Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра автоматизации систем и вычислительных комплексов
Лаборатория безопасности информационных систем

Павлов Дмитрий Сергеевич

Исследование безопасности Javascript-плагинов для сбора статистики

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Научный руководитель:
младший научный сотрудник
А.А. Петухов

Москва, 2017

Аннотация

Данная работа посвящена исследованию популярных Javascript-плагинов для сбора статистики, какие возможности предоставляют плагины и что из этого может извлечь потенциальный атакующий.

Приведена собранная статистика по использованию плагинов на страницах входа в интернет-банки.

Содержание

1 Введение 4

1.1 Javascript-плагины ... 4
1.2 Схема взаимодействия ... 4
1.3 Модель атакующего ... 5
1.4 Ограничения взаимодействия плагинов ... 5
1.5 Способы подключения плагинов ... 6
1.6 Варианты повышения защищенности приложения с плагинами ... 9
1.7 Цель работы ... 11
1.8 Постановка задачи ... 11
1.9 Актуальность ... 11

2 Обзор популярных плагинов 12

2.1 Google Analytics ... 12
2.2 Яндекс.Метрика ... 14
2.3 Рейтинг Mail.Ru ... 16
2.4 Google Tag Manager ... 17
2.5 Угрозы безопасности ... 18

3 Реализация инструмента 19

3.1 Архитектура инструмента ... 19
3.2 Детали реализации ... 20
3.3 Обоснование выбора инструментария ... 21

4 Эксперимент 23

4.1 Методика проведения эксперимента ... 23
4.2 Результаты эксперимента ... 24

5 Результаты 25

6 Приложение 26

7 Список литературы 27

1 Введение

Основную долю клиентов веб-приложения получают через поисковые системы, поэтому владельцы приложений уделяют большое внимание продвижению своих ресурсов в результатах поисковых выдач. Поисковой оптимизацией занимаются SEO-специалисты, которые с помощью комплекса мер поднимают сайт в рейтинге поисковых систем. Одним из инструментов, отслеживающих источники переходов на сайты приложений и поведение пользователей на самих сайтах, являются Javascript-плагины для сбора статистики, которые помогают SEO-специалистам проводить внутреннюю оптимизацию приложения[1].

1.1 Javascript-плагины

Javascript-плагин - программный компонент динамически подключаемый к основной программе предназначенный для расширения и/или использования её возможностей. Javascript-плагины, используемые в веб-приложениях, можно разделить на плагины от стороннего разработчика и внутренние. В то же время, сторонние можно разделить на типы: плагины пользовательского интерфейса и для сбора статистики.

Условимся в дальнейшем под плагином понимать Javascript-плагин для сбора статистики.

1.2 Схема взаимодействия

Рассмотрим схему взаимодействий клиента и веб-приложения с подключаемым сторонним Javascript-плагином. В схеме есть три активных субъекта:

- Владелец веб-приложения - контролирует веб-приложение (администрирует сервер веб-приложения, может вносить изменения в HTML-страницы возвращаемые клиентам).
- Клиент - пользователь веб-приложения, использующий веб-браузер для доступа к функциям приложения через веб-интерфейс, который отображается браузером в результате обработки HTML-страниц, полученных от сервера.

[1] К внутренней оптимизации относится повышение качества сайта, удобства работы с ним.

- Владелец Javascript-плагина - сторонний субъект, который на тех или иных условиях предоставляет Javascript-плагин владельцу веб-приложения для размещения его на HTML-страницах его приложения.

Javascript-код плагина в зависимости от условий предоставления Javascript-плагина может либо контролироваться владельцем приложения, так и быть вне зоны его ответственности. В последнем случае плагин может являться слабой точкой веб-приложения с точки зрения его защищенности.

1.3 Модель атакующего

Атакующий - субъект, который пытается нарушить целостность приложения, либо заполучить данные, которые ему недоступны, используя недостатки системы.

Цель атакующего - либо получение конфиденциальной информации пользователей, обрабатываемой приложением, либо отправка запросов к веб-приложению от лица клиентов приложения с целью выполнения от их имени тех или иных действий.

Атакующий пытается достичь своей цели, используя недостатки приложения, которые появляются на разных этапах жизни программного обеспечения (проектирование, кодирование, эксплуатация). В данной работе будет рассматриваться только использование недостатка этапа эксплуатации приложения, которое связано с использованием Javascript-плагинов для сбора статистики. При таких ограничениях для достижения цели требуется выполнить Javascript-код в браузере клиента, который считает данные из DOM[2] или сформирует и отправит AJAX[3] запрос из браузера клиента приложению.

1.4 Ограничения взаимодействия плагинов

Браузеры имеют встроенный механизм безопасности для изолирования потенциально вредоносных документов, который называют „Same-origin policy” [1] (Кросс-доменные ограничения). Данный механизм контролирует взаимодействие между ресурсами из различных источников. Рассмотрим вкратце данную технологию. Два URL

---

[2] DOM (от англ. Document Object Model — «объектная модель документа») - интерфейс независимый от платформы и языка, который позволяет программам и скриптам динамически получать доступ и обновлять данные, структуру и стиль документа.[2]

[3] AJAX (Asynchronous Javascript and XML - асинхронный Javascript и XMl) - подход к построению интерактивных пользовательских интерфейсов веб-приложения, когда обмен данными с сервером происходит в фоновом режиме.

считаются имеющими один и тот же источник, если у них одинаковые протокол, домен и порт. В Таблице 1 приведены результаты проверки Same-origin policy для URL „http://store.company.com/dir/page.html".

Таблица 1

URL	Результат	Причина
http://store.company.com/dir2/other.html	Успех	
http://store.company.com/dir/inner/another.html	Успех	
https://store.company.com/secure.html	Ошибка	Разные протоколы
http://store.company.com:81/dir/etc.html	Ошибка	Разные порты
http://news.company.com/dir/other.html	Ошибка	Разные хосты

1.5 Способы подключения плагинов

Рассмотрим три основных способа подключения плагинов [3].

1.5.1 Вставка кода

Самый простой способ заключается во встраивании кода в HTML-страницы, отправляемые сервером приложения (предполагаем в данном случае, что код плагина самодостаточен и не подгружает дополнительных ресурсов, иначе можно считать его упрощенным вариантом менеджера плагинов - см. пункт 1.5.3). Пример встраивания такого типа плагина приведен в Листинге 1.

```
<!-- Некоторый хост, например example.com, HTML код -->
<html>
  <head></head>
    <body>
      ...
      <script type="text/javascript">/* Код стороннего плагина
*/</script>
  </body>
</html>
```

Листинг 1

В данном случае после подключения плагина он переходит в зону ответственности разработчика веб-приложения, и чтобы приложение оставалось защищенным с точки зрения использования плагина, требуется проверить код плагина на отсутствие недостатков таких, как XSS [4] и/или утечки данных на сторонний сервер. При таком подключении плагина Javascript-код становится частью HTML-страниц, возвращаемых

сервером, и исполняется в контексте домена веб-приложения, поэтому не изолируется встроенным механизмом безопасности Same-origin policy, и ответственность за возможное снижение защищенности приложения лежит полностью на владельце приложения. Если код плагина обфусцирован, то проверка становится сложной задачей. Популярные плагины, такие как Google Analytics и Яндекс Метрика, поставляются в обфусцированном виде.

1.5.2 Подключение по URL

Альтернативным вариантом подключения плагина является запрашивание кода с хоста разработчика каждый раз, когда требуется его подключить, с помощью конструкции приведенной в Листинге 2.

```
<!-- Некоторый хост, например example.com, HTML код -->
<html>
  <head></head>
  <body>
    ...
    <!-- сторонний плагин -->
    <script src="https://analytics.vendor.com/v1.1/
script.js"></script>
  </body>
</html>
```

Листинг 2

Этот случай схож с предыдущим в том, что код плагина будет исполняться в контексте домена приложения и Same-origin policy не будет защищать клиентов приложения в случае вредоносного кода. Но теперь плагин не находится под контролем владельца приложения и атакующий или владелец плагина могут разместить другой код по тому же самому URL. Описанный способ включения может привести к изменению логики работы приложения (в случае встраивания вредоносного кода) или его неработоспособности (в случае несовместимости новой версии плагина с приложением).

1.5.3 Менеджер плагинов

Третий способ подключать плагины - использовать менеджер (агрегатор) плагинов. На страницу хоста вставляется код (Листинг 3), который запрашивает плагины у менеджера плагинов и подключает их.

Менеджер плагинов возвращает сторонние Javascript-файлы, которые владелец приложения настроил для встраивания в страницы. Каждый запрос к менеджеру плагинов может приводить к возврату и встраиванию в страницу множества различных файлов от разных разработчиков.

```
<!-- Некоторый хост, например example.com, HTML код -->
<html>
  <head></head>
  <body>
    ...
    <!-- Менеджер плагинов -->
    <script>(function(w, d, s, l, i){
        w[l] = w[l] || [];
        w[l].push({'tm.start':new Date().getTime(),
event:'tm.js'});
        var f = d.getElementsByTagName(s)[0], j =
d.createElement(s), dl = l != 'dataLayer' ? '&l=' + l : '';
        j.async=true;
        j.src='https://tagmanager.com/tm.js?id=' + i + dl;
        f.parentNode.insertBefore(j, f);
       })(window, document, 'script', 'dataLayer', 'TM-
FOOBARID');
    </script>
    <!-- /Менеджер плагинов -->
  </body>
</html>
```

Листинг 3

Контент, который возвращается менеджером плагинов, может быть динамически изменен владельцем приложения через пользовательский интерфейс настроек менеджера на его сайте, который дружелюбен для пользователей с нетехнической специализацией.

Менеджер плагинов может генерировать Javascript-код на основании настроек в пользовательском интерфейсе (какие данные и когда требуются для статистики). Таким образом, менеджер плагинов позволяет „нетехническим" пользователям проектировать Javascript-плагины без знания самого языка.

С точки зрения информационной безопасности менеджер плагинов имеет те же самые недостатки, что и способ подключения плагинов из пункта 1.5.2. Но так же в силу особенностей панели управления менеджером плагинов (возможность подключения плагинов, не изменяя кода самого приложения, или встраивание кода в HTML-страницы приложения) появляется вектор атаки направленный на захват к ней доступа.

1.6 Варианты повышения защищенности приложения с плагинами

1.6.1 Песочница

Можно изолировать плагины с помощью размещения их внутри iframe[4](см. Листинг 4 и Листинг 5). При таком подключении плагина из внешнего источника его Javascript-код не будет иметь прямого доступа к DOM и cookie веб-приложения, так как будет работать Same-origin policy. Основная страница и iframe могут общаться с помощью механизма postMessage. Кроме того, iframe может быть защищен с помощью атрибута sandbox, который позволяет ограничить, какой контент может присутствовать внутри iframe. Для приложений с высоким риском информационной безопасности рекомендуется использовать CSP (Content Security Policy)[5], которая при использовании менеджера плагинов позволит строго ограничить ресурсы откуда может загружаться плагин.

```
<!-- Некоторый хост, например example.com, HTML код -->
<html>
  <head></head>
    <body>
      ...
      <!-- iframe со сторонним плагином -->
      <iframe src=„https://help-host.net/analytics.html”
sandbox="allow-same-origin allow-scripts"></iframe>
  </body>
</html>
```

Листинг 4

[4] Данный способ подходит только для простейших плагинов для сбора статистики, которые не отслеживают действия пользователей. Так как иначе код, отвечающий за передачу событий с основной страницы в iframe, будет сравним по сложности с кодом плагина.

```
<!-- help-host.net/analytics.html -->
<html>
  <head></head>
    <body>
      ...
      <script>
      window.addEventListener("message", receiveMessage,
false);
      function receiveMessage(event) {
        if (event.origin !== "https://example.com:443") {
          return;
        } else {
        // Код формирующий DOM и нужные данные для стороннего
плагина
        }
      }
      </script>
      <!-- сторонний плагин -->
      <script src="https://analytics.vendor.com/v1.1/
script.js"></script>
  </body>
</html>
```

Листинг 5

1.6.2 Целостность подресурса

Целостность подресурса (SRI) [6] является средством защиты, которое позволяет браузерам проверять, что файлы, которые они загружают, не подверглись никаким изменениям (см. Листинг 6). Принцип работы основан на использовании хеша.

Данный механизм можно использовать для проверки, что код скрипта, который загружается браузером пользователя, является в точности тем, который планировал подключить владелец приложения. Браузер проверит, что хеш от полученного кода совпадает со значением атрибута integrity, перед запуском плагина[5].

```
<script src="https://example.com/example-framework.js"
        integrity="sha384-oqVuAfXRKap7fdgcCY5uykM6+R9GqQ8K/
uxy9rx7HNQlGYl1kPzQho1wx4JwY8wC"
        crossorigin="anonymous"></script>
```

Листинг 6

Использование SRI защищает приложение от атак, нацеленных на изменение кода подключаемого плагина в результате захвата контроля над хостом предоставляющим плагин.

[5] Данный механизм работает не во всех современных браузерах.

1.6.3 Копирование

Еще одним простейшим методом защиты является копирование плагина в окружение приложения. В таком случае плагин оказывается в зоне ответственности владельца приложения и исключается возможность подмены плагина без взлома самого приложения или его окружения. Но в таком случае владельцу приложения придется вручную следить за обновлениями плагина, чтобы держать на своем сервере актуальную версию.

1.7 Цель работы

Целью данной работы является анализ возможных угроз при использовании Javascript-плагинов для сбора статистики в веб-приложении.

1.8 Постановка задачи

Для достижения поставленной цели необходимо решить следующие задачи:

1. Изучить документацию и возможности популярных плагинов.
2. Исследовать недостатки привнесенные в приложение плагинами.
3. Исследовать использование плагинов на практике.

1.9 Актуальность

Высокая востребованность SEO-специалистов на рынке [7], использование Javascript-плагинов для сбора статистики, как одного из основных инструментов для SEO оптимизации сайта, а так же отсутствие исследований на заданную тему подтверждают актуальность данной работы.

2 Обзор популярных плагинов

2.1 Google Analytics

Официальная документация к плагину предлагает два способа подключения к приложению.

```
<!-- Google Analytics -->
<script>
(function(i,s,o,g,r,a,m)
{i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
(i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new
Date();a=s.createElement(o),
m=s.getElementsByTagName(o)
[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
})(window,document,'script','https://www.google-analytics.com/
analytics.js','ga');

ga('create', 'UA-XXXXX-Y', 'auto');
ga('send', 'pageview');
</script>
<!-- End Google Analytics -->
```

Листинг 7

Согласно документации [8], код, приведенный в Листинге 7, выполняет 4 основных задачи:

1. Создает элемент <script>, который инициирует асинхронную загрузку библиотеки analytics.js по URL https://www.google-analytics.com/analytics.js.
2. Инициирует глобальную функцию ga (называемую очередью команд ga()), которая позволяет запланировать выполнение команд после полной загрузки и готовности библиотеки analytics.js.
3. Добавляет в очередь команд ga() команду создания нового счетчика для ресурса, указанного в параметре 'UA-XXXXX-Y'.
4. Добавляет в очередь команд ga() команду отправить в Google Analytics обращение типа pageview для текущей страницы.

Для повышения скорости загрузки веб-приложения в современных браузерах, можно использовать второй способ приведенный в Листинге 8, тогда плагин будет загружаться асинхронно.

```
<!-- Google Analytics -->
<script>
window.ga=window.ga||function(){(ga.q=ga.q||
[]).push(arguments)};ga.l=+new Date;
ga('create', 'UA-XXXXX-Y', 'auto');
ga('send', 'pageview');
</script>
<script async src='https://www.google-analytics.com/
analytics.js'></script>
<!-- End Google Analytics -->
```

Листинг 8

Базовые возможности Google Analytics позволяют получать следующую информацию:

- Общее время нахождения пользователя на сайте.
- Время нахождения на каждой странице и порядок их посещения.
- Сведения о ссылках, которые пользователь нажимал.
- Географическое положение пользователя на основании его IP.
- Используемые браузер и операционная система.
- Разрешение экрана и наличие расширений для браузера (Java, Flash).
- Сайт-источник перехода на основании заголовка Referer при загрузке первой страницы.

Возможности Google Analytics можно расширить с помощью дополнений. Все команды к Google Analytics передаются через очередь команд „ga", и для подключения дополнения требуется конструкция приведенная в Листинге 9.

Листинг 9

```
<script>
ga('create', 'UA-XXXXX-Y', 'auto');
<!-- Команда require регистрирует дополнение -->
ga('require', 'linkTracker');
ga('send', 'pageview');
</script>

<!--Внимание! Дополнение должно быть подключено после
инициализирующего скрипта-->
<script async src="/path/to/link-tracker-plugin.js"></script>
```

Так же на страницах веб-приложений можно встретить старую версию плагина Google Analytics. Google не рекомендует ее использовать и может прекратить поддержку.

Основное отличие старой версии от новой в отсутствии механизма подключения дополнений.

Данный принцип подключения плагина для сбора статистики к приложению относится к типу 1.5.2, соответственно для него актуальны уже описанные угрозы информационной безопасности. В случае получения атакующим доступа к панели администрирования он сможет только получить данные статистики посещения, потому что управление плагином происходит в коде самого приложения. Еще одним вектором атаки остаются дополнения для плагина, если они подключаются из внешнего источника.

2.2 Яндекс.Метрика

Яндекс рекомендует подключать плагин с помощью кода приведенного в Листинге 10 [9].

Данный код можно разбить на 2 части: внутри блока <script> и внутри блока <noscript>. Если браузер поддерживает выполнение Javascript-кода, то он выполнит код из блока <script>, а <noscript> проигнорирует. Иначе будет загружена картинка по URL из блока <noscript>.

Если Javascript-код в браузере не выполняется, то собираемая информация существенно ограничена. Информация извлекается из HTTP-заголовков запроса на загрузку картинки:

- IP- адрес клиента.
- Адрес загруженной страницы.
- Информацию о браузере и операционной системе.
- Отсутствие поддержки Javascript (так был обработан блок <noscript>).

Если браузер поддерживает Javascript, то выполнение кода приведет к следующему:

1. Будет создан элемент <script>, который инициирует асинхронную загрузку библиотеки watch.js со страницы https://mc.yandex.ru/metrika/watch.js.
2. Создан элемент класса Ya.Metrika с настройками счетчика.

В связи с тем, что Javascript-код имеет доступ к информации о заголовках HTTP-запросов, URL источника перехода на страницу, параметрах экрана и окна браузера, дополнительных расширениях браузера (Flash, Silverlight, Java), количество собираемой

информации увеличивается. Самым существенным преимуществом выполнения Javascript-кода является возможность отслеживать события, происходящие при взаимодействии пользователя со страницей.

Листинг 10

```
<!-- Yandex.Metrika counter -->
<script type="text/javascript">
    (function (d, w, c) {
        (w[c] = w[c] || []).push(function() {
            try {
                w.yaCounterXXXXXXXX = new Ya.Metrika({
                    id:XXXXXXXX,
                    clickmap:true,
                    trackLinks:true,
                    accurateTrackBounce:true,
                    webvisor:true,
                    ut:"noindex"
                });
            } catch(e) { }
        });

        var n = d.getElementsByTagName("script")[0],
            s = d.createElement("script"),
            f = function () { n.parentNode.insertBefore(s,
n); };
        s.type = "text/javascript";
        s.async = true;
        s.src = "https://mc.yandex.ru/metrika/watch.js";

        if (w.opera == "[object Opera]") {
            d.addEventListener("DOMContentLoaded", f, false);
        } else { f(); }
    })(document, window, "yandex_metrika_callbacks");
</script>
<noscript><div><img src="https://mc.yandex.ru/watch/XXXXXXXX?
ut=noindex" style="position:absolute; left:-9999px;" alt="" /
></div></noscript>
<!-- /Yandex.Metrika counter -->
```

У Яндекс.Метрики есть возможность записывать поведение пользователей на страницах веб-приложения для дальнейшего воспроизведения в формате видео в панели управления плагином. Данная технология называется Вебвизором. С помощью нее можно проанализировать поведение пользователя - что делает пользователь на сайте, как перемещает мышку, какие ссылки кликает, как заполняет формы. Сейчас качество записей существенно зависит от сайта (некоторые виды динамического контента совсем не записываются) и настроек Вебвизора. Со слов разработчиков, Вебвизор 2.0 который ожидается к выходу в скором времени, лишится недостатков первой версии и будет

записывать поведение пользователей и контент сайта в точности, как его видел пользователь [10].

В связи с тем, что запрашивается только основной файл плагина из надежного источника и нет расширений функциональности, то единственным вектором атаки остается получение доступа к административной панели управления плагином. В таком случае атакующий при включенном Вебвизоре сможет увидеть пользовательские данные, введенные в формы на сайте, а так же любую другую информацию, которая отображается на странице. Если Вебвизор выключен, то будет доступ только к статистике и данным, которые владелец приложения добавил в параметры пользователя или сессии с помощью редактирования кода на HTML-страницах приложения.

2.3 Рейтинг Mail.Ru

Листинг 11

```
<!-- Rating@Mail.ru counter -->
<script type="text/javascript">
var _tmr = window._tmr || (window._tmr = []);
_tmr.push({id: "XXXXXXX", type: "pageView", start: (new
Date()).getTime()});
(function (d, w, id) {
  if (d.getElementById(id)) return;
  var ts = d.createElement("script"); ts.type = "text/
javascript"; ts.async = true; ts.id = id;
  ts.src = (d.location.protocol == "https:" ? "https:" :
"http:") + "//top-fwz1.mail.ru/js/code.js";
  var f = function () {var s = d.getElementsByTagName("script")
[0]; s.parentNode.insertBefore(ts, s);};
  if (w.opera == "[object Opera]")
{ d.addEventListener("DOMContentLoaded", f, false); } else
{ f(); }
})(document, window, "topmailru-code");
</script>
<noscript><div>
<img src="//top-fwz1.mail.ru/counter?id=XXXXXXX;js=na"
style="border:0;position:absolute;left:-9999px;" alt="" />
</div></noscript>
<!-- //Rating@Mail.ru counter -->
```

Код из Листинга 11 [11] можно разделить на 2 части, как и у Яндекс.Метрики. Принцип работы второго блока (<noscript>) и собираемые им данные не отличаются от уже рассмотренного случая. В то же время задачи, выполняемые Javascript-кодом из блока <script>, аналогичны решаемым в коде инициализации Google Analytics, с той разницей, что _tmr (аналог ga) является массивом.

Не смотря на схожесть кода инициализации с Google Analytics, Рейтинг Mail.Ru не поддерживает расширяющие плагины. Поэтому единственным вектором атаки остается административная панель управления плагином. Получив доступ к панели управления, атакующий сможет видеть только статистику сайта.

2.4 Google Tag Manager

Для установки Google Tag Manager на сайт требуется разместить фрагмент из Листинга 12 в блок <head>[12].

Листинг 12

```
<!-- Google Tag Manager -->
<script>(function(w,d,s,l,i){w[l]=w[l]||
[];w[l].push({'gtm.start':
new Date().getTime(),event:'gtm.js'});var
f=d.getElementsByTagName(s)[0],
j=d.createElement(s),dl=l!
='dataLayer'?'&l='+l:'';j.async=true;j.src=
'https://www.googletagmanager.com/gtm.js?
id='+i+dl;f.parentNode.insertBefore(j,f);
})(window,document,'script','dataLayer','GTM-XXXXXX');
</script>
<!-- End Google Tag Manager -->
```

Данный код, как и в предыдущих случаях, создает элемент <script>, который инициализирует асинхронную загрузку библиотеки. Также добавляет в используемый Google Tag Manager уровень данных новое событие „gtm.js”.

Так же требуется вставить в блок <body> фрагмент из Листинга 13.

Листинг 13

```
<!-- Google Tag Manager (noscript) -->
<noscript><iframe src="https://www.googletagmanager.com/
ns.html?id=GTM-XXXXXX"
height="0" width="0" style="display:none;visibility:hidden"></
iframe></noscript>
<!-- End Google Tag Manager (noscript) -->
```

В этом фрагменте добавляется сбор базовой информации для браузеров, не поддерживающих Javascript. Принцип работы не отличается от остальных плагинов.

Google Tag Manager является менеджером плагинов, все настройки и управление подключаемыми плагинами осуществляются на веб-странице панели управления. Если атакующий получит к ней доступ, то сможет настроить встраивание HTML-блока (в том числе содержащего <script>) в страницы приложения. В силу особенностей менеджера

плагинов это действие не требует никаких изменений в приложении со стороны его владельца. Поэтому в данном случае атакующий сможет добиться поставленной цели.

2.5 Угрозы безопасности

Владельцы плагинов осознают угрозы информационной безопасности при получении атакующим доступа к панели управления плагинами, и поэтому ответственно подходят к риску деанонимизации пользователя плагина (суть владельца приложения или субъекта, которому владелец делегировал доступ и управление) по данным, которые доступны публично. Для того чтобы получить доступ к панели управлении плагином атакующему нужно узнать почту на которую этот плагин зарегистрирован. Но владельцы плагинов заботятся о том, чтобы по доступной информации (id счетчика) получить такую информацию было затруднительно.

Единственный вопрос возникает к Mail.ru, которая является владельцем плагина Рейтинг Mail.ru. На странице восстановления доступа к панели управления предлагается два основных способа:

1. Ввести почту пользователя, с помощью которой происходила регистрации на сервисе, и тогда на нее будет отправлена ссылка для восстановления доступа.
2. Разместить в корневом каталоге сервера файл с именем top.mail.ru-XXXXXXX.txt (где XXXXXXX - id счетчика), в котором будет записана почта для восстановления доступа.

Если после восстановления доступа к счетчику с помощью второго способа оставить файл в открытом доступе, это позволить связать администратора статистики с его почтой (логином) и проводить целевые атаки на конкретный почтовый ящик с целью захвата доступа к панели управления плагином.

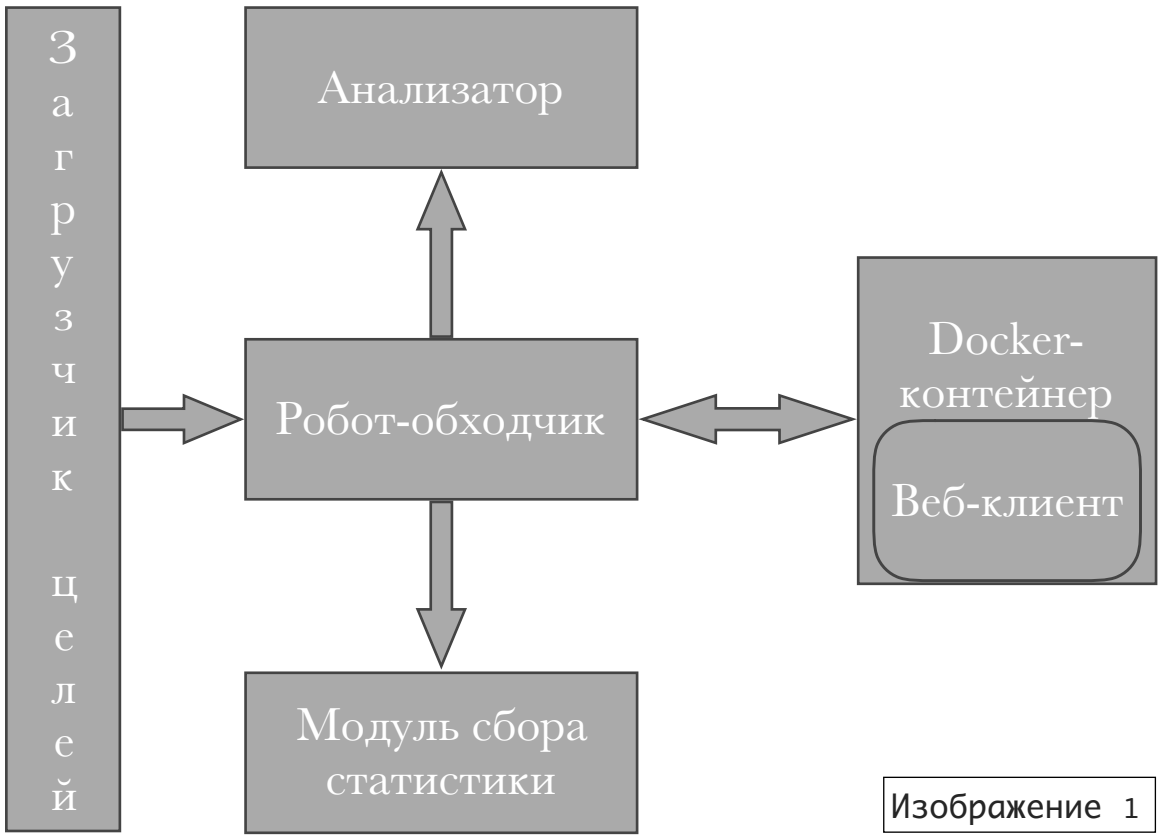
3 Реализация инструмента

В рамках данной работы был написан инструмент [13], позволяющий собирать статистику использования Javascript-плагинов в веб-приложениях. К реализации предъявлялись следующие требования:

- Возможность добавлять для отслеживания новые плагины.
- Возможность собирать статистику запросов к внешним ресурсам.
- Вывод итоговых результатов в распространенном формате.

3.1 Архитектура инструмента

Приложение реализовано на языке Python и имеет модульную архитектуру, так же используются Docker-контейнеры [14] (см. „Изображение 1”).



Изображение 1

- *Загрузчик целей.* Модуль используется для парсинга входного списка целей и заполнения внутренних структур приложения.
- *Робот-обходчик.* Робот-обходчик используется для сбора информации для всех целей и для обеспечения взаимодействия между модулями.
- *Веб-клиент.* В качестве веб-клиента используется scrapy Splash [15].

- *Docker-контейнер.* Веб-клиент запускается внутри Docker-контейнера, для изолированного формирования HAR[6] для каждой цели.
- *Анализатор.* Анализатор, на основании HAR и параметров для плагинов, решает используется ли плагин в данном веб-приложении или нет.
- *Модуль сбора статистики.* Модуль сбора статистки собирает информацию о самых частых запросах для всех целей.

3.2 Детали реализации

На вход инструменту подается JSON-форматированный список целей со структурой, показанной в Листинге 14.

```
{ "Target_class": [
    {
      "name" : „имя цели",
      "url" : „необязательный параметр url",
      "param" : „необязательный параметр",
      "another param" : „другой необязательный параметр,
специфичный для этой цели"
    },
    ...,
    {
      "name" : "имя цели"
    }
  ],
  "Another_target_class": [
    ...
  ]
}
```

Листинг 14

Задаются имена классов целей, которые будут использоваться загрузчиком для формирования внутреннего списка целей (по умолчанию используется класс Target). Каждому имени соответствует массив целей заданного класса. Цель задается с помощью обязательного поля „name" и необязательного поля „url". Если „url" не задан, то он будет сформирован на основании имени цели и дополнительных полей, задающих цель.

Полученный массив передается роботу-обходчику, который для каждой цели запускает Docker-контейнер с веб-клиентом и передает веб-клиенту URL цели. На

[6] HAR (HTTP Archive format) - JSON-форматированный формат архивного файла для логирования взаимодействий браузера с веб-приложением. Спецификация данного формата разрабатывается консорциумом W3C.[16]

каждый запрос веб-клиент возвращает HAR, которые робот-обходчик собирает в массив. После обхода всех целей робот-обходчик передает этот массив в модуль сбора статистики и анализатор.

Модуль сбора статистики по полученному массиву строит рейтинг самых популярных URL, среди запросов сделанных к доменам отличным от доменов целей. Анализатор по тому же массиву HAR-ов и набору регулярных выражений определяет какие плагины подключались приложением.

После обработки всех целей роботом-обходчиком и получения результатов работы модуля статистики и анализатора приложение формирует файл с результатами (result.csv) и выводит на стандартный поток вывода рейтинг самых популярных URL, на которые делались запросы. В итоговом файле каждой строке соответствует цель из входного списка, в столбцах располагается информация о имени цели, используемых плагинах, URL приложения (см. Листинг 15).

```
name,mail_top,yandex_metrika,google_tag,google_analytics,url
Имя цели 1,False,True,True,True,https://url_1.ru/
Имя цели 2,False,False,False,False,https://url_2.ru/
```

Листинг 15

3.3 Обоснование выбора инструментария

Основная задача веб-клиента в инструменте - это предоставление HAR корректной загрузки веб-приложения (как в браузере у пользователя). Поэтому удобное полученииe HAR и корректная загрузка являлись главными требованиями.

Scrapy Splash - легковесный веб-клиент с HTTP API, реализованный на Python и использующий Twisted [17] и QT [18]. Splash построен на „веб-движке” WebKit [19], который используется в современных веб-браузерах: Safari, Google Chrome и т.д., что позволяет рассчитывать на результат загрузки HTML-страниц приложения, как у обычного пользователя, который работает с веб-приложением через веб-браузер. В то же время Splash позволяет взаимодействовать с приложением, не затрачивая ресурсов на отображение контента. Используя HTTP API (Splash при запуске стартует веб-сервер и ждет запросы), можно легко получить HAR загрузки веб-приложения.

Selenium WebDriver [20], так же построенный на WebKit, не имеет встроенной поддержки формирования HAR, единственный выход в использовании связки Selenium Web Driver + BrowserMob Proxy [21]. Но данная связка в силу особенностей реализации

плохо масштабируется, и при увеличении входного списка целей время будет линейно увеличиваться.

Другим известным веб-клиентом построенным на WebKit является PhantomJS[22]. Он схож со Splash тем, что тоже не затрачивает ресурсов на отображение контекста веб-страницы пользователю. Но в нем нет встроенного решения для получения HAR, что требует дополнительной реализации такого компонента. Так же выбор был сделан не в пользу этого веб-клиента, потому что он реализован на Javascript, в то время как, было принято решение писать инструмент на Python. Из-за чего бы возникли проблемы с реализацией связи между инструментом и веб-клиентом.

Во время написания инструмента и работы со Splash было выяснено, что встроенный механизм очистки кэша работает некорректно, удаляя не все записи, и единственным выходом был перезапуск веб-клиента перед обработкой новой цели. Splash предлагается для установки и использования двумя способами:

1. Из исходников. Скачать исходный код, установить на локальной машине все требуемые зависимости.
2. С помощью Docker-контейнеров[7]. В этом случае не требуется устанавливать на машину зависимости, так как они все упакованы вместе со Splash в образ. Достаточно его скачать и запускать веб-клиент изолировано внутри контейнера.

В силу особенностей Docker-контейнеров перезапуск является не затратной операцией, при которой изменения внутри контейнера не сохраняются, как в виртуальной машине. В силу удобства и незначительного влияния на время готовности веб-клиента было принято решение использовать Docker-контейнеры.

[7] Docker - программное обеспечение для автоматизации развертывания и управления приложениями, которое позволяет распространять приложение в виде образа, включающего все зависимости.

4 Эксперимент

Одной из самых критичных к утечке пользовательских данных сфере относится банковское обслуживание. С помощью написанного в рамках данной работы инструмента была собрана статистика использования популярных плагинов на страницах аутентификации в интернет-банках.

Цель проведения эксперимента - проверка какие банки взяли на себя ответственность за риски информационной безопасности, появляющиеся при использовании сторонних скриптов на страницах с конфиденциальными данными, или не в курсе их [3]. В ходе эксперимента будут проверяться страницы входа в интернет-банк, где конфиденциальные данные (логин, пароль) вводит пользователь. Проанализировав данные, полученные с помощью инструмента, можно будет сделать выводы о использовании плагинов, частоте их использования и выявить самый популярный из них.

4.1 Методика проведения эксперимента

С сайта banki.ru был взят финансовый рейтинг банков России. При автоматизированном формировании списка URL страниц входа в интернет-банки по списку банков, с использованием поисковых систем, не удалось добиться приемлемой точности. Поэтому для каждого банка из рейтинга вручную был найден URL страницы входа в интернет-банк для физических лиц. На основании полученных данных был составлен список целей (если интернет-банк для физических лиц отсутствует, то url проставлялся null) в соотвествии с входным форматом для инструмента. Полученный файл использовался как входной список целей для инструмента.

Эксперимент можно разделить на 2 части:

- Сначала производится запуск инструмента с пустым списком фильтров для плагинов. Результаты работы модуля сбора статистки позволяют определить часто используемые запросы к внешним ресурсам, сопоставив которые с рекомендациями подключения плагинов, можно определить какие плагины используются на страницах входа в интернет-банки.
- Вторая часть эксперимента заключается в повторном запуске инструмента с тем же входным списком целей, но в этот раз с уже сформированным на основании результатов первой части списком фильтров для плагинов.

4.2 Результаты эксперимента

На основании результатов эксперимента (см. „таблица 2” в Приложении) можно сделать выводы:

- 55.555% банков имеют интернет-банк для физических лиц (см. Диаграмма 1).
- Медиана номеров банков, имеющих интернет-банк, равна 225, откуда следует, что чем выше финансовые показатели банка, тем больше вероятность наличия интернет-банка.
- ~30% банков, имеющих интернет-банк, пользуются услугами облачных банков (faktura.ru, handybank.ru).
- Все банки, пользующиеся услугами faktura.ru подключают плагин Google Analytics (новую версию) на страницах входа в интернет-банк, в то время как у пользователей handybank.ru подключается старая версия плагина.
- 17% подключают плагины на страницах входа в интернет банк (44% если учитывать облачные) (см. Диаграмма 2).
- Самым популярным плагином является Google Analytics используемый в 76% приложений, подключающих плагины(98% если учитывать старую версию).

Диаграмма 1

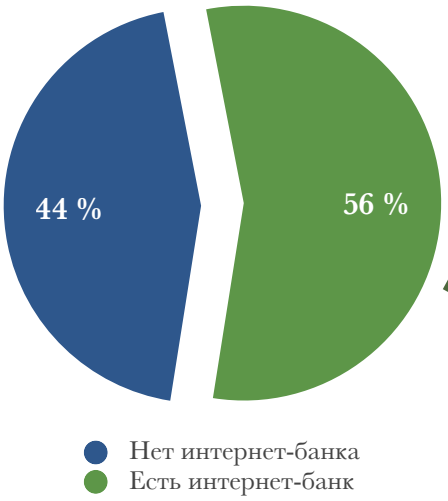


Диаграмма 2

5 Результаты

В рамках выполнения данной работы получены следующие результаты:

- Исследованы недостатки вносимые в веб-приложения использованием плагинов.
- Сформирован список URL входа в интернет-банки России.
- Разработано инструментальное средство для проверки использования Javascript-плагинов на страницах веб-приложений.
- Собрана и проанализирована статистика, использования банками сторонних плагинов на страницах входа в интернет-банк.

Поставленная задач полностью выполнена. Для дальнейшего развития реализованного инструментария можно предложить:

- Повышение скорости обработки списка целей за счет реализации параллельного робота-обходчика.
- Реализация проверок специфичных для каждого плагина.

6 Приложение

Условные обозначения в таблице:

- Яндекс Метрика - YM.
- Google Tag Manager - GT.
- Mail Рейтинг - MT.
- Google Analytics - GA.
- Google Analytics старая версия - GAO.

Если плагин используется на странице входа в интернет-банк, то в соответствующей ячейке стоит „+”, иначе „—”.

7 Список литературы

1. Barth A. The web origin concept. – 2011.
2. Le Hégaret P., Whitmer R., Wood L. Document object model (dom) //W3C recommendation (January 2005) http://www. w3. org/DOM. – 2002.
3. https://www.owasp.org/index.php/3rd_Party_Javascript_Management_Cheat_Sheet.
4. https://www.owasp.org/index.php/Cross-site_Scripting_(XSS).
5. World Wide Web Consortium (W3C), Content Security Policy Level 3.
6. World Wide Web Consortium (W3C), Subresource Integrity.
7. Edunews, Профессия „SEO-специалист", http://edunews.ru/professii/obzor/Tehnicheskie/seo-spezialist.html.
8. Google Analytics Guide, https://developers.google.com/analytics/devguides/collection/analyticsjs/.
9. Яндекс.Метрика начало работы, https://yandex.ru/support/metrika/quick-start.xml.
10. Блог Яндекс.Метрики, https://yandex.ru/blog/metrika/obnovlyaem-vebvizor.
11. Рейтинг Mail.ru установка, https://help.mail.ru/top/code/receive.
12. Google Tag Manager Guide, https://support.google.com/tagmanager/answer/6103696?hl=en&ref_topic=3441530.
13. Реализация инструмента, https://github.com/dsp25no/crawler
14. Docker - container platform, https://www.docker.com.
15. Splash - javascript rendering service, http://splash.readthedocs.io/en/stable/index.html.
16. Odvarko J., Jain A., Davies A. HTTP Archive (HAR) format //W3C draft. – 2012.
17. Twisted - event-driven networking engine, https://twistedmatrix.com/trac/.
18. QT for application development, https://www.qt.io/qt-for-application-development/.
19. The WebKit open source project, http://webkit.org/.
20. Selenium WebDriver, http://docs.seleniumhq.org/projects/webdriver/.
21. BrowserMob Proxy, https://bmp.lightbody.net.
22. PhantomJs, http://phantomjs.org.