Разработка веб-фильтра для контроля доступа к веб-ресурсам

Д.А. Бизин[1], С.А. Бурлов[1]

[1]Самарский национальный исследовательский университет им. академика С.П. Королева, Московское шоссе 34А, Самара, Россия, 443086

Аннотация. В результате данной работы был спроектирован и разработан веб прокси-сервер для анализа и контроля веб-трафика. Прокси-сервер классифицирует веб-контент и принимает решение об ограничении доступа. В качестве алгоритма классификации контента используется наивный байесовский классификатор. Также реализованы такие функциональности, как черный список URL/IP и веб-консоль для управления веб-фильтром. Прокси-сервер позволяет фильтровать как HTTP, так и HTTPS трафик. Для HTTPS используется механизм создания поддельного сертификата Trusted MITM. Серверная часть написана на Java. Клиентская часть написана на JavaScript с использованием библиотеки Bootstrap.

1. Введение

В последнее время в интернете все больше появляется противозаконного контента, ориентированного на школьников и подростков. Блокирование таких веб-ресурсов по IP-адресу не приносит особого результата, т.к. в основном эти сайты создаются сотнями каждый день, и требуется время для их обнаружения и внесения в черный список. В такой ситуации фильтрация на основании содержимого сайта будет давать наилучший результат. В связи с этим тема фильтрации (классификации) веб-контента очень актуальна в образовательных учреждениях. Также с целью соблюдения политики информационной безопасности на предприятиях можно использовать веб-фильтр для ограничения доступа сотрудников к нежелательному контенту (например, социальные сети).

Веб-фильтр (контент-фильтр) – это программное или программно-аппаратное устройство, которое ограничивает доступ к веб-сайтам на основании анализа их содержимого. Кроме анализа содержимого сайтов контент-фильтры также могут ограничивать доступ по IP-адресу веб-ресурса (черные списки сайтов).

В данной работе будет рассмотрено проектирование программного веб-фильтра, предназначенного для контроля доступа к веб-ресурсам на основании классификации контента в режиме реального времени при помощи наивного байсовского классификатора.

По своей сути контент-фильтр – это веб прокси-сервер, который перехватывает HTTP/HTTPS-запросы и анализирует их. Упрощенный алгоритм работы веб-фильтра по шагам:

1. Перехват HTTP/HTTPS-запроса пользователя к интернет-ресурсу.
2. Анализ запроса (проверка IP-адреса интернет-ресурса, анализ самого HTTP/HTTPS-запроса).

3. На основании анализа принять решение: позволить осуществить запрос к интернет-ресурсу или отклонить его.
4. Если было принято решение об отклонении запроса, то производится возврат пользователю информации об отклонении (например, веб-страница). Если же было решено пропустить запрос, то веб-фильтр делает запрос к ресурсу.
5. Перехват ответа от интернет-ресурса.
6. Анализ ответа (анализ возвращенного контента, классификация контента).
7. На основании анализа контента принять решение: пропустить ответ от ресурса пользователю или отклонить его.
8. Если было принято решение об отклонении ответа, то производится возврат пользователю информации об отклонении (например, веб-страница). Если же было решено пропустить ответ, то веб-фильтр возвращает ответ от интернет-ресурса пользователю.

На рисунке 1 показана схема работы веб прокси-сервера.

Рисунок 1. Схема работы веб-фильтра.

В приведенном выше алгоритме умалчивается один факт: контент, передаваемый по протоколу HTTPS, зашифрован. Поэтому анализировать его без расшифрования не представляется возможным. Это существенное ограничение функциональности контент-фильтра, т.к. на данный момент в интернете сайты, работающие по протоколу HTTPS, преобладают над сайтами, работающими по незащищенному протоколу HTTP. Поэтому для того, чтобы веб-фильтр работал максимально эффективно, нужно расшифровывать контент с использованием поддельного сертификата и применением технологии Trusted MITM [1].

2. Trusted MITM

Для максимальной эффективности работы веб-фильтра следует расшифровывать и анализировать HTTPS-трафик. Для этого предлагается использовать технологию Trusted MITM, суть которой заключается в динамической генерации поддельных сертификатов на стороне веб прокси-сервера. Trusted в данном случае означает, что пользователь знает о применении подхода MITM и согласен на это.

Схема использования технологии Trusted MITM описана ниже:

1. Перехват веб-фильтром запроса браузера к веб-серверу по протоколу HTTPS.
2. Прокси-сервер генерирует ключевую пару (открытый и закрытый ключи) и сертификат для запрашиваемого браузером доменного имени (в поля Common Name и Subject Alternative Name сертификата проставляется доменное имя). Сгенерированный сертификат подписывается доверенным корневым сертификатом (которому доверяет браузер). Веб-фильтр отправляет сгенерированный сертификат клиенту.
3. Браузер проверяет сгенерированный сертификат веб-фильтра. В случае прохождения проверок, браузер генерирует симметричный ключ, шифрует его открытым ключом из сертификата прокси-сервера и отправляет обратно.
4. Веб-фильтр расшифровывает симметричный ключ, используя закрытую часть ключа, и сохраняет его.
5. Веб прокси-сервер делает запрос из шага 1 к веб-серверу по протоколу HTTPS.
6. Веб-сервер отправляет прокси-серверу свой сертификат.

7. Веб-фильтр проверяет сертификат и, в случае прохождения проверок, генерирует симметричный ключ, зашифровывает его открытым ключом из сертификата сервера и отправляет обратно веб-серверу.
8. Веб-сервер расшифровывает симметричный ключ, используя закрытую часть, и отправляет прокси-серверу контент, зашифрованный симметричным ключом.
9. Веб-фильтр расшифровывает контент при помощи симметричного ключа и затем шифрует сохраненный контент симметричным ключом, сохраненным на шаге 4, и отправляет его клиенту.
10. Браузер расшифровывает контент при помощи симметричного ключа и отображает его.

Описанная выше схема по шагам показана на рисунке 2.

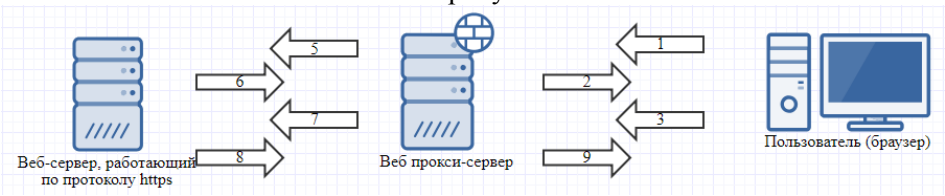

Рисунок 2. Схема работы веб-фильтра с HTTPS-трафиком.

3. Наивный байесовский классификатор

Формально постановку задачи классификации можно определить следующим образом [2]. Пусть имеются множество документов $D=\{d_i\}$ и множество заранее заданных категорий (классов) $C=\{c_j\}$. Неизвестная целевая функция $\Phi: D\times C\to\{0,1\}$ задается формулой (1):

$$\Phi(d_i,c_j)=\begin{cases}0, \text{ если } d_i\notin c_j\\ 1, \text{ если } d_i\in c_j\end{cases} \quad (1)$$

Требуется построить классификатор Φ' максимально близкий к Φ.

Если классификатор определяется формулой (1), то он называется точным. Если же классификатор возвращает значение из диапазона $[0,1]$ (вероятность попадания документа d_i в категорию c_j), то он называется вероятностным.

На этапе предобработки текста происходит удаление не значащих для классификации слов (союзы, предлоги, частицы) и стемминг. Стемминг – это процесс нахождения основы слова. Таким образом, каждое слово будет заменено на свою основу. В результате проведенных выше операций значительно сокращается размерность пространства признаков (слов), что необходимо для дальнейшей классификации.

Наивный байесовский классификатор [3] является одним из алгоритмов вероятностной классификации. Относится к категории алгоритмов машинного обучения с учителем.

Пусть дан документ d, представленный в виде вектора слов $d=\{t_1,t_2,...,t_n\}$. Также пусть заранее определено множество категорий $C=\{c_i\}$. Задача классификатора заключается в том, чтобы подобрать такие значения d и c_i, при которых значение $P(c_i\,|\,d)$ – вероятность того, что документ d принадлежит категории c_i – будет максимальным, т.е. необходимо найти (2):

$$\arg\left(\max_{c_i}\left(P(c_i\,|\,d)\right)\right) \quad (2)$$

Для вычисления значений $P(c_i\,|\,d)$ используется формула Байеса (3):

$$P(c_i\,|\,d)=\frac{P(c_i)\cdot P(d\,|\,c_i)}{P(d)}, \quad (3)$$

где $P(c_i)$ – априорная вероятность того, что документ отнесен к категории c_i; $P(d\,|\,c_i)$ – вероятность найти документ d в категории c_i; $P(d)$ – вероятность того, что документ d можно представить в виде вектора признаков (слов) $d=\{t_1,t_2,...,t_n\}$.

$P(c_i)$ является отношением количества документов из обучающей выборки, отнесенных к категории c_i, к общему числу документов. $P(d)$ не зависит от категории c_i, поэтому это значение константное и не влияет на выбор максимального из значений $P(c_i \mid d)$.

Т.к. в большинстве своем документ содержит большое количество слов (признаков), то вычисление значения $P(d \mid c_i)$ затруднительно. Поэтому делается «наивное» предположение о том, что любые два слова из документа d статистически не зависят друг от друга (два независимых события). Тогда для вычисления $P(d \mid c_i)$ можно воспользоваться формулой (4):

$$P(d \mid c_i) = \prod_{j=1}^{n} P(t_j \mid c_i) \quad (4)$$

К преимуществам байесовского классификатора относят высокую скорость работы, простую программную реализацию алгоритма и легкую интерпретируемость результатов работы алгоритма. К недостаткам относят низкое качество классификации и неспособность учитывать зависимость между признаками (словами). Но на практике байесовский классификатор показывает высокое качество классификации, и этому есть объяснения [4]:

- слова в документе в большинстве своем зависимы, но эта зависимость одинакова для разных классов и «взаимно сокращается» при оценке вероятностей;
- пусть на самом деле $P(x = v_0 \mid D) = 0{,}51$ и $P(x = v_1 \mid D) = 0{,}49$, а байесовский классификатор выдаст, что $P(x = v_0 \mid D) = 0{,}99$ и $P(x = v_1 \mid D) = 0{,}01$, но результат классификации от этого не изменится.

Существуют две основные модели [4], которые используются при реализации наивного байесовского классификатора и которые дают разные результаты: многомерная модель и мультиномиальная модель. В реализации веб-фильтра используется смешанная модель, описанная в [5], с применением сглаживания по Лапласу для разрешения проблемы неизвестных слов.

4. Результаты

Веб-фильтр написан на языке программирования Java. Данный язык был выбран по нескольким причинам: удобный API для работы с сетью; запуск программ на любых платформах (ОС), для которых реализована JVM; безопасность среды исполнения (JRE) программ на Java; достаточная простота разработки и надежность программ; поддержка сообществом.

Веб-фильтр обрабатывает проходящие через него запросы на уровне протоколов TCP/IP при помощи программных сокетов. Полученные байты преобразуются в сообщения протокола HTTP [6], и происходит работа с заголовками HTTP протокола. В частности, используются заголовки: Content-Type (для определения типа тела сообщения), Host (для определения запрашиваемого хоста), Content-Length (для определения длины тела сообщения) и Transfer-Encoding (для определения способа кодирования, примененного к телу сообщения для передачи). Функционал черных списков URL/IP использует данные из заголовков Host. Остальные три заголовка используются для извлечения тела сообщения и проверки его формата на принадлежность HTML.

После получения HTML-страницы происходит удаление всех тегов при помощи библиотеки Jsoup версии 1.11.2 [7]. Затем полученный список слов подвергается процедуре стемматизации при помощи библиотеки Apache Lucene Morphology для русского языка последней версии. Также удаляются все слова, длина которых меньше 4 символов. После чего обработанный список слов передается алгоритму классификации. Алгоритм возвращает ассоциативный массив категория -> вероятность попадания данного текста в эту категорию.

После получения результатов работы алгоритма можно принимать решение, запрещать ли данный контент. Например, можно использовать некоторое пороговое значение: если какая-то вероятность выше этого порогового значения, то нужно запретить контент. Либо можно просто записывать полученную информацию в журнал событий.

Веб-фильтр работает в многопоточном режиме, т.е. каждый запрос пользователя обрабатывается в отдельном потоке. Количество потоков, а также другие настройки можно задавать в веб-консоли управления. Веб-консоль разработана по архитектуре REST [8] (в частности, CRUD операции используют соответствующие методы HTTP). На сервере используется технология Java Servlets [9] для обработки HTTP-запросов. В качестве веб-сервера используется библиотека Jetty [10] версии 9.4.11.v20180605. Front-end написан с использованием JavaScript, библиотеки jQuery (в частности, используется технология AJAX) [11] версии 3.3.1 и библиотеки Bootstrap версии 4 [12]. Данные настроек хранятся в БД H2 [13] версии 1.4.197, которая работает в embeeded-режиме (т.е. БД представляет собой файл). Все операции веб-фильтра логируются. Файлы логов ежедневно ротируются в отдельные файлы с указанием даты в названии. Для генерации сертификатов формата X.509 и управления процедурой обмена рукопожатиями в протоколе TLS используется криптографическая библиотека Bouncy Castle версии 1.49 [14]. При генерации сертификатов происходит их кэширование для ускорения последующих обращений к этому же хосту.
Веб-фильтр можно устанавливать как на конечном компьютере, так и на шлюзе. В обоих случаях требуется указать IP-адрес компьютера, на который установлен веб-фильтр, в качестве IP-адреса прокси-сервера. Порты задаются в настройках веб-фильтра.
На рисунке 3 показана страница настроек веб-фильтра в веб-консоли управления.

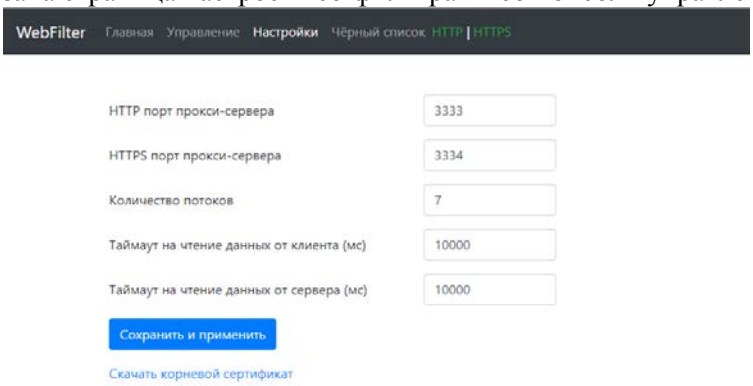

Рисунок 3. Страница настроек веб-фильтра в веб-консоли управления.

На рисунке 4 показан результат фильтрации сайта (в страницу сайта встраивается JavaScript-код, который показывает всплывающее окно с информацией).

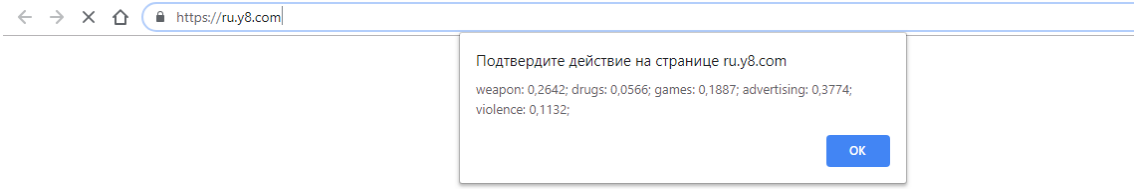

Рисунок 4. Результат фильтрации игрового сайта.

Если же поместить данный хост в черный список, то при следующем открытии сайта выйдет информация об этом как показано на рисунке 5.

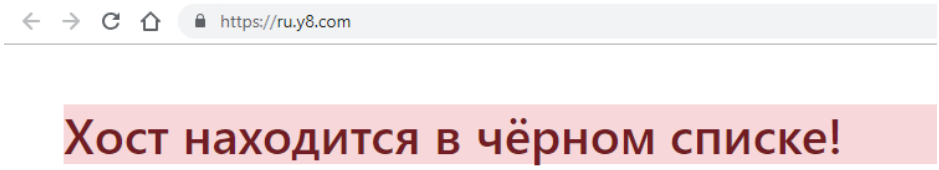

Рисунок 5. Результат работы функционала черных списков URL/IP.

5. Выводы

В результате проделанной работы был реализован программный продукт – веб-фильтр для контроля доступа к веб-ресурсам. Данное решение может быть использовано, например, в

образовательных учреждениях с целью ограждения несовершеннолетних от нежелательной информации в интернете.

К недостаткам разработанного решения можно отнести фильтрацию только русскоязычного контента. Также в качестве рекомендации по дальнейшему улучшению функциональности веб-фильтра стоит отметить добавление других алгоритмов классификации, которые не были рассмотрены в данной работе.

6. Литература

[1] Пискунов, И. Перехват и расшифровка HTTPS трафика [Электронный ресурс]. – Режим доступа: https://ipiskunov.blogspot.com/2016/06/https.html (18.11.2018).

[2] Батура, Т.В. Методы автоматической классификации текстов // Программные продукты и системы. – 2017. – Т. 30, № 1. – С. 85-99. DOI: 10.15827/0236-235X.030.1.085-099.

[3] Сизов, А. Наивный байесовский классификатор / А. Сизов, С. Николенко [Электронный ресурс]. – Режим доступа: https://logic.pdmi.ras.ru/~sergey/teaching/mlstc12/sem01-naivebayes.pdf (18.11.2018).

[4] Николенко, С. Байесовские классификаторы [Электронный ресурс]. – Режим доступа: https://logic.pdmi.ras.ru/~sergey/teaching/mlaptu11/03-classifiers.pdf (18.11.2018).

[5] Баженов, Д. Наивный байесовский классификатор [Электронный ресурс]. – Режим доступа: http://bazhenov.me/blog/2012/06/11/naive-bayes (18.11.2018).

[6] Hypertext Transfer Protocol -- HTTP/1.1 [Электронный ресурс]. – Режим доступа: https://tools.ietf.org/html/rfc2616#page-31 (18.11.2018).

[7] Библиотека Jsoup [Электронный ресурс]. – Режим доступа: https://jsoup.org/ (18.11.2018).

[8] Архитектура REST [Электронный ресурс]. – Режим доступа: https://habr.com/post/38730/ (18.11.2018).

[9] Java Servlets Technology [Электронный ресурс]. – Режим доступа: https://www.oracle.com/technetwork/java/index-jsp-135475.html (18.11.2018).

[10] Библиотека Jetty [Электронный ресурс]. – Режим доступа: http://www.eclipse.org/jetty/ (18.11.2018).

[11] Библиотека jQuery [Электронный ресурс]. – Режим доступа: https://jquery.com/ (18.11.2018).

[12] Библиотека Bootstrap 4 [Электронный ресурс]. – Режим доступа: https://getbootstrap.com/docs/4.0/getting-started/introduction/ (18.11.2018).

[13] База данных H2 [Электронный ресурс]. – Режим доступа: http://www.h2database.com/html/main.html (18.11.2018).

[14] Библиотека Bouncy Castle [Электронный ресурс]. – Режим доступа: http://www.bouncycastle.org/ (18.11.2018).

Development of a web filter to control access to web resources

D.A. Bizin[1], S.A. Burlov[1]

[1]Samara National Research University, Moskovskoe Shosse 34A, Samara, Russia, 443086

Abstract. Web proxy server was developed for analysis and control of web traffic as result of this work. Proxy server classifies web content and makes a decision to restrict access. A naive Bayesian classifier is used as an algorithm for classifying contents. Also implemented the functionalities such as black list of URLs/IPs and web console to manage the server. Proxy server allows you to filter as HTTP well as HTTPS traffic. For HTTPS uses the mechanism of creating fake certificate known as Trusted MITM. The server side is written in Java, and the client side is written in JavaScript.