_decode($json);
 }

 public function messages_send($peer_id, $message)
 {
 return VK::call('messages.send', [
 'random_id' => rand(),
 'peer_id' => $peer_id,
 'message' => $message
 ]);
 }
}
```

Последний класс в данном проекте служит для хранения вспомогательных методов, которые не организуют взаимодействие с VK API, а лишь упрощают повторное использование кода. В класс включён метод для получения строки, построенной таким образом, чтобы при её вставке в текст сообщения, пользователю приходило уведомление о том, что его упоминали по имени и фамилии. Получить имя и фамилию пользователя можно только оснастив класс VK методом users_get, собирающим и объединяющим данные для соответствующего метода VK API. Данный класс будет размещён в файле set_functions.php и будет выглядеть следующим образом:

Листинг 3

```
<?php

class SetFunctions
{
        public $vk;

        public function __construct($vk)
        {
                $this->vk = $vk;
        }

  public function getFullNameWithNotification($user_id): string
 {
 $users_get = $this->vk->users_get($user_id, '');
 return '[id' . $user_id . '|' . $users_get->response[0]->first_name . ' ' . $users_get->response[0]->last_name . ']';
 }
}
```

Первая часть алгоритма размещена в файле index.php и работает с приходящими событиями от ВКонтакте. Сначала в скрипт подключаются файлы классов VK и SetFunctions, сразу же после этого инициализируются их объекты. Создаётся подключение к базе данных. Проверяется тип события, записанный в поле data класса VK. Если это новое сообщение, алгоритм продолжит работу с пришедшими данными, в противном сценарии выполнение скрипта завершится. В первом случае идентификатор беседы, идентификатор пользователя и текст будут записаны в соответствующие переменные для удобного доступа к ним. Предварительно текст будет приведён к нижнему регистру для упрощения поиска ссылок в сообщении. После следует проверка на источник сообщения и на не отрицательность идентификатора отправителя. Проверять источник сообщения необходимо чтобы отличить переписку в беседе, от переписки в личных сообщениях, а не отрицательность идентификатора отправителя для того, чтобы отличить пользователя от сообщества. Если источник беседа, а отправитель сообщения человек, следует из страницы отправителя извлечь дату регистрации страницы. Для этого нужно получить XML содержимое по адресу 'https://vk.com/foaf.php?id=ИДЕНТИФИКАТОР_ПОЛЬЗОВАТЕЛЯ' и извлечь данные между <**ya:created dc:date**=» и «. Полученную информацию можно привести к времени в формате UNIX time с помощью strtotime(). На данном этапе проверяется наличие ссылок в сообщении и дата регистрации страницы ВКонтакте. Любая страница, созданная в течении двух недель, вероятно была создана с целью рассылки спама. Если страница создана в течении двух недель, а полученное сообщение содержит ссылки, то сообщение, содержащее спам будет удалено, а его отправитель исключён из беседы. Также будет отправлено сообщение об обнаружении спама. Чтобы совершить первые два действий необходимо в класс VK добавить методы messages_delete и messages_remove_chat_user, ссылающиеся на соответствующие методы VK API. Если последняя проверка не обнаружила признаков спама, информация о сообщении сохраняется в базе данных.

Для хранения информации о сообщениях потребуется всего одна таблица 'protection' в любой выделенной для реализации проекта базе данных. Структура таблицы выглядит следующим образом:

Столбцы	Тип данных	Описание
ID	int	Номер записи
peer_id	int	Идентификатор источника сообщения (беседы)
user_id	int	Идентификатор пользователя в социальной сети ВКонтакте
conversation_message_id	int	Идентификатор сообщения
Time	int	Время отправки сообщения в формате UNIX time

Файл protection.php содержит вторую часть алгоритма, которая использует таблицу, заполненную первой его частью. Вторая часть алгоритма вызывается с помощью планировщика. Рекомендуемый интервал вы-

полнения скрипта — каждую минуту. Сначала в скрипт подключаются файлы классов VK и SetFunctions, сразу же после этого инициализируются их объекты. Создаётся подключение к базе данных. Из таблицы 'protection' извлекаются все идентификаторы бесед, которые были записаны первой частью алгоритма. Необходимо выбрать только уникальные идентификаторы, поэтому данные в столбце peer_id запрашиваются с помощью оператора DISTINCT. Каждый полученный идентификатор обрабатывается в порядке, описанном далее. Для полученного идентификатора запрашивается список участников с дополнительной информацией — полем deactivated. Это поле определяет, была ли заблокирована страница пользователя. Чтобы запросить список участников необходимо добавить соответствующий метод messages_get_conversation_members в класс VK. Получив список участников, необходимо перебрать его для поиска заблокированных страниц. Для каждой найденной страницы необходимо запросить из таблицы 'protection' все записи содержащие идентификаторы сообщений, отправленных в беседу этим же пользователем. Все полученные идентификаторы нужно объединить в одну строку используя в качестве разделителя запятую. Используя уже спроектированные методы в классе VK, удаляем все сообщения заблокированного пользователя, исключаем его из беседы и отправляем сообщение о его подозрительном поведении. После выполнения перечисленных действий, из таблицы удаляем все записи о найденном пользователе в конкретной беседе и выходим из самого первого цикла перебора. Последним действием после выхода из цикла, является очистка таблицы 'protection' от записей, добавленных более двух часов назад.

Суть второй части алгоритма заключается в поиске участников, заблокированных в социальной сети и удалении их из беседы, а также всех их сообщений отправленных в течении двух последних часов. В основе принципа работы данной части алгоритма лежит механизм борьбы со спамом ВКонтакте. Любая страница, которая участвует в рассылке спама через небольшой промежуток времени подлежит блокировке. Однако ВКонтакте не чистит оставшийся после блокировки пользователя спам. Вторая часть скрипта находя заблокированные страницы занимается чисткой спама.

В данном материале была рассмотрена реализация алгоритма для противодействия спаму в беседах ВКонтакте. Преимуществами данного скрипта можно считать высокую точность в идентификации спама, а также соответствие стандарту оформления кода PSR. Данный алгоритм быстро избавляется от явного спама, однако обладает одним недостатком — для удаления спама в неявных его проявления необходимо дожидаться блокировки страницы со стороны ВКонтакте, что может занять до 20 минут с момента его появления. При наличии специальных навыков по работе с PHP, SQL и базами данных, модификация скрипта не должна быть затруднительной. С кодом проекта можно ознакомиться на GitHub.

ЛИТЕРАТУРА

1. Официальный веб-сайт «В Контакте», раздел «Для разработчиков» [Электронный ресурс]. URL: https://dev.vk.com/reference (дата обращения: 04.02.2022).
2. Официальный веб-сайт «php.net», раздел «Руководство по PHP» [Электронный ресурс]. URL: https://www.php.net/manual/ru/index.php (дата обращения: 04.02.2022).
3. Карпова И.П. Базы данных. — М.: Питер, 2013. С. 3–31.
4. Кузнецов М.В., Симдянов И.В. Самоучитель PHP 7. — СПб.: БХВ-Петербург, 2018. С. 143–151.
5. Джеймс Р. Грофф, Пол Н. Вайнберг, Эндрю Дж. Оппель. SQL. Полное руководство. — Вильямс, 2018. С. 95–148.
6. Официальный веб-сайт «GitHub», репозиторий [Электронный ресурс]. URL: https://github.com/Aiciokizava/algorithm-for-countering-spam-in-VKontakte-conversations

Журнал «Современная наука: актуальные проблемы теории и практики»