УДК 004.4

ТЕХНОЛОГИИ ВЕБ-БРАУЗЕРОВ, ИСПОЛЬЗУЕМЫЕ АНТИБОТ-СИСТЕМАМИ И ИМЕЮЩИЕ НЕЖЕЛАТЕЛЬНЫЕ ДЛЯ КОНЕЧНОГО ПОЛЬЗОВАТЕЛЯ ПОБОЧНЫЕ ЭФФЕКТЫ

Янковский Г.В. (БГТУ ВОЕНМЕХ),
Научный руководитель – Вальштейн К.В. (БГТУ ВОЕНМЕХ)

Введение. В связи с развитием технологий искусственного интеллекта разработчики систем, предотвращающих эксплуатацию автоматизированных программ-ботов на веб-ресурсы, вынуждены переходить от традиционных методов проверки на «человечность» пользователя путем ввода символов с картинки на более комплексные решения проблемы фильтрации трафика [1]. В большинстве интернет-браузеров присутствуют различные механизмы, необходимые для отображения динамического содержимого. Подобные технологии могут быть уязвимы, однако их использование закладывается разработчиками в антибот-системы, поэтому при посещении веб-сайтов с защитой от программ-ботов у пользователей не остается выбора, кроме как разрешить использование всех механизмов интернет-обозревателя, чтобы просмотреть содержимое веб-страницы.

Основная часть. Современные антибот-системы на стороне пользователя могут использовать механизмы, встроенные в любой современный веб-браузер. К таким технологиям можно отнести:

– WebGL API;
– Service Workers;
– Notifications API.

WebGL API (Canvas API) – интерфейс, предоставляющий возможности рисования изображений на графическом ускорителе, выполняя обращения через JavaScript [2]. Как правило, данная технология используется по своему прямому назначению, однако может иметь и другое применение. Через JavaScript можно получить информацию о возможностях графического ускорителя: параметры кадрового буфера, шейдеров и тому подобное. В браузерах на основе Chromium имеется возможность получить полное наименование видеокарты. Кроме того, из-за особенностей выполнения отрисовки фрагментов, а также взаимодействия с видеоускорителем разных версий библиотек и браузеров, появились механизмы снятия цифрового отпечатка устройства, с которого пользователь просматривает страницу [3]. С большой вероятностью можно узнать, с какого семейства веб-браузеров (иногда доступна приблизительная информация о версии программы интернет-обозревателя) или операционной системы открыт веб-ресурс. Обычно система защиты от DDOS-атак компании Cloudflare проводит операции с WebGL API перед тем, как разрешить пользователю посещать веб-сайт, и отклоняет все запросы, если контекст данного API заблокирован со стороны веб-браузера, или отпечаток интернет-обозревателя не является достоверным. Таким образом, заходя на сайты, использующие данную технологию, пользователь без своего ведома может получить нежелательную нагрузку на устройство, а также быть идентифицированным на всех веб-сайтах с защитой от одного и того же провайдера.

Service Workers – технология, позволяющая выполнять в независимом от веб-страницы контексте действия по управлению передаваемыми данными. Обычно используется для кеширования данных и офлайн-представлению веб-приложения. Однако трафик, который просматривается той или иной фоновой службой, может быть подменен – в контексте скрипта Service Worker имеется возможность выполнять запросы на сторонние ресурсы, производить вычислительные операции перед тем, как возвратить новое содержимое той или иной страницы веб-ресурса. Технология используется в том числе и в системах фильтрации роботизированного трафика. Например, при попытке посетить веб-ресурсы www.komus.ru, dns-shop.ru, spb.leroymerlin.ru/catalogue/, которые используют DDOS-защиту от компании

Qrator, через браузер LibreWolf, ориентированный на расширенную приватность, появляется текст с ошибкой – пользователь является роботом, запрос блокируется. Предположительно, одна из причин, почему это происходит – TypeError ошибка. Скрипт пытается вызвать 'get ready' для объекта, который не реализует интерфейс ServiceWorkerContainer. Активные Service Workers, зарегистрированные провайдером DDOS-защиты, от имени пользователя могут порождать излишний интернет-трафик в фоновом режиме без ведома владельца веб-ресурса.

Push и Notifications API – программные интерфейсы для отправки уведомлений пользователю на уровне операционной системы в фоновом режиме, доступны при использовании Service Workers. Сообщения, отправляемые через данные интерфейсы, появляются в окружении рабочего пространства пользователя. Существуют некоторые системы, которые под видом защиты от атак запрашивают у пользователя разрешение на отправку уведомлений через Push API. После предоставления прав пользователь может открыть веб-сайт, однако будет получать спам-сообщения на рабочем столе. Помимо рассылки нежелательных сообщений, подобная технология браузера может иметь изъяны в безопасности. В 2014 году для настольных версий движка браузера Firefox была обнаружена уязвимость, позволяющая выполнять код в привилегированной среде, используя механизм отправки подобных нотификаций [4].

На данный момент компания Google ведет разработку над API Web Integrity, который призван проверять клиентскую среду на предмет манипуляции цифровым контентом [5]. Предполагается, что технология позволит собирать данные о среде выполнения веб-браузера и уведомлять владельцев веб-ресурсов о том, используется ли на стороне пользователя расширения для блокировки рекламных блоков, являются ли действия пользователя добросовестными (ведется ли подбор паролей или фишинговая атака), входит ли устройство интернет-обозревателя в число скомпрометированных. Возможно, во время формирования решения о «честности» клиентской программы веб-браузера третьей стороной будет получено большое количество данных о среде выполнения, которое может быть собрано путем исполнения действий на привилегированном уровне операционной системы.

Выводы. Большинство компаний, обслуживающих системы фильтрации программ-ботов, не раскрывают секретов работы механизмов защиты, а из-за сильно обфусциированного исходного кода скриптов, которые выполняются на стороне веб-браузера, провести анализ действий крайне затруднительно. Подобные системы со временем могут запрашивать всё более привилегированные действия в операционной системе пользователей веб-ресурсов, и, соответственно, вносить изменения в работу более низкоуровневых компонентов, затрагивать больше конфиденциальной информации.

Список использованных источников:

1. Understanding CAPTCHA: сайт. – URL: https://dev.to/adityapratapbh1/understanding-captcha-history-usage-and-effectiveness-4jd7 (дата обращения: 18.01.2024). – Текст: электронный

2. Canvas API - Web APIs - MDN: сайт. – URL: https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API (дата обращения: 20.01.2024). – Текст: электронный

3. Browser Fingerprinting - IEEE Spectrum: сайт. – URL: https://spectrum.ieee.org/browser-fingerprinting-and-the-onlinetracking-arms-race (дата обращения: 23.01.2024). – Текст: электронный

4. Mozilla Foundation Security Advisory: сайт. – URL: https://www.mozilla.org/en-US/security/advisories/mfsa2014-42/ (дата обращения: 24.01.2024). – Текст: электронный

5. Web-Environment-Integrity/explainer.md: сайт. – URL: https://github.com/explainers-by-googlers/Web-Environment-Integrity/blob/main/explainer.md (дата обращения: 25.01.2024). – Текст: электронный