

МИНОБРНАУКИ РОССИИ

**Федеральное государственное бюджетное образовательное учреждение
высшего образования**

**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»**

Кафедра теоретических основ
компьютерной безопасности и
криптографии

Применение деревьев решений в поиске уязвимостей

АВТОРЕФЕРАТ

дипломной работы

студента 6 курса 631 группы
специальности 10.05.01 Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Бочкарева Матвея Сергеевича

Научный руководитель

доцент

А. Н. Гамова

19.01.2026 г.

Заведующий кафедрой

д. ф.-м. н., профессор

М. Б. Абросимов

19.01.2026 г.

Саратов 2026

ВВЕДЕНИЕ

С развитием цифровых технологий постоянно увеличивается количество программного обеспечения, устанавливаемого на рабочие станции, серверы и мобильные устройства. Вместе с этим растёт и число известных уязвимостей, которые фиксируются в международных и национальных базах данных, таких как Common Vulnerabilities and Exposures (CVE) и Банк данных уязвимостей ФСТЭК России (БДУ). Это создаёт необходимость систематического и автоматизированного контроля безопасности программных компонентов, поскольку ручная проверка их версий, конфигураций и зависимостей становится практически невозможной на практике.

Одним из подходов, применяемых в системах анализа, является использование деревьев решений – структур, описывающих процесс последовательных проверок условий. В отличие от методов машинного обучения, где деревья решений строятся автоматически и могут использовать статистические меры информативности признаков, в практических системах безопасности нередко применяется детерминированная модель дерева решений, реализованная в виде набора явно определённых правил. Подобные структуры хорошо подходят для задач сопоставления версий программ с известными уязвимостями, позволяя прозрачно формализовать логику анализа и обеспечить интерпретируемость результата.

Целью данной работы является разработка программного сканера уязвимостей, основанного на модели дерева решений и предназначенного для анализа программного обеспечения на различных операционных системах.

Для достижения поставленной цели необходимо решить следующие задачи:

- 1) провести обзор существующих подходов и инструментов для автоматического поиска уязвимостей;
- 2) исследовать теоретические основы применения деревьев решений в задачах анализа данных и экспертных систем;

3) разработать архитектуру программного средства, определить ключевые модули и алгоритмы;

4) реализовать программный сканер уязвимостей, включающий модуль идентификации ПО, парсер базы уязвимостей, механизм сопоставления и генератор отчётов;

5) провести тестирование программы, оценить корректность её работы и сформулировать выводы о проделанной работе.

Дипломная работа состоит из введения, 3 разделов, заключения, списка использованных источников и 11 приложений. Общий объем работы – 104 страниц, из них 53 страницы – основное содержание, включая 36 рисунков и 1 таблицу, список использованных источников из 20 наименований.

КРАТКОЕ СОДЕРЖАНИЕ

В первом разделе «Теоретические основы анализа уязвимостей» приведены основные концепции, необходимые для разработки программного сканера уязвимостей. Рассмотрены вопросы о классификации уязвимостей, их источниках, методах обнаружения вторжений и деревьях решений.

В подразделе 1.1 «Классификация уязвимостей и их источники» вводится понятие уязвимости; описываются стандартизированные источники, на основе которых строится анализ уязвимостей (CWE, CVE и БДУ ФСТЭК); выделяются пять основных категорий уязвимостей.

В подразделе 1.2 «Методологические подходы к обнаружению уязвимостей» описываются ключевые подходы выявления уязвимостей: как классические методы анализа кода, так и современные методы с применением машинного обучения и автоматизации.

В подразделе 1.3. «Построение деревьев решений» вводятся необходимые понятия и термины, описывающие структуру дерева решений и необходимые в дальнейшем при реализации программного сканера (корень, лист, разделение, сокращение и пр.). Также рассматривается пример построения дерева решений по таблице с уязвимостями программного обеспечения согласно описанному алгоритму. Пример построенного по таблице дерева показан на рисунке 1.

В подразделе 1.4 «Существующие инструменты поиска уязвимостей и их особенности» рассматриваются современные инструменты поиска уязвимостей, реализующие разные подходы из раздела 1.2 и ориентированные на автоматизацию сканирования, анализ кода и интеграцию с каталогами уязвимостей.

Раздел 2 «Структура баз данных уязвимостей» подробно описывает ключевые системы каталогизации уязвимостей, используемые в мировой и российской практике. В нем рассматриваются базы данных CVE и NVD, а также отечественный аналог – Банк данных угроз ФСТЭК соответственно по подразделам:

- 2.1 Common Vulnerabilities and Exposures (CVE),
- 2.2 National Vulnerability Database (NVD),
- 2.3 Банк данных угроз безопасности информации (БДУ ФСТЭК).

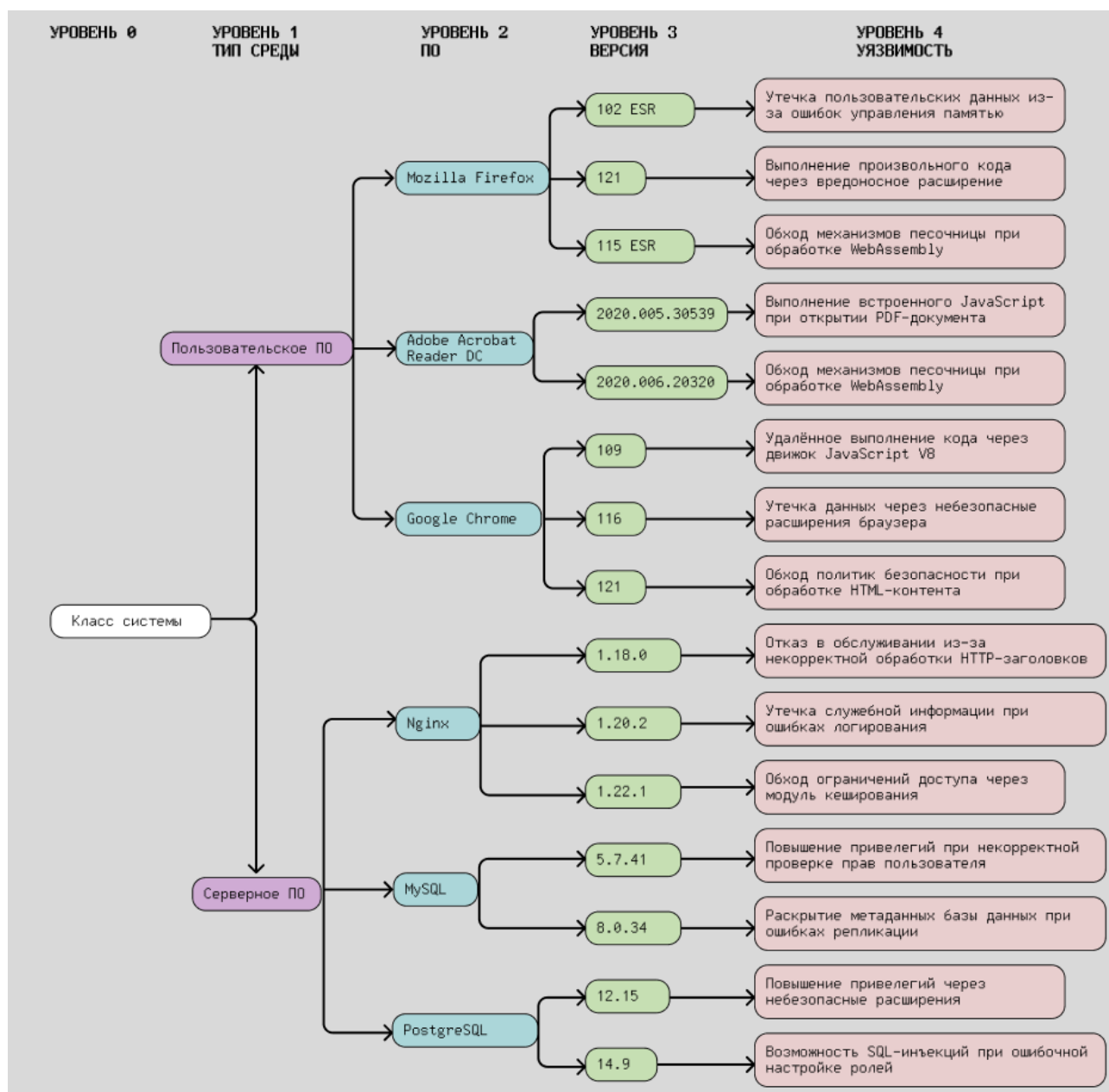


Рисунок 1 – Дерево решений для анализа программного обеспечения

В разделе 3 «Реализация программного сканера» описывается структура проекта (модульный принцип) и обуславливается выбор языка программирования (Python) и объектно-ориентированной архитектуры. Также отдельно отмечают обязательные к установке зависимости, чтобы проект был работоспособен на большинстве машин (с системами Windows и Linux).

Демонстрация и тестирование готового решения на разных системах являются логическим завершением раздела.

Подраздел 3.1 «Обработка исполняемых файлов и анализ метаданных» описывает особенности реализации в рамках логики обнаружения программного обеспечения – поиск и анализ исполняемых PE(Portable Executable)-файлов и их метаданных. В Linux для получения данных о ПО описаны функции для получения установленных пакетов определенным пакетным менеджером, установленным в дистрибутиве.

В подразделе 3.2 «Парсинг баз уязвимостей и построение дерева решений» описывается модуль `bdu_parser.py`, в котором реализуется парсинг базы уязвимостей ФСТЭК, валидация и нормировка полученных из Excel-файла значений. Построение структуры данных для реализации дерева решений происходит в файле `data_structures.py`, который является ключевой логикой проекта. Схема структуры построенного на базе файла дерева решений изображена на рисунке 10.

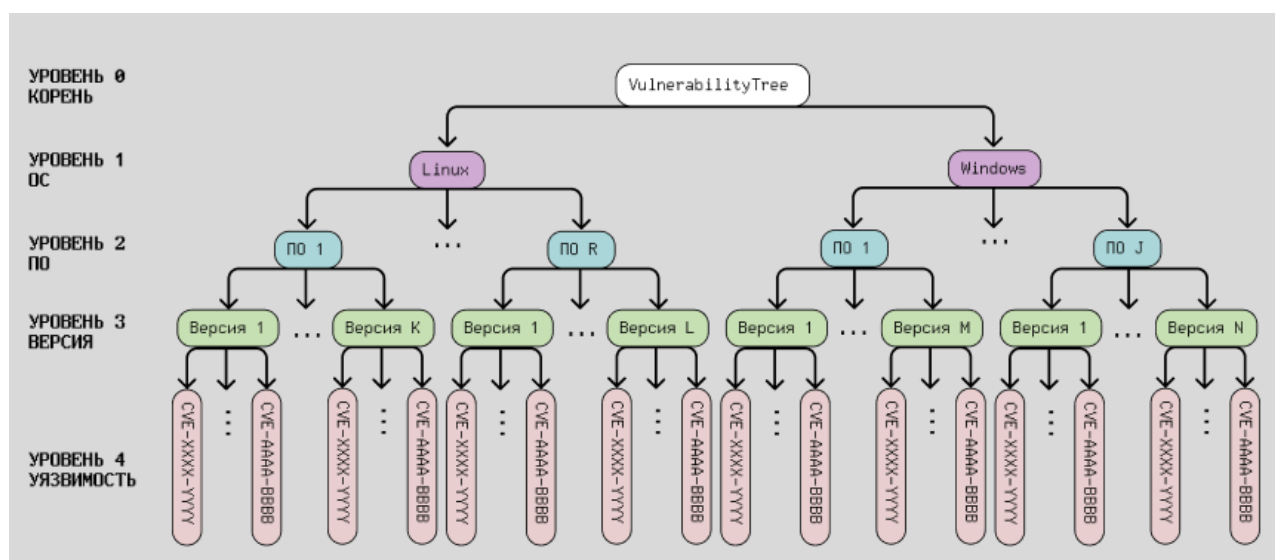


Рисунок 10 – Схема структуры построенного на базе файла дерева решений

В подразделе 3.3 «Поиск уязвимостей в программном обеспечении» описывается процесс соотнесения полученной о файле информации и дерева решений с данными из БДУ ФСТЭК.

В подразделе 3.4 «Формирование отчётов и интерфейс пользователя» описывается реализация функций, необходимых для управления системой

отчетов: формирование JSON-файла, создание визуализированных отчётов в формате HTML с применением CSS для стилизации и JavaScript для реализации динамического поведения, графический интерфейс с системой вкладок в три раздела, режимы сканирования, экспорт отчетов и т. д.

Подраздел 3.5 «Демонстрация работы программного сканера и оценка результатов» является логическим общим итогом применения описанной теории в практической реализации сканера уязвимостей. На рисунке 11 показан интерфейс программы в системе с Windows.

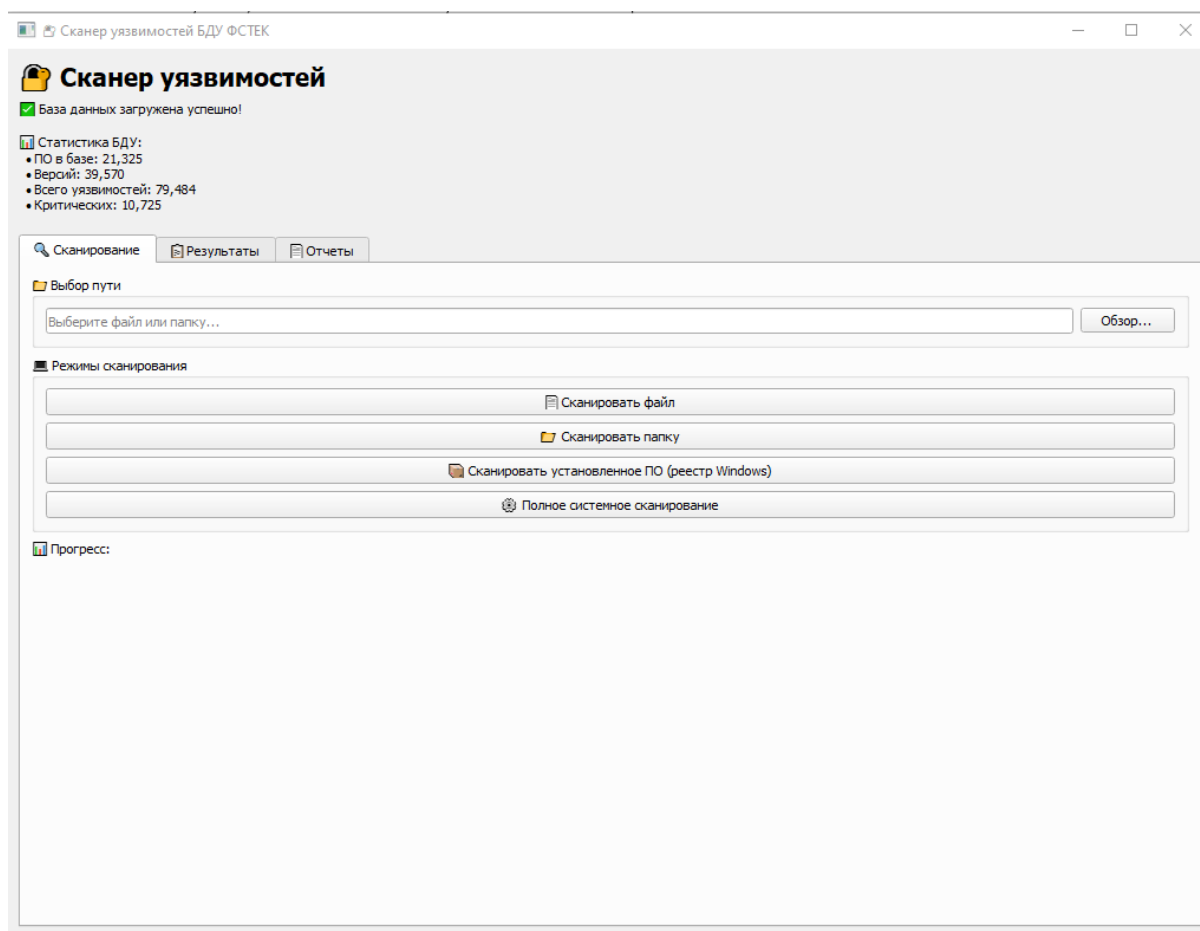


Рисунок 11 – Интерфейс программы

На рисунке 15 показан результат сканирования папки C:\Program Files\Docke.

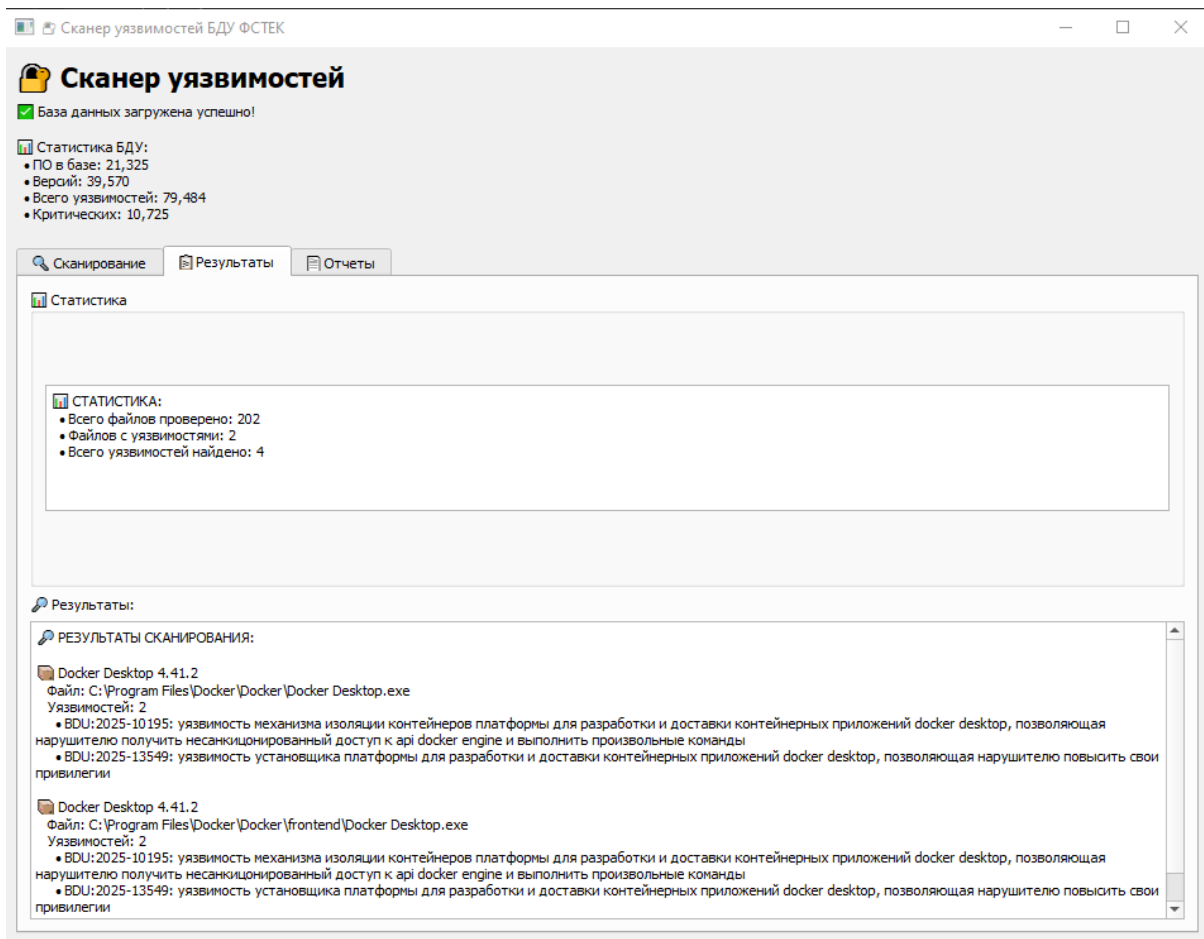


Рисунок 15 – Результат сканирования папки C:\Program Files\Docker

На рисунке 25 показан список уязвимостей после полного сканирования в HTML-отчете.

C:\Program Files\NVIDIA Corporation\NVIDIA App\CEF\NVIDIA App.exe	NVIDIA App	128.4.13.14	<p>BDU:2025-12851 (CVE-2025-23253) LOW</p> <p>уязвимость компонента драйверов nvidia nvcontainer единого центра управления nvidia app, связанная с использованием жестко закодированных констант, позволяющая нарушителю вызвать отказ в обслуживании или выполнить произвольный код</p> <p>Уязвимость компонента драйверов NVIDIA NvContainer утилиты NVIDIA App связана с использованием жестко закодированных констант. Эксплуатация уязвимости может позволить нарушителю вызвать отказ в обслуж...</p> <p>Рекомендация по устранению:</p> <p>Использование рекомендаций: https://nvidia.custhelp.com/app/answers/detail/a_id/5644</p>
C:\Program Files\WireGuard\wireguard.exe	wireguard	0.5.3	<p>BDU:2023-00658 (CVE-2021-46873) CRITICAL</p> <p>уязвимость реализации протокола синхронизации времени ntp vpn-сервиса wireguard операционных систем windows, позволяющая нарушителю вызвать отказ в обслуживании</p> <p>Уязвимость реализации протокола синхронизации времени NTP VPN-сервиса WireGuard операционных систем Windows связана с недостаточной проверкой входных данных. Эксплуатация уязвимости может позволить на...</p> <p>Рекомендация по устранению:</p> <p>Использование рекомендаций: https://lists.zx2c4.com/pipermail/wireguard/2021-August/006916.html</p> <p>Компенсирующие меры: 1) Использование протокола синхронизации времени аналогичного NTP (NTPSec, NTS, Chrony и др.) 2) Использование значения счетчика равное 1 и увеличение его в каждой последующей фазе на единицу</p>
C:\Users\mat_m\AppData\Local\Programs\Azure Data Studio\azuredatstudio.exe	Azure Data Studio	1.48.0	<p>BDU:2024-02110 (CVE-2024-26203) HIGH</p> <p>уязвимость программного средства разработки и управления данными с подключением к облачным и локальным базам данных Azure Data Studio, связанная с недостатками разграничения доступа, позволяющая нарушителю повысить свои привилегии</p> <p>Уязвимость программного средства разработки и управления данными с подключением к облачным и локальным базам данных Azure Data Studio, связанная с недостатками разграничения доступа, позволяющая нару...</p> <p>Рекомендация по устранению:</p> <p>Использование рекомендаций: https://learn.microsoft.com/en-us/azure-data-studio/release-notes-azure-data-studio</p>

Рисунок 25 – Список уязвимостей после полного сканирования в HTML-отчете

На рисунке 31 показан пример работы сканера на машине с системой Linux (Kali).

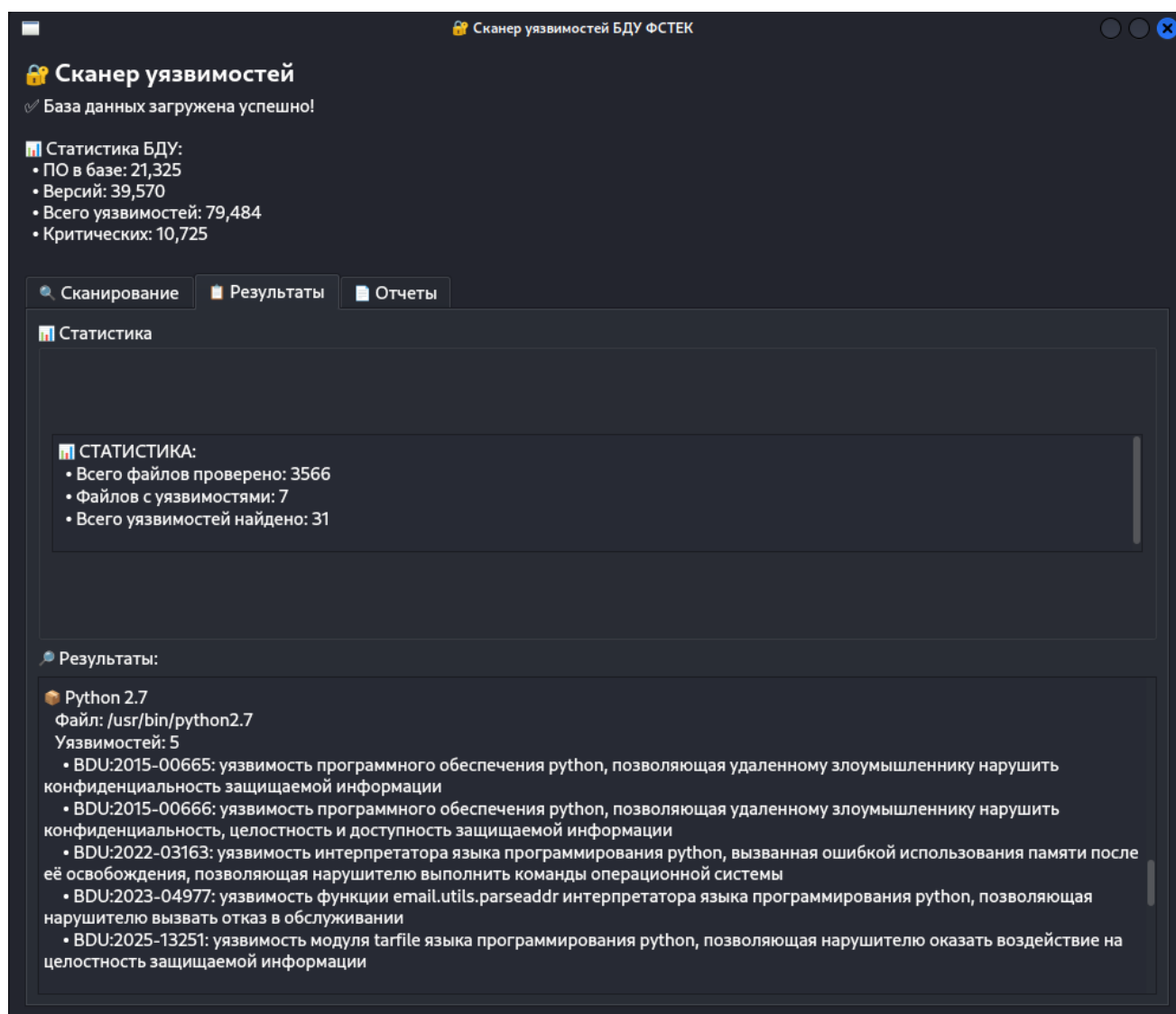


Рисунок 31 – Результат работы сканирования папки /usr/bin

Пример уязвимости ПО Python с сайта БДУ ФСТЭК представлен на рисунке 32.

Главная / Список уязвимостей / BDU:2015-00665					
BDU:2015-00665					
Вид ▾					
Описание уязвимости	Целочисленное переполнение в Python позволяет получить доступ к содержимому памяти, если в тип buffer передаются большие значения размера и смещения.				
Уязвимое ПО	Вендор	Наименование ПО	Версия ПО	Тип ПО	Архитектура (Платформа)
	Python Software Foundation	Python	от 2.7.0 до 2.7.8	Прикладное ПО информационных систем	Не указана
Операционные системы и аппаратные платформы	Данные уточняются				
Тип ошибки	Данные уточняются				
Класс уязвимости	Данные уточняются				
Дата выявления	01.07.2014				
Базовый вектор уязвимости	CVSS 2.0: AV:N/AC:L/Au:N/C:P/I:N/A:N				
Уровень опасности уязвимости	Средний уровень опасности (базовая оценка CVSS 2.0 составляет 5)				

Рисунок 32 – Пример уязвимости ПО Python с сайта БДУ ФСТЭК

HTML-отчет со списком просканированных файлов при полном сканировании в системе Linux представлен на рисунке 35.

Отчет об уязвимостях					
Дата сканирования: 2025-12-18T10:29:13.955578					
Просканировано файлов	Файлов с уязвимостями	Всего уязвимостей	Критических	Высокой опасности	
3925	8	32	3	6	
Уязвимости Просканированные файлы					
Просканированные файлы (3925)					
Файл/Программа	ПО	Версия	Уязвимостей	Статус	
/usr/bin/7zip	7zip	24.09+dfsg-5	0	Безопасно	
/usr/bin/accountsservice	accountsservice	23.13.9-7	0	Безопасно	
/usr/bin/acl	acl	2.3.2-2+b1	0	Безопасно	
/usr/sbin/adduser	adduser	3.137	0	Безопасно	
/usr/bin/adwaita-icon-theme	adwaita-icon-theme	47.0-2	0	Безопасно	
/usr/bin/aircrack-ng	aircrack-ng	1:1.7+git20230807.4bf83f1a-2	0	Безопасно	
/usr/bin/alsa-topology-conf	alsa-topology-conf	1.2.5.1-3	0	Безопасно	
/usr/bin/alsa-ucm-conf	alsa-ucm-conf	1.2.13-2	0	Безопасно	

Рисунок 35 – Список просканированного ПО в HTML-отчете

ЗАКЛЮЧЕНИЕ

В ходе работы достигнута цель исследования: разработан программный сканер уязвимостей, основанный на модели дерева решений и предназначенный для анализа программного обеспечения на различных операционных системах (Linux и Windows).

Основные результаты и выводы работы:

1) проведён анализ существующих инструментов и методов автоматического поиска уязвимостей, выявлены их ограничения, обоснована целесообразность применения детерминированных деревьев решений для анализа уязвимостей;

2) разработана архитектура программного средства, включающая модули идентификации ПО, парсинга баз данных уязвимостей, сопоставления версий и генерации отчётов;

3) реализован рабочий прототип сканера. Тестирование подтвердило его корректность и полноту выявления известных уязвимостей.

Все задачи, поставленные в рамках преддипломной практики решены в полном объёме.

Разработанный сканер может быть использован для автоматизации локального мониторинга безопасности программного обеспечения.