Государственное учреждение образования

“БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ”

Кафедра: Интеллектуальных информационных технологий

Дисциплина: Проектирование защищенных интеллектуальных информационных систем

Отчет по лабораторной работе №6

**Анализ безопасности кода.**

**Методика оценки безопасности кода**

Выполнил:

студент гр.121702

Витковская С. И.

Проверил:

Захаров В. В.

Минск 2024

**Задание:**

1. На основе полученных результатов предыдущей части лабораторной работы создать методику оценки безопасности кода.

2. По созданной методике сравнить предоставленный алгоритм со своим из предыдущей части лабораторной работы.

**Ход работы:**

1. **Методика**

Анализ безопасности исходного кода – это анализ программного обеспечения на предмет выявления уязвимостей информационной безопасности, допущенных при его разработке.

Существует три группы методов анализа исходного кода:

1. Динамические методы - методы анализа безопасности программного обеспечения, требующие выполнения программ на реальном или виртуальном процессоре, с доступом к исходному коду и среде его функционирования.
2. Статические методы – методы анализа безопасности программного обеспечения с доступом к исходному коду (или производным) приложения серверных и клиентских частей, но не требующие выполнения программ.
3. Гибридные методы – методы, совмещающие два предыдущих подхода.

Определим алгоритм проверки безопасности кода:

1. Анализ защищенности методами «черного» и «серого ящика» – динамический анализ безопасности программного обеспечения без доступа к исходному коду:

– метод «черного ящика» направлен на поиск уязвимостей, использование которых позволяет злоумышленнику не имеющему никаких привилегий реализовать следующие виды угроз: получение несанкционированного доступа к информации, полного или частичного контроля над приложением;

–метод «серого ящика» аналогичен предыдущему, с тем лишь исключением, что под злоумышленником подразумевается пользователь, обладающий определенным набором привилегий в информационной системе.

2. Анализ защищенности методом «белого ящика» – динамический и статический анализ безопасности исходного кода. У пользователя (тестировщика) есть доступ к исходному коду и документации проекта, что позволяет ему исследовать и проверять внутреннюю работу приложения.

3. Разработка рекомендаций и итогового отчета;

4. Проверка корректности устранения выявленных уязвимостей.

При тестировании программы необходимо проанализировать поведение приложения при вводе невалидных данных и в аварийных ситуациях. В первом случае в коде должны присутствовать проверки на корректность пользовательского ввода, во втором – проконтролировать, чтобы при возникновении аварийной ситуации программа не прекращала работу, а предупреждала пользователя об ошибке.

1. **Анализ**

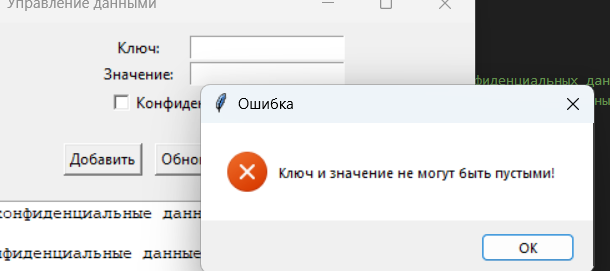
В рамках лабораторной работы не проводилось полноценного развертывания приложения, поэтому проводится ручное тестирование.

* 1. **Анализ безопасности методом «черного ящика»**

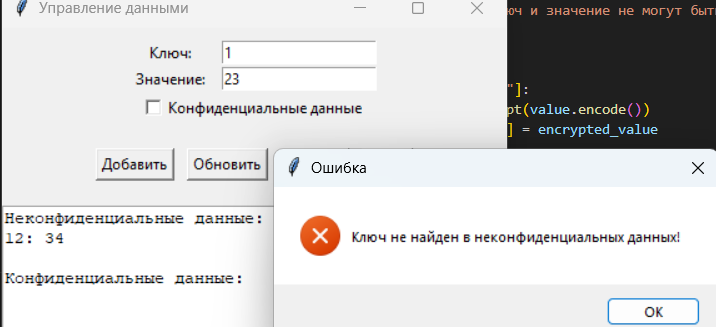
1. Существуют проверки корректности пользовательского ввода.

Невалидный ввод при:

Попытке добавления



Попытке обновления



2. При добавлении конфиденциальных данных, они не отображаются в пользовательском интерфейсе.

|  |  |
| --- | --- |

1. Присутствует ошибка при добавлении данных с уже существующим ключом – они просто перезаписываются.

**b. Анализ безопасности методом «белого ящика»**

1. Код организован в функции, что упрощает проверку и сопровождение.
2. Конфиденциальные данные могут быть защищены через функции хэширования.
3. Нет проверки типов данных, что может привести к ошибкам или некорректному поведению.
4. Данные остаются в памяти после использования и могут быть извлечены из дампа памяти, но конфиденциальные данные хранятся в зашифрованном виде.
5. Отсутствие журналирования попыток ввода некорректных данных.
6. Использование tracemalloc помогает отслеживать статистику памяти и находить потенциальные утечки.
7. Нет обработки исключений для таких случаев, как ошибка записи в файл (например, недостаточно прав), ошибка при доступе к памяти, при аварийной остановке не выполняется сохранение состояния.

**с. Рекомендации**

1. Очистка удаленных данных с помощью метода del для удаления элементов словаря.
2. Исключение перезаписи данных, создание предупреждающего сообщения.
3. Обработка большего количества внештатных ситуаций, использование блоков try-exept для всех участков кода, взаимодействующих с пользовательским вводом
4. Использование Argon2 для хэширования.
5. Разделение групп пользователей, создание привилегий разного уровня, добавление авторизации.
6. Лог-файлы для записи возникающих ошибок для последующего анализа.

**Вывод:**

В ходе лабораторной работы был определен алгоритм проверки безопасности программы. Было протестировано десктопное приложение для управления данными в оперативной памяти из лабораторной работы 3.1. Динамическими и статическими методами анализа. Были выделены уязвимости системы и на их основе даны рекомендации для усиления безопасности кода.