электронный журнал
МОЛОДЕЖНЫЙ НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК
Издатель ФГБОУ ВПО "МГТУ им. Н.Э. Баумана". Эл No. ФС77-51038.

УДК. 004.451.7.031.43

Система комплексного определения типа web-клиентов

***Болот Д.В.**, студент*
кафедра «Информационные Системы и Телекоммуникации»
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана

Научный руководитель: Павлов Ю. Н., д.т.н.
Россия, 105005, г. Москва, МГТУ им. Н.Э. Баумана
pavlov@bmstu.ru

Введение

В XXI веке, в интернете приобрели популярность веб-сайты, на которых можно оставлять комментарии (например, форумы и блоги) или те, которые можно свободно редактировать — вики (т.е. когда контент создает сам пользователь, т.н. «Веб 2.0»).

Но в интернете, кроме обычных пользователей, огромное количество роботов. Интернет-роботы уже давно широко используются в полезных и не очень целях, например, роботы-индексаторы анализируют информацию для поисковых систем. Сетевые спам-боты – это также вид интернет-роботов, которые размножают спам-содержимое через весь интернет, в основные, нацеленные на «Веб 2.0» интернет-приложения. Цель спам-робота такова: размещение на сайте ссылки, которая бы указывала на определенный ресурс, web-сервер или же просто на сайты, которые имеют подобную тематическую информацию. Зачастую такие ссылки не нужны. Поэтому со спам-роботами необходимо бороться.

Они грамотно спроектированы так, чтобы копировать человеческое поведение, для того, чтобы пройти проверки системы. Спам не только засоряет полезные ресурсы и заводит пользователей в заблуждение, в результате чего, они попадают на нежелательные веб-сайты (сайты с непосредственно вредоносным содержимым, вирусами), но и путает алгоритмы ранжирования поисковых систем.

Так как эти страницы открыты для свободного редактирования, на них может быть размещён спам — сплоги. [1]. Подобный спам тем более раздражает, что его труднее удалить (как правило, сообщения в форумах и блогах могут редактировать только привилегированные пользователи — соответственно, рядовой участник должен связаться с кем-нибудь из них).

Спам в комментариях преследует две цели: увеличение количества входящих ссылок на продвигаемый сайт (в частности, для повышения Google PR или ТИц Яндекса), а также для увеличения посещаемости продвигаемого сайта (если комментарий оставлен на более популярном ресурсе).

Другой разновидностью спама в блогах является поголовное, часто — автоматическое, внесение пользователей в список «друзей» без знакомства с их персональными страницами, как правило, с целью искусственного повышения собственного рейтинга путем получения «взаимной» дружбы, либо привлечения внимания к сетевой рекламе, размещенной в публичном журнале робота. Такой вид спама иногда называют «френдоспамом» (от англ. *friend* — друг). [1]

Особое внимание стоит обратить на размещение спама на досках объявлений. Обычно на таких досках размещают информацию коммерческого и полукоммерческого содержания. Зачастую спам-боты размещают своё объявление во всех доступных разделах, чтобы увеличить релевантность сообщения в индексируемых страницах поисковыми механизмами. Сами объявления маскируются под коммерческое объявление, иногда каждое слово выглядит как гиперссылка. Иногда в одном объявлении можно увидеть более десятка ссылок, ведущих на разные страницы. Фильтровать доски объявлений с большим количеством объявлений очень трудно, часто срабатывает человеческий фактор — невнимательность.

На сегодняшний день, спам-роботы по уровню технологий, достигают все более высоких вершин. Ведь они каждодневно усовершенствуются и засоряют сеть ненужной информацией и вредными ссылками. Цель спам-робота такова: размещение на сайте ссылки, которая бы указывала на определенный ресурс, web-сервер или же просто на сайты, которые имеют подобную тематическую информацию. Зачастую такие ссылки не нужны. Поэтому со спам-роботами необходимо бороться.

Принцип их работы такой: сначала они, в сети интернета, анализируют сайты, подходящие под их критерии. Далее, спам-роботы выполняют индексацию страницы сайта с помощью определенных форм. Дальше они делают анализ формы сайта и ее защиты. И, соответственно, если защита сайта слабая, спам-робот, успешно добавляет данные сайта в собственную базу данных. Теперь, на сайт или на электронный ящик владельца сайта, постоянно будет приходит спам-сообщения.

Чтобы защитить свой сайт от проникания нежелательной информации необходимо использовать ряд эффективных мер.

Идея моей системы состоит в том, предоставить владельцам сайтов простой (в использовании) способ автоматически определять нежелательных посетителей и не создавая при этом неудобств обычным рядовым пользователям, поскольку, в настоящее время, самым популярным методом определения является – CAPTCHA, одна из форм теста Тьюринга.

Принцип работы

Во время запроса конечным пользователем интернета сайта в интернете, на котором подключена данная система, его запрос будет протестирован ею (схема тестирования видна на Рис. 1), а также ряд последующих запросов, пока его тип не будет определен с достаточной точностью.

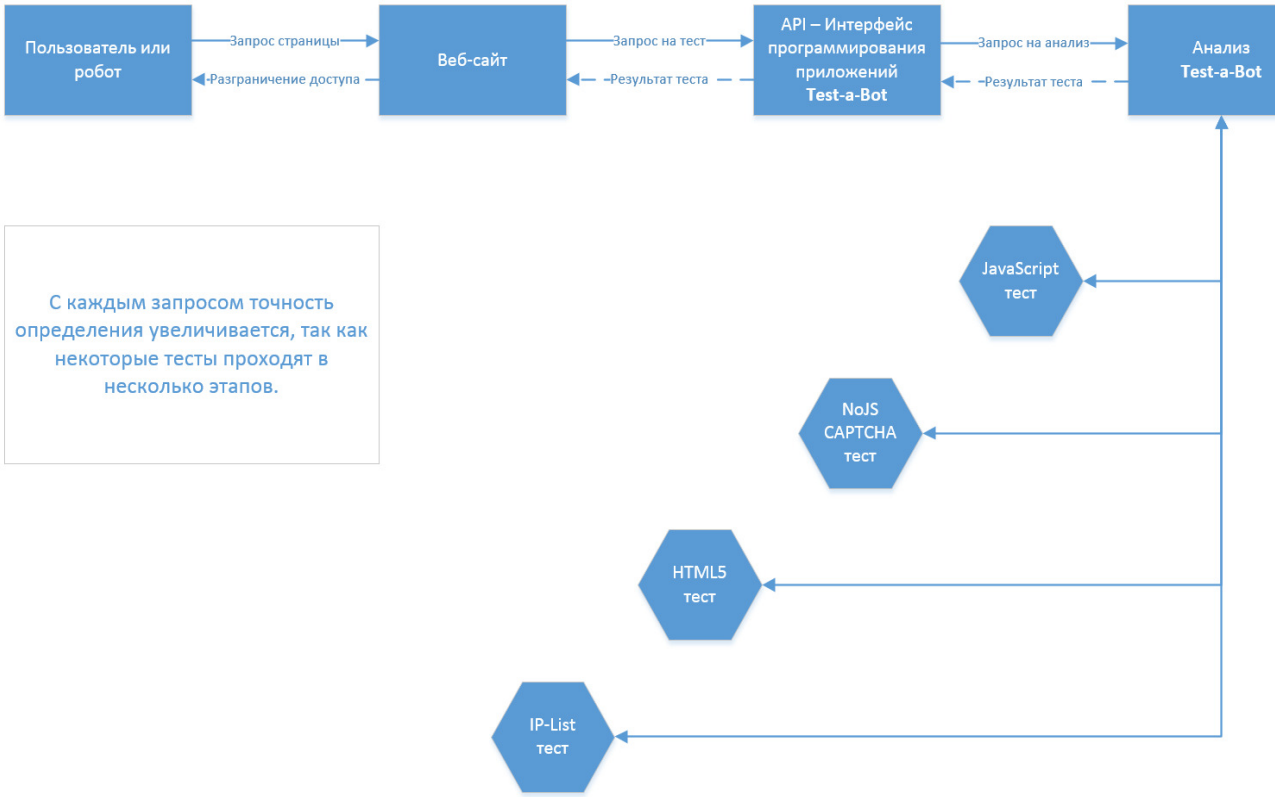

Рис. 1. Схема работы системы рассматриваемого веб-сервиса

Для работы системы, необходимо выделить основные типы клиентов пользователей и создать для них модели поведения, а потом провести эксперимент, используя эти модели и реальные данные.

Например, когда человек запрашивает страницу в интернете (например, http://stackoverflow.com/questions/246859/http-1-0-vs-1-1), то происходит следующий диалог между клиентом и сервером:

> Запрос клиента:

GET /questions/246859/http-1-0-vs-1-1
HTTP/1.1 Host: stackoverflow.com
Connection: keep-alive
Cache-Control: no-cache
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Pragma: no-cache
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/34.0.1847.45 Safari/537.36
Referer: https://www.google.com/
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8,ru;q=0.6

< Ответ сервера:

HTTP/1.1 200 OK
Cache-Control: public, max-age=49 Content-Type: text/html; charset=utf-8
Content-Encoding: gzip
Expires: Fri, 07 Mar 2014 03:34:06 GMT
Last-Modified: Fri, 07 Mar 2014 03:33:06 GMT
ETag: "13c853c-15931-398a179473b80"
Vary: *
X-Frame-Options: SAMEORIGIN
Date: Fri, 07 Mar 2014 03:33:16 GMT
Content-Length: 20412

[... Тело ответа...]

Здесь, отсылая запрос, клиент сообщает, что он еще не имеет никаких сведений о запрашиваемой странице, и оставляет соединение открытым, для более быстрого получения ответа от сервера, который в свою очередь присвоил странице уникальную

метку, а также сообщил, что страница недавно обновилась (за 10 секунд до запроса), и что следующий запрос содержимого имеет смысл делать не менее, чем через 50 секунд. Таким образом еще раз запрашивая эту же страницу, мы скорее всего увидим нечто следующее:

> Запрос клиента:

```
GET /questions/246859/http-1-0-vs-1-1 HTTP/1.1
Host: stackoverflow.com
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 6.2; WOW64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/34.0.1847.45 Safari/537.36
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8,ru;q=0.6
If-None-Match: "13c853c-15931-398a179473b80"
If-Modified-Since: Fri, 07 Mar 2014 03:34:46 GMT
```

< Ответ сервера:

```
HTTP/1.1 304 Not Modified
Date: Fri, 07 Mar 2014 03:35:42 GMT
Vary: *
Connection: Keep-Alive
Keep-Alive: timeout=5, max=512
ETag: "13c853c-15931-398a179473b80"
```

То есть сервер просто ответил нам, что содержимое не устарело и еще раз выдал нам туже метку. Этот механизм интересен нам тем, что если сервер прислал клиенту метку, а клиент, вновь запрашивая эту же страницу (в пределах одной сессии), ее обратно не отправил, то такое поведение можно считать подозрительным.

Таким образом, сервис продолжительно следит за действиями (поведением) клиента, и когда (если) он начнет вести себя слишком подозрительно, система отнесет его к типу зловредных роботов, и используя модифицированный вариант CAPTCHA, подтвердит подтверждать, либо опровергнет свой вердикт.

В конечном счете, появляется возможность объемные и подробные модели, которые будут с высокой долей вероятности определять тип клиента без использования CAPTCHA.

Типы веб-клиентов

Для определения конкретного веб-клиента и соотношения его с поведенческой моделью одного из основных типов, нужно, сначала определить эти типы:

- Индексирующие (поисковые) роботы – собирают информацию в мирных целях
- Рассылающие роботы (спам-боты) – рассылают информацию с недоброжелательными целями (реклама или мошенничество)
- Веб-браузер актуальной версии, управляемый живым человеком – самый обычный посетитель с обновленным браузером
- Устаревший веб-браузер, управляемый человеком, без поддержки ряда функционала (выключен JavaScript, картинки, Cookies…)
- Ненормальное поведение, сканнеры уязвимостей, специальные программы для скачивания сайтов и автономного просмотра или исследователь, который, увлекшись обратной инженерией выполнил действие, которое не предусмотрено моделью нормального поведения человека.

Для начала, нам нужно найти способ следить за конкретным пользователем. То есть, научиться выделять его из сотен и тысяч подобных. И на основе этого собирать подробную информацию о поведении известных клиентов: пользователей и поисковых роботов (которые, предоставляют возможность идентификации).

Типы данных и идентификационная фраза.

Для осуществления распознавания, компьютерной программе нужны данные. Чем больше - тем лучше (тем точнее результат). Главные критерии - чтобы данные были полезные (т.е. имеющие смысл), были интерпретируемы, могли храниться, могли быть обработаны и являлись не очевидными (не могли быть выведены из существующих данных). Последнее особенно важно, потому что в противном случае данные являются подтверждающими, но непригодными для использования в задачи распознавания.

Например, знание о том, что браузер использует протокол HTTP для доступа к ресурсам, очевидно, является неким примером тривиальности и не несет пользы. Гораздо более интересно - какие именно заголовки отсылает браузер при запросе (Request Headers). Они отличаются, и они могут быть полезны.

Все данные (вектор значений) делятся на 2 большие группы:

• Установленные нами (скриптами сайта)

• Существующие до нас (до входа на сайт) и не имеющие отношения к деятельности нашего сайта

Принцип в установленных нами значениях лежит в генерировании сложной, мало повторимой и трудно подделываемой фразы, которая сохраняется на браузере пользователя. В дальнейшем, сайт, получая доступ к хранилищу, ищет фразу и при совпадении идентифицирует клиента. Малая повторяемость влияет на частоту и ложность срабатывания. Защита от подделки препятствует выдачи браузера за чужого, путем подделки чужой фразы.

Существующие до нас данные должны обладать теми же характеристиками, что и установленные нами. Тут существуют 2 проблемы - потенциально высокая повторяемость и возможная подделка. Например, в среднем браузер Google Chrome установлен на 46.02 процентов компьютеров и это миллиарды браузеров.

Обычно, используются установленные нами значения, чем уже существующие. Но в общем случае структура получения идентификационной фразы (ключа) выглядит как на рис. 1. Выходящие данные (идентификационная фраза) - строка символов, точно определяющая выбранный браузер и никакой иной. В 90 процентов сайтов идентификационная фраза совпадает, например, с cookies (см. далее), но это необязательно.

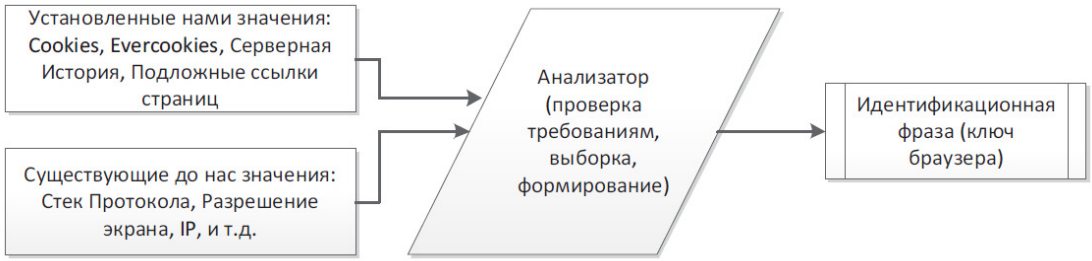

Рис. 2. Комбинирование данных для получения идентификационной фразы (ключа)

Основное требование к идентификационной фразе - полнота (точно характеризующая браузер) и максимальная сжатость (возможность использования её для ключа к СУБД, например).

Схемы получения данных

Cookies

Ку́ки (от англ. cookie 2 печенье) 2 небольшой фрагмент данных, отправленный веб-сервером и хранимый на компьютере пользователя. Веб-клиент (обычно веб-браузер) всякий раз при попытке открыть страницу соответствующего сайта пересылает этот фрагмент данных веб-серверу в виде HTTP-запроса. [4]. Применяется для сохранения данных на стороне пользователя, на практике обычно используется для:

• аутентификации пользователя;

• хранения персональных предпочтений и настроек пользователя;

• отслеживания состояния сессии доступа пользователя;

• ведения статистики о пользователях. [4].

Преимуществом является простота установки. Для установки не требуется поддержки JavaScript в браузере, т.к. это может быть спокойно осуществлено средствами языка PHP. Недостатками являются широкая известность и распространенность, поэтому существует огромное кол-во инструментов для модифицирования cookies или их очистки.

От изменения есть простое и достаточно эффективное решение - сохранять на сервере все выданные cookies, а в качестве cookie использовать длинную и специально подготовленную строку. В идеале - со сложным обратным преобразованием (например, md5). А для правильного изменения требуется расшифровать строку, что может быть трудной задачей.

Evercookies

Это идея cookies, с возможностью оставлять данные в местах, куда позволяет сохранить браузер. Повсеместное распространение программ для очистки и модификации cookies привели к поиску мест сохранение устанавливаемой фразы, остальной принцип действия остается таким же.

Были определены следующие места сохранения:

- Стандартные HTTP Cookies (как дубль).
- Local Shared Objects (Flash Cookies)
- Silverlight Isolated Storage
- cookies в RGB цветах PNGs
- Размещение cookies в Web History
- Размещение cookies в HTTP ETags
- Размещение cookies в Web cache

- window.name
- Internet Explorer userData storage
- HTML5 Session Storage
- HTML5 Local Storage
- HTML5 Global Storage
- HTML5 Database Storage

IP-адрес

IP (Сетевой адрес компьютера) известен на сервере при обращении к нему браузера. Это последний хост, перед нашим сервером. Его невозможно подделать. Использование IP для определения пользователя имеет очень существенные недостатки, и не будет работать (или будет работать плохо) для машин, которые имеют внутренний IP адрес, а выходят через шлюз с одним внешним IP. В данном случае будет определяться именно внешний IP или IP шлюза. Использование TOR или прокси серверов аналогично изменит IP и сделает привязку бессмысленной.

Несмотря на недостатки, IP привязка широко используется, например, для целей безопасности: при обращении с незнакомого IP, система может запросить дополнительное подтверждение.

Цифровой отпечаток браузера

Цифровой отпечаток браузера - это набор данных, которые мы можем получить, описывающий браузер. Примерами являются: Разрешение экрана, User-Agent, Заголовки HTTP_ACCEPT, Данные об установленных плагинах, Установленные Шрифты. Собрав их воедино, мы получаем цифровой отпечаток браузера, который может быть очень популярным или очень редким, стоит лишь пользователю установить редкий шрифт или добавить мало популярные плагины. Цифровой отпечаток - существующие до нас характеристики. Они могут меняться (например, при добавлении шрифта), но в целом, набор является более-менее фиксированным.

Дата и время посещения

Вероятность посещения того же самого ресурса в короткий срок (1-2 дня), выше чем при длинном промежутке времени. (Зависит от ресурса).

Пример и применение:
Если неизвестные пользователи с редким отпечатком браузера зашли в достаточно короткий промежуток времени несколько раз, то, скорее всего, пользователь один.

Использование времени может быть эффективными при точном понимании специфики обращений к ресурсу.

Кэш файлы

Интересный механизм, который работает следующим образом: формируется специальный файл, который помечается как кэшируемый, чтобы браузер его внес в кэш. При повторном доступе на сайт, браузер не будет считывать файл, потому что он есть в кэше, что и будет служить признаком повторного появления. Еще один вариант - страница может содержать ссылку <script type=9text/javascript9 src=9example.js9>. С загрузкой страницы загружается и example.js. Затем скрипт кэшируется и не требует загрузки при повторном посещении страницы. В результате, если скрипт содержит такое значение, как id=3243242, этот идентификатор остаётся в силе и может быть использован другим сценарием JavaScript или другой страницей, запрашивающей этот скрипт.

Это явный пример данных, устанавливаемый нами. По сути, он работает как cookie (evercookie), только технология доступа кардинально отличается.

Данные из ранее посещенных страниц.

При обращении к странице некоторые браузеры при отправке запроса указывают адрес предыдущей страницы в поле HTTP_REFERER (PHP). Его можно легко получить на сервере с помощью PHP: $_SERVER['HTTP_REFERER'];

Способ активно используется SEO-мастерами при раскрутке сайта и анализе его эффективности, т.к. это значение указывает на предыдущую страницу, откуда пришел пользователь. И может содержать поисковой запрос, к которому поисковая система «прикрепила» страницу.

Сравнение и применимость

Таблица 1

Сравнение

Схема	Требования	Преимущества	Недостатки	Комментарий
Cookie	Браузер должен принимать cookie	Работает при выключенном JS и при отсутствии Adobe Flash	Широко распространенная технология, и как следствие существует огромное кол-во программ для удаления	Пользователи могут изменять cookie, поэтому рекомендуется их генерировать максимально зашифрованным способом
Evercookie	Для каждого метода свой. Для некоторых нужен JS, для некоторых Flash, некоторые работают без дополнительного ПО	Огромное число мест хранения и более низкие шансы для полного удаления	Для некоторых методов требуется Flash или JavaScript	
IP-адрес	Нет	Работает всегда (IP всегда можно получить)	Ввиду огромного кол-ва аномайзеров: TOR, прокси-серверов — неточный метод определения.	
Цифровой отпечаток браузера	Требуется JavaScript и Flash.	В случае уникальности какого-то параметра, отпечаток может быть достаточно точным	Если рабочие станции идентичные, то определить точно какую-нибудь маловероятно.	
Дата и время посещения	Нет	Легко получить.	Не несет особой информации	Можно использовать, как дополнительный метод
Кэш файлы	Нет	Не требует дополнительных	Неточен, не работает после	

		настроек, разрешения на запуск или ПО.	очистки кэша, при переключении на иной браузер и т.д.	
Данные о ранее посещенных страницах	JavaScript		Работает очень редко и данные малополезны	Можно использовать, как дополнительный метод

Математический расчет необходимого объема данных для метода цифрового отпечатка.

На основании данных, полученных Electronic Frontier Foundation, приводим математическую базу. Допустим, у нас есть алгоритм получения идентификационной фразы (ключа), основанной на одном или группе методов описанных выше.

Примем F такое, что при появлении x новых браузеров (новых версий), вероятностная функция P имеет следующий вид:

$$P(f_n), n \in [0,1,..,N]$$

Положим названием уникальность (или сюрприз) выход следующей функции:

$$I(F(x) = f_n) = -log_2(P(f_n))$$

Уникальность I, измеряется в битах (т.к. мы прологарифмировали уникальность отпечатка по основанию 2).

Энтропия распределения P(fn) - это ожидаемое значение уникальности по всем браузерам:

$$H(F) = -\sum_{n=0}^{N} P(f_n) log_2(P(f_n))$$

Уникальность (или удивление) имеет следующий физический смысл: это кол-во информации об объекте, над которым применятся функция цифрового отпечатка, где каждый бит вдвое уменьшает кол-во вариантов (т.к. мы применили логарифм). Если веб-сайт регулярно посещают X различных браузеров, то конкретный браузер $x \in X$, будет точно распознан, если:

$$I\big(F(x)\big) \geqq log_2|X|$$

Обобщенный алгоритм действий для идентификации пользователя

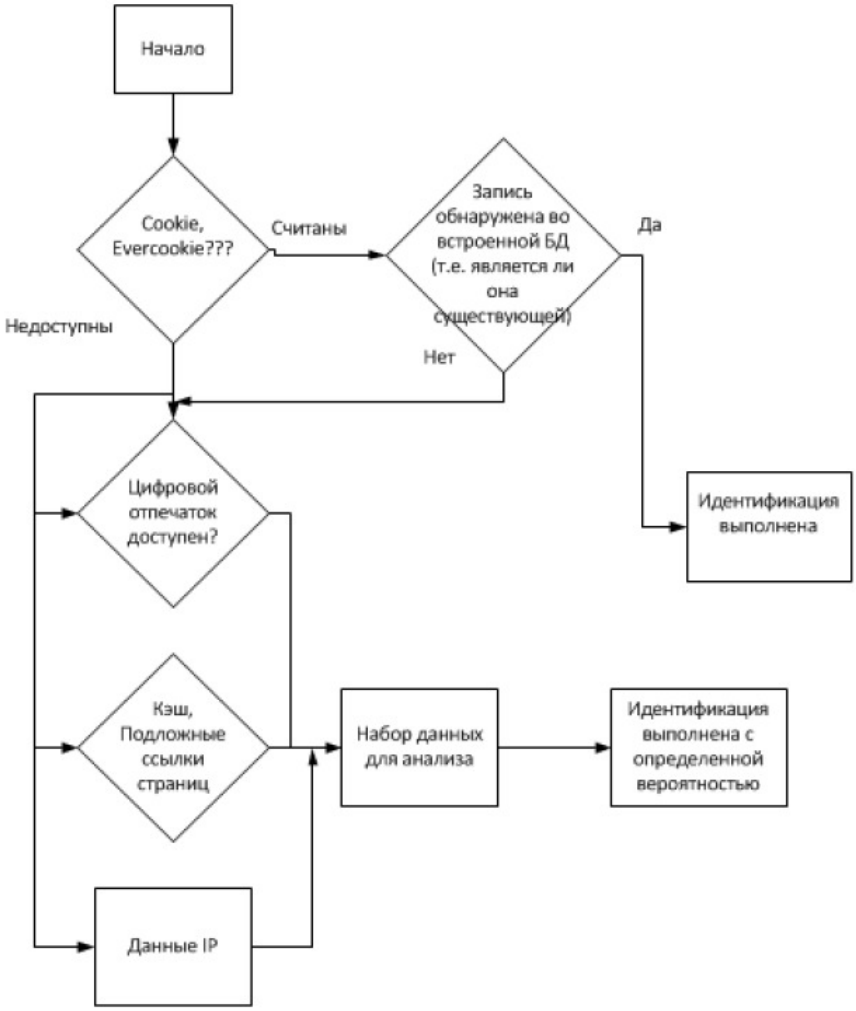

Рис. 3. Обобщенный алгоритм действий

Тестирование

Теперь, имея «на крючке» нашего пользователя клиента мы, наконец-то, имеем возможность протестировать его. Для чего используется следующий набор тестов:

Обнаружение человеческой активности (события JavaScript)

Под человеческой активностью, здесь понимаются события типа движения мышкой или нажатия клавиш на клавиатуре, потому что, в большинстве спам-ботов, навигация и взаимодействие с сервисом происходит на более низком программно-интерфейсном уровне, в результате чего такие события не будут зафиксированы сервисом.

Еще один приятный момент состоит в том, что мы получаем возможность проследить пути веб-клиента на сайте и составить граф его перемещения внутри сайта.

Ловушки (honeypot)

В коде сервиса содержатся невидимые для живых пользователей ссылки, перейдя по которым веб-клиент вполне явно сообщает нам, что им оперирует машина или какой-нибудь хакер, производящий анализ кода.

Белый список (white list)

Пользуясь тем, что большинство крупных поисковых сервисов не держат свои IP-адреса в секрете, а некоторые особо крупные, еще и предоставляют методы проверки принадлежности клиента к поисковым роботам соответствующих сервисов, была создана и постоянно обновляется база «белых» адресов роботов на основе параллельных обратных запросов DNS.

Опять Тьюринг (CAPTCHA)

Достоверно известно, что спам-роботы часто пытаются маскироваться под устаревшие версии браузеров. В данный момент ситуация такова, что у всех вменяемых пользователей интернета стоит последняя версия браузера, так как у всех крупных игроков этого рынка уже имеется система автоматического обновления, которая по умолчанию включена. По этому, без дополнительного теста в данном случае будет не обойтись. Так как мы предполагаем, что у веб-клиента минимальные возможности отображения, и не хотим быть уязвимыми для сервисов автоматического распознавания подобного рода тестов, то данный тест реализован в виде ASCII-графики.

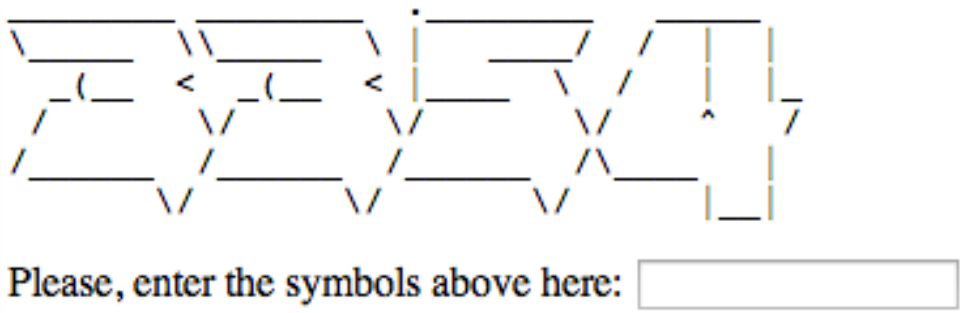

Рис. 4. Пример модифицированного теста CAPTCHA

Результаты эксперимента и вывод

Для проведения эксперимента, я тестировал посетителей ресурса cocucok.net с ноября 2013 года по февраль 2014, результаты которого, приведены в таблице 2.

Таблица 2

Результаты эксперимента

Тест	Кол-во клиентов	Процент (%)
Запросы изображений	165 364	32.9
Выполнение JavaScript	133 219	26.5
Обнаружены движения мышью	102 482	20.4
Пройден тест CAPTCHA	89 161	17.7
Переходы по скрытым ссылкам	12 454	2.5
Всего	502 680	100

Из результатов эксперименты мы можем получить количество «человеческих» сессий: $S_{ЧЕЛ} = (S_{ЗИ} \cup S_{ДМ}) - (S_{JS} - S_{ДМ}) = 26{,}8\%$

В силу специфики ресурса, данный показатель, можно считать достоверным с определенной точностью. И это, только начало работы моей системы. Мой опыт, говорит мне, что скорость и точность определения можно повысить, если применить нейронные сети, и я обязательно постараюсь реализовать данную идею, для того, чтобы еще больше повысить возможность идентификации клиентов.

Список литературы

1. Спам. Режим доступа: http://ru.wikipedia.org/wiki/Спам (дата обращения: 10.01.2014).
2. Обзор способов распознавания посетителей веб ресурсов на основе получаемой или оставляемой ими информации / Цатурян Т. Ш., Гапанюк Ю. Е. // Молодежный научно-технический вестник. МГТУ им. Н.Э. Баумана. Электронный журнал. Режим доступа: http://sntbul.bmstu.ru/doc/688991.html (дата обращения 23.12.2013).
3. HTTP/1.0. Stackoverflow. Режим доступа: http://stackoverflow.com/questions/246859/http-1-0-vs-1-1 (дата обращения 14.01.2014).

4. HTTP Cookie. Википедия. Режим доступа: http://ru.wikipedia.org/wiki/HTTP_cookie (дата обращения 19.01.2014).
5. How Unique Is Your Web Browser? Electronic Frontier Foundation. Режим доступа: http://dl.acm.org/citation.cfm?id=1881152 (дата обращения 19.01.2014).
6. Securing Web Service by Automatic Robot Detection / KyoungSoo Park, Vivek S. Pai, Kang-Won Lee, Seraphin Calo // Annual Tech '06: 2006 USENIX Annual Technical Conference. Режим доступа: https://www.usenix.org/legacy/event/usenix06/tech/full_papers/park/park_html/paper.html (дата обращения: 22.01.2014).
7. Balachander Krishnamurthy. Key Differences between HTTP/1.0 and HTTP/1.1. Mogul Jeffrey C., Kristol David M. Режим доступа: http://www8.org/w8-papers/5c-protocols/key/key.html (дата обращения 23.01.2014).
8. Как проверить, что робот принадлежит Яндексу // help.yandex.ru: Яндекс. Режим доступа: http://help.yandex.ru/webmaster/robot-workings/check-yandex-robots.xml (дата обращения: 05.02.2014).
9. Роботы рунета // robotstxt.org.ru: RobotsTxtRu. Режим доступа: http://robotstxt.org.ru/rurobots (дата обращения 12.02.2014).
10. Robots Database // robotstxt.org: RobotsTxt. Режим доступа: http://www.robotstxt.org/db.html (дата обращения 02.02.2014).