**Министерство науки и высшего образования Российской Федерации** ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Факультет безопасности информационных технологий Дисциплина:**

«Организация и обеспечение аудита настроек средств защиты информации»

**ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4**

«**Установка продвинутых компонентов СЗИ**»

**Выполнили:**

Молитвин Илья Алексеевич, студент группы N3345

(подпись)

**Проверил:**

Пенин Андрей Семенович

(отметка о выполнении)

(подпись)

Содержание

Оглавление

[Этап 1 4](#_Toc195685439)

[Этап 2 6](#_Toc195685440)

[Этап 3 8](#_Toc195685441)

[**Заключение** 11](#_Toc195685442)

**ХОД РАБОТЫ**

**Цели:**

Необходимо установить, настроить и протестировать современные средства SPI от НСД на виртуальной машине с операционной системой Windows. К таким расширенным средствам относятся средство обнаружения вторжений и система контроля целостности. Кроме того, необходимо реализовать модель системы аутентификации. Система должна поддерживать хранение логина/пароля в защищенном виде и двухфакторную аутентификацию (способ выбирается на усмотрение пользователя).

Этап 1. Установка и настройка системы обнаружения вторжений (IDS)  
Этап 2. Установка и настройка системы контроля целостности (ICS)

Этап 3. Разработка модели системы аутентификации.

# Этап 1

Этап 1: Установка и настройка системы обнаружения вторжений (IDS)

1. **Выбор и установка**

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.  
Выбрана программа Suricata, установлена с оф сайта.

Рисунок 1 - Suricata

Настроен ямл кфг, папка для логов LogFile, добавлена как служба с помощью команды suricata -c suricata.yaml -i 192.168.200.145 -1 LogFile --service-install. Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 2 – LogFile до записи

Было создано правило на детект Nmap скана.Добавлена в папку рулс и в кфг.  
Содержание правила «alert ip any any -> any any (msg:"Nmap SYN Scan Detected"; flags:S,12; sid:1000002;)»  
С хостовой системы сканирование порта на целевом хосте 192.168.241.130, пытаясь обнаружить открытые порты и версии сервисов, которые их обслуживают, что вызывает срабатывания, так как это нсд.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 4 - Nmap

Фиксация в логах нмап детект через наше правило  
Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 5 – Nmap SYN Scan Detected

# Этап 2

Этап 2: Установка и настройка системы контроля целостности (ICS).

Устанавливаем AIDE на линукс систему.

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рисунок 6 – aide

Иницилизируем дб с нашим измененным кфг,переиминовываем дб,добавляем тест файл в etc,сверяем отчеты и фиксируем изменения в report.txt

Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки. **Рисунок 7 – aide.conf**Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

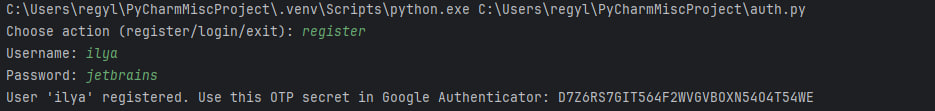
**Рисунок 8 – report.txt**

# Этап 3

Этап 3. Разработка модели системы аутентификации.

**Код программы:**import sqlite3  
import bcrypt  
import pyotp  
import re  
  
# Создание базы данных  
conn = sqlite3.connect("users.db")  
cursor = conn.cursor()  
cursor.execute('''  
CREATE TABLE IF NOT EXISTS users (  
 username TEXT PRIMARY KEY,  
 password\_hash TEXT NOT NULL,  
 otp\_secret TEXT NOT NULL  
)  
''')  
conn.commit()  
  
# Проверка логина и пароля  
def validate\_input(username, password):  
 return bool(re.match(r"^[a-zA-Z0-9\_.-]{3,20}$", username)) and len(password) >= 6  
  
# Регистрация  
def register(username, password):  
 if not validate\_input(username, password):  
 return "Invalid username or password format."  
  
 # Хэширование пароля  
 hashed = bcrypt.hashpw(password.encode(), bcrypt.gensalt())  
 # Создание секретного ключа для 2FA  
 otp\_secret = pyotp.random\_base32()  
  
 try:  
 cursor.execute("INSERT INTO users VALUES (?, ?, ?)", (username, hashed.decode(), otp\_secret))  
 conn.commit()  
 print(f"User '{username}' registered. Use this OTP secret in Google Authenticator: {otp\_secret}")  
 except sqlite3.IntegrityError:  
 return "Username already exists."  
  
# Аутентификация  
def login(username, password, otp\_code):  
 cursor.execute("SELECT password\_hash, otp\_secret FROM users WHERE username = ?", (username,))  
 result = cursor.fetchone()  
 if not result:  
 return "User not found."  
  
 stored\_hash, otp\_secret = result  
 if not bcrypt.checkpw(password.encode(), stored\_hash.encode()):  
 return "Incorrect password."  
  
 totp = pyotp.TOTP(otp\_secret)  
 if not totp.verify(otp\_code):  
 return "Invalid 2FA code."  
  
 return "Authentication successful."  
  
# Пример использования  
if \_\_name\_\_ == "\_\_main\_\_":  
 while True:  
 action = input("Choose action (register/login/exit): ").strip()  
 if action == "register":  
 u = input("Username: ")  
 p = input("Password: ")  
 print(register(u, p))  
 elif action == "login":  
 u = input("Username: ")  
 p = input("Password: ")  
 c = input("Enter 2FA code: ")  
 print(login(u, p, c))  
 elif action == "exit":  
 break

Регестрируемся,получаем OTP Secret для 2fa,вводим в authenticator, получаем 2fa.



**Рисунок 9 – OTP secret для Google Authenticator**

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

**Рисунок 10 – Google Authenticator**

Вводим из проложения код, логинимся.

Изображение выглядит как текст, снимок экрана, программное обеспечение, Мультимедийное программное обеспечение

Контент, сгенерированный ИИ, может содержать ошибки.

**Рисунок 11 – Успешный логин**

## **Заключение**

В ходе выполнения лабораторной работы были успешно установлены и настроены средства для защиты информации. В первом этапе был установлен Suricata как система обнаружения вторжений (IDS), настроены параметры мониторинга сети для выявления подозрительных запросов и протестированы искусственные сценарии атак с использованием Nmap. На втором этапе была установлена система контроля целостности файлов AIDE, настроены параметры для мониторинга изменений в критичных каталогах, а также проведен тест с искусственной модификацией файлов. В третьем этапе была реализована система аутентификации с сохранением логинов и паролей в зашифрованном виде, а также двухфакторная аутентификация через Google Authenticator и Python. Все этапы были протестированы и задокументированы с соответствующими скриншотами.