МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева»

(Самарский университет)

Институт информатики, математики и электроники

Факультет информатики

Кафедра геоинформатики и информационной безопасности

КУРСОВОЙ ПРОЕКТ ПО ДИСЦИПЛИНЕ

«Инженерия программного обеспечения»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

«Анализатор программ, написанных на С++, на уязвимости»

|  |  |
| --- | --- |
| Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись)* | Н.М. Фролов |
|  |  |
| Руководитель работы \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  *(подпись)* | В.А. Бердников |
|  |  |

САМАРА 2022

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования «Самарский национальный исследовательский университет имени академика С.П. Королева»

(Самарский университет)

Институт информатики, математики и электроники

Факультет информатики

Кафедра геоинформатики и информационной безопасности

**ЗАДАНИЕ НА КУРСОВОЙ ПРОЕКТ**

Студенту ***Фролову Никите Максимовичу*** группы 6412-100503D

Тема проекта: ***«*Анализатор программ, написанных на С++, на уязвимости*»***

|  |  |  |
| --- | --- | --- |
| Планируемые результаты освоения образовательной программы (компетенции) | Планируемые результаты | Содержание задания |
| ПК-5 | **Знать:**  Этапы оценки и управления рисками, современные методики оценки рисков, смысловое содержание численных показателей оценок рисков.  **Уметь:**  определять необходимую и достаточную совокупность средств защиты информации, организационных мер направленных на снижение рисков ИБ, разработать архитектуру СИБ, направленную на снижение рисков информационной безопасности автоматизированных систем.  **Владеть:**  навыками определения приемлемого для организации уровня риска; идентификации, анализа и оценки рисков; ранжирование рисков; принятия решения по рискам и разработка плана реагирования на риски; реализации мероприятий и оценки эффективности реализованных мер. | Знать способы оценки рисков при реализации основных функций программы и смысловое содержание численных показателей этих оценок. Уметь определять средства защиты программного обеспечения. Владеть навыками определения приемлемого уровня риска для организации, принятия решения по рискам и разработка плана реагирования на риски. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Дата выдачи задания | 8 февраля 2022 г. | | | |
| Срок представления на кафедру пояснительной записки | | |  | |
| Руководитель курсового проекта профессор кафедры ГИиИБ, д.т.н. | |  | | В.А.Бердников |
|  | | *(подпись)* | |  |
| Задание принял к исполнению  студент группы № 6412-100503D | |  | | Н.М. Фролов |
|  | | *(подпись)* | |  |

содержание

[РЕФЕРАТ 6](#_Toc104843872)

[распределение обязанностей 8](#_Toc104843873)

[ВВЕДЕНИЕ 9](#_Toc104843874)

[1 АНАЛИЗ СУЩЕСТВУЮЩЕГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ОБНАРУЖЕНИЯ УЯЗВИМОСТЕЙ В ПРОГРАММНОМ ОБЕСПЕЧЕНИИ НА C++ 10](#_Toc104843875)

[2 КРАТКИЙ ОБЗОР АНАЛИЗИРУЕМХ УЯЗВИМОСТЕЙ 12](#_Toc104843876)

[2.1 Описание уязвимостей 12](#_Toc104843877)

[2.1.1 Переполнение буфера 12](#_Toc104843878)

[2.1.2 Ошибка форматной строки 13](#_Toc104843879)

[2.1.3 Внедрение команд 13](#_Toc104843880)

[2.1.4 Внедрение SQL команд 13](#_Toc104843881)

[2.1.5 Пренебрежение обработкой ошибок //TODO 14](#_Toc104843882)

[2.1.6 Пренебрежение безопасным хранением данных //TODO 14](#_Toc104843883)

[2.1.7 Утечка информации 14](#_Toc104843884)

[2.1.8 Неккоректный доступ к файлам 14](#_Toc104843885)

[2.1.9 Случайные числа криптографического качества 15](#_Toc104843886)

[2.1.10 Переполнение целых чисел //TODO 15](#_Toc104843887)

[2.1.11 Гонки 15](#_Toc104843888)

[2.1.12 Читатели - писатели 15](#_Toc104843889)

[3 БИБЛИОТЕКИ/ПРОГРАММНЫЕ СРЕДСТВА, ВЫБОР ЯЗЫКА ПРОГРАММИРОВАНИЯ И СРЕДЫ РАЗРАБОТКИ 16](#_Toc104843890)

[3.1 Библиотеки и программные средства 16](#_Toc104843891)

[3.2 Выбор средств разработки и системных программных средств 17](#_Toc104843892)

[3.3 Требования к разрабатываемому программному обеспечению 17](#_Toc104843893)

[4 МИНИМАЛЬНЫЕ СИСТЕМНЫЕ ТРЕБОВАНИЯ 18](#_Toc104843894)

[5 ИНСТРУКЦИЯ ПО УСТАНОВКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 18](#_Toc104843895)

[6 ЗАПУСК ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 20](#_Toc104843896)

[7 ФУНКЦИОНАЛЬНОЕ ОПИСАНИЕ РАЗРАБОТАННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 21](#_Toc104843897)

[8 ДЕТАЛЬНОЕ ОПИСАНИЕ РАЗРАБОТАННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ 21](#_Toc104843898)

[8.1 Средства сборки, язык, каркас и модули программы 21](#_Toc104843904)

[8.2 Инициализация ресурсов 22](#_Toc104843905)

[8.3 Описание сущностей 22](#_Toc104843906)

[8.4 Итоговый отчет 23](#_Toc104843907)

[8.6 Пользовательский интерфейс 25](#_Toc104843908)

[9 НАЧАЛО РАБОТЫ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ 26](#_Toc104843909)

[9.1 Описание консольной версии программы 26](#_Toc104843910)

[9.2 Описание графического клиента программы 26](#_Toc104843911)

[9.3 Подготовка приложения к работе 28](#_Toc104843912)

[10 ВЫВОДЫ И РЕЗУЛЬТАТ 28](#_Toc104843913)

[11 КОМПЛЕКСНАЯ ОТЛАДКА 28](#_Toc104843914)

[ЗАКЛЮЧЕНИЕ 29](#_Toc104843915)

[СПИСОК ИСПОЛЬЗОВАННХ ИСТОЧНИКОВ 30](#_Toc104843916)

[ПРИЛОЖЕНИЕ А 31](#_Toc104843917)

РЕФЕРАТ

**Пояснительная записка к курсовому проекту:** 87 с., 6 рисунков, 2 таблицы, 4 источника, 1 приложение.

Цель работы - в данной работе рассмотрены вопросы поиска уязвимостей в программах, написанных на С++ путем анализа их текстов при помощи специально разработанных программ - анализаторов. Поскольку уязвимости в ПО вносятся на этапе разработки и многие разработчики ищут только ошибки ПО игнорируя уязвимости. Эти уязвимости могут быть использованны вредоносным ПО, так что для исправления этого целесообразно разработать анализатор уязвимостей, а после сделать анализ уязвимостей одним из этапов разработки. Разработка программ анализаторов и является целью проекта.

Уязвимости ПО - это точки, в которых имеются нарушения защиты или она отсутствует исходно и через которые можно провести атаку на доступность, конфиденциальность или целостность информации. Наличие уязвимости не приводит к получению неправильного ошибочного результата. Нужны дополнительные условия. Однако, иногда ошибка кроме неправильного результата способствует появлению уязвимости ПО. Для реализации информационной атаки злоумышленнику необходимо использовать ту или иную определенную им уязвимость АС. Наличие уязвимостей АС может быть обусловлено рядом факторов:

* ошибками в программном обеспечении;
* наличием ПО без установленных модулей обновлений (для закрытия ранее обнаруженных ошибок и уязвимостей);
* отсутствием средств контроля входных данных, ошибками в ЭТД;
* отсутствием средств защиты информации при прерываниях работы ПО;
* ошибочной конфигурацией защиты сетевых служб. Уязвимости могут быть внесены в ПОАС как на технологическом, так и на эксплуатационном этапах жизненного цикла. На технологических этапах жизненного цикла АС уязвимости могут быть внесены: 1.На стадии проектирования АС, в процессе которой могут быть выбраны такие технологии получения, хранения, преобразования и передачи информации, в которых отсутствуют механизмы защиты (не предусмотрены, например, межсетевые экраны).   
  2. На стадии разработки программно-аппаратного обеспечения АС могут быть допущены и не обнаружены такие ошибки, которые приводят к уязвимости и могут привести к нарушению ИБ.   
  3.На стадии установки и развертывания АС, в процессе которой выбираются параметры работы АС, уязвимые для потенциальных атак.

Эксплуатационные уязвимости вносятся при неправильной настройке и использовании программно-аппаратного обеспечения АС, например, при неправильной настройки тех же межсетевых экранов. Устранение эксплуатационных уязвимостей требует меньших усилий, так как это связанно с изменениями конфигурации АС, а не переработкой проекта АС. В работе рассматриваются уязвимости, внесенные в ПО на стадии его разработки т.е. уязвимости кода ПО.Поиск уязвимостей может осуществляться путем анализа текстов ПО без его исполнения, а также при исполнении текста ПО. В работе рассматриваются статические анализаторы текстов ПО, не требующие запуска ПО.

распределение обязанностей

**Громов В.В. –** разработал GUI клиент программы. Сделал привязку GUI клиента к разработанной библиотеке анализа кода. Оформил документацию для курсового проекта на git.

**Гафаров И.Н. –** создал механизм, позволяющий хранить уязвимости и список правил для них на кластерном сервере MongoDB, что позволило облегчить процесс взаимодействия с базой данных. Участвовал в создании функционала, позволяющего автоматизировать функциональное тестирование. Также реализовал алгоритмы, позволяющие выявить уязвимости утечка информации, некорректный доступ к файлам, случайные числа криптографического качества. Написал junit тесты, для разрабатываемого программного обеспечения.

**Кузнецов В.А. –** реализовал функционал, позволяющий выявлять уязвимости внедрение SQL команд, внедрение команд, гонки.

**Брагин Р.А. –** реализовал функционал, позволяющий выявлять уязвимости пренебрежение обработкой ошибок, пренебрежение безопасным хранением данных, переполнение целых чисел.

**Фролов Н.М. –** руководитель проекта. Организовал структуру проекта, распределил обязанности. Создал механизм создания отчетов в формате .html с помощью Apache Velocity, реализовал модель функционального тестирования программы. Также реализовал алгоритмы, позволяющие выявить уязвимости переполнение буфера, ошибка форматной строки, читатели и писатели.

ВВЕДЕНИЕ

Критические уязвимости в программах обычно появляются вследствие высокой структурной сложности кода и субъективных факторов, среди которых можно выделить: проблемы организации работы над проектом больших коллективов разработчиков (нечеткая спецификация требований, недостаточность информации, проектные предположения и пр.); социально-психологические проблемы в коллективах разработчиков, конфликты интересов, склонность к игнорированию стандартов и банальная невнимательность.   
 Существуют меры обеспечения качества кода, которые зависят от требований конкретного проекта. Стандартным набором пунктов для обеспечения качества кода являются:

* надлежащее документирование и учет изменений в коде;
* управление версиями (репозиторий кода);
* функциональное тестирование (на соответствие документации);
* тестирование по методам "черного", "серого", "белого" ящиков;
* тестирование исполнимого кода сканерами уязвимости;
* анализ качества исходных текстов программ;

В настоящее время активно развивается технологическое направление, связанное с автоматизацией анализа исходных кодов программ. Соответствующие средства, называемые статическими анализаторами, обладают следующими достоинствами:

* независимостью от функциональной специфики программ, что позволяет создавать универсальные анализаторы, направленные на выявление широкого спектра уязвимостей;
* высоким уровнем абстракции описания ветвей алгоритмов и функциональных объектов, отсутствием необходимости задания исходных (входных) данных анализируемых программ;
* стандартизованными способами описания и распространения сведений об уязвимостях, что повышает качество (полноту) контроля и снижает их реактивность (время, проходящее от момента выявления уязвимости до включения в БД анализатора описания уязвимости);
* высоким уровнем масштабируемости контролирующих процедур, которые могут применяться как в отношении отдельных фрагментов программы (вплоть до одной строки кода), так и в отношении сложных комплексов программ;
* минимальными ограничениями, налагаемыми на процесс разработки и тестирования программ.

Статические анализаторы могут использоваться практически на всех этапах разработки ПО, что позволяет в максимальной степени дисциплинировать разработчика и существенно снизить издержки за счет выявления уязвимостей на более ранних стадиях разработки. Статические анализаторы могут использоваться практически на всех этапах разработки ПО, что позволяет в максимальной степени дисциплинировать разработчика и существенно снизить издержки за счет выявления уязвимостей на более ранних стадиях разработки.

В данной работе будет реализован статический анализатор текстов ПО, написанного на языке программирования C++, который не будет требовать запуска самого ПО.

1 АНАЛИЗ СУЩЕСТВУЮЩЕГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ОБНАРУЖЕНИЯ УЯЗВИМОСТЕЙ В ПРОГРАММНОМ ОБЕСПЕЧЕНИИ НА C++

В данном разделе произведен краткий анализ существующих инструментов, позволяющих выявить уязвимости в исходном коде, написанном на языке программирования C++.

Рассмотрим одни из самых популярных статических анализаторов кода на C/C++:

1. **CppCheck** является бесплатным кроссплатформенным статическим анализатором с открытым исходным кодом (GPLv3). Он доступен в пакетах многих \*nix систем из коробки. Также CppCheck умеет интегрироваться со многими IDE. CppCheck хорош тем, что он довольно быстро работает, что позволяет без задней мысли добавить его запуск в систему непрерывной интеграции.
2. **Clang Static Analyzer** бесплатный кроссплатформенный статический анализатор с открытыми исходным кодом. Является частью так называемого LLVM-стэка. В отличие от CppCheck работает существенно медленнее, но и ошибки находит куда более серьезные. На выходе получается HTML-отчет с подробными пояснениями, возможностью фильтровать ошибки по их типу, и так далее. Отличительной особенностью является процесс работы данной программы так как изначально инструмент вызывает clean, а затем вызывает сборку под scan-build, что позволяет точно выявить уязвимости в исходном коде.
3. **PVS-Studio** - это инструмент для выявления ошибок и потенциальных уязвимостей в исходном коде программ, написанных на языках С, C++, C# и Java. Работает в 64-битных системах на Windows, Linux и macOS и может анализировать код, предназначенный для 32-битных, 64-битных и встраиваемых ARM-платформ. Данный инструмент выполняет статический анализ кода и генерирует отчёт, помогающий программисту находить и устранять ошибки. PVS-Studio выполняет широкий спектр проверок кода, но наиболее силён в поисках опечаток и последствий неудачного Copy-Paste. Основная ценность статического анализа заключается в его регулярном использовании, благодаря чему многие ошибки могут быть выявлены и устранены на самых ранних этапах.

В результате анализа существующих анализаторов кода, было сформировано целостное представление о том, какие функции должен выполнять в конечном варианте курсовой проект.

2 КРАТКИЙ ОБЗОР АНАЛИЗИРУЕМХ УЯЗВИМОСТЕЙ

Разрабатываемый в рамках курсового проекта анализатор уязвимостей, позволяет выявить следующий список уязвимостей:

1. переполнение буфера;
2. ошибка форматной строки;
3. внедрение SQL команд;
4. внедрение команд;
5. пренебрежение обработкой ошибок;
6. пренебрежение безопасным хранением данных;
7. утечка информации;
8. неккоректный доступ к файлам;
9. случайные числа криптографического качества;
10. переполнение целых чисел;
11. гонки;
12. читатели-писатели.

## 2.1 Описание уязвимостей

Рассмотрим каждую из уязвимостей.

### 2.1.1 Переполнение буфера

Переполнение буфера (англ. Buffer Overflow) – явление, возникающее, когда компьютерная программа записывает данные за пределами выделенного в памяти буфера.Переполнение буфера обычно возникает из-за неправильной работы с данными, полученными извне, и памятью, при отсутствии жесткой защиты со стороны подсистемы программирования (компилятор или интерпретатор) и операционной системы. В результате переполнения могут быть испорчены данные, расположенные следом за буфером (или перед ним). Переполнение буфера является одним из наиболее популярных способов взлома компьютерных систем, так как большинство языков высокого уровня использует технологию стекового кадра – размещение данных в стеке процесса, смешивая данные программы с управляющими данными (в том числе адреса начала стекового кадра и адреса возврата из исполняемой функции).

### 2.1.2 Ошибка форматной строки

Эксплойт Format String возникает, когда отправленные данные входной строки оцениваются приложением как команда. Таким образом, злоумышленник может выполнить код, прочитать стек или вызвать ошибку сегментации в работающем приложении, вызывая новые действия, которые могут поставить под угрозу безопасность или стабильность системы.

### 2.1.3 Внедрение команд

Внедрение команд – одна из самых опасных уязвимостей, так как дает злоумышленнику возможность внедрения собственных команд через вызов системных функций (примеры приведены в бд, безопасных аналогов у них нет), что может оказаться фатальным для системы, так как влияет на нее напрямую. Фиксится фильтрацией входящих данных и аккуратным использованием этих самых системных функций.

### 2.1.4 Внедрение SQL команд

Внедрение SQL команд – может оказаться критичной в некоторых ситуациях. При плохом написании логики работы с базами данных злоумышленник можетполучить все данные из базы или полностью ее изменить по своему усмотрению. Исправляется с помощью фильтрации входящих для запросов данных и испльзования подготовленных запросов(PreparedStatement).

### 2.1.5 Пренебрежение обработкой ошибок //TODO

### 2.1.6 Пренебрежение безопасным хранением данных //TODO

### 2.1.7 Утечка информации

Уязвимость заключается в том, что противник получает данные, которые могут привести к явному или неявному нарушению политики безопасности. Целью могут быть как сами данные, так и информация, с помощью которой противник сможет решить стоящую перед ним задачу.

Есть два основных пути утечки информации:

1) Случайно. Данные считаются ценными, но каким-то образом из-за логической ошибки в программе или по какому-либо неочевидному каналу они просочились наружу;

2) Намеренно. Иногда команда проектировщиков и конечный пользователь не одинаково понимают, что именно нужно защищать. Обычно это касается охраны тайны личной жизни.

Есть два вида побочных каналов:

1. Связанные с хронометражём. По хронометрируемому каналу противник получает информацию о внутреннем устройстве системы, измеряя время выполнения операций. Хронометрируемый канал – это наиболее распространенный вид побочного канала, но есть еще и каналы, связанные с хранением;
2. Побочные каналы, связанные с хранением. В общем случае не так интересны, как основной коммуникационный канал. Например, если все передаваемые по сети данные криптографически защищены, имя пользователя для процедуры аутентификации обычно приходится передавать в открытом виде. А это дает противнику отправную точку для атаки с угадыванием пароля или с применением социальной инженерии.

### 2.1.8 Неккоректный доступ к файлам

Данную уязвимость можно определить тремя проблемами безопасности:

1. «Гонки». Между моментом проверки условий защиты для файла и моментом использования этого файла часто есть некоторое окно, когда файл уязвим;
2. «Это не файл». Суть в том, что программа открывает нечто, считая, что это файл на диске, тогда как фактически это нечто является ссылкой на другой файл, именем устройства или канала;
3. Контроль над файлом, к которому не должно быть доступа. В результате противник может прочитать, а возможно даже изменить конфиденциальные данные.

### 2.1.9 Случайные числа криптографического качества

Случайные числа – это одна из основ криптографии, поэтому они встречаются в самых разных языках. А ошибки при их использовании тоже присущи любому языку.

Обычно ошибка при работе со случайными числами заключается в том, что они используются не там, где нужно.

### 2.1.10 Переполнение целых чисел //TODO

### 2.1.11 Гонки

Уязвимость гонок – тяжелоотлавливаемая уязвимость, так как для ее выявления на любом языке нужно знать особенности работы языка и работы операционной системы (уязвимость завязана на потоках). При работе нескольких потоков с одним и тем же ресурсом данные могут изменяться некорректно (один поток перезаписал свои локальные данные вместо локальных данных другого потока), что может нарушить дальнейшую логику программы. Исправляется использованием мьютексов.

### 2.1.12 Читатели - писатели

Читатели и писателти – одна из важнейших задач параллельного программирования. Формулируется она так: Есть область памяти, допускающая чтение и запись. Несколько потоков имеют к ней доступ, при этом одновременно могут читать сколько угодно потоков, но писать — только один. Как обеспечить такой режим доступа? Можно обойтись обычным мьютексом, но это не оптимально — компьютерная память, как правило, устроена так, что несколько потоков могут свободно читать и писать её (единственная проблема – нет гарантии, что в процессе обработки переменная внезапно не изменится). Кому отдавать приоритет – читателю или писателю – остаётся за программистом. Процессы-читатели считывают, а процессы-писатели записывают информацию в общую область памяти. Одновременно мо­жет быть несколько активных процессов-читателей. При записи информации область памяти рассматривается как критический ресурс для всех процессов, т. е. если работает процесс-писатель, то он должен быть единственным активным процессом. Задача состоит в определении структуры управления, которая не приведет к тупику и не допустит нарушения критерия взаимного исключения.

3 БИБЛИОТЕКИ/ПРОГРАММНЫЕ СРЕДСТВА, ВЫБОР ЯЗЫКА ПРОГРАММИРОВАНИЯ И СРЕДЫ РАЗРАБОТКИ

## 3.1 Библиотеки и программные средства

Для реализации функционала программного обеспечения использовался следующий список библиотек:

1. Mongo-java-driver – библиотека, позволяющая работать с базой данных MongoDB, с помощью использования Java кода;
2. Velocity – движок для обработки шаблонов (template engine), который позволяет напрямую обращаться к методам и полям Java классов. В данной работе использовалась для создания результирующих отчетов;
3. Junit – библиотека для модульного тестирования программного обеспечения на языке Java;
4. Log4j – быстрая и гибкая среда ведения журналов (логирования), написанная на Java;
5. Maven-antrun-plugin – плагин, который позволяет запускть ant задачи, с помощью maven. Ant задачи могут быть описаны внутри pom.xml, использовался для автоматизации функционального тестирования;
6. JavaFX – представляет инструментарий для создания кроссплатформенных графических приложений на платформе Java;

## 3.2 Выбор средств разработки и системных программных средств

Для реализации поставленной задачи был выбран язык программирования Java. Главными преимуществами данного языка являются простота изучения, объектно-ориентированный подход в разработке программного обеспечения, кросплатформенность, высокопроизводительность.

В качестве среды разработки было принято решение использовать IntelliJ IDEA – то IDE, интегрированная среда разработки (комплекс программных средств, который используется для написания, исполнения, отладки и оптимизации кода) для Java, JavaScript, Python и других языков программирования от компании JetBrains. Отличается обширным набором инструментов для рефакторинга (перепроектирования) и оптимизации кода.

## 3.3 Требования к разрабатываемому программному обеспечению

Разрабатываемая программа должна отвечать следующим требованиям, обусловленным ее назначением и сферой применения:

1. Поиск уязвимостей в программах, написанных на С++ путем анализа их текстов при помощи специально разработанной программы - анализатора;
2. Составление отчета по найденным уязвимостям, при этом должно быть указано в какой строчке кода найдена уязвимость, описание уязвимости и способ её устранения;
3. Наличие графического интерфейса, для удобного взаимодействия с программой;

4 МИНИМАЛЬНЫЕ СИСТЕМНЫЕ ТРЕБОВАНИЯ

Требования к рабочей станции для установки данного программного обеспечения:

* Процессор — Intel Core i3 или другой схожий процессор с количеством ядер 2 или более;
* Объем оперативной памяти — не менее 2 ГБайт (рекомендуется 4 Гбайт);
* Свободное место на жестком диске — не менее 100 Мбайт;
* Операционная система — Windows 7 / Windows 8 / Windows 8.1 / Windows 10 / Windows 11;
* Система сборки apache maven 3.8.4
* Java 17;
* Java Development Kit (JDK) 17.

5 ИНСТРУКЦИЯ ПО УСТАНОВКЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Шаги по установке программного обеспечения:

1. Проверьте, что дистрибутив языка программирования Java версии 17 (или выше) установлен на вашей рабочей станции. Если он отсутствует в системе, пожалуйста, установите Java, скачав дистрибутив с официального сайта: <https://www.java.com/ru/download/manual.jsp> ;
2. Установите JDK (Java Development Kit) 17 (или выше). Обычно JDK располагается в C:\Program Files\Java\jdk-X, где X – версия JDK.

<https://www.oracle.com/java/technologies/downloads/>

1. Установите Apache Maven последней рабочей версии <https://maven.apache.org/download.cgi> (скачать желательно бинарный архив). После установки откройте архив и переместите папку с «мавеном» в C:\Program Files\Maven
2. Установите JavaFX 17 версии и выше <https://gluonhq.com/products/javafx>  
   (Type – SDK). Распакуйте архив в C:\Program Files\Java\
3. Установите переменную среды JAVA\_HOME на директорию установленного JDK. Для этого в поиске (в системе) необходимо ввести «Изменение системных переменных среды» и открыть появившуюся в окне результата утилиту. В открывшемся окне необходимо выбрать вкладку «Дополнительно», а затем нажать на кнопку «Переменные среды…». В нижней части («Системные переменные») нового открывшегося окна необходимо нажать на кнопку «Создать». Во вновь открывшемся диалоговом окне нужно указать название переменной «JAVA\_HOME» и указать полный путь до JDK.  
   Аналогично установите переменную среды «M2\_HOME», в которой укажите полный путь до Maven (см. рисунок 1).  
   Следующим шагом в перменной «Path» пропишите пути до папки «bin»:  
   %JAVA\_HOME%\bin и %M2\_HOME%\bin

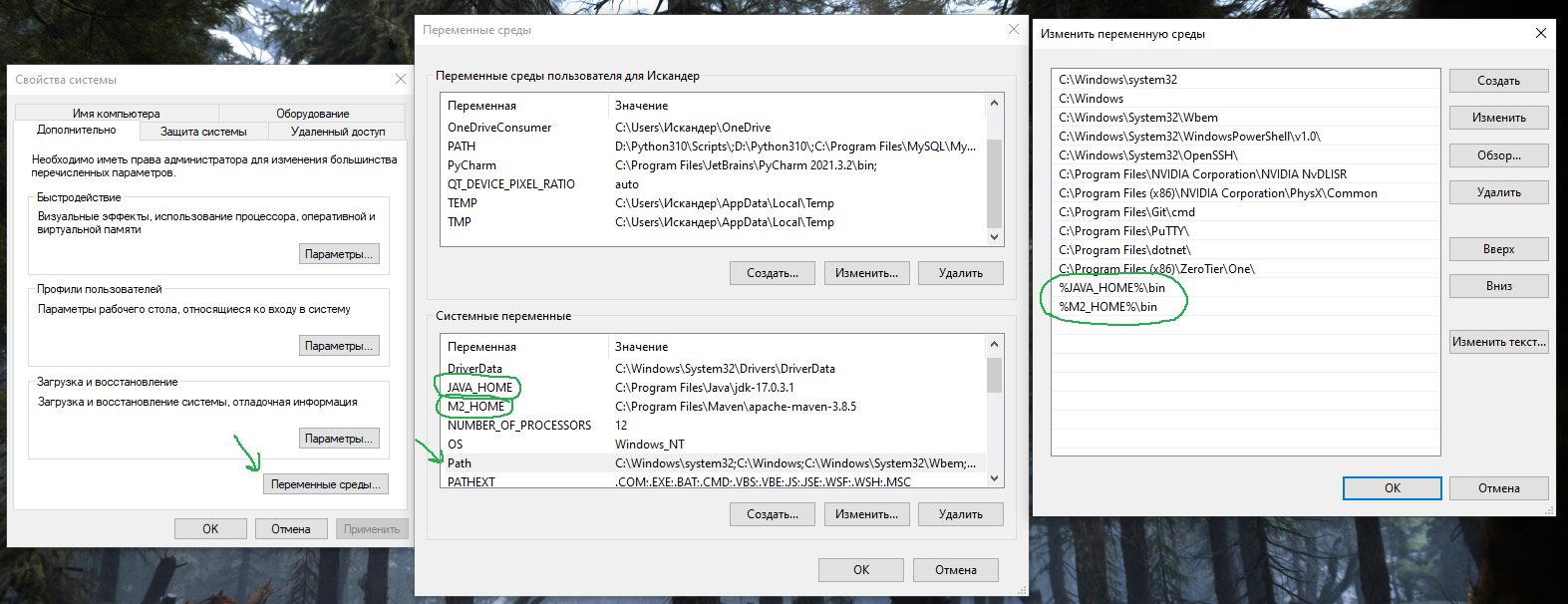


Рисунок 1. Установка переменной среды JAVA\_HOME и M2\_HOME

1. Проверьте, что в переменной среды Path существует директория System32 (обычно находится по пути C:\Windows\System32). Если такого пути нет в переменной среды, добавьте его;
2. Запустите командную строку и введите java -version и mvn -version. Должны быть показаны версии JDK и Maven (см. рисунок 2);

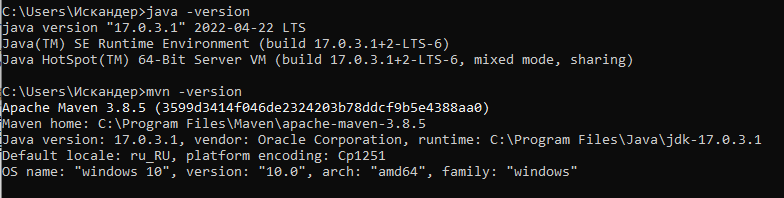


Рисунок 2. Проверка версий установленных JDK и Maven

6 ЗАПУСК ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Запуск GUI разработанного ПО осуществляется командой:

java --module-path "C:\Program Files\Java\javafx-sdk-17.0.2\lib" --add-modules javafx.controls,javafx.fxml -jar GUI\_for\_Course-1.0-SNAPSHOT-jar-with-dependencies.jar

где javafx-sdk-17.0.2 – это название папки, полученной на пункте 3.4

Открываем командную строку в папке с проектом и вводим данную команду.

(см. рисунок 3)

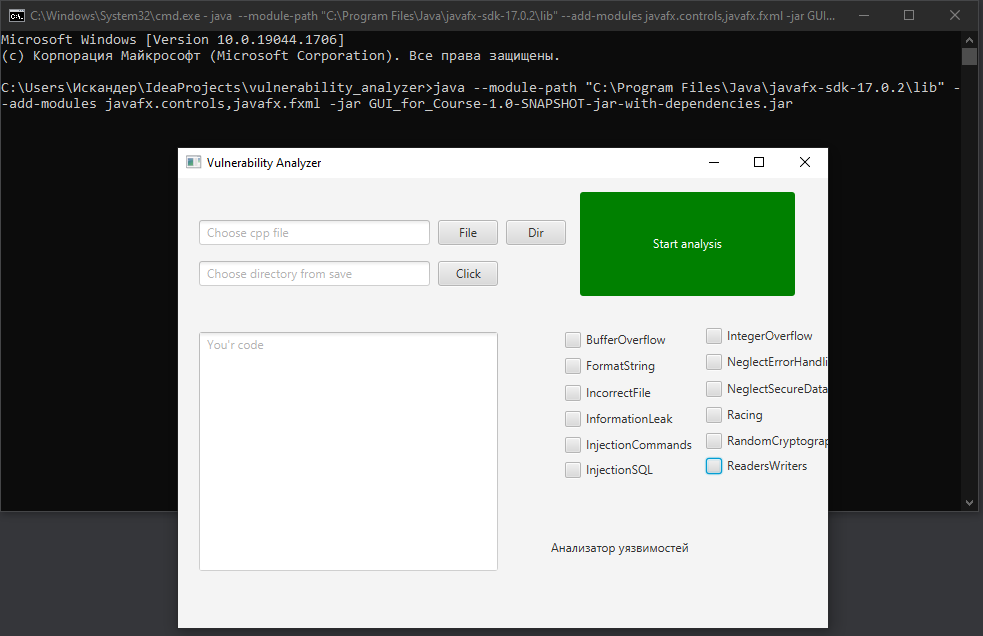


Рисунок 3. Успешный запуск ПО

7 ФУНКЦИОНАЛЬНОЕ ОПИСАНИЕ РАЗРАБОТАННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Программное обеспечение представляет собой инструмент, с помощью которого можно проверить код, написанный на языке программирования C++ на уязвимости. База данных с уязвимостями находится на удаленном сервере, что позволяет избежать дополнительной локальной настройки. Приложение может функционировать, как консольное, но при этом есть возможность использовать GUI интерфейс для более удобного взаимодействия. Все модули программы написаны на языке Java.

Обший принцип работы ПО: разработанный инструмент построчно анализирует исходный код, написанный на языке C++. Каждая строчка проверяется на соответствие маскам уязвимостей. В свою очередь маски представляют из себя регулярные выражения, которые хранятся в базе данных. Для взаимодействия с программой пользователю необходимо указывает, файл или директорию в которых находится исходный код, написанный на C++, после этого необходимо указать директорию, в которую будет сохранен отчет о наличии уязвимостей в исходном коде, также необходимо указать список уязвимостей, наличие которых будет проверятся в исходных файлах. В результате работы программы получаются отчеты в формате .html, в которых указаны найденные уязвимости, номер строчки кода, в которой была обнаружена уязвимость, описание уязвимости и способ её устранения.

8 ДЕТАЛЬНОЕ ОПИСАНИЕ РАЗРАБОТАННОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ



## 8.1 Средства сборки, язык, каркас и модули программы

Программное обеспечение написано на языке Java версии 17.0.1. Для сборки программного обеспечения был использован Maven (фреймворк для автоматизации сборки проектов на основе описания их структуры в файлах на языке POM, являющемся подмножеством XML).

Программное обеспечение было разделено на 3 модуля: analyzer-library, functional-tests и gui. Основная логика программы находится в модуле analyzer-library, в модуле functional-tests находится код для автоматического тестирования сборки модуля analyzer-library, в модуле gui находится код для использования и функционирования графического интерфейса программы.

Фундамент разработанного программного обеспечения служит Java SE. Java SE или Core Java – это базовое API, которые обычно используется для создания десктопных приложений.

В процессе разработки были задействованы такие технологии как Apache Velocity, Junit, JavaFX, Mockito и Mongo-java-driver.

Основынми элементами приложения являются классы нахождения уязвимостей, также есть сервис, необходимый для взаимодействия с удаленной базой данных, кроме того в приложении находится блок макросов, для создания отчетов в формате .html.

## 8.2 Инициализация ресурсов

При запуске программного обеспечения происходит инициализация исходных ресурсов с последующей обработкой конфигураций, написанных разработчиком. В этом приложении используется конфигурация для настройки объекта доступа к базе данных (все необходимые сведения для подключения), а также конфигурация настройки сервлетов (специальных объектов, предназначенных для обработки запросов пользователя и выдачи ответов).

## 8.3 Описание сущностей

В приложении можно выделить четыре типа сущностей: уязвимости, которые наследуются от абстрактного класса Vulnerability, классы для создания отчетов, классы для взаимодействия с базой данных, а также утильные классы, которые служат для выполнения определенных операций. Рассмотрим более подробно каждый тип сущностей.

Уязвимости являются наследниками класса Vulnerabilities и содержат в себе алгоритмы для нахождения уязвимостей в исходном коде, анализ кода производится построчно, что позволяет распознать уязвимость в любом месте программы.

Классы для создания отчетов необходимы для предоставления пользователям сведений о найденных уязвимостях в формате .html, а также для взаимодействия с макросами, написанных на vml.

Классы для взаимодействия с базой данных, позволяют выполнять CRUD операции с базой данных. При запуске программы происходит считывание необходиммых сведений для обнаружения уязвимстей в исходном коде.

Утильные классы предоставляют такие возможности, как взаимодействие с файловой системой, обработкой и валидацией входных данных с консольного клиента, а также хранение констант для успешного функционирования программы.

## 8.4 Итоговый отчет

После окончания работы программы в указанной директории формируется отчет. В случае, если на вход программы подавался файл формата .cpp, то результирующий отчет будет представлять из себя один файл формата .html, пример отчета изображен на рисунке 4.

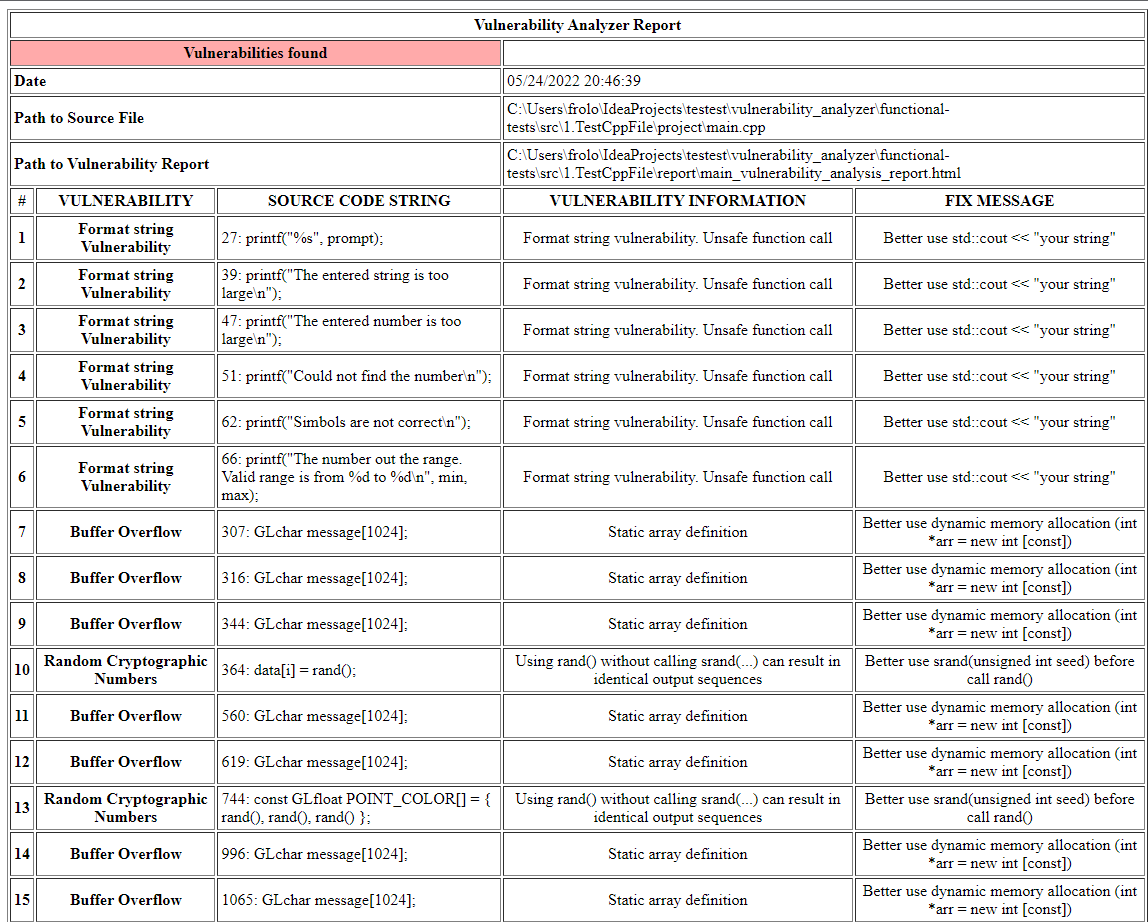


Рисунок 4 - пример отчета для одиночного файла формата .cpp

В случае если на вход была подана директория, содержащая в себе множество файлов формата .cpp, то на выходе мы получим отчет для кажого из исходных файлов и общий отчет, в котором будет находиться инофрмация о успешном или неудачном прохождении анализа, для каждого .cpp файла, пример общего отчета изображен на рисунке 5.

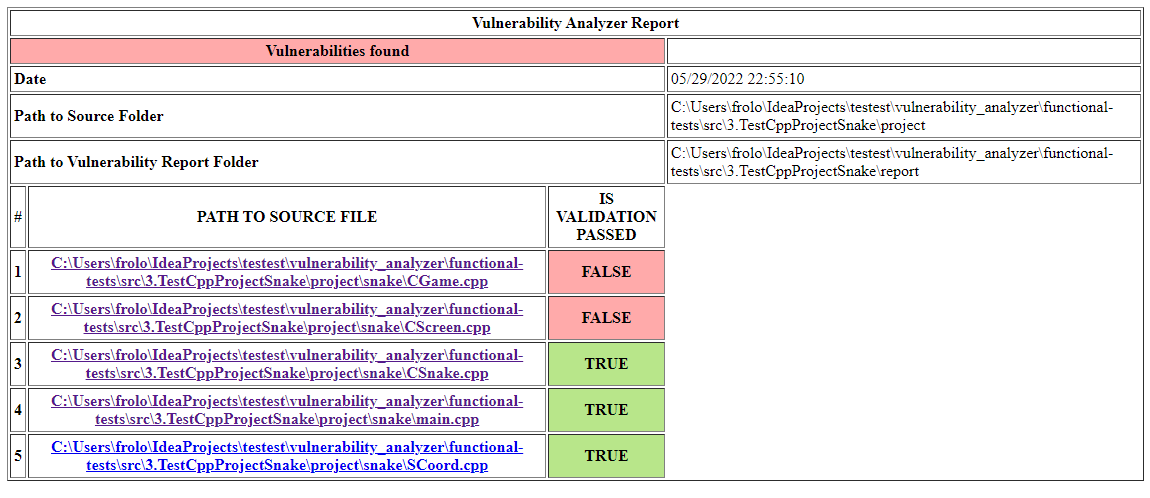


Рисунок 5 - пример общего отчета

В данном отчете содержиться информация, в каких файлах были найдены уязвимости, а в каких уязвимостей вовсе нет.

## 8.6 Пользовательский интерфейс

Для удобного взаимодействия с программой приложение содержит графический интерфейс, в котором можно выбрать проект или файл, который необходимо проверить на уязвимости, директорию в которую будет собран финальный отчет и список уязвимостей для проверки. Пример графического интерфейса представлен на рисунке 6.

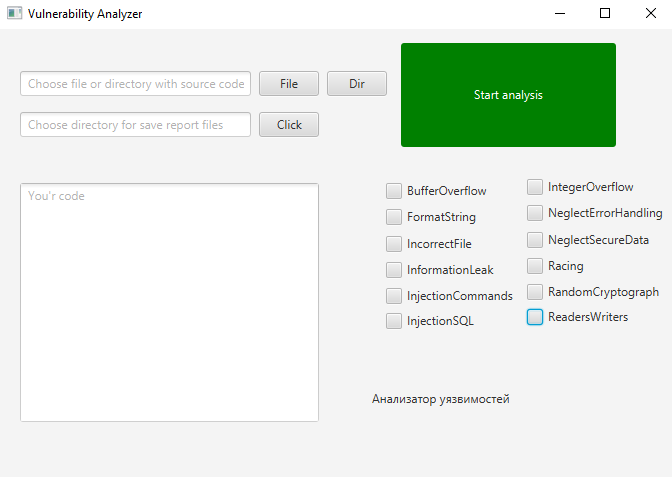


Рисунок 6. Пользовательский интерфейс

9 НАЧАЛО РАБОТЫ С ПРОГРАММНЫМ ОБЕСПЕЧЕНИЕМ

### 9.1 Описание консольной версии программы

Для запуска консольного приложения необходимо скачать файл analyzer-library-1.0-SNAPSHOT.jar и установить jdk 17.0.1. Параметры для запуска консольного приложения прдставлены в таблице 1.

Таблица 1 – Параметры для запуска консольной версии анализатора уязвимостей

|  |  |  |
| --- | --- | --- |
| Название ключа | Значение ключа | Пример использования |
| -source | -s | Путь до файла или до директории с исходными файлами в формате C++ | C:\Users\test\project\main.cpp  C:\Users\test\project |
| -report | -r | Путь до директории в которой будут лежать отчеты в формате .html | C:\Users\test\reportFolder |
| -vulnerabilities | -v | Список уязвимостей | "randomCryptographicNumbers, bufferOverflow, formatStringError" |

Пример полной команды для запуска программы представлен на рисунке 7.

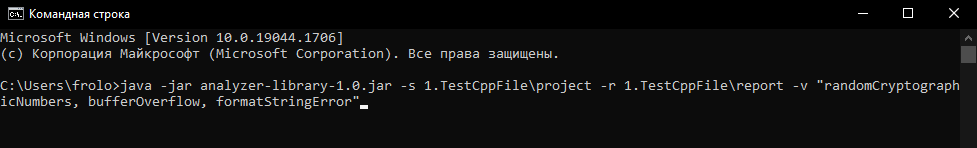


Рисунок 7 - запуск консольной версийй анализатора уязвимостей

### 9.2 Описание графического клиента программы

Пример графического интерфейса серверного приложения представлен на рисунках 5 и 6.

Описание полей, представленных в графическом интерфейсе серверного приложения, представлено в таблице 2.

Таблица 2 – Описание полей графического интерфейса

|  |  |
| --- | --- |
| Описание элементов интерфейса | Описание поля |
| Choose file or directory with source code | Для выбора одиночного файла необходимо нажать на кнопку File, после чего будет открыт проводник, в котором нужно указать файл с исходныи кодом в формате .cpp.  Для выбора директории в которой находятся несколько файлов в формает .cpp необходимо нажать на кнопку Dir. |
| Перечень чекбоксов | В правой строне графического интерфейса находится перечень чекбоксов. Для проверки определенной уязвимости следует нажать на пустой квадрат, для того чтобы в нем появилась галочка, которая служит сигналом о том, что данная уязвимость будет проанализирована в указанных файлах. |
| Зеленая кнопка “Start analysis” | Запускает анализатор уязвимостей |
| Поле “code here” | В случае, если на вход программы подаваля одиночный файл, то его исходный код будет расположен в данном окнке. |

Графический интерфейс изображен на рисунке 6.

### 9.3 Подготовка приложения к работе

Перед стартом приложения необходимо заполнить все поля, указанные в графическом пользовательском интерфейсе. Необходимо корректно указать сетевой входные и выходные директории. В случае если указанных файлов или директорий не существует программа отработает неккоректно.

# 10 ВЫВОДЫ И РЕЗУЛЬТАТ

В результате выполнения курсовой работы был написан программный продукт, позволяющий передавать конфиденциальную информацию через физически отключаемый канал. Результаты тестов показали, что разработанная программа успешно справляется с поставленной задачей. Все созданные алгоритмы были проверены, возникновение ситуации пропуска уязвимостей в исходном коде минимальна.

# 11 КОМПЛЕКСНАЯ ОТЛАДКА

Отладка проходила в несколько этапов. Первым этапом было запуск ПО на заведо уязвимых .cpp файлах. После чего было проведено внутренней работы программы с помощью Junit тестов. В самую последнюю очередь был создан механизм для автоматического тестирования API, с помощью функциональных тестов. Функциональные тесты запускаются во время сборки основной программы и содржат в себе различные комбинации проверяемых уязвимостей, после отработки всех тестов в модуле с функциольными тестами формируются подробные отчеты по каждому из тестов.

В ходе работы были проверены и отлажены алгоритмы обнаружения уязвимостей.

Было отлажено взаимодействие кода с удаленной базой данных MongoDB.

Была отлажена работа API по созданию отчетов в формате .html.

ЗАКЛЮЧЕНИЕ

В результате работы над курсовым проектом были изучены методы обнаружения уязвимостей в программном обеспечении, написанном на языке программирования C++, была опробована работа с NoSQL базой данных, а также с Apache Velocity для составления отчетов. В итоге получился готовый программный инструмент, который можно использовать для нахождения уязвимостей в исходном коде.

СПИСОК ИСПОЛЬЗОВАННХ ИСТОЧНИКОВ

1. Майкл Ховард, Дэвид Лебланк, Джон Виега. Уязвимости в программном коде и борьба с ними // ДМК Пресс 2011. – 2011. – 268с.   
2. Документация MongoDB [Электронный ресурс]: https://www.mongodb.com.  
3. Документация Apache Velocity [Электронный ресурс]: https://velocity.apache.org.

4. Документация ibm [Электронный ресурс]: www.ibm.com.

# ПРИЛОЖЕНИЕ А

Код программы

**Модуль: analyzer-library**

**Пакет: utils**

**//TODO CONSTANTS**

public class FileHelper {

private FileHelper() {

//Empty

}

public static List<String> readDataFromCppFile(String pathToFile) {

if (validateInputFile(pathToFile)) {

try {

return Files.lines(Path.of(pathToFile), StandardCharsets.ISO\_8859\_1)

.collect(Collectors.toCollection(ArrayList::new));

} catch (IOException e) {

e.printStackTrace();

}

} else {

throw new IllegalArgumentException("Invalid file path or extension\nExpected extension .cpp");

}

return null;

}

public static List<String> readCppFilesFromDirectory(String pathToFolder) {

List<String> files = new ArrayList<>();

findFiles(pathToFolder, files);

return files;

}

public static void findFiles(String baseDirName, List<String> resultFileList) {

File baseDirectory = new File(baseDirName);

if (!baseDirectory.exists() || !baseDirectory.isDirectory()) {

System.out.println("Error find file: " + baseDirectory);

} else {

String[] fileList = baseDirectory.list();

for (String s : fileList) {

File readFile = new File(baseDirectory + "\\" + s);

if (!readFile.isDirectory()) {

if (readFile.getAbsolutePath().contains(".cpp")) {

resultFileList.add(readFile.getAbsolutePath());

}

} else if (readFile.isDirectory()) {

findFiles(readFile.getAbsolutePath(), resultFileList);

}

}

}

}

public static void createDirs(String reportFolder, String fileReportPath) {

String diff = reportFolder.replace(fileReportPath, Constants.EMPTY\_STRING);

String[] dirs = diff.split("\\\\");

String path = fileReportPath;

for (int i = 1; i < dirs.length; ++i) {

path += "\\" + dirs[i];

File file = new File(path);

if (!file.exists()) {

file.mkdir();

}

}

}

private static boolean validateInputFile(String pathToFile) {

File file = new File(pathToFile);

return file.isFile() && file.exists() && file.getName().contains(".cpp");

}

public static void recursiveDelete(File file) {

if (!file.exists())

return;

if (file.isDirectory()) {

for (File f : file.listFiles()) {

recursiveDelete(f);

}

}

file.delete();

}

}  
  
public enum InputParametersPattern {

SOURCE\_CODE\_PATH {

@Override

public String getPattern() {

return "-source|-s";

}

}, REPORT\_FOLDER\_PATH {

@Override

public String getPattern() {

return "-report|-r";

}

}, VULNERABILITIES\_LIST {

@Override

public String getPattern() {

return "-vulnerabilities|-v";

}

};

public abstract String getPattern();

public static InputParametersPattern getPatternByValue(String value) {

for (InputParametersPattern parametersPattern: values()) {

if (value.matches(parametersPattern.getPattern())) {

return parametersPattern;

}

}

throw new IllegalArgumentException("Parameter " + value + " is not supported");

}

public static boolean checkValidatePattern(String value) {

for (InputParametersPattern parametersPattern : values()) {

if (value.matches(parametersPattern.getPattern())) {

return true;

}

}

return false;

}

}  
  
public class ParametersParser {

public static Map<InputParametersPattern, String> parseInputParameters(String []args) {

if (args.length % 2 != 0) {

throw new IllegalArgumentException("Parameters is always a pair of name and value");

}

Map<InputParametersPattern, String> parameters = new EnumMap(InputParametersPattern.class);

for (int i = 0; i < args.length; i += 2) {

if ((InputParametersPattern.checkValidatePattern(args[i]))

&& !(InputParametersPattern.checkValidatePattern(args[i+1]))) {

parameters.put(InputParametersPattern.getPatternByValue(args[i]), args[i+1]);

}

else {

throw new IllegalArgumentException("Invalid parameters + value format");

}

}

return parameters;

}

}  
  
**Пакет: launcher**public class VulnerabilityAnalyzerLauncher {

private final List<String> activeVulnerability;

private final String pathToSourceFile;

private final String pathToReportFolder;

public VulnerabilityAnalyzerLauncher(List<String> activeAbstractVulnerability

, String pathToSourceFile, String pathToReportFolder) {

this.activeVulnerability = activeAbstractVulnerability;

this.pathToSourceFile = pathToSourceFile;

this.pathToReportFolder = pathToReportFolder;

}

public void execute() {

File reportFolder = new File(pathToReportFolder);

FileHelper.recursiveDelete(reportFolder);

reportFolder.mkdir();

File file = new File(pathToSourceFile);

if (file.isFile()) {

launchAnalysis(pathToSourceFile, pathToReportFolder);

} else if (file.isDirectory()) {

List<String> fileList = FileHelper.readCppFilesFromDirectory(pathToSourceFile);

List<CppFileReportInformation> reportInformation = new ArrayList<>();

for (String curFile : fileList) {

File sourceFile = new File(curFile);

String reportPath = calculateReportPath(curFile);

CppFileReportInformation fileReport = new CppFileReportInformation(curFile

, reportPath + Constants.BACKSLASH

+ sourceFile.getName().replace(Constants.POINT\_CPP, Constants.EMPTY\_STRING)

+ "\_" + Constants.REPORT\_FILE\_NAME

, launchAnalysis(curFile, reportPath));

reportInformation.add(fileReport);

}

ReportCreator.createGeneralHtmlReport(pathToSourceFile, pathToReportFolder, reportInformation);

}

}

private boolean launchAnalysis(String pathToFile, String filePathToReportFolder) {

List<String> sourceCode = FileHelper.readDataFromCppFile(pathToFile);

VulnerabilityFactory factory = new VulnerabilityFactory();

List<ReportInformation> reportInformation = new ArrayList<>();

for (String vulnerabilityName : activeVulnerability) {

Vulnerability vulnerability = factory.getVulnerabilityInstance(vulnerabilityName);

List<ReportInformation> curVulnerabilityReportInformation = vulnerability.checkRules(sourceCode);

if (curVulnerabilityReportInformation.size() != 0) {

reportInformation.addAll(curVulnerabilityReportInformation);

}

}

File file = new File(pathToFile);

String fileName = file.getName().replace(Constants.POINT\_CPP, Constants.EMPTY\_STRING);

reportInformation.sort(Comparator.comparingInt(ReportInformation::getLineNumber));

FileHelper.createDirs(filePathToReportFolder, pathToReportFolder);

ReportCreator.createHtmlReport(pathToFile, filePathToReportFolder, reportInformation, fileName);

return reportInformation.size() == 0;

}

private String calculateReportPath(String curFile) {

File file = new File(curFile);

String pathDifference = curFile.replace(pathToSourceFile, Constants.EMPTY\_STRING);

String resultPath = (pathToReportFolder + pathDifference).replace(file.getName(), Constants.EMPTY\_STRING);

return resultPath.substring(0, resultPath.length() - 1);

}

}  
  
**Пакет: data**

public class MongoService {

private static final String MONGO\_URI = "mongodb+srv://root:admin@cluster0.h6ker.mongodb.net/test";

private static final String DB\_NAME = "root";

private static final String COLLECTION\_NAME = "vulnerability";

private final MongoCollection<VulnerabilityEntity> mongoCollection = getMongoCollection();

private MongoCollection<VulnerabilityEntity> getMongoCollection() {

MongoClient client = MongoClients.create(MONGO\_URI);

MongoDatabase database = client.getDatabase(DB\_NAME);

return database.withCodecRegistry(CodecRegistries.fromRegistries(

MongoClientSettings.getDefaultCodecRegistry()

, CodecRegistries.fromProviders(PojoCodecProvider.builder().automatic(true).build())))

.getCollection(COLLECTION\_NAME, VulnerabilityEntity.class);

}

public List<Rule> getRulesByVulnerability(String vulnerabilityName) {

VulnerabilityEntity entity = getVulnerabilityEntity(vulnerabilityName);

if (entity != null)

return entity.getRules();

return Collections.emptyList();

}

public VulnerabilityEntity addVulnerabilityEntity(VulnerabilityEntity entity) {

mongoCollection.insertOne(entity);

return entity;

}

public VulnerabilityEntity deleteVulnerabilityEntity(String vulnerabilityName) {

VulnerabilityEntity toDelete = getVulnerabilityEntity(vulnerabilityName);

mongoCollection.deleteOne(Filters.eq("vulnerabilityName", vulnerabilityName));

return toDelete;

}

private VulnerabilityEntity getVulnerabilityEntity(String vulnerabilityName) {

return mongoCollection.find(Filters.eq("vulnerabilityName", vulnerabilityName)).first();

}

}  
  
public class VulnerabilityEntity {

private String vulnerabilityName;

private List<Rule> rules;

public VulnerabilityEntity() {

}

public VulnerabilityEntity(String vulnerabilityName, List<Rule> rules) {

this.vulnerabilityName = vulnerabilityName;

this.rules = rules;

}

public String getVulnerabilityName() {

return vulnerabilityName;

}

public void setVulnerabilityName(String vulnerabilityName) {

this.vulnerabilityName = vulnerabilityName;

}

public List<Rule> getRules() {

return rules;

}

public void setRules(List<Rule> rules) {

this.rules = rules;

}

@Override

public String toString() {

return "VulnerabilityEntity {"

+ "\nvulnerabilityName = " + vulnerabilityName

+ ",\nrules = " + rules

+ "\n}";

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

VulnerabilityEntity that = (VulnerabilityEntity) o;

return Objects.equals(vulnerabilityName, that.vulnerabilityName) && Objects.equals(rules, that.rules);

}

@Override

public int hashCode() {

return Objects.hash(vulnerabilityName, rules);

}

}  
  
**Пакет: report**

public class CppFileReportInformation {

private final String pathToFile;

private final String pathToReportFile;

private final boolean validationPassed;

public CppFileReportInformation(String pathToFile, String pathToReportFile, boolean validationPassed) {

this.pathToFile = pathToFile;

this.pathToReportFile = pathToReportFile;

this.validationPassed = validationPassed;

}

public String getPathToFile() {

return pathToFile;

}

public boolean getValidationPassed() {

return validationPassed;

}

public String getPathToReportFile() {

return pathToReportFile;

}

}  
  
public class ReportCreator {

private static final String DATE\_FORMAT = "MM/dd/yyyy HH:mm:ss";

private static final String PATH\_TO\_REPORT\_TEMPLATE = "report-file-template.vm";

private static final String PATH\_TO\_GENERAL\_REPORT\_TEMPLATE = "general-report-file-template.vm";

public static void createHtmlReport(String pathToSourceFile, String pathToReportFolder

, List<ReportInformation> reportInformationList, String fileName) {

String pathToReportFile = pathToReportFolder + Constants.BACKSLASH

+ fileName + "\_"

+ Constants.REPORT\_FILE\_NAME;

boolean isAnalysisPassed = reportInformationList.size() == 0;

VelocityHelper.saveReportToFile(PATH\_TO\_REPORT\_TEMPLATE, pathToReportFile

, setParametersForReport(pathToSourceFile

, pathToReportFile, isAnalysisPassed, reportInformationList));

}

public static void createGeneralHtmlReport(String pathToSourceFolder, String pathToReportFolder

, List<CppFileReportInformation> cppFilesReport) {

boolean isAnalysisPassed = true;

for (CppFileReportInformation cppFile : cppFilesReport) {

if (!cppFile.getValidationPassed()) {

isAnalysisPassed = false;

break;

}

}

VelocityHelper.saveReportToFile(PATH\_TO\_GENERAL\_REPORT\_TEMPLATE, pathToReportFolder

, setParametersForGeneralReport(pathToSourceFolder, pathToReportFolder

, isAnalysisPassed, cppFilesReport));

}

private static Map<String, Object> setParametersForReport(String pathToSourceFile, String pathToReportFile

, boolean isAnalysisPassed, List<ReportInformation> reportInformationList) {

Map<String, Object> parameters = new HashMap<>();

parameters.put(Constants.VALIDATION\_DATE, new SimpleDateFormat(DATE\_FORMAT, Locale.ENGLISH).format(new Date()));

parameters.put(Constants.PATH\_TO\_SOURCE\_FILE, pathToSourceFile);

parameters.put(Constants.PATH\_TO\_REPORT\_FILE, pathToReportFile);

parameters.put(Constants.GENERAL\_ANALYSIS\_PASSED, isAnalysisPassed);

parameters.put(Constants.REPORT\_INFORMATION\_LIST, reportInformationList);

return parameters;

}

private static Map<String, Object> setParametersForGeneralReport(String pathToSourceFolder

, String pathToReportFolder, boolean isAnalysisPassed

, List<CppFileReportInformation> reportInformationList) {

Map<String, Object> parameters = new HashMap<>();

parameters.put(Constants.VALIDATION\_DATE, new SimpleDateFormat(DATE\_FORMAT, Locale.ENGLISH).format(new Date()));

parameters.put(Constants.PATH\_TO\_SOURCE\_FILE, pathToSourceFolder);

parameters.put(Constants.PATH\_TO\_REPORT\_FILE, pathToReportFolder);

parameters.put(Constants.GENERAL\_ANALYSIS\_PASSED, isAnalysisPassed);

parameters.put(Constants.REPORT\_INFORMATION\_LIST, reportInformationList);

return parameters;

}

}  
  
public class ReportInformation {

private final int lineNumber;

private final String sourceCodeString;

private final String vulnerabilityName;

private final Rule rule;

public ReportInformation(String sourceCodeString

, String vulnerabilityName, Rule rule, int lineNumber) {

this.sourceCodeString = sourceCodeString;

this.vulnerabilityName = vulnerabilityName;

this.rule = rule;

this.lineNumber = lineNumber;

}

public String getSourceCodeString() {

return sourceCodeString;

}

public String getVulnerabilityName() {

return vulnerabilityName;

}

public Rule getRule() {

return rule;

}

public int getLineNumber() {

return lineNumber;

}

}  
  
public class VelocityHelper {

private static final Logger log = LogManager.getLogger(VelocityHelper.class);

public static void saveReportToFile(String pathToTemplateFile, String pathToReportFile

, Map<String, Object> parameters) {

Template template = loadVelocityTemplate(pathToTemplateFile);

VelocityContext context = createVelocityContext(parameters);

if (pathToTemplateFile.equals("general-report-file-template.vm")) {

pathToReportFile = pathToReportFile + Constants.BACKSLASH + "GENERAL\_REPORT.html";

}

File reportFile = new File(pathToReportFile);

Writer writer = null;

try {

writer = new OutputStreamWriter(new FileOutputStream(reportFile), StandardCharsets.UTF\_8);

template.merge(context, writer);

} catch (FileNotFoundException e) {

e.printStackTrace();

} finally {

if (writer != null) {

try {

writer.close();

} catch (IOException e) {

log.warn(e);

}

}

}

}

private static Template loadVelocityTemplate(String pathToTemplateFile) {

VelocityEngine velocityEngine = new VelocityEngine();

velocityEngine.setProperty(RuntimeConstants.RESOURCE\_LOADER, "classpath");

velocityEngine.setProperty("classpath.resource.loader.class", ClasspathResourceLoader.class.getName());

velocityEngine.setProperty("runtime.log.system.class", "org.apache.velocity.runtime.log.NullLogChute");

velocityEngine.init();

return velocityEngine.getTemplate(pathToTemplateFile);

}

private static VelocityContext createVelocityContext(Map<String, Object> parameters) {

VelocityContext context = new VelocityContext();

for (Map.Entry<String, Object> parameter : parameters.entrySet()) {

context.put(parameter.getKey(), parameter.getValue());

}

return context;

}

}  
  
**Пакет: vulnerabilities**

public class VulnerabilityAnalyzerToolClient {

public static void main(String[] args) {

Map<InputParametersPattern, String> parameters = ParametersParser.parseInputParameters(args);

String sourceCodeAbsolutePath = FileSystems.getDefault()

.getPath(parameters.get(InputParametersPattern.SOURCE\_CODE\_PATH))

.normalize().toAbsolutePath().toString();

String reportFolderAbsolutePath = FileSystems.getDefault()

.getPath(parameters.get(InputParametersPattern.REPORT\_FOLDER\_PATH))

.normalize().toAbsolutePath().toString();

VulnerabilityAnalyzerLauncher launcher = new VulnerabilityAnalyzerLauncher(

Arrays.stream(parameters.get(InputParametersPattern.VULNERABILITIES\_LIST).split(", ")).toList()

, sourceCodeAbsolutePath

, reportFolderAbsolutePath);

launcher.execute();

}

}  
  
public class VulnerabilityFactory {

public Vulnerability getVulnerabilityInstance(String vulnerabilityName) {

switch (vulnerabilityName) {

case "bufferOverflow":

return new BufferOverflowVulnerability();

case "formatStringError":

return new FormatStringErrorVulnerability();

case "incorrectFileAccess":

return new IncorrectFileAccessVulnerability();

case "informationLeak":

return new InformationLeakVulnerability();

case "injectionCommands":

return new InjectionCommandsVulnerability();

case "injectionSQLCommands":

return new InjectionSQLCommandsVulnerability();

case "integerOverflow":

return new IntegerOverflowVulnerability();

case "neglectErrorHandling":

return new NeglectErrorHandlingVulnerability();

case "neglectSecureDataStorage":

return new NeglectSecureDataStorageVulnerability();

case "racing":

return new RacingVulnerability();

case "randomCryptographicNumbers":

return new RandomCryptographicNumbersVulnerability();

case "readersWriters":

return new ReadersWritersVulnerability();

default:

throw new IllegalArgumentException("temp");

}

}

}  
  
public abstract class Vulnerability {

private final MongoService mongoService = new MongoService();

public abstract List<ReportInformation> checkRules(List<String> sourceCode);

public List<Rule> initializeRules(Object object) {

String vulnerabilityName = object.getClass().getSimpleName();

return mongoService.getRulesByVulnerability(vulnerabilityName);

}

}  
  
**Пакет: vulnerabilities.impl**

public class BufferOverflowVulnerability extends Vulnerability {

private final List<Rule> validationRules;

public BufferOverflowVulnerability() {

validationRules = initializeRules(this);

}

public BufferOverflowVulnerability(List<Rule> validationRules) {

this.validationRules = validationRules;

}

public List<Rule> getValidationRules() {

return validationRules;

}

@Override

public List<ReportInformation> checkRules(List<String> sourceCode) {

List<ReportInformation> result = new ArrayList<>();

for (Rule curRule : validationRules) {

Pattern pattern = Pattern.compile(curRule.getMask());

Matcher matcher;

int codeStringNumber = 0;

for (String curCodeString : sourceCode) {

++codeStringNumber;

String curStringWithoutTab = curCodeString.replace("\t", "");

curStringWithoutTab = curStringWithoutTab.trim();

matcher = pattern.matcher(curStringWithoutTab);

if (matcher.find()) {

if (curRule.getRuleName().equals("Static array definition ")

&& (curStringWithoutTab.contains("return")) || (curStringWithoutTab.contains("-"))) {

break;

} else {

ReportInformation reportInformation = new ReportInformation(

codeStringNumber + ": " + curStringWithoutTab

, Constants.BUFFER\_OVERFLOW

, curRule

, codeStringNumber);

result.add(reportInformation);

}

}

}

}

return result;

}

}  
  
public class FormatStringErrorVulnerability extends Vulnerability {

private final List<Rule> validationRules;

public FormatStringErrorVulnerability() {

validationRules = initializeRules(this);

}

public List<Rule> getValidationRules() {

return validationRules;

}

@Override

public List<ReportInformation> checkRules(List<String> sourceCode) {

List<ReportInformation> result = new ArrayList<>();

for (Rule curRule : validationRules) {

Pattern pattern = Pattern.compile(curRule.getMask());

Matcher matcher;

int codeStringNumber = 0;

for (String curCodeString : sourceCode) {

++codeStringNumber;

String curStringWithoutTab = curCodeString.replace("\t", "");

curStringWithoutTab = curStringWithoutTab.trim();

matcher = pattern.matcher(curStringWithoutTab);

if (matcher.find()) {

ReportInformation reportInformation = new ReportInformation(

codeStringNumber + ": " + curStringWithoutTab

, Constants.FORMAT\_STRING\_ERROR

, curRule

, codeStringNumber);

result.add(reportInformation);

}

}

}

return result;

}

}  
  
public class IncorrectFileAccessVulnerability extends Vulnerability {

private final List<Rule> validationRules;

public IncorrectFileAccessVulnerability() {

validationRules = initializeRules(this);

}

public List<Rule> getValidationRules() {

return validationRules;

}

@Override

public List<ReportInformation> checkRules(List<String> sourceCode) {

String realPathRegex = "(\\W)realpath\\(";

String openRegex = "(\\W)open\\(";

List<ReportInformation> result = new ArrayList<>();

for (Rule curRule : validationRules) {

Pattern pattern = Pattern.compile(curRule.getMask());

Matcher matcher;

int codeStringNumber = 0;

if (curRule.getRuleName().equals("Create or open file")) {

for (String curCodeString : sourceCode) {

++codeStringNumber;

String curStringWithoutTab = curCodeString.replace("\t", "");

matcher = pattern.matcher(curStringWithoutTab);

if (matcher.find()) {

// Create or Open file method founded

// Let's search realpath() method

boolean realPathMethodFounded = false;

Pattern patternRealPath = Pattern.compile(realPathRegex);

Matcher matcherRealPath;

for (String codeStringForFindRealPath : sourceCode) {

String codeStringForFindRealPathWithoutTab = codeStringForFindRealPath.replace("\t", "");

matcherRealPath = patternRealPath.matcher(codeStringForFindRealPathWithoutTab);

if (matcherRealPath.find()) {

// realpath() founded

realPathMethodFounded = true;

break;

}

}

// if realpath() method not founded -> it's VULNERABILITY

if (!realPathMethodFounded) {

ReportInformation reportInformation = new ReportInformation(

codeStringNumber + ": " + curStringWithoutTab

, Constants.INCORRECT\_FILE\_ACCESS

, curRule

, codeStringNumber);

result.add(reportInformation);

}

}

}

}

else if (curRule.getRuleName().equals("Access + Open file")) {

for (String curCodeString : sourceCode) {

++codeStringNumber;

String curStringWithoutTab = curCodeString.replace("\t", "");

matcher = pattern.matcher(curStringWithoutTab);

if (matcher.find()) {

// access() file method founded

// Let's search open() method

boolean openMethodFounded = false;

Pattern patternOpenMethod = Pattern.compile(openRegex);

Matcher matcherOpenMethod;

for (String codeStringForFindOpenMethod : sourceCode) {

String codeStringForFindOpenMethodWithoutTab = codeStringForFindOpenMethod.replace("\t", "");

matcherOpenMethod = patternOpenMethod.matcher(codeStringForFindOpenMethodWithoutTab);

if (matcherOpenMethod.find()) {

// open() founded

openMethodFounded = true;

break;

}

}

// if open() method founded -> it's VULNERABILITY

if (openMethodFounded) {

ReportInformation reportInformation = new ReportInformation(

codeStringNumber + ": " + curStringWithoutTab

, Constants.INCORRECT\_FILE\_ACCESS

, curRule

, codeStringNumber);

result.add(reportInformation);

}

}

}

}

else {

for (String curCodeString : sourceCode) {

++codeStringNumber;

String curStringWithoutTab = curCodeString.replace("\t", "");

matcher = pattern.matcher(curStringWithoutTab);

if (matcher.find()) {

ReportInformation reportInformation = new ReportInformation(

codeStringNumber + ": " + curStringWithoutTab

, Constants.INCORRECT\_FILE\_ACCESS

, curRule

, codeStringNumber);

result.add(reportInformation);

}

}

}

}

return result;

}

/\*public void createRules() {

List<Rule> rules = Arrays.asList(

new Rule("Create or open file"

, "(\\W)(CreateFile|OpenFile)\\("

, "Create or open file without checking file path via realpath() function"

, "Better use char\* realpath(const char \*original\_path, char resolved\_path[PATH\_MAX]) for check suspicious path to file"),

new Rule("Access + Open file"

, "(\\W)access\\("

, "Access and Open file methods in one thread"

, "Make sure you have one thread while opening a file like this!")

);

}\*/

}  
  
public class InformationLeakVulnerability extends Vulnerability {

private final List<Rule> validationRules;

public InformationLeakVulnerability() {

validationRules = initializeRules(this);

}

public List<Rule> getValidationRules() {

return validationRules;

}

@Override

public List<ReportInformation> checkRules(List<String> sourceCode) {

List<ReportInformation> result = new ArrayList<>();

for (Rule curRule : validationRules) {

Pattern pattern = Pattern.compile(curRule.getMask());

Matcher matcher;

int codeStringNumber = 0;

for (String curCodeString : sourceCode) {

++codeStringNumber;

String curStringWithoutTab = curCodeString.replace("\t", "");

matcher = pattern.matcher(curStringWithoutTab);

if (matcher.find()) {

ReportInformation reportInformation = new ReportInformation(

codeStringNumber + ": " + curStringWithoutTab

, Constants.INFORMATION\_LEAK

, curRule

, codeStringNumber);

result.add(reportInformation);

}

}

}

return result;

}

}  
  
public class InjectionCommandsVulnerability extends Vulnerability {

private final List<Rule> validationRules;

public InjectionCommandsVulnerability() {

validationRules = initializeRules(this);

}

public List<Rule> getValidationRules() {

return validationRules;

}

@Override

public List<ReportInformation> checkRules(List<String> sourceCode) {

List<ReportInformation> result = new ArrayList<>();

for (Rule curRule : validationRules) {

Pattern pattern = Pattern.compile(curRule.getMask());

Matcher matcher;

int codeStringNumber = 0;

for (String curCodeString : sourceCode) {

++codeStringNumber;

String curStringWithoutTab = curCodeString.replace("\t", "");

matcher = pattern.matcher(curStringWithoutTab);

if (matcher.find()) {

ReportInformation reportInformation = new ReportInformation(

codeStringNumber + ": " + curStringWithoutTab

, Constants.INJECTION\_COMMANDS\_ERROR

, curRule

, codeStringNumber);

result.add(reportInformation);

}

}

}

return result;

}

}  
  
public class RandomCryptographicNumbersVulnerability extends Vulnerability {

private final List<Rule> validationRules;

public RandomCryptographicNumbersVulnerability() {

validationRules = initializeRules(this);

}

public List<Rule> getValidationRules() {

return validationRules;

}

@Override

public List<ReportInformation> checkRules(List<String> sourceCode) {

List<ReportInformation> result = new ArrayList<>();

for (Rule curRule : validationRules) {

boolean srandFounded = false;

if (curRule.getRuleName().equals("rand() without srand(...)")) {

String findSrandRegex = "(\\W|)srand\\(.\*?\\)";

String findBraceStart = "\\{";

String findBraceEnd = "\\}";

String findMainRegex = "int main\\(.\*?\\{";

int braceCounter = 1;

int codeStringNumber = 0;

List<String> mainCode = new ArrayList<>();

Pattern pattern = Pattern.compile(findMainRegex);

Matcher matcher;

boolean isMain = false;

for (String curCodeString : sourceCode) {

if (!isMain) { // find start of main

++codeStringNumber;

String curStringWithoutTab = curCodeString.replace("\t", "");

matcher = pattern.matcher(curStringWithoutTab);

if (matcher.find()) {

mainCode.add(curStringWithoutTab);

isMain = true;

continue;

}

}

else { // start of main founded => find all code of main thanks to '{' and '}'

if (braceCounter == 0) { // all main code founded => find srand(...) function

for (String mainCodeString : mainCode) {

pattern = Pattern.compile(findSrandRegex);

matcher = pattern.matcher(mainCodeString);

if (matcher.find()) {

srandFounded = true;

break;

}

}

if (srandFounded) break;

}

++codeStringNumber;

String curStringWithoutTab = curCodeString.replace("\t", "");

mainCode.add(curStringWithoutTab);

pattern = Pattern.compile(findBraceStart);

matcher = pattern.matcher(curStringWithoutTab);

if (matcher.find()) {

++braceCounter;

}

else {

pattern = Pattern.compile(findBraceEnd);

matcher = pattern.matcher(curStringWithoutTab);

if (matcher.find()) {

--braceCounter;

}

}

}

}

if (!srandFounded) {

codeStringNumber = 0;

for (String curCodeString : sourceCode) {

++codeStringNumber;

String curStringWithoutTab = curCodeString.replace("\t", "");

pattern = Pattern.compile(curRule.getMask());

matcher = pattern.matcher(curStringWithoutTab);

if (matcher.find()) {

ReportInformation reportInformation = new ReportInformation(

codeStringNumber + ": " + curStringWithoutTab

, Constants.RANDOM\_CRYPTO\_NUMBERS

, curRule

, codeStringNumber);

result.add(reportInformation);

}

}

}

} else {

Pattern pattern = Pattern.compile(curRule.getMask());

Matcher matcher;

int codeStringNumber = 0;

for (String curCodeString : sourceCode) {

++codeStringNumber;

String curStringWithoutTab = curCodeString.replace("\t", "");

matcher = pattern.matcher(curStringWithoutTab);

if (matcher.find()) {

ReportInformation reportInformation = new ReportInformation(

codeStringNumber + ": " + curStringWithoutTab

, Constants.RANDOM\_CRYPTO\_NUMBERS

, curRule

, codeStringNumber);

result.add(reportInformation);

}

}

}

}

return result;

}

public void createRules() {

List<Rule> rules = Arrays.asList(

new Rule("Seed std:time"

, ".\*(\\.seed){1}.\*(time){1}.\*"

, "Possible incompatibility of operating systems"

, "Better use std::random\_device as seed"),

new Rule("Seed constant"

, ".\*((\\.seed|srand|drand48|erand48|jrand48|lrand48|nrand48|seed48)\\()[0-9]+\\)"

, "Constant as seed"

, "Better use non constant value as seed"),

new Rule("rand() without srand(...)"

, "(\\W)rand\\(\\)"

, "Using rand() without calling srand(...) can result in identical output sequences"

, "Better use srand(unsigned int seed) before call rand()")

);

}

// std::mt19937 engine; // mt19937 как один из вариантов

// engine.seed(std::time(nullptr));

// /\*

// на случай, если у вас UNIX-чё-то или компилятор не MinGW

// std::random\_device device;

// engine.seed(device());

}  
  
public class ReadersWritersVulnerability extends Vulnerability {

private static final String OMP\_LOCK\_T\_PATTERN = "omp\_lock\_t";

private final List<Rule> validationRules;

public ReadersWritersVulnerability() {

validationRules = initializeRules(this);

}

public List<Rule> getValidationRules() {

return validationRules;

}

@Override

public List<ReportInformation> checkRules(List<String> sourceCode) {

List<ReportInformation> result = new ArrayList<>();

for (Rule curRule : validationRules) {

Pattern pattern = Pattern.compile(curRule.getMask());

Matcher matcher;

int codeStringNumber = 0;

for (String curCodeString : sourceCode) {

++codeStringNumber;

String curStringWithoutTab = curCodeString.replace("\t", "");

curStringWithoutTab = curStringWithoutTab.trim();

matcher = pattern.matcher(curStringWithoutTab);

if (matcher.find()) {

if (!ompLockFinder(sourceCode)) {

ReportInformation reportInformation = new ReportInformation(

codeStringNumber + ": " + curStringWithoutTab

, Constants.READERS\_WRITERS

, curRule

, codeStringNumber);

result.add(reportInformation);

}

}

}

}

return result;

}

private boolean ompLockFinder(List<String> sourceCode) {

Pattern ompLockPattern = Pattern.compile(OMP\_LOCK\_T\_PATTERN);

Matcher ompMatcher;

for (String curCodeString : sourceCode) {

ompMatcher = ompLockPattern.matcher(curCodeString);

if (ompMatcher.find()) {

return true;

}

}

return false;

}

}  
  
**Пакет: vulnerabilities.rule**

public class Rule {

private String ruleName;

private String mask;

private String vulnerabilityMessage;

private String fixMessage;

public Rule(String ruleName, String mask, String vulnerabilityMessage, String fixMessage) {

this.ruleName = ruleName;

this.mask = mask;

this.vulnerabilityMessage = vulnerabilityMessage;

this.fixMessage = fixMessage;

}

public Rule() {

}

public String getRuleName() {

return ruleName;

}

public String getMask() {

return mask;

}

public String getVulnerabilityMessage() {

return vulnerabilityMessage;

}

public String getFixMessage() {

return fixMessage;

}

public void setRuleName(String ruleName) {

this.ruleName = ruleName;

}

public void setMask(String mask) {

this.mask = mask;

}

public void setVulnerabilityMessage(String vulnerabilityMessage) {

this.vulnerabilityMessage = vulnerabilityMessage;

}

public void setFixMessage(String fixMessage) {

this.fixMessage = fixMessage;

}

@Override

public String toString() {

return "\n\tRule {"

+ "\n\t\truleName = " + ruleName

+ ", \n\t\tmask = " + mask

+ ", \n\t\tvulnerabilityMessage = " + vulnerabilityMessage

+ ", \n\t\tfixMessage = " + fixMessage

+ "\n\t}";

}

}  
  
**VML файл: general-report-file-body-template.vm**

#addBodyPart($reportInformationList)

##

#macro(addBodyPart $reportInformationList)

#getTableBodyHeaderRow($headers)

#set($rowCounter = $reportInformationList.size() + 1)

#addBodyLines(1, $reportInformationList)

#end

##

#macro(addBodyLines $firstRowNumber $reportInformationList)

#foreach($reportInformation in $reportInformationList)

#@addTableRow()

#addTableEntry("" "" $firstRowNumber "bold")

#addGeneralTableEntry("" "" $reportInformation.getPathToFile() $reportInformation.getPathToReportFile() "bold")

#if($reportInformation.getValidationPassed())

#addTableEntry($SUCCESS\_COLOR "" "TRUE" "bold")

#else

#addTableEntry($FAIL\_COLOR "" "FALSE" "bold")

#end

#set($firstRowNumber = $firstRowNumber + 1)

#end

#end

#end

##

**VML файл: general-report-file-template.vm**

#set($headers = ["#", "PATH TO SOURCE FILE", "IS VALIDATION PASSED"])

#parse("macros.vm")

<html>

<head>

<meta http-equiv="Content-Type" context="text/html; charset=UTF-8">

</head>

<body>

<table width="100%" border="1" cellpadding="3">

#addTableHeaderRow(true "" "Vulnerability Analyzer Report")

#if($generalAnalysisPassed)

#addTableHeaderWholeRow($SUCCESS\_COLOR "Vulnerabilities not found")

#else

#addTableHeaderRow(false "$FAIL\_COLOR" "Vulnerabilities found")

#end

#parse("general-report-headers.vm")

#parse("general-report-file-body-template.vm")

</table>

</body>

</html>  
  
**VML файл: general-report-headers.vm**

#addTableHeaderRow(false "" "Date" $validationDate)

#addTableHeaderRow(false "" "Path to Source Folder" $pathToSourceFile)

#addTableHeaderRow(false "" "Path to Vulnerability Report Folder" $pathToReportFile)

**VML файл: macros.vm**

#set($DELIMITER\_COLOR = "#C0C0C0")

#set($DOT = ".")

#set($ES = "")

#set($FAIL\_COLOR = "#FFAAAA")

#set($LEFT\_TABLE\_PART\_COL\_COUNT = 3)

#set($NS = "#")

#set($TOTAL\_TABLE\_COL\_COUNT = 5)

#set($RIGHT\_TABLE\_PART\_COL\_COUNT = $TOTAL\_TABLE\_COL\_COUNT - $LEFT\_TABLE\_PART\_COL\_COUNT)

#set($SUCCESS\_COLOR = "#B8E68A")

#set($WARNING\_COLOR = "#FFFFB0")

##

##

#macro(addTableEntry $bgcolor $align $content $font)

<th bgcolor="$!bgcolor">

<div align="$!align"><span style="font-weight: $!font;">$!content</span></div>

</th>

#end

##

#macro(addGeneralTableEntry $bgcolor $align $content $linktext $font $width)

<th bgcolor="$!bgcolor">

<a href="$linktext"><div align="$!align"><span style="font-weight: $!font;">$!content</span></div></a>

</th>

#end

##

#macro(addTableHeader $bgcolor $colspan $content)

<th bgcolor="$!bgcolor" colspan="$!colspan">

$!content

</th>

#end

##

#macro(makeBold $content)

<b>

$!content

</b>

#end

##

#macro(addTableData $bgcolor $colspan $context)

<td bgcolor="$!bgcolor" colspan="$!colspan">

$!context

</td>

#end

##

#macro(addTableRow $id)

<tr id="$!id">

$!bodyContent

</tr>

#end

##

#macro(addTableHeaderWholeRow $bgcolor $content)

<tr>

<th bgcolor="$!bgcolor" colspan=$TOTAL\_TABLE\_COL\_COUNT>

$!content

</th>

</tr>

#end

##

#macro(getTableBodyHeaderRow $headers)

#@addTableRow()

#foreach($header in $headers)

#addTableHeader("" "" $header)

#end

#end

#end

##

#macro(addTableHeaderRow $isWholeRow $bgcolor $rowName $rowValue $rowID)

#if($isWholeRow)

#addTableHeaderWholeRow($bgcolor $rowName)

#else

#@addTableRow($rowID)

#if($bgcolor == "")

#addTableData("" $LEFT\_TABLE\_PART\_COL\_COUNT "#makeBold($rowName)")

#else

#addTableHeader($bgcolor $LEFT\_TABLE\_PART\_COL\_COUNT $rowName)

#end

#addTableData("" $RIGHT\_TABLE\_PART\_COL\_COUNT $rowValue)

#end

#end

#end

##  
  
**VML файл: report-file-body-template.vm**

#addBodyPart($reportInformationList)

##

#macro(addBodyPart $reportInformationList)

#getTableBodyHeaderRow($headers)

#set($rowCounter = $reportInformationList.size() + 1)

#addBodyLines(1, $reportInformationList)

#end

##

#macro(addBodyLines $firstRowNumber $reportInformationList)

#foreach($reportInformation in $reportInformationList)

#@addTableRow()

#addTableEntry("" "" $firstRowNumber "bold")

#addTableEntry("" "" $reportInformation.getVulnerabilityName() "bold")

#addTableEntry("" "left" $reportInformation.getSourceCodeString() "normal")

#addTableEntry("" "" $reportInformation.getRule().getVulnerabilityMessage() "normal")

#addTableEntry("" "" $reportInformation.getRule().getFixMessage() "normal")

#set($firstRowNumber = $firstRowNumber + 1)

#end

#end

#end

##  
  
**VML файл: report-file-headers.vm**

#addTableHeaderRow(false "" "Date" $validationDate)

#addTableHeaderRow(false "" "Path to Source File" $pathToSourceFile)

#addTableHeaderRow(false "" "Path to Vulnerability Report" $pathToReportFile)

**VML файл: report-file-template.vm**

#set($headers = ["#", "VULNERABILITY", "SOURCE CODE STRING", "VULNERABILITY INFORMATION", "FIX MESSAGE"])

#parse("macros.vm")

<html>

<head>

<meta http-equiv="Content-Type" context="text/html; charset=UTF-8">

</head>

<body>

<table width="100%" border="1" cellpadding="3">

#addTableHeaderRow(true "" "Vulnerability Analyzer Report")

#if($generalAnalysisPassed)

#addTableHeaderWholeRow($SUCCESS\_COLOR "Vulnerabilities not found")

#else

#addTableHeaderRow(false "$FAIL\_COLOR" "Vulnerabilities found")

#end

#parse("report-file-headers.vm")

#parse("report-file-body-template.vm")

</table>

</body>

</html>  
  
**Тестовые классы:**

public class FileHelperTest extends BaseTest {

@Test

public void testValidateInputFilePositive() {

FileHelper.readDataFromCppFile("src/test/resources/fileHelper/validate\_file\_positive.cpp");

}

@Test

public void testValidateInputFileNegative() {

exceptionRule.expect(IllegalArgumentException.class);

exceptionRule.expectMessage("Invalid file path or extension\nExpected extension .cpp");

FileHelper.readDataFromCppFile("src/test/resources/fileHelper/validate\_file\_positive.json");

}

}  
  
public class BaseTest {

@Rule

public ExpectedException exceptionRule = ExpectedException.none();

}  
  
**Модуль gui:**

public class HelloApplication extends Application {

@Override

public void start(Stage stage) throws IOException {

FXMLLoader fxmlLoader = new FXMLLoader(HelloApplication.class.getResource("hello-view.fxml"));

Scene scene = new Scene(fxmlLoader.load(), 650, 450);

stage.setTitle("Vulnerability Analyzer");

stage.setScene(scene);

stage.show();

}

public static void main(String[] args) {

launch();

}

}

public class Controller {

@FXML

private ResourceBundle resources;

@FXML

private URL location;

@FXML

private Button buttonDirectoryChoose;

@FXML

private Button buttonFileChoose;

@FXML

private Button buttonStart;

@FXML

private TextArea codeCpp;

@FXML

private TextField pathChoosenDirectoryForSave;

@FXML

private TextField pathChoosenFile;

@FXML

private CheckBox bufferOverflow;

@FXML

private CheckBox formatString;

@FXML

private CheckBox incorrectFile;

@FXML

private CheckBox informationLeak;

@FXML

private CheckBox injectionCommands;

@FXML

private CheckBox injectionSQL;

@FXML

private CheckBox integerOverflow;

@FXML

private CheckBox neglectErrorHandling;

@FXML

private CheckBox negletSecureData;

@FXML

private CheckBox racing;

@FXML

private CheckBox randomCryptograph;

@FXML

private CheckBox readersWriters;

public Controller() {

}

@FXML

void initialize() {

}

List<String> list = new ArrayList<>();

public List<String> getList() {

return list;

}

@FXML

private void boxBufferOverflow() {

if (bufferOverflow.isSelected()){

list.add("bufferOverflow");

System.out.println(list);

} else {

list.removeIf(list -> list.equals("bufferOverflow"));

System.out.println(list);

}

}

@FXML

private void boxReadersWriters() {

if (readersWriters.isSelected()){

list.add("readersWriters");

System.out.println(list);

} else {

list.removeIf(list -> list.equals("readersWriters"));

System.out.println(list);

}

}

@FXML

private void boxRandomCryptographicNumbers() {

if (randomCryptograph.isSelected()){

list.add("randomCryptographicNumbers");

System.out.println(list);

} else {

list.removeIf(list -> list.equals("randomCryptographicNumbers"));

System.out.println(list);

}

}

@FXML

private void boxRacing() {

if (racing.isSelected()){

list.add("racing");

System.out.println(list);

} else {

list.removeIf(list -> list.equals("racing"));

System.out.println(list);

}

}

@FXML

private void boxNeglectSecureDataStorage() {

if (negletSecureData.isSelected()){

list.add("neglectSecureDataStorage");

System.out.println(list);

} else {

list.removeIf(list -> list.equals("neglectSecureDataStorage"));

System.out.println(list);

}

}

@FXML

private void boxNeglectErrorHandling() {

if (neglectErrorHandling.isSelected()){

list.add("neglectErrorHandling");

System.out.println(list);

} else {

list.removeIf(list -> list.equals("neglectErrorHandling"));

System.out.println(list);

}

}

@FXML

private void boxIntegerOverflow() {

if (integerOverflow.isSelected()){

list.add("integerOverflow");

System.out.println(list);

} else {

list.removeIf(list -> list.equals("integerOverflow"));

System.out.println(list);

}

}

@FXML

private void boxInjectionSQLCommands() {

if (injectionSQL.isSelected()){

list.add("injectionSQLCommands");

System.out.println(list);

} else {

list.removeIf(list -> list.equals("injectionSQLCommands"));

System.out.println(list);

}

}

@FXML

private void boxInjectionCommands() {

if (injectionCommands.isSelected()){

list.add("injectionCommands");

System.out.println(list);

} else {

list.removeIf(list -> list.equals("injectionCommands"));

System.out.println(list);

}

}

@FXML

private void boxInformationLeak() {

if (informationLeak.isSelected()){

list.add("informationLeak");

System.out.println(list);

} else {

list.removeIf(list -> list.equals("informationLeak"));

System.out.println(list);

}

}

@FXML

private void boxIncorrectFileAccess() {

if (incorrectFile.isSelected()){

list.add("incorrectFileAccess");

System.out.println(list);

} else {

list.removeIf(list -> list.equals("incorrectFileAccess"));

System.out.println(list);

}

}

@FXML

private void boxFormatStringError() {

if (formatString.isSelected()){

list.add("formatStringError");

System.out.println(list);

} else {

list.removeIf(list -> list.equals("formatStringError"));

System.out.println(list);

}

}

private void printLog(TextField textField, List<File> files) {

if (files == null || files.isEmpty()) {

return;

}

for (File file : files) {

textField.appendText(file.getAbsolutePath() + "\n");

}

}

private String filePath;

public String getFilePath() {

return filePath;

}

public void fileChoosen(ActionEvent actionEvent) throws IOException {

FileChooser fileChooser = new FileChooser();

Stage primaryStage = new Stage();

pathChoosenFile.clear();

fileChooser.setInitialDirectory(new File("C:\\"));

fileChooser.setTitle("Select cpp file: ");

fileChooser.getExtensionFilters().add(new FileChooser.ExtensionFilter(".cpp", "\*.cpp"));

File file = fileChooser.showOpenDialog(primaryStage);

if (file != null) {

List<File> files = List.of(file);

printLog(pathChoosenFile, files);

filePath = file.getAbsolutePath();

}

System.out.println(filePath);

printToCodeCpp();

}

@FXML

public void dirChoosen(ActionEvent actionEvent) throws IOException {

DirectoryChooser directoryChooser = new DirectoryChooser();

Stage primaryStage = new Stage();

pathChoosenFile.clear();

directoryChooser.setInitialDirectory(new File("C:\\"));

directoryChooser.setTitle("Select directory: ");

File file = directoryChooser.showDialog(primaryStage);

if (file != null) {

List<File> files = List.of(file);

printLog(pathChoosenFile, files);

filePath = file.getAbsolutePath();

}

System.out.println(filePath);

}

private void printToCodeCpp() throws IOException {

codeCpp.clear();

BufferedReader br = new BufferedReader(new FileReader(filePath));

String line = null;

while ((line = br.readLine()) != null) {

codeCpp.appendText(line + '\n');

}

}

private String directoryPath;

public String getDirectoryPath() {

return directoryPath;

}

public void directoryChoosen(ActionEvent actionEvent) {

DirectoryChooser directoryChooser = new DirectoryChooser();

Stage primaryStage = new Stage();

pathChoosenDirectoryForSave.clear();

directoryChooser.setInitialDirectory(new File("C:\\"));

directoryChooser.setTitle("Select directory for save: ");

File file = directoryChooser.showDialog(primaryStage);

if (file != null) {

List<File> files = List.of(file);

printLog(pathChoosenDirectoryForSave, files);

directoryPath = file.getAbsolutePath();

}

System.out.println(directoryPath);

}

public void startAnalyzer(ActionEvent actionEvent) { // 90 строка - list с уязвимостями, 236 путь к файлу .cpp или директории, 282 - директория для сохранения

VulnerabilityAnalyzerLauncher launcher = new VulnerabilityAnalyzerLauncher(getList(), getFilePath(), getDirectoryPath());

launcher.execute();

System.out.println("Done");

}

}  
  
**Ресуры:**

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.control.Button?>

<?import javafx.scene.control.CheckBox?>

<?import javafx.scene.control.Label?>

<?import javafx.scene.control.TextArea?>

<?import javafx.scene.control.TextField?>

<?import javafx.scene.layout.AnchorPane?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity" minWidth="-Infinity" prefHeight="463.0" prefWidth="684.0" xmlns="http://javafx.com/javafx/17" xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.example.gui\_for\_course.Controller">

<children>

<AnchorPane layoutX="373.0" layoutY="115.0" prefHeight="203.0" prefWidth="267.0">

<children>

<CheckBox fx:id="readersWriters" layoutX="155.0" layoutY="164.0" mnemonicParsing="false" onAction="#boxReadersWriters" prefHeight="17.0" prefWidth="105.0" text="ReadersWriters" />

<CheckBox fx:id="randomCryptograph" layoutX="155.0" layoutY="139.0" mnemonicParsing="false" onAction="#boxRandomCryptographicNumbers" prefHeight="17.0" prefWidth="144.0" text="RandomCryptograph" />

<CheckBox fx:id="racing" layoutX="155.0" layoutY="113.0" mnemonicParsing="false" onAction="#boxRacing" prefHeight="17.0" prefWidth="98.0" text="Racing" />

<CheckBox fx:id="negletSecureData" layoutX="155.0" layoutY="87.0" mnemonicParsing="false" onAction="#boxNeglectSecureDataStorage" prefHeight="17.0" prefWidth="131.0" text="NeglectSecureData" />

<CheckBox fx:id="neglectErrorHandling" layoutX="155.0" layoutY="60.0" mnemonicParsing="false" onAction="#boxNeglectErrorHandling" prefHeight="17.0" prefWidth="144.0" text="NeglectErrorHandling" />

<CheckBox fx:id="integerOverflow" layoutX="155.0" layoutY="34.0" mnemonicParsing="false" onAction="#boxIntegerOverflow" prefHeight="17.0" prefWidth="114.0" text="IntegerOverflow" />

<CheckBox fx:id="bufferOverflow" layoutX="14.0" layoutY="38.0" mnemonicParsing="false" onAction="#boxBufferOverflow" prefHeight="17.0" prefWidth="105.0" text="BufferOverflow" />

<CheckBox fx:id="formatString" layoutX="14.0" layoutY="64.0" mnemonicParsing="false" onAction="#boxFormatStringError" prefHeight="17.0" prefWidth="98.0" text="FormatString" />

<CheckBox fx:id="incorrectFile" layoutX="14.0" layoutY="91.0" mnemonicParsing="false" onAction="#boxIncorrectFileAccess" prefHeight="17.0" prefWidth="98.0" text="IncorrectFile" />

<CheckBox fx:id="informationLeak" layoutX="14.0" layoutY="117.0" mnemonicParsing="false" onAction="#boxInformationLeak" prefHeight="17.0" prefWidth="114.0" text="InformationLeak" />

<CheckBox fx:id="injectionCommands" layoutX="14.0" layoutY="143.0" mnemonicParsing="false" onAction="#boxInjectionCommands" prefHeight="17.0" prefWidth="131.0" text="InjectionCommands" />

<CheckBox fx:id="injectionSQL" layoutX="14.0" layoutY="168.0" mnemonicParsing="false" onAction="#boxInjectionSQLCommands" prefHeight="17.0" prefWidth="98.0" text="InjectionSQL" />

</children>

</AnchorPane>

<TextArea fx:id="codeCpp" layoutX="21.0" layoutY="154.0" prefHeight="239.0" prefWidth="299.0" promptText="You'r code" />

<Label layoutX="373.0" layoutY="318.0" prefHeight="103.0" prefWidth="244.0" text="Анализатор уязвимостей" />

<TextField fx:id="pathChoosenFile" layoutX="21.0" layoutY="42.0" prefHeight="25.0" prefWidth="231.0" promptText="Choose cpp file" />

<TextField fx:id="pathChoosenDirectoryForSave" layoutX="21.0" layoutY="83.0" prefHeight="25.0" prefWidth="231.0" promptText="Choose directory from save" />

<Button fx:id="buttonStart" layoutX="402.0" layoutY="14.0" mnemonicParsing="false" onAction="#startAnalyzer" prefHeight="103.0" prefWidth="215.0" style="-fx-background-color: green;" text="Start analysis" textFill="#ffffff" />

<Button fx:id="buttonFileChoose" layoutX="260.0" layoutY="42.0" mnemonicParsing="false" onAction="#fileChoosen" prefHeight="25.0" prefWidth="60.0" text="File" />

<Button fx:id="buttonDirectoryChoose" layoutX="260.0" layoutY="83.0" mnemonicParsing="false" onAction="#directoryChoosen" prefHeight="25.0" prefWidth="60.0" text="Click" />

<Button fx:id="buttonDirectoryChoose\_1" layoutX="328.0" layoutY="42.0" mnemonicParsing="false" onAction="#dirChoosen" prefHeight="25.0" prefWidth="60.0" text="Dir" />

</children>

</AnchorPane>

**pom.xml файл модуля analyzer-library:**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<parent>

<artifactId>vulnerability\_analyzer</artifactId>

<groupId>groupId</groupId>

<version>1.0-SNAPSHOT</version>

</parent>

<modelVersion>4.0.0</modelVersion>

<artifactId>analyzer-library</artifactId>

<properties>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

<build>

<plugins>

<plugin>

<artifactId>maven-assembly-plugin</artifactId>

<executions>

<execution>

<phase>package</phase>

<goals>

<goal>single</goal>

</goals>

</execution>

</executions>

<configuration>

<!-- ... -->

<archive>

<manifest>

<mainClass>VulnerabilityAnalyzerToolClient</mainClass>

</manifest>

</archive>

<descriptorRefs>

<descriptorRef>jar-with-dependencies</descriptorRef>

</descriptorRefs>

<outputDirectory>${project.build.directory}/execute</outputDirectory>

</configuration>

</plugin>

</plugins>

</build>

<dependencies>

<dependency>

<groupId>org.apache.velocity</groupId>

<artifactId>velocity</artifactId>

<version>1.7</version>

</dependency>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.17</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mockito</groupId>

<artifactId>mockito-all</artifactId>

<version>1.10.18</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.mongodb</groupId>

<artifactId>mongo-java-driver</artifactId>

<version>3.12.10</version>

</dependency>

<dependency>

<groupId>org.apache.mahout</groupId>

<artifactId>mahout-math</artifactId>

<version>0.13.0</version>

</dependency>

</dependencies>

</project>

**pom.xml файл модуля functional-tests:**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<parent>

<artifactId>vulnerability\_analyzer</artifactId>

<groupId>groupId</groupId>

<version>1.0-SNAPSHOT</version>

</parent>

<modelVersion>4.0.0</modelVersion>

<artifactId>functional-tests</artifactId>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-antrun-plugin</artifactId>

<version>1.6</version>

<executions>

<execution>

<phase>pre-integration-test</phase>

<configuration>

<target>

<exec executable="cmd.exe">

<arg value="/c"/>

<arg value="start.bat"/>

</exec>

</target>

</configuration>

<goals>

<goal>run</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

**pom.xml файл модуля gui:**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>GUI\_for\_Course</artifactId>

<version>1.0-SNAPSHOT</version>

<name>GUI\_for\_Course</name>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<junit.version>5.8.1</junit.version>

</properties>

<dependencies>

<dependency>

<groupId>org.openjfx</groupId>

<artifactId>javafx-controls</artifactId>

<version>17.0.1</version>

</dependency>

<dependency>

<groupId>org.openjfx</groupId>

<artifactId>javafx-fxml</artifactId>

<version>17.0.1</version>

</dependency>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-api</artifactId>

<version>${junit.version}</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>org.junit.jupiter</groupId>

<artifactId>junit-jupiter-engine</artifactId>

<version>${junit.version}</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>groupId</groupId>

<artifactId>analyzer-library</artifactId>

<version>1.0-SNAPSHOT</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<source>17</source>

<target>17</target>

</configuration>

</plugin>

<plugin>

<groupId>org.openjfx</groupId>

<artifactId>javafx-maven-plugin</artifactId>

<version>0.0.8</version>

<executions>

<execution>

<!-- Default configuration for running with: mvn clean javafx:run -->

<id>default-cli</id>

<configuration>

<mainClass>com.example.gui\_for\_course/com.example.gui\_for\_course.HelloApplication

</mainClass>

<launcher>app</launcher>

<jlinkZipName>app</jlinkZipName>

<jlinkImageName>app</jlinkImageName>

<noManPages>true</noManPages>

<stripDebug>true</stripDebug>

<noHeaderFiles>true</noHeaderFiles>

</configuration>

</execution>

</executions>

</plugin>

<plugin>

<artifactId>maven-assembly-plugin</artifactId>

<executions>

<execution>

<phase>package</phase>

<goals>

<goal>single</goal>

</goals>

</execution>

</executions>

<configuration>

<!-- ... -->

<archive>

<manifest>

<mainClass>com.example.gui\_for\_course.HelloApplication</mainClass>

</manifest>

</archive>

<descriptorRefs>

<descriptorRef>jar-with-dependencies</descriptorRef>

</descriptorRefs>

<outputDirectory>${project.build.directory}/execute</outputDirectory>

</configuration>

</plugin>

</plugins>

</build>

</project>

**Агрегатор pom.xml файл:**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>groupId</groupId>

<artifactId>vulnerability\_analyzer</artifactId>

<packaging>pom</packaging>

<version>1.0-SNAPSHOT</version>

<modules>

<module>analyzer-library</module>

<module>functional-tests</module>

<module>gui</module>

</modules>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<configuration>

<source>17</source>

<target>17</target>

</configuration>

</plugin>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-resources-plugin</artifactId>

<version>3.2.0</version>

<executions>

<execution>

<id>copy-resources</id>

<!-- here the phase you need -->

<phase>validate</phase>

<goals>

<goal>copy-resources</goal>

</goals>

<configuration>

<outputDirectory>src/</outputDirectory>

<resources>

<resource>

<directory>../analyzer-library/target/execute</directory>

</resource>

</resources>

</configuration>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

**start.bat:**

echo ==============START\_FUNCTIONAL\_TESTS==============

cd src

java -jar analyzer-library-1.0-SNAPSHOT-jar-with-dependencies.jar -s 1.TestCppFile\project -r 1.TestCppFile\report -v "randomCryptographicNumbers, bufferOverflow, formatStringError"

java -jar analyzer-library-1.0-SNAPSHOT-jar-with-dependencies.jar -s 2.TestCppProject\project -r 2.TestCppProject\report -v "randomCryptographicNumbers, bufferOverflow, formatStringError"

java -jar analyzer-library-1.0-SNAPSHOT-jar-with-dependencies.jar -s 3.TestCppProjectSnake\project -r 3.TestCppProjectSnake\report -v "randomCryptographicNumbers, bufferOverflow, formatStringError"

java -jar analyzer-library-1.0-SNAPSHOT-jar-with-dependencies.jar -s 4.TestReaderWriter\project -r 4.TestReaderWriter\report -v "readersWriters"

echo ===============END\_FUNCTIONAL\_TESTS===============