Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Информационные сети. Основы безопасности

ОТЧЕТ

к лабораторной работе №2

на тему

**ИДЕНТИФИКАЦИЯ И АУТЕНТИФИКАЦИЯ ПОЛЬЗОВАТЕЛЕЙ. ПРОТОКОЛ KERBEROS**

Студент Д. С. Кончик

Преподаватель Е. А. Лещенко

Минск 2024

СОДЕРЖАНИЕ

[Введение 3](#_Toc157471308)

[1 Результат выполнения 4](#_Toc157471309)

[Заключение 5](#_Toc157471310)

[Приложение А (обязательное) Листинг кода 6](#_Toc157471311)

**ВВЕДЕНИЕ**

Лабораторная работа посвящена исследованию протокола *Kerberos*, который представляет собой эффективный механизм идентификации и аутентификации пользователей в распределенных вычислительных сетях. Протокол *Kerberos* является одним из наиболее широко используемых реализаций протоколов аутентификации с третьей стороной, обеспечивая высокий уровень безопасности и снижая количество передаваемых сообщений между участниками.

В ходе работы предполагается изучение теоретических основ протокола *Kerberos*, включая его принципы функционирования и алгоритмы, а также алгоритма шифрования *DES (Data Encryption Standard)*.

Также предполагается реализация основных этапов протокола *Kerberos* на языке программирования *Python*, включая взаимодействие между клиентом, сервером аутентификации *(AS)*, сервером выдачи разрешений *(TGS)* и конечным сервером *(SS)*.

1. **РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ**

В результате работы было создано четыре приложения, представляющих собой клиента *(C)*, сервер аутентификации *(AS)*, сервер выдачи разрешения *(TGS)* и конечный сервер *(SS)*.

Также для наглядности взаимодействия сторон предусмотрен подробный вывод в консоль исходный данных протокола и данных, передаваемых каждой из сторон.

На рисунке 1 представлено логирование в консоль всех данных, обрабатываемых и отправляемых клиентом.

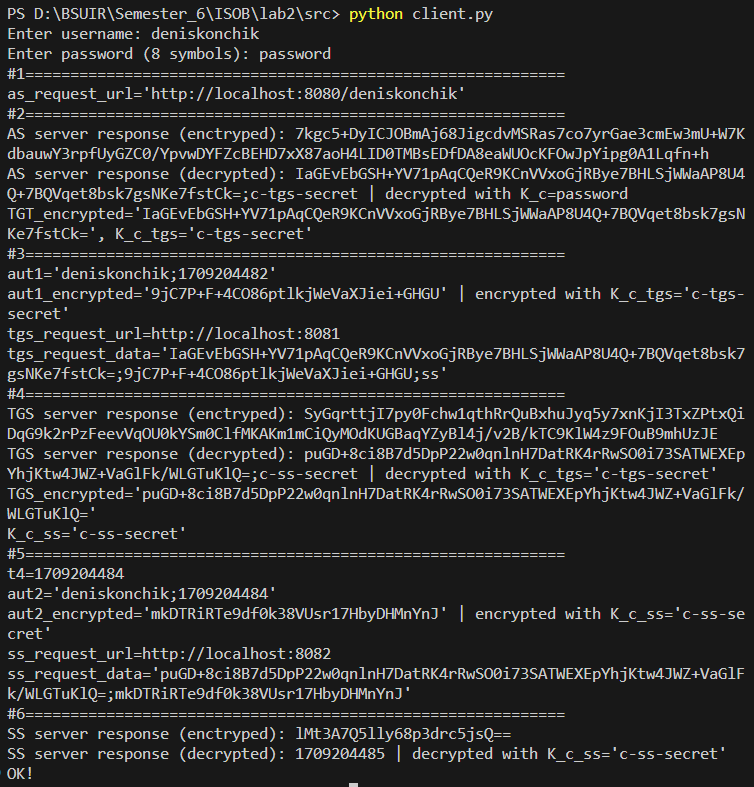


Рисунок 1 – Вывод приложения «Клиент»

# ЗАКЛЮЧЕНИЕ

В ходе выполнения лабораторной работы было успешно исследовано и реализовано взаимодействие по протоколу *Kerberos* с использованием языка программирования Python. Созданные приложения – клиент, сервер аутентификации, сервер выдачи разрешений и конечный сервер – были разработаны с учетом основных этапов протокола *Kerberos*.

Основными достижениями работы является понимание принципов функционирования протокола *Kerberos*, включая процессы аутентификации, обмена билетами и шифрования данных с использованием алгоритма *DES*. Созданные приложения обеспечивают наглядное взаимодействие между сторонами протокола и подробный вывод в консоль исходных данных и данных, передаваемых каждой из сторон.

Таким образом, выполнение лабораторной работы позволило успешно достичь поставленных целей и задач, а также приобрести практические навыки в области обеспечения информационной безопасности и применения протокола *Kerberos*.

# ПРИЛОЖЕНИЕ А (обязательное) Листинг кода

Листинг 1 – Файл *client.py*

import requests

import time

from common import encrypt, decrypt

SERVER\_URL = {

    "AS": "http://localhost:8080",

    "TGS": "http://localhost:8081",

    "SS": "http://localhost:8082"

}

SERVER\_ID = {

    "SS": "ss"

}

def main():

    username = input("Enter username: ")

    password = input("Enter password (8 symbols): ")

    while len(password) != 8:

        print("Password must be 8 symbols long.")

        password = input("Enter password (8 symbols): ")

    # 1

    print('#1============================================================')

    as\_request\_url = f"{SERVER\_URL['AS']}/{username}"

    print(f'{as\_request\_url=}')

    response = requests.get(as\_request\_url)

    if response.status\_code != 200:

        print("AS server answered with:", response.status\_code)

        exit(-1)

    # 2

    print('#2============================================================')

    print("AS server response (enctryped):", response.text)

    response = decrypt(response.text, password)

    print("AS server response (decrypted):", response, f"| decrypted with K\_c={password}")

    TGT\_encrypted, K\_c\_tgs = response.split(';')

    print(f'{TGT\_encrypted=}, {K\_c\_tgs=}')

    # 3

    print('#3============================================================')

    aut1 = f"{username};{int(time.time())}"

    print(f'{aut1=}')

    aut1\_encrypted = encrypt(aut1, K\_c\_tgs)

    print(f'{aut1\_encrypted=} | encrypted with {K\_c\_tgs=}')

    tgs\_request\_data = f"{TGT\_encrypted};{aut1\_encrypted};{SERVER\_ID['SS']}"

    print(f'tgs\_request\_url={SERVER\_URL["TGS"]}')

    print(f'{tgs\_request\_data=}')

    response = requests.post(SERVER\_URL["TGS"], data=tgs\_request\_data)

    if response.status\_code != 200:

        print("TGS server answered with:", response.status\_code)

        exit(-1)

    # 4

    print('#4============================================================')

    print("TGS server response (enctryped):", response.text)

    response = decrypt(response.text, K\_c\_tgs)

    print("TGS server response (decrypted):", response, f"| decrypted with {K\_c\_tgs=}")

    TGS\_encrypted, K\_c\_ss = response.split(';')

    print(f'{TGS\_encrypted=}')

    print(f'{K\_c\_ss=}')

    # 5

    print('#5============================================================')

    t4 = int(time.time())

    aut2 = f"{username};{t4}"

    aut2\_encrypted = encrypt(aut2, K\_c\_ss)

    print(f'{t4=}')

    print(f'{aut2=}')

    print(f'{aut2\_encrypted=}', f'| encrypted with {K\_c\_ss=}')

    ss\_request\_data = f"{TGS\_encrypted};{aut2\_encrypted}"

    print(f'ss\_request\_url={SERVER\_URL["SS"]}')

    print(f'{ss\_request\_data=}')

    response = requests.post(SERVER\_URL["SS"], data=ss\_request\_data)

    if response.status\_code != 200:

        print("SS server answered with:", response.status\_code)

        exit(-1)

    # 6

    print('#6============================================================')

    print("SS server response (enctryped):", response.text)

    response = decrypt(response.text, K\_c\_ss)

    print("SS server response (decrypted):", response, f'| decrypted with {K\_c\_ss=}')

    if int(response.split(';')[0]) != t4 + 1:

        print("SS server verification failed!")

        exit(1)

    print("OK!")

if \_\_name\_\_ == '\_\_main\_\_':

    main()

Листинг 2 – Файл *as.py*

from flask import Flask

from common import encrypt, decrypt

import time

app = Flask(\_\_name\_\_)

users = {

    "deniskonchik" : {

        "password" : "password"

    }

}

KEYS = {

    "AS\_TGS" : 'as-tgs-secret',

    "C\_TGS" : 'c-tgs-secret'

}

SERVER\_ID = {

    "TGS": 'tgs'

}

@app.route('/<username>', methods=['GET'])

def handle(username):

    print(f'{username=}')

    if username not in users:

        return "Forbidden", 403

    current\_time = int(time.time())

    end\_time = current\_time + 3600

    print(f'{current\_time=}, {end\_time=}')

    TGT = f"{username};{SERVER\_ID['TGS']};{current\_time};{end\_time};{KEYS['C\_TGS']}"

    TGT\_encrypted = encrypt(TGT, KEYS["AS\_TGS"])

    print(f'{TGT=}')

    print(f'{TGT\_encrypted=} | encrypted with AS\_TGS={KEYS["AS\_TGS"]}')

    answer = f"{TGT\_encrypted};{KEYS['C\_TGS']}"

    answer\_encrypted = encrypt(answer, users[username]['password'])

    print(f'{answer=}')

    print(f'{answer\_encrypted=} | encrypted with K\_c={users[username]["password"]}')

    return answer\_encrypted, 200

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(host='127.0.0.1', port=8080)

Листинг 3 – Файл *ss.py*

from flask import Flask, request

from common import encrypt, decrypt

app = Flask(\_\_name\_\_)

KEYS = {

    "TGS\_SS": 'tgs-ss-secret'

}

@app.route('/', methods=['POST'])

def handle():

    data = request.data.decode('utf-8')

    print(f'{data=}')

    TGS\_encrypted, aut2\_encrypted = data.split(';')

    print(f'{TGS\_encrypted=}')

    print(f'{aut2\_encrypted=}')

    TGS\_decrypted = decrypt(TGS\_encrypted, KEYS['TGS\_SS'])

    print(f'{TGS\_decrypted=} | decrypted with TGS\_SS={KEYS["TGS\_SS"]}')

    tgs\_username, ss\_id, tgs\_start\_time, tgs\_end\_time, K\_c\_ss = TGS\_decrypted.split(';')

    print(f'{tgs\_username=}, {ss\_id=}, {tgs\_start\_time=}, {tgs\_end\_time=}, {K\_c\_ss=}')

    aut2\_decrypted = decrypt(aut2\_encrypted, K\_c\_ss)

    print(f'{aut2\_decrypted=} | decrypted with {K\_c\_ss=}')

    aut2\_username, aut2\_time = decrypt(aut2\_encrypted, K\_c\_ss).split(';')

    print(f'{aut2\_username=}, {aut2\_time=}')

    if tgs\_username != aut2\_username:

        return "Forbidden (invalid user)", 403

    if not (int(tgs\_start\_time) <= int(aut2\_time) <= int(tgs\_end\_time)):

        return "Unauthorized (ticket expired)", 401

    answer = f'{int(aut2\_time) + 1}'

    print(f'{answer=}')

    answer\_encrypted = encrypt(answer, K\_c\_ss)

    print(f'{answer\_encrypted=} | encrypted with {K\_c\_ss=}')

    return answer\_encrypted, 200

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(host='127.0.0.1', port=8082)

Листинг 4 – Файл *tgs.py*

from flask import Flask, request

import time

from common import encrypt, decrypt

app = Flask(\_\_name\_\_)

KEYS = {

    "AS\_TGS" : 'as-tgs-secret',

    "C\_SS": 'c-ss-secret',

    "TGS\_SS": 'tgs-ss-secret'

}

@app.route('/', methods=['POST'])

def handle():

    data = request.data.decode('utf-8')

    print(f'{data=}')

    TGT\_encrypted, aut1\_encrypted, ss\_id = data.split(';')

    print(f'{TGT\_encrypted=}, {aut1\_encrypted=}, {ss\_id=}')

    TGT\_decrypted = decrypt(TGT\_encrypted, KEYS['AS\_TGS'])

    print(f'{TGT\_decrypted=}')

    tgt\_username, tgs\_id, tgt\_start\_time, tgt\_end\_time, K\_c\_tgs = TGT\_decrypted.split(';')

    print(f'{tgt\_username=}, {tgs\_id=}, {tgt\_start\_time=}, {tgt\_end\_time=}, {K\_c\_tgs=}')

    aut1\_decrypted = decrypt(aut1\_encrypted, K\_c\_tgs)

    print(f'{aut1\_decrypted=} | decrypted with {K\_c\_tgs=}')

    aut1\_username, aut1\_time = aut1\_decrypted.split(';')

    print(f'{aut1\_username=}, {aut1\_time=}')

    if tgt\_username != aut1\_username:

        return "Forbidden (invalid user)", 403

    if not (int(tgt\_start\_time) <= int(aut1\_time) <= int(tgt\_end\_time)):

        return "Unauthorized (ticket expired)", 401

    current\_time = int(time.time())

    end\_time = current\_time + 3600

    print(f'{current\_time=}, {end\_time=}')

    TGS = f"{tgt\_username};{ss\_id};{current\_time};{end\_time};{KEYS['C\_SS']}"

    print(f'{TGS=}')

    TGS\_encrypted = encrypt(TGS, KEYS['TGS\_SS'])

    print(f'{TGS\_encrypted=} | encrypted with TGS\_SS={KEYS["TGS\_SS"]}')

    answer = f"{TGS\_encrypted};{KEYS['C\_SS']}"

    print(f'{answer=}')

    answer\_encrypted = encrypt(answer, K\_c\_tgs)

    print(f'{answer\_encrypted=} | ecrnypted with {K\_c\_tgs=}')

    return answer\_encrypted, 200

if \_\_name\_\_ == '\_\_main\_\_':

    app.run(host='127.0.0.1', port=8081)