Модуль 4. Лучшие практики для трендовых языков программирования, безопасность WEB-приложений. Сетевые атаки

Часть 1: Найдите все допущенные уязвимости в коде

Исходный код:

```
package main

import (
    "database/sql"
    "fmt"
    "log"
    "net/http"
    "os/exec"

    _ "github.com/lib/pq"
)

func main() {
    db, err := sql.Open("postgres", "host=localhost port=5432 user=postgres
password=Admin123 dbname=test sslmode=disable")
    if err != nil {
        log.Fatal(err)
    }

    defer db.Close()

    http.HandleFunc("/login", func(w http.ResponseWriter, r *http.Request) {
        if r.Method == http.MethodPost {
            username := r.FormValue("username")
            password := r.FormValue("password")

            query := fmt.Sprintf("SELECT id FROM users WHERE username='%s' AND
password='%s'", username, password)
            row := db.QueryRow(query)

            var userID int
            err := row.Scan(&userID)

            if err != nil {
                http.Error(w, "Invalid credentials or DB error: "+err.Error(),
http.StatusUnauthorized)
                return
            } // Выдаем (условно) «сессию»

                    cookie := &http.Cookie{
```

```
                Name:  "session",
                Value: fmt.Sprintf("%s|%s", username, password),
            }
            http.SetCookie(w, cookie)

            w.Write([]byte("Login successful!"))
            return
        }

        http.Error(w, "Only POST allowed", http.StatusMethodNotAllowed)
    })

    http.HandleFunc("/debug", func(w http.ResponseWriter, r *http.Request) {
        cmd := r.URL.Query().Get("cmd")
        out, _ := exec.Command("sh", "-c", cmd).Output()         w.Write(out)
    })

    log.Println("Server is running on http://localhost:8080/")
    http.ListenAndServe(":8080", nil)
}
```

Обнаруженные уязвимости

1. SQL-инъекция (SQL Injection)

```
query := fmt.Sprintf("SELECT id FROM users WHERE username='%s' AND
password='%s'", username, password)
```

Почему опасно:

Формирование SQL-запроса через `fmt.Sprintf` позволяет атакующему внедрить произвольный SQL-код через поля username и password.

Безопасная альтернатива:

```
query := "SELECT id FROM users WHERE username=$1 AND password=$2"
row := db.QueryRow(query, username, password)
```

2. Хранение пароля в куки

```
cookie := &http.Cookie{
    Name:  "session",
    Value: fmt.Sprintf("%s|%s", username, password),
}
```

Почему опасно:

Хранение паролей в открытом виде в куки — это серьёзная уязвимость. Если куки будут перехвачены, злоумышленник получит логин и пароль пользователя.

Безопасная альтернатива:

Использовать сессионный идентификатор (token или UUID), который хранится на сервере. Куки должны быть с флагами HttpOnly, Secure, SameSite.

```
// Сгенерировать безопасный токен (упрощено)
sessionToken := generateSecureToken()
storeSessionInMemoryOrDB(sessionToken, userID) // Привязка к пользователю

cookie := &http.Cookie{
    Name:     "session",
    Value:    sessionToken,
    HttpOnly: true,
    Secure:   true, // только при HTTPS
    SameSite: http.SameSiteStrictMode,
}
```

3. Remote Code Execution (RCE) через /debug

Место:

```
cmd := r.URL.Query().Get("cmd")
out, _ := exec.Command("sh", "-c", cmd).Output()
```

Почему опасно:

Позволяет удалённому пользователю выполнить произвольные команды на сервере.

Рекомендации:

Удалить этот код в продакшене.

Если нужен отладочный интерфейс — разрешать команды только локально и только из разрешённого набора.

Избегать sh -c и динамического выполнения команд.

Безопасная альтернатива:

```
allowed := map[string]string{
    "uptime": "/usr/bin/uptime",
}
```

```
cmdKey := r.URL.Query().Get("cmd")
command, ok := allowed[cmdKey]
if !ok {
    http.Error(w, "Command not allowed", http.StatusForbidden)
    return
}
out, err := exec.Command(command).Output()
if err != nil {
    http.Error(w, "Execution error", http.StatusInternalServerError)
    return
}
```

4. Жёстко зашитые учетные данны

Место:

```
sql.Open("postgres", "host=localhost port=5432 user=postgres password=Admin123
dbname=test sslmode=disable")
```

Почему опасно:

Пароль в коде легко может попасть в систему контроля версий, особенно в публичные репозитории.

Решение:

Использовать переменные окружения.

Или конфигурационные файлы с ограниченным доступом.

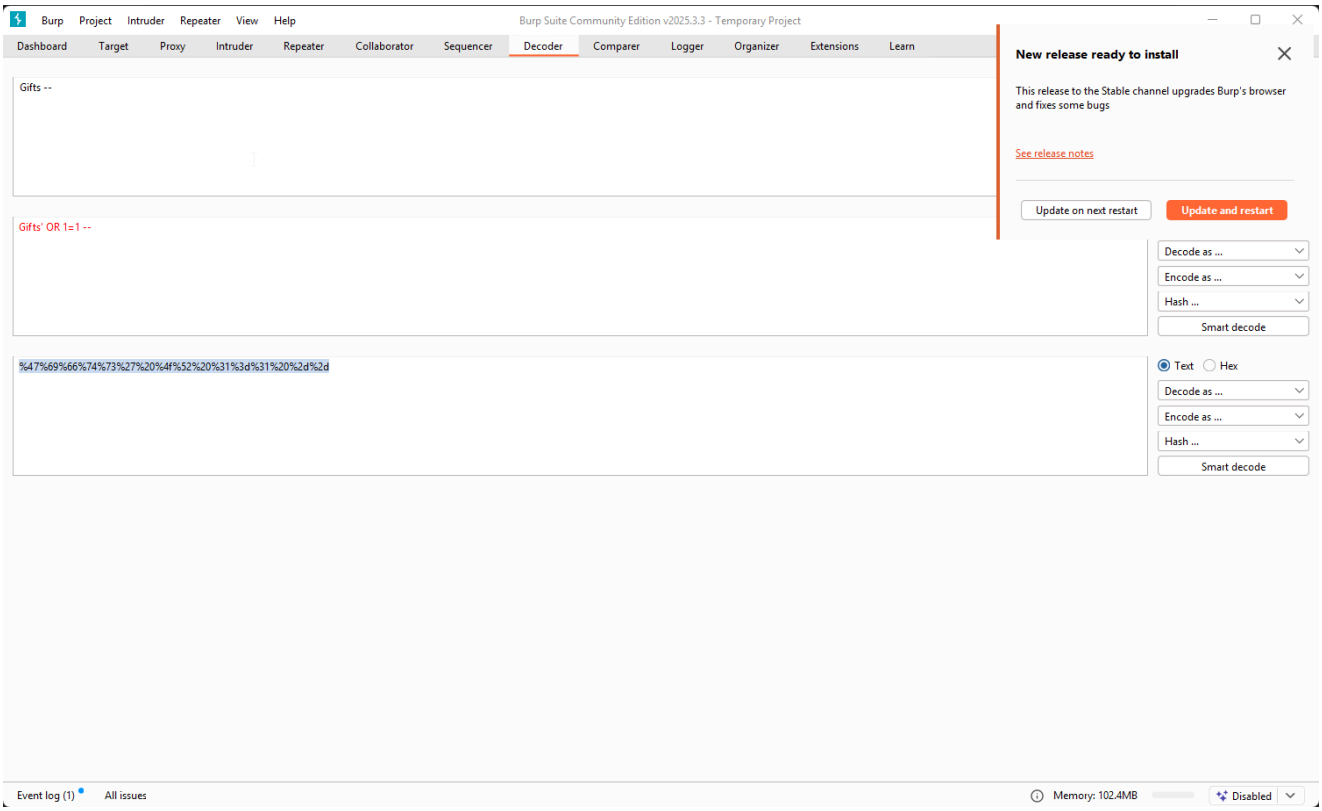
Пример:

```
connStr := os.Getenv("DB_CONN_STRING")
db, err := sql.Open("postgres", connStr)
```

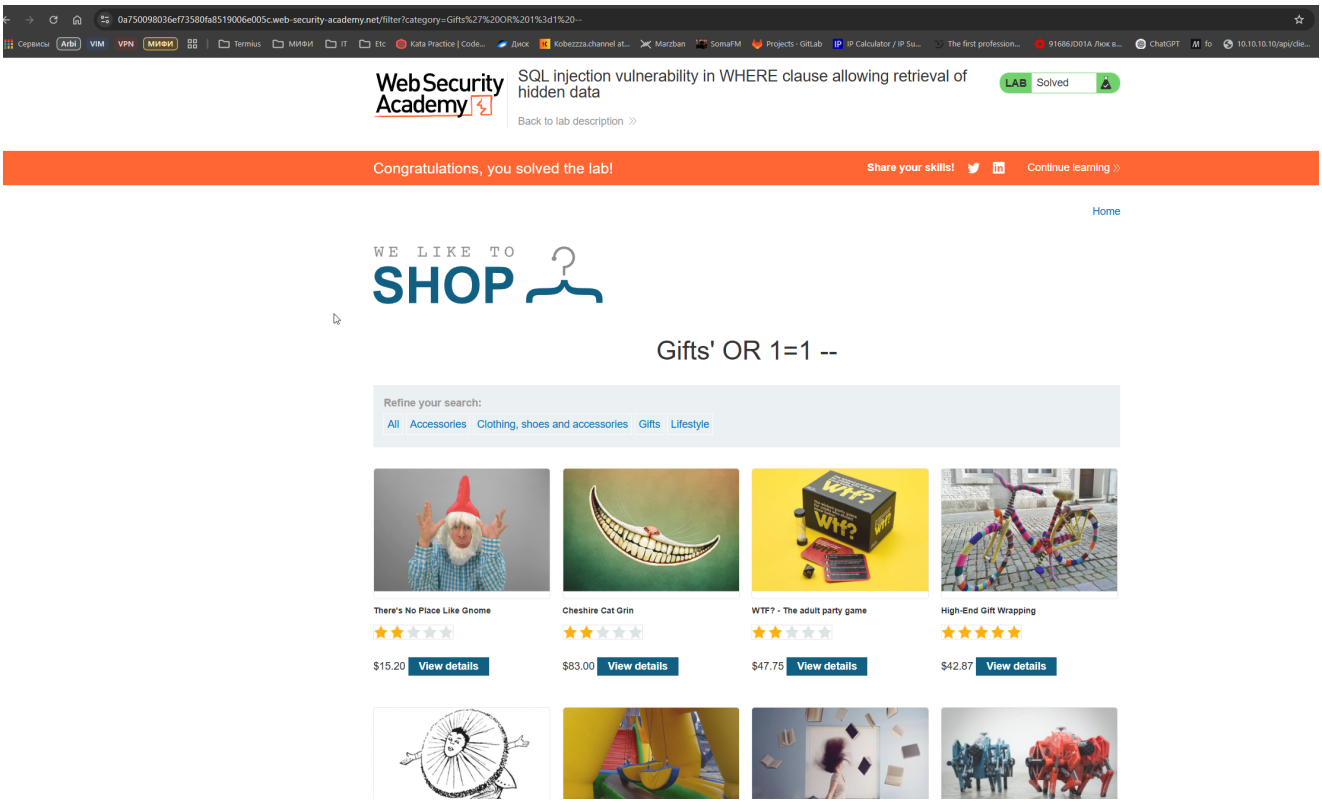
Часть 2: Решите лабораторные работы на сайте PortSwigger

SQL injection vulnerability in WHERE clause allowing retrieval of hidden data

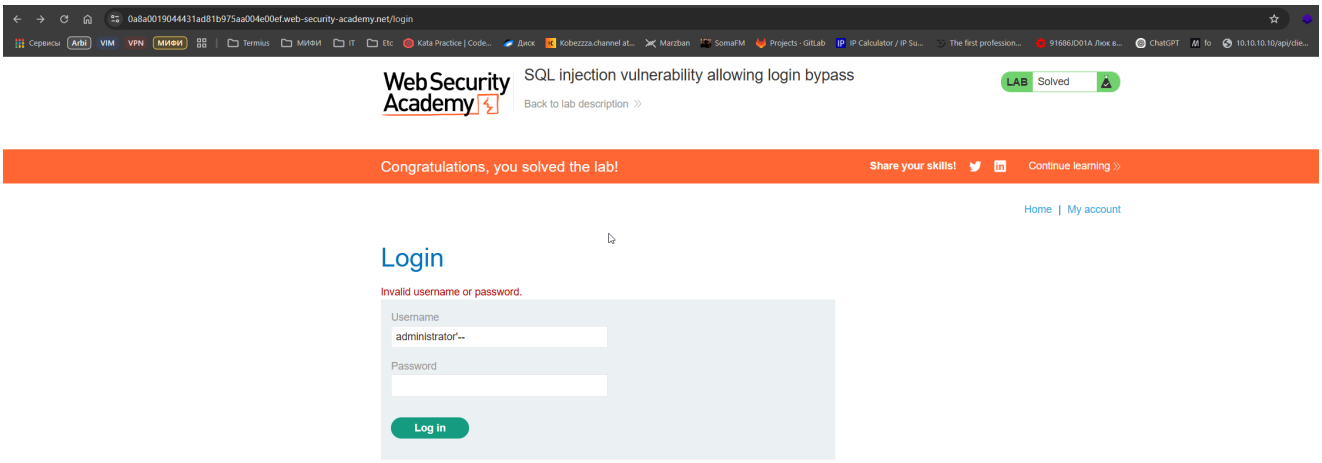
Проведем подготовку SQL инъекции, закодировав нужные данные SQL инъекции в URL-encoded строку с помощью Burp Suite:



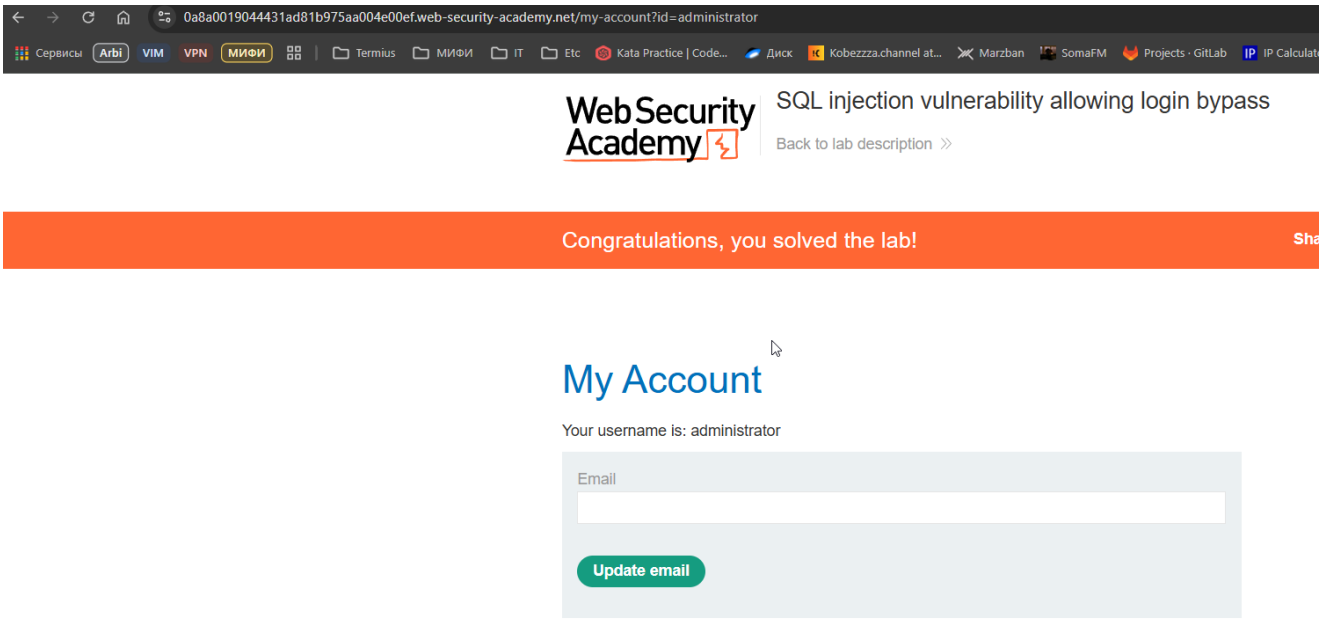
Выполним атаку:



SQL injection vulnerability allowing login bypass



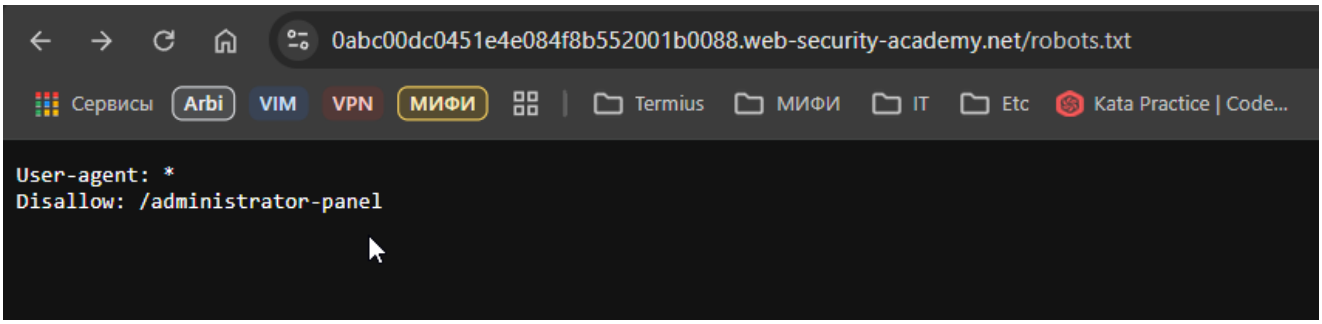
Вторая лабораторная решается аналогичным образом - передаем логин `administrator`,закрываем кавычкой запрос и дальше `--` для комментирования последующего содержимого запроса.



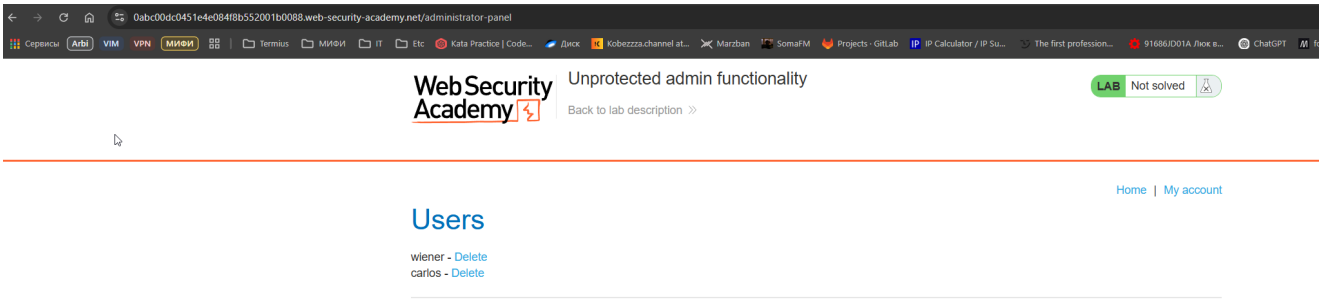
Успешный вход

Lab: Unprotected admin functionality

Если попробовать пройти по ссылке `/robots.txt`, то мы увидим незащищенную функциональность - файл robots.txt, который раскрывает нам некоторую дополнительную информацию о содержимом сервера.
Мы видим, что сайт имеет незакрытый аутентификацией путь `/administrator-panel`

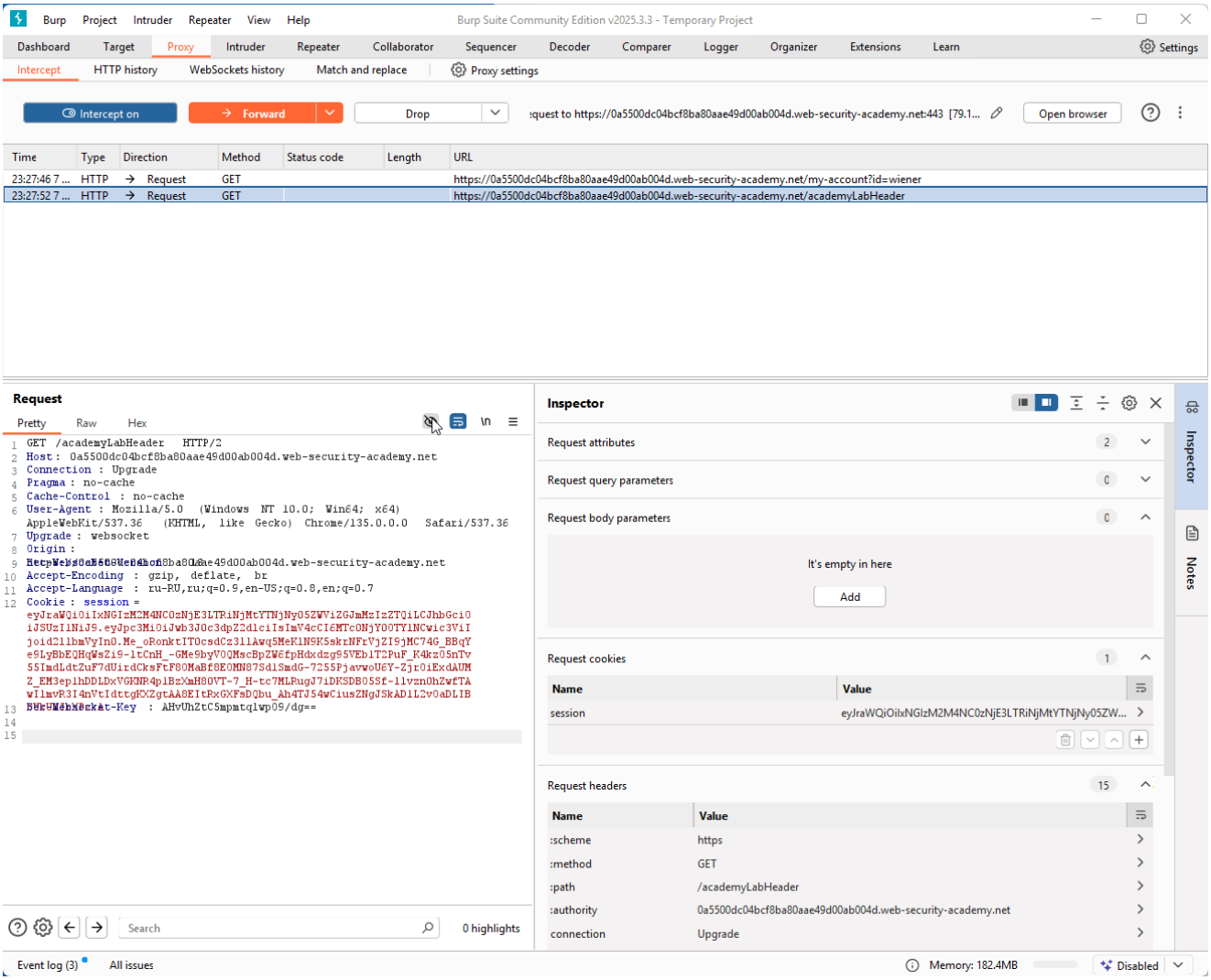


Пройдя по этому пути - нам открывается админ панель, не защищенная никаким паролем и возможность удалить нужного пользователя.

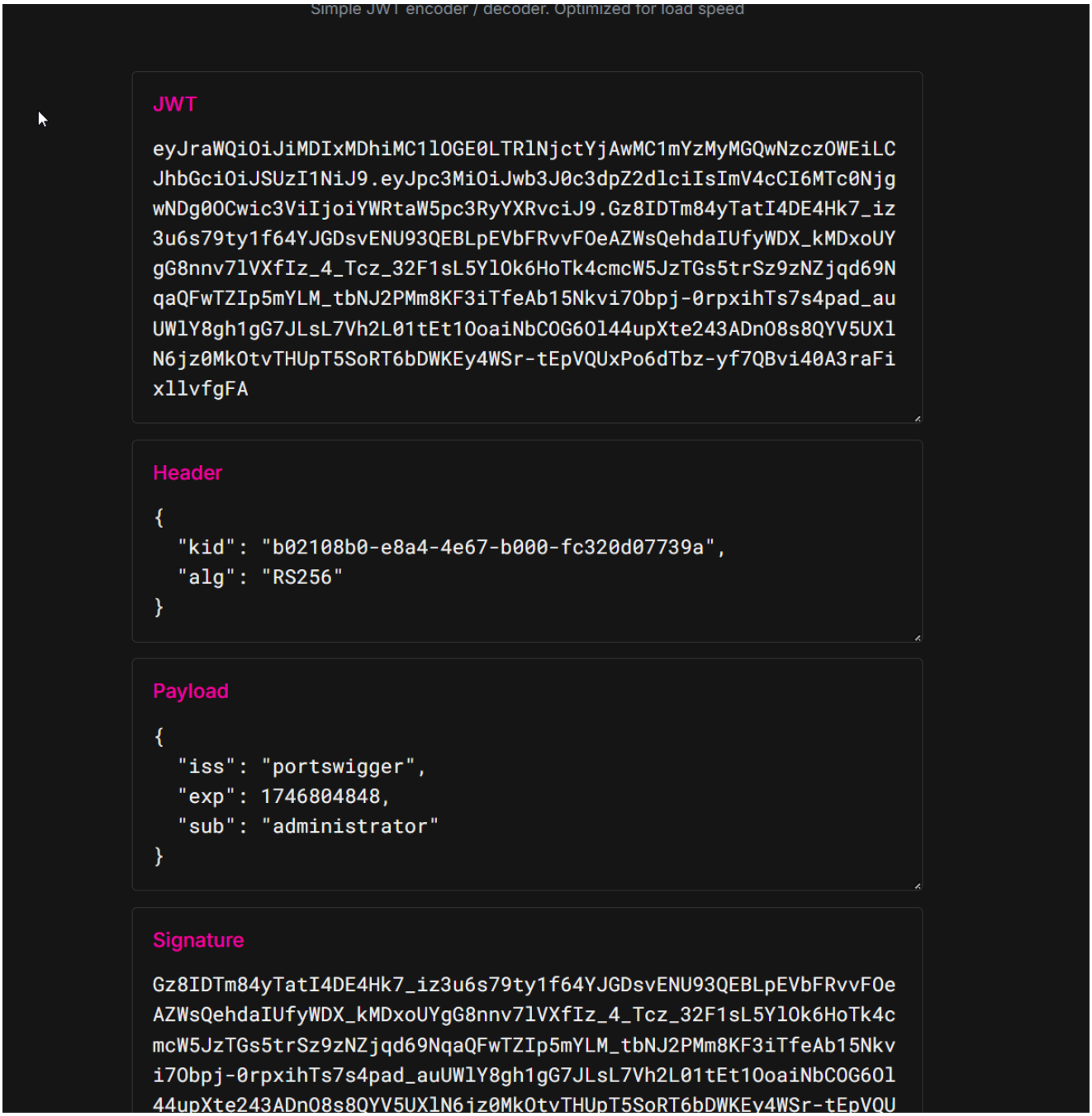


JWT authentication bypass via unverified signature

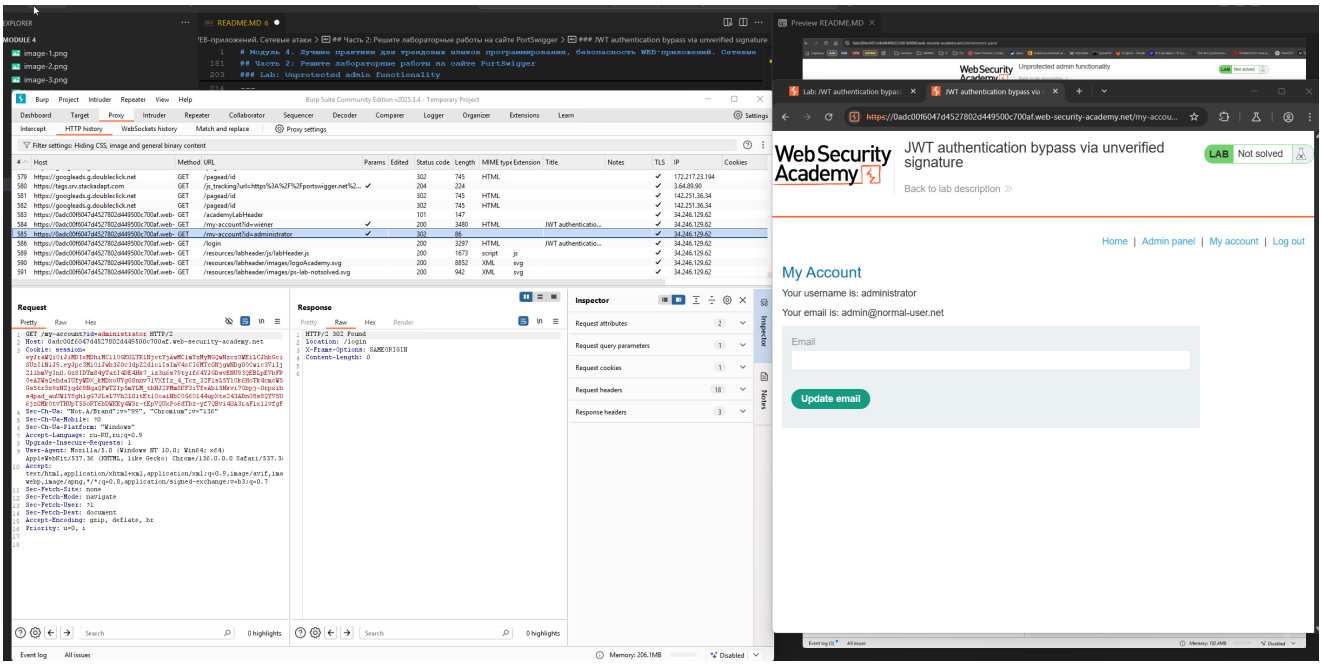
Через Burp Suite получаем JWT токен после авторизации под доступным нам пользователем



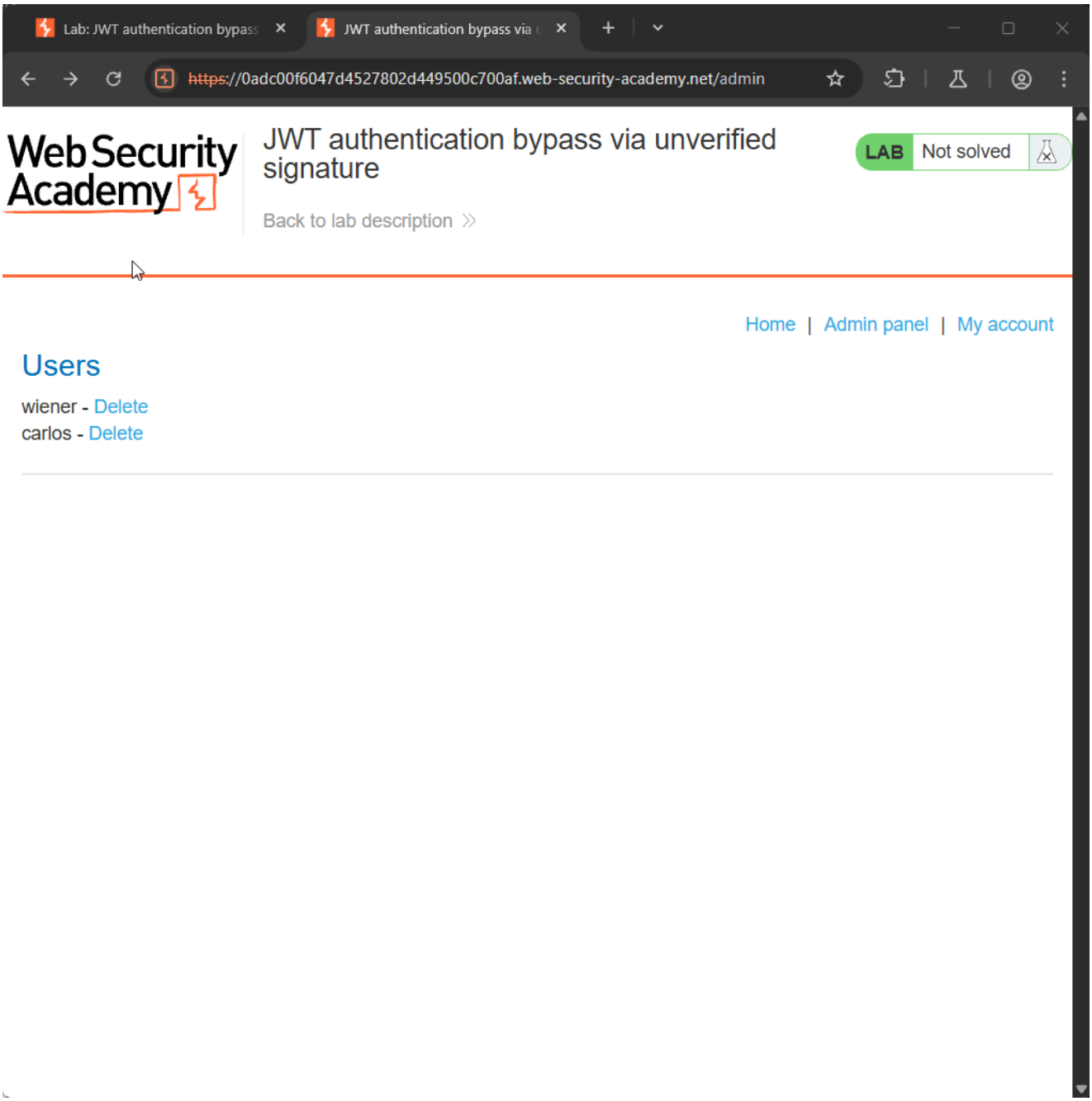
Раскодируем JWT-токен через любой онлайн сервис по работе с JWT токенами и подменяем username на **administrator**



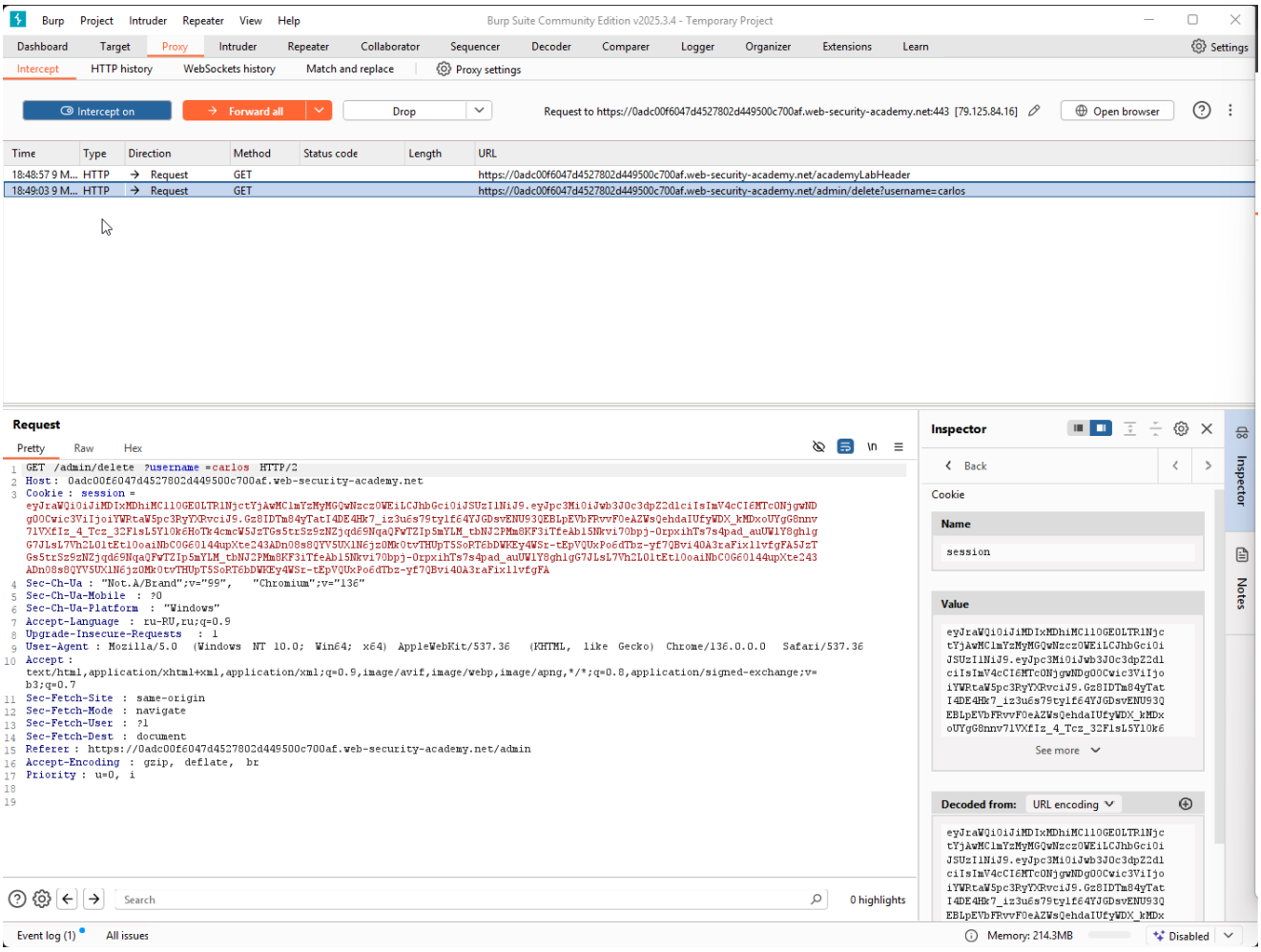
Полученный в результате подмены JWT-токен заменяем в заголовке Cookie **session** для GET-запроса `/my-account`. Так же заменяем query-параметр **id** в запросе на **administrator**

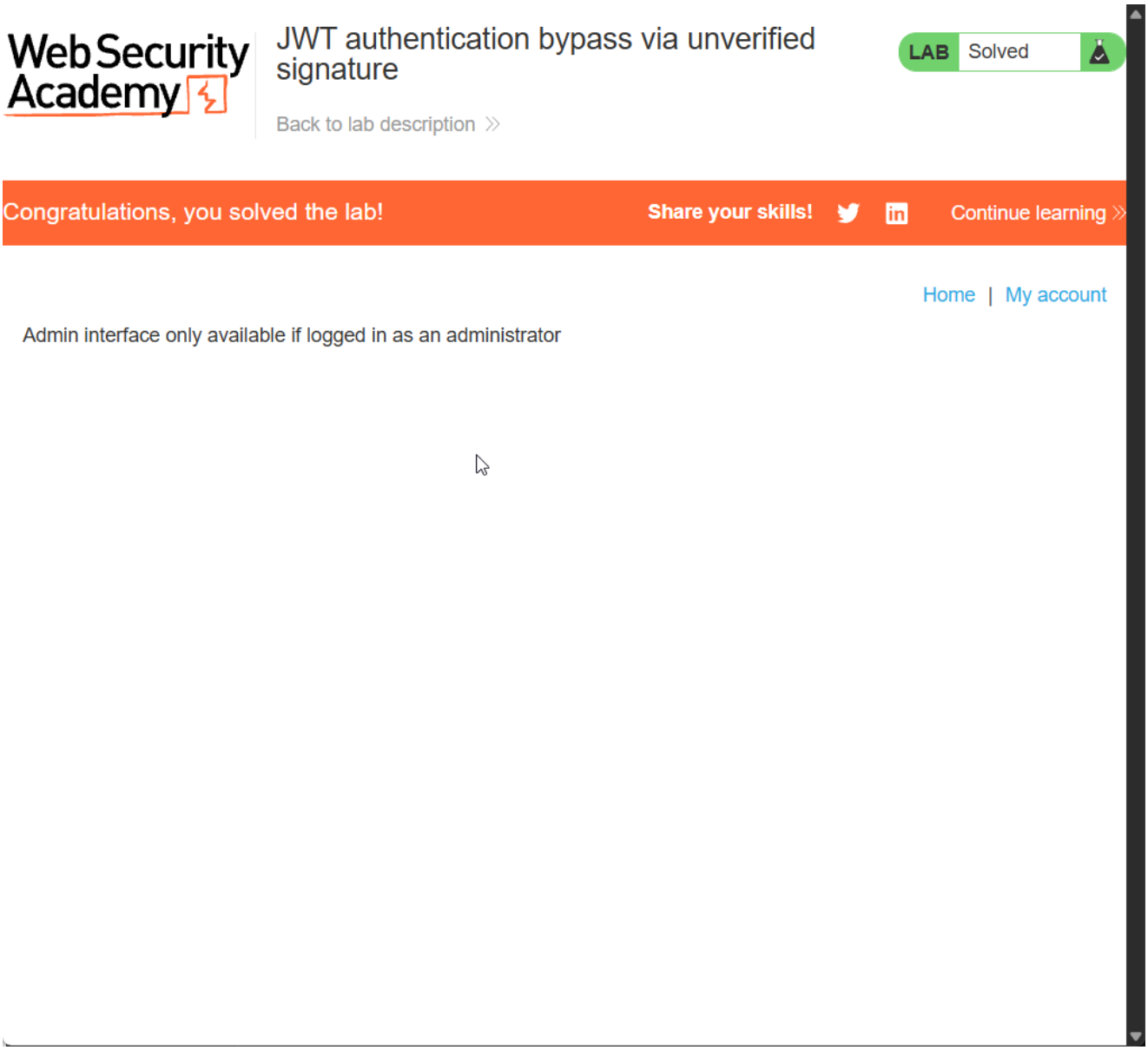


В результате с полученным JWT-токеном мы попадаем в личный кабинет пользователя с рашширенными правами администратора и получаем доступ к доп. пункту меню **Admin panel**



Удаляем нужную учетную запись (запрос отправляется с тем же подмененным токеном, иначе не пройдет)

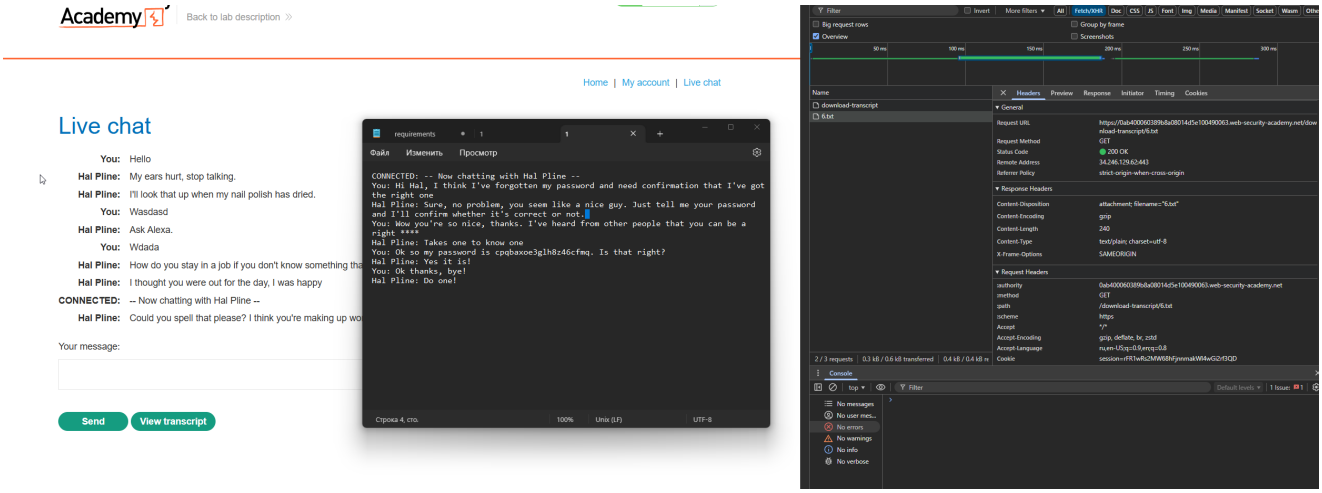

Burp Project Intruder Repeater View Help
Burp Suite Community Edition v2025.3.4 - Temporary Project
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings
Intercept HTTP history WebSockets history Match and replace Proxy settings
Intercept on Forward all Drop
Request to https://0adc00f6047d4527802d449500c700af.web-security-academy.net:443 [79.125.84.16]
Open browser
Time Type Direction Method Status code Length URL
18:48:57 9 M... HTTP Request GET https://0adc00f6047d4527802d449500c700af.web-security-academy.net/academyLabHeader
18:49:03 9 M... HTTP Request GET https://0adc00f6047d4527802d449500c700af.web-security-academy.net/admin/delete?username=carlos
Request
Pretty Raw Hex
1 GET /admin/delete ?username =carlos HTTP/2
2 Host : 0adc00f6047d4527802d449500c700af.web-security-academy.net
3 Cookie : session =
eyJraWQiOiJiMDIxMDhiMC1lOGE0LTRlNjctYjAwMC1mYzMyMGQwNzc0WEiLCJhbGci0iJSUzIlNiJ9.eyJpc3Mi0iJwb3J0c3dpZ2dlciIsImV4cCI6MTc0NjgwND
g00Cwic3ViIjoiYWRtaW5pc3RyYXRvciJ9.Gz8IDTm84yTatI4DE4Hk7_iz3u6s79tylf64YJGDsvENU93QEBLpEVbFRvvF0eAZWsQehdaIUfyWDX_kMDxoUYgG8nnv
71VXfIz_4_Tcz_32FlsL5Yl0k6HoTk4cmcW5JzTGs5trSzSzNZjqd69NqaQFwTZIp5mYLM_tbNJ2PMm8KF3iTfeAb15Nkvi70bpj-OrpxihTs7s4pad_auUWlY8ghlg
G7JLsL7Vh2L01tEtl0oaiNbC0G6l44upXte243ADn08s8QYV5UXlN6jz0Mk0tvTHUpT5SoRT6bDWKEy4WSr-tEpVQUxPo6dTbz-yf7QBvi40A3raFixllvfgFA5JzT
Gs5trSzSzNZjqd69NqaQFwTZIp5mYLM_tbNJ2PMm8KF3iTfeAb15Nkvi70bpj-OrpxihTs7s4pad_auUWlY8ghlgG7JLsL7Vh2L01tEtl0oaiNbC0G6l44upXte243
ADn08s8QYV5UXlN6jz0Mk0tvTHUpT5SoRT6bDWKEy4WSr-tEpVQUxPo6dTbz-yf7QBvi40A3raFixllvfgFA
4 Sec-Ch-Ua : "Not.A/Brand";v="99", "Chromium";v="136"
5 Sec-Ch-Ua-Mobile : ?0
6 Sec-Ch-Ua-Platform : "Windows"
7 Accept-Language : ru-RU,ru;q=0.9
8 Upgrade-Insecure-Requests : 1
9 User-Agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/136.0.0.0 Safari/537.36
10 Accept :
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=
b3;q=0.7
11 Sec-Fetch-Site : same-origin
12 Sec-Fetch-Mode : navigate
13 Sec-Fetch-User : ?1
14 Sec-Fetch-Dest : document
15 Referer : https://0adc00f6047d4527802d449500c700af.web-security-academy.net/admin
16 Accept-Encoding : gzip, deflate, br
17 Priority : u=0, i
18
19
Search
0 highlights
Inspector
Back
Cookie
Name
session
Value
eyJraWQiOiJiMDIxMDhiMC1lOGE0LTRlNjc
tYjAwMC1mYzMyMGQwNzc0WEiLCJhbGci0i
JSUzIlNiJ9.eyJpc3Mi0iJwb3J0c3dpZ2dl
ciIsImV4cCI6MTc0NjgwNDg00Cwic3ViIjo
iYWRtaW5pc3RyYXRvciJ9.Gz8IDTm84yTat
I4DE4Hk7_iz3u6s79tylf64YJGDsvENU93Q
EBLpEVbFRvvF0eAZWsQehdaIUfyWDX_kMDx
oUYgG8nnv7lVXfIz_4_Tcz_32FlsL5Yl0k6
See more
Decoded from: URL encoding
eyJraWQiOiJiMDIxMDhiMC1lOGE0LTRlNjc
tYjAwMC1mYzMyMGQwNzc0WEiLCJhbGci0i
JSUzIlNiJ9.eyJpc3Mi0iJwb3J0c3dpZ2dl
ciIsImV4cCI6MTc0NjgwNDg00Cwic3ViIjo
iYWRtaW5pc3RyYXRvciJ9.Gz8IDTm84yTat
I4DE4Hk7_iz3u6s79tylf64YJGDsvENU93Q
EBLpEVbFRvvF0eAZWsQehdaIUfyWDX_kMDx
Event log (1)
All issues
Memory: 214.3MB
Disabled



Insecure direct object references

Тут все просто.

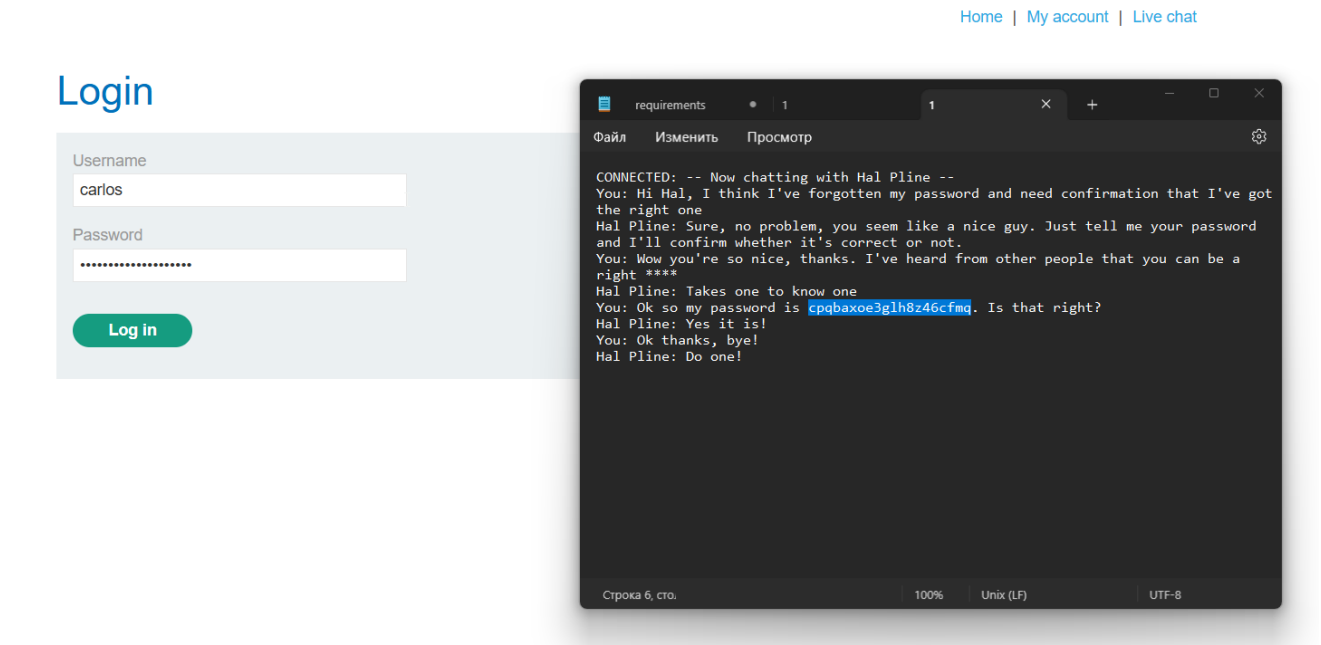
Через network inspector в хроме - смотрим какая ссылка генерируется для скачивания текстового файла.



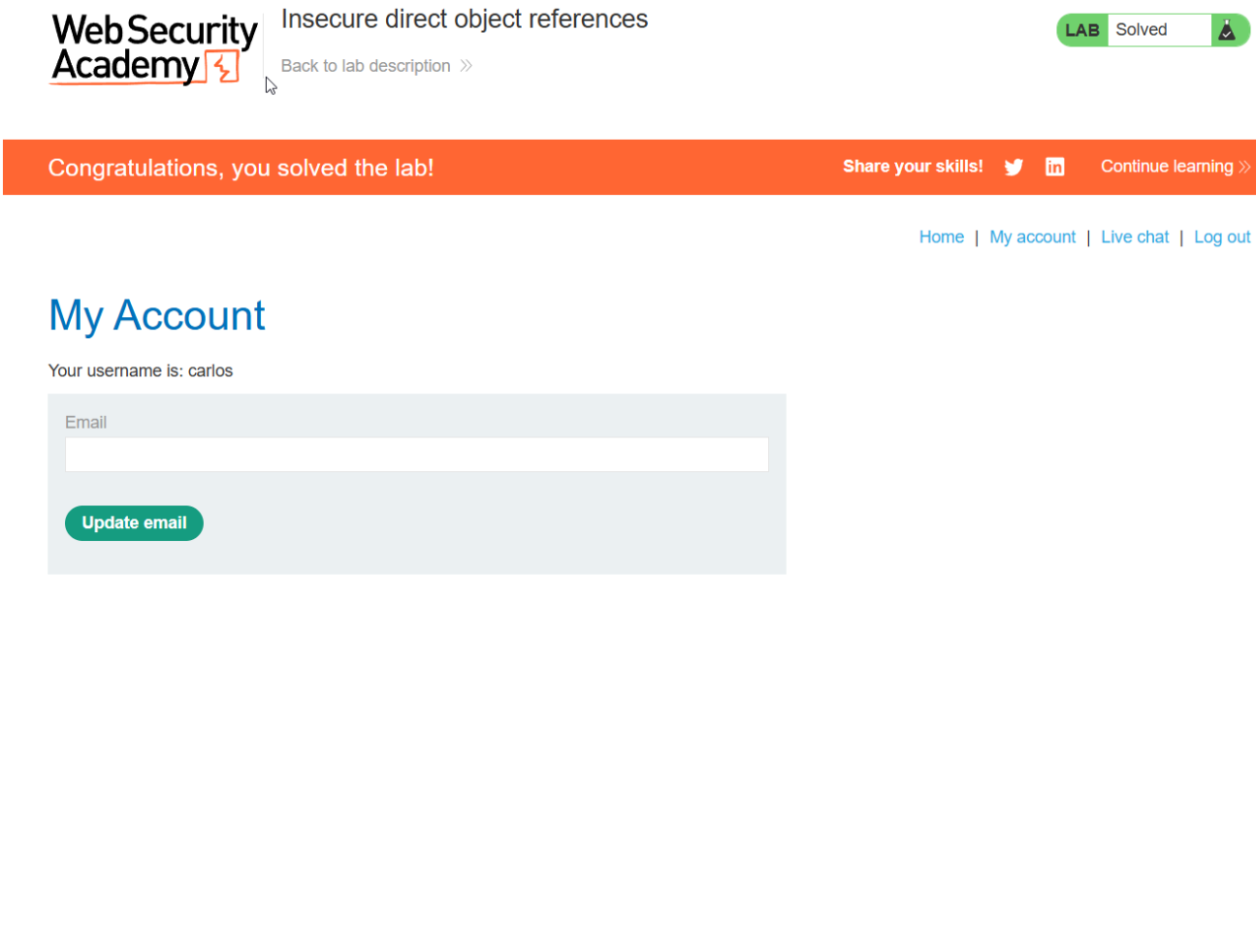
Обращаем внимание, что при скачивании файла был пропущен 1.txt - наверное это то, что нам надо!

Скачиваем файл 1.txt и смотрим его содержимое. В его содержимом видим переписку, в которой упоминается искомый пароль.

Пробуем...



Успех!


Web Security Academy
Insecure direct object references
Back to lab description
LAB Solved
Congratulations, you solved the lab!
Share your skills!
Continue learning
Home | My account | Live chat | Log out
My Account
Your username is: carlos
Email
Update email