# Модуль III, Урок 3: Атаки на контроль доступа

## Оглавление

* [Теория](#теория)
* [Демонстрация](#демонстрация)
* [Практика с PortSwigger](#практика-с-portswigger)
* [Тест](#тест)

## Теория

**Контроль доступа** – это механизм, определяющий, какие действия и ресурсы разрешены для конкретных пользователей. Он работает на основе аутентификации (определения личности пользователя) и управления сессиями (отслеживания пользователя между запросами). После того как приложение *знает*, кто вы, и поддерживает вашу сессию, задача контроля доступа – решить, **что вам позволено делать** и **к чему вы можете обращаться**.

### Типы уязвимостей контроля доступа

Контроль доступа делится на три категории: **вертикальный**, **горизонтальный** и **контекстно-зависимый** (бизнес-логический):

* **Вертикальный контроль доступа** разделяет права между разными уровнями пользователей (например, обычный пользователь vs. администратор). Вертикальное нарушение – это *повышение привилегий*: ситуация, когда пользователь получает доступ к функциям, не предназначенным для его роли. Например, если обычный пользователь может открыть административную страницу и удалить учетные записи, это вертикальное повышение привилегий.
* **Горизонтальный контроль доступа** ограничивает каждого пользователя набором объектов одного типа. Пользователю разрешено работать только со *своими* данными (письмами, счетами, заказами и т.п.). Горизонтальное нарушение – это *обход ограничений по горизонтали*: когда пользователь получает доступ к ресурсам *другого* пользователя на том же уровне. Например, если клиент банка может видеть баланс чужого счета, или пользователь почты читает чужие письма – это горизонтальное повышение привилегий.
* **Контекстно-зависимый (бизнес-логический) контроль доступа** учитывает состояние приложения или последовательность действий. Он гарантирует правильный *порядок* операций. Нарушение здесь позволяет обойти бизнес-логику приложения – например, пройти шаги процесса не в том порядке или пропустить важный этап. Классический пример: злоумышленник обходит этап оплаты в интернет-магазине и получает товар бесплатно.

**Замечание:** Все три типа уязвимостей сводятся к одному: **пользователь делает то, что не должен иметь возможности сделать**. Различаются только сценарии и методы эксплуатации.

Нарушение контроля доступа может иметь каскадный эффект. Например, удачная *горизонтальная* атака часто ведет к *вертикальной*: получив доступ к чужой учетной записи, злоумышленник может взять на прицел аккаунт с повышенными правами. Классический случай – сначала через уязвимость доступа сменить пароль другого пользователя, а затем захватить учетную запись администратора.

Наихудшая ситуация – когда **неаутентифицированный** (гость) может получить доступ к привилегированным функциям. Такие критические сбои контроля доступа встречаются реже, но приводят к полному компромету системы.

### Распространённые уязвимости и ошибки реализации

Большинство проблем с контролем доступа возникают из-за ошибок разработчиков при внедрении проверок. Ниже перечислены распространённые уязвимости контроля доступа:

* **Полностью незащищённая функциональность.** В приложении есть *скрытые* разделы (например, админ-панель), которые не отображаются обычным пользователям, но *никак не защищены* на сервере. Например, административные страницы доступны по прямому URL всем желающим. Если злоумышленник узнает или угадает адрес (/admin, /administrator-panel и т.п.), он получит к ним доступ. Разработчики могут попытаться *спрятать* такие страницы (security through obscurity), например дать им нестандартный URL и не показывать ссылки. Однако это слабая мера: URL можно обнаружить в коде (например, в файлах robots.txt или скриптах интерфейса) или подобрать перебором. **Пример:** Файл robots.txt может содержать строку Disallow: /admin – этот запрет для поисковых ботов невольно раскрывает путь к админ-разделу[[1]](https://portswigger.net/web-security/access-control/lab-unprotected-admin-functionality#:~:text=).
* **Функции, основанные на идентификаторах.** Сюда относятся уязвимости типа **IDOR (Insecure Direct Object Reference)** – когда приложение использует переданный от клиента идентификатор ресурса (ID объекта) без надлежащей проверки прав. Если идентификатор предсказуемый или известный, злоумышленник может подставить чужой ID и получить доступ к чужим данным. **Пример:** Пользователь переходит по адресу /profile?user\_id=123 для просмотра своего профиля. Если заменить 123 на 124, приложение покажет профиль другого пользователя – значит, горизонтальный контроль отсутствует. Подобные уязвимости часто встречаются в API и URL, принимающих номера счетов, id документов, имена файлов и т.д.
* **Многоэтапные процессы без полноценных проверок.** Многие ключевые действия реализованы через несколько шагов (пошаговые мастера, процедуры с подтверждением). Уязвимость возникает, если разработчики проверяют права *только на одних шагах*, предположив, что до завершающего шага пользователь дойдет легитимно. Злоумышленник может пропустить начальные шаги и отправить запрос непосредственно на финальный URL, минуя проверки. **Пример:** В приложении есть админ-функция «Повысить права пользователя», которая реализована в 2 шага: (1) выбрать пользователя и (2) подтвердить действие. Если проверка «администратор ли это?» сделана только на шаге 1, но не дублируется на шаге 2, то обычный пользователь может напрямую вызвать запрос подтверждения с нужными параметрами и повысить себе роль[[2]](https://portswigger.net/web-security/access-control#:~:text=Sometimes%2C%20a%20website%20will%20implement,step%20with%20the%20required%20parameters)[[3]](https://portswigger.net/web-security/access-control#:~:text=administrative%20page%20at%20,then%20the%20request%20is%20allowed).
* **Незащищённые статические ресурсы.** Статичные файлы (документы, изображения, отчёты и пр.), лежащие на сервере, зачастую отдаются веб-сервером напрямую, обходя логику приложения. Если URL статического контента угадываемый, злоумышленник может скачать чужой ресурс без проверки доступа. **Пример:** После покупки книги приложение перенаправляет пользователя на https://site.com/download/9780123456789.pdf. Если сервер не проверяет права доступа, зная шаблон (например, ISBN книги), атакующий может перебором скачать все книги издательства[[4]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%BA%D0%B0%D1%82%D0%B0%D0%BB%D0%BE%D0%B3%20%D0%BA%D0%BD%D0%B8%D0%B3%20%D0%B8%20%D0%BF%D0%BE%D0%BA%D1%83%D0%BF%D0%B0%D1%82%D1%8C%20%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5,%D0%92%20%D0%B4%D0%B0%D0%BD%D0%BD%D0%BE%D0%BC%20%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B5%20%D0%B8%D0%BC%D1%8F%20%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0)[[5]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%B0%2C%20%D0%B8%20%D1%87%D1%82%D0%BE%20%D0%BB%D1%8E%D0%B1%D0%BE%D0%B9%2C%20%D0%BA%D1%82%D0%BE,%D1%84%D0%B0%D0%B9%D0%BB%D0%B0%D0%BC%20%D0%B8%20%D0%B4%D1%80%D1%83%D0%B3%D0%B8%D0%BC%20%D0%BA%D0%BE%D0%BD%D1%84%D0%B8%D0%B4%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%BC). Решение – либо перенести выдачу файла в код приложения с проверкой, либо настроить веб-сервер так, чтобы защищать папку с файлами (напр. с помощью авторизации или привязки к сессии).
* **Ошибки конфигурации платформы.** Некоторые приложения полагаются на средства веб-сервера или фреймворка для контроля доступа (например, настроены правила, какие URL доступны ролям). Если эти правила неполны или несовершенны, их можно обойти. Возможные ошибки:
* *Неполный учет HTTP-методов.* Например, запретить метод POST на /admin/deleteUser для неролей, но не запретить GET. Если приложение (или платформа) не различает методы, то отправив запрос как GET /admin/deleteUser?..., злоумышленник обойдет запрет[[6]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%9A%D0%B0%D0%BA%20%D0%BE%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%BE%20%D0%B2%20%D0%93%D0%BB%D0%B0%D0%B2%D0%B5%203%2C,%D1%8D%D1%82%D0%BE%D1%82%20%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8C%2C%20%D0%BE%D1%82%D0%BF%D1%80%D0%B0%D0%B2%D0%B8%D0%B2%20%D1%82%D0%BE%D1%82%20%D0%B6%D0%B5)[[7]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%BD%D0%B5%D1%81%D0%B0%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%BC%D1%83%20%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D1%83,%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81%D0%B5%20%D0%B4%D0%BB%D1%8F%20%D0%BD%D0%B5%D1%81%D0%B0%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%20%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8). Более того, HEAD-запросы часто обрабатываются так же, как GET (но без тела ответа). Злоумышленник может воспользоваться методами HEAD или даже выдуманным методом (POSTX) – некоторые сервера в ответ на неизвестный метод все равно вызовут обработчик GET. Таким образом можно обойти ограничения на методы[[8]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%A7%D1%82%D0%BE%20%D0%BD%D0%B0%20%D0%BF%D0%B5%D1%80%D0%B2%D1%8B%D0%B9%20%D0%B2%D0%B7%D0%B3%D0%BB%D1%8F%D0%B4%20%D0%B5%D1%89%D0%B5,%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81%D1%8B%20%D1%87%D0%B0%D1%81%D1%82%D0%BE)[[9]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%92%20%D0%BD%D0%B5%D0%BA%D0%BE%D1%82%D0%BE%D1%80%D1%8B%D1%85%20%D1%81%D0%BB%D1%83%D1%87%D0%B0%D1%8F%D1%85%20%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D1%8B%20%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%B0%D1%82%D1%8B%D0%B2%D0%B0%D1%8E%D1%82,%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81%D0%B5%20%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9%20%D0%BD%D0%B5%D0%B4%D0%B5%D0%B9%D1%81%D1%82%D0%B2%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9%20%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%20HTTP).
* *Несоответствие правил и реальных URL.* Если механизм контроля сопоставляет URL строго, а приложение принимает разные варианты, возможны расхождения. Например, приложение может считать /Admin/Panel эквивалентом /admin/panel (игнорируя регистр или слэш на конце), а модуль контроля – нет. Тогда запрос с нестандартным написанием может миновать защиту[[10]](https://portswigger.net/web-security/access-control#:~:text=,matching%20discrepancies)[[11]](https://portswigger.net/web-security/access-control#:~:text=enabled%20the%20,option%20is%20enabled%20by%20default).
* *Специальные заголовки.* Некоторые веб-серверы/фреймворки поддерживают нестандартные заголовки (X-Original-URL, X-Rewrite-URL), позволяющие переопределить URL запроса. Если контроль доступа смотрит только на исходный путь, а приложение реально обрабатывает путь из заголовка, атакующий добавит заголовок и обойдет правило[[12]](https://portswigger.net/web-security/access-control#:~:text=leading%20to%20access%20control%20bypasses)[[13]](https://portswigger.net/web-security/access-control#:~:text=Some%20application%20frameworks%20support%20various,a%20request%20like%20the%20following).
* **Небезопасные методы контроля доступа.** Существуют заведомо уязвимые подходы к проверке прав, от которых следует отказаться:
* *Контроль доступа на основе параметра, отправляемого клиентом.* Приложение решает, админ пользователь или нет, на основании *данных от самого пользователя* – например, скрытого поля формы, куки или параметра URL вроде role=admin или Admin=true. Такое значение легко подделать и получить доступ к запрещенным функциям[[14]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%9A%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8C%20%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%B0%20%D0%BD%D0%B0%20%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%B5%20%D0%BF%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%D0%BE%D0%B2,%D0%B0%D0%B4%D1%80%D0%B5%D1%81%D0%B0%20%D1%81%D0%BB%D0%B5%D0%B4%D1%83%D1%8E%D1%89%D0%B5%D0%B3%D0%BE%20%D0%B2%D0%B8%D0%B4%D0%B0)[[15]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=https%3A%2F%2Fwahh,%D0%B2%20%D1%80%D0%B0%D1%81%D0%BA%D1%80%D1%8B%D1%82%D0%B8%D0%B8%20%D1%8D%D1%82%D0%BE%D0%B3%D0%BE%20%D0%BC%D0%B5%D1%85%D0%B0%D0%BD%D0%B8%D0%B7%D0%BC%D0%B0%20%D0%BF%D1%80%D0%B8). Поскольку параметр находится под контролем клиента, злоумышленник просто установит admin=true в своих запросах и станет администратором.
* *Контроль доступа на основе заголовка Referer.* Приложение проверяет, откуда пришел запрос, и на этом основании решает, разрешать ли действие. Например, прямой переход на админ-страницу /admin/deleteUser разрешается только если Referer содержит /admin (то есть запрос инициирован из админского меню). Такой подход неэффективен: заголовок Referer полностью контролируется пользователем и может быть отправлен с любым нужным значением[[16]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%9A%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8C%20%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%B0%20%D0%BD%D0%B0%20%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%B5%20Referer,%D0%AD%D1%82%D0%B0%20%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C)[[17]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D1%84%D1%83%D0%BD%D0%B4%D0%B0%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%20%D0%BF%D0%BE%D1%80%D0%BE%D1%87%D0%BD%D0%B0%2C%20%D0%BA%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D0%BE%20%D0%B6%D0%B5%2C%20%D0%BF%D0%BE%D1%82%D0%BE%D0%BC%D1%83,%D0%B1%D1%8B%D1%82%D1%8C%20%D1%83%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%20%D0%B2%20%D0%BB%D1%8E%D0%B1%D0%BE%D0%B5%20%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D0%B5).
* *Контроль доступа на основе местоположения.* Ограничение функций по геолокации пользователя (IP-адресу или GPS) легко обходится с помощью прокси или VPN. Если бизнес-логика требует гео-ограничений, их следует воспринимать лишь как *дополнительный* фактор, но не как надежную защиту от злоумышленника[[18]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%9C%D0%BD%D0%BE%D0%B3%D0%B8%D0%B5%20%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D0%B8%20%D0%B8%D0%BC%D0%B5%D1%8E%D1%82%20%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B5%20%D0%B8%D0%BB%D0%B8,%D0%BF%D1%80%D0%BE%D0%BA%D1%81%D0%B8%2C%20%D1%80%D0%B0%D1%81%D0%BF%D0%BE%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%B2%20%D1%82%D1%80%D0%B5%D0%B1%D1%83%D0%B5%D0%BC%D0%BE%D0%BC%20%D0%BC%D0%B5%D1%81%D1%82%D0%B5)[[19]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%97%D0%BB%D0%BE%D1%83%D0%BC%D1%8B%D1%88%D0%BB%D0%B5%D0%BD%D0%BD%D0%B8%D0%BA%D1%83%20%D0%BE%D1%82%D0%BD%D0%BE%D1%81%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%20%D0%BB%D0%B5%D0%B3%D0%BA%D0%BE%20%D0%BE%D0%B1%D0%BE%D0%B9%D1%82%D0%B8%20%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8C,%D0%9F%D1%80%D1%8F%D0%BC%D0%BE%D0%B5%20%D0%BC%D0%B0%D0%BD%D0%B8%D0%BF%D1%83%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%BA%D0%BB%D0%B8%D0%B5%D0%BD%D1%82%D1%81%D0%BA%D0%B8%D0%BC%D0%B8%20%D0%BC%D0%B5%D1%85%D0%B0%D0%BD%D0%B8%D0%B7%D0%BC%D0%B0%D0%BC%D0%B8%20%D0%B3%D0%B5%D0%BE%D0%BB%D0%BE%D0%BA%D0%B0%D1%86%D0%B8%D0%B8).

### Методики атак и тестирования контроля доступа

Тестирование контроля доступа – трудоемкий процесс, ведь **каждый запрос** в приложении потенциально требует проверки прав. Ниже представлены методы, которые помогают выявлять уязвимости контроля доступа:

* **Тестирование с разными учетными записями.** Самый наглядный подход – получить несколько аккаунтов разного уровня и сравнить их возможности. Зайдите под администратором или пользователем с расширенными правами, изучите все доступные функции (разделы меню, страницы). Затем войдите под обычной учетной записью и *попробуйте открыть* те URL или выполнить действия, которые вы видели у привилегированного пользователя. Если «низший» пользователь получает доступ к тому, что предназначено только для «высшего», – налицо вертикальное повышение привилегий[[20]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%9F%D0%A0%D0%90%D0%9A%D0%A2%D0%98%D0%A7%D0%95%D0%A1%D0%9A%D0%98%D0%95%20%D0%A8%D0%90%D0%93%D0%98%20%D0%92%D0%BE%D1%82%20%D0%BD%D0%B5%D1%81%D0%BA%D0%BE%D0%BB%D1%8C%D0%BA%D0%BE%20%D0%B2%D0%BE%D0%BF%D1%80%D0%BE%D1%81%D0%BE%D0%B2%2C,%D0%B4%D0%BB%D1%8F%20%D0%B5%D0%B3%D0%BE%20%D0%BD%D0%B0%D1%81%D1%82%D1%80%D0%BE%D0%B9%D0%BA%D0%B8%20%D0%B8%20%D0%BC%D0%BE%D0%BD%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3%D0%B0)[[21]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%BE%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%BD%D0%BE%D0%BC%D1%83%20%D0%BF%D0%BE%D0%B4%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D1%83%20%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%2C%20%D0%BA%D0%BE%D1%82%D0%BE%D1%80%D0%BE%D0%B5%20%D0%BF%D1%80%D0%B8%D0%BD%D0%B0%D0%B4%D0%BB%D0%B5%D0%B6%D0%B8%D1%82,%D1%82%D0%BE%D0%BC%2C%20%D1%87%D1%82%D0%BE%20%D0%BF%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%20%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D1%83%D0%B5%D1%82%D1%81%D1%8F%20%D0%B4%D0%BB%D1%8F). Аналогично, если два обычных пользователя имеют доступ каждый к своим данным, попробуйте под одним пользователем обратиться к ресурсам другого – проверяя горизонтальное разделение. Это можно делать вручную, копируя ссылки и ID, или автоматизировать. Инструмент **Burp Suite** обладает функцией *сравнения карты сайта* для двух разных сессий, что значительно ускоряет поиск расхождений в контенте для разных пользователей[[22]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D1%87%D0%B5%D0%BB%D0%BE%D0%B2%D0%B5%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B3%D0%BE%20%D0%B8%D0%BD%D1%82%D0%B5%D0%BB%D0%BB%D0%B5%D0%BA%D1%82%D0%B0%20%D0%B4%D0%BB%D1%8F%20%D1%8D%D1%84%D1%84%D0%B5%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%B2%D1%8B%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F,%D0%BA%D0%B0%D0%B6%D0%B4%D0%BE%D0%BC%D1%83%20%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8E%2C%20%D1%81%D0%BE%D0%B2%D0%BF%D0%B0%D0%B4%D0%B0%D0%B5%D1%82%20%D0%B8%D0%BB%D0%B8%20%D1%80%D0%B0%D0%B7%D0%BB%D0%B8%D1%87%D0%B0%D0%B5%D1%82%D1%81%D1%8F)[[23]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=2,%D1%84%D0%B0%D0%B9%D0%BB%20%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D1%8F%2C%20%D0%BF%D0%BE%D0%BB%D1%83%D1%87%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9%20%D0%BF%D0%BE%D1%81%D0%BB%D0%B5%20%D0%BA%D0%B0%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F). Автоматическое сравнение подсвечивает отличия, но **требует анализа эксперта**: не каждое отличие – уязвимость, и не каждый одинаковый ответ – безопасен[[24]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81%D0%B5%20%D0%BD%D0%B5%D1%82%20%D0%BD%D0%B0%20%D0%BD%D0%B5%D0%B5%20%D1%81%D1%81%D1%8B%D0%BB%D0%BE%D0%BA,%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D1%8E%20%D0%BF%D1%80%D0%BE%D1%84%D0%B8%D0%BB%D1%8F%20%D0%B4%D0%BB%D1%8F%20%D1%82%D0%B5%D0%BA%D1%83%D1%89%D0%B5%D0%B3%D0%BE%20%D0%B2%D0%BE%D1%88%D0%B5%D0%B4%D1%88%D0%B5%D0%B3%D0%BE)[[25]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%BD%D0%B5%D0%B4%D0%BE%D1%81%D1%82%D0%B0%D1%82%D0%BE%D1%87%D0%BD%D0%BE%20%D0%B4%D0%BB%D1%8F%20%D0%BE%D1%86%D0%B5%D0%BD%D0%BA%D0%B8%20%D1%8D%D1%84%D1%84%D0%B5%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D0%B8%20%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8F,%D0%98%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D1%83%D1%8F). Тем не менее, этот метод помогает быстро сузить область поиска потенциальных проблем.
* **Тестирование многоэтапных процессов.** Пошаговые процедуры (создание заказа, изменение роли, перевод денег и т.д.) нужно проверять, атакуя *каждый этап отдельно*. Даже если первые шаги требуют определенных прав или определенной последовательности, убедитесь, что *на каждом запросе* есть проверка. Попробуйте отправить запросы финальных шагов без выполнения начальных, изменить параметры, которые передаются через клиент между шагами, и т.п. Цель – выяснить, не предполагает ли приложение ошибочно, что раз пользователь дошел до финала через UI, то он авторизован. Любое подобное предположение дает лазейку: злоумышленник может эмулировать последний шаг и обойти предыдущие проверки[[26]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%BF%D1%80%D0%B8%D1%87%D0%B8%D0%BD%D0%B0%D0%BC%2C%20%D0%BD%D0%B5%20%D1%81%D0%B2%D1%8F%D0%B7%D0%B0%D0%BD%D0%BD%D1%8B%D0%BC%20%D1%81%20%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5%D0%BC,%D0%9A%D0%B0%D0%B6%D0%B4%D1%8B%D0%B9%20%D1%88%D0%B0%D0%B3%20%D1%8D%D1%82%D0%BE%D0%B3%D0%BE%20%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%B0)[[27]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%BD%D0%B5%D0%BE%D0%B1%D1%85%D0%BE%D0%B4%D0%B8%D0%BC%D0%BE%20%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D1%82%D1%8C%20%D0%B8%D0%BD%D0%B4%D0%B8%D0%B2%D0%B8%D0%B4%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%2C%20%D1%87%D1%82%D0%BE%D0%B1%D1%8B%20%D1%83%D0%B1%D0%B5%D0%B4%D0%B8%D1%82%D1%8C%D1%81%D1%8F%2C,net%2Fauth%2F471).
* **Тестирование с ограниченным доступом (одна учетная запись).** Если у вас в распоряжении только обычный пользователь (или даже доступ без входа), придется действовать изобретательно:
* Проведите **инвентаризацию функциональности**: используя техники обнаружения скрытого контента (сканирование путей, перебор типичных имен файлов и директорий, просмотр публичных файлов вроде /robots.txt и /sitemap.xml). Цель – найти страницы, которые **не видны** текущему пользователю, но могут существовать (устаревшие разделы, новые фичи, админские панели). Часто даже низкопривилегированный аккаунт может напрямую перечислить некоторые чувствительные функции, если знать где искать[[28]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%B2%D1%81%D0%B5%D1%81%D1%82%D0%BE%D1%80%D0%BE%D0%BD%D0%BD%D0%B5%D0%B3%D0%BE%20%D1%82%D0%B5%D1%81%D1%82%D0%B0%20%D0%B2%20%D0%BB%D1%8E%D0%B1%D0%BE%D0%BC%20%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B5,%D0%BD%D0%B8%D0%B7%D0%BA%D0%BE%D0%BF%D1%80%D0%B8%D0%B2%D0%B8%D0%BB%D0%B5%D0%B3%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F%20%D1%87%D0%B0%D1%81%D1%82%D0%BE%20%D0%B1%D1%8B%D0%B2%D0%B0%D0%B5%D1%82%20%D0%B4%D0%BE%D1%81%D1%82%D0%B0%D1%82%D0%BE%D1%87%D0%BD%D0%BE)[[29]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D1%81%D1%83%D1%89%D0%B5%D1%81%D1%82%D0%B2%D0%BE%D0%B2%D0%B0%D1%82%D1%8C%20%D0%BF%D0%BB%D0%BE%D1%85%D0%BE%20%D0%B7%D0%B0%D1%89%D0%B8%D1%89%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F%20%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C%2C%20%D0%BD%D0%B0,%D0%BF%D0%BE%D0%BB%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%BF%D1%80%D1%8F%D0%BC%D0%BE%D0%B3%D0%BE%20%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%B0%20%D0%BA%20%D0%BA%D0%BE%D0%BD%D1%84%D0%B8%D0%B4%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B9).
* На подозрительных страницах попробуйте добавлять параметры типа admin=true или role=admin в URL и формы – вдруг приложение использует их (см. выше про небезопасные методы). Также удаление или модификация заголовка **Referer** в важных запросах: если запрос без Referer отклоняется, вероятно приложение доверяет этому заголовку[[30]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=3,%D1%81%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D1%8B%20%D0%B2%20%D0%BF%D0%BE%D0%B8%D1%81%D0%BA%D0%B5%20%D1%81%D1%81%D1%8B%D0%BB%D0%BE%D0%BA%20%D0%BD%D0%B0)[[31]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%B4%D0%BB%D1%8F%20%D0%BF%D1%80%D0%B8%D0%BD%D1%8F%D1%82%D0%B8%D1%8F%20%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B9%20%D0%BE%20%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5,%D0%B7%D0%BD%D0%B0%D1%87%D0%B8%D0%BC%D0%BE%D0%BC%D1%83%20%D0%B8%D0%B7%D0%BC%D0%B5%D0%BD%D0%B5%D0%BD%D0%B8%D1%8E%20%D0%B2%20%D0%BE%D1%82%D0%B2%D0%B5%D1%82%D0%B5%20%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F).
* Анализируйте весь получаемый HTML/JavaScript. Иногда в коде интерфейса присутствуют *скрытые ссылки* или фрагменты, рассчитанные на другой уровень прав (например, скрипт, который добавляет кнопку админа, если isAdmin=true – такой код выдаст название секретного URL[[32]](https://portswigger.net/web-security/access-control#:~:text=This%20might%20not%20be%20directly,based%20on%20the%20user%27s%20role)[[33]](https://portswigger.net/web-security/access-control#:~:text=%60,script)). Исследуйте также исходники подключаемых файлов, скриптов, сторонних плагинов – там могут быть прямые ссылки на защищенные ресурсы.
* Проводите **фаззинг идентификаторов**. Если приложение использует числовые ID, попробуйте подбирать разные значения. Сгенерируйте несколько объектов (записей) под своей учеткой, посмотрите, какие ID им присвоены, и попытайтесь угадать или перебрать близкие ID, принадлежащие другим. Если ID выглядят случайными (например GUID), атака перебором вряд ли успешна. Но если они предсказуемы или последовательны – горизонтальная атака очень вероятна. Инструменты вроде **Burp Intruder** помогут автоматизировать перебор ID и сбор ответов. В случае удачи можно выгрузить много данных сразу (например, все счета клиентов, все переписки и т.д.)[[34]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%9F%D0%A0%D0%90%D0%9A%D0%A2%D0%98%D0%A7%D0%95%D0%A1%D0%9A%D0%98%D0%95%20%D0%A8%D0%90%D0%93%D0%98%201,%D0%B2%20%D0%B8%D0%B4%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82%D0%BE%D1%80%D0%B0%D1%85%2C%20%D0%BA%D0%BE%D1%82%D0%BE%D1%80%D1%8B%D0%B5%20%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%B5%D1%82%20%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5)[[35]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=3,%D0%BA%D0%B0%D1%81%D1%82%D0%BE%D0%BC%D0%BD%D0%BE%D0%B9%20%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B9%20%D0%B0%D1%82%D0%B0%D0%BA%D0%B8%20%D1%81%20%D1%86%D0%B5%D0%BB%D1%8C%D1%8E).

**Совет:** Обнаружив уязвимость горизонтального контроля доступа, сразу подумайте о дальнейшем повышении привилегий. Если вы смогли собрать данные о чужих учетных записях (например, список пользователей и хэши паролей), попробуйте найти среди них административную учетку. Часто администраторы имеют ID 1, 2 или другие крайние значения – проверьте их первыми. Либо используйте скрипт перебора: автоматически входите под каждым украденным логином и проверяйте что-то, доступное только администратору (например, чужую домашнюю страницу). Так вы быстро вычислите, какой аккаунт обладает правами администратора[[36]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=http%3A%2F%2Fmdsec,%D1%87%D1%82%D0%BE%20%D1%81%D0%B0%D0%BC%D1%8B%D0%B5%20%D0%BD%D0%B8%D0%B7%D0%BA%D0%B8%D0%B5%20%D0%BD%D0%BE%D0%BC%D0%B5%D1%80%D0%B0%20%D0%BF%D1%80%D0%B8%D1%81%D0%B2%D0%BE%D0%B5%D0%BD%D1%8B)[[37]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%BD%D0%B5%20%D0%BE%D0%B1%D1%80%D0%B0%D0%B4%D1%83%D0%B5%D1%82,%D0%92%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%2C%20%D0%B0%D0%B4%D0%BC%D0%B8%D0%BD%D0%B8%D1%81%D1%82%D1%80%D0%B0%D1%82%D0%BE%D1%80%D1%8B%20%D0%BC%D0%BE%D0%B3%D1%83%D1%82).

* **Прямой вызов серверных методов.** В некоторых архитектурах веб-приложение может предоставлять точку входа для удаленных вызовов (например, общий servlet или API-метод, которому передается имя функции и параметры). Такие бэкэнд-методы тоже нужно проверять. Используйте перечисление, чтобы выявить скрытые методы, и попытайтесь вызывать их напрямую. Особенно опасны известные унаследованные методы, которые могут быть оставлены «по умолчанию». **Пример:** В Java-сервлетах существует диагностический метод IBMTrackerDebug – вызвав POST /svc с параметром servlet=com.ibm.ws.webcontainer.httpsession.IBMTrackerDebug, можно получить чувствительную отладочную информацию из сессий[[38]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%BF%D1%80%D1%8F%D0%BC%D0%BE%D0%B3%D0%BE%20%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%B0%20%D0%BA%20%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%B0%D0%BC,Encoding%3A%20gzip%2C%20deflate)[[39]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=servlet%3Dcom,%D1%81%D0%BC%D0%BE%D0%B6%D0%B5%D1%82%D0%B5%20%D0%BF%D0%BE%D0%BB%D1%83%D1%87%D0%B8%D1%82%D1%8C%20%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%20%D0%B8%20%D0%BA). Если такие вызовы не требуют авторизации, это брешь.
* **Проверка доступа к статическим ресурсам.** Как отмечалось, прямые запросы к файлам могут обходить проверки. Поэтому:
* Выполните легитимный сценарий скачивания защищенного файла под *правильным* пользователем, посмотрите конечный URL (например, ссылка на PDF отчет после авторизации).
* Затем, не будучи авторизованным или под другим аккаунтом, попробуйте перейти напрямую по этому URL. Если удалось – налицо уязвимость.
* Попытайтесь понять систему именования файлов и, возможно, автоматизировать скачивание множества файлов (например, всех отчетов за разные периоды) путём перебора имен[[40]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%9F%D0%A0%D0%90%D0%9A%D0%A2%D0%98%D0%A7%D0%95%D0%A1%D0%9A%D0%98%D0%95%20%D0%A8%D0%90%D0%93%D0%98%201,%D0%95%D1%81%D0%BB%D0%B8%20%D0%B2%D0%BE%D0%B7%D0%BC%D0%BE%D0%B6%D0%BD%D0%BE%2C%20%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%B9%D1%82%D0%B5)[[41]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D1%82%D1%80%D0%B5%D0%B1%D1%83%D0%B5%D0%BC%D1%83%D1%8E%20%D0%BF%D0%BE%D0%BA%D1%83%D0%BF%D0%BA%D1%83,%D0%93%D0%BB%D0%B0%D0%B2%D1%83%2014).
* **Тестирование ограничений на HTTP-методы.** Если подозреваете, что приложение или веб-сервер ограничивает доступ на уровне методов (например, POST запрещен, а GET – нет), проведите такую проверку:
* Под привилегированным пользователем найдите несколько действий, которые выполняются POST-запросами (создание пользователя, изменение настроек и т.д.).
* Если эти запросы не защищены CSRF-токенами или другими привязками, попробуйте выполнить их, **сменив метод**: отправьте тот же запрос как GET, HEAD или даже с фальшивым методом (X-HTTP-Method-Override или просто произвольное слово вместо POST).
* Если приложение принимает альтернативные методы и выполняет действие – значит, ограничение по методам на сервере можно обойти[[42]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%BF%D1%80%D0%B8%D0%B2%D0%B8%D0%BB%D0%B5%D0%B3%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%20%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81%D0%BE%D0%B2%2C%20%D0%BA%D0%BE%D1%82%D0%BE%D1%80%D1%8B%D0%B5%20%D0%B2%D1%8B%D0%BF%D0%BE%D0%BB%D0%BD%D1%8F%D1%8E%D1%82%20%D0%BA%D0%BE%D0%BD%D1%84%D0%B8%D0%B4%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5,%D0%9F%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9%20%D0%BD%D0%B5%D0%B4%D0%B5%D0%B9%D1%81%D1%82%D0%B2%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9%20%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%20HTTP)[[43]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=,).

### Рекомендации по обеспечению безопасности

Контроль доступа, с одной стороны, прост для понимания, с другой – требует очень дисциплинированной реализации. Чтобы защитить приложение от этих уязвимостей, придерживайтесь следующих принципов:

* **Не полагайтесь на скрытность.** Предполагается, что злоумышленнику известны все URL, ID и переменные вашего приложения. Не рассчитывайте защититься только тем, что «никто не догадается о таком адресе». Всегда реализуйте проверку прав на сервере.
* **Не доверяйте данным от клиента для принятия решений о доступе.** Любые параметры типа isAdmin, userRole и т.д., поступающие от пользователя (даже в куки или скрытых полях), могут быть изменены. Решения об авторизации должны основываться на достоверной серверной информации (например, сохранённой роли в сессии или БД, а не переданной обратно от клиента).
* **Не делайте предположений о порядке действий пользователя.** Пользователь вовсе не обязан переходить по вашим страницам в задуманной последовательности. Прямые запросы к внутренним шагам, обход ссылок через адресную строку – это нормально, и злоумышленник активно пользуется такими приемами. Поэтому каждая страница/эндпоинт должна *сама* решать, можно ли этому пользователю её просматривать или использовать, независимо от того, откуда он пришёл.
* **Повторно валидируйте чувствительные данные.** Если на шаге 1 вы проверили, что пользователь может редактировать объект X, а на шаге 2 этот же ID X передается через форму, *не доверяйте* тому, что пришло на шаг 2. Повторно проверьте на сервере, что X принадлежит этому же пользователю и он имеет право на операцию. Никогда не рассчитывайте, что раз вы один раз проверили и запомнили, то можно принимать данные из клиента без повторной проверки.
* **“Запрещать по умолчанию”.** При проектировании системы контроля доступа используйте подход **deny by default** – если явно не разрешено, то запрещено. Это касается как кода приложения, так и возможных правил на уровне веб-сервера или брандмауэра. Лучше сперва заблокировать всё, а затем открывать нужное, чем наоборот.
* **Централизуйте и унифицируйте контроль доступа.** Разрозненные проверки по всему коду сложно сопровождать и легко пропустить. Реализуйте единый *центральный компонент*, который на основе сессии пользователя принимает решение, можно ли выполнять запрос. Пропускайте каждый запрос через эту прослойку. Удобный подход – заставить каждого разработчика объявлять на странице требуемый уровень доступа (например, класс контроллера указывает нужную роль). Так ни одна страница не будет забыта.
* **Документируйте права для каждой функции.** На этапе разработки определите, какие роли или пользователи имеют доступ к каждой странице и каждому типу данных. Эта матрица поможет и при реализации, и при тестировании. В ней же заложите принцип наименьших привилегий – у каждого пользователя должны быть только те права, которые необходимы для его задач.
* **Многоуровневая модель привилегий (Defense-in-Depth).** Реализуйте контроль доступа на *нескольких уровнях системы*, чтобы подстраховать приложение. Например, помимо проверок в коде приложения, имеет смысл:
* Настроить *веб-сервер/app-сервер* на ограничение доступа к определённым URL по ролям (например, к /admin/\* только группе админов). Это отсечёт часть нелегитимных запросов ещё до обработки логикой.
* Использовать разные **учетные записи БД** при обращении к базе от имени разных пользователей. Например, запросы от пользователей-членов можно выполнять под аккаунтом БД, которому разрешено *только SELECT*, а админские действия – под аккаунтом с правами *UPDATE/DELETE*. Таким образом, даже если кто-то обойдёт проверки в приложении, запрос на повышение привилегий может не выполниться из-за отсутствия прав у SQL-пользователя[[44]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=,%D0%BF%D1%80%D0%B8%D0%B2%D0%B8%D0%BB%D0%B5%D0%B3%D0%B8%D1%8F%D0%BC%D0%B8%2C%20%D0%BA%D0%BE%D1%82%D0%BE%D1%80%D1%8B%D0%B5%20%D0%B4%D0%B5%D0%B9%D1%81%D1%82%D0%B2%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%20%D1%82%D1%80%D0%B5%D0%B1%D1%83%D1%8E%D1%82%D1%81%D1%8F%20%D0%BA%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D1%83)[[45]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%92%20%D0%B4%D0%BE%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%BA%20%D1%80%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8%20%D1%8D%D1%84%D1%84%D0%B5%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B3%D0%BE,%D1%81%D0%BB%D0%B5%D0%B4%D1%83%D0%B5%D1%82%20%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D1%8C).
* Ограничить привилегии учетных записей ОС, под которыми работают компоненты приложения. Каждый сервис (веб-сервер, сервер приложений, СУБД) должен запускаться от отдельного пользователя ОС с минимальными правами, нужными ему для работы[[46]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=,6%20%D0%BF%D1%80%D0%B8%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%20%D1%87%D0%B0%D1%81%D1%82%D0%B8%D1%87%D0%BD%D1%8B%D0%B9%20%D0%BF%D1%80%D0%B8%D0%BC%D0%B5%D1%80)[[47]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=,%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D1%8B%20%D0%BF%D1%80%D0%B8%D0%B2%D0%B8%D0%BB%D0%B5%D0%B3%D0%B8%D0%B9%20%D0%B4%D0%BB%D1%8F%20%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F). Это предотвратит эскалацию привилегий на уровне сервера, если уязвимость позволит выполнить произвольный код.
* Грамотно использовать **роли (RBAC)** внутри приложения. Вводите такое количество ролей, которое позволит точно разделить доступ, но не делайте слишком много тонких ролей – это усложняет администрирование. Слишком мало ролей – риск, что у пользователей появляются лишние права[[48]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%BA%D0%BE%D1%82%D0%BE%D1%80%D1%8B%D0%BC%20%D0%BC%D0%BE%D0%B3%D1%83%D1%82%20%D0%BE%D0%B1%D1%80%D0%B0%D1%89%D0%B0%D1%82%D1%8C%D1%81%D1%8F%20%D0%BE%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5%20%D1%82%D0%B8%D0%BF%D1%8B,%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%20%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D1%8B%20%D0%B4%D0%BB%D1%8F%20%D0%BE%D0%B3%D1%80%D0%B0%D0%BD%D0%B8%D1%87%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%B0)[[49]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D1%81%D0%B1%D0%B0%D0%BB%D0%B0%D0%BD%D1%81%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D1%82%D1%8C%20%D0%BA%D0%BE%D0%BB%D0%B8%D1%87%D0%B5%D1%81%D1%82%D0%B2%D0%BE%20%D1%80%D0%BE%D0%BB%D0%B5%D0%B9%2C%20%D1%87%D1%82%D0%BE%D0%B1%D1%8B%20%D0%BE%D0%BD%D0%B8,deny). Найдите баланс и используйте роли для первичного отбора доступа (быстрая проверка: если роль не подходит – сразу отказ).
* Если применяются декларативные (вне кода) механизмы контроля доступа, также проектируйте их по принципу «запрет по умолчанию» с явным перечислением разрешений и финальным блокирующим правилом[[50]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%95%D1%81%D0%BB%D0%B8%20%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D1%83%D1%8E%D1%82%D1%81%D1%8F%20%D1%81%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B2%D0%B0%20%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8F%20%D0%BD%D0%B0,%D0%BF%D1%80%D0%B8%D0%BC%D0%B5%D0%BD%D1%8F%D0%B5%D1%82%20%D1%80%D0%B0%D0%B7%D0%BD%D1%8B%D0%B5%20%D1%83%D1%87%D0%B5%D1%82%D0%BD%D1%8B%D0%B5%20%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D0%B8%20%D0%B4%D0%BB%D1%8F)[[51]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D1%81%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D1%8B%20%D1%81%20%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%D0%BC%20%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%20%C2%AB%D0%B7%D0%B0%D0%BF%D1%80%D0%B5%D1%89%D0%B5%D0%BD%D0%BE,%D0%BD%D0%B8%20%D0%BE%D0%B4%D0%BD%D0%BE%D0%BC%D1%83%20%D0%B8%D0%B7%20%D0%BF%D1%80%D0%B5%D0%B4%D1%8B%D0%B4%D1%83%D1%89%D0%B8%D1%85%20%D0%BF%D1%80%D0%B0%D0%B2%D0%B8%D0%BB).
* **Дополнительные меры для критических функций.** Для особо чувствительных операций (финансовые транзакции, изменение важных настроек) внедряйте *дополнительную аутентификацию* или *двойное подтверждение*. Например, подтверждение паролем или кодом 2FA при добавлении нового платежного получателя, или требование, чтобы действие подтвердил второй пользователь с соответствующими правами. Эти меры усложняют жизнь злоумышленнику, даже если он взломал сессию или частично обошел контроль доступа.
* **Логирование и мониторинг.** Каждое обращение к защищенному ресурсу и отказ в доступе должны логироваться. По журналам администраторы безопасности могут заметить подозрительную активность: повторные обращения к запрещенным URL, попытки перебора идентификаторов, доступ к админке с неизвестного IP и т.д. Своевременное обнаружение атаки – половина успеха в ее пресечении.

Следуя этим рекомендациям и тщательно тестируя приложение на этапах разработки и пенетестинга, можно значительно снизить риск уязвимостей контроля доступа.

## Демонстрация

Рассмотрим пару практических примеров атак на контроль доступа из реального мира веб-приложений:

**Кейс 1: Вертикальное повышение привилегий (доступ к скрытой админ-панели).** Пусть в приложении имеется административный раздел по адресу /admin для управления пользователями. Ссылку на него видят только администраторы, а обычным пользователям она не отображается. Однако разработчики *не внедрили проверку прав* при открытии этой страницы – они полагались на то, что «если нет ссылки, никто не найдет». Злоумышленник, обладая обычным аккаунтом, вручную переходит по URL /admin и неожиданно обнаруживает, что страница загружается без авторизации. Он получает полный доступ к админ-интерфейсу и выполняет запрещенные действия – например, создает себе учетную запись администратора или удаляет чужие. **Почему так произошло?** Админ-панель была *полностью незащищена* на уровне сервера. Только UI скрывал ее, что не является защитой. **Как предотвратить?** Всегда проверять роль пользователя на сервере при обращении к админским URL, независимо от того, есть ссылка в интерфейсе или нет.

**Кейс 2: Горизонтальное повышение привилегий (чтение чужих данных).** Представим простое почтовое веб-приложение. У каждого пользователя есть страница вида /mailbox?user\_id=XXX, где XXX – его уникальный ID. Обычно этот параметр берется из сессии и ссылок в профиле, но приложение также принимает user\_id из URL. Пользователь с ID 1001, войдя в свою почту, замечает в адресной строке user\_id=1001 и экспериментирует: меняет значение на 1002. Если проверка доступа не реализована, сервер возвратит содержимое почтового ящика пользователя 1002. Так наш герой читает чужие письма. Это классический пример **небезопасного прямого обращения к объектам (IDOR)**. **Почему это стало возможным?** Приложение доверяет параметру user\_id и не удостоверяется, что запрошенный ящик принадлежит текущему вошедшему пользователю. **Развитие атаки:** Злоумышленник может автоматизировать перебор user\_id и скачать переписку многих пользователей. А если повезет найти ID администратора, то прочитает и его почту, возможно обнаружив там пароли или ссылки для входа – что уже приведет к вертикальному захвату привилегий. **Как предотвратить?** Никогда не полагаться на идентификатор ресурса от клиента. Сервер должен извлечь ID пользователя из сессии и использовать *только его* для выборки писем, либо хотя бы сверять: запрашиваемый user\_id должен совпадать с текущим пользователем (если только это не админ, которому позволено читать чужое).

Оба примера демонстрируют, что ошибки контроля доступа часто очень просты по сути – забыли проверить важный условный оператор или доверились входному параметру. Тем не менее, последствия таких ошибок крайне серьезны. Поэтому тестирование должно быть тщательным: проверяйте каждый чувствительный URL, каждый параметр, каждый шаг процессов с точки зрения "а мог ли простой пользователь сюда прорваться?" и "может ли пользователь А получить данные пользователя Б?".

## Практика с PortSwigger

Для отработки навыков поиска уязвимостей контроля доступа отлично подходят интерактивные лабораторные работы из **PortSwigger Web Security Academy**. Эти лаборатории предоставляют готовое окружение (уязвимое приложение, доступное через браузер), поэтому вам **не нужно настраивать** сервер или устанавливать что-либо локально – вы сразу приступаете к экспериментации. Ниже представлены ключевые задачи по теме контроля доступа (Broken Access Control) с ссылками на соответствующие лаборатории и подсказками:

* [**Lab: Unprotected admin functionality**](https://portswigger.net/web-security/access-control/lab-unprotected-admin-functionality) – *Незащищённая админ-панель*. В этой базе вам дается веб-приложение с административным разделом, куда случайно открыт прямой доступ. **Задача:** найти путь к админ-панели и удалить пользователя carlos. Подсказка: всегда проверяйте файлы вроде robots.txt – там может скрываться указание на закрытые разделы. В решении этой лабораторной выясняется, что robots.txt выдаёт путь /administrator-panel, который и нужно открыть, будучи авторизованным под обычным пользователем, чтобы получить доступ к функциям админа (удалить carlos).
* [**Lab: User role controlled by request parameter**](https://portswigger.net/web-security/access-control/lab-user-role-controlled-by-request-parameter) – *Повышение привилегий через подконтрольный параметр*. В приложении роль пользователя определяется на этапе входа и передается клиенту (через куки). **Задача:** получить доступ к админ-панели и удалить carlos, не зная админских учетных данных. Подсказка: обратившись к /admin, вы увидите отказ. Попробуйте залогиниться под обычным пользователем и перехватить ответ сервера: он выставляет куку Admin=false. Если изменить ее на Admin=true и повторно зайти на /admin, приложение пустит вас как администратора. Этот лаб наглядно демонстрирует, почему нельзя доверять флагам админа, хранящимся у пользователя.
* [**Lab: Referer-based access control**](https://portswigger.net/web-security/access-control/lab-referer-based-access-control) – *Контроль на основе Referer*. Приложение проверяет заголовок Referer для доступа к некоторым админ-действиям. **Задача:** повысить свою роль до администратора (через уязвимость контроля доступа) и удалить пользователя. Подсказка: сначала войдите под админом (учетку дают) и изучите, как происходит, скажем, повышение прав другого пользователя – такой запрос содержит определенный путь (/admin-roles?...) и, возможно, заголовок Referer. Затем, зайдя под обычным пользователем, попробуйте повторить этот запрос: он сначала отвергнется (нет Referer). Решение – использовать инструмент (Burp Repeater) и отправить запрос повышения привилегий **с вашим пользовательским cookie**, но **с тем же Referer**, что был у админа. Таким образом, приложение решит, что запрос исходно шел из админ-панели, и выполнит его. Вы получите права администратора.
* [**Lab: Method-based access control can be circumvented**](https://portswigger.net/web-security/access-control/lab-method-based-access-control-can-be-circumvented) – *Обход контроля доступа по HTTP-методу*. В этом сценарии доступ к админ-функциям ограничен на уровне платформы: например, только POST запросы разрешены админам. **Задача:** став обычным пользователем, сделать себя администратором. Подсказка: анализируя трафик администратора (опять же, демонстрационная учетка administrator:admin позволяет увидеть работу админки), вы узнаете, какой запрос меняет роли. Под обычным пользователем попробуйте выполнить тот же запрос, но изменяя HTTP-метод. Здесь трюк в том, что запрос с некорректным методом (например POSTX) частично проходит в приложение (возникает ошибка "missing parameter"), а перевод его в метод GET позволяет обойти правило платформы. В итоге, отправив запрос на повышение своей роли методом GET, вы добиваетесь успеха. Вывод: если контроль завязан только на метод, всегда проверяйте альтернативные методы (GET, HEAD и прочие) – возможно, приложение их не ожидает, но обработает.
* [**Lab: Multi-step process with no access control on one step**](https://portswigger.net/web-security/access-control/lab-multi-step-process-with-no-access-control-on-one-step) – *Многошаговый процесс с отсутствующей проверкой на одном из шагов*. Приложение содержит админ-панель, где изменение роли пользователя реализовано через последовательность запросов (например, сначала действие, потом подтверждение). **Задача:** стать администратором, эксплуатируя пробел в контроле доступа на этапе подтверждения. Подсказка: изучите, как процесс работает у настоящего администратора (учетные данные даны). Заметьте, какие параметры передаются при подтверждении действия (например, confirmed=true). Затем, под обычным пользователем, попробуйте выполнить *только финальный шаг* – отправив запрос подтверждения с нужными параметрами (и, возможно, сессионным ID обычного пользователя). Если один из шагов не проверяет права, приложение выполнит административное действие (повысит вам роль) без прохождения предыдущих этапов. Этот лаб учит: **каждый этап** должен контролироваться, нельзя "верить" тому, что пользователь прошел правильный путь.
* [**Lab: Insecure direct object references**](https://portswigger.net/web-security/access-control/lab-insecure-direct-object-references) – *Небезопасные прямые ссылки на объекты (горизонтальный доступ)*. Сценарий: на сервере хранятся файлы (например, логи чата пользователей) с предсказуемыми именами. Приложение никак не проверяет, кто запрашивает файл – нужно лишь знать правильный URL. **Задача:** найти пароль пользователя carlos в одном из таких файлов и войти под его учетной записью. Подсказка: начните чат от своего имени, получите ссылку на сгенерированный transcript (например, transcript/5.txt). Видно, что номер 5 – это ID вашего чата. Попробуйте открыть transcript/1.txt, 2.txt и т.д. – скорее всего, наткнетесь на чужие записи. Один из первых транскриптов содержит пароль carlos. Таким образом, вы воспользовались IDOR: известен шаблон имени файла и нет проверки авторства. Решение таких проблем – делать файлы недоступными напрямую или включать контроль (например, требовать скачивания через авторизованный запрос).

Все приведённые лабораторные работы работают прямо в браузере, что очень удобно: вы фокусируетесь на поиске уязвимости и эксплуатации, а не на настройке окружения. Рекомендуется последовательно пройти эти задания. Они охватывают разные аспекты контроля доступа – от тривиальных просчетов (как отсутствие проверок) до нетривиальных атак (подмена метода, обход Referer, эксплойт бизнес-логики). Выполняя их, обращайте внимание на методологию: PortSwigger снабжает каждый лаб детальными объяснениями. После практики вы будете лучше вооружены знаниями для тестирования реальных приложений.

## Тест

Ответьте на следующие вопросы, чтобы проверить усвоение материала урока и практических упражнений. Вопросы разных типов: с одним или несколькими вариантами ответа, утверждения «верно/неверно», а также открытые вопросы, на которые нужно ответить самостоятельно.

1. Какое из ниже перечисленного наиболее точно описывает **вертикальное повышение привилегий**?
2. A. Обычный пользователь получает доступ к функциям администратора.
3. B. Пользователь читает или изменяет данные другого равноправного пользователя.
4. C. Атакующий перехватывает трафик другого пользователя.
5. D. Пользователь проходит аутентификацию под чужими учетными данными.
6. **Верно или неверно?** Уязвимости контроля доступа возникают только после аутентификации (то есть *только* для уже вошедших пользователей).
7. Какое из утверждений про **горизонтальный контроль доступа** верно?
8. A. Определяет, какие разные роли существуют в системе (например, пользователь, модератор, админ).
9. B. Ограничивает пользователя набором объектов, к которым он имеет индивидуальные права.
10. C. Разрешает пользователям одного уровня видеть данные друг друга.
11. D. Выполняет проверку только на уровне базы данных.
12. Укажите **пример горизонтального нарушения** контроля доступа:
13. A. Клиент интернет-банка смог просмотреть баланс счета другого клиента.
14. B. Заблокированный пользователь смог войти в систему.
15. C. Пользователь получил доступ к странице настроек админ-панели.
16. D. Злоумышленник обнаружил резервную копию базы данных через прямой URL.
17. **Верно или неверно?** Если приложение использует GUID (случайные глобальные идентификаторы) вместо последовательных числовых ID, то горизонтальные атаки (типа IDOR) полностью невозможны.
18. Что из перечисленного **не является хорошей практикой** при реализации контроля доступа?
19. A. Проверять права доступа на каждый запрос на стороне сервера.
20. B. Хранить роли пользователей в сессии на сервере, а не доверять данным клиента.
21. C. Скрывать чувствительные URL (например, админские) и не документировать их.
22. D. Следовать принципу «запрещено по умолчанию», явно разрешая только нужное.
23. В чем основная опасность уязвимости типа **IDOR (Insecure Direct Object Reference)**?
24. A. SQL-инъекция при прямом обращении к объекту базы данных.
25. B. Обход аутентификации через уязвимость в токенах сессии.
26. C. Доступ к чужим данным или ресурсам путем подставления чужого идентификатора.
27. D. Выполнение привилегированных команд на сервере через уязвимый API.
28. **Верно или неверно?** Если на странице нет ссылки на админский раздел, обычный пользователь никак не сможет туда попасть.
29. Выберите **лучшее описание** контекстно-зависимого контроля доступа:
30. A. Контроль, основанный на контенте (содержимом) просматриваемых данных.
31. B. Контроль, учитывающий состояние приложения или последовательность действий пользователя.
32. C. Синоним горизонтального контроля доступа.
33. D. Модель, при которой администратор сам определяет, кому дать доступ к своим ресурсам.
34. Какой пример отражает **нарушение бизнес-логики** в контроле доступа?
    * A. Пользователь вызывает API метод, предназначенный только для администраторов, и получает ответ.
    * B. Пользователь пропускает этап оплаты товара и получает товар бесплатно.
    * C. Пользователь авторизуется без ввода пароля из-за уязвимости.
    * D. Пользователь подделывает JWT-токен, повышая себе роль.
35. **Верно или неверно?** Отсутствие ошибки «Доступ запрещен» при попытке открыть неизвестный URL означает, что контроля доступа нет и ресурс точно доступен злоумышленнику.
36. Что из перечисленного чаще всего является **причиной распространенности** уязвимостей контроля доступа?
    * A. Разработчики часто не понимают принципов аутентификации.
    * B. Решения о доступе сложны и принимаются людьми, из-за чего возможны ошибки проектирования.
    * C. Сложность внедрения HTTPS для защиты каналов связи.
    * D. Слабое шифрование паролей в базе данных пользователей.
37. Какой подход разработчика **наиболее опасен** с точки зрения безопасности контроля доступа?
    * A. Использование фреймворка, предоставляющего встроенные механизмы авторизации.
    * B. Применение нескольких уровней проверки (в приложении и на уровне БД).
    * C. Полагаться исключительно на то, что пользователи не будут вручную менять URL или параметры.
    * D. Разделение пользователей на группы ролей с четко определенными правами.
38. **Верно или неверно?** Если функция повышения привилегий пользователя (например, выдача роли модератора) реализована в два шага – сначала запрос, потом подтверждение – то достаточно проверять права на первом шаге, ведь ко второму шагу пользователь не сможет перейти без прохождения первого.
39. Какое из приведенных ниже утверждений **наиболее точно описывает принцип "наименьших привилегий"**?
    * A. Назначать пользователю как можно меньше ролей при регистрации.
    * B. Пользователь должен иметь минимально необходимый набор прав для выполнения своих задач, и не более.
    * C. У всех новых пользователей сначала нет никаких прав, пока администратор явно не выдаст.
    * D. Раз в месяц автоматически понижать привилегии всех пользователей на один уровень для безопасности.
40. Представьте, что вы тестируете веб-приложение с зонами для разных групп пользователей (гости, зарегистрированные, премиум, админы). **Как лучше всего обнаружить страницы, доступные только админам, если у вас нет админской учетной записи?**
    * A. Проанализировать ответы сервера на наличие признаков админских элементов (например, в HTML-коде скрытые ссылки или скрипты, связанные с админом).
    * B. Использовать SQL-инъекцию, чтобы выгрузить структуру сайта.
    * C. Активно сканировать все возможные URL с помощью словаря типичных админских путей.
    * D. Оба варианта A и C помогут в совокупности.
41. **Верно или неверно?** Заголовок HTTP Referer может служить надежным фактором для определения, откуда пользователь пришел, и на основании этого можно строить контроль доступа.
42. Что означает правило контроля доступа, следующее принципу **"deny by default"**?
    * A. Все запросы от незнакомых IP отклоняются.
    * B. Пользователю запрещено всё, что явно не разрешено.
    * C. Пользователь по умолчанию имеет права администратора, пока не будет ограничен.
    * D. Если модуль контроля доступа падает с ошибкой, запросы блокируются.
43. При тестировании интернет-магазина вы заметили, что после оформления заказа URL страницы подтверждения – /confirm?order=12345&step=3. Что вы сделаете для проверки на уязвимость контроля доступа?
    * A. Попробую заменить параметр order=12345 на номер чужого заказа, чтобы посмотреть детали чужого заказа.
    * B. Изменю step=3 на разные значения (1,2 или 4), пытаясь зайти на этапы вне последовательности.
    * C. Удалю параметр step или поставлю некорректное значение, чтобы проверить реакцию системы.
    * D. Все перечисленные действия являются осмысленными проверками.
44. **Верно или неверно?** Если у приложения реализован жесткий контроль доступа в коде, то дополнительные механизмы вроде файрвола, сетевых ACL или ролевых правил веб-сервера уже не нужны (они избыточны).
45. В чем заключается **опасность оставления устаревшей функциональности** (страниц, API) в приложении, даже если на них нет ссылок?
    * A. Они занимают место на сервере и расходуют память.
    * B. Пользователи могут случайно наткнуться на них и запутаться.
    * C. Если про них забыли, на них могли не распространить новые политики контроля доступа, и злоумышленник может их вызвать напрямую.
    * D. Старая функциональность обычно содержит неэффективный код, что может замедлить приложение, но с безопасностью не связано.
46. Вы обнаружили, что в ответах сервера присутствует скрытый HTML-блок:

* <!-- Admin Panel: <a href="/admin123">Go to Admin</a> -->
* Что это означает в контексте безопасности?
  + A. Ничего особенного, это закомментировано и не влияет.
  + B. Возможно, разработчики *спрятали* ссылку на админ-панель, думая, что раз она в комментарии, пользователи не узнают URL.
  + C. Эта строка случайно выдала путь к админской панели, хотя пользователь ее не видит визуально. Злоумышленник, изучив код страницы, теперь знает адрес /admin123.
  + D. Такое невозможно, ни один разработчик не оставит явно admin-ссылку в HTML.

1. **Верно или неверно?** При горизонтальном разделении прав важно убедиться, что даже администратор не может получить доступ к данным, которыми он не должен управлять (например, админ техподдержки не должен видеть финансовые отчеты, если это вне его зоны ответственности).
2. Допустим, система контролирует доступ к API по ролям: менеджер может видеть только свои проекты, администратор – все проекты. Какой **надежный способ** реализовать это на уровне базы данных?
   * A. В SQL-запрос всегда добавлять условие WHERE project.owner\_id = текущий\_пользователь для менеджера, а для админа – не добавлять.
   * B. Использовать представления (VIEW) в базе, разные для менеджера и админа, с соответствующими фильтрами, и подключаться к базе под учетной записью с нужными правами.
   * C. Никак, контроль доступа всегда должен быть исключительно в коде приложения, а в базе – отдавать все.
   * D. Сохранять в каждой таблице поле с ролями, которым доступен каждый конкретный ряд.
3. В ходе теста вы нашли URL /download/Reports/Q1\_financial.pdf, доступный без авторизации. Что из ниже перечисленного **НЕправильно** с точки зрения безопасности?
   * A. Хранение финансовых отчетов в статической директории, открытой в веб-сервере.
   * B. Название файла содержит информацию (Q1\_financial), что может привлечь внимание злоумышленника.
   * C. Отсутствие проверки сессии или прав при прямом обращении к PDF-файлу.
   * D. Применение PDF-формата для хранения отчетов.
4. **Верно или неверно?** Если приложение генерирует для каждого файла уникальное случайное имя, то можно считать, что доступ к статическим файлам защищен (невозможно угадать URL, значит и проверка не нужна).
5. Какой из следующих шагов **наименее эффективен** при проверке горизонтального контроля доступа?
   * A. Получить две учетных записи одного уровня и пробовать под одной открыть данные, принадлежащие другой.
   * B. Перебрать несколько идентификаторов ресурсов (например, заказов) вокруг тех, что вам известны, и посмотреть, не станет ли доступен чужой ресурс.
   * C. Проверить, не позволяет ли API запросить данные всех пользователей сразу (например, /api/getAllUsers).
   * D. Искать уязвимость XSS на страницах входа.
6. Вы проводите аудит безопасности. Что из перечисленного вы охарактеризуете как **критическую уязвимость контроля доступа**?
   * A. В системе управления пользователями админ может сбросить пароли пользователей.
   * B. Любой анонимный посетитель может получить доступ к странице конфиденциальной статистики, минуя логин.
   * C. Менеджер отдела видит документы своего отдела, но не другого – однако через прокси-сервер смог получить доступ и к чужим (сменив IP на IP другого офиса).
   * D. Пользователь без премиум-подписки нашел способ воспользоваться функцией, доступной только премиум (путем изменения параметра подписки в запросе).
7. **Верно или неверно?** Проверку доступа можно выполнять на клиентской стороне (например, скрывать кнопки, поля ввода, ссылки), и этого достаточно, ведь если пользователь не видит запрещенную функцию, он не сможет ею воспользоваться.
8. В лаборатории PortSwigger «Unprotected admin functionality» ключевую роль играет файл:
   * A. .htaccess
   * B. config.yaml
   * C. robots.txt
   * D. admin.log
9. Какой метод HTTP **чаще всего** злоупотребляется злоумышленниками для обхода запрета на использование POST?
   * A. PUT
   * B. HEAD
   * C. DELETE
   * D. CONNECT
10. В контексте многоуровневой модели привилегий, выберите правильное утверждение:
    * A. Если приложение защищено на уровне кода, нет смысла ограничивать права на уровне базы данных.
    * B. Разделение привилегий между компонентами (приложение, БД, ОС) позволяет смягчить последствия взлома одного уровня.
    * C. Глубокоэшелонированная оборона означает, что для доступа к системе нужно пройти несколько этапов аутентификации.
    * D. Многоуровневая модель привилегий – это концепция резервного копирования данных с разными уровнями шифрования.
11. **Верно или неверно?** Реализация контроля доступа через централизованный компонент приложения улучшает безопасность, но создает единую точку отказа (single point of failure), поэтому всегда лучше делать проверки разрозненно в каждом модуле.
12. Что такое **матрица привилегий** в контексте контроля доступа?
    * A. Таблица в базе данных, хранящая пароли пользователей в зашифрованном виде.
    * B. Диаграмма или таблица, отображающая, какие роли (строки) имеют какие права (столбцы) на различных уровнях системы.
    * C. Способ шифрования доступа с помощью матричного преобразования (как в одноимённом фильме «Матрица»).
    * D. Лог-файл, где записаны все случаи предоставления и отзыва привилегий.
13. При проверке контроля доступа вы используете прокси Burp Suite. Какие функции Burp могут **автоматически помочь** обнаружить аномалии в доступе?
    * A. Comparer (сравнение ответов) для выявления различий между разными сессиями.
    * B. Intruder для перебора идентификаторов ресурсов и анализа ответов.
    * C. Scanner, который при активном сканировании убирает заголовок Referer и смотрит на изменения ответов.
    * D. Все вышеперечисленные.
14. **Верно или неверно?** Если сравнение карт сайта (site maps) для админа и обычного пользователя показало, что некоторые ссылки присутствуют только у админа, значит контроль доступа точно реализован правильно для этих функций.
15. Какой из примеров **указывает на отсутствие горизонтального контроля** в приложении?
    * A. Запрос GET /profile?id=501 возвращает статус 200 и данные профиля, даже если пользователь с id=500.
    * B. При попытке удалить чужой файл сервер возвращает 403 Forbidden.
    * C. В коде клиента отсутствуют ссылки на профили других пользователей.
    * D. У пользователя нет возможности изменить свой собственный email.
16. В каком случае **недостаточно** только вертикального разделения доступа, и нужна комбинация с горизонтальным?
    * A. В корпоративном приложении у сотрудников есть роли (менеджер, директор), но каждый видит только документы своего отдела.
    * B. В форуме есть обычные пользователи и модераторы; модераторы могут удалять любые сообщения.
    * C. В фотогалерее все зарегистрированные пользователи равны, но фотографии имеют отметки «только для друзей» или «публичные».
    * D. В онлайн-игре есть уровни доступа к контенту по подписке (бесплатный, платный).
17. **Верно или неверно?** Если веб-сервер настроен блокировать доступ неадминистраторов к пути /admin, то добавление суффикса, например запрос к /admin/ (со слэшем) или /admin?. может обойти такое ограничение в некоторых случаях.
18. В ходе пентеста вы заметили, что при запросе удаленного ресурса API возвращает HTTP 302 Redirect на страницу логина, *но при этом в теле ответа присутствуют данные* запрошенного ресурса. Как лучше интерпретировать эту ситуацию?
    * A. Это нормальное поведение, просто кэш или отладочная информация.
    * B. Это уязвимость: хоть система и пытается перенаправить на логин, она уже включила защищенные данные в ответ (возможно, по недосмотру).
    * C. Это свидетельствует о межсайтовом скриптинге (XSS).
    * D. Сервер таким образом шифрует содержимое, и на логин-странице оно расшифруется.
19. Почему **опасно полагаться только на JavaScript на стороне клиента** для ограничения доступа (например, не отрисовывать админ-кнопки для обычных пользователей)?
    * A. Злоумышленник может вручную вызвать нужный запрос, минуя UI.
    * B. Можно отключить JS в браузере и получить доступ ко всем элементам.
    * C. JavaScript-код легко изменить или подделать в инструментах разработчика.
    * D. Все вышеперечисленное.
20. **Верно или неверно?** Для проверки механизма контроля доступа можно использовать метод черного ящика: вообще не зная деталей кода, просто на основании логики приложения и экспериментов понять, где может быть уязвимость.
21. В лабораторной работе PortSwigger «User role controlled by request parameter» основной уязвимостью было:
    * A. Отсутствие HTTPS, из-за чего злоумышленник украл cookie администратора.
    * B. Хранение роли пользователя в куки, которую можно изменить на Admin=true.
    * C. SQL-инъекция в запросе на вход, позволяющая войти как админ.
    * D. XSS на странице входа, что привело к обходу авторизации.
22. Какой подход поможет **минимизировать человеческие ошибки** при добавлении нового функционала в контексте контроля доступа?
    * A. Обучение разработчиков основам безопасности и контролю доступа.
    * B. Использование конвейера CI/CD, в котором автоматически проверяется наличие уязвимостей доступа.
    * C. Принуждение разработчиков явно указывать требования доступа для каждого нового эндпоинта (например, аннотациями или конфигурацией).
    * D. Привлечение внешних аудиторов для ревью каждого коммита.
23. **Верно или неверно?** Проверка авторизации должна выполняться после всех основных операций в коде страницы (например, сначала обработать запрос, изменить данные, а в конце убедиться, что пользователь имел на это право).
24. Когда уместно использовать **двойную авторизацию (two-man rule)** как часть контроля доступа?
    * A. Для операций, несущих большой риск или ценные изменения, требующих подтверждения второго ответственного лица (например, перевод крупной суммы денег).
    * B. Для всех операций с базой данных, чтобы уменьшить нагрузку на сервер.
    * C. Для авторизации через две разные OAuth-службы одновременно.
    * D. Никогда, это устаревшая концепция.
25. Что из перечисленного **НЕ относится** к методике тестирования контроля доступа?
    * A. Сравнение ответов приложения для пользователей с разными правами.
    * B. Перебор URL-адресов и параметров вне предусмотренных сценариев.
    * C. Декомпиляция и анализ исходного кода или компонентов (клиентских скриптов, мобильного приложения).
    * D. Использование сложных паролей для всех тестовых учетных записей.
26. В чем риски, если **забыть повторно проверить параметр, переданный через клиент** (например, ID ресурса) на завершающем шаге процесса?
    * A. Пользователь может получить ошибку, но это не влияет на безопасность.
    * B. Злоумышленник может изменить этот параметр и получить доступ к объекту, который изначально не должен был быть доступен ему.
    * C. Это приведет к утечке памяти на сервере.
    * D. Ничего страшного, если начальная проверка прошла, значит все хорошо.
27. **Верно или неверно?** HTTP-заголовок X-Original-URL или X-Rewrite-URL может быть использован злоумышленником для обхода проверок, если веб-сервер поддерживает их, но система контроля об этом «не знает».
28. Что из ниже перечисленного является **надежным способом** защитить статические файлы, которые должны быть доступны только авторизованным пользователям?
    * A. Помещать их в недоступную из веба директорию и выдавать через скрипт (динамически) после проверки сессии.
    * B. Использовать скрытые, трудногадаемые имена файлов и не публиковать ссылки.
    * C. Сжимать файлы в ZIP с паролем и давать пароль только авторизованному пользователю.
    * D. Переименовывать файлы каждый день и обновлять ссылки.
29. После проведения тестирования вы хотите убедиться, что покрыли все чувствительные функции. Назовите **два вопроса**, которые важно себе задать при анализе контролей доступа (открытый вопрос).
30. Перечислите **три наиболее распространенных ошибки** разработчиков при реализации контроля доступа (открытый вопрос).
31. Опишите вкратце, как уязвимость контроля доступа в горизонтальном разрезе **может привести** к вертикальному компрометации системы (открытый вопрос).
32. Вспомните лабораторию PortSwigger про Referer. **Какой заголовок** и **с каким значением** необходимо было добавить к запросу, чтобы обойти проверку? (открытый вопрос)
33. Почему принцип **«доверяй, но проверяй»** не подходит для контроля доступа? Что следует использовать вместо него? (открытый вопрос)
34. Как может помочь **логирование доступа** в обнаружении попыток эксплуатации уязвимостей контроля доступа? Приведите пример (открытый вопрос).
35. Представьте, что у приложения есть URL /admin/deleteUser?userId=123. Какие проверки должны быть выполнены на сервере перед удалением пользователя с ID 123? (открытый вопрос)
36. Чем опасна уязвимость, при которой **в HTML-коде интерфейса фигурирует условие** вроде if (isAdmin) { ... показать админ-кнопку ... }? (открытый вопрос)
37. Опишите последовательность действий, как вы бы протестировали функцию **«просмотр чужого профиля»** на уязвимость типа IDOR, не имея прямого доступа к чужому профилю через UI (открытый вопрос).
38. Что означает **RBAC** и какую проблему избыточных или недостаточных привилегий он решает? (открытый вопрос)
39. В чем разница между **дискреционным** и **ролевым** контролем доступа? (открытый вопрос)
40. Зачем может потребоваться **повторная аутентификация** пользователя перед совершением особо важного действия, даже если сессия активна? (открытый вопрос)
41. Приведите пример заголовка, который активный сканер Burp Suite убирает из запросов для выявления уязвимостей контроля доступа, и объясните почему (открытый вопрос).
42. Почему **автоматические сканеры** часто не могут полностью обнаружить уязвимости контроля доступа без участия человека? (открытый вопрос)
43. В лаборатории про multi-step процесс уязвимость заключалась в пропущенной проверке на шаге подтверждения. **Как разработчику избежать подобных ошибок** при создании многоэтапных процессов? (открытый вопрос)
44. Какие действия должен предпринять разработчик, обнаружив в своем приложении уязвимость типа IDOR, чтобы обезопасить данные пользователей? (открытый вопрос)
45. Если вы обнаружили, что через уязвимость контроля доступа можно извлечь список всех пользователей системы, какие дальнейшие шаги атаки вы предпримете? (открытый вопрос)
46. Что такое **ACL (Access Control List)** и где она может применяться в современных веб-приложениях? (открытый вопрос)
47. Почему **методы HTTP** TRACE или нестандартные (например, SEARCH) могут представлять проблему для контроля доступа? (открытый вопрос)
48. Вы провели серию тестов и не нашли очевидных дыр в контроле доступа. Назовите **минимум три меры**, которые вы все равно порекомендуете заказчику внедрить для усиления контроля доступа (открытый вопрос).
49. **Верно или неверно?** Принцип минимальных привилегий должен применяться не только к пользователям приложения, но и к самим компонентам системы (учетным записям сервисов, БД, ОС).
50. **Верно или неверно?** Если URL содержит длинный случайный токен (например, для скачивания файла), то можно считать, что доступ к этому ресурсу защищен не хуже, чем сессией.
51. **Верно или неверно?** Отключение метода HTTP HEAD на веб-сервере гарантированно предотвращает описанные обходы с использованием HEAD-запросов.
52. **Верно или неверно?** Контроль доступа должен охватывать не только HTTP-запросы страниц, но и действия типа AJAX-вызовов, загрузки файлов, WebSocket-сообщения – все входящие запросы без исключения.
53. **Верно или неверно?** Использование современного фреймворка (например, с аннотациями контроллеров для ролей) полностью решает проблему – разработчик не может пропустить проверку, если она задается декларативно.
54. **Верно или неверно?** В AJAX-приложениях (SPA) контроль доступа можно перенести на клиент, ведь сервер обычно выдает только данные, а интерфейс решает, что показать.
55. **Верно или неверно?** Если приложение использует JWT (JSON Web Token) для хранения роли пользователя, важно на стороне сервера **повторно проверять** роль при каждом запросе, а не доверять тому, что внутри токена.
56. **Верно или неверно?** При обнаружении уязвимости контроля доступа тестировщику следует попробовать эксплуатировать ее до конца (например, получить админ-права), прежде чем сообщать о ней, так как без доказательства воздействия баг могут не признать.
57. **Верно или неверно?** Глубокоэшелонированная оборона (defense-in-depth) подразумевает дублирование функций безопасности на разных слоях, поэтому даже компрометация одного слоя не дает полного доступа злоумышленнику.
58. **Верно или неверно?** Контроль доступа *никогда* не должен выполняться на стороне клиента, но *может* дублироваться на клиенте для удобства (например, чтобы сразу не показывать лишнее) – при условии, что сервер все равно проверит.
59. В чем заключается **основное назначение** механизма сессий (cookie, токены) в контексте контроля доступа?
    * A. Обеспечить сохранение состояния входа пользователя, на основании которого потом принимаются решения о доступе.
    * B. Зашифровать все запросы пользователя.
    * C. Хранить на клиенте информацию о всех его правах доступа (внутри токена, например).
    * D. Предотвратить атаки методом перебора паролей.
60. С помощью какого инструмента Burp Suite можно **массово протестировать набор URL** под разными учетными записями?
    * A. Repeater
    * B. Intruder
    * C. Sequencer
    * D. Decoder
61. Какой из заголовков HTTP **не должен** использоваться для принятия решений о предоставлении доступа?
    * A. Referer
    * B. X-User-Role (самостоятельно придуманного)
    * C. Authorization (с токеном Bearer)
    * D. X-Forwarded-For (с IP адресом)
62. Как администраторы системы могут обнаружить, что злоумышленник эксплуатирует уязвимость контроля доступа **горизонтального типа**?
    * A. По ошибкам 500 в логах сервера.
    * B. По множественным последовательным запросам к ресурсам разных пользователей (например, скачивание 1000 файлов подряд).
    * C. По увеличению потребления CPU на сервере.
    * D. Никак, это невозможно обнаружить.
63. Почему **отключение каталогов по умолчанию** (типа /admin/ если не используется, или удаление страниц примеров фреймворка) – важный шаг в конфигурировании сервера?
    * A. Они могут содержать уязвимости XSS.
    * B. Они могут предоставить злоумышленнику функциональность без авторизации (например, стандартные админ-панели, оставшиеся незащищёнными).
    * C. Это нужно для экономии места.
    * D. Чтобы соответствовать требованиям лицензии фреймворка.
64. **Верно или неверно?** Если приложение использует функцию, завязанную на IP-адрес (например, админка доступна только из офиса), это полностью решает проблему контроля доступа для внешних злоумышленников.
65. В чем состоит **основная сложность** при проектировании надежного контроля доступа?
    * A. Необходимо учесть множество сценариев и исключений вручную, технология не может автоматически решить, кому что позволено.
    * B. Требуется писать много кода на низком уровне (на ассемблере) для контроля доступа.
    * C. Нужно использовать только шифрование во всех запросах.
    * D. В частых обновлениях браузеров, которые могут обходить контроль доступа.
66. Какой принцип должен быть выполнен при конфигурации правил доступа на веб-сервере (например, для URL и методов) согласно лучшим практикам?
    * A. Чем более общие правила, тем лучше – одно правило "разрешить всем все методы" достаточно, а остальное сделает приложение.
    * B. Правила должны быть максимально детальными (конкретный метод + конкретный путь + конкретная роль), а в конце – правило, запрещающее все остальное.
    * C. Не смешивать ограничения по методам и по URL – выбирать что-то одно.
    * D. Отдавать контроль доступа полностью в руки приложения, а на сервере ничего не настраивать.
67. **Верно или неверно?** Если в приложении реализована многофакторная аутентификация при входе, то уязвимости контроля доступа уже не будут критичными, так как вероятность проникновения сильно снижена.
68. Перечислите **три способа** обхода контроля доступа, основанного на географическом местоположении пользователя (открытый вопрос).
69. Какие **основные вопросы** должен задать себе тестировщик в начале анализа системы на уязвимости контроля доступа? Назовите не менее трех (открытый вопрос).
70. Что происходит, если **автоматизировать перебор** тысяч возможных ID ресурсов в приложении без достаточных ограничений? Какие риски и последствия? (открытый вопрос)
71. Объясните на примере, как *горизонтальное повышение привилегий* может быть использовано для компрометации *вертикального* контроля доступа (открытый вопрос).
72. Какие индикаторы в логах или мониторинге могут указывать на попытки обхода контроля доступа (приведите 2-3 примера) (открытый вопрос).
73. Почему важно **изолировать среду администрирования** (например, админ-панель на отдельном поддомене или с дополнительной авторизацией)? (открытый вопрос)
74. Ваше веб-приложение хранит файлы пользователей в Amazon S3 (облачное хранилище). Какие меры вы примете, чтобы реализовать контроль доступа к этим файлам? (открытый вопрос)
75. Что следует делать при **дизайн-ревью** новой фичи, чтобы убедиться, что контроль доступа для нее будет адекватным? (открытый вопрос)
76. Если при тестировании обнаружено, что REST API возвращает "isAdmin": true в ответе обычному пользователю (просто не используясь на клиенте), является ли это проблемой? Почему? (открытый вопрос)
77. Опишите стратегию тестирования, когда у вас **нет учетных записей с разными ролями** (например, у вас только пользователь, а роли менеджера/админа недоступны). Как все же проверить, нет ли уязвимостей контроля доступа? (открытый вопрос)
78. Почему **контроль доступа** стоит на первом месте в актуальном списке OWASP Top 10 (2021)? Объясните своими словами, основываясь на материалах урока (открытый вопрос).

[[1]](https://portswigger.net/web-security/access-control/lab-unprotected-admin-functionality#:~:text=) Lab: Unprotected admin functionality | Web Security Academy

<https://portswigger.net/web-security/access-control/lab-unprotected-admin-functionality>

[[2]](https://portswigger.net/web-security/access-control#:~:text=Sometimes%2C%20a%20website%20will%20implement,step%20with%20the%20required%20parameters) [[3]](https://portswigger.net/web-security/access-control#:~:text=administrative%20page%20at%20,then%20the%20request%20is%20allowed) [[10]](https://portswigger.net/web-security/access-control#:~:text=,matching%20discrepancies) [[11]](https://portswigger.net/web-security/access-control#:~:text=enabled%20the%20,option%20is%20enabled%20by%20default) [[12]](https://portswigger.net/web-security/access-control#:~:text=leading%20to%20access%20control%20bypasses) [[13]](https://portswigger.net/web-security/access-control#:~:text=Some%20application%20frameworks%20support%20various,a%20request%20like%20the%20following) [[32]](https://portswigger.net/web-security/access-control#:~:text=This%20might%20not%20be%20directly,based%20on%20the%20user%27s%20role) [[33]](https://portswigger.net/web-security/access-control#:~:text=%60,script) Access control vulnerabilities and privilege escalation | Web Security Academy

<https://portswigger.net/web-security/access-control>

[[4]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%BA%D0%B0%D1%82%D0%B0%D0%BB%D0%BE%D0%B3%20%D0%BA%D0%BD%D0%B8%D0%B3%20%D0%B8%20%D0%BF%D0%BE%D0%BA%D1%83%D0%BF%D0%B0%D1%82%D1%8C%20%D1%8D%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5,%D0%92%20%D0%B4%D0%B0%D0%BD%D0%BD%D0%BE%D0%BC%20%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B5%20%D0%B8%D0%BC%D1%8F%20%D0%B4%D0%BE%D0%BA%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%B0) [[5]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%B0%2C%20%D0%B8%20%D1%87%D1%82%D0%BE%20%D0%BB%D1%8E%D0%B1%D0%BE%D0%B9%2C%20%D0%BA%D1%82%D0%BE,%D1%84%D0%B0%D0%B9%D0%BB%D0%B0%D0%BC%20%D0%B8%20%D0%B4%D1%80%D1%83%D0%B3%D0%B8%D0%BC%20%D0%BA%D0%BE%D0%BD%D1%84%D0%B8%D0%B4%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%BC) [[6]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%9A%D0%B0%D0%BA%20%D0%BE%D0%BF%D0%B8%D1%81%D0%B0%D0%BD%D0%BE%20%D0%B2%20%D0%93%D0%BB%D0%B0%D0%B2%D0%B5%203%2C,%D1%8D%D1%82%D0%BE%D1%82%20%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8C%2C%20%D0%BE%D1%82%D0%BF%D1%80%D0%B0%D0%B2%D0%B8%D0%B2%20%D1%82%D0%BE%D1%82%20%D0%B6%D0%B5) [[7]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%BD%D0%B5%D1%81%D0%B0%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%BC%D1%83%20%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D1%83,%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81%D0%B5%20%D0%B4%D0%BB%D1%8F%20%D0%BD%D0%B5%D1%81%D0%B0%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F%20%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8) [[8]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%A7%D1%82%D0%BE%20%D0%BD%D0%B0%20%D0%BF%D0%B5%D1%80%D0%B2%D1%8B%D0%B9%20%D0%B2%D0%B7%D0%B3%D0%BB%D1%8F%D0%B4%20%D0%B5%D1%89%D0%B5,%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81%D1%8B%20%D1%87%D0%B0%D1%81%D1%82%D0%BE) [[9]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%92%20%D0%BD%D0%B5%D0%BA%D0%BE%D1%82%D0%BE%D1%80%D1%8B%D1%85%20%D1%81%D0%BB%D1%83%D1%87%D0%B0%D1%8F%D1%85%20%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D1%8B%20%D0%BE%D0%B1%D1%80%D0%B0%D0%B1%D0%B0%D1%82%D1%8B%D0%B2%D0%B0%D1%8E%D1%82,%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81%D0%B5%20%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9%20%D0%BD%D0%B5%D0%B4%D0%B5%D0%B9%D1%81%D1%82%D0%B2%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9%20%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%20HTTP) [[14]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%9A%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8C%20%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%B0%20%D0%BD%D0%B0%20%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%B5%20%D0%BF%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%D0%BE%D0%B2,%D0%B0%D0%B4%D1%80%D0%B5%D1%81%D0%B0%20%D1%81%D0%BB%D0%B5%D0%B4%D1%83%D1%8E%D1%89%D0%B5%D0%B3%D0%BE%20%D0%B2%D0%B8%D0%B4%D0%B0) [[15]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=https%3A%2F%2Fwahh,%D0%B2%20%D1%80%D0%B0%D1%81%D0%BA%D1%80%D1%8B%D1%82%D0%B8%D0%B8%20%D1%8D%D1%82%D0%BE%D0%B3%D0%BE%20%D0%BC%D0%B5%D1%85%D0%B0%D0%BD%D0%B8%D0%B7%D0%BC%D0%B0%20%D0%BF%D1%80%D0%B8) [[16]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%9A%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8C%20%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%B0%20%D0%BD%D0%B0%20%D0%BE%D1%81%D0%BD%D0%BE%D0%B2%D0%B5%20Referer,%D0%AD%D1%82%D0%B0%20%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C) [[17]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D1%84%D1%83%D0%BD%D0%B4%D0%B0%D0%BC%D0%B5%D0%BD%D1%82%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%20%D0%BF%D0%BE%D1%80%D0%BE%D1%87%D0%BD%D0%B0%2C%20%D0%BA%D0%BE%D0%BD%D0%B5%D1%87%D0%BD%D0%BE%20%D0%B6%D0%B5%2C%20%D0%BF%D0%BE%D1%82%D0%BE%D0%BC%D1%83,%D0%B1%D1%8B%D1%82%D1%8C%20%D1%83%D1%81%D1%82%D0%B0%D0%BD%D0%BE%D0%B2%D0%BB%D0%B5%D0%BD%20%D0%B2%20%D0%BB%D1%8E%D0%B1%D0%BE%D0%B5%20%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) [[18]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%9C%D0%BD%D0%BE%D0%B3%D0%B8%D0%B5%20%D0%BA%D0%BE%D0%BC%D0%BF%D0%B0%D0%BD%D0%B8%D0%B8%20%D0%B8%D0%BC%D0%B5%D1%8E%D1%82%20%D0%BD%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D1%8B%D0%B5%20%D0%B8%D0%BB%D0%B8,%D0%BF%D1%80%D0%BE%D0%BA%D1%81%D0%B8%2C%20%D1%80%D0%B0%D1%81%D0%BF%D0%BE%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%B2%20%D1%82%D1%80%D0%B5%D0%B1%D1%83%D0%B5%D0%BC%D0%BE%D0%BC%20%D0%BC%D0%B5%D1%81%D1%82%D0%B5) [[19]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%97%D0%BB%D0%BE%D1%83%D0%BC%D1%8B%D1%88%D0%BB%D0%B5%D0%BD%D0%BD%D0%B8%D0%BA%D1%83%20%D0%BE%D1%82%D0%BD%D0%BE%D1%81%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%20%D0%BB%D0%B5%D0%B3%D0%BA%D0%BE%20%D0%BE%D0%B1%D0%BE%D0%B9%D1%82%D0%B8%20%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8C,%D0%9F%D1%80%D1%8F%D0%BC%D0%BE%D0%B5%20%D0%BC%D0%B0%D0%BD%D0%B8%D0%BF%D1%83%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%BA%D0%BB%D0%B8%D0%B5%D0%BD%D1%82%D1%81%D0%BA%D0%B8%D0%BC%D0%B8%20%D0%BC%D0%B5%D1%85%D0%B0%D0%BD%D0%B8%D0%B7%D0%BC%D0%B0%D0%BC%D0%B8%20%D0%B3%D0%B5%D0%BE%D0%BB%D0%BE%D0%BA%D0%B0%D1%86%D0%B8%D0%B8) [[20]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%9F%D0%A0%D0%90%D0%9A%D0%A2%D0%98%D0%A7%D0%95%D0%A1%D0%9A%D0%98%D0%95%20%D0%A8%D0%90%D0%93%D0%98%20%D0%92%D0%BE%D1%82%20%D0%BD%D0%B5%D1%81%D0%BA%D0%BE%D0%BB%D1%8C%D0%BA%D0%BE%20%D0%B2%D0%BE%D0%BF%D1%80%D0%BE%D1%81%D0%BE%D0%B2%2C,%D0%B4%D0%BB%D1%8F%20%D0%B5%D0%B3%D0%BE%20%D0%BD%D0%B0%D1%81%D1%82%D1%80%D0%BE%D0%B9%D0%BA%D0%B8%20%D0%B8%20%D0%BC%D0%BE%D0%BD%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D0%BD%D0%B3%D0%B0) [[21]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%BE%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%BD%D0%BE%D0%BC%D1%83%20%D0%BF%D0%BE%D0%B4%D0%BC%D0%BD%D0%BE%D0%B6%D0%B5%D1%81%D1%82%D0%B2%D1%83%20%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%2C%20%D0%BA%D0%BE%D1%82%D0%BE%D1%80%D0%BE%D0%B5%20%D0%BF%D1%80%D0%B8%D0%BD%D0%B0%D0%B4%D0%BB%D0%B5%D0%B6%D0%B8%D1%82,%D1%82%D0%BE%D0%BC%2C%20%D1%87%D1%82%D0%BE%20%D0%BF%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%20%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D1%83%D0%B5%D1%82%D1%81%D1%8F%20%D0%B4%D0%BB%D1%8F) [[22]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D1%87%D0%B5%D0%BB%D0%BE%D0%B2%D0%B5%D1%87%D0%B5%D1%81%D0%BA%D0%BE%D0%B3%D0%BE%20%D0%B8%D0%BD%D1%82%D0%B5%D0%BB%D0%BB%D0%B5%D0%BA%D1%82%D0%B0%20%D0%B4%D0%BB%D1%8F%20%D1%8D%D1%84%D1%84%D0%B5%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%B2%D1%8B%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D1%8F,%D0%BA%D0%B0%D0%B6%D0%B4%D0%BE%D0%BC%D1%83%20%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8E%2C%20%D1%81%D0%BE%D0%B2%D0%BF%D0%B0%D0%B4%D0%B0%D0%B5%D1%82%20%D0%B8%D0%BB%D0%B8%20%D1%80%D0%B0%D0%B7%D0%BB%D0%B8%D1%87%D0%B0%D0%B5%D1%82%D1%81%D1%8F) [[23]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=2,%D1%84%D0%B0%D0%B9%D0%BB%20%D1%81%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D1%8F%2C%20%D0%BF%D0%BE%D0%BB%D1%83%D1%87%D0%B5%D0%BD%D0%BD%D1%8B%D0%B9%20%D0%BF%D0%BE%D1%81%D0%BB%D0%B5%20%D0%BA%D0%B0%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F) [[24]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%B8%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81%D0%B5%20%D0%BD%D0%B5%D1%82%20%D0%BD%D0%B0%20%D0%BD%D0%B5%D0%B5%20%D1%81%D1%81%D1%8B%D0%BB%D0%BE%D0%BA,%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D0%B8%D1%8E%20%D0%BF%D1%80%D0%BE%D1%84%D0%B8%D0%BB%D1%8F%20%D0%B4%D0%BB%D1%8F%20%D1%82%D0%B5%D0%BA%D1%83%D1%89%D0%B5%D0%B3%D0%BE%20%D0%B2%D0%BE%D1%88%D0%B5%D0%B4%D1%88%D0%B5%D0%B3%D0%BE) [[25]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%BD%D0%B5%D0%B4%D0%BE%D1%81%D1%82%D0%B0%D1%82%D0%BE%D1%87%D0%BD%D0%BE%20%D0%B4%D0%BB%D1%8F%20%D0%BE%D1%86%D0%B5%D0%BD%D0%BA%D0%B8%20%D1%8D%D1%84%D1%84%D0%B5%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D0%B8%20%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8F,%D0%98%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D1%83%D1%8F) [[26]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%BF%D1%80%D0%B8%D1%87%D0%B8%D0%BD%D0%B0%D0%BC%2C%20%D0%BD%D0%B5%20%D1%81%D0%B2%D1%8F%D0%B7%D0%B0%D0%BD%D0%BD%D1%8B%D0%BC%20%D1%81%20%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5%D0%BC,%D0%9A%D0%B0%D0%B6%D0%B4%D1%8B%D0%B9%20%D1%88%D0%B0%D0%B3%20%D1%8D%D1%82%D0%BE%D0%B3%D0%BE%20%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%B0) [[27]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%BD%D0%B5%D0%BE%D0%B1%D1%85%D0%BE%D0%B4%D0%B8%D0%BC%D0%BE%20%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D1%82%D1%8C%20%D0%B8%D0%BD%D0%B4%D0%B8%D0%B2%D0%B8%D0%B4%D1%83%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%2C%20%D1%87%D1%82%D0%BE%D0%B1%D1%8B%20%D1%83%D0%B1%D0%B5%D0%B4%D0%B8%D1%82%D1%8C%D1%81%D1%8F%2C,net%2Fauth%2F471) [[28]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%B2%D1%81%D0%B5%D1%81%D1%82%D0%BE%D1%80%D0%BE%D0%BD%D0%BD%D0%B5%D0%B3%D0%BE%20%D1%82%D0%B5%D1%81%D1%82%D0%B0%20%D0%B2%20%D0%BB%D1%8E%D0%B1%D0%BE%D0%BC%20%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B5,%D0%BD%D0%B8%D0%B7%D0%BA%D0%BE%D0%BF%D1%80%D0%B8%D0%B2%D0%B8%D0%BB%D0%B5%D0%B3%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8F%20%D1%87%D0%B0%D1%81%D1%82%D0%BE%20%D0%B1%D1%8B%D0%B2%D0%B0%D0%B5%D1%82%20%D0%B4%D0%BE%D1%81%D1%82%D0%B0%D1%82%D0%BE%D1%87%D0%BD%D0%BE) [[29]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D1%81%D1%83%D1%89%D0%B5%D1%81%D1%82%D0%B2%D0%BE%D0%B2%D0%B0%D1%82%D1%8C%20%D0%BF%D0%BB%D0%BE%D1%85%D0%BE%20%D0%B7%D0%B0%D1%89%D0%B8%D1%89%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F%20%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D1%8C%2C%20%D0%BD%D0%B0,%D0%BF%D0%BE%D0%BB%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%BF%D1%80%D1%8F%D0%BC%D0%BE%D0%B3%D0%BE%20%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%B0%20%D0%BA%20%D0%BA%D0%BE%D0%BD%D1%84%D0%B8%D0%B4%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B9) [[30]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=3,%D1%81%D0%BA%D1%80%D0%B8%D0%BF%D1%82%D1%8B%20%D0%B2%20%D0%BF%D0%BE%D0%B8%D1%81%D0%BA%D0%B5%20%D1%81%D1%81%D1%8B%D0%BB%D0%BE%D0%BA%20%D0%BD%D0%B0) [[31]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%B4%D0%BB%D1%8F%20%D0%BF%D1%80%D0%B8%D0%BD%D1%8F%D1%82%D0%B8%D1%8F%20%D1%80%D0%B5%D1%88%D0%B5%D0%BD%D0%B8%D0%B9%20%D0%BE%20%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D0%B5,%D0%B7%D0%BD%D0%B0%D1%87%D0%B8%D0%BC%D0%BE%D0%BC%D1%83%20%D0%B8%D0%B7%D0%BC%D0%B5%D0%BD%D0%B5%D0%BD%D0%B8%D1%8E%20%D0%B2%20%D0%BE%D1%82%D0%B2%D0%B5%D1%82%D0%B5%20%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F) [[34]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%9F%D0%A0%D0%90%D0%9A%D0%A2%D0%98%D0%A7%D0%95%D0%A1%D0%9A%D0%98%D0%95%20%D0%A8%D0%90%D0%93%D0%98%201,%D0%B2%20%D0%B8%D0%B4%D0%B5%D0%BD%D1%82%D0%B8%D1%84%D0%B8%D0%BA%D0%B0%D1%82%D0%BE%D1%80%D0%B0%D1%85%2C%20%D0%BA%D0%BE%D1%82%D0%BE%D1%80%D1%8B%D0%B5%20%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%B5%D1%82%20%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D0%B5) [[35]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=3,%D0%BA%D0%B0%D1%81%D1%82%D0%BE%D0%BC%D0%BD%D0%BE%D0%B9%20%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D0%B8%D0%B7%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B9%20%D0%B0%D1%82%D0%B0%D0%BA%D0%B8%20%D1%81%20%D1%86%D0%B5%D0%BB%D1%8C%D1%8E) [[36]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=http%3A%2F%2Fmdsec,%D1%87%D1%82%D0%BE%20%D1%81%D0%B0%D0%BC%D1%8B%D0%B5%20%D0%BD%D0%B8%D0%B7%D0%BA%D0%B8%D0%B5%20%D0%BD%D0%BE%D0%BC%D0%B5%D1%80%D0%B0%20%D0%BF%D1%80%D0%B8%D1%81%D0%B2%D0%BE%D0%B5%D0%BD%D1%8B) [[37]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%BD%D0%B5%20%D0%BE%D0%B1%D1%80%D0%B0%D0%B4%D1%83%D0%B5%D1%82,%D0%92%D0%B5%D1%80%D0%BE%D1%8F%D1%82%D0%BD%D0%BE%2C%20%D0%B0%D0%B4%D0%BC%D0%B8%D0%BD%D0%B8%D1%81%D1%82%D1%80%D0%B0%D1%82%D0%BE%D1%80%D1%8B%20%D0%BC%D0%BE%D0%B3%D1%83%D1%82) [[38]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%A2%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%20%D0%BF%D1%80%D1%8F%D0%BC%D0%BE%D0%B3%D0%BE%20%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%B0%20%D0%BA%20%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%B0%D0%BC,Encoding%3A%20gzip%2C%20deflate) [[39]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=servlet%3Dcom,%D1%81%D0%BC%D0%BE%D0%B6%D0%B5%D1%82%D0%B5%20%D0%BF%D0%BE%D0%BB%D1%83%D1%87%D0%B8%D1%82%D1%8C%20%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%20%D0%B8%20%D0%BA) [[40]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%9F%D0%A0%D0%90%D0%9A%D0%A2%D0%98%D0%A7%D0%95%D0%A1%D0%9A%D0%98%D0%95%20%D0%A8%D0%90%D0%93%D0%98%201,%D0%95%D1%81%D0%BB%D0%B8%20%D0%B2%D0%BE%D0%B7%D0%BC%D0%BE%D0%B6%D0%BD%D0%BE%2C%20%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%B9%D1%82%D0%B5) [[41]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D1%82%D1%80%D0%B5%D0%B1%D1%83%D0%B5%D0%BC%D1%83%D1%8E%20%D0%BF%D0%BE%D0%BA%D1%83%D0%BF%D0%BA%D1%83,%D0%93%D0%BB%D0%B0%D0%B2%D1%83%2014) [[42]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%BF%D1%80%D0%B8%D0%B2%D0%B8%D0%BB%D0%B5%D0%B3%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D1%8B%D1%85%20%D0%B7%D0%B0%D0%BF%D1%80%D0%BE%D1%81%D0%BE%D0%B2%2C%20%D0%BA%D0%BE%D1%82%D0%BE%D1%80%D1%8B%D0%B5%20%D0%B2%D1%8B%D0%BF%D0%BE%D0%BB%D0%BD%D1%8F%D1%8E%D1%82%20%D0%BA%D0%BE%D0%BD%D1%84%D0%B8%D0%B4%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5,%D0%9F%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9%20%D0%BD%D0%B5%D0%B4%D0%B5%D0%B9%D1%81%D1%82%D0%B2%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9%20%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%20HTTP) [[43]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=,) [[44]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=,%D0%BF%D1%80%D0%B8%D0%B2%D0%B8%D0%BB%D0%B5%D0%B3%D0%B8%D1%8F%D0%BC%D0%B8%2C%20%D0%BA%D0%BE%D1%82%D0%BE%D1%80%D1%8B%D0%B5%20%D0%B4%D0%B5%D0%B9%D1%81%D1%82%D0%B2%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%20%D1%82%D1%80%D0%B5%D0%B1%D1%83%D1%8E%D1%82%D1%81%D1%8F%20%D0%BA%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D1%83) [[45]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%92%20%D0%B4%D0%BE%D0%BF%D0%BE%D0%BB%D0%BD%D0%B5%D0%BD%D0%B8%D0%B5%20%D0%BA%20%D1%80%D0%B5%D0%B0%D0%BB%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D0%B8%20%D1%8D%D1%84%D1%84%D0%B5%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B3%D0%BE,%D1%81%D0%BB%D0%B5%D0%B4%D1%83%D0%B5%D1%82%20%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D1%82%D1%8C) [[46]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=,6%20%D0%BF%D1%80%D0%B8%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%20%D1%87%D0%B0%D1%81%D1%82%D0%B8%D1%87%D0%BD%D1%8B%D0%B9%20%D0%BF%D1%80%D0%B8%D0%BC%D0%B5%D1%80) [[47]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=,%D0%BC%D0%B0%D1%82%D1%80%D0%B8%D1%86%D1%8B%20%D0%BF%D1%80%D0%B8%D0%B2%D0%B8%D0%BB%D0%B5%D0%B3%D0%B8%D0%B9%20%D0%B4%D0%BB%D1%8F%20%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%BF%D1%80%D0%B8%D0%BB%D0%BE%D0%B6%D0%B5%D0%BD%D0%B8%D1%8F) [[48]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%BA%D0%BE%D1%82%D0%BE%D1%80%D1%8B%D0%BC%20%D0%BC%D0%BE%D0%B3%D1%83%D1%82%20%D0%BE%D0%B1%D1%80%D0%B0%D1%89%D0%B0%D1%82%D1%8C%D1%81%D1%8F%20%D0%BE%D0%BF%D1%80%D0%B5%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5%20%D1%82%D0%B8%D0%BF%D1%8B,%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%20%D0%BF%D0%BB%D0%B0%D1%82%D1%84%D0%BE%D1%80%D0%BC%D1%8B%20%D0%B4%D0%BB%D1%8F%20%D0%BE%D0%B3%D1%80%D0%B0%D0%BD%D0%B8%D1%87%D0%B5%D0%BD%D0%B8%D1%8F%20%D0%B4%D0%BE%D1%81%D1%82%D1%83%D0%BF%D0%B0) [[49]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D1%81%D0%B1%D0%B0%D0%BB%D0%B0%D0%BD%D1%81%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D1%82%D1%8C%20%D0%BA%D0%BE%D0%BB%D0%B8%D1%87%D0%B5%D1%81%D1%82%D0%B2%D0%BE%20%D1%80%D0%BE%D0%BB%D0%B5%D0%B9%2C%20%D1%87%D1%82%D0%BE%D0%B1%D1%8B%20%D0%BE%D0%BD%D0%B8,deny) [[50]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D0%95%D1%81%D0%BB%D0%B8%20%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D1%83%D1%8E%D1%82%D1%81%D1%8F%20%D1%81%D1%80%D0%B5%D0%B4%D1%81%D1%82%D0%B2%D0%B0%20%D0%BA%D0%BE%D0%BD%D1%82%D1%80%D0%BE%D0%BB%D1%8F%20%D0%BD%D0%B0,%D0%BF%D1%80%D0%B8%D0%BC%D0%B5%D0%BD%D1%8F%D0%B5%D1%82%20%D1%80%D0%B0%D0%B7%D0%BD%D1%8B%D0%B5%20%D1%83%D1%87%D0%B5%D1%82%D0%BD%D1%8B%D0%B5%20%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D0%B8%20%D0%B4%D0%BB%D1%8F) [[51]](file://file-HdZZXpPEkbGxwwDDd6Mw1w#:~:text=%D1%81%D0%BF%D1%80%D0%BE%D0%B5%D0%BA%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D1%8B%20%D1%81%20%D0%B8%D1%81%D0%BF%D0%BE%D0%BB%D1%8C%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5%D0%BC%20%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%20%C2%AB%D0%B7%D0%B0%D0%BF%D1%80%D0%B5%D1%89%D0%B5%D0%BD%D0%BE,%D0%BD%D0%B8%20%D0%BE%D0%B4%D0%BD%D0%BE%D0%BC%D1%83%20%D0%B8%D0%B7%20%D0%BF%D1%80%D0%B5%D0%B4%D1%8B%D0%B4%D1%83%D1%89%D0%B8%D1%85%20%D0%BF%D1%80%D0%B0%D0%B2%D0%B8%D0%BB) WAHH - Глава 8.txt

<file://file-HdZZXpPEkbGxwwDDd6Mw1w>