Министерство науки и высшего образования Российской Федерации

**Муромский институт (филиал)**

федерального государственного бюджетного образовательного учреждения высшего образования

**«Владимирский государственный университет   
имени Александра Григорьевича и Николая Григорьевича Столетовых»**

(МИВлГУ)

Факультет ФИТР

Кафедра ФПМ

КУРСОВАЯ

РАБОТА

по Технологии и методам программирования

(наименование дисциплины)

Тема Реализация системы оповещения о подозрительной активности в реальном времени.

Руководитель

Соколов М.С.

(оценка) (фамилия, инициалы)

Члены комиссии

(подпись) (дата)

Студент ИБ-123

(подпись) (Ф.И.О) (группа)

Хохлов Д.А.

(подпись) (Ф.И.О) (фамилия, инициалы)

(подпись) (Ф.И.О) (подпись) (дата)

Муром 2025

В курсовой работе отражена реализация системы оповещения о подозрительной активности в реальном времени. Разработано приложение на основе MFC (Microsoft Foundation Classes) для операционной системы Windows. Программа представляет особый класс ПО для безопасности внутри компьютерной системы. Программа позволяет пользователю определить количество опасных ошибок, предупреждений и информационных сообщений, которые сохраняются внешней системой в качестве лог-файлов (текстовый файл, куда автоматически записывается важная информация о работе системы или программы). На основе этих лог-файлов программа оценивает, насколько большая опасность угрожает системе и сообщает об этом пользователю.

Оглавление

[1 Анализ технического задания 6](#_Toc20982)

[2 Разработка программы 11](#_Toc2577)

[3 Тестирование программного продукта 19](#_Toc15286)

[4 Руководство пользователя 24](#_Toc6165)

[Заключение 26](#_Toc29054)

[Список использованной литературы 27](#_Toc32339)

[Приложение А Графический материал 28](#_Toc3294)

Введение

В современном мире информационные технологии играют ключевую роль в функционировании практически всех сфер деятельности. С увеличением сложности и масштабов информационных систем растёт и объем данных, которые они генерируют. Одним из основных источников такой информации являются файлы трассировки OC Linux (лог), которые содержат записи о событиях, ошибках, предупреждениях и других действиях, происходящих в системе. Эти данные представляют собой ценный ресурс для анализа работы системы, выявления аномалий и предотвращения потенциальных угроз. Однако ручной анализ файлов трассировки становится все менее эффективным из-за их огромного объёма и высокой скорости генерации. В связи с этим возникает необходимость в автоматизации процессов анализа и мониторинга файла трассировки для оперативного выявления подозрительных активностей.

Актуальность разработки системы оповещения о подозрительных активностях в реальном времени обусловлена ростом числа компьютерных угроз и необходимостью своевременного реагирования на инциденты. Современные злоумышленники используют сложные методы атак, которые могут оставаться незамеченными в течение длительного времени, если система мониторинга не способна оперативно выявлять аномалии. Автоматизированная система, анализирующая лог в реальном времени, позволяет минимизировать время обнаружения угроз и снизить потенциальный ущерб. Кроме того, такая система может быть полезна не только для обеспечения безопасности, но и для повышения общей стабильности работы приложений, так как она способна выявлять ошибки и сбои на ранних этапах.

Целью данной курсовой работы является разработка системы, которая способна анализировать лог в реальном времени, классифицировать события по уровням подозрительности (например, низкий уровень, средний, высокий) и генерировать оповещения по необходимости. Для достижения этой цели необходимо решить ряд задач, включая анализ существующих методов обработки файлов трассировки, разработку алгоритмов классификации событий, проектирование архитектуры системы и её тестирование на реальных данных.

Основными задачами работы являются:

1. Исследование существующих подходов к анализу файлов трассировки и выявлению подозрительных активностей.

2. Разработка алгоритма для классификации событий по уровням подозрительности на основе анализа файла трассировки типа strace в OC Linux.

3. Проектирование системы, способной обрабатывать файлы трассировки в реальном времени и генерировать оповещения.

4. Реализация прототипа системы и его тестирование на реальных данных для оценки эффективности.

В процессе работы будут использованы методы анализа данных и разработки программного обеспечения.

Практическая значимость работы заключается в создании инструмента, который может быть внедрён в существующие информационные системы для повышения их безопасности и устойчивости. Разработанная система позволит автоматизировать процесс выявления подозрительных активностей, снизить нагрузку на администраторов и специалистов по безопасности, а также повысить оперативность реагирования на инциденты.

Следует отметить, что разработка системы оповещения о подозрительных активностях в реальном времени является важным шагом в обеспечении безопасности современных информационных систем. Результаты работы могут быть использованы как в коммерческих, так и в научных целях, а также послужить основой для дальнейших исследований в области анализа лог-файлов и автоматизации процессов мониторинга.

1 Анализ технического задания

В данной работе поставлена задача разработки программы, уведомляющей пользователя о подозрительной активности в реальном времени. В ходе подготовки и принятия решений по проекту было решено, что сама система будет основана по принципу разделения обязанностей, а именно – программа будет производить работу по уведомлению о подозрительной активности по уже приготовленным лог-файлам системы коллег.

Основная мысль заключается в обработке файлов с расширением «.log», далее – по готовности пользователя, отображение сведений о состоянии системы целиком, количестве ошибок, предупреждений, а так же вывод на экран структурированных логов из файла. Все это нужно для того, чтобы пользователь системы смог увидеть насколько подозрительно ведёт себя система, и в дальнейшем произвести некоторые манипуляции для устранения уязвимых мест с помощью другого ПО.

Программа будет состоять из следующих логических действий:

* открытие файла логов или папки с логами.
* чтение файла и запись данных в некий объект определённой подготовленной структуры.
* просмотр объекта структуры приложения и обновление его при необходимости (если пользователь будет выбирать различные элементы фильтрации.)[1]
* фильтрация данных объекта при необходимости.
* по готовности пользователя (нажав кнопку с надписью «Проверка системы»), открывается диалоговое окно с детальной информацией по лог-файлу.
* по завершении чтения и подсчёта необходимой информации (количество ошибок и предупреждений), выдаётся вердикт по системе, которая содержит данные.
* если пользователю нужно сохранить информацию о текущей сессии, то это можно доступно сделать при помощи меню «Журналирование» с выбором кнопки «Сохранить отчёт» или «Сохранить отчёт с фильтрацией».

По завершении всех пяти логических блоков, программа пришла к своей цели.

Поставленная задача требует определённых навыков работы с написанием приложений на языке C++ и MFC (Microsoft Foundation Classes). Несколько подумав над тем, какую версию C++ можно использовать, а так же какая кодировка приложения будет, а это очень важно, ведь сейчас практически все используемые файлы используют кодировку типа «UTF-8», было решено остановиться на данных настройках:

версия языка - C++20

тип кодировки – «Unicode», для работы с различными кодировками.

В связи с этими вводными, а так же в связи с тем, что код приложения должен быть написан грамотно, приложение так же в своей основе будет использовать встроенные классы MFC для работы с текстом, диалогами, файловой структурой и прочим, так как это позволяет приложению быть достаточно простым при его разработке.

Программа будет построена по модульному принципу, что позволит легко расширять её функциональность и поддерживать код. Основные модули программы:

Модуль работы с файловой системой:

Отвечает за открытие и чтение файла трассировки, а также за навигацию по папкам. Этот модуль будет использовать стандартные классы MFC, такие как CFile и CFileDialog[5-7], для работы с файлами и диалогами выбора.

Модуль парсинга данных:

Отвечает за извлечение данных из файла трассировки и их преобразование в структурированный формат. В зависимости от формата файла, этот модуль будет использовать различные алгоритмы.

Модуль фильтрации и анализа данных:

Отвечает за применение фильтров и выполнение аналитических операций (подсчёт вероятности подозрительности). Этот модуль будет взаимодействовать с пользовательским интерфейсом для получения параметров фильтрации и отображения результатов.

Модуль пользовательского интерфейса:

Отвечает за отображение данных и взаимодействие с пользователем. Этот модуль будет использовать классы MFC, такие как CListCtrl для таблиц, CDialog для диалоговых окон и CButton для кнопок.[5-7]

Модуль сохранения отчёта:

Отвечает за сохранение результатов анализа в файл. Этот модуль будет поддерживать формат вывода «.log» (по умолчанию - файл «NotOfS.log»).

Безопасность и надежность программы

При разработке программы необходимо уделить внимание вопросам безопасности и надёжности:

Обработка ошибок:

Программа должна корректно обрабатывать ошибки, такие как отсутствие файла, повреждение данных или нехватка памяти. Для этого использование механизмы исключений C++ (try/catch)[2, 4] и уведомление пользователя о возникших проблемах.

Защита от переполнения буфера:

При работе с большими объёмами данных необходимо обеспечить защиту от переполнения буфера и других уязвимостей. Это особенно важно при обработке данных и работе с файловой системой. Это достигается с помощью использования стандартных модулей из библиотеки STL, как пример, vector, map и других контейнеров[2], работы с уже написанными алгоритмами из модуля «algorythm» а также передачи глобальных переменных по ссылкам, а не по значениям.

Отчёт действий программы:

Для упрощения работы с обработанным файлом трассировки, программа имеет возможность сохранять обработанную информацию в пользовательский файл. Это реализуется с помощью встроенного механизма журналирования.

Тестирование и отладка:

Перед выпуском программы необходимо провести тщательное тестирование всех модулей. Программа будет тестироваться вручную при использовании разного рода файлов.

Пользовательский интерфейс:

Пользовательский интерфейс программы должен быть интуитивно понятным и удобным в использовании. Основные элементы интерфейса:

Главное окно:

Содержит меню для выбора файлов и папки, сохранения данных о сессии. В окне располагается элемент вывода пути к логу. После – кнопка, инициирующая проверку. Далее – блок с элементами фильтрации по дате, процессу, командам strace, Диалоговое окно с результатами анализа:

Отображает детальную информацию о файле трассировки, включая количество подозрений высокого, среднего и низкого уровня.

Окно открытия файлов:

Предоставляет пользователю возможность выбрать нужный файл или папку со своего компьютера под системой Windows.

Окно сохранения данных:

Предоставляет пользователю возможность выбрать формат и место сохранения результатов анализа. Это окно содержит дополнительные параметры, такие как начальное имя файла, его тип.

Окно уведомлений:

Реализуется с помощью диалогового окна уведомления для того, чтобы программа работала даже под ОС Linux в программе «Wine».

Разработка программы для анализа лог-файлов на основе MFC и C++ представляет собой комплексную задачу, которая требует тщательной проработки требований:

Использование C++20

Использование MFC v143

Использование многобитовой кодировки «Unicode»

2 Разработка программы

Как уже и говорилось ранее, основной целью программы будет определение подозрительности системы путём чтения файла трассировки команды «strace». Учитывая это, реализация программы подразумевает создание приложения, способного читать файл типа .log (так как предоставляемый файл имеет такое расширение), приводить сложную структуру «strace.log» в человеко-читаемый вид, уметь фильтровать данные по полям «Не показывать ошибки», «Не показывать системную информацию», «Поиск по процессу», «Поиск по времени», «Команда strace», «ID процесса».

Всё же, основной частью программы является предоставление читаемых данных о происходящих процессах во время работы «strace» и давать оценку о подозрительности действий внутри системы. Такой подход к визуализации информации позволит пользователю в случае чего прибегнуть к своим методам защиты системы или сообщить о подозрительных действиях администратору своей корпоративной сети, если она таковой является.

Сама программа написана на языке C++20, MFC(Microsoft Foundation Classes) v143. Программа работает на многобитовой кодировке Unicode, который так же является UTF-16, что означает кодировку, в которой на 1 символ отводится 16 бит памяти. Однако файл «strace.log», как и множество других файлов, реализован в кодировке UTF-8, что является первой проблемой, которую необходимо было решить, прежде чем приступить к написанию остальной логики, отвечающей за фильтрацию, отображение в виде таблицы и т.д. Эту проблему решают 2 функции, которые необходимы в данном проекте. Первая функция - std::wstring UTF8ToUTF16(const std::string& utf8). Эта функция отвечает за перевод сисволов типа UTF-8 в тип UTF-16. Вторая функция - std::wstring UTF16ToUTF8(const std::string& utf16), эта же функция - наоборот - переводит символы из типа UTF-16 в тип UTF-8, нужная она во-первых для того, чтобы получившиеся человеко-читаемые строки можно было сохранить в файл журналирования программы для удобства пользователей или других разработчиков.

Следующая часть приложения - чтение самого файла. На данном этапе была реализована структура данных, которая будет принимать в себя нужные данные, которые по итогу и отобразятся в диалоговом окне приложения.

Структура[1] LogEntry вынесена в отдельный заголовочный файл и представляет собой следующий участок кода программы:

struct LogEntry {

int PID;

CTime time;

std::wstring process;

std::wstring sysMsg;

std::wstring details;

};

,где:

PID - поле, принимающее целочисленное значение идентификатора процесса, вызываемого данной строкой;

Time - поле, принимающее значение встроенного типа(класса) CTime, которое показывает, в какое время совершился процесс или подпроцесс.

Process - поле, принимающее строку, которая отвечает за процесс.

sysMsg - поле, принимающее строку, отвечающую за сообщение программы о подозрительности процесса или подпроцесса (действия, совершаемого внутри системы.)

Details - поле, которое принимает строку, которая сама является дополнением к основной информации. В случае этой программы она будет включать в себя название выполняемой команды strace[], и переданные ей параметры.

Функции поиска и фильтрации организованы через функции «слушателей» событий и переменных, которые были заданы при помощи мастера классов в конструкторе формы MFC. Основные функции:

void CSecurityDlg::OnSearchingByDate()

Данная функция реализована для фильтрации исходных строк по времени выполнения процедуры.

void CSecurityDlg::OnSearchingByProc()

Данная функция «слушает» элемент «чекбокс», который отвечает за активацию поля ввода имени процесса.

void CSecurityDlg::OnChoiceDateFrom(NMHDR\* pNMHDR, LRESULT\* pResult)

Это функция для выбора времени начала выполнения процесса.

void CSecurityDlg::OnChoiceDateTill(NMHDR\* pNMHDR, LRESULT\* pResult)

Функция выбора времени конца выполнения процесса или процедуры. Например, пользователь выбрал время по титулу «С» «18:05:04» и время по титулу «до» «18:15:06». Тогда, программа будет выбирать только те строки логов, где значения по столбцу «Время» будет большее либо равное выбранного времени по титулу «С» и меньшее либо равное выбранного времени по титулу «до».

void CSecurityDlg::OnChangeProcName()

Функция написана для поиска входящих строк логов только по имени процесса. Пример: пользователь вводит в поле значение «malware.sh», после нажатия кнопки «Применить» в буфер программы будут сохранены только строки логов, где значение столбца «Процесс» равно «malware.sh».

void CSecurityDlg::OnHideInfo()

Функция чекбокса, отвечающего за скрытие служебной(системной информации).

void CSecurityDlg::OnHideWarns()

Функция чекбокса, отвечающего за скрытие ошибок в системе, которые зафиксировала команда strace

void filterByParameters(const AppState currentState, const std::vector<LogEntry>& logs, std::vector<LogEntry>& copied\_logs)

Данная функция собирает все состояния чекбоксов, которые в итоге приходят в объект структуры AppState. Функция принимает в качестве параметров неизменяемый объект структуры AppState, двусвязный список vector STL объекта структуры LogEntry по ссылке для изменения этого объекта вне функции. Третий параметр отвечает за копию логов для передачи уже измененных логов во второй диалог.

void CSecurityDlg::OnApplyFilter()

Назначение: применения всех фильтров к строкам исходных логов. Функция возвращает тип void.

В предоставленном файле реализовано несколько функций, каждая из которых выполняет определенную задачу. Ниже представлено их подробное описание:

clearInfoListInputs()  
   Назначение: Обнуляет значения глобальных переменных cpl\_l, cpl\_m и cpl\_h. Данная функция предназначена для сброса данных между операциями.    
   Параметры: Функция не принимает никаких аргументов.

WStoS(const std::wstring& wstr)  
   Назначение: Преобразует строку, представленную в формате UTF-16 (std::wstring), в строку UTF-8 (std::string).    
   Параметры: Принимает один аргумент типа const std::wstring.    
   Возвращаемое значение: Строка типа std::string.

splitLine(const std::wstring& line)  
   Назначение: Разделяет входную строку на отдельные части и анализирует содержащиеся в ней команды. Функция также используется для извлечения PID процесса и его имени.    
   Параметры: Один аргумент типа const std::wstring.    
   Возвращаемое значение: Вектор строк типа std::vector, состоящий из пяти элементов.

filterByParameters(const AppState currentState, const std::vector<Log>& logs, std::vector<Log>& copied\_logs)  
   Назначение: Осуществляет фильтрацию массива логов в зависимости от текущего состояния приложения (AppState) и заданных параметров.    
   Параметры:    
   - currentState: Текущее состояние приложения.    
   - logs: Исходный массив логов.    
   - copied\_logs: Массив логов, в который будут записаны результаты фильтрации (передаётся по ссылке).    
   Возвращаемое значение: Изменённый массив copied\_logs (через передачу по ссылке).

readFile(std::ifstream& file, std::vector<Log>& logs)  
   Назначение: Читает содержимое файла с логами построчно. Каждая строка преобразуется из формата UTF-8 в UTF-16, после чего добавляется в массив logs.    
   Параметры:    
   - file: Поток файла, предназначенный для чтения.    
   - logs: Массив логов, в который будет производиться запись (передаётся по ссылке).    
   Возвращаемое значение: Изменённый массив logs (через передачу по ссылке).

UTF8ToUTF16(const std::string& utf8)  
   Назначение: Преобразует строку, представленную в формате UTF-8 (std::string), в строку формата UTF-16 (std::wstring).    
   Параметры: Один аргумент типа const std::string.    
   Возвращаемое значение: Строка типа std::wstring.

WStringToCTime(std::wstring& timeStr, CTime& outTime)  
   Назначение: Конвертирует строку, содержащую время, в объект типа CTime.    
   Параметры:    
   - timeStr: Входная строка, представляющая время.    
   - outTime: Объект типа CTime, в который будет записано преобразованное время (передаётся по ссылке).    
   Возвращаемое значение: Логическое значение, указывающее на успешность выполнения операции.

CTimeToWString(const CTime& time)  
   Назначение: Преобразует объект типа CTime в строку формата UTF-16 (std::wstring) с заданным форматом времени.    
   Параметры: Один аргумент типа const CTime.    
   Возвращаемое значение: Строка типа std::wstring.

calculateSuspicionPercentage(const int& low, const int& medium, const int& high)  
   Назначение: Вычисляет процент подозрительности на основе количества записей низкого, среднего и высокого уровня риска.    
   Параметры:    
   - low: Количество записей низкого уровня риска.    
   - medium: Количество записей среднего уровня риска.    
   - high: Количество записей высокого уровня риска.    
   Возвращаемое значение: Процент подозрительности в виде значения типа double.

Рассмотрим глобальные переменные, определённые в файле Globals.cpp:

* std::vector<LogEntry> logs

Назначение: Данная переменная предназначена для хранения всех записей логов.

Тип данных: Вектор объектов типа LogEntry.

* std::vector<LogEntry> copied\_logs

Назначение: Эта переменная используется для хранения копий записей логов, которые могут быть подвергнуты фильтрации.

Тип данных: Вектор объектов типа LogEntry.

* AppState state

Назначение: Переменная представляет текущее состояние приложения.

Тип данных: Перечисление типа AppState.

* AppState recentState

Назначение: В данной переменной сохраняется последнее зафиксированное состояние приложения.

Тип данных: Перечисление типа AppState.

* CString logPath

Назначение: Хранит путь к файлу, в который записываются логи.

Тип данных: Строковая переменная типа CString.

int cpl\_l, int cpl\_m, int cpl\_h

Назначение: Эти переменные предназначены для подсчета количества записей логов с различными уровнями риска:

* cpl\_l: количество логов с низким уровнем риска.
* cpl\_m: количество логов со средним уровнем риска.
* cpl\_h: количество логов с высоким уровнем риска.

Тип данных: Переменные типа int.

* std::map<std::wstring, std::wstring> processes

Назначение: Данная отображение хранит пары, где ключом является идентификатор процесса, а значением — его описание.

Тип данных: Ассоциативный контейнер типа std::map, где ключами и значениями являются строки типа std::wstring.

std::map<std::wstring, CommandRisk> commandRisk

Назначение: Переменная предназначена для хранения информации о командах и их описании с указанием уровня риска.

Тип данных: Ассоциативный контейнер типа std::map, где ключами являются строки типа std::wstring, а значениями — объекты типа CommandRisk.[5-7]

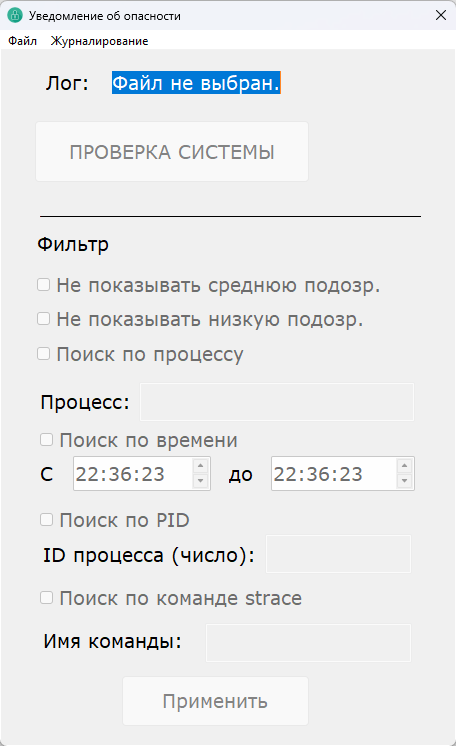
3 Тестирование программного продукта

Открыв данное приложение, мы видим стандартный пользовательский интерфейс windows. Выделим три основных блока:

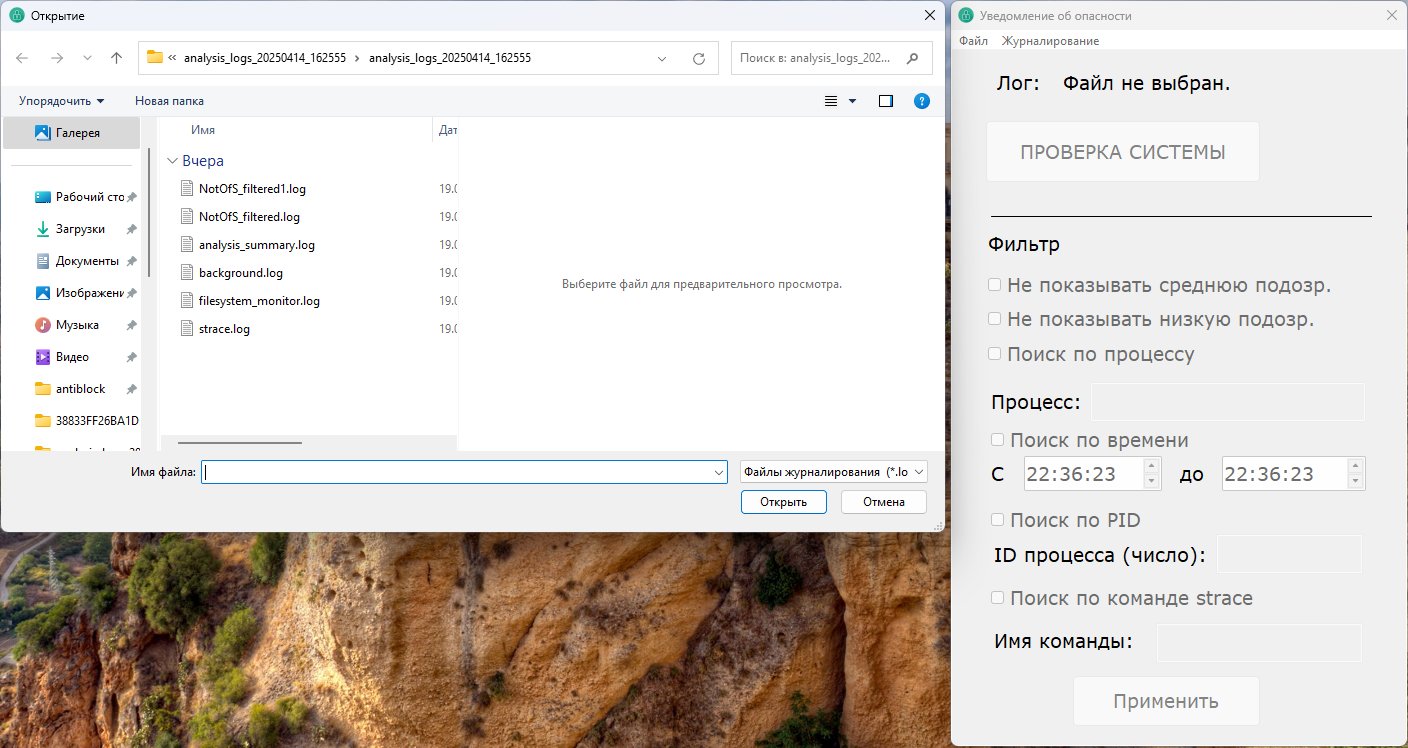
- меню приложения.

- поле, покзаывающее расположение файла журналирования и кнопка «Проверка системы».

- фильтрация выводимых логов.

Рисунок 1. Главное окно приложения

Как можно заметить, практически все элементы являются неактивными до того момента, пока не будет выбран файл журналирования.

Рисунок 2. Выбор файла журналирования

Следует отметить, что не все элементы становятся активными после выбора файла. Возможность активирования элементов реализована для чекбоксов, которые отвечают за различные поля ввода, которые без выбора данного фильтра не станут активными. Это дает пользователю возможность точно понять, фильтрация по каки критериям его интересует. Например, чтобы активировать фильтр по времени или команде, пользователь сначала должен отметить чекбокс «Поиск по времени» или «Поиск по команде strace» соответственно.

Строгая выборка позволяет реализовать программу с наилучшей структурностью кода, а так же предотвращение случайных нажатий клавиш, которые могут внести ненужные данные в поля поиска, что может ввести пользователя в заблуждение.

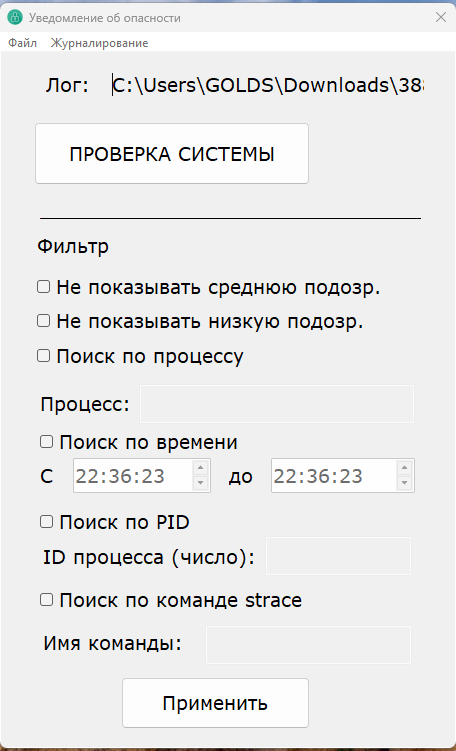
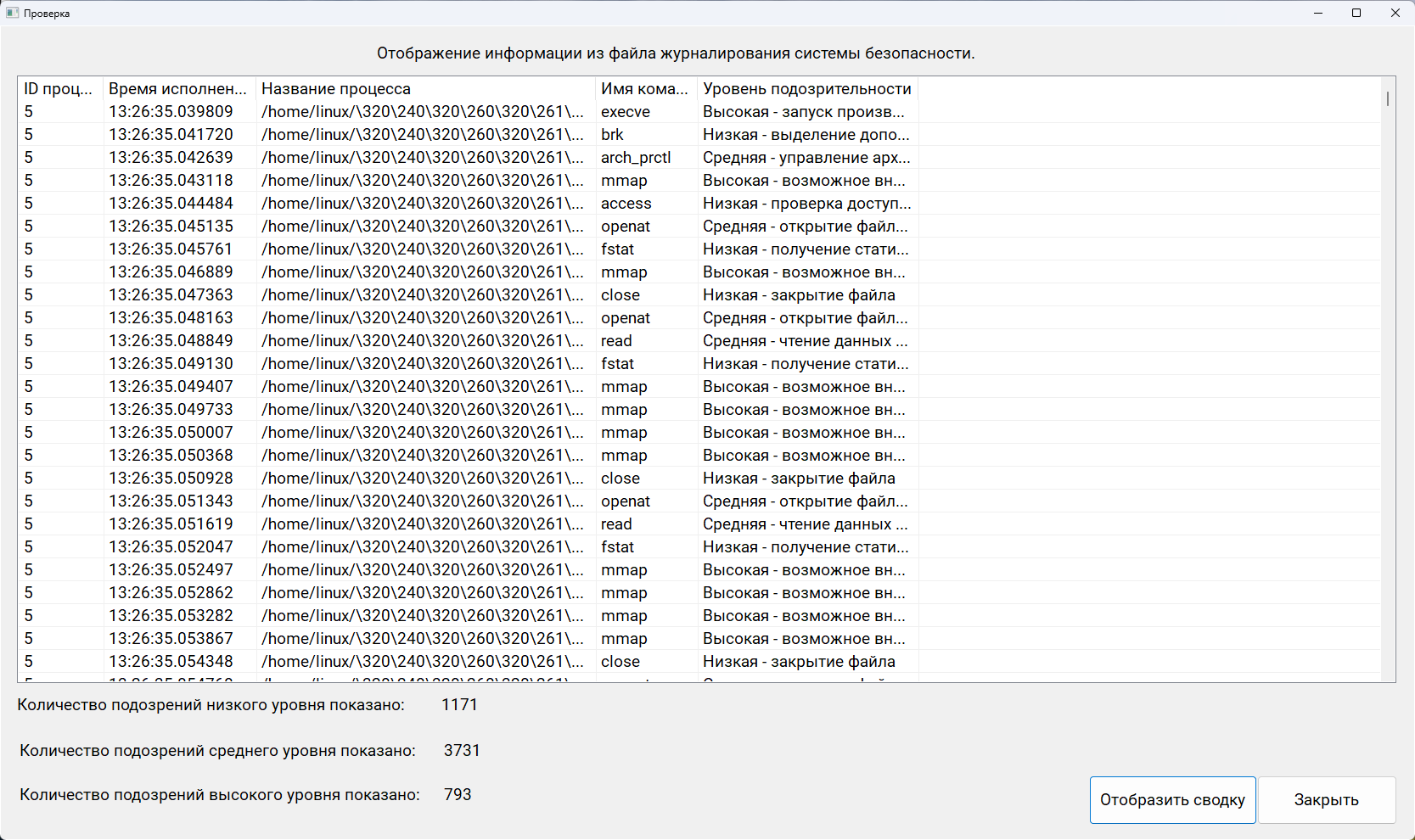


Рисунок 3. Состояние активности элементов.

Рисунок 4. Окно отображения логов

Приложение примечательно тем, что оно подсчитывает процент подозрительности системы по обычной формуле:

где h - это количество отображённых строк с высокой подозрительностью.

m - количество строк со средней подозрительностью.

l - количество строк с низкой подозрительностью.

А их множители - это соответствующие веса уровней подозрительности, где hw = 1, mw = 0.5, lw = 0.2, total - количество всех обработанных записей.

Отметим, что в расчеты идут именно отображенные на экране пользователя строки, что поволяет видеть процентное количество подозрительности пр разных настройках фильтрации.

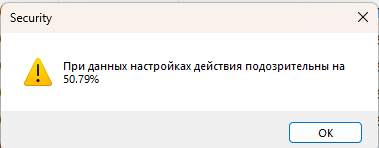


Рисунок 5. Окно уведомления об уровне подозрительности

Для того, чтобы пользователь мог в любой момент увидеть без приложения выведенные в протокол системных вызовов в приложении, есть две функции в меню приложения: «сохранить» и «сохранить отфильтрованные».

«Сохранить отчёт» - сохраняет обработанные данные файла системных вызовов, даже скрытые при помощи фильтрации.(По умолчанию сохраняется в файл «NotOfS.log»)

«Сохранить отчёт с фильтрацией» - сохраняет только те данные файла системных вызовов, которые были выведены в дочернее окно приложения. (По умолчанию сохраняется в файл «NotOfS\_filtered.log»). Такое простое разделение функций призвано ускорить поиск нужных вам данных, если нужны только фильтрованные данные и прочитать все строки для более полной «картины» о действиях в системе.

4 Руководство пользователя

Для начала стоит выполнить желаемые системные требования, хотя программа и не является слишком ресурсоёмкой.

Операционная система:

Windows 7 SP1 или более поздняя версия (рекомендуется использовать Windows 10 или Windows 11 для оптимальной производительности и поддержки новых API).

Требования к среде выполнения:

Установленный компонент MFC в системе (обычно входит в состав Visual C++Redistributable).

Поддержка стандартов C++20, если они используются в приложении (требуется соответствующий компилятор MSVC).

Разрешение экрана:

Минимум 1280x720 для отображения элементов графического интерфейса без сжатия или деформации.

Аппаратные требования:

Процессор:

Минимум двухъядерный процессор с тактовой частотой 1 ГГц (например, «Intel Core i3»).

Для многозадачной среды работы рекомендуется использовать процессор с 64-разрядной архитектурой.

Оперативная память:

Минимум 2 ГБ RAM, учитывая потребление 17 МБ в простое.

Рекомендуется минимум 4 ГБ RAM для нормальной работы операционной системы и других приложений.

ПЗУ:

Минимум 1 ГБ свободного пространства для установки приложения и сопутствующих файлов.

Рекомендуется использовать SSD-диск для более быстрой загрузки программы.

Видеокарта:

Интегрированная видеокарта (например, «Intel UHD Graphics»), достаточная для базового графического интерфейса.

Следующие действия направлены на корректную работу с программой:

Открыть приложение

Выбрать файл «strace.log» или файл с другим названием, но главное, чтобы структура файла показывала была следующей «PID TIME COMMAND». Это минимальная структура файла системных вызовов, которую программа обработает правильно.

По необходимости - выбрать нужные элементы фильтрации и обязательно нажать на кнопку «Применить», иначе изменения не применятся.

Для того чтобы увидеть результат обработки, нажать на кнопку «ПРОВЕРИТЬ СИСТЕМУ» в верхней части приложения.

Чтобы увидеть сводку по уровню подозрительности, нажать на кнопку «Отобразить сводку».

Если нужно сохранить все обработанные данные, - нажать на кнопку меню «Журналирование», далее - «Сохранить отчёт». После чего будет открыто окно сохранения, если же наоборот нужно сохранить только отфильтрованные значения, - нажать «Сохранить отчёт с фильтрацией». В противном случае результат обработки нигде не сохранится.

Заключение

В ходе данной курсовой работы были углублены знания, полученные в ходе лекционных и практических занятий по технологиям и методам программирования. Так же, разрабатывая работу, были получены и новые теоретические и практические знания о языке программирования C++20, а так же о библиотеке MFC. Были получены навыки по формализации, реализации и отладке программы. Было реализована программа на языке C++20 с помощью библиотеки для создания графических интерфейсов MFC (Microsoft Foundation Classes). Внутри приложения есть поддержка работы с файлами внутри операционной системы Windows, а так же работа со строками формата UTF-8. Приложение представлено в форме стандартного приложения с графическим интерфейсом.

Список использованной литературы

1. Лафоре Р. Объектно-ориентированное программирование в C++. — СПб.: Питер, 2022. — 928 с.
2. Эккель Б. Философия C++. Введение в стандартный C++. — М.: Диалектика, 2021. — 672 с.  
    3. Майерс С. Эффективное использование C++. 55 верных советов улучшить структуру и код ваших программ. — М.: Вильямс, 2020. — 320 с.  
    4. Лакос Дж., Ромео В., Хлебников Р., Мередит А. Современный C++: безопасное использование. — М.: Вильямс, 2021. — 688 с.  
    5. Сидорина Т.В. MFC и MS Visual Studio C++. Разработка приложений. — М.: БХВ-Петербург, 2019. — 480 с.  
    6. Сидорина Т.В. Самоучитель Microsoft Visual Studio C++ и MFC. — М.: БХВ-Петербург, 2020. — 512 с.  
    7. Литвиненко Н.А. Программирование WinAPI-приложений на C++. — М.: Наука и Техника, 2020. — 320 с.

**Приложение А**

**(справочное)**

Графический материал

Структура работы приложения