DPA
ANALYTICS

ИССЛЕДОВАНИЕ БЕЗОПАСНОСТИ РОССИЙСКИХ FRONTEND-ПРИЛОЖЕНИЙ

1 полугодие 2026

dpa-analytics.ru

СОДЕРЖАНИЕ

ОБ ИССЛЕДОВАНИИ

Как правило при обеспечении информационной безопасности (ИБ) веб-приложений основное внимание уделяется защите серверной части (backend) приложения, в том числе используются решения класса WAF, NGFW, Anti-DDOS, EDR, VM и т.п.

Безопасности клиентской части приложения (frontend) уделяется значительно меньше внимания, т.к. frontend-приложения обычно выполняются вне контролируемой зоны - в браузере пользователя, являющимся "слепой" зоной для ИБ.

Однако добавление вредоносного кода на веб-страницы предоставляет злоумышленникам множество способов монетизации, например, кража данных со страниц прямо в браузере пользователя, показ фишинговых баннеров от имени компании, майнинг криптовалюты и заражение устройств пользователей через эксплуатацию уязвимостей браузера, в том числе для проникновения в корпоративные сети.

Отсутствие регулярного глубокого анализа поведения frontend-приложений и недостаточное внимание к безопасности клиентской части веб-приложений приводят к увеличению времени присутствия (недели, месяцы, годы) вредоносного кода и максимизации прибыли злоумышленников, которые все чаще выбирают в качестве цели именно frontend-приложения, что подтверждается известными инцидентами.

Целью данного исследования является оценка безопасности российских frontend-приложений, в том числе на основании определения эффективности использования механизмов безопасности браузера, зависимости от зарубежных сетей доставки контента (CDN), использования зарубежных систем аналитики, выполнения трансграничных передач персональных данных с веб-страниц и других факторов.

ПРИМЕРЫ ИНЦИДЕНТОВ

2018 – British Airways – js-сниффер, похищающий данные банковских карт

Злоумышленники добавили в один из файлов скриптов на сервере код js-сниффера. В момент клика на кнопку "Забронировать билет" сниффер перехватывал введенные в форму данные банковской карты и отправлял их на хост злоумышленников прямо в браузере пользователя. Обнаружили инцидент через 15 дней после жалоб пользователей на несанкционированные списания денег. За это время утекли данные карт 380 000 пользователей. Ущерб компании на выплату компенсаций и штраф по GDPR составил более 2 000 000 000 £

Вектор: Компрометация веб-сервера

Google Tag Manager

2021 – В 316 интернет-магазинах обнаружен js-сниффер, скрытый в Google Tag Manager

Злоумышленники взломали множество интернет-магазинов через уязвимости CMS (WordPress, Shopify, BigCommerce). Монетизация взлома, как в предыдущем инциденте, выполнялась через кражу данных банковских карт js-снифферером из формы оплаты, однако в этот раз злоумышленники добавляли на страницу скрипт Google Tag Manager (GTM). Это популярный у маркетологов инструмент, который может добавлять на страницы любые скрипты и управляется из веб-интерфейса Google. Использование на первый взгляд легитимного инструмента усложняет обнаружение вредоносного кода при ручном анализе, повышает время присутствия вредоносного кода в приложении и прибыль злоумышленников.

Вектор: Компрометация веб-сервера + Тег-менеджеры

2022 – Компрометация сервиса аналитики onthe.io, в js-скрипт добавлен вредоносный код

В результате вредоносный код стал исполняться на сайтах всех компаний-пользователей данного сервиса. Код злоумышленников выполнял "дефейс" сайтов и отображал на них политические лозунги. Пострадали более 1900 сайтов, в том числе крупных российских СМИ, банков и других компаний.

Вектор: Внешний сервис / Supply Chain Attack

Polyfill.js

2024 – Вредоносный код в библиотеке Polyfill.js

Владельцы популярной библиотеки и сервиса Polyfill.js умышленно добавили вредоносный код в проект для монетизации. Этот код по ночам на мобильных устройствах выполнял редирект пользователей на сайты онлайн-букмекеров. Не смотря на то, что жалобы на GitHub появились в первый же день, глобальная реакция сообщества и добавление в базы вредоносных библиотек произошло только через 3 месяца. Инцидент затронул > 350 000 веб-приложений, в том числе крупных международных компаний.

Вектор: Зависимости js-приложения / Внешний сервис / Supply Chain Attack

npm

2025 – Компрометация 18 npm-пакетов, имеющих более 2 000 000 000 загрузок в неделю

Злоумышленники захватили токен разработчика через фишинговое письмо и опубликовали от его имени версии библиотек с добавленным вредоносным кодом. Целью злоумышленников были frontend-приложения с функциями перевода криптовалюты. Вредоносный код перехватывал сетевые запросы на странице и при обнаружении запроса на перевод подменял адрес получателя.

Вектор: Зависимости js-приложения / Supply Chain Attack

РЕЗУЛЬТАТЫ ДЛЯ CISO / ДИРЕКТОРОВ ПО ИБ

1. Frontend-приложения - любые веб-страницы, открывающиеся в браузере пользователя (сайт, лендинг, личный кабинет, корпоративный портал, маркетплейс, онлайн-банк, CRM, ERP, любые веб-интерфейсы, в том числе доступные только внутри корпоративной сети).

2. Frontend-приложения (даже внутрисетевые) могут быть использованы злоумышленниками для кражи данных и заражения рабочих станций и мобильных устройств пользователей/посетителей.

3. Уровень безопасности российских frontend-приложений является неудовлетворительным (49 из 100). В отдельных отраслях оценка ниже 30.

4. В онлайн-банках/ДБО оценка 75 из 100 - выше, чем в других отраслях, но явно недостаточно для критичных приложений.

5. Классические средства защиты и анализаторы (WAF, NGFW, API Security, Container Security, сканеры уязвимостей, SAST, DAST, SCA) не снижают риски реализации frontend-угроз и не выявляют актуальные угрозы.

6. Работа frontend-приложений в "слепой" зоне для ИБ и низкая оценка рисков приводит к увеличению времени присутствия вредоносного кода в приложении и максимальной монетизации для злоумышленников.

7. Кража персональных данных скриптами на веб-страницах - это полноценная утечка данных.

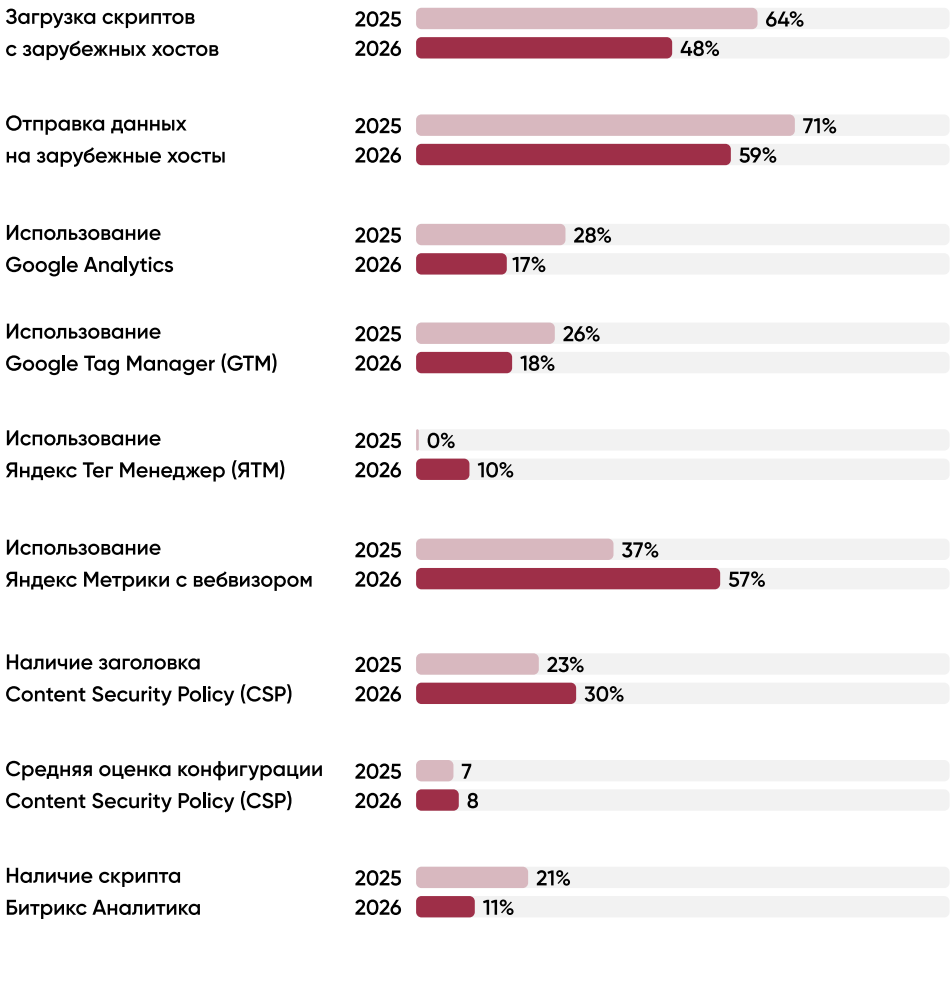
8. Утечка персональных данных в frontend-приложении либо невыполнение требований 152-ФЗ по трансграничной передаче ПД с 1 июля 2025 года могут привести к значительным штрафам для российских компаний: КоАП РФ 13.11 ч.8-9 от 1 до 18 млн. руб., КоАП РФ 13.11 ч.12-14 от 3 до 15 млн. руб.

9. Рекомендуется построить и использовать в работе модель угроз по фреймворку Frontend Kill Chain (https://dpa-analytics.ru/frontend-threat-model).

10. Рекомендуется провести пилот FAST-анализатора (Frontend Application Security Testing) для обнаружения актуальных угроз/утечек данных/вредоносного кода.

ДИНАМИКА ИЗМЕНЕНИЙ ПО СРАВНЕНИЮ С 2025 ГОДОМ

Ниже представлены основные изменения по сравнению с результатами исследования, проведенного в 1-ом полугодии 2025 года.



Общая оценка безопасности увеличилась с 39 до 49 баллов из 100.

На улучшение оценки повлияли:

- серия докладов и публикаций от DPA Analytics о безопасности frontend-приложений (https://dpa-analytics.ru/research);
- проведение пилотов и внедрений FAST-анализатора в российских компаниях;
- запрет использования зарубежных систем аналитики (152-ФЗ, ст. 18, ч. 5);
- проблемы с доступом к зарубежным хостинг-провайдерам (замедление/блокировка).

Несмотря на положительную динамику, 49 баллов - неудовлетворительная оценка, а в отдельных отраслях оценка по прежнему ниже 30 баллов.

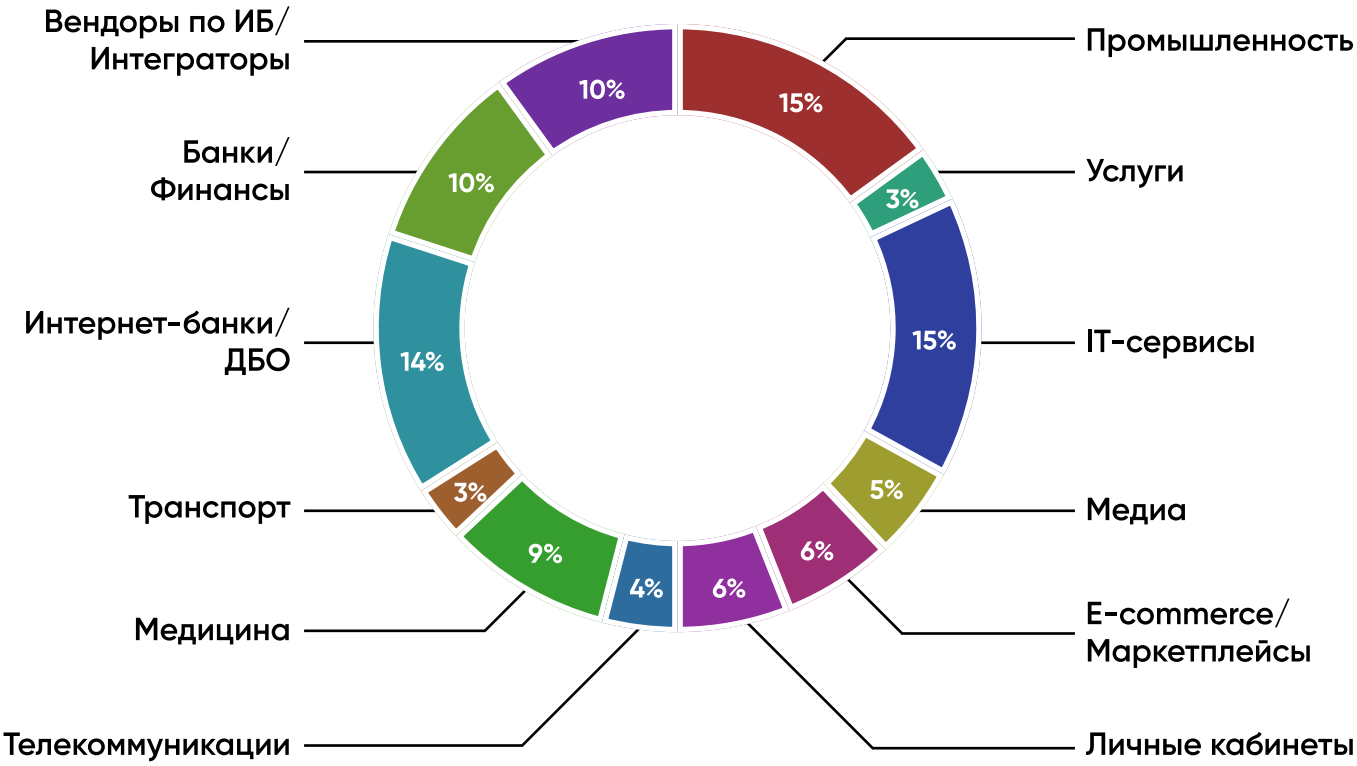
ОБЪЕКТ ИССЛЕДОВАНИЯ

В рамках исследования были проанализированы более 3000 публично доступных frontend-приложений крупнейших российских коммерческих компаний.

Главные страницы (корпоративных сайтов, порталов, личных кабинетов, социальных сетей, маркетплейсов и т.п.) открывались в frontend-песочнице (анализаторе безопасности класса Frontend Application Security Testing (FAST)), фиксирующей составляющие и действия frontend-приложения (в первую очередь поведение JavaScript-кода).

В течение одной минуты на странице эмулировались действия пользователя (скролл, движение мыши). Анализ не оказывал влияния/нагрузки на приложения и основан на общедоступной информации.

Анализ проводился во 2-ом квартале 2026 года, включал приложения из следующих ключевых отраслей.



ОПРЕДЕЛЕНИЕ ОБЩЕЙ ОЦЕНКИ БЕЗОПАСНОСТИ

Для определения уровня информационной безопасности frontend-приложений была разработана Общая оценка безопасности.

Этот показатель определялся по 100-балльной шкале на основании множества критериев.

Критерии и их удельный вес приведены ниже.

Механизмы безопасности

- 13% Конфигурация CSP
- 4% Использование SRI

Элемент iframe

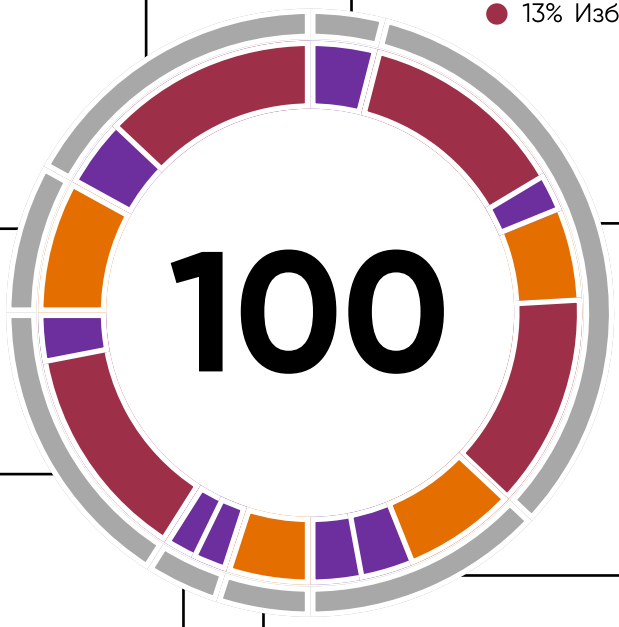
- 8% Наличие iframe зарубежных

Сетевые запросы

- 4% Передача данных за рубеж

Внешние сервисы

- 13% Тег-менеджеры
- 2% Скрипты аналитики
- 5% Скрипты аналитики (зарубежные)
- 13% Избыточный сбор данных



Скрипты

- 3% Скрипты с зарубежных серверов
- 13% Наличие inline скриптов

API браузера (WebAPI)

- 7% Использование eval()
- 3% Чтение cookie из js-кода
- 3% Показ нотификаций ОС

Качество приложения

- 2% Наличие html-комментариев
- 2% Наличие ошибок в консоли

Другое

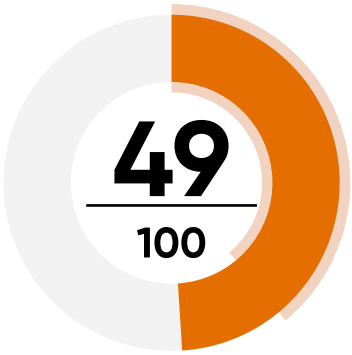
- 5% Прочие факторы

ОБЩАЯ ОЦЕНКА БЕЗОПАСНОСТИ

По результатам проведенного исследования общая оценка безопасности российских frontend-приложений по всем отраслям составила 49 из 100 (в прошлом году результат составил 39).

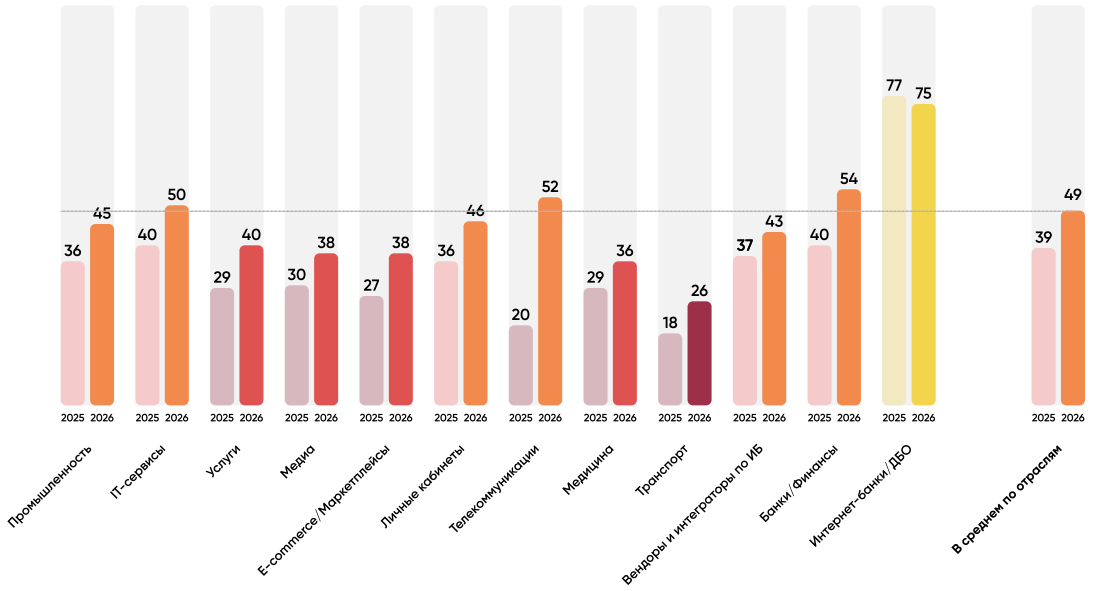
Ожидаемо, максимальный средний показатель оценки в категории Интернет-банки/ДБО - 75 из 100 (в прошлом году результат составил 77). Однако для такой критичной категории приложений это недостаточный показатель.

Так, например, некоторые системы ДБО загружают скрипты из-за рубежа, используют системы аналитики, осуществляющие избыточный сбор данных со страниц, не используют Content Security Policy и т.д.



Общая оценка безопасности

Общий показатель безопасности по отраслям (за I полугодие 2025 и 2026)



Средняя оценка безопасности без учета Интернет-банков/Систем ДБО составляет 42 из 100.

Отдельно следует отметить, что уровень безопасности приложений из категории Банки/Финансы практически не отличается от других отраслей, что весьма критично, т.к. данные приложения являются одной из приоритетных целей для злоумышленников.

Худший показатель безопасности - в категориях Транспорт (26) и Медицина (36).

С учетом того, что многие категории приложений обрабатывают персональные данные пользователей, текущее состояние безопасности российских frontend-приложений следует признать **неудовлетворительным**.

Возможно, в компаниях нет понимания рисков, связанных с frontend-приложениями, и (или) анализ/инвентаризация скриптов/потоков данных не проводится.

Учитывая регулярные крупные инциденты с frontend-приложениями и запрет сбора персональных данных с использованием баз данных за пределами РФ с 1 июля 2025 года, ситуацию необходимо менять в кратчайшие сроки.

ВСЕ ОТРАСЛИ

Ключевые риски

-16 **48%** Загружают скрипты с зарубежных хостов

-12 **59%** Отправляют данные на зарубежные хосты

-8 **18%** Используют Google Tag Manager

Механизмы безопасности браузера

-1 **1%** Использование Subresource Integrity (SRI)

+7 **30%** Использование Content Security Policy (CSP)

+1 **8/100** Оценка конфигурации Content Security Policy (CSP)

Общая оценка безопасности

+10 **49** / 100

Скрипты

Среднее количество всех скриптов

51 | 28 file, 23 inline

Средний размер всех скриптов

3.4 МБ

- Наличие скриптов на внешних хостах — 83%
- Наличие скриптов на зарубежных хостах — -16 48%
- Наличие inline-скриптов — +1 92%
- Наличие скриптов в атрибутах события (onclick, onerror и т.п.) — +1 47%

Внешние сервисы

Веб-аналитика

- Google Analytics — -11 17%
- Яндекс Метрика — 72%
- Яндекс Метрика с Вебвизором — +20 57%
- Битрикс Аналитика — -10 11%

Тег менеджеры

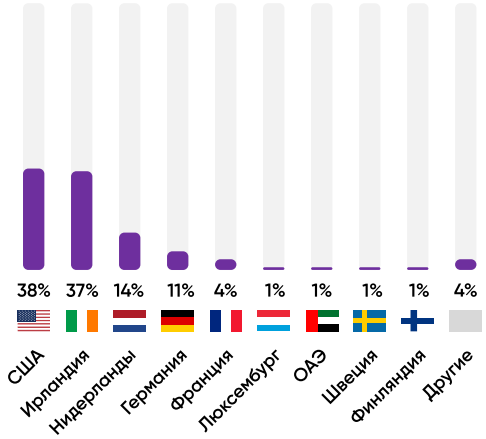
- Google Tag Manager — -8 18%
- Яндекс Тег Менеджер — 10%
- CleverData Tag Manager (РФ) — 5%
- Другие — < 1%

Другие

- Google Tag — -12 26%
- Google reCAPTCHA — -3 10%
- Яндекс Smart Captcha — 6%

Сетевые запросы

- Наличие запросов на зарубежные хосты — -12 59%
- Среднее количество хостов, на которые отправляются запросы с веб-страниц — 13



API браузера (WebAPI)

Среднее количество вызовов API браузера **6002**

- eval() — 45%
- Notification — 1%
- Workers — 23%
- CookieGet — 87%
- CookieSet — 79%
- LocalStorage — 86%
- IndexedDB — 28%

Элемент iframe

-30 **45%** Наличие iframe

-14 **21%** Наличие iframe с зарубежным источником

Качество приложения

+1 **46%** Наличие ошибок в консоли браузера

-2 **82%** Наличие html-комментариев

КРАТКИЙ ОБЗОР ВЫВОДОВ

1. Уровень безопасности российских frontend-приложений остаётся неудовлетворительным (49 из 100)

2. При защите веб-приложений в российских компаниях фокус смещен на защиту бэкенда (WAF, NGFW, API Security, сканеры уязвимостей). Обеспечению безопасности frontend-части веб-приложений практически не уделяется внимания.

3. Низкий уровень безопасности и множество способов монетизации делают frontend-приложения приоритетной целью для злоумышленников, что подтверждается регулярно обнаруживаемыми инцидентами.

4. Более 70% российских компаний рискуют получить штрафы за нарушение 152-ФЗ “О персональных данных” за сбор ПД с использованием зарубежных баз данных и трансграничную передачу данных (КоАП РФ 13.11 ч.8-9 от 1 до 18 млн. руб).

5. Для обеспечения безопасности frontend-приложений необходимо использовать специализированные классы инструментов: FAST-анализаторы (Frontend Application Security Testing) и Frontend Security Observability.

КЛЮЧЕВЫЕ РИСКИ

Далее будут подробно рассмотрены ключевые “явления”, влияющие на безопасность frontend-приложений, и их наличие в приложениях различных отраслей.

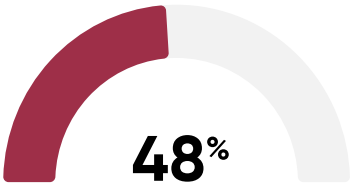
Для каждого “явления” приводятся риски и способы их снижения.

Будут рассмотрены:

- скрипты с зарубежных хостов;
- отправка данных на зарубежные хосты;
- Google Tag Manager (GTM);
- Яндекс Тег Менеджер (ЯТМ);
- Яндекс Метрика с вебвизором;
- функция eval();
- (не)использование Content Security Policy (CSP);
- эффективность конфигурации Content Security Policy (CSP);
- (не)использование Subresource Integrity (SRI);
- HTML-комментарии.

СКРИПТЫ С ЗАРУБЕЖНЫХ ХОСТОВ

Скрипты на веб-странице могут быть подключены двумя способами: инлайн и в виде файла. В случае с инлайн-скриптами, код скрипта указывается прямо в теле страницы. В случае с файловыми скриптами указывается URL файла с JavaScript-кодом. При загрузке страницы браузер выполняет сетевой запрос для загрузки файла и исполняет полученный код.



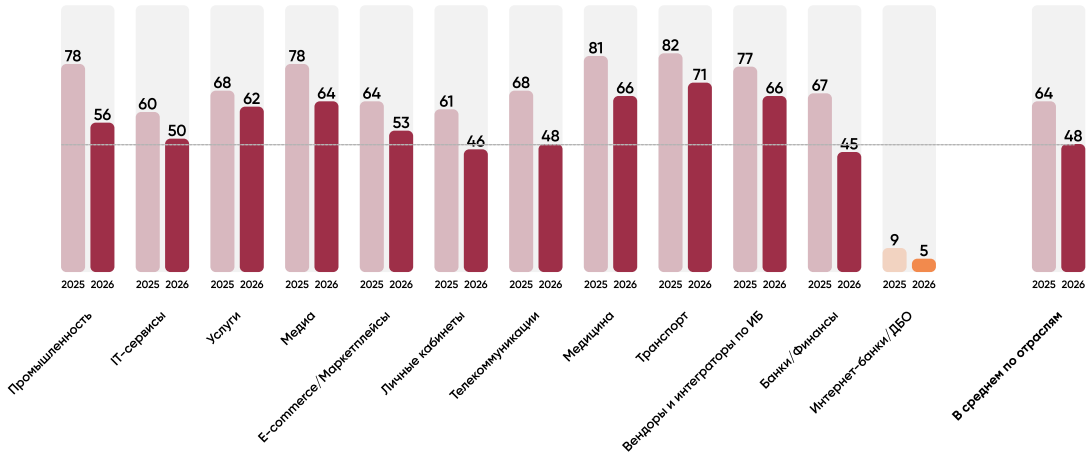
frontend-приложений используют скрипты с зарубежных хостов

В 2000-е годы, когда средний размер js-кода на страницах не превышал 200 КБ, интернет был медленным, а хостинг дорогим, традиционным считалось подключение сторонних JavaScript-библиотек из публичных CDN (Content Delivery Network).

Так, например, подключение библиотеки jQuery в виде файла в CDN (https://code.jquery.com/jquery-3.7.1.min.js, 85 КБ, Fastly CDN) экономило трафик к вашему серверу и в целом ускоряло загрузку веб-страниц за счет возможного кэширования при посещении других сайтов и географически распределенных серверов CDN.

Сегодня средний размер js-кода на странице составляет 3.3 МБ. Крупные компании используют собственные / платные российские CDN. Вашу jQuery давно пора “переместить” на свой сервер, т.к. загрузка скриптов с зарубежных хостов приносит не пользу, а только существенные риски для frontend-приложений.

Использование скриптов с зарубежных хостов по отраслям



Риски

- Доступ к зарубежным хостинг-провайдерам, в том числе CDN, может быть заблокирован / замедлен в РФ, что приведет к обратному эффекту - замедлению загрузки веб-страниц и даже к отказу в обслуживании.
- Зарубежные CDN могут накладывать ограничения для пользователей из РФ и не иметь кэширующих серверов на территории РФ (так, CDN https://www.fastly.com/ сообщает, что не работает в РФ).
- Зарубежные CDN могут подменить содержимое скриптов для пользователей из РФ (например, добавить Protestware).
- Внешний сервер/CDN могут быть скомпрометированы, что может привести к добавлению вредоносного кода в Ваши скрипты.
- Загрузка скрипта из внешнего источника может быть признаком работы вредоносного кода в frontend-приложении.

Что делать?

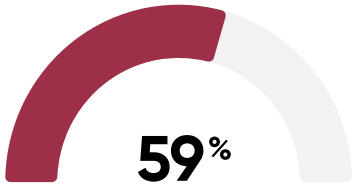
- Проводить регулярную инвентаризацию внешних скриптов.
- Минимизировать количество внешних скриптов.
- Отказаться от использования зарубежных CDN.
- Перенести сторонние js-библиотеки на собственные серверы либо использовать доверенные CDN.
- Контролировать целостность файловых скриптов за счет использования механизмов SRI и CSP (о них смотрите далее в исследовании).

ОТПРАВКА ДАННЫХ НА ЗАРУБЕЖНЫЕ ХОСТЫ

С 1 июля 2025 года сбор персональных данных (ПД) с использованием баз данных, находящихся за пределами территории РФ, не допускаются (152-ФЗ, ст. 18, ч. 5), а с 2023 года запрещена трансграничная передача ПД без уведомления Роскомнадзора.

Идентификаторы пользователей, записанные в cookie, используемые для анализа поведения пользователей, признаются ПД регуляторами многих стран, в том числе РФ.

Если на сайте установлен счетчик посещаемости или система веб-аналитики, то frontend-приложение обрабатывает ПД, даже если их нет на страницах в “явном” виде.



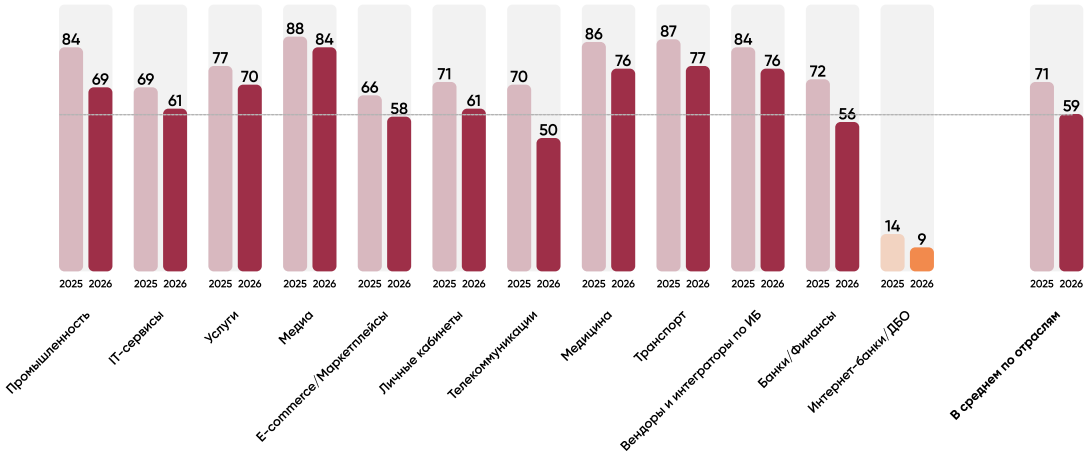
59%

frontend-приложений отправляют данные на зарубежные хосты

Даже при загрузке браузером картинки со стороннего сервера пользователю может быть установлен cookie для дальнейшего отслеживания интересов пользователя (по посещению других сайтов).

Сотрудники РКН могут проверить, в какие страны отправляют данные Ваши веб-страницы в рамках мониторинга, и сверить эту информацию с согласием/политикой обработки ПД на сайте. При несоответствии могут быть применены санкции регулятора.

Отправка данных на зарубежные хосты по отраслям



Риски

- Штрафы за передачу ПД третьим лицам без согласия.
- Штрафы за трансграничную передачу ПД без уведомления РКН.
- Штрафы за сбор ПД с использованием баз данных, находящихся за пределами территории РФ.
- Штрафы за утечку ПД.
- Передача данных на внешние хосты может быть признаком наличия вредоносного кода в frontend-приложении (в т.ч. js-сниффер, js-майнер) или недекларированных возможностей (НДВ) в используемых js-библиотеках / системах управления контентом (CMS).

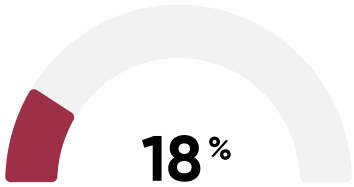
Что делать?

- Проводить регулярную инвентаризацию отправок данных на внешние хосты с веб-страниц.
- Минимизировать/отказаться от трансграничной передачи данных.
- Проводить регулярную инвентаризацию скриптов систем аналитик, тег-менеджеров, фреймов, форм, пикселей отслеживания.
- По результатам инвентаризации проверять актуальность согласия и политики обработки ПД на сайте (перечень третьих лиц, трансграничная передача).

GOOGLE TAG MANAGER (GTM)

Тег-менеджер - внешний сервис и размещаемый на веб-странице скрипт, который позволяет динамически добавлять на веб-страницы другие скрипты/фрагменты по определенным условиям (URL, время, регион, выполнение действия и другие).

Используется маркетологами для добавления кода разных скриптов систем аналитики на разные страницы или выполнения небольших скриптов "достижения цели" при определенных условиях.



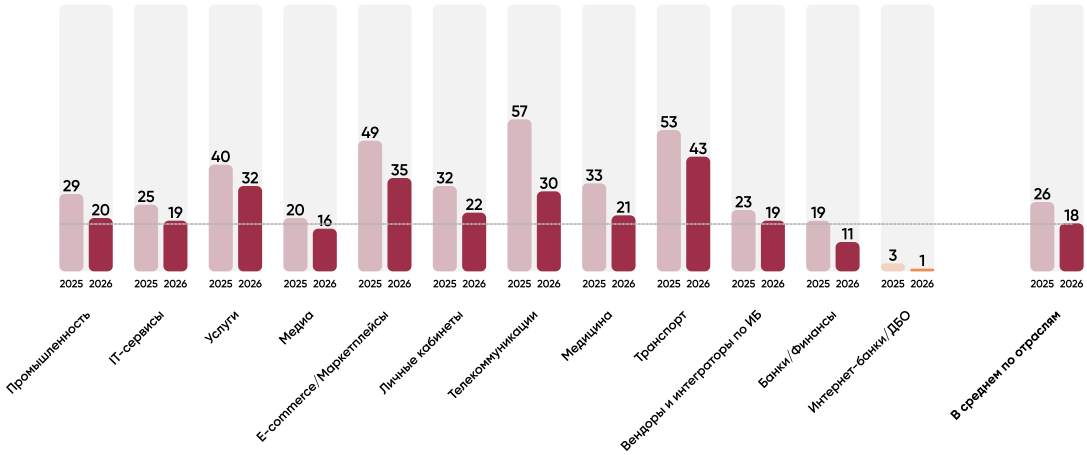
frontend-приложений используют GTM

Один из самых популярных сервисов - Google Tag Manager (GTM). Тег-менеджер - облачный сервис. Скрипты и условия их добавления на страницы конфигурируются в веб-интерфейсе Google.

С точки зрения ИБ тег-менеджер можно назвать "frontend-бэкдором", который позволяет менять исходный код frontend-приложения по заданным условиям. Через GTM на страницы можно добавить js-снифферы, js-майнеры и другие вредоносные скрипты.

Часто доступ к консоли управления GTM есть только у маркетологов, а добавляемые скрипты не контролируются ИБ, что значительно снижает защищенность веб-приложения. В инциденте 2019 года злоумышленники размещали на страницах взломанных сайтов GTM, а через него уже добавляли js-сниффер при загрузке страницы в браузере.

Использование GTM по отраслям



Риски

- Зарубежный сервис, отказ в обслуживании, возможность добавления Protestware со стороны сервиса.
- В случае компрометации Google-аккаунта, злоумышленник получает возможность добавлять вредоносный код на страницы.
- Целенаправленная атака на маркетологов: "установите скрипт новой передовой системы аналитики". В результате вредоносный код появляется в frontend-приложении.
- Злоумышленники могут использовать легитимные инструменты для сокрытия вредоносного кода, например, GTM. При ручном анализе кажется: "это нормально, какой-то скрипт маркетологов".

Что делать?

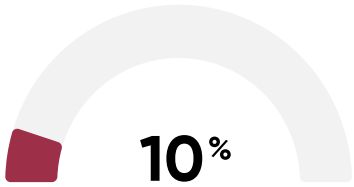
- Инвентаризация скриптов тег-менеджеров (возможно один легитимный, а второй - злоумышленников).
- По возможности отказаться от использования тег-менеджеров.
- Доступ к консолям управления тег-менеджерами должен быть только у ИБ. Процесс управления изменениями и анализа скриптов на вредоносное поведение. Маркетологи не осознают риски.
- Переход на использование отечественных тег-менеджеров (Яндекс Тег Менеджер)
- Анализ поведения frontend-приложения в

ЯНДЕКС ТЕГ МЕНЕДЖЕР (ЯТМ)

Яндекс Тег Менеджер (ЯТМ) - российский тег-менеджер, представленный в июне 2025 года. ЯТМ имеет функциональность, аналогичную Google Tag Manager (добавление на веб-страницы скриптов по заданным триггерам через облачную консоль управления).

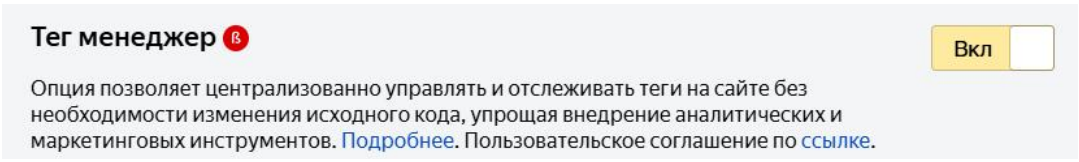
Функции ЯТМ встроены в Яндекс Метрику, для новых счетчиков ЯТМ включен по умолчанию, а для старых достаточно включить функцию в веб-интерфейсе.

Владелец счетчика ЯМ имеет полный доступ к ЯТМ, добавлению произвольных скриптов на веб-страницы и предоставлению доступов к ЯТМ другим пользователям Яндекса.

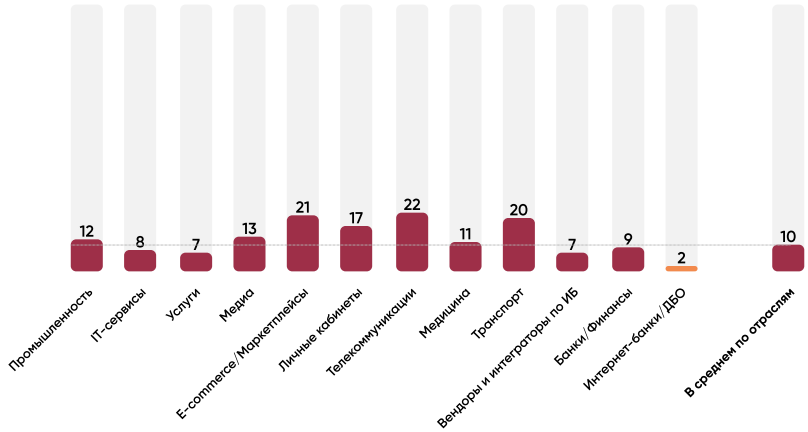


10%

frontend-приложений используют ЯТМ



Использование ЯТМ по отраслям



Риски

- В случае компрометации Яндекс-аккаунта, злоумышленник получает возможность добавлять вредоносный код на страницы.
- Целенаправленная атака на маркетологов: "установите скрипт новой передовой системы аналитики". В результате вредоносный код появляется в frontend-приложении.
- Злоумышленники могут использовать легитимные инструменты (в том числе ЯТМ) для сокрытия вредоносного кода.
- Раннее согласованные счетчики ЯМ с июня 2025 года через ЯТМ могут добавлять в приложение произвольные скрипты, в т.ч. вредоносные.
- Часто владельцами счетчиков ЯМ являются личные Яндекс-аккаунты сотрудников/маркетологов/подрядчиков. Эти аккаунты не управляются корпоративной IDM.
- Уволенный сотрудник/подрядчик могут добавить в ваше приложение js-майнер, рекламные скрипты конкурентов, что приведет к утечке лидов/персональных данных.

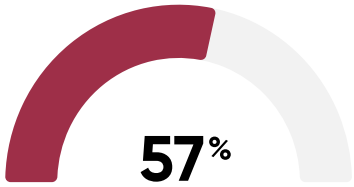
Что делать?

- Инвентаризация счетчиков Яндекс Метрики (возможно один легитимный, а другие - злоумышленников/старых подрядчиков).
- По возможности отказаться от использования тег-менеджеров в критичных приложениях (личные кабинеты и т.п.).
- Проверить, что у всех счетчиков ЯМ владельцем является корпоративный Яндекс-аккаунт.
- Минимизировать доступы к управлению ЯТМ, в частности на добавление новых скриптов.
- Реализовать процесс управления изменениями с проверкой поведения всех добавляемых скриптов.
- Использовать FAST-анализатор для глубокого обнаружения ЯТМ в приложениях, контроля несанкционированных изменений и анализа поведения добавляемых скриптов.

ЯНДЕКС МЕТРИКА С ВЕБВИЗОРОМ

Яндекс Метрика - российская система веб-аналитики. Используется на большинстве российских сайтов. По результатам данного исследования - на 72% frontend-приложений.

Вебвизор - модуль Яндекс Метрики, позволяющий просматривать запись сеансов пользователей практически в виде видеозаписи. Вебвизор собирает и отправляет на серверы Яндекса полный контент веб-страниц, движения мыши, нажатия клавиш пользователем и другие данные.



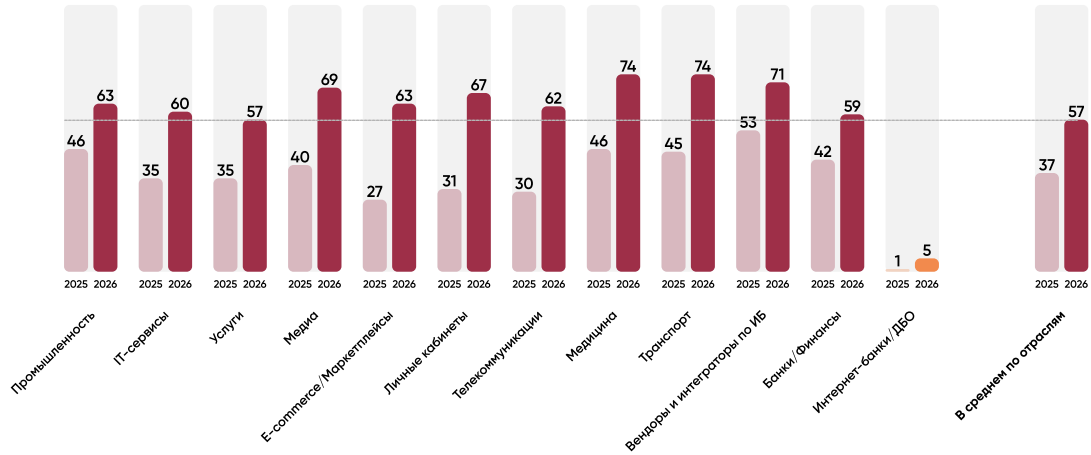
frontend-приложений используют Яндекс Метрику с вебвизором

Использование вебвизора в frontend-приложениях, отображающих конфиденциальную информацию (личные кабинеты, CRM-системы, онлайн-банки, сайты с формами обратной связи), равносильно передаче такой информации на серверы Яндекса с возможностью просмотра в Яндекс-аккаунте, управляющим счетчиком (обычно доступ есть у маркетологов).

Модуль вебвизор включен по умолчанию при создании счетчика Яндекс Метрики. В модуле может быть настроено исключение захвата определенного контента (например, таблица с ПД клиентов в CRM, однако для этого необходима тонкая настройка / указание атрибутов у элементов страницы, содержащих конфиденциальную информацию).

Злоумышленник может разместить на веб-страницах скрипт Яндекс Метрики с вебвизором для кражи данных с помощью данного легитимного инструмента.

Использование Яндекс Метрики с вебвизором по отраслям



Риски

- Избыточная передача ПД в Яндекс.
- Штрафы по 152-ФЗ при некорректно составленных согласиях.
- Раскрытие конфиденциальной информации сотрудникам, имеющим доступ к управлению счетчиком (маркетологи).
- Злоумышленники могут использовать вебвизор для кражи данных с веб-страниц. При ручном анализе кажется: “это нормально, какой-то скрипт маркетологов”.

Что делать?

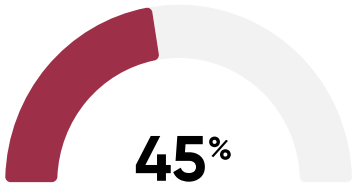
- Отказаться от использования вебвизора.
- Регулярная инвентаризация подключенных скриптов Яндекс Метрики, флага включения вебвизора и проверки атрибутов элементов с конфиденциальной информацией (предотвращение захвата их контента вебвизором).
- Контроль доступа к Яндекс-аккаунту, управляющему счетчиком.
- Проверка на сайте согласия на передачу ПД в Яндекс.

ФУНКЦИЯ EVAL()

JavaScript-функция eval() предназначена для выполнения js-кода из строки.

Во многих инцидентах eval() используется злоумышленниками для динамической дешифровки вредоносного кода перед выполнением с целью сокрытия кода и затруднения анализа безопасности.

В настоящее время практически отсутствуют обоснованные причины использования данной функции в js-приложениях, а ее использование значительно снижает защищенность приложения.



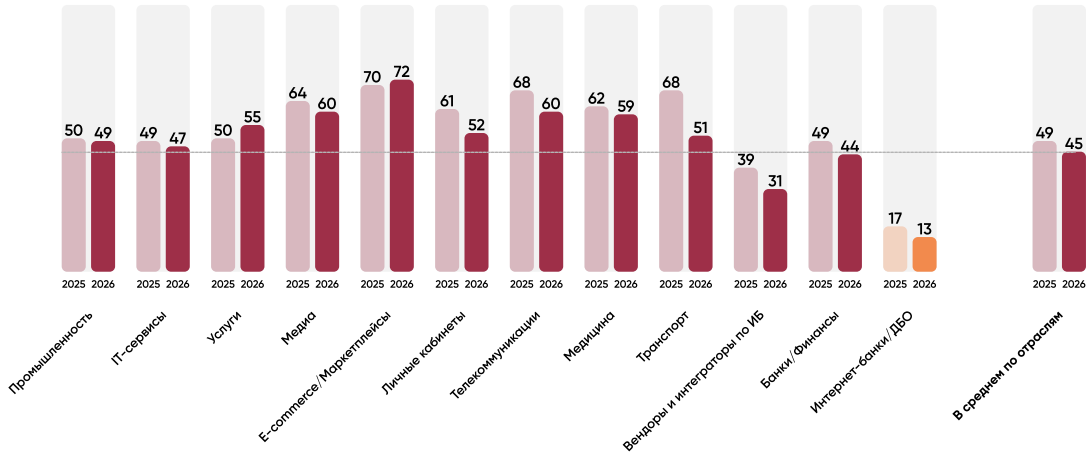
frontend-приложений используют функцию eval()

Функция eval() может легитимно использоваться сложными обфускаторами или, например, скриптами антифрод-систем для затруднения анализа кода злоумышленниками / конкурентами.

Использование eval() в приложении делает невозможным настроить наиболее строгую Content Security Policy (CSP), что также снижает защищенность приложения и оставляет больше возможностей потенциальному злоумышленнику.

Статические анализаторы (SAST) имеют сложности как с анализом кода внутри eval(), так и с обнаружением фактов вызова eval() (например, конструкций вида window['ev' + 'al']('attacker code')). Для этих целей эффективно применять frontend-песочницы/FAST-анализаторы.

Использование функции eval() в frontend-приложениях по отраслям



Риски

- Наличие вызова eval() может являться признаком наличия вредоносного кода в frontend-приложении.
- Снижение защищенности приложения, больше возможностей для злоумышленника при других атаках.

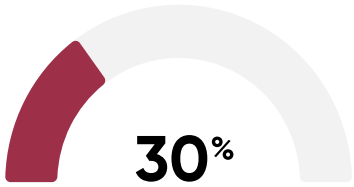
Что делать?

- Отказаться от использования eval().
- Установить запрет на использование функции в CSP, контролировать неизменность CSP, реализовать оповещение ИБ о несанкционированных изменениях CSP.
- Если функция eval() легитимно используется в frontend-приложении (например скрипт антифрод системы), необходимо контролировать вызовы eval() различными скриптами с помощью frontend-песочницы и реагировать на новые (неразрешенные) факты вызова функции

(НЕ)ИСПОЛЬЗОВАНИЕ CONTENT SECURITY POLICY (CSP)

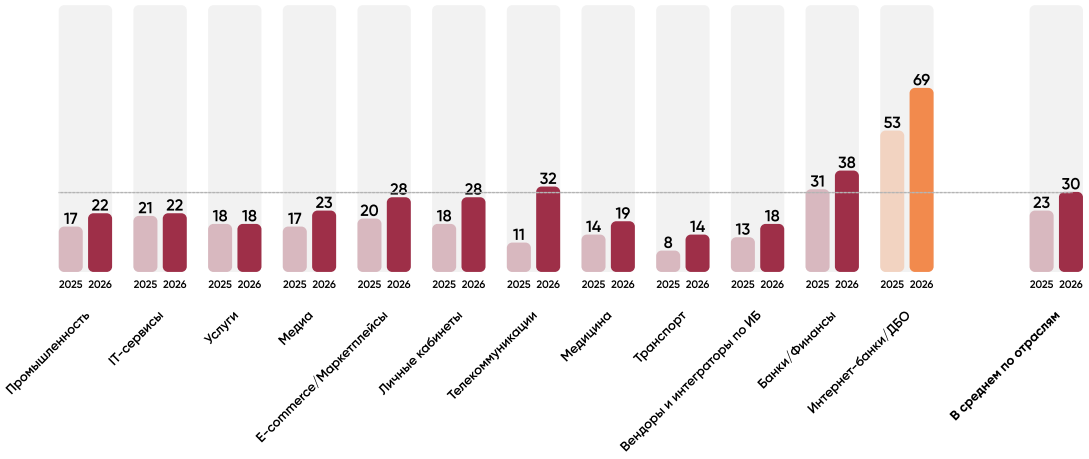
Content Security Policy (CSP) – механизм безопасности, позволяющий создателю frontend-приложения указать браузеру пользователя, из каких источников веб-странице разрешается загружать различные ресурсы (скрипты, картинки, шрифты).

Также CSP позволяет контролировать целостность скриптов и других ресурсов посредством указания в HTTP-заголовке CSP разрешенных хэш-сумм; возможность использования функции eval(); разрешение инлайн-скриптов; хостов, с которыми веб-странице разрешается устанавливать сетевые соединения.



frontend-приложений используют CSP

Наличие заголовка CSP по отраслям



Важно! В спецификации CSP указано: "CSP не предназначена для использования в качестве основной защиты от инъекций контента (в том числе XSS), а подходит для дополнительного уровня безопасности, снижающего последствия от таких атак".

Важно! Даже при самой строгой CSP, в JavaScript-коде есть возможность отправки сетевого запроса с произвольными данными на любой внешний хост. То есть, если вредоносный код попал в frontend-приложение по любому из возможных векторов, у злоумышленника всегда есть возможность отправить украденные со страницы данные на свой хост.

Статья: Content Security Policy (CSP) защитит от js-сниффероов и утечек?

В статье приведен код стенда, на котором демонстрируется отправка данных с веб-страницы на сторонний хост при самой строгой CSP.

https://habr.com/ru/articles/849402/

В реальных бизнес-приложениях "самой строгой" CSP практически не встречается. Если на веб-страницах установлена система веб-аналитики, то в CSP разрешается загрузка скриптов с соответствующего внешнего хоста. Этим могут воспользоваться злоумышленники для похищения данных через размещение своего счетчика данной системы аналитики (как в кейсе с Яндекс Метрикой с вебвизором).

Также есть проблемы с разграничением ответственности внутри компании за настройку и контроль изменений CSP. Саму CSP формируют разработчики (т.к. хэш-сумма js-файлов фиксируется сразу после релиза), HTTP-заголовок может добавляться DevOps на уровне единой точки входа в приложение / реверс-прокси. CSP может быть отключена/ослаблена на этих уровнях, поэтому у ИБ / AppSec должен быть независимый контроль за изменением заголовков CSP.

В связи с этими проблемами и актуальностью угроз js-сниффероов соответствующие требования были включены в стандарт PCI DSS 4.0.1 (действует с 31.03.2025): "11.6.1 Контроль изменений на платежных страницах, контроль изменения HTTP-заголовков, оповещение персонала о несанкционированных изменениях".

ЭФФЕКТИВНОСТЬ КОНФИГУРАЦИИ CONTENT SECURITY POLICY (CSP)

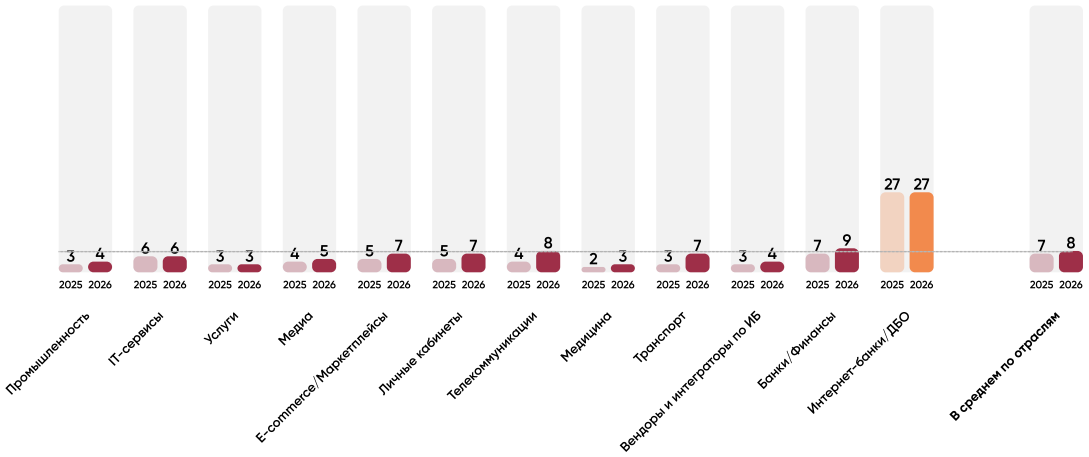
Само по себе наличие заголовка CSP не делает приложение безопасным. Так, в рамках исследования были обнаружены приложения, в CSP которых не было указано ни одной “запрещающей” директивы либо такие директивы разрешали всё (*).

Для определения эффективности конфигурирования CSP была разработана методика оценки. Наиболее строгая конфигурация CSP (разрешающая обращения только на собственный хост и контролирующая целостность ресурсов по хэш-суммам) соответствует максимальной оценке (100). Далее за каждое исключение или ослабление CSP оценка снижалась.

8/100

Средняя оценка конфигурации CSP

Эффективность конфигурации CSP по отраслям



Проблемы CSP

- Может контролировать целостность файловых ресурсов (скриптов, стилей) только совместно с механизмом Subresource Integrity (SRI).
- Не может заблокировать возможность утечки данных на сторонние хосты даже при самой строгой конфигурации.
- Скрипты внешних сервисов могут меняться без предупреждения в рамках обновления версий, что приведет к их блокированию при использовании контроля целостности в CSP.
- Неработоспособность frontend-приложения при ошибках в конфигурации CSP.
- Различная поддержка браузерами.
- CSP может быть отключена при компрометации бэкенда.
- Разграничение ответственности за конфигурирование и контроль целостности CSP (разработчик, DevOps, AppSec, ИБ).
- Механизм уведомления о нарушениях CSP требует наличие бэкенд-части для получения от браузеров пользователей.
- Ложные срабатывания в уведомлениях о нарушении политики (“шум” от плагинов браузеров, скриптов антивирусов и т.п.)

Усиление CSP

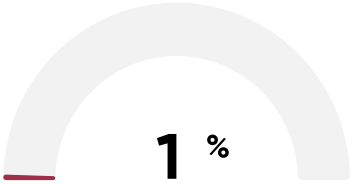
- Использовать CSP совместно с механизмом SRI (для контроля целостности файловых скриптов/стилей).
- Использовать все возможные директивы CSP.
- Внедрить процесс управления изменениями с согласованием ИБ.
- Использовать инструмент для инвентаризации скриптов, хостов-получателей данных и уведомления о несанкционированных изменениях CSP (например, FAST-анализатор).
- Использовать инструмент дифференциального выявления нарушений, например, одному скрипту разрешено использовать функцию eval(), а всем остальным запрещено. CSP может разрешить/запретить только для всех скриптов.

(НЕ)ИСПОЛЬЗОВАНИЕ SUBRESOURCE INTEGRITY (SRI)

Subresource Integrity (SRI) - механизм контроля целостности файловых ресурсов frontend-приложения (скриптов, стилей).

Разработчик frontend-приложения может указать в атрибуте integrity соответствующего тега (например, скрипта) хэш-сумму файла. После загрузки файла браузер рассчитает его хэш-сумму, если она не совпадет с указанной в атрибуте, то данный скрипт не будет выполнен.

SRI эффективен для контроля целостности файловых ресурсов, размещенных на сторонних серверах (например, в CDN), для защиты от несанкционированного изменения.



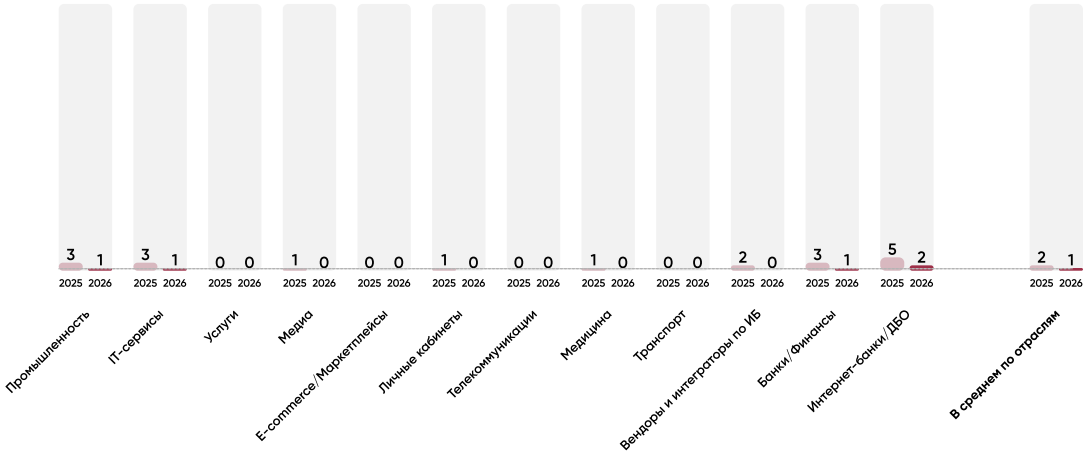
1 %

frontend-приложений используют SRI

Не защищает от попадания вредоносного кода в frontend-приложение через npm-зависимости, добавлении вредоносного кода в приложение при компрометации бэкенда и других векторов.

Также не подходит для контроля внешних js-сервисов (например, систем веб-аналитики), т.к. такие сервисы могут вносить изменения в свои скрипты без предупреждения в рамках развития функциональности.

Использование SRI по отраслям



Проблемы SRI

- Применим только к элементам script и link.
- Не применим для инлайн-скриптов, скриптов в атрибутах событий.
- Не может проверить, что атрибут integrity указан у всех элементов, для этого необходимо использовать внешнее средство контроля / CSP.
- Не имеет механизма уведомления о нарушениях.
- Может привести к “тихому” отказу части функций приложения.
- Неэффективен для защиты от большинства векторов проникновения вредоносного кода в приложение.
- Скрипты внешних сервисов могут меняться без предупреждения в рамках обновления версий, что приведет к их блокированию.
- Разграничение ответственности за конфигурирование (разработчик, DevOps, AppSec, ИБ).

Усиление SRI

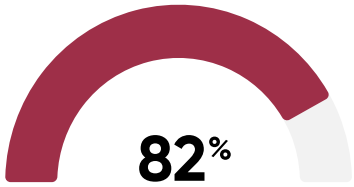
- Использовать совместно с CSP (может контролировать целостность инлайн-скриптов).
- Внедрить процесс управления изменениями с согласованием ИБ.
- Использовать инструмент для инвентаризации скриптов и уведомления о несанкционированных изменениях (например, FAST-анализатор).

HTML-КОММЕНТАРИИ

Одна из лучших практик OWASP гласит: “В продуктивной среде код, доступный пользователю, должен быть очищен от комментариев, которые могут раскрыть реализацию функций бэкенда или другую чувствительную/конфиденциальную информацию”.

Однако, если открыть в браузере исходный код любой веб-страницы (Ctrl + U в Chrome), то практически всегда на страницах присутствуют комментарии.

Комментарии в HTML и JavaScript-коде в продуктивной среде не приносят пользы никому, кроме злоумышленников и конкурентов.

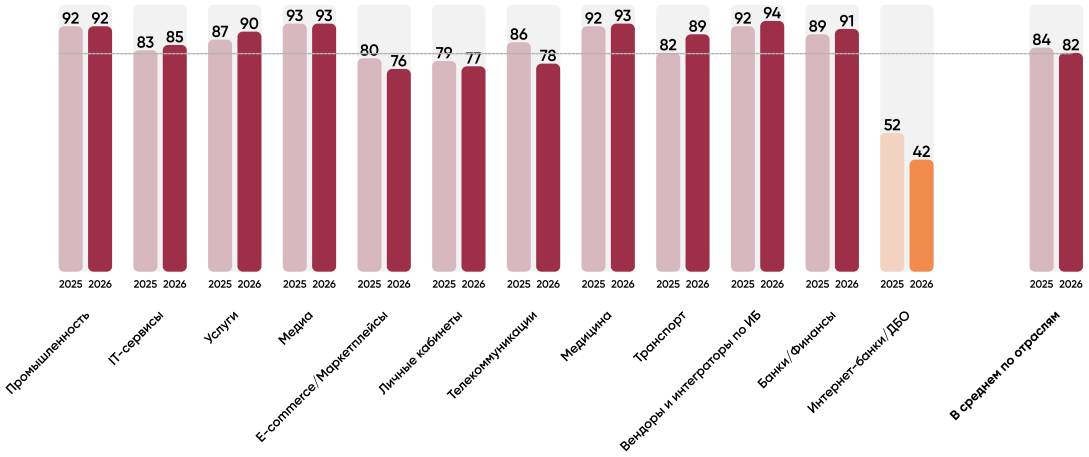


frontend-приложений имеют оставленные html-комментарии

Комментарии могут быть “безобидными”, а могут и раскрывать учетные записи, контакты разработчиков или реализацию функций бэкенда.

Все комментарии должны удаляться перед публикацией приложения, либо должна проводиться инвентаризация комментариев с контролем их добавления/изменения со стороны ИБ.

Наличие html-комментариев по отраслям



“Безобидные” комментарии

```
<!-- Yandex.Metrika counter -->
<script type="text/javascript" >
   (function(m,e,t,r,i,k,a){m[i]=m[i]||
   m[i].l=1*new Date();
   for (var j = 0; j < document.scripts
   k=e.createElement(t),a=e.getElements
   (window, document, "script", "https:

   ym(26993514, "init", {
        clickmap:true,
        trackLinks:true,
        accurateTrackBounce:true,
        webvisor:true
   });
</script>
<noscript><div><img src="https://mc.yan
<!-- /Yandex.Metrika counter -->
```

Раскрытие чувствительных данных

```
<span class="DDKf1c">
<div class="eqA2re UnOTSe" style="height:26px;width:26px">
</div>
</span>

<!-- Показания датчиков загрязнения воздуха -
замена нулей случайными числами
в пределах заданного диапазона - просьба заказчика  -->

<div class="CA5RN"><div><span class="VuuXrf">
</span>
</div>
<div class="byrV5b"><cite class="gLRx3b tjvcx GvPZzd cHaqb"
```

```
</div>
</div>
<!-- Проверка второго фактора аутентификации
временно отключена по требованию заказчика.
Учетная запись manager3412 / Qwerty123(@
-->
</div>
</div>
```

ИНТЕРПРЕТАЦИЯ ОБЩЕЙ ОЦЕНКИ БЕЗОПАСНОСТИ

Общая оценка безопасности рассчитывалась на основании критериев, которые снижают защищенность frontend-приложения, увеличивают поверхность атаки и расширяют возможности злоумышленников.

Контекст приложения / бизнес-требования при расчете оценки не учитывались, поэтому оценку необходимо правильно интерпретировать.

Само по себе наличие запросов на зарубежные хосты, наличие подключенного Google Tag Manager (GTM), вызовов eval() и другое не является инцидентом, если это является бизнес-требованием и вы знаете об их наличии.

Не снижает риск:

- «Знаем конечно, у нас разрешено»;
- «Да, у нас разрешена Яндекс Метрика с вебвизором»;
- «Всё хорошо, у нас eval() не запрещен в CSP, т.к. он используется скриптом нашей антифрод-системы».

При таком подходе frontend-приложение по прежнему находится в "слепой" зоне для ИБ, и в случае попадания вредоносного кода по любому вектору Вы можете узнать об инциденте через недели/месяцы случайно после жалоб пользователей или из СМИ.

Что снижает риск:

- регулярный автоматизированный глубокий анализ поведения frontend-приложения;
- контроль изменений;
- оповещения персонала о несанкционированных изменениях;
- проверка каждые 6-12 часов.

Это снижает риски, выводит frontend-приложение из "слепой" зоны, делает обнаружение инцидентов оперативным, а приложение - безопасным.

При таком подходе общая оценка безопасности Вашего frontend-приложения может быть увеличена на 40 баллов.

Для решения данной задачи можно, например, использовать FAST-анализатор DPA FAST Analyzer с запуском анализа поведения приложения по расписанию.

ИНТЕРПРЕТАЦИЯ ОБЩЕЙ ОЦЕНКИ БЕЗОПАСНОСТИ

Далее приведены примеры кейсов, в которых бизнес-требования к приложению и отсутствие глубокого анализа поведения frontend-приложения со стороны ИБ/AppSec может привести к масштабным инцидентам / утечкам данных без возможности оперативного обнаружения и реагирования.

1

Скрипт корпоративной антифрод-системы легитимно использует функцию eval(), поэтому ее использование не запрещено в CSP. После обновления версий зависимостей, скомпрометированная транзитивная зависимость начинает использовать eval() для запуска вредоносного кода.

- ✗ При ручном анализе / без анализа нелегитимный вызов не будет обнаружен.
- ✓ При автоматизированном глубоком анализе выявляется количество вызовов, инициаторы, аргументы функции и т.д.

2

В приложении легитимно используется Google Tag Manager (GTM). Ваш сервер был взломан, злоумышленники добавили еще один скрипт GTM, к которому они имеют доступ.

- ✗ При ручном анализе это может быть пропущено (“У нас разрешен GTM”).
- ✓ При автоматизированном глубоком анализе будет обнаружен второй скрипт GTM и его вредоносные действия, запросы и т.д.

3

Злоумышленник добавляет во frontend-часть личного кабинета еще один скрипт Яндекс Метрики, к которой он имеет доступ, с включенным вебвизором. Далее извлекает собранные ПД/логины/пароли клиентов из UI Яндекс Метрики.

- ✗ При ручном анализе это может быть пропущено (“У нас Яндекс Метрика разрешена”).
- ✓ При автоматизированном глубоком анализе будет обнаружен второй скрипт Яндекс Метрики, флаг включения вебвизора, значительное повышение трафика / количества запросов к хосту Яндекса.

ИТОГОВЫЕ ВЫВОДЫ

1. Уровень безопасности российских frontend-приложений является неудовлетворительным (49 из 100).

2. Несмотря на лучший показатель безопасности в категории онлайн-банки (75 из 100) по сравнению с другими отраслями, данная оценка является недостаточной с учетом уровня критичности данных приложений.

3. При защите веб-приложений в российских компаниях фокус смещен на защиту бэкенда (WAF, NGFW, API Security, сканеры уязвимостей). Обеспечению безопасности frontend-части веб-приложений практически не уделяется внимания.

4. Низкий показатель безопасности в первую очередь связан с непониманием угроз (“приложение и так работает в браузере пользователя, что там взламывать?”).

5. Риски, связанные с frontend-приложениями, субъективно оцениваются низко.

6. Встроенные механизмы безопасности браузеров в российских приложениях не используются либо используются крайне неэффективно (оценка 8 из 100).

7. Работа frontend-приложений в “слепой” зоне для ИБ и низкая оценка рисков приводит к увеличению времени присутствия вредоносного кода в приложении и максимальной монетизации для злоумышленников.

8. Низкий уровень безопасности и множество способов монетизации делают frontend-приложения приоритетной целью для злоумышленников, что подтверждается регулярно обнаруживаемыми инцидентами.

9. Контроль/управление изменениями при использовании тег-менеджеров, систем веб-аналитики должен осуществляться с участием ИБ-специалистов, т.к. такие сервисы используются злоумышленниками для сокрытия вредоносного кода / кражи данных.

10. Масштаб утечек персональных данных через js-снифферы может быть сопоставим с компрометацией бэкенда за счет длительного времени присутствия вредоносного кода в приложении (недели, месяцы).

11. Международные и российские регуляторы начинают включать требования по безопасности скриптов/frontend-приложений в требования (НСПК, PCI DSS 4.0.1) и рекомендации (НКЦКИ).

12. Более 59% российских компаний рискуют получить штрафы за нарушение 152-ФЗ “О персональных данных”, т.к. их сайты/frontend-приложения производят отправку данных пользователей за пределы РФ либо осуществляют сбор ПД с использованием баз данных, размещенных за пределами РФ (использование зарубежных систем веб-аналитики).

13. Утечка персональных данных в frontend-приложении либо невыполнение требований 152-ФЗ по трансграничной передаче ПД с 1 июля 2025 года могут привести к значительным штрафам для российских компаний: КоАП РФ 13.11 ч.8-9 от 1 до 18 млн. руб., КоАП РФ 13.11 ч.12-14 от 3 до 15 млн. руб.

ИТОГОВЫЕ ВЫВОДЫ

14. Значительное число инцидентов, связанных с добавлением вредоносного кода в npm-зависимости frontend-приложений, показывает необходимость проведения анализа безопасности приложения в процессе безопасной разработки (РБПО, SSDLC, SDL, DevSecOps, FrontSecOps) до релиза.

15. Применение классических анализаторов безопасности SAST, DAST, SCA, OSA не снижает актуальные риски для frontend-приложений.

16. Применение классических средств защиты веб-приложений (WAF, NGFW, сканер уязвимостей) не снижает актуальные риски для frontend-приложений.

17. Инциденты, связанные с компрометацией сторонних js-сервисов и монетизации взлома бэкенда через добавление злоумышленниками вредоносных скриптов на frontend, показывают необходимость проведения регулярного анализа поведения приложения (например, каждые 6-12 часов).

18. Ручной анализ поведения frontend-приложения неэффективен, а злоумышленники усложняют техники сокрытия вредоносного кода, в том числе с использованием легитимных скриптов (тег-менеджеры, системы аналитики).

19. Для обеспечения безопасности frontend-приложений необходимо использовать специализированные классы инструментов: FAST-анализаторы (Frontend Application Security Testing) и системы мониторинга поведения приложения в реальном времени (Frontend Security Observability).

РЕКОМЕНДАЦИИ

1. Проведите ручной анализ поведения Ваших frontend-приложений (из каких скриптов состоят, где хранятся скрипты, на какие хосты/страны отправляются запросы со страниц, какие системы веб-аналитики используются)

2. Проверьте соответствие согласий на обработку ПД на сайтах, политик обработки ПД фактической ситуации (перечень третьих лиц, трансграничная передача ПД).

3. По возможности откажитесь от использования зарубежных систем веб-аналитики. Проверьте, отправлено ли уведомление о трансграничной передаче ПД в Роскомнадзор.

4. Разработайте модель угроз для frontend-приложений/веб-страниц. Вы можете использовать бесплатный сервис для моделирования угроз от DPA Analytics: https://dpa-analytics.ru/frontend-threat-model

5. Проверьте, кто в компании имеет доступ к управлению тег-менеджерами, системами веб-аналитики. Внедрите процесс управления изменениями с согласованием ИБ.

6. Проведите глубокий анализ поведения frontend-приложений/веб-страниц с помощью FAST-анализатора.

7. При наличии в компании собственной разработки ПО внедрите этап проверки на безопасность frontend-части веб-приложений с помощью анализа поведения в процесс безопасной разработки, в идеале в виде автоматизированной проверки в CI/CD.

8. Внедрите проведение регулярного (каждые 6-12 часов) анализа поведения frontend-приложений/веб-страниц в продуктивной среде и/или используйте решения класса Frontend Security Observability. Настройте оповещения об обнаруженных инцидентах и реагируйте на них.

ПРОДУКТЫ И СЕРВИСЫ DPA ANALYTICS

Онлайн-сервис для моделирования угроз для frontend-приложений

Бесплатный сервис для создания модели угроз для Вашего frontend-приложения с оценкой рисков, подробным перечнем мер противодействия, их обоснованием их эффективности и планом внедрения.

https://dpa-analytics.ru/frontend-threat-model

DPA FAST Analyzer (Frontend Application Security Testing)

Анализатор безопасности frontend-приложений класса FAST для проведения анализа поведения приложения в виде периодических аудитов и/или использования в процессе РБПО/SSDLC / DevSecOps для выпуска приложений Secure by Design. Обнаруживает закладки, НДВ и аномалии в js-приложениях.

https://dpa-analytics.ru/dpa-frontend-application-security-testing-fast-analyzer

DPA Frontend Observability Platform (FOP)

Выполняет мониторинг работы client-side приложений в реальном времени, контролирует целостность приложений, предотвращает утечки информации в браузерах пользователей.

https://dpa-analytics.ru/dpa-frontend-observability-platform

Telegram-канал FrontSecOps

В канале публикуются разборы инцидентов, лучшие практики по безопасной разработке frontend-приложений, разборы инструментов, аналитика угроз, а также актуальные риски и методы защиты современных js-приложений.

https://t.me/FrontSecOps

dpa-analytics.ru

t.me/FrontSecOps

DPA
ANALYTICS