**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

|  |  |
| --- | --- |
|  | **ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ**  **Кафедра**  **«Защита информации»** |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к научно-исследовательской работе**

**«Исследование возможности выявления иностранных зависимостей в интернет-ресурсах российских информационных систем»**

|  |  |  |
| --- | --- | --- |
| Исполнитель:  Студент гр. Б19-525 |  | Швецова А.А. |
| Научный руководитель: | подпись, дата | Мельников Д.А. |
| Руков. образов. программы: | подпись, дата | Пудовкина М.А. |
|  | подпись, дата |  |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Москва – 2022**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**РЕФЕРАТ**

Отчёт 20 стр., 6 рис., 1 табл., 6 источников

АНАЛИЗ ИНТЕРНЕТ-РЕСУРСОВ, ИНОСТРАННЫЕ ЗАВИСИМОСТИ, ИЗВЛЕЧЕНИЕ ДАННЫХ С ВЕБ-СТРАНИЦ

Объект исследования – интернет-ресурсы российских информационных систем.

Цель работы – выявление иностранных зависимостей в интернет-ресурсах российских информационных систем.

В ходе работы были определены основные технологии, использующиеся в создании и при работе интернет-ресурсов, была определена методика выявления иностранных зависимостей интернет-ресурсов российских информационных систем на основе анализа их программного кода, были проанализированы распространённые инструменты для достижения цели исследования.

Получен алгоритм, позволяющий выявлять иностранные зависимости в интернет-ресурсах, содержащих статические веб-страницы. Найдены иностранные зависимости в некоторых российских интернет-ресурсах.

В дальнейшем планируется разработка методики выявления иностранных зависимостей в интернет-ресурсах, основанных на динамическом отображении веб-страниц.

**СОДЕРЖАНИЕ**

ВВЕДЕНИЕ 5

1. Разработка алгоритма выявления иностранных зависимостей в интернет-ресурсах российских информационных систем 10

1.1. Веб-технологии, используемые в работе интернет-ресурсов 10

1.2. Определение требований к алгоритму выявления иностранных зависимостей 13

2. Разработка приложения, реализующего алгоритм выявления иностранных зависимостей в интернет-ресурсах 13

ЗАКЛЮЧЕНИЕ 13

Список использованных источников 15

**ОПРЕДЕЛЕНИЯ, ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ**

СПО – свободное программное обеспечение.

ЯП – язык программирования.

ПО – программное обеспечение.

# ВВЕДЕНИЕ

В настоящее время свободное программное обеспечение активно используется во множестве крупных проектов, таких как: операционная система Linux, браузер Mozilla Firefox, поисковые системы Google и Yandex, системы управления базами данных PostgreSQL и MariaDB, система управления содержимым сайтов WordPress и т. д. Идея свободного распространения исходного кода программ возникла с появлением компьютеров, которые сначала использовались в качестве инструментов для проведения исследований, а исходный код программ, написанных в результате исследований, мог свободно распространяться без какой-либо лицензии [1]. Со временем компьютеры стали использоваться и для создания коммерческого программного обеспечения, нелегальное использование которого может пресекаться с помощью программно-аппаратных, юридических и других мер защиты. В 1985 году разработчик Ричард Столлман основал Фонд Свободного Программного Обеспечения (англ. – Free Software Foundation) с целью продвижения идеи свободного программного обеспечения и, в частности, развития проекта GNU [2]. Для достижения этих целей он разработал GNU General Public License (GNU GPL) – лицензию на свободное программное обеспечение, опубликованную в 1989 году, согласно которой, все пользователи должны иметь следующие права, или «свободы»:

* свободу запуска программы с любой целью;
* свободу изучения того, как программа работает, и её модификации (предварительным условием для этого является доступ к исходному коду);
* свободу распространения копий как исходного, так и исполняемого кода;
* свободу улучшения программы и выпуска улучшений в публичный доступ (предварительным условием для этого является доступ к исходному коду).

Р. Столлман разработал такие примеры свободного программного обеспечения, как GNU C Compiler и GNU Emacs и вдохновил многих других разработчиков на проекты со свободным программным обеспечением. Один из первых таких проектов – операционная система GNU/Linux Debian, создание которой было начато в 1993 году и которая популярна и по сей день.

Однако понятие «free software» в английском языке непрозрачно: оно может означать не только «свободный», но и «бесплатный». Согласно второму значению этого слова проекты, которые распространяются на бесплатной основе, но скрывают исходный код своего программного обеспечения, могут быть так же названы «free software», что противоречит определению этого понятия. С целью решения этой проблемы, было введено новое понятие «open source», которое однозначно описывает идею распространения исходного кода программного обеспечения. Продвижению идеи открытого программного обеспечения посвящена организация Open Source Initiative, основанная Брюсом Перенсом и Эриком Реймондом. Однако, чтобы программное обеспечение могло называться «открытым», его лицензия должна удовлетворять несколько другим критериям:

1. Лицензия не должна ограничивать любую сторону от продажи или раздачи программного обеспечения как компонента совокупного распространения, содержащего программное обеспечение из разных источников.
2. Программа должна включать исходный код и должна разрешать распространение как в исходном коде, так и в скомпилированном виде. В тех случаях, когда какая-либо форма продукта не распространяется с исходным кодом, должны существовать широко разрекламированные средства бесплатной загрузки исходного кода через Интернет. Исходный код должен быть предпочтительной формой, в которой программист мог бы модифицировать программу. Намеренно запутанный исходный код не допускается. Промежуточные формы, такие как выходные данные препроцессора или транслятора, не допускаются.
3. Лицензия должна разрешать модификации и производные работы и должна разрешать их распространение на тех же условиях, что и лицензия на оригинальное программное обеспечение.
4. Лицензия может ограничить распространение исходного кода в измененном виде только в том случае, если лицензия разрешает распространение «файлов исправлений» с исходным кодом с целью модификации программы во время сборки.
5. Лицензия не должна дискриминировать какое-либо лицо или группу лиц.
6. Лицензия не должна ограничивать кого-либо в использовании программного обеспечения в определенной области деятельности.
7. Права, связанные с программой, должны распространяться на всех, кому распространяется программа, без необходимости получения дополнительной лицензии.
8. Права, прилагаемые к программе, не должны зависеть от того, является ли программа частью конкретного распространения программного обеспечения. Если программа извлекается из этого распространения и используется или распространяется в соответствии с условиями лицензии на программу, все стороны, которым распространяется программа, должны иметь те же права, что и те, которые предоставляются вместе с исходным распространением программного обеспечения.
9. Лицензия не должна налагать ограничений на другое программное обеспечение, распространяемое вместе с лицензионным программным обеспечением.

Б. Перенс и Э. Реймонд определяют не только то, каким должно быть СПО, но и то, как оно должно распространяться. Также они приводят лицензию GPL в качестве примера, какой должна быть лицензия свободного программного обеспечения.

Основное преимущество свободного программного обеспечения заключается в том, что улучшать, разрабатывать такое программное обеспечение и находить ошибки в нём может огромное количество людей, что обеспечивает более динамическое развитие и своевременное исправление всяческих недочётов и уязвимостей.

На сегодняшний день СПО начинает вытеснять проприетарное программное обеспечение, о чём свидетельствуют ежегодные отчёты компании Red Hat «The State of Enterprise Open Source», в котором приведены результаты опроса 1296 лидеров ИТ-компаний из разных регионов мира [3]. Согласно отчёту, 55% респондентов используют свободное программное обеспечение, и исследователи ожидают, что это число в будущем будет только увеличиваться.

В России похожее исследование провела компания Accenture при поддержке фонда «Сколково» в сентябре 2021 [4]. По результатам этого исследования 100% опрошенных компаний-разработчиков и 85% опрошенных компаний-пользователей используют Open Source решения в направлении ИТ-инфраструктуры. Также высок процент использования в области ИТ-поддержки – 83% и 59% соответственно и в сфере аналитики данных – 17% и 44%. А к 2026 году по прогнозам Accenture свободное программное обеспечение будут использовать более 90% компаний.

Однако несмотря на преимущества Open Source, существуют и негативные аспекты использования свободного программного обеспечения в России. Во-первых, это возможное внедрение в такие проекты вредоносного кода, функции которого могут варьироваться от несанкционированного размещения баннеров с политическими призывами и провокационным содержанием до вирусов-шифровальщиков. Часто проекты с открытым исходным кодом используются без выполнения анализа на наличие уязвимостей и вредоносного программного обеспечения. Во-вторых, в связи со сложившейся политической ситуацией, платформы для размещения проектов со свободным программным обеспечением, такие как GitHub, начали блокировать доступ к платформе для российских пользователей и компаний, в частности, государственных структур, системообразующих коммерческих компаний и предприятий, а разработчики стали блокировать использование своих проектов. В связи с этим многие российские проекты, зависимые от иностранного свободного программного обеспечения, в том числе интернет-ресурсы российских информационных систем, могут нестабильно выполнять свою работу или даже внезапно прекратить её выполнение. Выявление таких зависимостей, в частности, в интернет-ресурсах, может позволить заблаговременно избавиться от них и предотвратить вышеуказанные нежелательные последствия.

# Разработка алгоритма выявления иностранных зависимостей в интернет-ресурсах российских информационных систем

## Веб-технологии, используемые в работе интернет-ресурсов

Веб-приложение – это программное обеспечение, доступ к которому пользователь получает по Всемирной компьютерной сети Интернет. Работа таких приложений осуществляется по принципу «клиент-сервер»: пользователь использует клиент для передачи запросов по сети серверу, а сервер динамически генерирует ответ [5]. За счёт наличия исполняемой части, веб-приложения способны выполнять практически те же операции, что и обычные приложения, работающие под управлением операционной системы (ОС), лишь с тем отличием, что программный код исполняется на сервере, в качестве интерфейса часто выступает браузер, а средой, посредством которой происходит обмен данными, является Интернет. Наиболее распространёнными операциями, свойственными веб-приложениям, в частности, являются:

* Приём данных от пользователя и хранение их на сервере.
* Выполнение различных действий по запросу пользователя: извлечение данных из базы данных, добавление, удаление, изменение данных в базе данных, проведение сложных вычислений и других действий.
* Аутентификация пользователя и отображение интерфейса системы, соответствующего данному пользователю.
* Отображение постоянно изменяющейся актуальной информации.

Схема работы веб-приложения изображена на Рисунке 1.

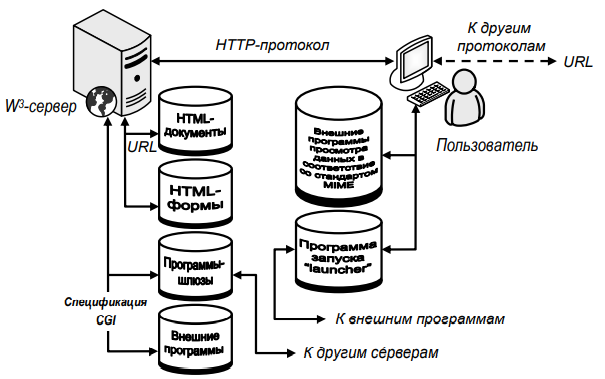


Рисунок 1 – Архитектура взаимодействия программного обеспечения в WWW-системе

Таким образом, структурными компонентами веб-приложения являются клиентская и серверная стороны [6]. Клиентский компонент может быть разработан с использованием языков программирования CSS, HTML, JavaScript или TypeScript. Серверный компонент может быть создан с помощью языков программирования C# Java, Python, Node.JS, Ruby, PHP, C/C++, Python, Perl и др. Сервер состоит из двух частей – логики приложения и базы данных. Логика приложения – это центр управления веб-приложением. База данных отвечает за хранение информации.

Можно выделить следующие уровни логики веб-приложений:

1. Уровень представления. Отображает пользовательский интерфейс и упрощает взаимодействие с пользователем. Уровень представления имеет компоненты пользовательского интерфейса, которые визуализируют и показывают данные для пользователей. Также существуют компоненты пользовательского процесса, которые задают взаимодействие с пользователем. Предоставляет всю необходимую информацию клиентской стороне. Основная цель уровня представления - получить входные данные, обработать запросы пользователей, отправить их в службу данных и показать результаты.
2. Уровень обслуживания данных. Передаёт данные, обработанные уровнем бизнес-логики, на уровень представления. Этот уровень гарантирует безопасность данных, изолируя бизнес-логику со стороны клиента.
3. Уровень бизнес-логики. Несёт ответственность за надлежащий обмен данными, определяет логику бизнес-операций и правил.
4. Уровень доступа к данным. Предлагает упрощённый доступ к данным, хранящимся в постоянных хранилищах, таких как двоичные файлы и файлы XML, а также управляет операциями CRUD – создание, чтение, обновление, удаление.

В зависимости от того, как логика приложения распределяется между клиентской и серверной частями, можно выделить несколько типов архитектуры веб-приложений:

* Одностраничное приложение или SPA (Single Page Application) – приложение, которое загружает всю необходимую информацию при загрузке веб-страницы. Одностраничные приложения имеют одно существенное преимущество перед другими типами архитектур – они обеспечивают весьма удобный пользовательский интерфейс, так как пользователи не нуждаются в перезагрузке веб-страниц. Наиболее часто разрабатываются с использованием фреймворков, написанных на ЯП JavaScript, таких как Angular, React и др.
* Многостраничные веб-приложения более популярны в Интернете. Такая архитектура является предпочтительнее в тех случаях, когда объём требуемой для размещения информации достаточно велик.
* Архитектура микросервисов направлена на взаимодействие насколько это возможно небольших, слабо связанных и легко изменяемых модулей – микросервисов. Такой тип архитектуры стал распространён в середине 2010-х годов в связи с развитием практик гибкой разработки и DevOps. Если в традиционных вариантах сервис-ориентированной архитектуры модули могут быть сами по себе достаточно сложными программными системами, а взаимодействие между ними зачастую полагается на стандартизованные тяжеловесные протоколы (такие, как SOAP, XML-RPC), в микросервисной архитектуре системы выстраиваются из компонентов, выполняющих относительно элементарные функции, и взаимодействующие с использованием экономичных сетевых коммуникационных протоколов (в стиле REST с использованием, например, JSON, Protocol Buffers, Thrift). За счёт повышения гранулярности модулей архитектура нацелена на уменьшение степени зацепления и увеличение связности, что позволяет проще добавлять и изменять функции в системе в любое время.
* Бессерверная архитектура позволяет создавать и запускать приложения и сервисы без необходимости управления инфраструктурой. Такие приложения основаны на облачных вычислениях.

Прогрессивные веб-приложения используют совокупность определённых технологий: Service Worker, Web App manifest, App shell, Push Notifications для достижения целевых показателей надёжности, быстроты и привлекательности для пользователя.

## Определение требований к алгоритму выявления иностранных зависимостей

Под иностранной зависимостью подразумевается, что это какой-либо элемент веб-страницы, загружаемый с внешнего источника. В данном случае, представляют интерес те источники, которые являются иностранными интернет-ресурсами. Также могут данные зависимости могут отличаться по степени угрозы работоспособности сайта. По ссылке, ведущей на иностранную веб-страницу, могут загружаться изображения, аудио- или иные файлы, отсутствие которых на веб-странице не понесёт негативных последствий для его работы. Также это могут быть программные модули, написанные на ЯП JavaScript или TypeScript, которые могут быть ответственны за выполнение важных функций, обеспечивающих работоспособность интернет-ресурса и при блокировке доступа к которым она может быть нарушена. Не стоит исключать и возможность наличия в данных программных модулях вредоносного ПО, что не скажется на обеспечении работоспособности интернет-ресурса, но понесёт негативные последствия для его посетителей.

Для данного исследования интерес представляют иностранные зависимости, являющиеся именно программными модулями.

Загружаемые программные модули можно обнаружить по соответствующим тегам *<script>* в программном коде веб-страницы. Они могут находиться и локально на сервере веб-приложения, такие программные модули, очевидно, не могут являться иностранными зависимостями и должны отсеиваться. Также программные модули могут быть полностью вписаны в сам HTML-код, в этом случае в теге *<script>* должен отсутствовать атрибут *src*, определяющий источник программного модуля. Соответствующие примеры программных модулей изображены на Рисунке 2, Рисунке 3 и Рисунке 4.



Рисунок 2 – Хранящийся локально на сервере веб-приложения программный модуль



Рисунок 3 – Загружаемый с внешнего источника программный модуль

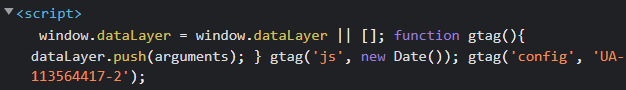


Рисунок 4 – Встроенный в HTML-код веб-страницы программный модуль

Требуемый алгоритм должен выявлять ссылки именно тех программных модулей, которые загружаются из внешних источников. Для этого сначала должен быть получен доступ к файлам, содержащим HTML-код страниц сайта посредством HTTP-запроса к серверу интернет-ресурса. При этом должны быть получены все веб-страницы данного интернет-ресурса, это может быть сделано путём рекурсивных переходов от главной веб-страницы интернет-ресурса по тегу *<a>.* Но данный тег может содержать также ссылки на различные файлы, хранящиеся на сервере интернет-ресурса, что приводит к дополнительным условиям отбора местонахождения данных ресурсов.

При получении программного кода веб-страницы она должна быть просканирована на наличие тегов *<script>*, удовлетворяющих заявленным выше требованиям, а из данного тега должен извлекаться атрибут *src*, ответственный за наличие ссылки, ведущей на источник программного модуля.

Полученные таким образом ссылки анализируются посредством выявления страны регистратора домена интернет-ресурса. Информация о регистраторе домена может быть получена из WHOIS-серверов. WHOIS – это сетевой протокол, ответственный за получение информации о регистраторе доменов. Также при этом стоит учесть, что информация о регистраторах российских интернет-ресурсов не содержат информации о стране регистратора, но в названии регистратора содержится префикс «RU». Но несмотря на это, страна администратора домена может быть уже иной, соответственно, это поле тоже должно проверяться. Также существует возможность для администраторов доменов скрыть данные о соответствующем домене, но тем не менее страна регистратора оставляется доступной, как показано на Рисунке 5.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 5 – Предоставленная WHOIS-сервером информация о домене

В соответствии с изложенными выше требованиями был разработан алгоритм, представленный на Рисунке 6

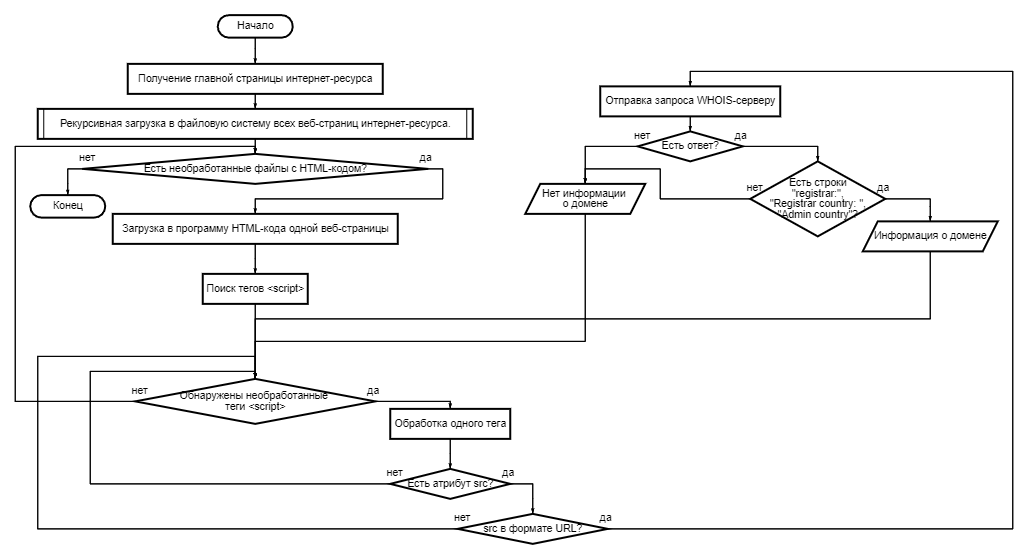


Рисунок 6 – Блок-схема алгоритма выявления иностранных зависимостей

# Разработка приложения, реализующего алгоритм выявления иностранных зависимостей в интернет-ресурсах

Существует множество онлайн-инструментов и приложений, позволяющих извлекать данные с интернет-ресурсов, таких как Scraper, Spinn3r, VisualScraper, Webhouse.io и др., но все такие инструменты не подразумевают анализ источников программных модулей на принадлежность регистратора к той или иной стране, а особенно проверяющих, что эта страна не какая-либо конкретная. Также обработка динамических веб-страниц зачастую не реализована, хотя такие веб-страницы также являются важным объектом данного исследования. В связи с этим сделан вывод о необходимости разработки собственного программного решения данной задачи.

В качестве ЯП был выбран JavaScript на платформе Node.js, так как это позволяет писать как клиентские, так и серверные приложения, что необходимо для реализации алгоритма приложения. Также для данной платформы создано множество библиотек, программных модулей и иных программных платформ, предназначенных для работы с интернет-ресурсами. Одним из таких программных модулей является website-scraper, позволяющий рекурсивно загружать веб-страницы, выбирать, какие объекты нужно загружать с помощью селекторов, ставить фильтры на загружаемые объекты, а также поддерживает расширение функционала с помощью встраиваемых программных модулей. Именно этот инструмент был выбран для рекурсивной загрузки веб-страниц в файловую систему.

Программный модуль website-scraper поддерживает следующие нужные для исследования опции загрузки:

* Рекурсивная загрузка файлов до определённой глубины.
* Фильтр адресов загружаемых веб-страниц.
* Селекторы для определения тегов и атрибутов загружаемых объектов.
* Встраиваемые модули, использующиеся при загрузке веб-страниц.

Данный инструмент позволяет реализовать рекурсивную загрузку веб-страниц, являющихся частью домена главной страницы сайта. Но при этом остаются нереализованными извлечение источников программных модулей и отсеивание веб-страниц, являющихся не HTML-файлами.

Для реализации данных задач был разработан соответствующий встраиваемый программный модуль. После получения ответа от сервера интернет-ресурса и до загрузки объекта в файловую систему данный модуль проверяет, что данный объект является HTML-страницей. После загрузки объекта в файловую систему модуль загружает HTML-файл в программу, а затем анализирует его на наличие тегов *<script>* с помощью программной библиотеки cheerio, с помощью которой извлекается из HTML-кода массив объектов, соответствующих искомым тегам. Проверяется, есть ли атрибуты *src* и что они формата URL. Далее посылается запрос серверу WHOIS, соответствующему домену источника. Это реализовано с помощью программного модуля whois, содержащего адреса серверов сетевого протокола WHOIS, соответствующих конкретным доменным зонам. После этого ответ сервера анализируется на предмет наличия строк, содержащих искомую информацию о регистраторе. Результат анализа выводится в консольный интерфейс.

# ЗАКЛЮЧЕНИЕ

Были разработаны методика выявления иностранных зависимостей в интернет-ресурсах российских информационных систем и алгоритм выявления данных зависимостей для статических веб-страниц. Были проанализированы несколько российских интернет-ресурсов и выявлены зависимости, представленные в

Таблица 1 – Обнаруженные иностранные зависимости

|  |  |  |
| --- | --- | --- |
| Название интернет-ресурса | Источник зависимости | Страна регистратора источника |
| Astralinux.ru | Google.com | US |
| Googletagmanager.com | US |
| Pochta.ru | bootstrapcdn.com | IS |
| Jquery.com | US |
| jsdelivr.net | IS |
| googleapis.com | US |
| Pochtabank.ru | googleoptimize.com | US |

Данные результаты были получены при анализе статических веб-страниц, но есть множество сайтов, которые основаны на динамической генерации веб-страниц, и алгоритм извлечения данных с таких сайтов планируется разработать в дальнейшем. Но полученные результаты на данном этапе исследования подчёркивают его актуальность и необходимость развития.

# Приложение А. Исходный код основной программы выявления иностранных зависимостей

import scrape from 'website-scraper'

import ExternalDependencies from "./ext-deps.js"

if (process.argv.length !== 4) {

console.log("Error! Enter address of website and directory for saving files.");

process.exit(-1)

}

const site = process.argv[2]

global.dir = process.argv[3]

global.links = new Set()

const options = {

urls: [site],

urlFilter: function(url) {

return url.indexOf(site) === 0; },

directory: dir,

sources: { selector: 'a', attr: 'href' },

recursive: true,

maxRecursiveDepth: 2,

maxDepth: 2,

plugins: [ new ExternalDependencies() ]

}

try {

const result = await scrape(options);

}

catch (error) {

console.log("Error!")

}

# Список использованных источников

1. Perens, Bruce. (1998). Perens Open Source Definition LG #26. Linux Gazette.
2. М. Брауде-Золотарев, Г. Гребнев, П. Протасов, А. Ралько, Е. Сербина. Свободное программное обеспечение в организации. Сборник материалов / М. Брауде-Золотарев. — 3-е. — М.: «Интернет-Полиграфия» INFO-FOSS.RU, 22.09.2008. — 124 с. — 1000 экз. — ISBN 978-5-903423-03-3 УДК: 681.3.06 ББК: 32.973.26.
3. The State of Enterprise Open Source Report // Red Hat URL: https://www.redhat.com/en/resources/state-of-enterprise-open-source-report-2022 (дата обращения: 12.06.2022).
4. Исследование потенциала технологий Open Source // Accenture URL: https://www.accenture.com/\_acnmedia/PDF-173/Accenture-Open-Source-Research.pdf (дата обращения: 12.06.2022).
5. Кулаков, Кирилл Александрович. K90 Архитектура и фреймворки веб-приложений: учебное электронное пособие / К. А. Кулаков, В. М. Димитров; М-во науки и высшего образования Рос. Федерации, Федер. гос. бюджет. образоват. учреждение высш. образования Петрозавод. гос. ун-т. — Электрон. дан. — Петрозаводск: Издательство ПетрГУ, 2020.
6. Петрова А. Г., Кириллин Д. В., Бускаров В.В. Архитектура веб-приложений // Актуальные научные исследования в современном мире. 2021. № 10–13 (78). С. 233–237.