Лабораторная работа 21

ПСКП

ПОИТ-3

**Задание 01**

1. Разработайте приложение **21-01**, представляющее собой EXPRESS-HTTP-сервер, прослушивающий порт ***3000***.
2. Приложение **21-01** предназначено для предоставления аутентифицированного http-доступа к ресурсу.
3. Приложение **21-01** должно обеспечивать BASIC-аутентификацию с помощью пакета **passport**.
4. Приложение **21-01** должно реагировать на запросы к следующим ресурсам.

|  |  |
| --- | --- |
| GET /login | Ввод имени и пароля для аутентифицированного доступа к ресурсу |
| GET /logout | Отключить аутентифицированный доступ к ресурсу |
| GET /resource | Ресурс отправляет сообщение RESOURCE  При попытке неаутентифицированного доступа выполняет переадресацию на GET /login |
| Остальные URI | Сообщение со статусом 404 |

1. Список разрешенных данных для аутентификации (имя, пароль) в приложении должен быть представлен в виде json-объекта, доступ к которому осуществляется с помощью функции **require**.
2. Тестирование выполнить с помощью браузера.

**Задание 02**

1. Разработайте приложение **21-02**, представляющее собой EXPRESS-HTTP-сервер, прослушивающий порт ***3000***. Приложение **21-02** должно быть функционально подобно приложению **21-01**, но при этом использовать DIGEST- аутентификацию (пакет **passport**).

**Задание 03** Ответьте на следующие вопросы

1. Перечислите все известные виды аутентификации. Какие из них описаны в RFC?
2. Опишите схему BASIC-аутентификации.
3. Опишите схему DIGEST-аутентификации.
4. Почему в чистом виде http-аутентификация не является надежной?
5. Перечислите три вида http-аутентификации, какие из них описаны в RFC?

\* BASIC да

\* DIGEST да

\* FORMS нет

1. Опишите схему BASIC-аутентификации.

username и password передаются в заголовке Authorization в незашифр. виде (base65-encoded)

1. Опишите схему DIGEST-аутентификации.

Сервер посылается уник. значение nonce, а браузер передает хэш пароля, вычисленный с исп-ем указ. nonce

1. Опишите схему FORMS-аутентификации.

в веб прил включ HTML-фомрма, в кот. юзер должен ввести username/password и отпр их на сервер через *HTTP POST* для аутент

успех – веб-прил создает *session token*, кот помещ. в cookies

при послед. запросах *token* авто- перед на сервер и позв. прил-ю получ инфу о тек. юзере для авторизации запроса

1. Почему в чистом виде http-аутентификация не является надежной?

легко данные перезватить и модиф

# HTTP Authentication

**Идентификация** – заявление пользователя о себе.

**Аутентификация** – процедура проверки подлинности идентификации юзера.

**Авторизация** – процедура проверки прав аутентифицированного пользователя.

**Аутентификация –** процедура, направленная на подтверждение идентификатора пользователя (логин и пароль, как правило). На основании какого-то секрета.

**Авторизация** – после аутентификации; проверка есть ли права на выполнение тех или иных действий.

Исп-ся **1 код возврата** (401 (и 200ый само собой))

401 код – Unauthorized – ваш запрос является не авторизированным.   
407 код – для прокси серверов.

**2 заголовка**, кот.мы будем использовать:

\*authorization

\*www-autherticate

Все открыто, злоумышленники могут работать вместо клиента, надо шифровать.

Схема аутентификации (клиент и сервер)

**Способы аутентификации:**

1. BASIC

Схема

Ключевое слово **basic** от сервера к клиенту – указывает на вид аутентификации

Realm – То, что знает клиент (твоя аутентификация действует в рамках этого сектора), т.е. определяет область действия аутентификации.

Если на стороне клиента – браузер, то 401 код заставляет браузер зажечь окошечко, которое попросит вас ввести логин и пароль.

Base64 строка – сцепленные через «:» имя и пароль в заголовке authorization.

С получает, проверяет валидность, если все хорошо – отвечает 200ым кодом, если нет – сначала все идет, т.к.выдается код 401.

7617 протокол

1. Дайджест - DIGEST

Схема

7616 протокол

Описание схемы:

Идет запрос

Дальше идет ответ 401, в котором указываем заголовки WW-Authenticate и указывает DIGEST. Особенность в том, что указывается не имя и пароль, а хэш имени и пароля.

Как захешировать имя и пароль?

\*добавить домен и т.д.

У С есть БД, где существует хеш и остается проверить валидный хеш или нет.

Является большей защитой, чем basic.

С т.зр. криптографии – также беззащитен как basic.

Исп-ся реже, т.к. не защищает (ур-нь защиты почти как у basic), но больше мороки => исп-ют в основном basic.

1. FORMS

Нет стандарт (нигде не описан = народный способ)

Схема

Описание схемы:

Инфа о пользователе и пароле передается в куки

Процедура аутентификации:

Идет запрос

С проверет есть ли куки, если нет – запрос не аутентифицирован

Если куки есть – проверяет содержимое, есть инфа, кот.интересует и она валидна – отвечаем ресурсом; иначе – отправляем страницу, в кот.просим ввести имя и пароль.

Юзер вводит и жмет сабмит, эта инфа отправляется на С. С обрабатывает запрос и формирует на основании имени и пароля токен (бит.посл-сть, кот.С м.проверить на валидность; имеет время жизни). Отправляется на К SetCookie И указывает токен, который должен отправляться клиентом в каждом запросе.

Используется чаще всего, т.к.простой и понятный.

Все 3 способа – слабые. Если не исп-ть HTTPS – данные легко перехватить и модифицировать.

**TLS – аутентификация**

Аутентификация, которая осуществляется на основе сертификата x.509

**x.509** – стандартный формат хранения и транспортироваки отрибутов безс-ти.

- стандарт, кот.разработан международным институтом телекоммун., кот.лежит в основе HTTPS и TLS-аутентификации.

- эл.док, кот. выдается Центром Сертификации.

Содержит: имя держателя, адрес, серийный номер сертификата, даты проверки, открытый ключ держателя.

Картинка

Вместе с сертификатом выдается секретный и публичный ключ.

Публичный находится на самом сертификате.

Секретный ключ выдается отдельно.

.