Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Информационные сети. Основы безопасности

ОТЧЁТ
к лабораторной работе №3
на тему

ИДЕНТИФИКАЦИЯ И АУТЕНТИФИКАЦИЯ ПОЛЬЗОВАТЕЛЕЙ. ПРОТОКОЛ KERBEROS

Выполнил: студент гр. 253505
Сенько Н. С.

Проверил: ассистент кафедры информатики
Герчик А. В.

Минск 2025

СОДЕРЖАНИЕ

1 ЦЕЛЬ РАБОТЫ

Целью данной лабораторной работы является изучение принципов идентификации и аутентификации пользователей с использованием протокола Kerberos. В ходе работы необходимо ознакомиться с архитектурой протокола, его основными компонентами и механизмами работы. Также предполагается реализовать простую симуляцию процесса аутентификации с использованием Kerberos, что позволит лучше понять его функционирование и применение в современных системах безопасности.

2 ХОД РАБОТЫ

Протокол Kerberos был разработан для обеспечения безопасной аутентификации пользователей в сетевых системах. Он использует централизованный подход, где Центр распределения ключей (KDC) играет ключевую роль в процессе аутентификации.

2.1 Архитектура Kerberos

Kerberos состоит из следующих основных компонентов:
– Клиент: Пользователь или приложение, запрашивающее доступ к сервису.
– Сервер: Сервис, к которому клиент хочет получить доступ.
– KDC (Центр распределения ключей): Доверенный сервер, который выдает ключи и билеты для аутентификации.

2.2 Процесс аутентификации

Процесс аутентификации в Kerberos включает несколько этапов:
– Запрос аутентификации (AS_REQ): Клиент отправляет запрос на KDC с идентификационными данными.
– Ответ KDC (AS_REP): KDC проверяет данные и отправляет клиенту зашифрованный сеансовый ключ и TGT (Ticket Granting Ticket).
– Запрос мандата (TGS_REQ): Клиент использует TGT для запроса доступа к конкретному сервису.
– Ответ TGS (TGS_REP): KDC отправляет клиенту мандат для доступа к запрашиваемому сервису.

Таким образом, в ходе выполнения лабораторной работы было реализовано и протестировано программное средство шифрования и дешифрования текстовых файлов двумя шифрами. В процессе разработки были изучены принципы работы выбранных алгоритмов шифрования, их особенности, преимущества и недостатки.

Программа была протестирована на различных входных данных, что позволило убедиться в её корректности и работоспособности. Кроме того, в ходе выполнения лабораторной работы была проведена отладка кода, исправлены возможные ошибки и оптимизированы некоторые участки программы для повышения её производительности.

ЗАКЛЮЧЕНИЕ

В ходе работы была изучена архитектура и принципы работы протокола Kerberos, а также реализована его простая симуляция. Протокол Kerberos обеспечивает надежную аутентификацию пользователей в сетевых системах, используя централизованный подход и симметричное шифрование.

Были рассмотрены основные этапы аутентификации, включая запросы и ответы между клиентом и KDC. Реализованная симуляция позволила лучше понять механизмы работы протокола и его применение в современных системах безопасности.

Таким образом, проделанная работа способствовала углублению знаний в области аутентификации и идентификации пользователей, а также позволила применить на практике навыки программирования для реализации протоколов безопасности.

Приложение А (обязательное)
Исходный код программного продукта

```
import hashlib
import os

class KerberosServer:
    def __init__(self):
        self.clients = {}
        self.services = {}
        self.session_keys = {}

    def register_client(self, client_id, secret):
        self.clients[client_id] = secret

    def register_service(self, service_id, secret):
        self.services[service_id] = secret

    def authenticate(self, client_id):
        if client_id in self.clients:
            session_key = os.urandom(16)
            self.session_keys[client_id] = session_key
            # Здесь вместо настоящего шифрования мы просто возвращаем
хеш от секретного ключа клиента
            client_secret = self.clients[client_id]
            return hashlib.sha256(client_secret +
session_key).hexdigest()
        else:
            raise ValueError("Unknown client")

    def validate_request(self, client_id, service_id, authenticator):
        if client_id in self.clients and service_id in self.services:
            expected_authenticator =
hashlib.sha256(self.clients[client_id] +
self.session_keys[client_id]).hexdigest()
            if authenticator == expected_authenticator:
                return True
        return False


class Client:
    def __init__(self, client_id, secret, kerberos_server):
        self.client_id = client_id
        self.secret = secret
        self.kerberos_server = kerberos_server

    def request_access(self):
        authenticator =
self.kerberos_server.authenticate(self.client_id)
        return authenticator


class Service:
    def __init__(self, service_id, secret, kerberos_server):
        self.service_id = service_id
        self.secret = secret
        self.kerberos_server = kerberos_server

    def grant_access(self, client_id, authenticator):
        if self.kerberos_server.validate_request(client_id,
self.service_id, authenticator):
            print("Access granted to client:", client_id)
```

```
            else:
                print("Access denied for client:", client_id)



    # Пример успешной аутентификации
    print("Пример успешной аутентификации:")
    kerberos_server = KerberosServer()

    # Регистрация клиента и сервиса
    kerberos_server.register_client("client1", b"client_secret")
    kerberos_server.register_service("service1", b"service_secret")

    # Успешный сценарий
    client = Client("client1", b"client_secret", kerberos_server)
    authenticator = client.request_access()
    service = Service("service1", b"service_secret", kerberos_server)
    service.grant_access("client1", authenticator)  # Ожидается успешный
доступ

    print("\nПример неуспешной аутентификации:")

    # Пример неизвестного клиента
    try:
        unknown_client = Client("unknown_client", b"client_secret",
kerberos_server)
        authenticator = unknown_client.request_access()
    except ValueError as e:
        print("Ошибка:", e)  # "Unknown client"

    # Пример неверного аутентификатора
    fake_authenticator = "invalid_authenticator"
    service.grant_access("client1", fake_authenticator)  # Ожидается отказ в
доступе

    # Пример неизвестного сервиса
    unknown_service = Service("unknown_service", b"service_secret",
kerberos_server)
    unknown_service.grant_access("client1", authenticator)  # Ожидается
отказ в доступе
```