WEB SECURITY GUIDE • FROM ZERO

На вашем сайте не хватает 7 важных строк кода.

Из-за отсутствующих HTTP security headers ваш сайт становится лёгкой целью для атак.



Хакеры это замечают.

@lionsoulbatyshchikov

ЛИСТАЙ ДАЛЬШЕ →

WEB SECURITY BASICS • FROM ZERO

Что такое HTTP header?

Каждый запрос отправляет сообщение на сервер, и сервер возвращает ответ. HTTP headers — это метки в этих сообщениях.



ПРЕДСТАВЬТЕ, ЧТО ВНУТРИ ПОСЫЛКИ

Что внутри
`Content-Type: text/html`

Показывает тип передаваемого контента (например, HTML, JSON, изображение и т.д.).

Сохраняем контекст
`Set-Cookie: id=a1b2`

Позволяет серверу сохранять данные о вашей сессии (например, через Cookie).

Используйте защищённый канал
`Strict-Transport-Security`

Сообщает браузеру, что сайт доступен только по HTTPS (защита от downgrade атак).

Инструкции по безопасности

Security headers — это правила, которые сервер даёт вашему браузеру, чтобы повысить безопасность.

Маленькие строки. Большая защита.

ЛИСТАЙ ДАЛЬШЕ →

@lionsoulbatyshchikov

WEB SECURITY BASICS • FROM ZERO

Разбираем структуру HTTP ответа

Сервер отвечает на ваш запрос статусной строкой, а затем набором HTTP headers и телом ответа.

server response

```
HTTP/1.1 200 OK                                          STATUS
Content-Type: text/html                                  GENERAL
Content-Length: 5120
Strict-Transport-Security: max-age=31536000              SECURITY
Content-Security-Policy: default-src 'self'
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
-------------------------------------------------------  BLANK LINE
<!DOCTYPE html>                                          BODY
<html>...
```

ФОРМАТ КАЖДОГО ЗАГОЛОВКА ОДИНАКОВЫЙ

Referrer-Policy: **strict-origin**

NAME
название заголовка
(что регулирует)

VALUE
значение
(насколько строго)

Имя: значение

Каждый заголовок на отдельной строке. Сначала имя, затем двоеточие и значение.

Несколько значений

Используйте точку с запятой, чтобы задать несколько значений в одном заголовке. Например: a; b; c

Пустая строка

Пустая строка отделяет заголовки от тела ответа (body). После неё начинается контент.

Маленькие детали — большая безопасность.

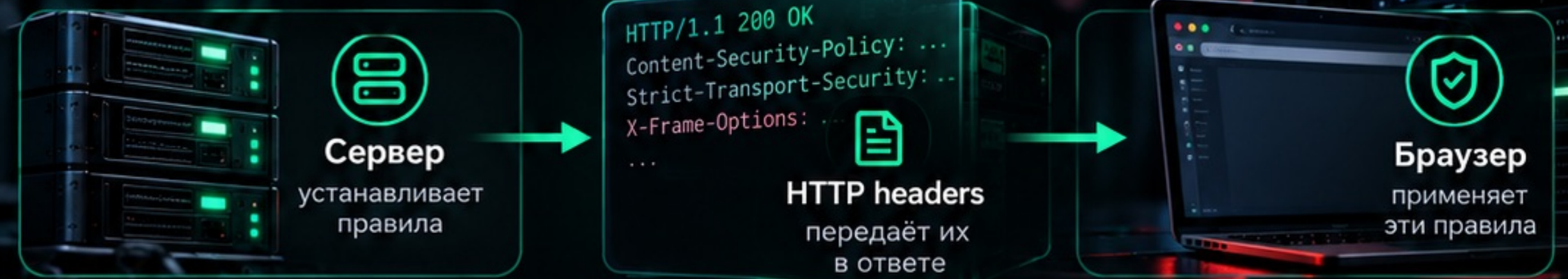
ЛИСТАЙ ДАЛЬШЕ →

@lionsoulbatyshchikov

WEB SECURITY BASICS • FROM ZERO

Почему HTTP заголовки важны?

Они задают правила, которые сервер передаёт браузеру, чтобы повысить безопасность и снизить риск атак.



ОДИН ЗАГОЛОВОК — ОДНА ЗАЩИТА

HSTS Strict-Transport-Security	→	**Принудительный HTTPS** Не позволяет перейти на HTTP (защита от downgrade атак).	05
CSP Content-Security-Policy	→	**Блокирует XSS** Контролирует, какие источники контента можно загружать и выполнять.	07
X-Frame-Options	→	**Защита от clickjacking** Запрещает или ограничивает встраивание вашего сайта в <iframe> на других ресурсах.	10
nosniff X-Content-Type-Options	→	**Правильный тип файлов** Запрещает браузеру “угадывать” тип контента, снижая риск выполнения вредоносных файлов.	11
Referrer-Policy	→	**Контроль утечки данных** Ограничивает, какие Referrer передаются, чтобы не раскрывать внутренние URL.	12
Permissions-Policy	→	**Доступ к чувствительным API** Контролирует использование камеры, микрофона, геолокации и других возможностей браузера.	13
COOP • COEP Cross-Origin APIs	→	**Изоляция и защита от утечек** Снижают риск cross-origin атак и утечки данных между окнами/ресурсами.	14

Небольшие заголовки. Большая защита.

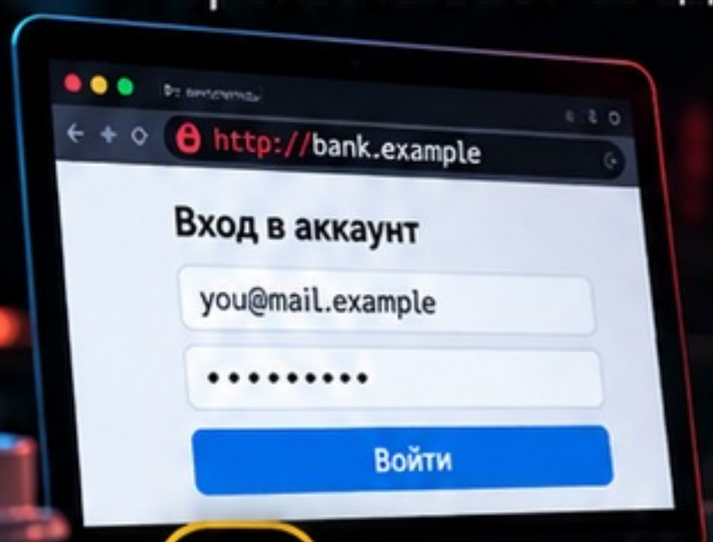
@lionsoulbatyshchikov

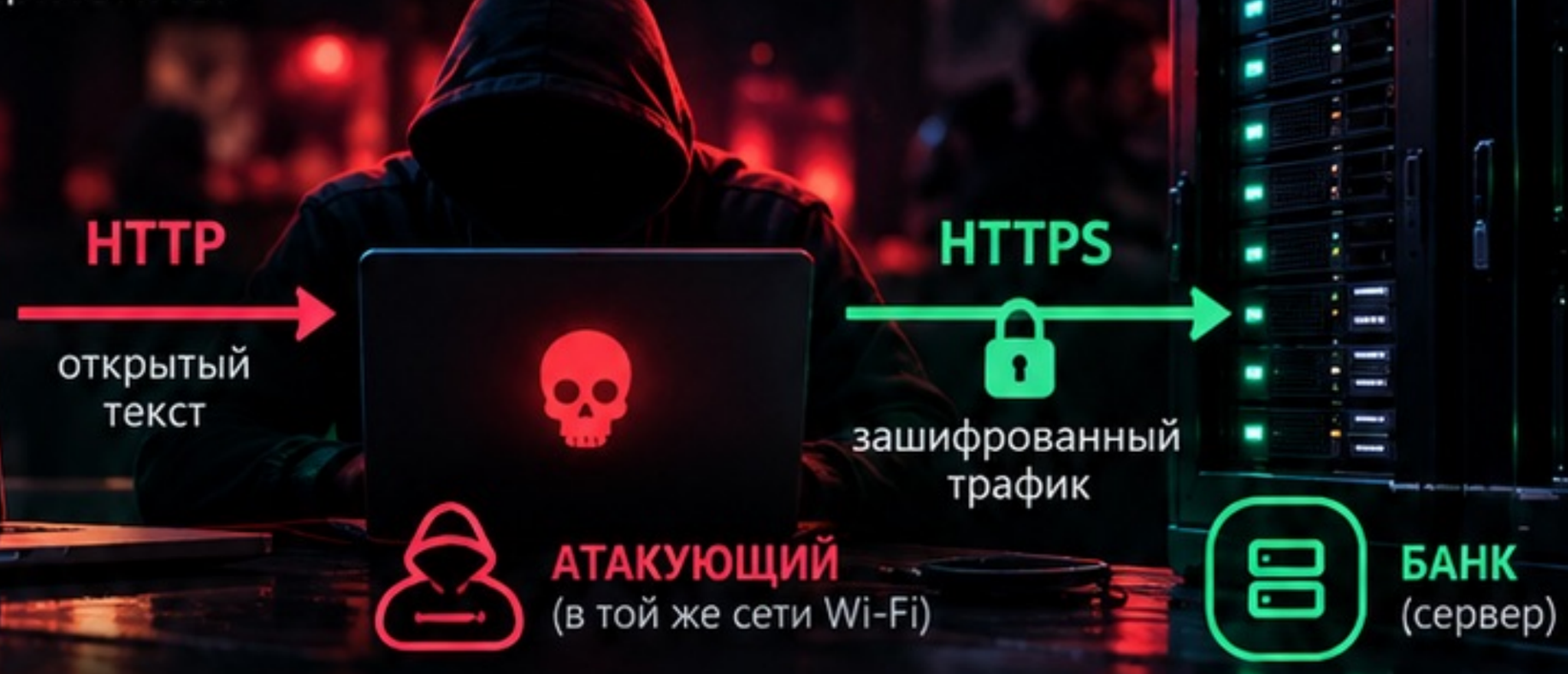
ЛИСТАЙ ДАЛЬШЕ →

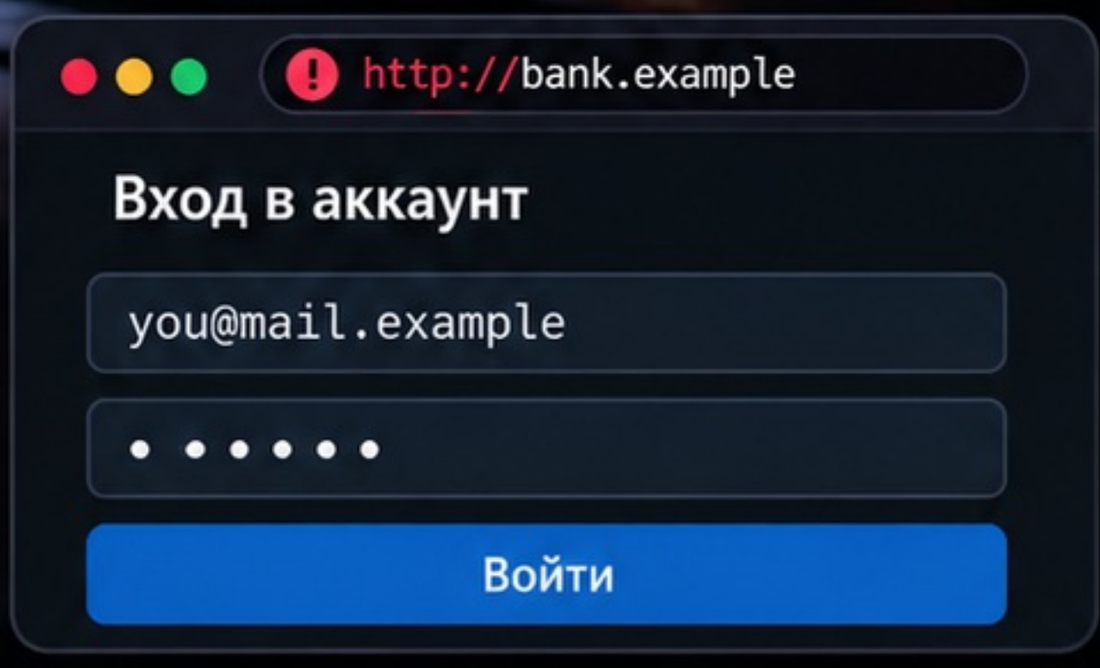
HTTP HEADERS · HSTS

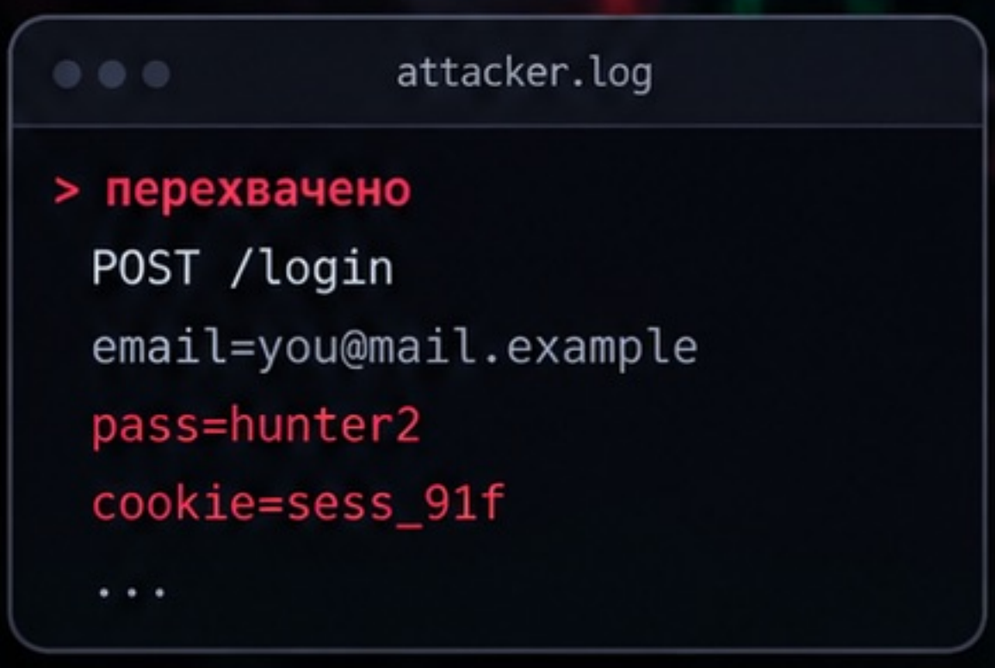
Атака: SSL stripping

Вы подключаетесь к публичному Wi-Fi и вводите bank.example без https://. Кто-то в той же сети перехватывает соединение.









1. **Вы не указываете https://**
   Первый запрос уходит по HTTP (открытый текст).
2. **Атакующий перехватывает соединение**
   Он блокирует переход на HTTPS и подменяет трафик.
3. **Вы остаетесь на HTTP**
   Вся введённая вами информация (логины, пароли, cookies) читается атакующим.

Один заголовок — большая защита.

@lionsoulbatyshchikov

ЛИСТАЙ ДАЛЬШЕ →

HTTP HEADERS • HSTS

Решение: только HTTPS

Заголовок **HSTS** говорит браузеру: всегда используй HTTPS для этого сайта и его поддоменов.



Strict-Transport-Security

```
Strict-Transport-Security:
  max-age=31536000;                # 1 год в секундах
  includeSubDomains;               # применять к поддоменам
  preload                          # включить в preload list
```

max-age=31536000

Браузер запоминает правило на 1 год. Значение указывается в секундах.

includeSubDomains

Правило действует и на все поддомены (например: shop., api., mail.).

preload

Позволяет добавить домен в список браузеров заранее — чтобы они всегда использовали HTTPS, даже при первом визите.

РАЗНИЦА ЗАМЕТНА С ПЕРВОГО ВИЗИТА

Без preload

1-й визит по HTTP (уязвимо) → Следующие визиты авто-переход на HTTPS

С preload

Сразу и всегда авто-переход на HTTPS

Совет:

Начните с `max-age=300` (5 минут) для тестирования и затем постепенно увеличивайте значение.

Один заголовок — большая защита.

ЛИСТАЙ ДАЛЬШЕ →

@lionsoulbatyshchikov

HTTP HEADERS • CSP

Атака: XSS

Злоумышленник оставляет комментарий со скриптом. Сайт сохраняет его, и когда вы открываете страницу — скрипт выполняется в вашем браузере, крадя ваши данные.





Злоумышленник оставляет комментарий со скриптом

Сайт сохраняет его как обычный текст

Вы открываете страницу

Ваш cookie отправляется к злоумышленнику

В чём проблема?

Браузер не может отличить ваш скрипт от чужого, если нет ограничений — он выполнит любой скрипт на странице.

Как помогает CSP?

Браузер проверяет список разрешённых источников (CSP) и блокирует подозрительные скрипты.

```
console
✓ Загружено:  /js/app.js  (same origin: allowed)
✗ Заблокировано: evil.example/x.js  (не в списке)
✗ Заблокировано: inline <script>  (в отзыве)
```

Совет: Начните с `max-age=300` (5 минут) для тестирования, затем постепенно увеличивайте значение.

@lionsoulbatyshchikov

HTTP HEADERS • CSP

Guest list в Content-Security-Policy

Один заголовок — много правил. Вы задаёте, какие источники контента разрешены для разных типов файлов и куда их можно загружать.

Content-Security-Policy

```
Content-Security-Policy:
  default-src 'self';                  # правило по умолчанию
  script-src 'self' cdn.example;       # где можно брать JavaScript
  img-src 'self' data:;                # откуда можно загружать изображения
  frame-ancestors 'none'               # кто может встраивать страницу в <iframe>
```

DIRECTIVE = ЧТО ЗА ФАЙЛ · VALUE = КТО МОЖЕТ ЕГО ЗАГРУЖАТЬ

default-src

Правило по умолчанию для всего, что не описано отдельно.

script-src

Источник для JavaScript. Самое важное правило.

img-src

Откуда можно загружать изображения и иконки.

connect-src

Разрешённые адреса для fetch, XHR и WebSocket.

style-src

Источник для CSS стилей (stylesheets).

frame-ancestors

Кто может встраивать вашу страницу в <iframe> (защита от clickjacking).

ЧАСТО ИСПОЛЬЗУЕМЫЕ ЗНАЧЕНИЯ

'self'	'none'	https:	'unsafe-inline'
Только ваш домен	Полный запрет	Любой HTTPS источник	Опасно: разрешает встроенный код

Контролируйте, откуда загружается контент.

@lionsoulbatyshchikov

HTTP HEADERS • CSP

Nonces и безопасный запуск

CSP позволяет разрешать только доверенные inline-скрипты с помощью nonce. Это защищает от XSS, даже если злоумышленник сможет внедрить код на страницу.



how a nonce works

```
# сервер генерирует случайное значение для каждого ответа
Content-Security-Policy: script-src 'nonce-k9X2pQ'

# на странице только скрипты с этим nonce будут выполняться
<script nonce="k9X2pQ">app() </script>
<script> steal() </script>
```

ВЫПОЛНЯЕТСЯ

ЗАБЛОКИРОВАНО

Избегайте 'unsafe-inline'

Он разрешает выполнение любых внедрённых скриптов, поэтому лишает CSP смысла.

ВНЕДРЯЕМ ПОЭТАПНО • НИЧЕГО НЕ ЛОМАЕМ

1. Наблюдаем

`Content-Security-Policy-Report-Only`

Сначала включаем режим Report-Only: собираем отчёты о нарушениях и ничего не блокируем.

2. Исправляем

`выносим inline-код · добавляем nonce`

Убираем ненужные inline-скрипты, добавляем nonce к тем, что должны выполняться.

3. Включаем защиту

`Content-Security-Policy`

Переключаемся на боевой режим. Теперь политика блокирует все неподходящие скрипты.

Постепенно, но надёжно.

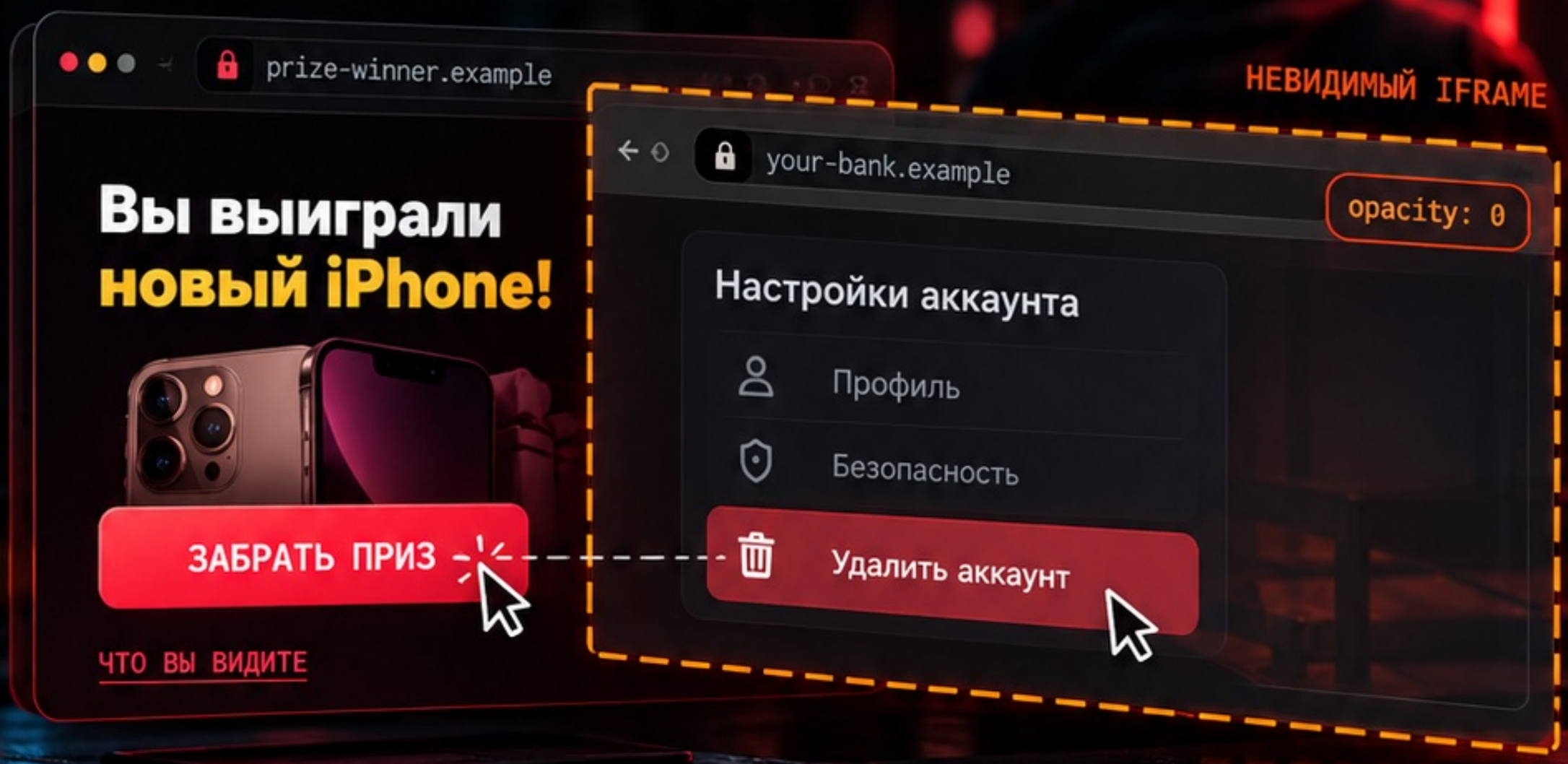
@lionsoulbatyshchikov

HTTP HEADERS • X-FRAME-OPTIONS

Кликджекинг

10/20

Злоумышленник прячет ваш сайт в невидимом iframe и подсовывает под клики свои элементы. Вы думаете, что нажимаете одно, а на самом деле — другое.



Как защититься — используйте правильные заголовки

X-Frame-Options: DENY — ПРОСТО

Полностью запрещает встраивать вашу страницу в iframe на любых сайтах.

X-Frame-Options: SAMEORIGIN — ГИБКО

Разрешает iframe только с вашего же домена.

CSP: frame-ancestors 'none' — СОВРЕМЕННО

Современный способ. Рекомендуется задавать вместе с X-Frame-Options для поддержки старых браузеров.

Не давайте злоумышленнику управлять вашими кликами.

@lionsoulbatyshchikov

HTTP HEADERS • X-CONTENT-TYPE-OPTIONS

MIME sniffing

Браузеры иногда игнорируют указанный **Content-Type** и сами определяют тип файла по содержимому. Такой догадке браузера называется **MIME sniffing**.

Загруженный злоумышленником файл notes.txt

```
1  Content-Type: text/plain
2
3  <script>alert(document.cookie)</script>
```

notes.txt
label: text

БЕЗ nosniff

notes.txt
label: text

Браузер «нюхает» содержимое
Видит, что это код

Выполняет
как скрипт

С nosniff

notes.txt
label: text

Браузер доверяет заголовку
Видит: это обычный текст

Отказывается
выполнять

X-Content-Type-Options: nosniff

Единственно верное значение. Добавьте его и забудьте.

Важно: Указывайте корректный Content-Type и всегда добавляйте **X-Content-Type-Options: nosniff**.

@lionsoulbatyshchikov

HTTP HEADERS • REFERRER-POLICY

Утечка ссылок

12/20

Браузер отправляет заголовок Referer и тем самым может раскрыть, откуда пришёл пользователь. Это часто приводит к утечке чувствительных ссылок, например с токенами или ID.

shop.example/reset?token=8f3a91c

Сброс пароля

Нужна помощь? Откройте наш **help center**

СБРОСИТЬ ПАРОЛЬ

Вы нажимаете, и браузер отправляет запрос на **help-center.example**

Запрос, который видит другой сайт

```
Referer: https://shop.example/reset?token=8f3a91c
# вот ваша ссылка на сброс пароля — утекла
```

ЧТО ВИДЯТ ДРУГИЕ САЙТЫ ПРИ РАЗНЫХ НАСТРОЙКАХ

no-referrer

Другие сайты ничего не видят.

`Referer:` (отсутствует)

МАКСИМАЛЬНАЯ ПРИВАТНОСТЬ

strict-origin-when-cross-origin

Другие сайты видят только ваш домен и протокол (без пути и токенов).

`Referer: https://shop.example`

РЕКОМЕНДУЕТСЯ

unsafe-url

Другие сайты видят полный URL, включая токены, ID и другие параметры.

`Referer: https://shop.example/reset?token=8f3a91c`

НЕКОГДА НЕ ИСПОЛЬЗОВАТЬ

Факт: заголовок Referer — это историческая особенность HTTP, и его нельзя полностью исправить. Используйте **Referrer-Policy**, чтобы минимизировать риски.

@lionsoulbatyshchikov

HTTP HEADERS • PERMISSIONS-POLICY

Отключаем ненужные возможности

С помощью **Permissions-Policy** можно запретить использование мощных API браузера, даже если на странице есть вредоносный скрипт. Пользователь не сможет их включить обратно.

```
Permissions-Policy: camera=(), microphone=(), ...
```

Камера	`camera=()`	выкл
Микрофон	`microphone=()`	выкл
Геолокация	`geolocation=(self)`	вкл
USB устройства	`usb=()`	выкл
Полноэкранный режим	`fullscreen=(self)`	вкл



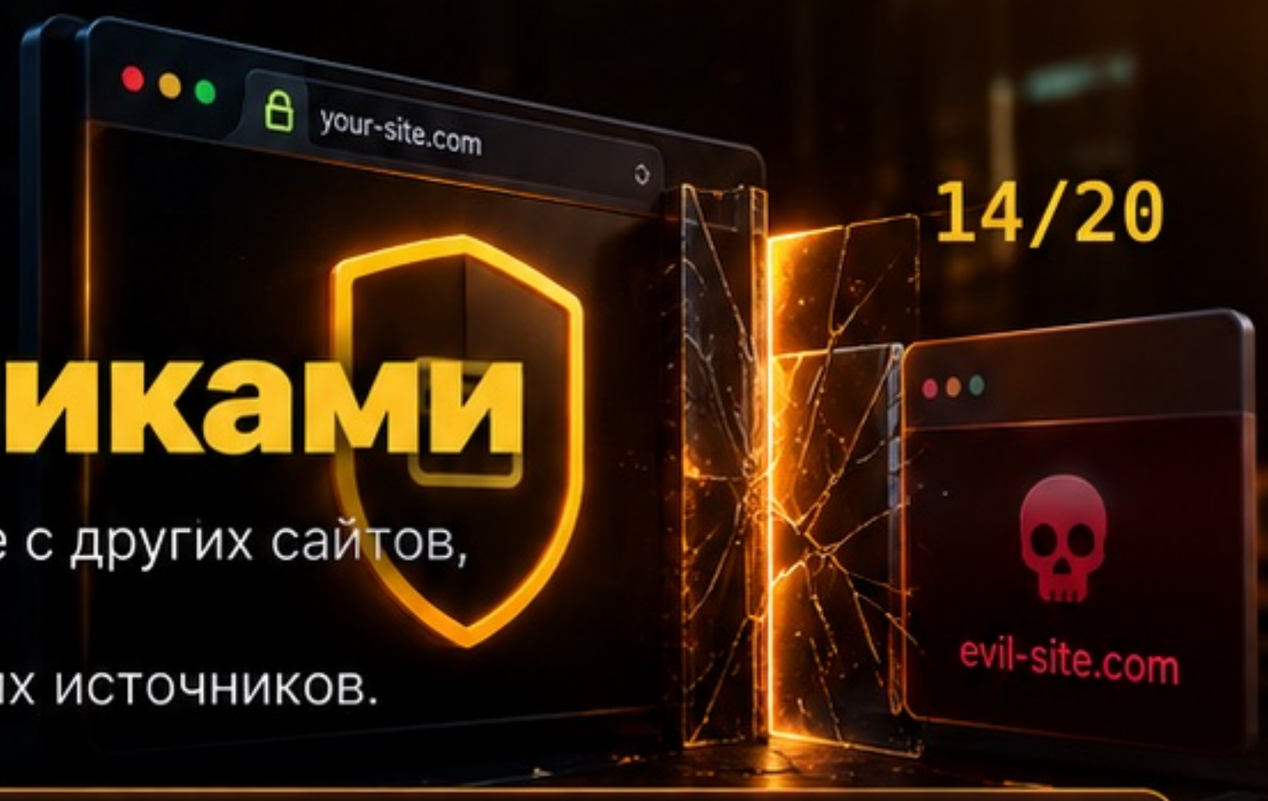
КАК ЧИТАТЬ ЗНАЧЕНИЯ

`()`	Полностью выключено для всех сайтов	Максимальная защита
`(self)`	Разрешено только для вашего домена	Ограниченный доступ
`(self "https://maps.example")`	Ваш домен + доверенный партнёр	Гибкая настройка

Даже если на странице есть вредоносный скрипт или реклама — они **не смогут получить доступ** к камере, микрофону и другим API.

HTTP HEADERS • COOP / COEP / CORP

Изоляция между источниками



Некоторые атаки позволяют читать данные с других сайтов, которые открываются в том же браузере.
Эти заголовки изолируют ваш сайт от чужих источников.

COOP Cross-Origin-Opener-Policy

value **same-origin**

Разрывает связь между вашим окном/вкладкой и всплывающими окнами с других сайтов.



COEP Cross-Origin-Embedder-Policy

value **require-corp**

Разрешает загружать на страницу только ресурсы, которые явно это допускают.



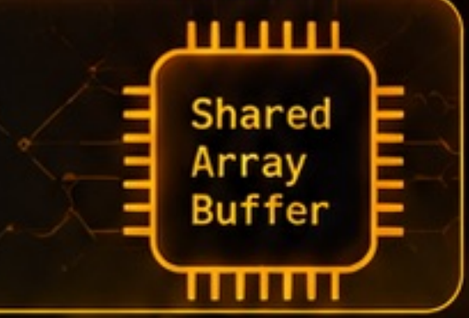
CORP Cross-Origin-Resource-Policy

value **same-origin**

Не позволяет другим сайтам вставлять ваши ресурсы (изображения, скрипты, и т.д.).



Бонус: изоляция открывает доступ к мощным браузерным API, например **SharedArrayBuffer**.

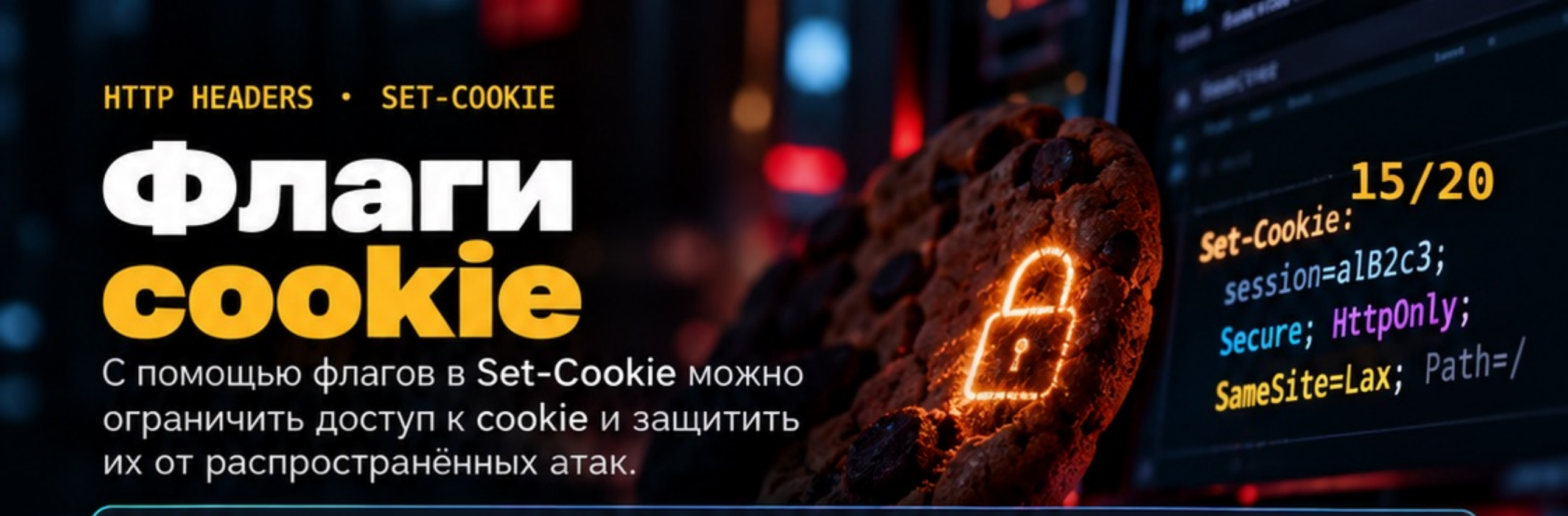


@lionsoulbatyshchikov

HTTP HEADERS • SET-COOKIE

Флаги cookie

С помощью флагов в **Set-Cookie** можно ограничить доступ к cookie и защитить их от распространённых атак.



Пример Set-Cookie для сессионного cookie

```
Set-Cookie: session=alB2c3; Secure; HttpOnly; SameSite=Lax; Path=/
```

Secure

Передаётся только по HTTPS.
Защищает от перехвата, например, через публичный Wi-Fi.

Блокирует перехват в незащищённых сетях

Пример

```
Set-Cookie: session=alB2c3;
Secure;
```

HttpOnly

Недоступен для JavaScript.
Не позволяет украсть cookie через XSS.

Блокирует кражу через XSS

document.cookie → ""

Пример

```
Set-Cookie: session=alB2c3;
HttpOnly;
```

SameSite=Lax

Не отправляется при переходе с других сайтов (в большинстве случаев).
Помогает защититься от CSRF.

Блокирует CSRF

Пример

```
Set-Cookie: session=alB2c3;
SameSite=Lax;
```

ТРИ УРОВНЯ SAMESITE

Strict

Никогда не отправлять при кросс-сайте запросах.

Lax

Безопасное значение по умолчанию для большинства случаев.

None

Всегда отправлять.
Требует Secure.

Важно: Правильно выставленные флаги значительно снижают риски перехвата, XSS и CSRF-атак.

@lionsoulbatyshchikov

HTTP HEADERS · REMOVE & HARDEN

Убираем лишние заголовки

16/20

Некоторые заголовки выдают версии используемого ПО и другие детали, которые помогают атакующему. Лучше просто удалить их.

Server: ~~WebServer/2.4.29~~

X-Powered-By: ~~Framework/7.2.1~~

1 Перестаньте «светить» версии

Пример ответных заголовков сервера

Server: ~~WebServer/2.4.29~~ УДАЛИТЬ

X-Powered-By: ~~Framework/7.2.1~~ УДАЛИТЬ

Эти данные помогают атакующему подобрать известные уязвимости для вашей версии ПО.

2 Не кешируйте приватные страницы

Вы выходите из аккаунта на общем устройстве

Следующий пользователь нажимает Back

Ваша страница может открыться из кэша браузера

Назад Alt+Left

Вперёд Alt+Right

Перезагрузить Ctrl+R

Cache-Control: no-store

Браузер не сохраняет копии приватных страниц в кэше (back/forward cache и обычный кэш).

@lionsoulbatyshchikov

HTTP HEADERS • СТАРЫЕ ЗАГОЛОВКИ

17/20

Устаревшие заголовки

Эти заголовки когда-то считались хорошей практикой, но большинство браузеров их больше не поддерживают. Некоторые даже создавали проблемы.

R.I.P.

X-XSS-Protection

2010 – 2019

Встроенный XSS-фильтр в браузерах. Его удалили, потому что он был ненадёжным и мог создавать новые уязвимости (XSS).

ИСПОЛЬЗУЙТЕ ВМЕСТО

CSP

R.I.P.

Expect-CT

2016 – 2022

Позволял проверять наличие сертификата в Certificate Transparency. Теперь это делается по умолчанию в браузерах.

ИСПОЛЬЗУЙТЕ ВМЕСТО

ничего не нужно

R.I.P.

Public-Key-Pins

2013 – 2018

Идея была хорошая: зафиксировать доверенные ключи. Но одна ошибка могла заблокировать доступ ко всему сайту на месяцы.

ИСПОЛЬЗУЙТЕ ВМЕСТО

HSTS

Важно: Если вы всё ещё встречаете эти заголовки в старых руководствах — знайте, что они устарели и больше не рекомендуются.

@lionsoulbatyshchikov

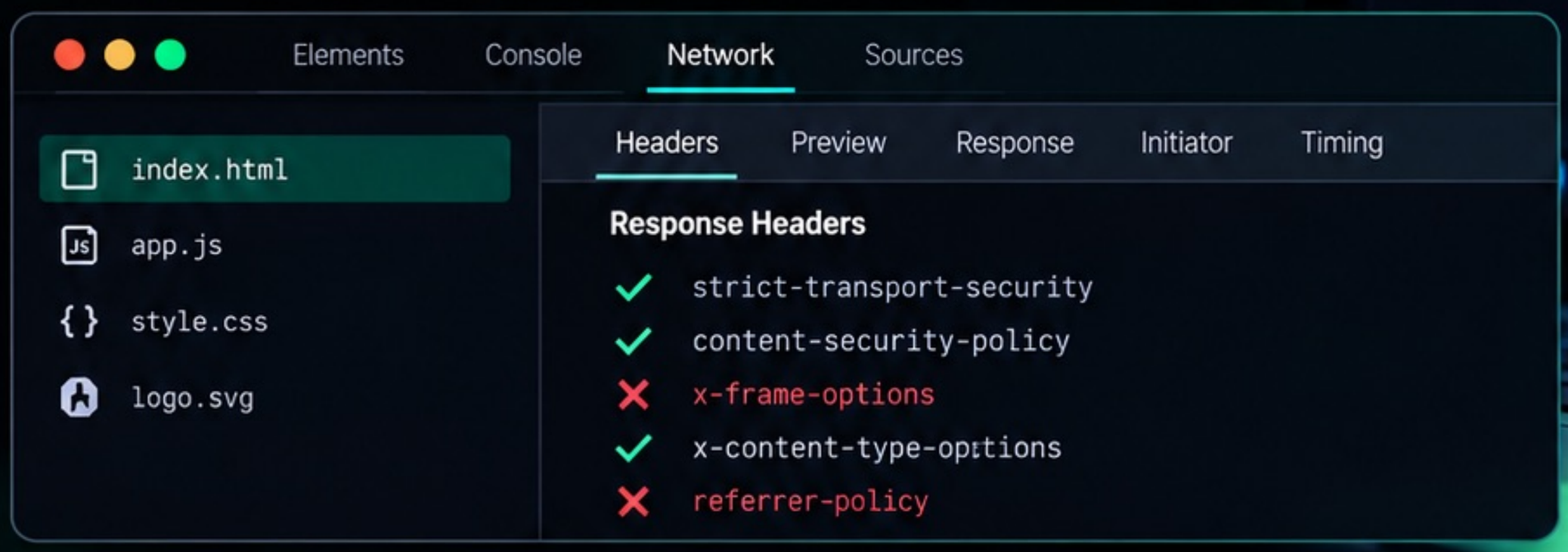
2-MINUTE AUDIT • ПРОВЕРЬ СВОЙ САЙТ

Проверь свой сайт

Быстрая проверка поможет понять, какие заголовки уже настроены, а какие отсутствуют — и что стоит улучшить.

1 Откройте DevTools в браузере

F12 → Network → Reload → Click page



2 Или используйте одну команду в терминале

terminal

```
$ curl -I https://your-site.example                    # -I = только заголовки
HTTP/2 200
strict-transport-security: max-age=31536000
content-security-policy: ...
...
```

3 Либо вставьте URL в онлайн-сканер заголовков

https://your-site.example ПРОВЕРИТЬ →

Получите оценку и рекомендации по недостающим заголовкам.

Зачем это делать?
Правильно настроенные security headers снижают риск XSS, clickjacking и утечки данных — это базовая гигиена современной веб-безопасности.

@lionsoulbatyshchikov

SECURITY HEADERS • БАЗОВАЯ КОНФИГУРАЦИЯ

Скопируйте базовый набор

Этот набор заголовков — хорошая отправная точка для большинства веб-сайтов. Он закрывает ключевые векторы атак и заметно повышает безопасность по умолчанию.



security-headers.conf

Copy

```
Strict-Transport-Security: max-age=31536000; includeSubDomains
Content-Security-Policy: default-src 'self'; object-src 'none';
                         base-uri 'self'; frame-ancestors 'none'
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Referrer-Policy: strict-origin-when-cross-origin
Permissions-Policy: camera=(), microphone=(), geolocation=()
Cross-Origin-Opener-Policy: same-origin
Set-Cookie: id=...; Secure; HttpOnly; SameSite=Lax
# при необходимости удалите: Server, X-Powered-By
```

КАК ЭТО ВЛИЯЕТ НА ВАШУ ОЦЕНКУ

F D C B A A+

до → после

Сначала протестируйте CSP в режиме Report-Only.
Строгая политика может привести к поломке рабочего сайта.

Добавляйте HSTS preload только после того,
как убедитесь, что все субдомены корректно работают по HTTPS.

@lionsoulbatyshchikov

YOU MADE IT • 20/20

Теперь ваш сайт под защищой.

Вы разобрали ключевые security headers и знаете, как правильно их настраивать. Этот набор — ваш базовый щит для современного веб-приложения.



- HSTS
- CSP
- Report-Only
- X-Frame-Options
- nosniff
- Referrer-Policy
- Permissions-Policy
- COOP
- COEP
- CORP
- Cookie flags
- Cache-Control

Сохраните
чтобы быстро вернуться при следующем деплое.

Поделитесь
с командой разработчиков и коллегами.

БЕЗОПАСНЫЙ ВЕБ — ЭТО НЕ ОПЦИЯ,
А ХОРОШАЯ ПРИВЫЧКА.

@lionsoulbatyshchikov