Отчет о комплексном проведении исследования приложения «Nocommerce» и его исходных кодов.

[1. Вводная часть 3](#_Toc1)

[1.1 Описание задачи 3](#_Toc2)

[1.2 Исходные данные 3](#_Toc3)

[1.3 Методы анализа, нормативные документы 3](#_Toc4)

[2. Исследование и анализ кода на выявление уязвимостей 3](#_Toc5)

[2.1 Используемое ПО в процессе анализа 3](#_Toc6)

[2.2 Результаты SATS (красиво оформить) 3](#_Toc7)

[2.3 Отчет по найденным уязвимостям 4](#_Toc8)

[2.4 Отчет по соблюдению рекомендаций и нормативных документов при разработке 4](#_Toc9)

[3. Заключение и рекомендации 5](#_Toc10)

#### Вводная часть

##### **1.1** Описание задачи

Необходимо провести комплексный анализ процесса разработки и протестировать на наличие других веб-уязвимостей.

Провести комплексное исследования приложения «Nocommerce» и его исходных кодов.

##### 1.2 Исходные данные

Компания предоставляет комплексное веб-приложение в сфере электронной коммерции «Necommerce». Компания недавно столкнулась с массовой утечкой данных о своих пользователях и их покупках. «Брешь», через которую произошла утечка данных устранили, по информации фирмы подрядчика, которая ведет саму разработку приложения.

Своих специалистов по информационной безопасности у компании нет. Деталей по найденным и исправленным уязвимостям компании подрядчик не предоставил.

##### 1.3 Методы анализа, нормативные документы

﻿При анализе кода был применен метод SAST (Static Application Security Testing) - статический анализатор исходного кода.

Анализ сетевого трафика с помощью инструмента Kali Linux – Wireshark.

Использование нормативных документов: ГОСТ Р 58412 - 2019, PCI DSS.

#### Исследование и анализ кода на выявление уязвимостей

##### 2.1 Используемое ПО в процессе анализа

Развернута виртуальная сеть, состоящая из двух виртуальных машин на базе Oracle VM VirtualBox. На одной ВМ были развернуты предоставленные docker образы и запущены Frontend и Backend сервера «Nocommerce». Вторая ВМ с операционной системой Kali Linux использовалась для проведения тестирования и поиска уязвимостей при работе приложения.

Анализе кода SAST проводился на платформах gitlab.com и snyk.io. Анализировались предоставленные репозитории:

1. netology-code/necommerce-backend (отвечающий за серверную часть).
2. netology-code/necommerce-frontend (отвечающий за веб интерфейс).

Инструмент для анализа трафика Wireshark (версия 4), применялся для исследования обмена трафиком между клиентом (frontend) и сервером (backend).

##### **2.2 Результаты SATS (красиво оформить)**

**1) При анализе кода репозитория netology-code/necommerce-backend обнаружены следующие уязвимости:**

Имеющие отношение к docker образу:

13 критических уязвимостей, 25 уязвимостей высокого уровня и 21 уязвимость среднего уровня.

Имеющие отношение к коду:

1 уязвимость высокого уровня.

**2) При анализе кода репозитория netology-code/necommerce-frontend выявлены следующие уязвимости:**

Имеющие отношение к docker образу:

1 критическая уязвимость, 2 уязвимости высокого уровня, 1 уязвимость среднего уровня

Уязвимости связанные с package.json:

23 уязвимости высокого уровня, 33 уязвимости

##### **2.3 Отчет по найденным уязвимостям**

В ходе проведения статического анализа кода была выявлена дна уязвимость высокого уровня :

src/main/kotlin/ru/netology/necommerce/config/AppWebSecurityConfigurerAdapter.kt

**Cross-Site Request Forgery (CSRF) CWE-352**

С помощью Cross-Site Request Forgery и произошла утечка данных пользователей, так как не были соблюдены меры следующие меры защиты от атак CSRF:

* На сервере реализовать механизм «CSRF токенов». Это такой механизм, когда для каждой сессии пользователя генерируется новый токен и сервер проверяет его валидность при любых запросах с клиента.
* На сервере проверять заголовки Origin и Referer, в которых содержится адрес источника запроса. Но эти заголовки могут отсутствовать.
* Также необходимо всегда требовать от пользователя подтверждать критические действия вводом пароля или вторым фактором аутентификации.

Выявленные уязвимости связанные с образами docker можно устранить на начальном этапе разработки. Для этого необходимо использовать стандартный дистрибутив операционной системы, установить его на «голую» машину, поставить необходимые пакеты и исправления, создать "идеальный" docker-образ. Данный образ необходимо переодически проверять на уязвимости и обновлять установленные приложения.

##### 2.4 Отчет по соблюдению рекомендаций и нормативных документов при разработке

При разработке программного продукта не были учтены следующие требования по информационной безопасности стандарта PCI DSS, ГОСТ Р 58412 - 2019:

Стандарт безопасности данных PCI DSS «Требования и процедуры оценки безопасности»:

* Клиентские сетевые потоки данных должны передаваться в зашифрованном виде (пункт 4.1.с PCI DSS). Нарушение - канал передачи данных между клиентом и сервером не зашифрован.
* Базы данных должны размещаться в подсети, отдельной от подсети приложений (пункт 1.3.6 PCI DSS). Нарушение - БД находится на одном сервере с программой.
* Проверка на OWASP (рекомендация PCI DSS). Нарушения - проект не проходил проверку на основные угрозы OWASP.
* Требования к стойкости пароля (пункт 8.2.3 PCI DSS). Нарушения - не обеспечено соответствие паролей требованиям безопасности.

Требования ГОСТ Р 58412—2019 "Национальный стандарт Российской Федерации. Защита информации. Разработка безопасного программного обеспечения. Угрозы безопасности информации при разработке программного обеспечения":

* Угроза внедрения уязвимостей программы в исходный код программы в ходе его разработки (пункт 5.3.1 ГОСТ Р 58412—2019). Нарушения в создании docker образов, в коде обнаружены уязвимости
* Угрозы безопасности информации при выполнении квалификационного тестирования программного обеспечения (пункт 5.4 ГОСТ Р 58412—2019). Угроза выявления уязвимостей вследствие раскрытия информации о тестировании программного обеспечения (пункт 5.4.2 ГОСТ Р 58412—2019). Нарушения в игнорировании выявленных уязвимостей в ходе статического тестирования (SAST)
* Угрозы безопасности информации при выполнении инсталляции программы и поддержки приемки программного обеспечения (пункт 5.5 ГОСТ Р 58412—2019). Угроза внедрения уязвимостей в обновления программного обеспечения (пункт 5.5.3 ГОСТ Р 58412—2019). Нарушения в создании docker образов - наличие не обновлённого программного обеспечения с выявленными уязвимостями
* Угрозы безопасности информации при решении проблем в программном обеспечении (пункт 5.6 ГОСТ Р 58412—2019) в процессе эксплуатации. Угроза не исправления обнаруженных уязвимостей программы (пункт 5.6.1 ГОСТ Р 58412—2019). Нарушения в игнорировании выявленных уязвимостей и/или отсутствии компенсационных мер по недопустимости реализации уязвимости. Выявленные в ходе тестов (SAST) уязвимости не отслеживаются и не исправляются.

#### **Заключение и** рекомендации

3.1 Рекомендации по улучшению процесса разработки

3.2 ссылки на стандарты