Браузер пользователя – идеальное место для кражи данных и атак. Как обнаружить утечки там, где WAF не видит?

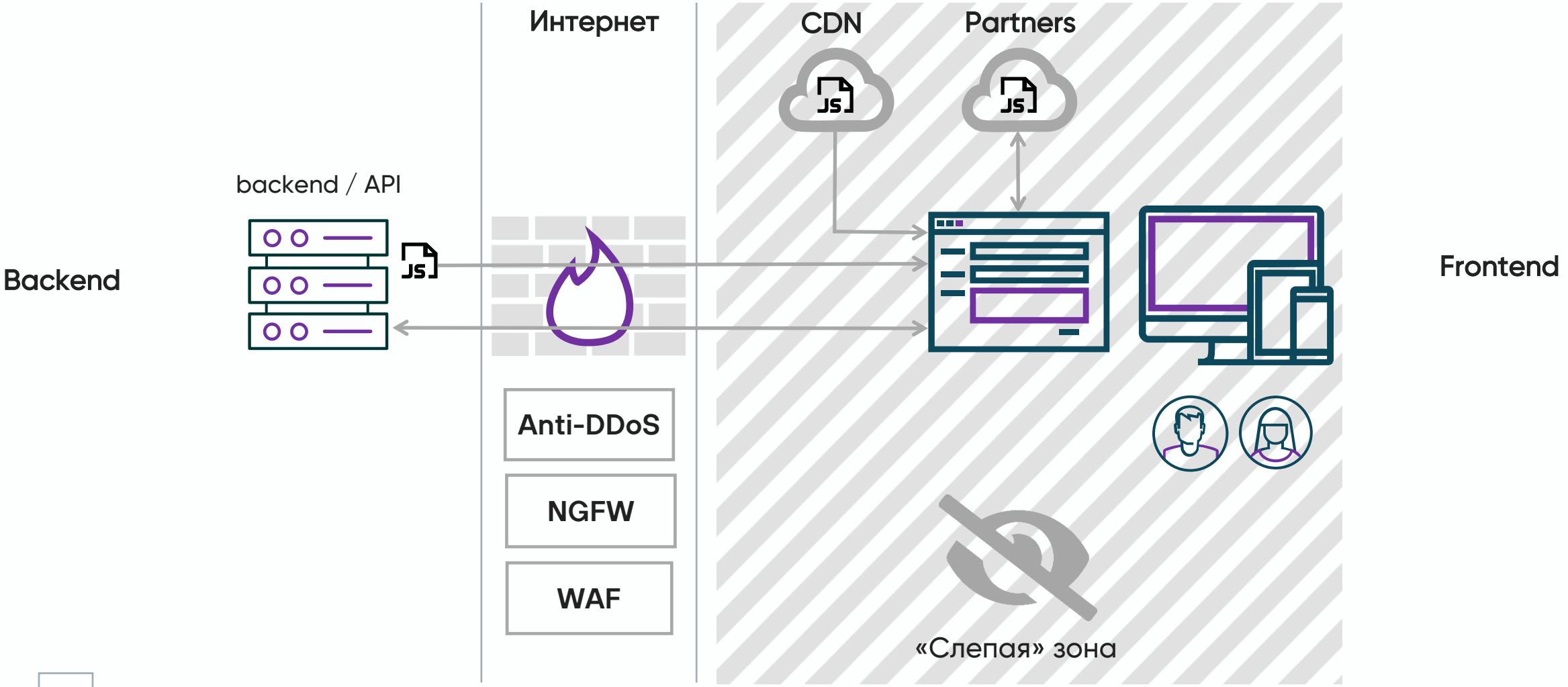
Или остановим Supply Chain атаки на фронтенде

Михаил Парфенов
Application Security Architect

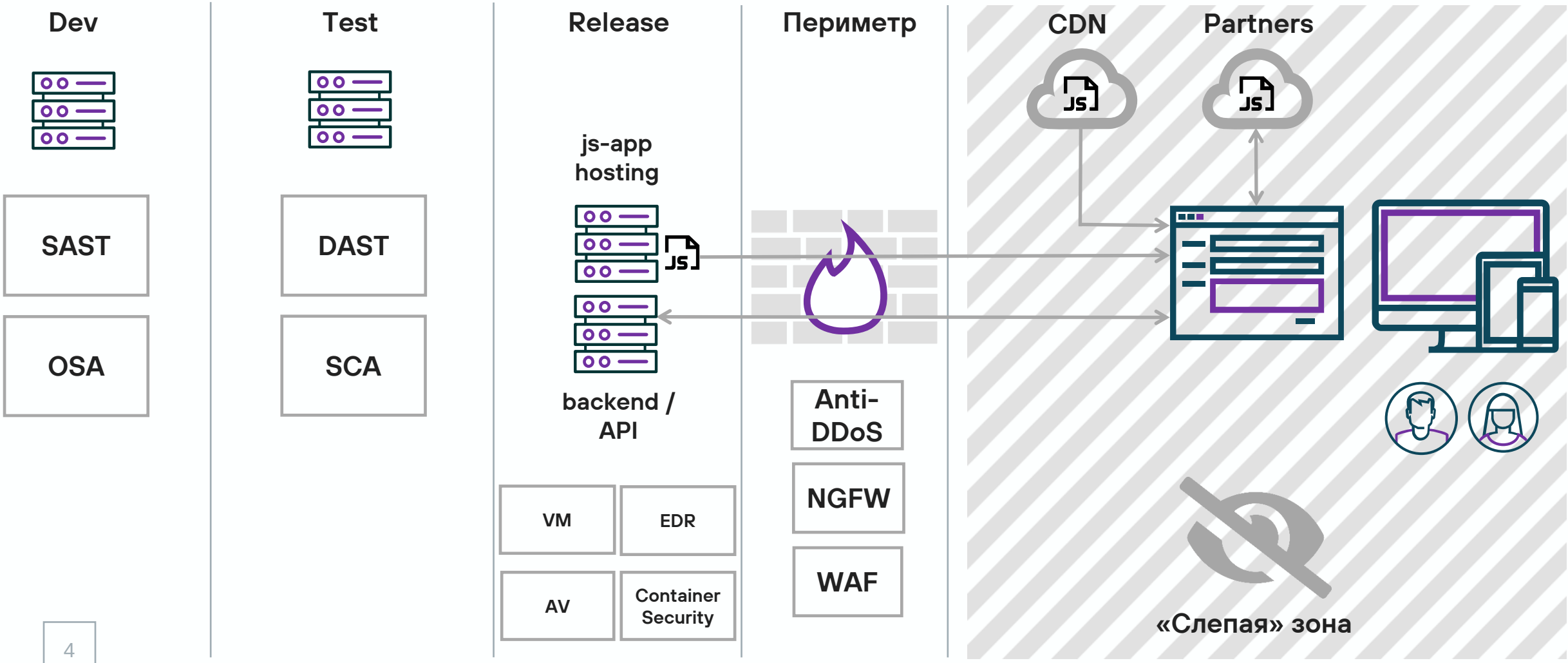
Обо мне

- 10 лет – в ИБ (управление рисками, vulnerability management, безопасность веб-приложений)
- 5 лет - Application Security Architect, DevSecOps
- Исследую методы поведенческого анализа frontend-приложений в DevSecOps (FAST, frontend-sandbox, frontend observability)
- Telegram-канал @FrontSecOps

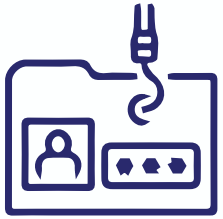
Backend или frontend?



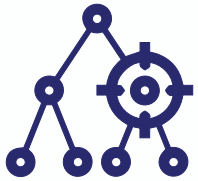
Безопасность веб-приложений



В чем выгода злоумышленника от внедрения вредоносного кода в frontend-приложение?



Сбор и кража критичных данных со страниц web-приложения



Загрузка на страницу других скриптов злоумышленника

Выполнение действий от имени пользователя

Показ пользователю фишинговых баннеров

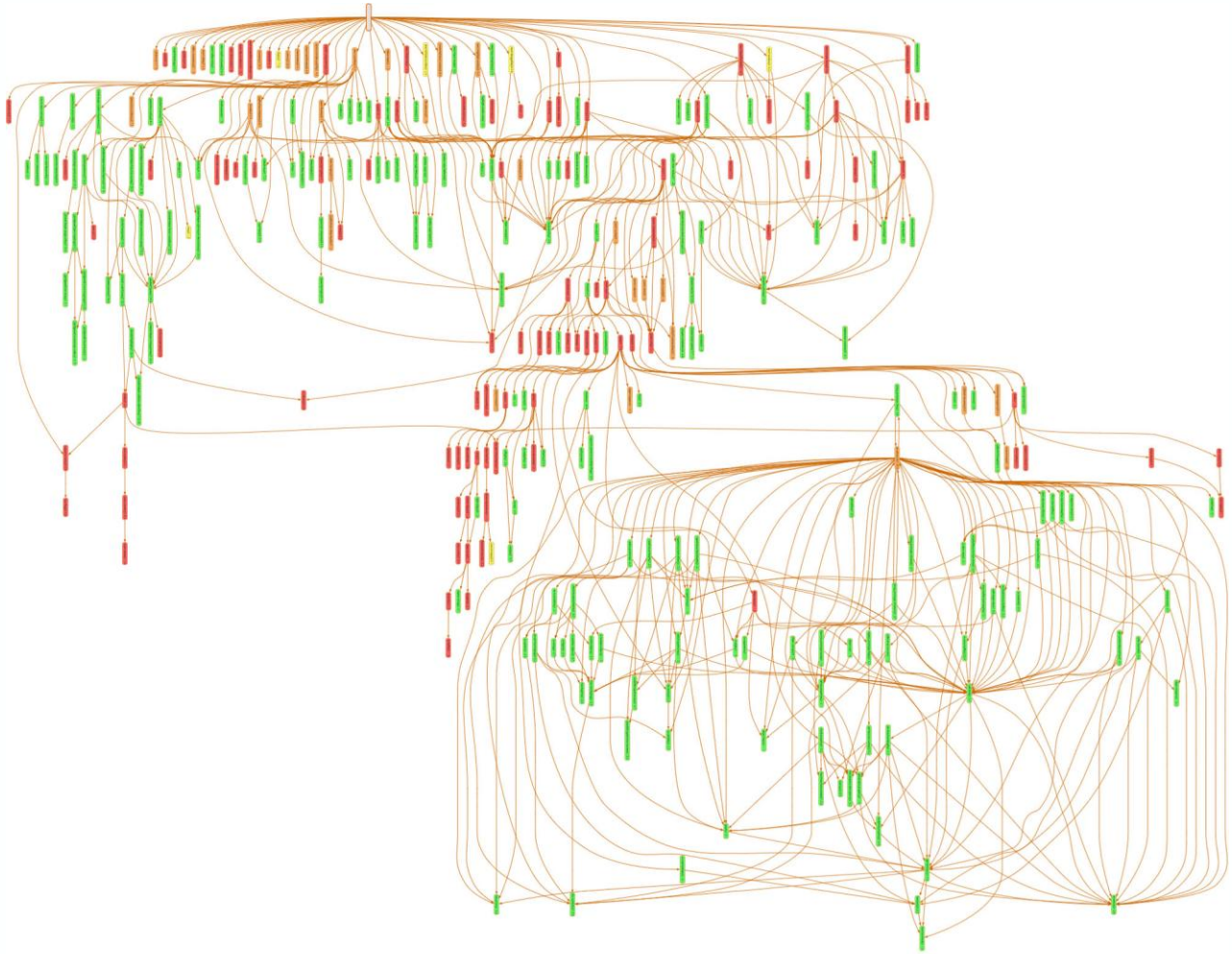
Майнинг криптовалюты в браузере пользователя

Заражение устройства пользователя через уязвимости браузера

Как вредоносный код может попасть в frontend-приложение?

Вектор		Инциденты
Компрометация внешнего js-сервиса, подключенного на веб-страницах (системы аналитики, счетчики, онлайн-чаты и т.п.)	Supply Chain Attack	2022 - Взлом сервиса onthe.io, в js-скрипт добавлен вредоносный код, затронуло сайты СМИ «Коммерсантъ», Forbes, РБК, ТАСС, Известия и других крупных российских компаний
Зависимости js-приложения / сторонние библиотеки с новыми версиями	Supply Chain Attack	2024 – вредоносный код в библиотеке Polyfill.js 2024 - вредоносный код в библиотеке lottie-player (LottieFiles) 2024 - вредоносный код в библиотеке solana/web3.js
Компрометация учетной записи от Google Tag Manager (или его отечественных аналогов)	Supply Chain Attack	2021 - В 316 интернет-магазинах обнаружен js-сниффер, скрытый в Google Tag Manager
Компрометация веб-сервера		2018 – British Airways – размещен js-сниффер, похищающий данные банковских карт
Умышленное размещение кода сотрудником / подрядчиком (разработчик, маркетолог, администратор сайта)		2019 – по информации НКЦКИ, были обнаружены факты размещения js-майнеров на сайтах государственных и муниципальных организаций
Копирование кода из недоверенных источников, генерация кода «плохой» нейросетью		-

Зависимости в JavaScript-приложениях



Количество

94

Глубина

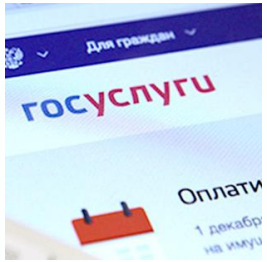
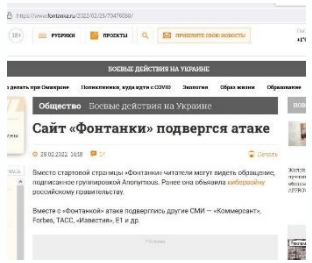
15

Размер (МБ)

12

Минификация и обфускация

Инциденты

			Magento	Google Tag Manager		Polyfill.io
Год	**2017**	**2018**	**2019**	**2021**	**2022**	**2024**
Инцидент	Размещены iframe с неизвестными доменами в Нидерландах	Злоумышленник встроил в одну из js-библиотек js-сниффер	В 100 000+ интернет-магазинов встроен js-сниффер	В 316 интернет-магазинах обнаружен js-сниффер, скрытый в Google Tag Manager	Внедрен код на сайты СМИ «Коммерсантъ», Forbes, РБК, ТАСС, «Известия» и других крупных российских компаний	Внедрен вредоносный код в библиотеку Polyfill.js. Код выполнялся на > 350 000 веб-приложений.
Вектор	N/A	Взлом через уязвимость	Взлом через уязвимость в CMS Magento	Уязвимости CMS: WordPress, Shopify, BigCommerce	Взломан внешний сервис статистики onthe.io, изменен код js-скрипта	Supply chain attack. Код внедрен владельцами библиотеки.
Время присутствия	N/A	15 дней	5 месяцев	N/A	1-3 дня	> 4 месяцев
Последствия	N/A	Похищены данные банковских карт 380 000 клиентов	Похищены персональные данные клиентов (1.5 млн посетителей / день), банковские карты 500 000 клиентов.	Похищены данные банковских карт	Неработоспособность ресурсов. Политические лозунги на страницах.	Редирект пользователей мобильных устройств на сайты онлайн-букмекеров.
Ущерб	N/A Устранено через 4 часа после публикации статьи Dr. Web	2 280 000 000 £ + штраф 20 000 000 £ по GDPR		N/A	N/A	N/A Неработоспособность сайтов после блокировки домена.

Frontend-приложения

1. Работают в «слепой» зоне для ИБ
2. Средства защиты и анализаторы ИБ не обнаруживают актуальные угрозы
3. Время присутствия вредоносного кода – недели / месяцы в известных инцидентах
4. Часто игнорируются ИБ-специалистами
5. Максимальная монетизация для злоумышленника

Требования регуляторов

НКЦКИ «Рекомендации по повышению уровня защищенности российских web-приложений» № ALRT-20220311.1 от 11 марта 2022 г.

19. **Перед использованием** на web-ресурсах JavaScript-кода, подгружаемого со сторонних ресурсов, **осуществлять его проверку на предмет вредоносного воздействия** на отображаемую в браузерах пользователя информацию и возможность кражи аутентификационных данных и файлов-cookie пользователей.

20. Осуществлять периодическую проверку хэш-сумм, используемых JavaScript. В случае изменения хэш-сумм отключать использование JavaScript на сайте и **выполнять повторную проверку функциональности**.

НСПК PCI DSS COMPLIANT

PCI DSS 4.0.1 (Вступили в силу 31.03.2025)
Выполнение требований стандарта указано в Программе безопасности ПС «Мир»

6.4.3 Все скрипты платежных страниц, которые загружаются и выполняются в браузере пользователя, управляются следующим образом:
- Реализован метод подтверждения **авторизации каждого скрипта**.
- Реализован метод, обеспечивающий **целостность каждого скрипта**.
- Актуальная **инвентаризация всех скриптов** с письменным обоснованием необходимости каждого из них.

11.6.1 Обнаружение и реагирование на несанкционированное изменение платежных страниц:
- Контроль **изменений на платежных страницах**
- Контроль **изменений HTTP-заголовков**
- Оповещение персонала о несанкционированных изменениях

Что необходимо контролировать?

1

Скрипты и другие активные элементы

Особое внимание на скрипты/фреймы со сторонних источников / зарубежных серверов!

2

Сетевые запросы, выполняемые js-приложением в браузере

Особое внимание на зарубежные хосты (трансграничная передача ПД)!

3

API-браузера, используемое js-приложением

Особое внимание на eval(), clipboard API, Notification API, WASM, Payment API и другие

“Единственное место, где можно обнаружить изменения и признаки вредоносной активности – это браузер пользователя, где страница полностью собрана и выполнен весь JavaScript-код”

PCI DSS 4.0.1

Способы контроля

1

Ручной анализ

- Долго
- Необходимо выполнять все Use Case при каждом анализе
- Ограниченная видимость

2

Решения класса Frontend Application Security Testing (FAST)

- Анализатор
- Автоматизированное выполнение Use Case
- Глубокий анализ на контентном слое браузера

3

Решения класса Frontend Observability

- JS-агент работает на страницах приложения в браузерах пользователей
- Контроль в реальном времени

Frontend Application Security Testing (FAST)

SAST

- JavaScript-код может быть выполнен «на лету» из зашифрованного текста, что не выявляется SAST.
- Также из анализа обычно исключаются сторонние библиотеки, что снижает покрытие кода анализом до 95%.

DAST

- Обычно DAST анализирует frontend-приложения для определения API-эндпойнтов бэкенда и нескольких видов XSS.
- Поведение приложения не анализируется, т. к. оно не отличимо от нормальных бизнес-функций.

SCA / OSA

- Анализ зависимостей обнаруживает известные уязвимости, но не НДВ.
- JS-код может динамически подгружать модули прямо в браузере. Анализ внешних js-сервисов не производится.

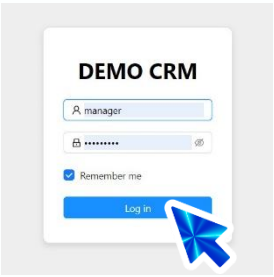
FAST

- Выполняет весь js-код в реальном браузере, включая динамически загруженный.
- Анализирует поведение js-кода во время выполнения реальных Use Case.
- Сравнивает полученный профиль поведения с разрешенным.

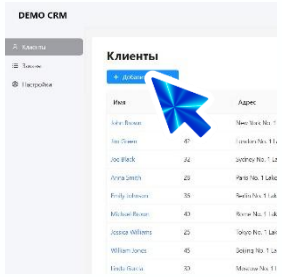
Frontend Application Security Testing (FAST)

Открытие страницы

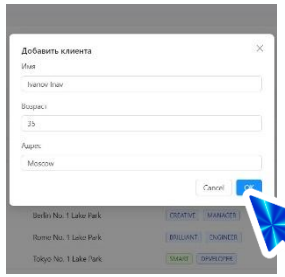
Ввод логина, пароля, клик по кнопке



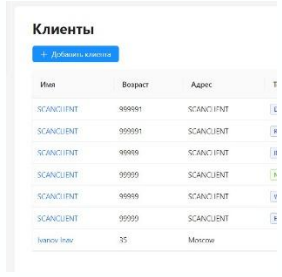
Успешная аутентификация, редирект



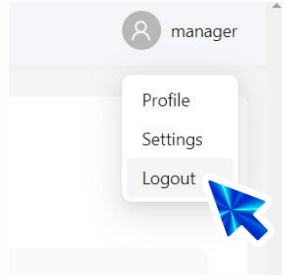
Открытие формы создания клиента, заполнение данных



Клик по кнопке, клиент успешно добавлен



Клик по кнопке Logout, закрытие страницы



Автоматизированное выполнение E2E-сценария (Use Case)

Элементы

script, iframe, embed, form и др.

Запросы

xhr, fetch, img, websocket и др.

API браузера

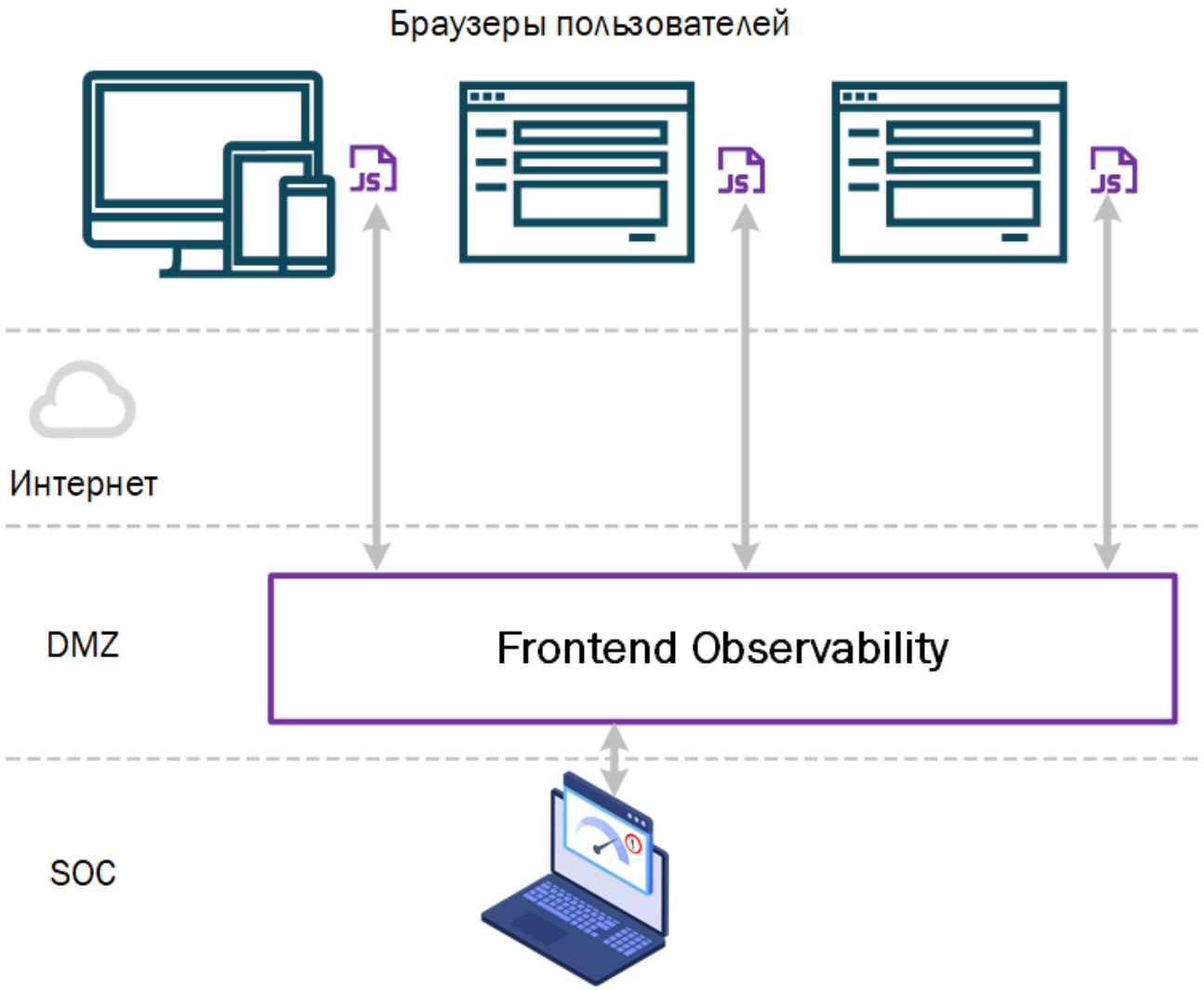
eval, clipboard, geolocation, cookie, notification и др.

Software Bill of Behavior (SBOB)

Контентный слой браузера

Frontend Observability Platform (FOP)

- JS-агент, подключенный на страницы
- Инвентаризация скриптов и других активных элементов в реальном времени
- Мониторинг всех сетевых запросов js-приложения
- Выявления фактов появления на страницах конфиденциальных данных
- Применение правил корреляции
- Выявление критичных инцидентов
- Отправка событий в SIEM / SOC



Модель зрелости обеспечения ИБ frontend-приложений

- Ручной анализ
- Периодическая инвентаризация всех скриптов

- Регулярный анализ приложения в frontend-sandbox (FAST-анализаторе), круглосуточно / каждый час.
  - Инвентаризация всех скриптов / активных элементов
  - Карта сетевых запросов
  - Используемые API-браузера
- Реагирование на любые отклонения от доверенной политики
- Особое внимание после новых релизов приложения

- Мониторинг поведения frontend-приложения в реальном времени:
  - Скрипты / активные элементы
  - Все сетевые запросы браузера
- Контроль объема / состава данных, передаваемых сторонним сервисам, контроль трансграничной передачи данных
- Выявление аномалий
- Реагирование на срабатывания правил корреляции (инциденты, затрагивающие значительное число пользователей)

Безопасное frontend-приложение

Задачи средств защиты веб-приложений

WAF

- Обнаружение признаков атак во входящих HTTP-запросах пользователей по правилам / сигнатурам

FAST / FOP

- Обнаружение утечек информации в frontend-приложениях в браузере
- Обнаружение НДВ / вредоносного кода в js-зависимостях
- Обнаружение злоупотреблений доступом системами аналитики и другими внешними js-сервисами

NGFW

- Обнаружение сетевых атак

Сканер уязвимостей

- Обнаружение пакетов ОС, ПО с известными уязвимостями
- Обнаружение уязвимостей backend-приложения

Что делать?

- Ответить на вопрос: «Я знаю/уверен, что делает frontend-приложение прямо сейчас? Куда отправляет данные?»
- Выполнять мониторинг/контроль поведения frontend-приложений
- Использовать средства автоматизации (FAST/FOP)
- Согласование с ИБ:
  - изменения js-кода;
  - новые хосты, с которыми взаимодействует js-приложение;
  - новые сторонние js-сервисы;
  - новые API-браузера, используемые js-кодом.

Telegram-канал FrontSecOps

- Разбор инцидентов
- Лучшие практики
- Frontend Observability
- DevSecOps для frontend-приложений
- Обзоры инструментов

@FRONTSECOPS

Спасибо!

Михаил Парфенов

Application Security Architect

TG: @mkparfenov

https://t.me/FrontSecOps

@FRONTSECOPS