

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

Информационная безопасность

Работа №7

«Безопасность браузера и анализ сетевого трафика»

Барсуков Максим Андреевич

Группа: Р3415

Выполнение

Настройка браузера

Моим основным браузером является Google Chrome, будем выполнять работу в нем. Для начала откроем настройки конфиденциальности и безопасности, как показано на рисунке 1:

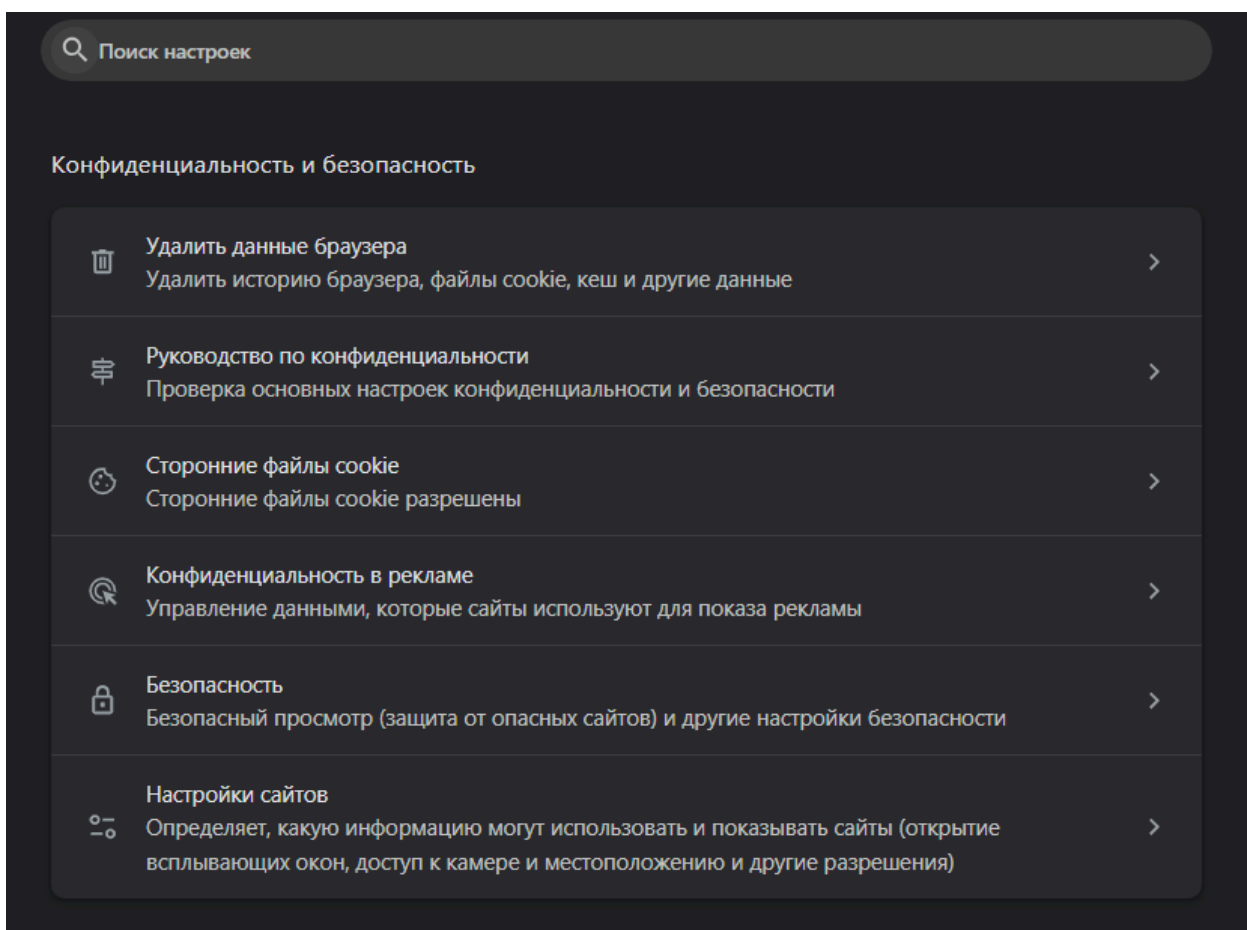


Рисунок 1 — Настройки конфиденциальности и безопасности

Откроем раздел «Сторонние файлы cookie» (рисунок 2):

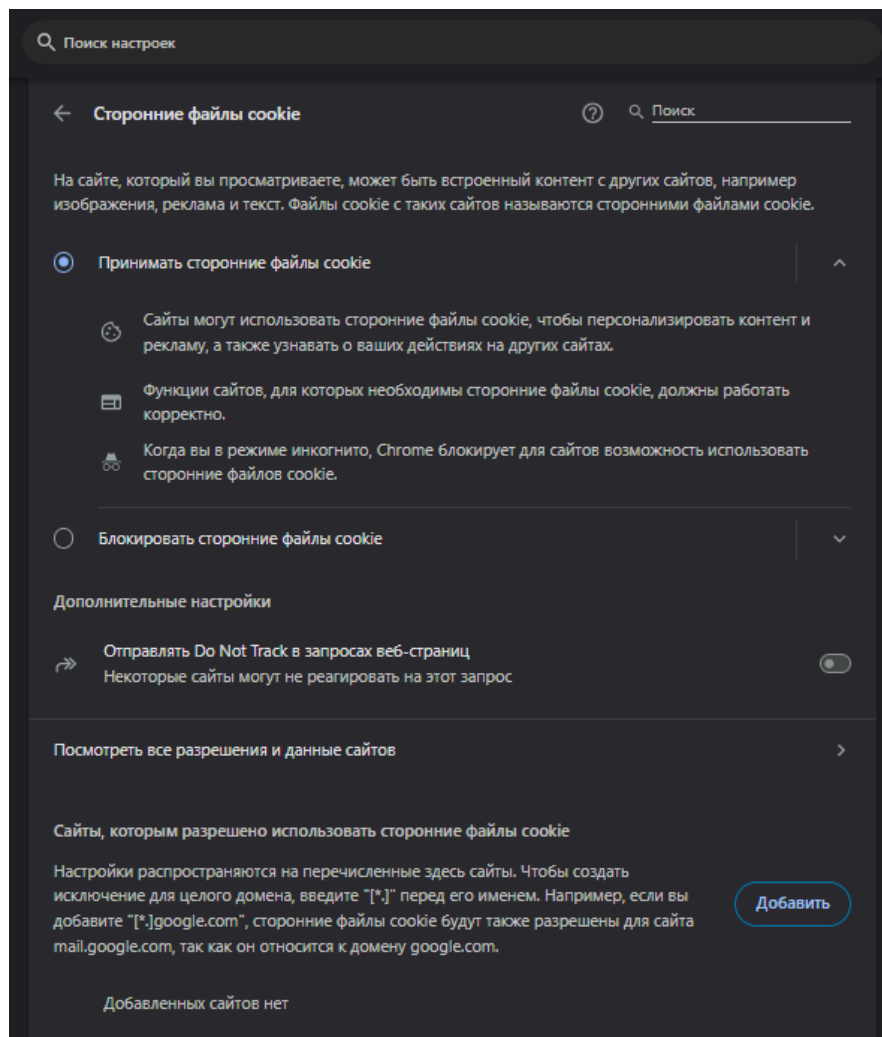


Рисунок 2 — Начальные настройки сторонних файлов cookie

Изначально браузер принимал сторонние файлы cookie. Это значит, что браузер разрешает сайтам использовать сторонние куки (те, что устанавливаются не тем сайтом, который посещается в данный момент, а другими сайтами). Можно, например, с рекламными целями отследить, какие страницы в соцсетях были посещены, чтобы показывать таргетированную рекламу. Выключаем эту функцию.

Do not track – это специальный заголовок, который посылает на каждый сайт браузер, имея в виду, что действия пользователя не стоит отслеживать, но сайты могут этот заголовок проигнорировать. Включаем эту функцию.

Далее идет белый список сайтов, которым разрешено использовать сторонние файлы куки. Сюда можно вручную добавить сайты, которым человек доверяет, и которым можно разрешить использовать сторонние файлы куки, даже при общей политике блокировки. Ничего не добавляем.

Далее зайдём в настройки конфиденциальности в рекламе:

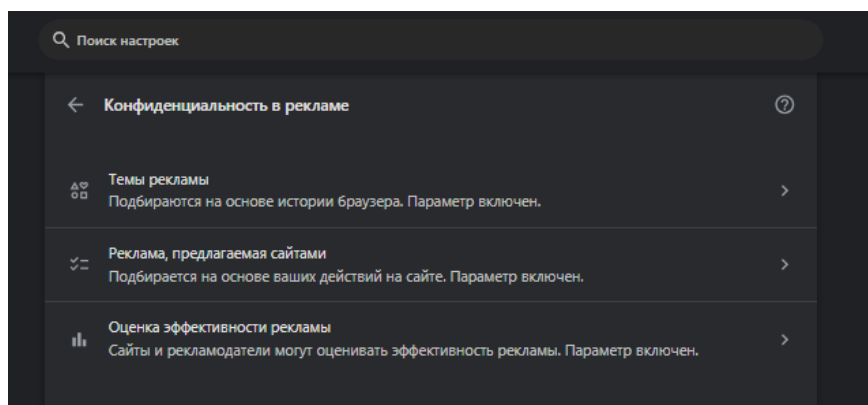


Рисунок 3 — Раздел «Конфиденциальность в рекламе»

Откроем подраздел «Темы рекламы», как показано на рисунке 4:

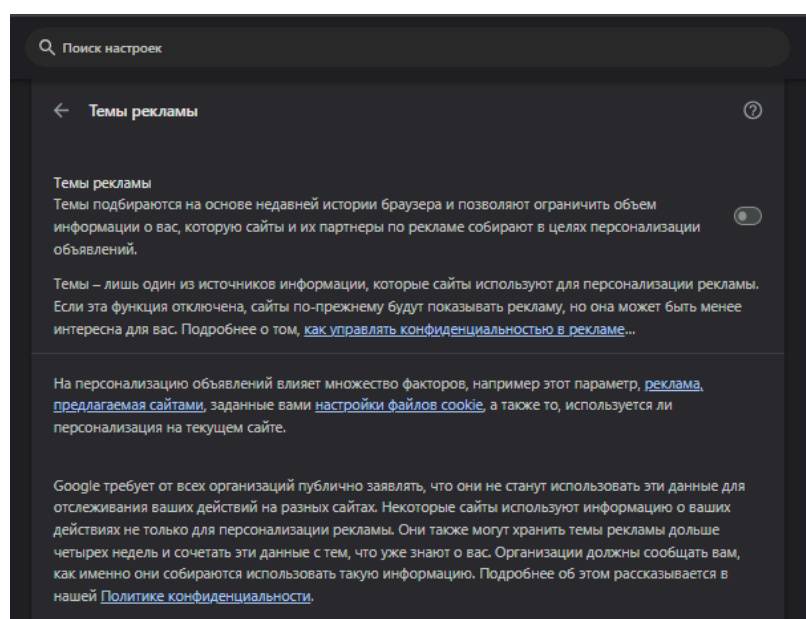


Рисунок 4 — Подраздел «Темы рекламы»

Chrome пытается заменить сторонние куки новой технологией. Вместо того чтобы следить за человеком лично, браузер анализирует историю просмотров и определяет, к какой «группе интересов» (например, «автомобилисты», «любители кулинарии») вы относитесь. Затем сайтам передается не ваш личный идентификатор, а ID этой группы. Отключаем эту функцию, по своей сути это все равно отслеживание поведения для показа рекламы.

Перейдем в подраздел «Реклама, предлагаемая сайтами» (рисунок 5):

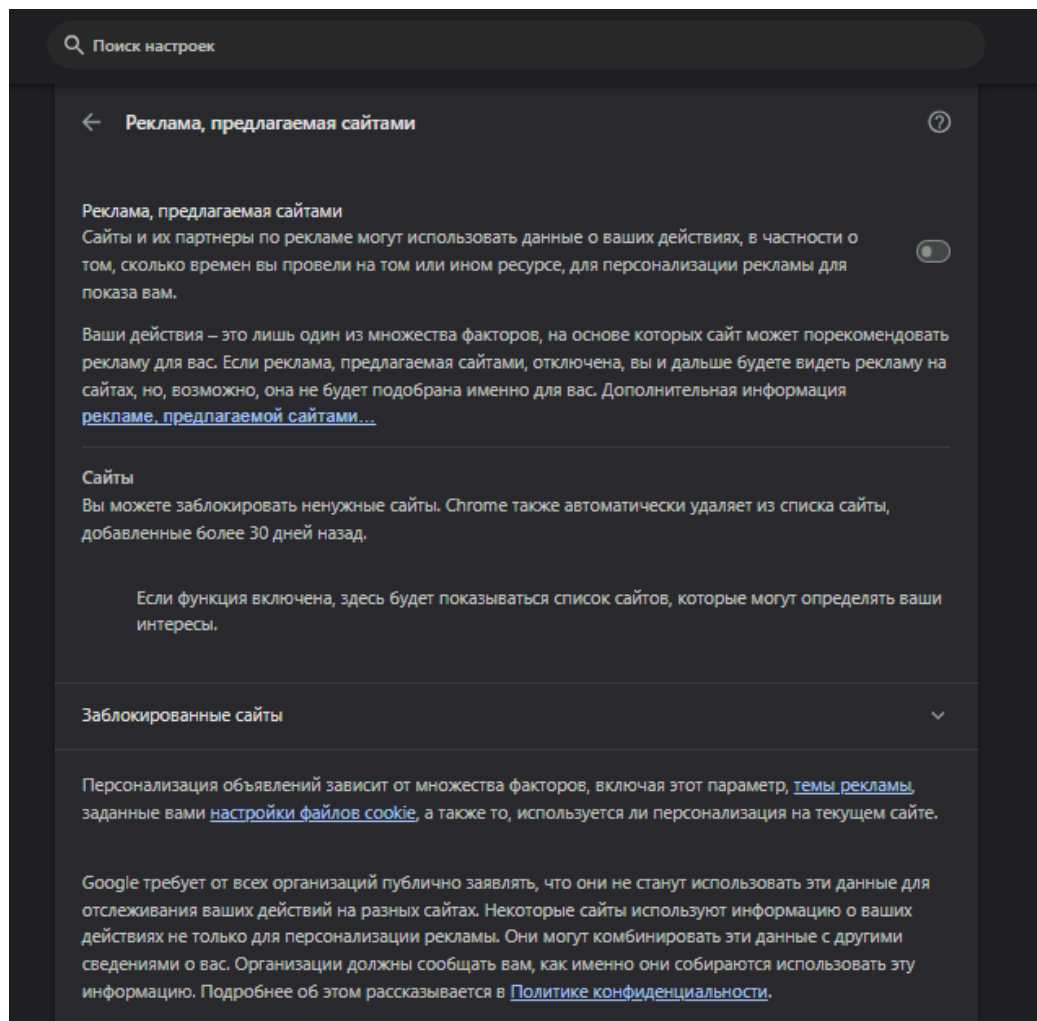


Рисунок 5 — Подраздел «Реклама, предлагаемая сайтами»

Отдельный сайт может использовать данные о ваших действиях именно на нём (что смотрели, сколько времени провели), чтобы показывать вам релевантную рекламу как на своем сайте, так и через своих рекламных партнеров на других сайтах. Выключаем эту функцию.

Перейдем в подраздел «Оценка эффективности рекламы» (рисунок 6):

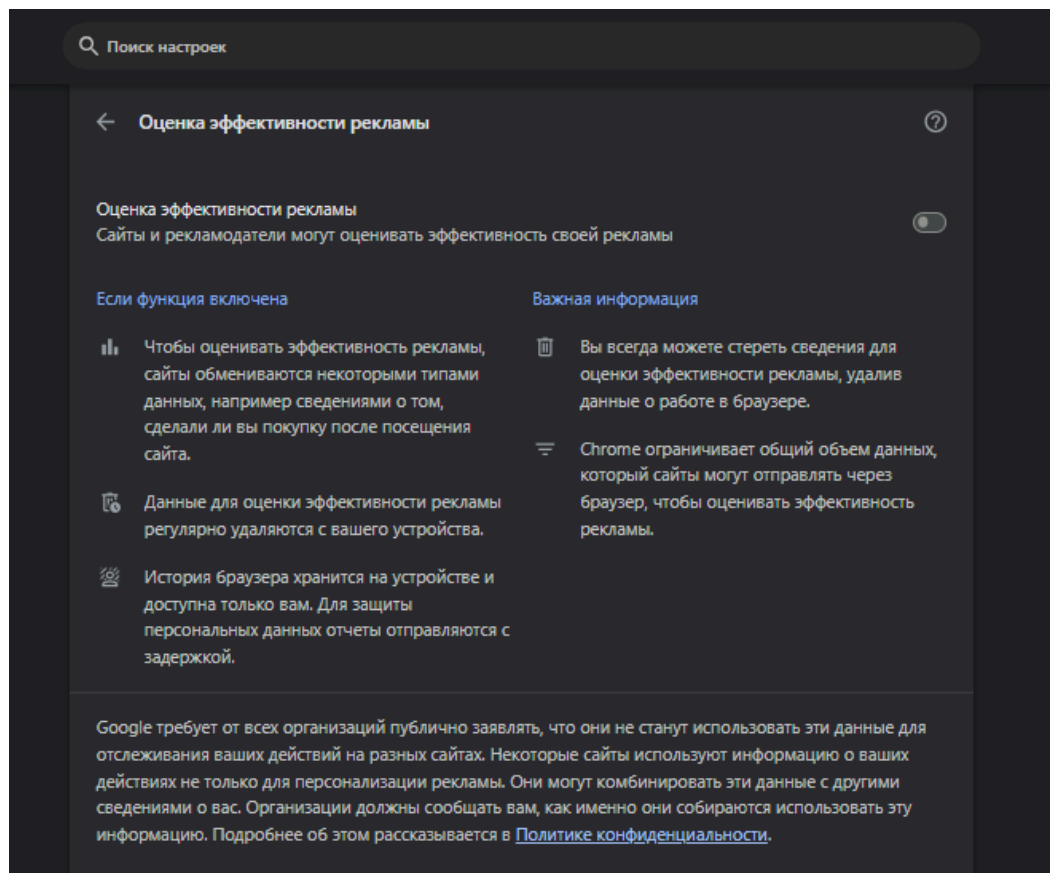


Рисунок 6 — Подраздел «Оценка эффективности рекламы»

Это позволяет рекламодателям получать отчеты о том, привела ли их реклама к желаемому результату (например, купили ли вы товар после клика по объявлению). Для этого с вашего устройства могут передаваться обезличенные данные о «конверсиях». Отключаем эту функцию, так как несмотря на то, что Google заверяет о том, что это анонимно и имеет только ограниченную функциональность, это все равно передача данных о поведении рекламодателю.

Откроем раздел «Безопасность» (рисунок 7):

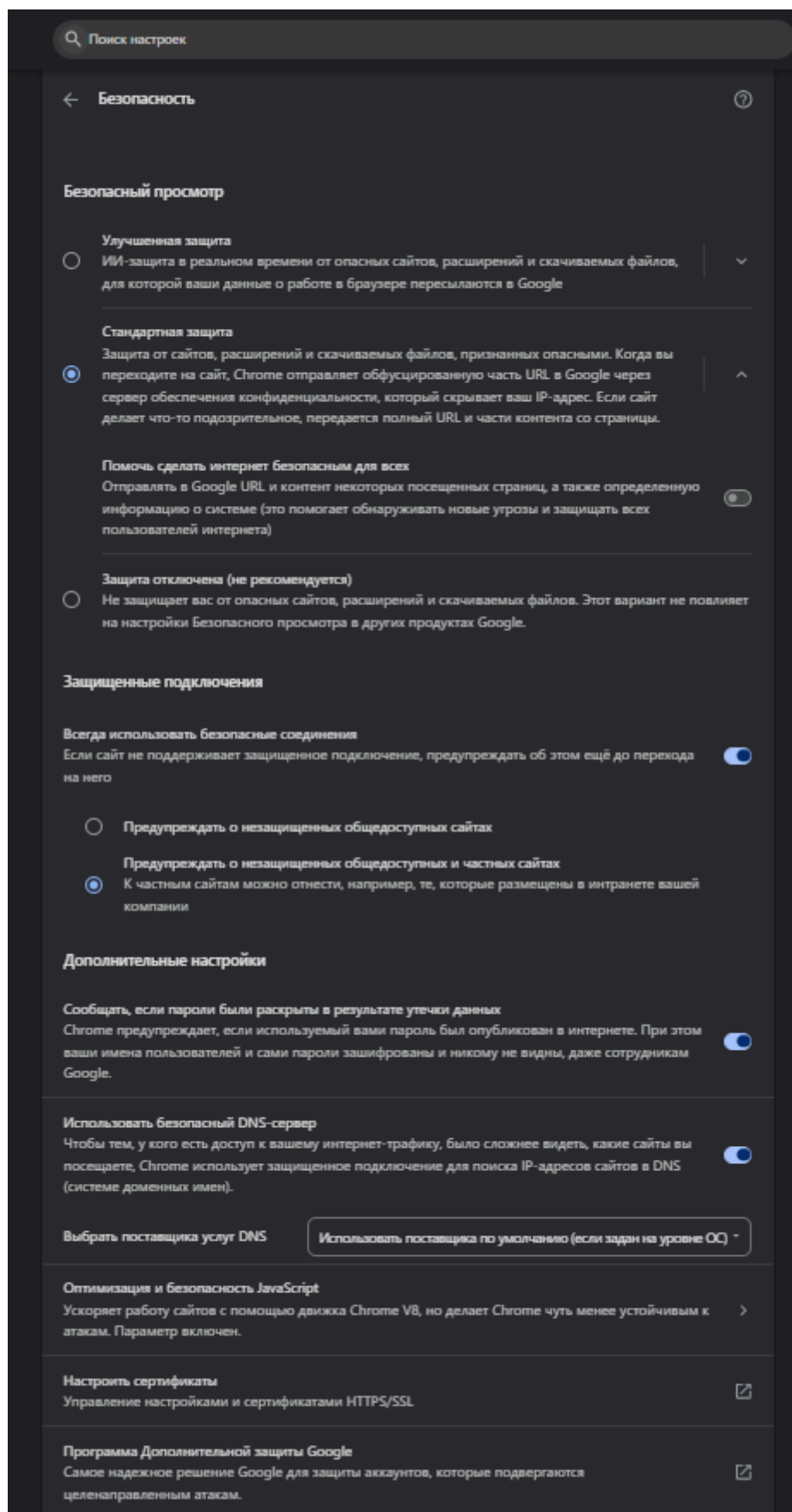


Рисунок 7 — Раздел «Безопасность»

Оставляем стандартную защиту, так как это стабильное, протестированное средство для защиты браузера, защищающий во многих случаях от фишинга и вредоносных сайтов, без него в браузере просто опасно находиться.

Оставляем функцию сообщения о раскрытии паролей, тогда можно будет быстрее поменять пароль в случае утечки и снизить риски получения несанкционированного доступа к аккаунту. Google все равно не видит пароли в чистом виде, а проверка происходит на основе хешей паролей.

Оставляем функцию использования безопасного DNS-сервера, это усиливает приватность. Запросы DNS будут шифроваться и интернет-провайдер не сможет видеть, куда человек заходит.

Оставляем функцию «Всегда использовать безопасные соединения», при попытке перехода на любой сайт с устаревшим протоколом HTTP браузер будет блокировать загрузку и выводить предупреждение.

Перейдем в подраздел «Оптимизация и безопасность JavaScript» (рисунок 8):

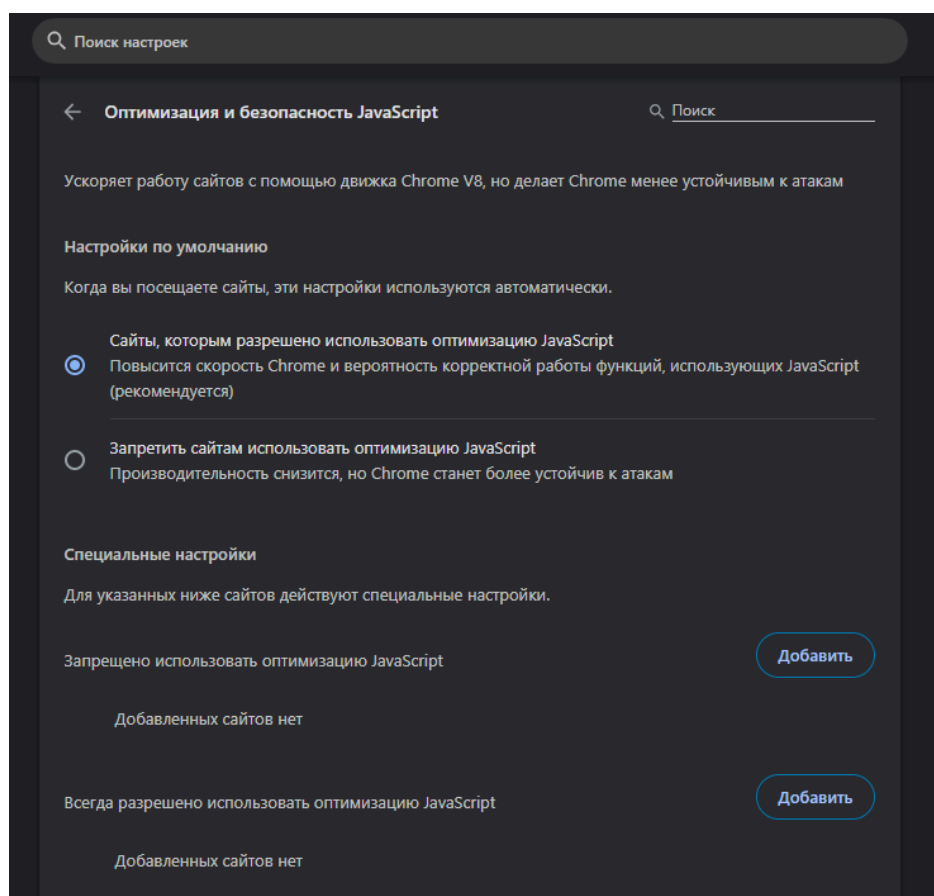


Рисунок 8 — Подраздел «Оптимизация и безопасность JavaScript»

Не будем изменять настройки оптимизации JavaScript, оставив режим по умолчанию, так как практике для большинства пользователей выигрыш в производительности и корректности работы веб-приложений критически важен.

Перейдем в подраздел «Локальные сертификаты» (рисунок 9):

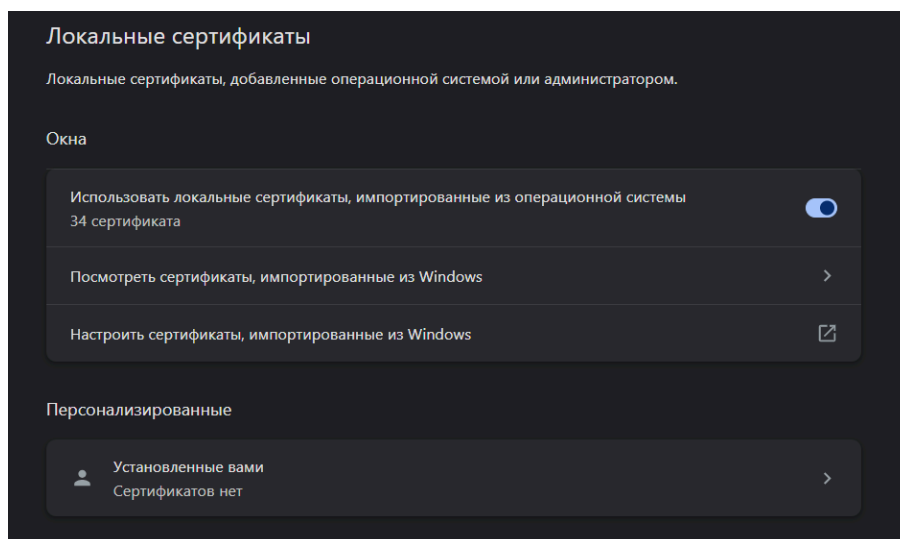


Рисунок 9 — Подраздел «Локальные сертификаты»

В разделе управления сертификатами видно, что браузер использует локальные сертификаты операционной системы (34 сертификата), что обеспечивает корректную работу с защищенными HTTPS-сайтами, как показано на рисунке 10:

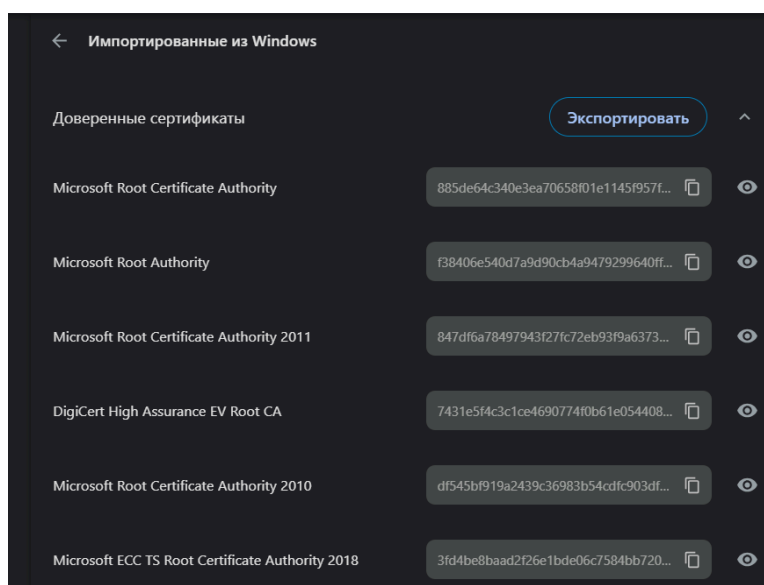


Рисунок 10 — Импортированные из Windows сертификаты

Перейдем в раздел «Настройки сайтов» (рисунки 11–13):

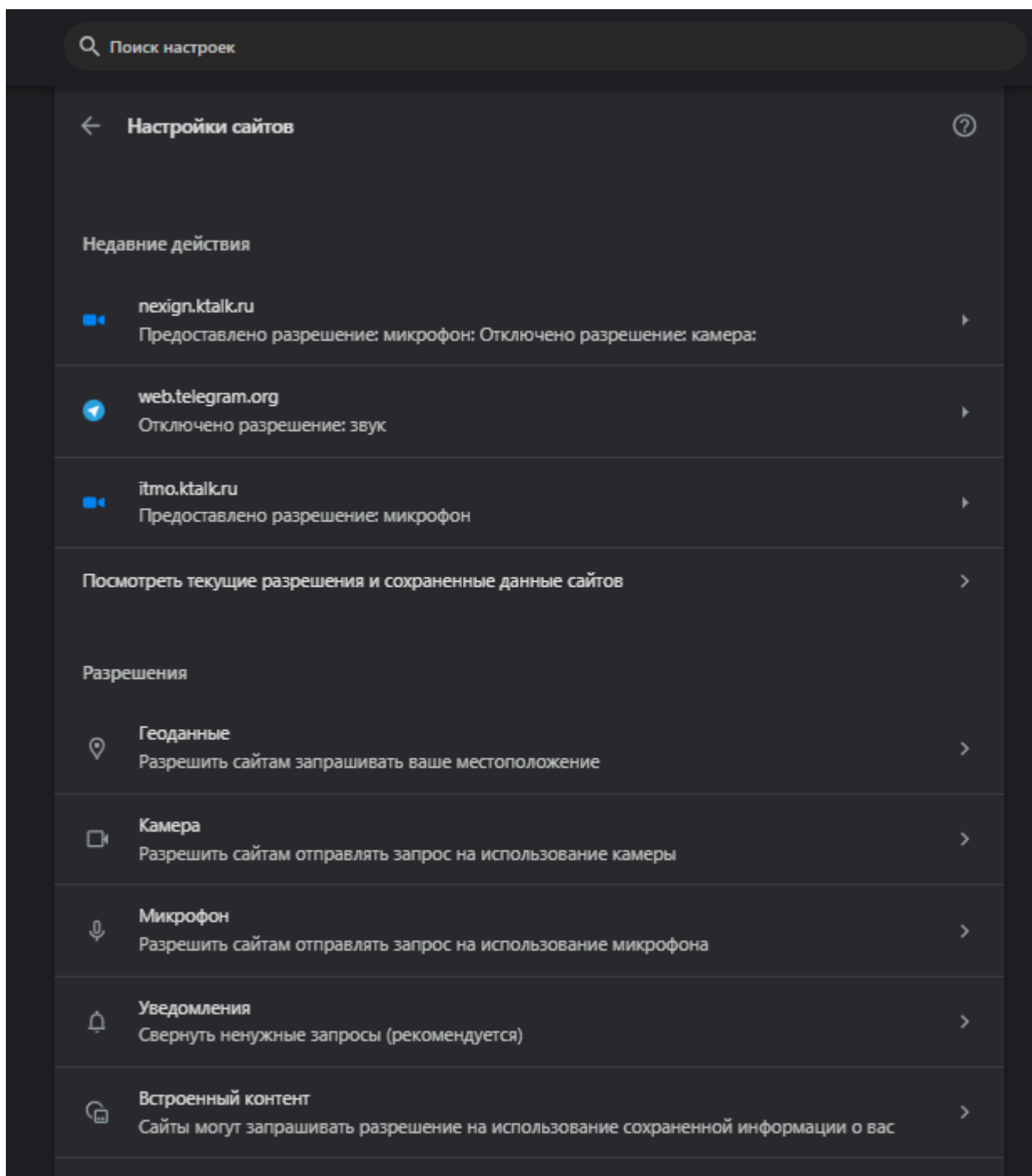


Рисунок 11 — Основные разрешения сайтов

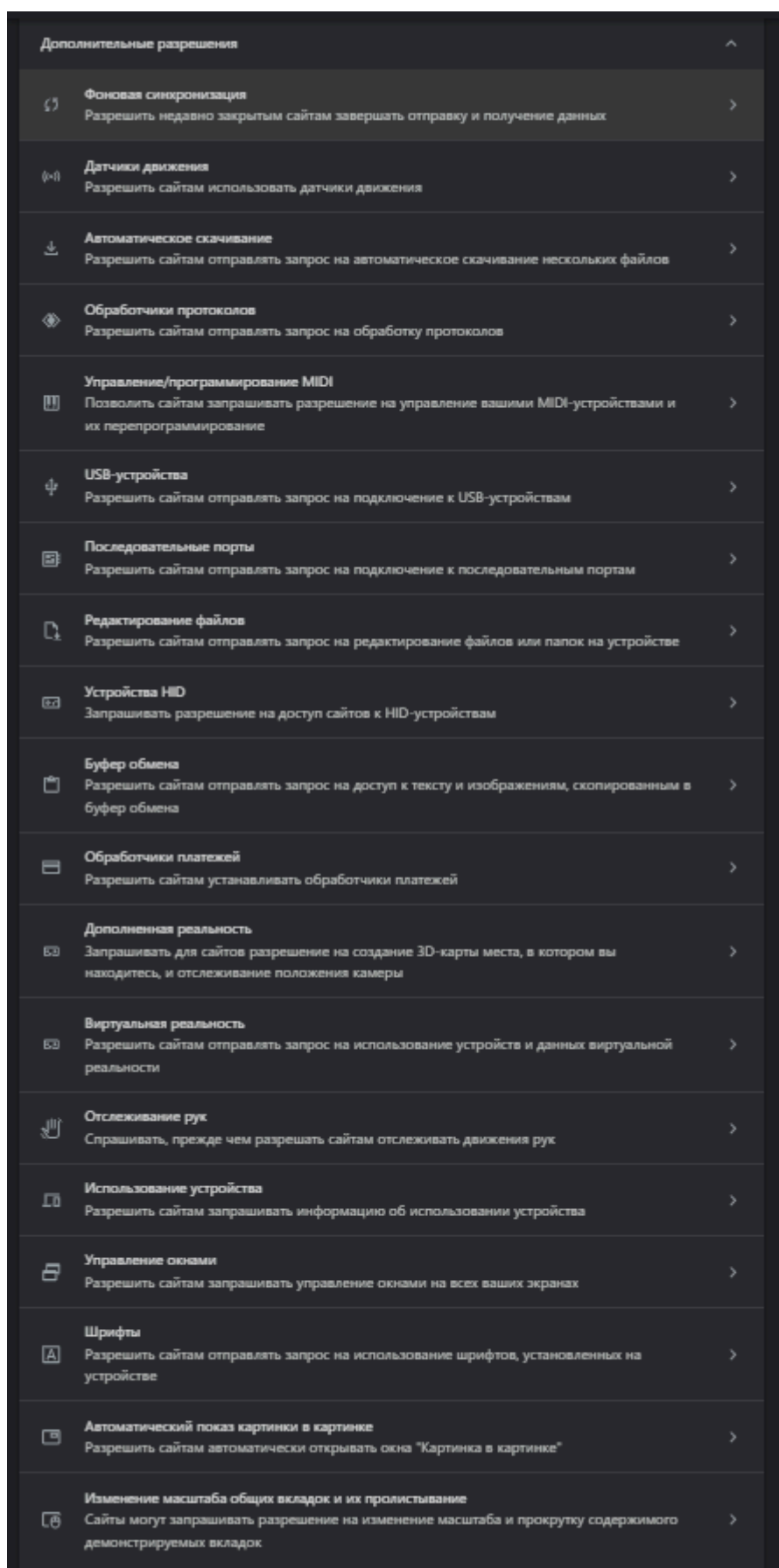


Рисунок 12 — Дополнительные разрешения сайтов

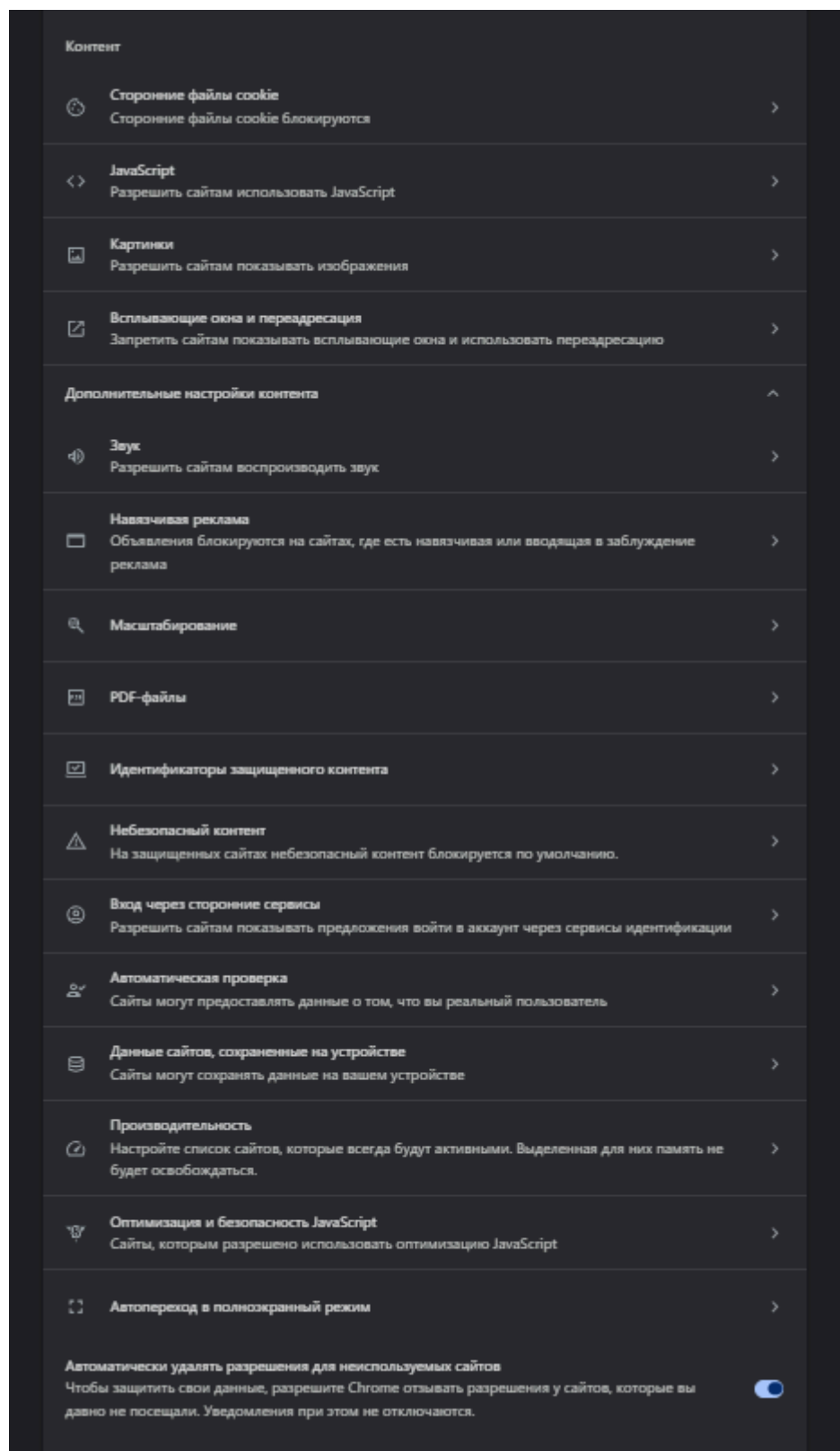


Рисунок 13 — Контент и дополнительные настройки контента

Проведем политику максимального ограничения доступа:

Отключены опасные функции доступа к USB-устройствам, файловой системе, портам, датчикам, midi-устройствам. Запрещена фоновая синхронизация и автоматические загрузки. Активировано автоматическое удаление разрешений для неиспользуемых сайтов.

При этом оставим работу JavaScript (без нее вообще невозможно нормальное функционирование в браузере), разрешим показывать картинки.

Для микрофона, камеры и геоданных оставим разрешение на запрос о доступе, и будем включать его только при необходимости.

Далее очистим кэш и куки, как показано на рисунках 14 и 15:

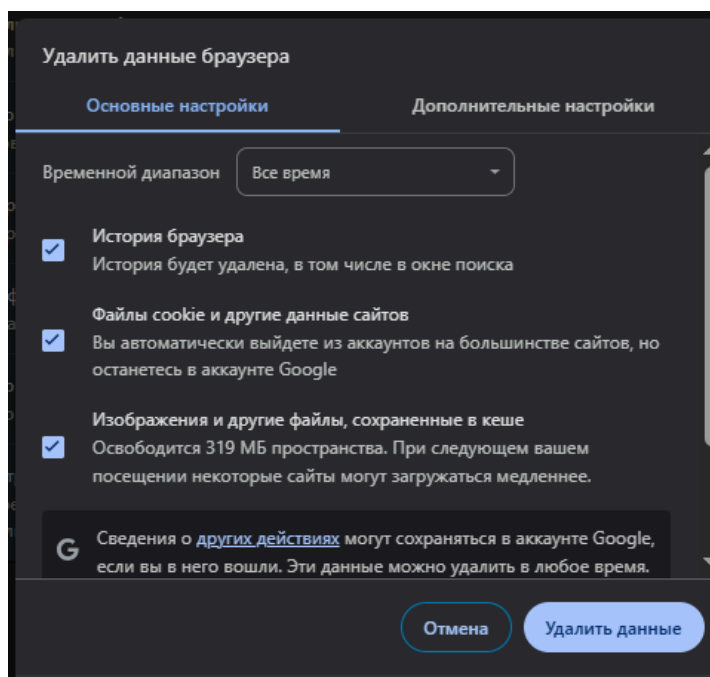


Рисунок 14 — Основные настройки удаления данных браузера

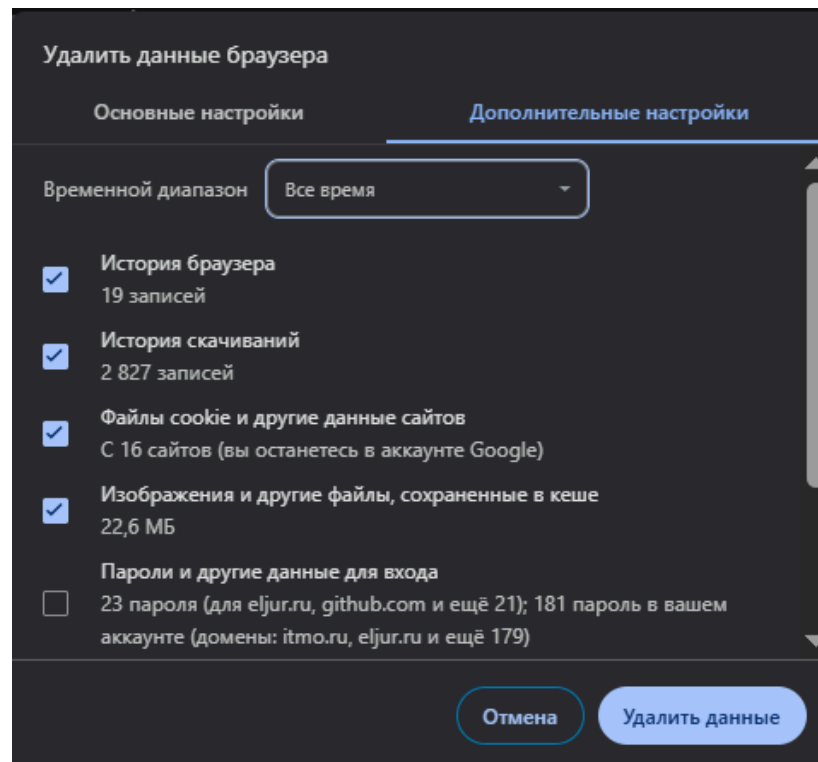


Рисунок 15 — Дополнительные настройки удаления данных браузера

После очистки кэша и куки освободится более 400 МБ пространства.

Анализ трафика

Далее зайдем на сайт <https://se.ifmo.ru/>. Перейдем во вкладку «Сеть», как показано на рисунке 16:

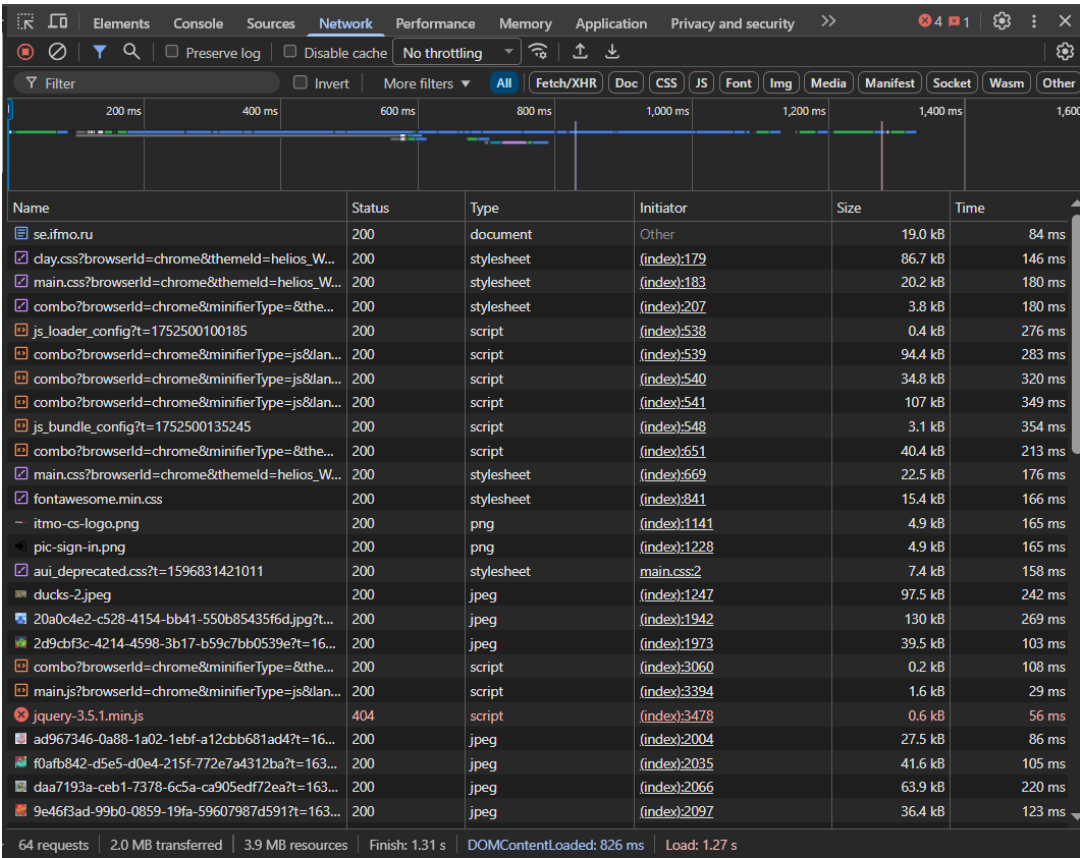


Рисунок 16 — Вкладка «Сеть»

Видим, что при обычной загрузке страницы было послано 64 HTTP запроса. Зайдем в заголовки самого первого GET-запроса (рисунок 17):

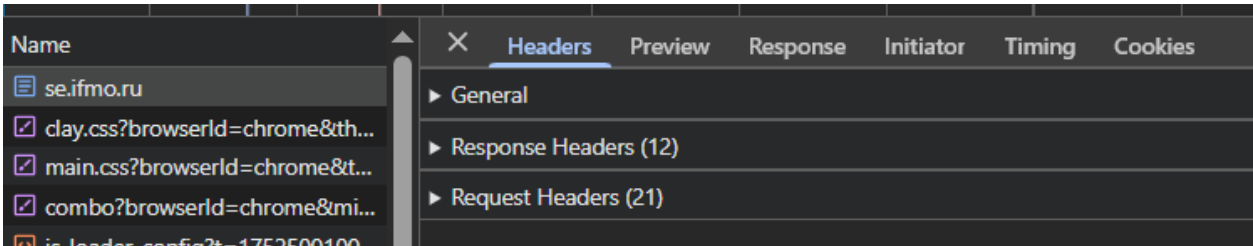


Рисунок 17 — Типы заголовков первого GET-запроса

Видим, что тут находится три типа заголовков — общие, заголовки ответов и запросов.

Посмотрим на общие заголовки (рисунок 18):

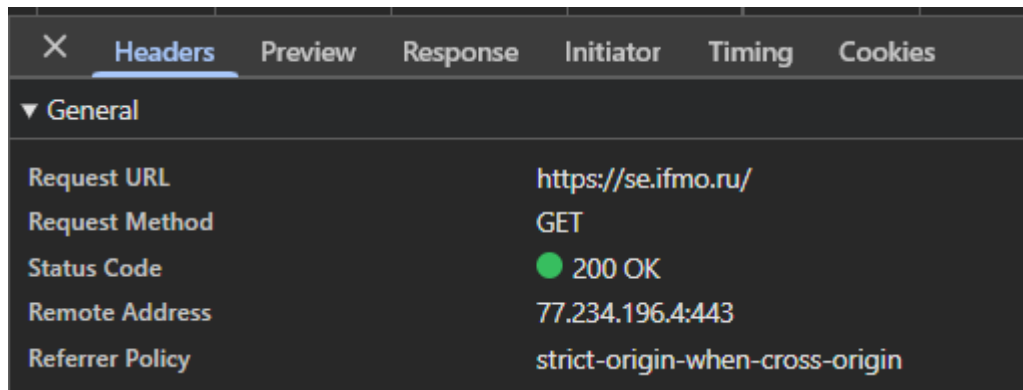


Рисунок 18 — Общие заголовки

Тут представлены: код статуса, удаленный адрес (IP сервера se.ifmo.ru + порт 443, стандартный для https протокола) и правило перехода, контролирующее, какую информацию отправлять при переходе между сайтами (strict – строгое ограничение, origin – передается только домен, но не полный URL, cross-origin – при переходе на другой сайт).

Рассмотрим заголовки ответа (рисунок 19):

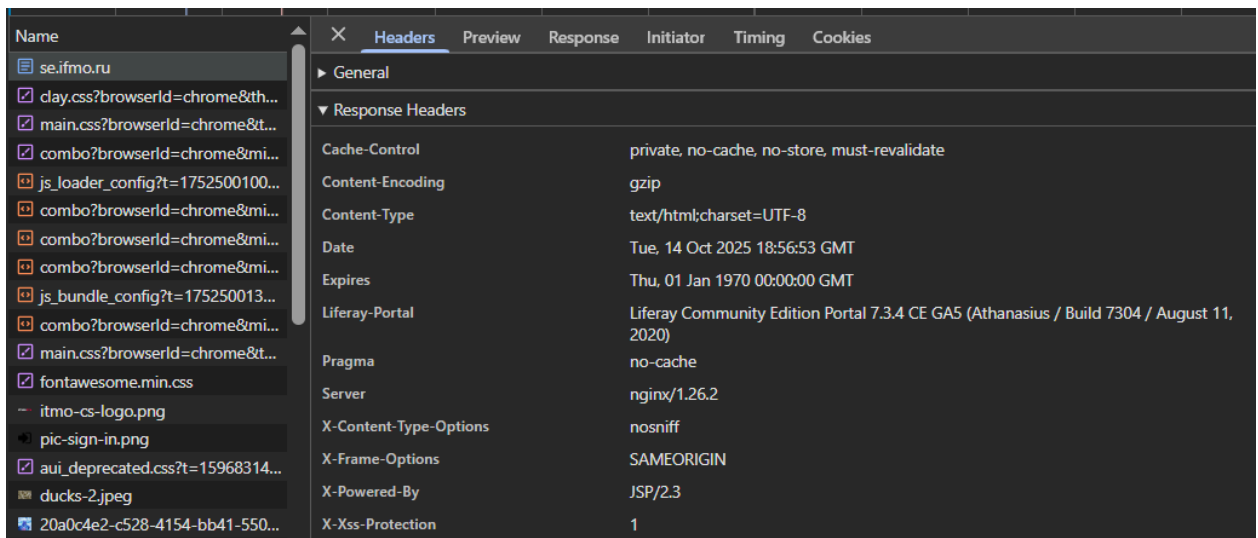


Рисунок 19 — Заголовки ответа

Заголовки ответа:

- `cache-control: private, no-cache, no-store, must-revalidate` – строгая политика кэширования: контент не сохраняется в публичном кэше, всегда требуется проверка актуальности на сервере, полный запрет на хранение в любом кэше.
- `content-encoding: gzip` – контент передан в сжатом формате для уменьшения объема передаваемых данных.
- `content-type: text/html; charset=UTF-8` – тип контента HTML с кодировкой UTF-8.
- `date: Tue, 14 Oct 2025 18:56:53 GMT` – точное время формирования ответа сервером.
- `expires: Thu, 01 Jan 1970 00:00:00 GMT` – установка заведомо прошедшей даты истечения срока действия, чтобы контент считался устаревшим.
- `liferay-portal: Liferay Community Edition Portal 7.3.4 CE GA5` – информация о используемой версии портала Liferay, что может раскрывать потенциальные уязвимости.
- `pragma: no-cache` – устаревший заголовок для обратной совместимости, запрещающий кэширование в HTTP/1.0.
- `server: nginx/1.26.2` – информация о веб-сервере и его версии, раскрытие которой не рекомендуется в целях безопасности.
- `x-content-type-options: nosniff` – защита от MIME-спуфинга: браузер должен доверять указанному типу контента и не пытаться определять его самостоятельно.
- `x-frame-options: SAMEORIGIN` – защита от кликджекинга: страница может быть встроена во фрейм только на сайтах с тем же происхождением.
- `x-powered-by: JSP/2.3` – информация о используемой технологии серверной части (Java Server Pages).
- `x-xss-protection: 1` – включение встроенного XSS-фильтра браузера для блокировки обнаруженных межсайтовых скриптов.

Рассмотрим заголовки запросов (рисунок 20):

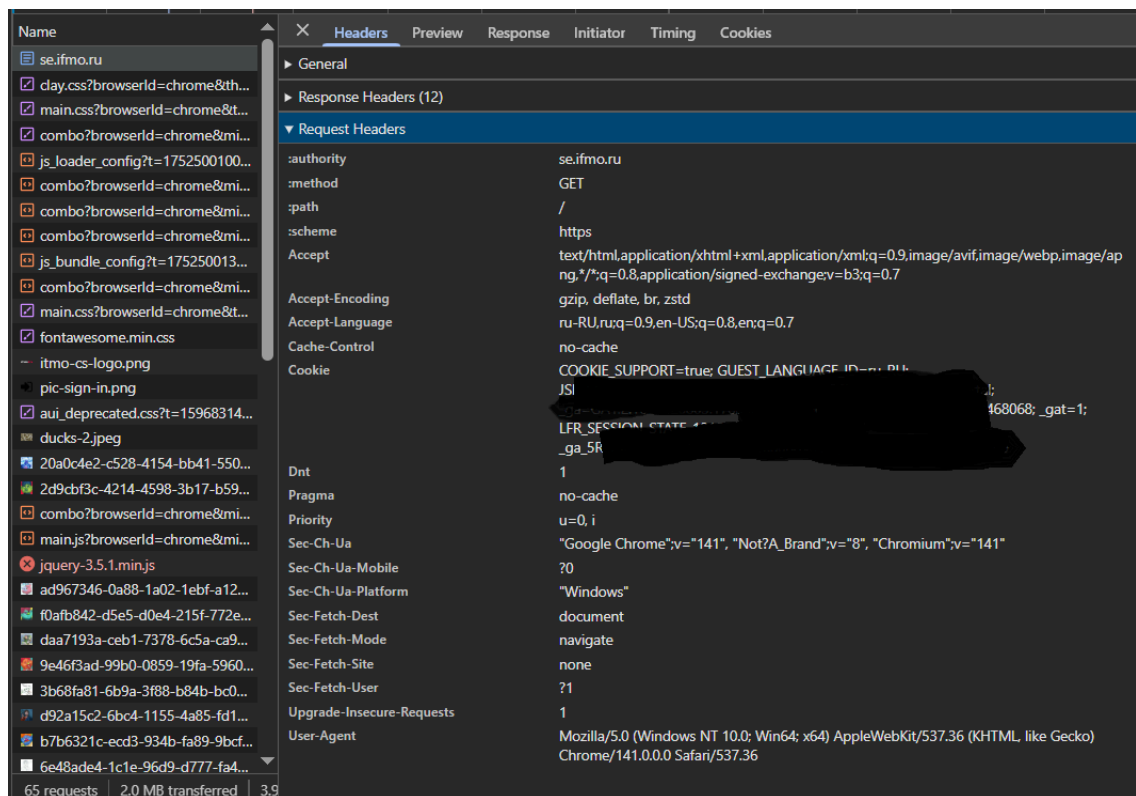


Рисунок 20 — Заголовки запроса

Заголовки идентификации браузера и системы:

- User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 - браузер представляется как Chrome 141 на Windows 10, что помогает серверу оптимизировать контент для конкретной платформы.
- Sec-Ch-Ua: "Google Chrome";v="141", "Not?A_Brand";v="8", "Chromium";v="141" - обновленная версия User-Agent для уменьшения цифрового отпечатка, сообщает только ключевую информацию о версии браузера.
- Sec-Ch-Ua-Mobile: ?0 - указание, что устройство не является мобильным.
- Sec-Ch-Ua-Platform: "Windows" - информация об операционной системе пользователя.
- Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7 - предпочтение интерфейса на русском языке с поддержкой английского как запасного варианта.

Заголовки управления запросом и кэшированием:

- Cache-Control: no-cache - требование не использовать кэшированную версию без предварительной проверки на сервере.
- Pragma: no-cache - устаревший аналог Cache-Control для обратной совместимости.
- Accept: text/html,application/xhtml+xml,application/xml;q=0.9 - предпочтительные форматы контента с указанием коэффициентов качества.
- Accept-Encoding: gzip, deflate, br, zstd - поддержка сжатия контента для уменьшения объема передаваемых данных.
- Priority: u=0, i - указание приоритета запроса как низкого.

Заголовки безопасности и приватности:

- Dnt: 1 - включенный заголовок "Do Not Track", выражающий предпочтение пользователя не отслеживаться.
- Sec-Fetch-Dest: document - указание, что запрашивается основной документ (HTML-страница).
- Sec-Fetch-Mode: navigate - запрос является навигационным (переход по ссылке или ввод URL).
- Sec-Fetch-Site: none - запрос инициирован напрямую, а не с другого сайта.
- Sec-Fetch-User: ?1 - указание, что запрос инициирован действием пользователя.
- Upgrade-Insecure-Requests: 1 - предпочтение использования HTTPS для всех ресурсов.

Работа с сессиями и аутентификацией:

- Cookie: COOKIE_SUPPORT=true; GUEST_LANGUAGE_ID=ru_RU; JSESSIONID=... - передача сессионных данных, включая идентификатор сессии JSESSIONID, язык интерфейса и аналитические cookies Google Analytics (_ga, _gid, _gat).
- :authority: se.ifmo.ru - целевой домен в формате HTTP/2.
- :method: GET - метод HTTP-запроса.
- :path: / - запрашиваемый путь на сервере.
- :scheme: https - использование защищенного протокола HTTPS.

Теперь рассмотрим POST-запрос (рисунок 21):

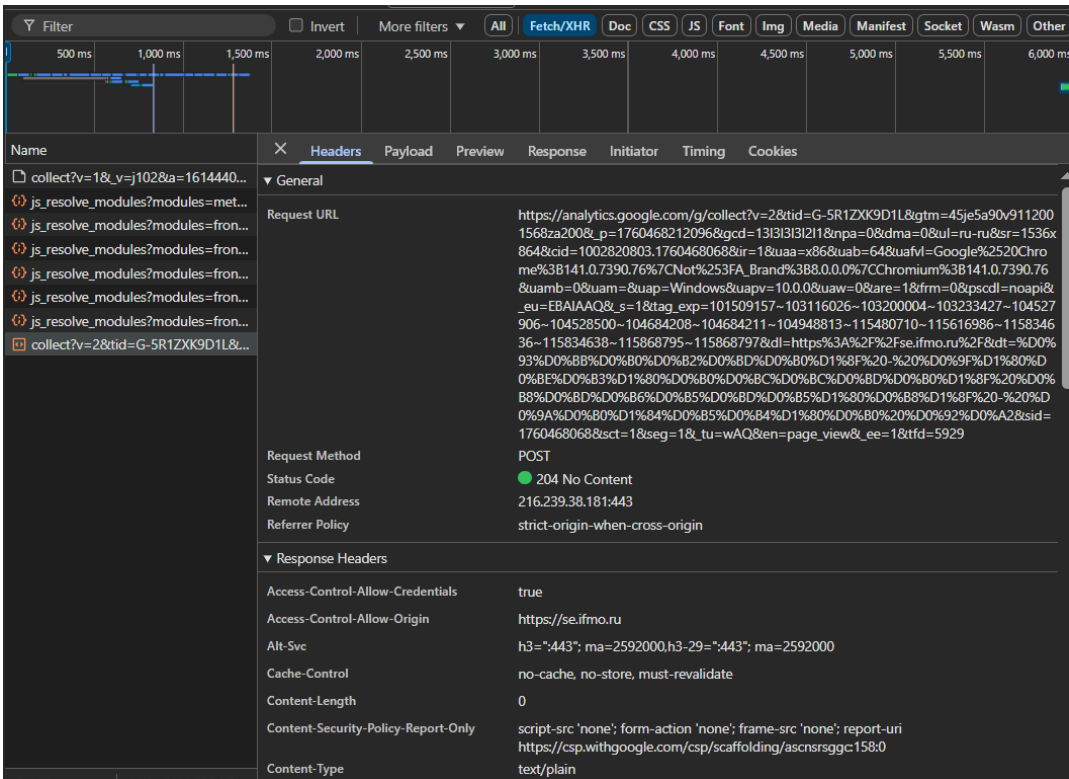


Рисунок 21 — Вкладка «Сеть» при POST-запросе

Видим те же общие заголовки с тем же назначением.

Рассмотрим заголовки ответа при POST-запросе (рисунок 22):

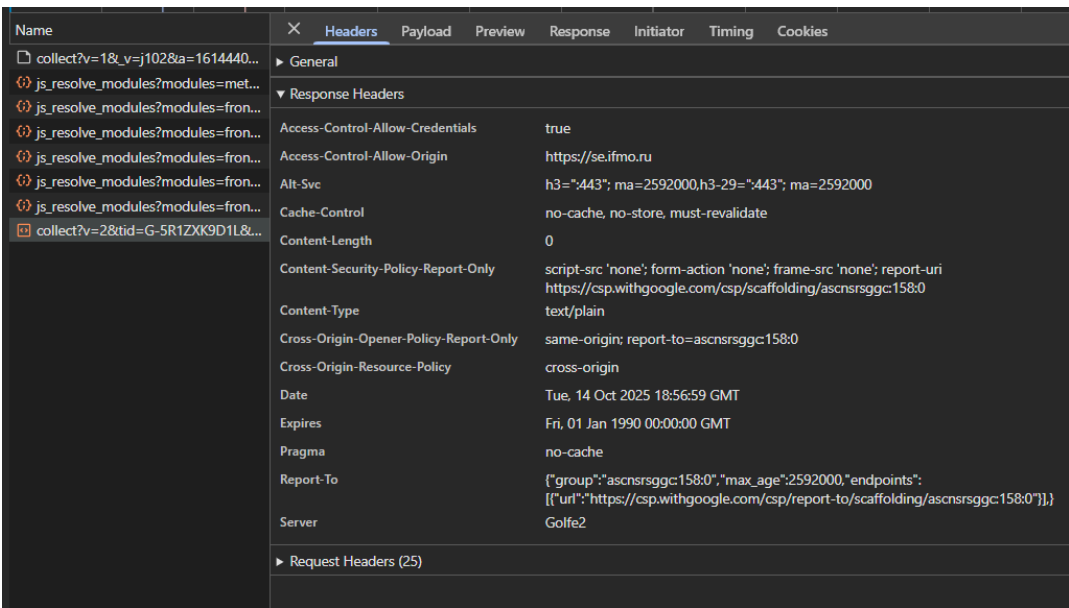


Рисунок 22 — Заголовки ответа при POST-запросе

Заголовки кросс-доменной политики (CORS):

- Access-Control-Allow-Origin: `https://se.ifmo.ru` - сервер явно разрешает кросс-доменные запросы только с конкретного домена `se.ifmo.ru`, что обеспечивает контролируемый доступ.
- Access-Control-Allow-Credentials: `true` - разрешает передачу cookies и аутентификационных данных при кросс-доменных запросах, что необходимо для работы сессий между разными доменами.
- Cross-Origin-Resource-Policy: `cross-origin` - политика разрешающая доступ к ресурсу с любого домена, что в сочетании с CORS настройками создает гибкую систему контроля доступа.

Заголовки безопасности контента:

- Content-Security-Policy-Report-Only: `script-src 'none'; form-action 'none'; frame-src 'none'` - политика безопасности контента в режиме отчетности: полный запрет на выполнение скриптов, отправку форм и встраивание фреймов, с отправкой отчетов о нарушениях.
- Cross-Origin-Opener-Policy-Report-Only: `same-origin; report-to=ascnsrsggc:158:0` - политика изоляции окон в режиме отчетности: запрет на совместное использование контекста с окнами из другого источника.
- Report-to: `{"group":"ascnsrsggc:158:0","max_age":2592000...` - настройка эндпоинта для отправки отчетов о нарушениях политик безопасности с периодом действия 30 дней.

Заголовки управления кэшированием:

- Cache-Control: `no-cache, no-store, must-revalidate` - строгая политика: запрет на кэширование, обязательная проверка актуальности данных на сервере.
- Expires: `Fri, 01 Jan 1990 00:00:00 GMT` - установка заведомо прошедшей даты истечения срока действия для гарантии отсутствия кэширования.
- Pragma: `no-cache` - устаревший заголовок запрета кэширования для обратной совместимости.

Технические заголовки:

- Content-Length: `0` - указание нулевого размера тела ответа, что характерно для ответов на POST-запросы без возвращаемых данных.
- Server: `Golfe2` - информация о серверном программном обеспечении.
- Alt-Svc: `h3=":443"; ma=2592000` - поддержка HTTP/3 на порту 443 с периодом доступности 30 дней.
- Date: `Tue, 14 Oct 2025 18:56:59 GMT` - точное время формирования ответа сервером.

Заголовки браузера и системы:

- User-Agent: Mozilla/5.0 (Windows NT 10.0... Chrome/141.0.0.0 - идентификация браузера Chrome 141 на Windows 10.
- Sec-Ch-Ua: "Google Chrome";v="141", "Not?A_Brand";v="8" - обновленная версия User-Agent с информацией о версии браузера.
- Sec-Ch-Ua-Mobile: ?0 - указание, что устройство не мобильное.
- Sec-Ch-Ua-Platform: "Windows" - информация об операционной системе.
- Accept-Language: ru-RU,ru;q=0.9 - предпочтение русского языка интерфейса.

Заголовки безопасности браузера:

- Sec-Fetch-Dest: empty - указание, что запрос не предназначен для конкретного типа ресурса.
- Sec-Fetch-Mode: no-cors - режим запроса без поддержки CORS, ограничивающий доступ к ответу.
- Sec-Fetch-Site: cross-site - указание, что запрос идет на другой домен (с se.ifmo.ru на analytics.google.com).
- Sec-Fetch-Storage-Access: none - отсутствие доступа к хранилищу при кросс-доменном запросе.
- Dnt: 1 - включенный заголовок "Do Not Track", выражающий предпочтение пользователя не отслеживаться.

Технические параметры:

- Content-Length: 0 - нулевой размер тела запроса, данные передаются в URL параметрах.
- Accept: */* - готовность принять любой тип контента в ответе.
- Accept-Encoding: gzip, deflate, br, zstd - поддержка всех современных методов сжатия.
- Priority: u=1, i - высокий приоритет запроса.

Специфичные заголовки Google Chrome:

- X-Browser-Channel: stable - указание стабильной версии браузера.
- X-Browser-Copyright: Copyright 2025 Google LLC - информация о правообладателе.
- X-Browser-Validation: AGaxImjg97xQkd0h3geRTArJi8Y= - криптографическая проверка подлинности браузера.
- X-Browser-Year: 2025 - год версии браузера.
- X-Client-Data: CJa2yQEIo7bJAQipncoBCP6PywE... - закодированные данные о вариациях клиента для A/B тестирования функциональности Chrome.

Форма входа

Далее на этом же сайте (<https://se.ifmo.ru/>) введем тестовые данные в форму входа, как показано на рисунке 24:

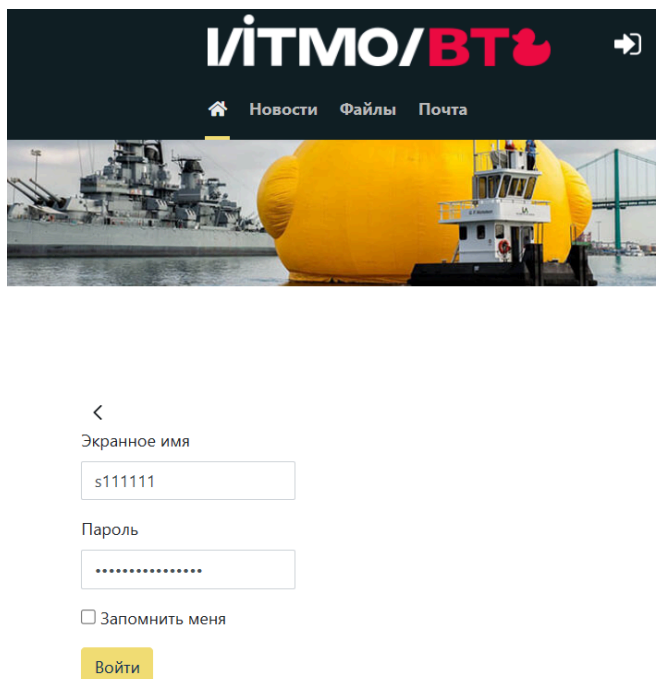


Рисунок 24 — Ввод тестовых данных

После ввода данных в форму появился такой POST-запрос (рисунки 25 и 26):

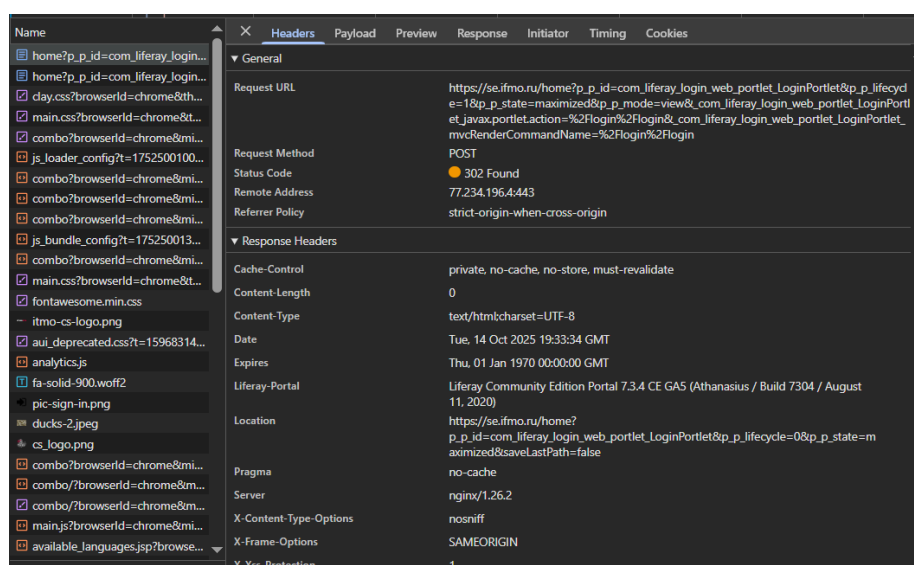


Рисунок 25 — POST-запрос после отправки данных формы, заголовки ответа

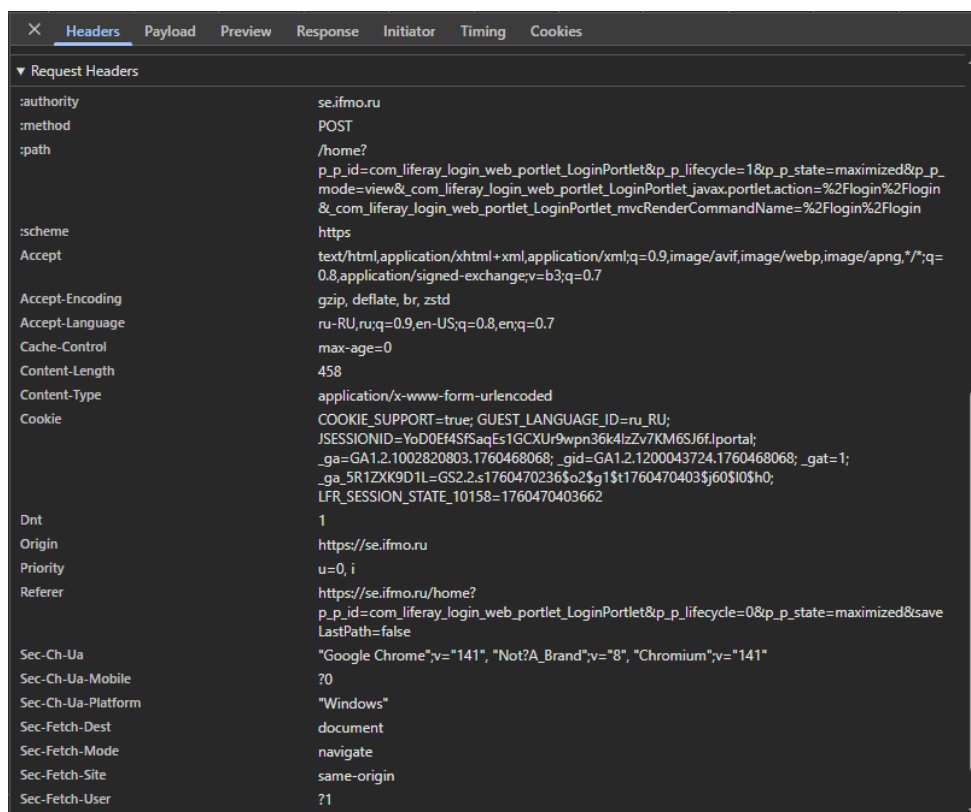


Рисунок 26 — POST-запрос после отправки данных формы, заголовки запроса

Если открыть полезную нагрузку, то увидим следующее (рисунок 27):

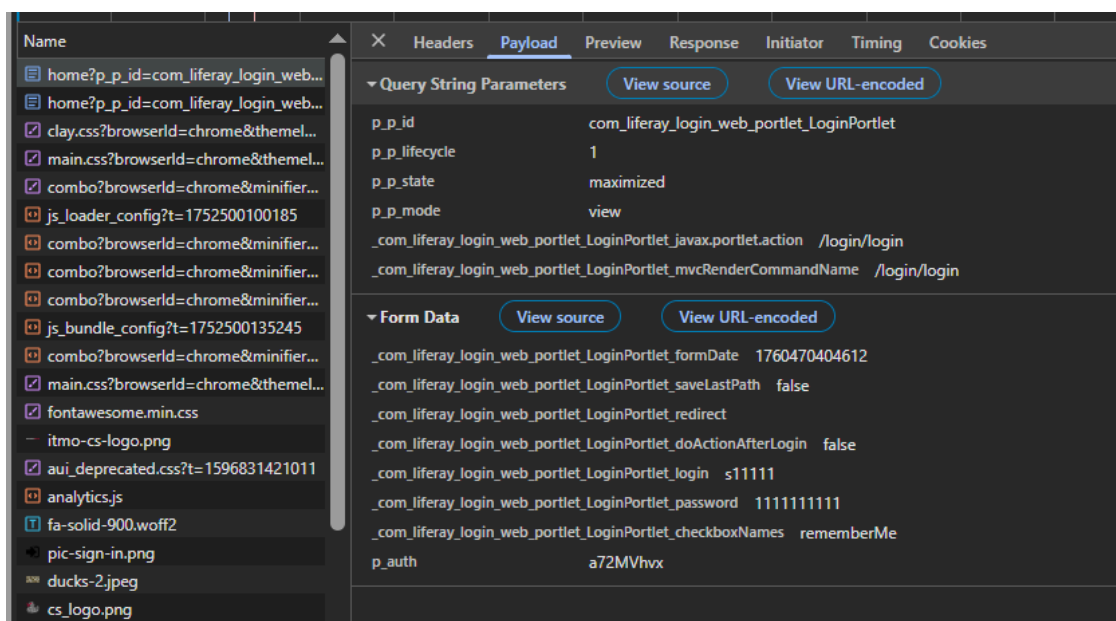


Рисунок 27 — Полезная нагрузка POST-запроса

Результаты анализа POST-запроса при авторизации:

Статус ответа: 302 Found – указывает на перенаправление после попытки входа, что является стандартным поведением для веб-приложений после отправки формы авторизации.

Пароль передается в открытом виде в поле `_com_liferay_login_web_portlet_LoginPortlet_password` со значением `1111111111`. Это представляет серьезную угрозу безопасности, даже при использовании HTTPS. Данные могут быть видны в логах промежуточных прокси-серверов. Кроме того, открытый пароль может сохраняться в истории форм. При компрометации SSL-сертификата пароль становится доступен.

Проанализируем передаваемые параметры:

Учетные данные:

- `_com_liferay_login_web_portlet_LoginPortlet_login`: `s11111` - логин пользователя
- `_com_liferay_login_web_portlet_LoginPortlet_password`: `1111111111` - пароль в открытом виде

Технические параметры формы:

- `_com_liferay_login_web_portlet_LoginPortlet_formDate`: `1760470404612` - временная метка для защиты от повторной отправки
- `p_auth`: `a72MVhvx` - токен аутентификации, вероятно CSRF-защита
- `_com_liferay_login_web_portlet_LoginPortlet_saveLastPath`: `false` - настройка сохранения пути

Нет хеширования на клиенте – пароль передается в исходном виде без предварительного хеширования. Нет дополнительного шифрования – только базовое HTTPS-соединение. Пароль виден в DevTools – легко обнаруживается при анализе сетевых запросов

Таким образом, система использует небезопасный метод передачи паролей. Несмотря на наличие HTTPS-шифрования и CSRF-токена, передача пароля в открытом виде нарушает современные стандарты безопасности. Данная реализация подвергает пользователей риску компрометации учетных записей даже при использовании защищенного соединения.

Как чаще всего передаются данные аутентификации (логин/пароль) в современных веб-приложениях?

В современных веб-приложениях данные аутентификации обычно передаются исключительно через защищенное HTTPS-соединение с использованием POST-запросов и Content-Type: application/x-www-form-urlencoded или application/json, где информация помещается в тело запроса, а не в URL.

Пароли не передаются в открытом виде — вместо этого применяется предварительное хеширование на стороне клиента с использованием алгоритмов вроде bcrypt или PBKDF2 с уникальной солью.

После успешной аутентификации дальнейшая работа происходит через одноразовые токены доступа (access tokens) и обновляемые refresh-токены, что исключает постоянную передачу пароля.

Обязательно используются CSRF-токены для защиты от межсайтовой подделки запросов, а сессионные данные хранятся в HTTP-only cookies с настройками SameSite, недоступные из JavaScript. Дополнительную безопасность обеспечивают современные заголовки HTTP Strict Transport Security, CSP и X-Content-Type-Options.

В передовых реализациях все чаще применяются протоколы OAuth 2.0, OpenID Connect, а также беспарольные методы аутентификации через email-ссылки, биометрию и многофакторную аутентификацию, что в совокупности обеспечивает надежную защиту учетных данных пользователей.