IntegrityMonitor или HashWatch

Что это такое?

Этот код на Python представляет собой простой мониторинговый инструмент, который следит за изменениями в файлах в указанных каталогах. Он вычисляет хэш-сумму каждого файла и сравнивает ее с сохраненной хэш-суммой в базе данных. Если хэши не совпадают, программа обнаруживает аномалию и отправляет уведомление в Telegram.

## Как это работает?

1. \*\*Конфигурация:\*\*

- Указываются каталоги, которые нужно отслеживать (`MONITORED\_DIRS`).

- Задается токен бота и идентификатор чата для отправки уведомлений в Telegram (`TELEGRAM\_BOT\_TOKEN` и `TELEGRAM\_CHAT\_ID`).

- Создается файл базы данных для хранения хэшей файлов (`DB\_FILE`).

2. \*\*Вычисление хэшей:\*\*

- Для каждого файла вычисляется хэш-сумма с использованием алгоритма SHA-256.

3. \*\*Обнаружение аномалий:\*\*

- Программа проходит по всем файлам в отслеживаемых каталогах.

- Сравнивает текущий хэш с сохраненным в базе данных.

- Если хэши не совпадают, добавляет аномалию в список.

4. \*\*Уведомления в Telegram:\*\*

- Если обнаружены аномалии, программа формирует сообщение и отправляет его в Telegram.

## Зачем это нужно?

Этот инструмент полезен для мониторинга целостности файлов на серверах, обнаружения внезапных изменений и быстрой реакции на потенциальные угрозы.

‘’’

import os

import hashlib

import logging

import requests

import sqlite3

# Конфигурация

MONITORED\_DIRS = ['/etc', '/usr/bin', '/var/www']

TELEGRAM\_BOT\_TOKEN = 'YOUR\_TELEGRAM\_BOT\_TOKEN'

TELEGRAM\_CHAT\_ID = 'YOUR\_TELEGRAM\_CHAT\_ID'

DB\_FILE = 'changes.db'

# Настройка журналирования

logging.basicConfig(filename='anomaly\_detector.log', level=logging.INFO)

def calculate\_hash(file\_path):

"""Вычисляет хэш-сумму файла"""

with open(file\_path, 'rb') as f:

return hashlib.sha256(f.read()).hexdigest()

def detect\_anomalies():

"""Обнаруживает аномалии в отслеживаемых каталогах"""

anomalies = []

changes = []

for directory in MONITORED\_DIRS:

for root, dirs, files in os.walk(directory):

for file in files:

file\_path = os.path.join(root, file)

try:

stored\_hash = get\_stored\_hash(file\_path)

current\_hash = calculate\_hash(file\_path)

if stored\_hash != current\_hash:

anomalies.append({

'file': file\_path,

'timestamp': os.path.getmtime(file\_path),

'hash': current\_hash

})

changes.append({

'file': file\_path,

'timestamp': os.path.getmtime(file\_path),

'old\_hash': stored\_hash,

'new\_hash': current\_hash

})

store\_hash(file\_path, current\_hash)

except FileNotFoundError:

logging.error(f"Файл не найден: {file\_path}")

if anomalies:

send\_notification(anomalies)

log\_changes(changes)

def get\_stored\_hash(file\_path):

"""Получает сохраненную хэш-сумму файла из базы данных"""

conn = sqlite3.connect(DB\_FILE)

c = conn.cursor()

c.execute("SELECT new\_hash FROM changes WHERE file = ?", (file\_path,))

result = c.fetchone()

conn.close()

return result[0] if result else None

def store\_hash(file\_path, hash\_value):

"""Сохраняет хэш-сумму файла в базу данных"""

conn = sqlite3.connect(DB\_FILE)

c = conn.cursor()

c.execute("""CREATE TABLE IF NOT EXISTS changes

(file TEXT, timestamp TEXT, old\_hash TEXT, new\_hash TEXT)""")

c.execute("INSERT INTO changes VALUES (?, ?, ?, ?)", (

file\_path, os.path.getmtime(file\_path), None, hash\_value

))

conn.commit()

conn.close()

def send\_notification(anomalies):

"""Отправляет уведомление о найденных аномалиях в Telegram чат"""

message = "\n".join([

f"Обнаружена аномалия: {anomaly['file']}",

f"Время: {anomaly['timestamp']}",

f"Хэш: {anomaly['hash']}"

for anomaly in anomalies

])

url = f"https://api.telegram.org/bot{TELEGRAM\_BOT\_TOKEN}/sendMessage"

data = {

"chat\_id": TELEGRAM\_CHAT\_ID,

"text": message

}

requests.post(url, data=data)

def log\_changes(changes):

"""Записывает изменения в базу данных"""

conn = sqlite3.connect(DB\_FILE)

c = conn.cursor()

c.execute('''CREATE TABLE IF NOT EXISTS changes

(file TEXT, timestamp TEXT, old\_hash TEXT, new\_hash TEXT)''')

for change in changes:

c.execute("INSERT INTO changes VALUES (?, ?, ?, ?)", (

change['file'], change['timestamp'], change['old\_hash'], change['new\_hash']

))

conn.commit()

conn.close()

def main():

"""Точка входа программы"""

detect\_anomalies()

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

main()

‘’’

Разберем этот код по шагам:

1. **Конфигурация:**
   * MONITORED\_DIRS: Список каталогов, которые нужно отслеживать.
   * TELEGRAM\_BOT\_TOKEN и TELEGRAM\_CHAT\_ID: Токен бота и идентификатор чата для отправки уведомлений в Telegram.
   * DB\_FILE: Имя файла базы данных для хранения хэшей файлов.
2. **Функция calculate\_hash(file\_path)**:  
   * Вычисляет хэш-сумму файла, используя алгоритм SHA-256.
3. **Функция detect\_anomalies()**:  
   * Проходит по каждому файлу в отслеживаемых каталогах.
   * Сравнивает хэш текущего файла с сохраненным хэшем в базе данных.
   * Если хэши не совпадают, добавляет аномалию в список и обновляет хэш в базе данных.
4. **Функции get\_stored\_hash(file\_path) и store\_hash(file\_path, hash\_value)**:  
   * Получают и сохраняют хэш-сумму файла в базе данных.
5. **Функция send\_notification(anomalies)**:  
   * Формирует сообщение о найденных аномалиях.
   * Отправляет уведомление в Telegram.
6. **Функция log\_changes(changes)**:  
   * Записывает изменения в базу данных.
7. **Точка входа main()**:  
   * Запускает обнаружение аномалий.

Этот код предназначен для мониторинга файлов в указанных каталогах, обнаружения изменений и отправки уведомлений в Telegram.