**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 МОДУЛЮ №4**

«Установка продвинутых компонентов СЗИ»

**Выполнил:**

Чу Ван Доан, студент группы N3347

*\_\_\_\_\_\_\_\_\_\_*

(подпись)

**Проверил:**

Пенин Андрей Семенович

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(отметка о выполнении)

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(подпись)

# Содержание

[**Содержание**](#_heading=h.3jbepr30actf) **2**

[**Введение**](#_heading=h.gjdgxs) **3**

[**Задание**](#_heading=h.1fob9te) **4**

[1. Этап 1: Установка и настройка IDS (Snort)](#_heading=h.k303886ij2nw) 5

[2. Этап 2. Установка и настройка системы контроля целостности (ICS)](#_heading=h.gde9cww4ynp) 7

[3. Этап 2. Установка и настройка системы контроля целостности (ICS)](#_heading=h.5ryqxtgjwxqr) 8

[**Заключение**](#_heading=h.tyjcwt) **11**

[**ПРИЛОЖЕНИЕ А**](#_heading=h.ylg7gfdem23c) **12**

# Введение

В условиях постоянных угроз информационной безопасности важным аспектом является защита информационных систем на разных уровнях. Настоящая работа направлена на практическое освоение трёх ключевых компонентов системы защиты информации:

* На первом этапе осуществляется установка и базовая настройка системы обнаружения вторжений (IDS) с использованием инструмента Suricata, предназначенного для анализа сетевого трафика и выявления подозрительной активности.
* Второй этап посвящён реализации системы контроля целостности (ICS), которая позволяет отслеживать изменения в критически важных файлах. В рамках работы используется аудит-демон auditd, встроенный в большинство Linux-систем.
* На третьем этапе разрабатывается модель системы аутентификации, обеспечивающая безопасное хранение паролей и поддержку двухфакторной аутентификации.

Каждый из этапов направлен на реализацию базовых механизмов обеспечения безопасности, которые в совокупности формируют комплексную систему защиты информации.

# Задание

It is necessary to install, configure and test advanced SPI tools from NSD on a virtual machine with the Windows operating system.. Such advanced tools include an intrusion detection tool and an integrity monitoring system. In addition, it is necessary to implement an authentication system model. The system must support storing the login/ password in a secure form and two-factor authentication (the method is at the discretion of the user). The practical work consists of 3 stages

**Stage 1. Installation and configuration of the Intrusion Detection System (IDS).** (up to 3 points)

* Install software for an intrusion detection system (for example, Snort, Suricata or an equivalent).
* Configure network/file monitoring settings to detect unauthorized access attempts.
* Test the system on artificially created scenarios (for example, making suspicious requests, using network scanners).

**Stage 2. Installation and configuration of the Integrity Control System (ICS)** (up to 3 points)

* Install a software solution for integrity monitoring (for example, AIDE, Tripwire or an equivalent).
* Configure the file system integrity check parameters (monitoring changes to certain files or directories).
* Create an artificial file modification scenario that can be detected by the integrity monitoring system.

**Stage 3. Development of an authentication system model.** (up to 4 points)

**System requirements:**

* Saving user logins and passwords in encrypted (and protected) form (the encryption method is at your discretion).
* Checks the correctness of the entered data at the entrance.
* Supports two-factor authentication (for example, using a code from an SMS or an application, a Q&A system, or other methods, at your discretion)

All this must be recorded in a report, and screenshots of all key work steps are required.

In addition, be prepared to demonstrate VMs on job defence.

**Ход работы**

## Этап 1: Установка и настройка IDS (Snort)

* Я использовал Ubuntu Server 20.10 вместо Windows

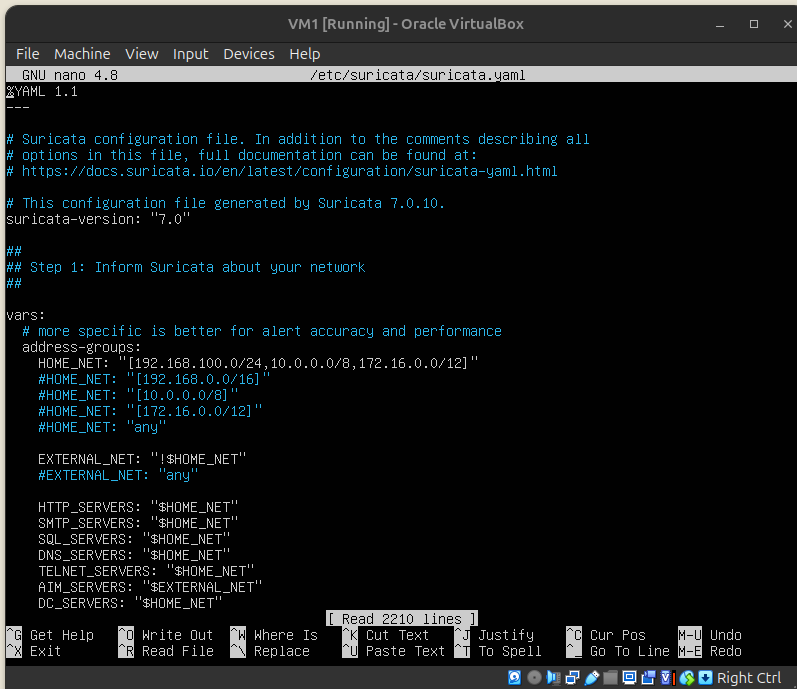


Рисунок 1 - Настройка Suricata

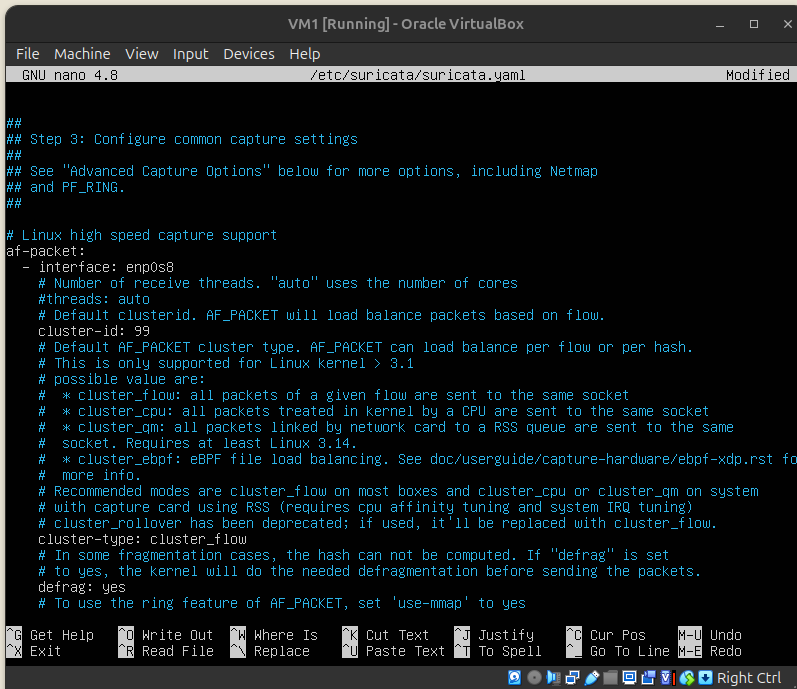


Рисунок 2 - Настройка Suricata

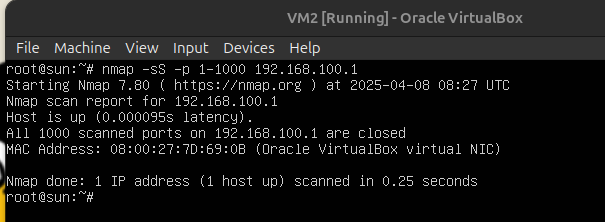


Рисунок 3 - Проверка

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

* Настройка системы контроля изменений в важных файлах (ICS – система проверки целостности), обеспечивающей своевременное обнаружение несанкционированных изменений в конфигурационных файлах, таких как /etc/hosts.

Инструменты, используемые:

* **auditd**: Демон аудита в Linux, отслеживает и записывает события доступа/изменения файлов.
* **auditctl**: Используется для добавления правил аудита.
* **ausearch**: Применяется для поиска и просмотра зарегистрированных событий.

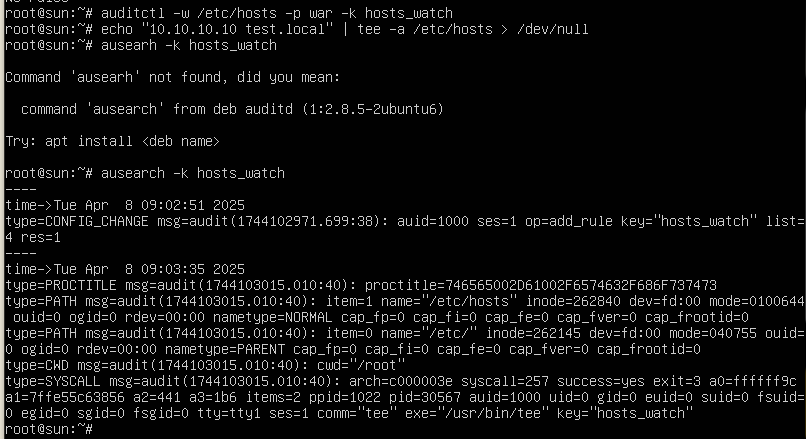


Рисунок 4 - Проверка

Результат:  
 Система точно зафиксировала событие изменения файла **/etc/hosts**.

Можно получить подробную информацию о времени, процессе и пользователе, выполнившем изменение.

Обеспечивается выполнение требований по контролю целостности системы.

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

Функциональность программы аутентификации (main.py)

Программа реализует систему аутентификации пользователей с базовыми и расширенными механизмами безопасности. Основные функции:

* Регистрация пользователей с безопасным хранением пароля, зашифрованного с помощью алгоритма bcrypt.
* Вход в систему с проверкой имени пользователя и пароля.
* Возможность включения двухфакторной аутентификации (2FA) с выбором метода: SMS или мобильное приложение.
* В случае включённой 2FA пользователь должен ввести код подтверждения для завершения входа.
* Все попытки входа (успешные и неуспешные) записываются в файл журнала login.log.

Программа выполнена в одном файле main.py и работает через интерфейс командной строки.

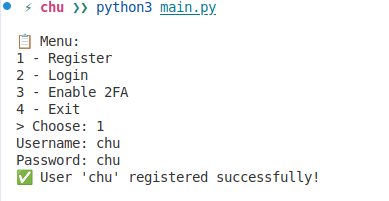


Рисунок 5 - Проверка

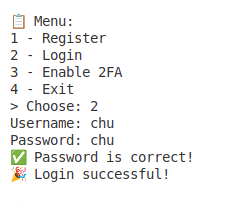


Рисунок 6 - Проверка

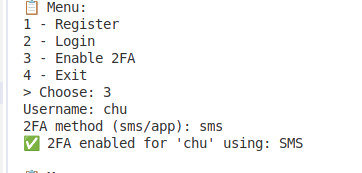


Рисунок 7 - Проверка

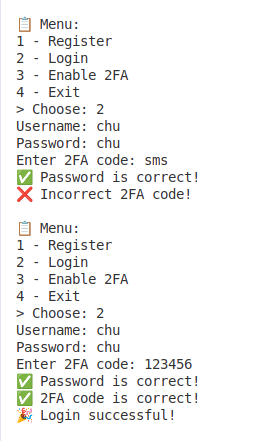


Рисунок 8 - Проверка

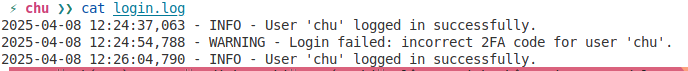


Рисунок 9 - file log

# Заключение

* В ходе выполненной работы были последовательно реализованы три ключевых компонента системы защиты информации: система обнаружения вторжений (IDS), система контроля целостности (ICS) и модель аутентификации с поддержкой двухфакторной проверки. Каждое из решений было успешно настроено и протестировано на примерах, имитирующих реальные сценарии угроз. Полученные результаты подтвердили эффективность базовых механизмов обеспечения безопасности в информационных системах.

# ПРИЛОЖЕНИЕ А

**Листинг А.1 – Код файла main.py**

import bcrypt

import logging

from datetime import datetime

# Cấu hình ghi log

logging.basicConfig(

filename="login.log",

level=logging.INFO,

format="%(asctime)s - %(levelname)s - %(message)s"

)

class AuthSystem:

def \_\_init\_\_(self):

self.user\_database = {}

def encrypt\_password(self, password):

salt = bcrypt.gensalt()

hashed = bcrypt.hashpw(password.encode(), salt)

return hashed

def register\_user(self, username, password):

if username in self.user\_database:

print("❌ Username already exists!")

return

self.user\_database[username] = {

"password": self.encrypt\_password(password),

"2fa\_enabled": False,

"2fa\_method": None

}

print(f"✅ User '{username}' registered successfully!")

def login(self, username, password):

if username not in self.user\_database:

print("❌ User not found!")

logging.warning(f"Login failed: user '{username}' not found.")

return False

stored\_hash = self.user\_database[username]["password"]

if bcrypt.checkpw(password.encode(), stored\_hash):

print("✅ Password is correct!")

return True

else:

print("❌ Incorrect password!")

logging.warning(f"Login failed: incorrect password for user '{username}'.")

return False

def enable\_2fa(self, username, method="sms"):

if username not in self.user\_database:

print("❌ User not found!")

return

self.user\_database[username]["2fa\_enabled"] = True

self.user\_database[username]["2fa\_method"] = method

print(f"✅ 2FA enabled for '{username}' using: {method.upper()}")

def verify\_2fa(self, username, code):

user = self.user\_database.get(username)

if not user:

print("❌ User not found!")

return False

if not user["2fa\_enabled"]:

print("⚠️ 2FA is not enabled for this user.")

return True

method = user["2fa\_method"]

if method == "sms":

correct\_code = "123456"

elif method == "app":

correct\_code = "654321"

else:

print("❌ Invalid 2FA method!")

return False

if code == correct\_code:

print("✅ 2FA code is correct!")

return True

else:

print("❌ Incorrect 2FA code!")

logging.warning(f"Login failed: incorrect 2FA code for user '{username}'.")

return False

def full\_login(self, username, password, code=None):

if not self.login(username, password):

return False

if self.user\_database[username]["2fa\_enabled"]:

if code is None:

print("⚠️ 2FA code required!")

return False

if not self.verify\_2fa(username, code):

return False

print("🎉 Login successful!")

logging.info(f"User '{username}' logged in successfully.")

return True

def menu(self):

while True:

print("\n📋 Menu:")

print("1 - Register")

print("2 - Login")

print("3 - Enable 2FA")

print("4 - Exit")

choice = input("> Choose: ")

if choice == "1":

username = input("Username: ")

password = input("Password: ")

self.register\_user(username, password)

elif choice == "2":

username = input("Username: ")

password = input("Password: ")

user = self.user\_database.get(username)

if user and user["2fa\_enabled"]:

code = input("Enter 2FA code: ")

self.full\_login(username, password, code)

else:

self.full\_login(username, password)

elif choice == "3":

username = input("Username: ")

method = input("2FA method (sms/app): ").strip().lower()

self.enable\_2fa(username, method)

elif choice == "4":

print("Goodbye!")

break

else:

print("❌ Invalid option.")

if \_\_name\_\_ == "\_\_main\_\_":

app = AuthSystem()

app.menu()