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

[1. Вводная часть 3](#_Toc1)

[1.1 Постановка 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. 0.62 смЗаключен и рекомендации 6](#_Toc10)

# Вводная часть

## 1.1 Постановка 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:

а) 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.

б) На сервере проверять заголовки Origin и Referer, в которых содержится адрес источника запроса. Но эти заголовки могут отсутствовать.

в) 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:

1. Клиентские сетевые потоки данных должны передаваться в зашифрованном виде  
(пункт 4.1.с PCI DSS).

Критичные данные должны передаваться по шифрованному каналу. Сетевые критичные данные должны передаваться по шифрованному каналу. Формально, достаточно использование протокола безопасности TLS, но предпочтительнее обеспечить шифрование передаваемой информации на уровне ПО

«Necommerce» - канал передачи данных между клиентом и сервером не зашифрован.

2. Базы данных должны размещаться в подсети, отдельной от подсети приложений  
(пункт 1.3.6 PCI DSS).

Для уменьшения рисков утечки критичных данных, БД должны размещаться отдельно от подсети, в которой размещены приложения. Учитывая повсеместную виртуализацию, рекомендуется также рассмотреть возможность дополнительного разделения БД и приложений.

«Necommerce» - БД находится на одном сервере с программой.

3. Проверка на OWASP (рекомендация PCI DSS)

«Necommerce» - проект не проходил проверку на основные угрозы OWASP.

4. Требования к стойкости пароля (пункт 8.2.3 PCI DSS).

Должны выбираться пароли, которые не поддаются атаке перебором. Хранение и передача пароля должна быть обеспечена таким образом, чтобы минимизировать вероятность его компрометации (парольные хранилища, раздельное хранение и пр.).

«Necommerce» - не обеспечено соответствие паролей требованиям безопасности.

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

5.1 Угроза внедрения уязвимостей программы в исходный код программы в ходе его разработки (пункт 5.3.1 ГОСТ Р 58412—2019)

Данная угроза заключается в преднамеренном или непреднамеренном внедрении в исходный код программы ошибок (недостатков), которые при условии их не обнаружения или не исправления могут стать причиной появления уязвимостей программы

Угроза обусловлена недостатками в реализованных разработчиком ПО мерах контроля доступа и контроля целостности, применяемых к объектам среды разработки ПО, и мерах по разработке безопасного ПО, в частности: некачественным или неполным проведением статического анализа или экспертизы исходного кода программы, не учётом при создании программы проекта архитектуры программы и/или порядка оформления исходного кода программы, недостатками в мерах, связанных с управлением конфигурацией ПО, обучением работников разработчика ПО в области разработки безопасного ПО и проведением систематического поиска уязвимостей программы.

Нарушения в создании docker образов, в коде обнаружены уязвимости

5.2 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)

Данная угроза заключается в преднамеренном или непреднамеренном раскрытии информации, связанной с тестированием ПО (планы тестирования, описание выполняемых тестов и инструментальных средств, используемых для тестирования программы, фактические результаты тестирования, перечень выявленных при тестировании ПО уязвимостей программы и ошибок ПО). Нарушение конфиденциальности данной информации может способствовать выявлению недостатков ПО и уязвимостей программы, которые в дальнейшем могут быть использованы с целью выполнения компьютерных атак на информационные системы пользователей, применяющих ПО.

Нарушения в игнорировании выявленных уязвимостей в ходе статического тестирования (SAST)

5.3 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)

Данная угроза заключается в получении пользователем обновлений ПО, содержащих внедренные уязвимости программы или уязвимости программы, появившиеся в результате ошибок или неквалифицированных действий работников разработчика ПО при определении и реализации процедуры обновления ПО. Уязвимости программы могут быть внедрены в обновления компонентов ПО собственной

разработки, а также в обновления компонентов ПО, которые заимствуют у сторонних разработчиков ПО. Внедрение уязвимостей программы может быть осуществлено путем модификации (включая подмену) обновлений ПО при их передаче пользователю из среды разработки ПО, при их передаче пользователю из среды разработки стороннего разработчика ПО и при их передаче разработчику из среды разработки стороннего разработчика ПО. Уязвимости программы могут быть внедрены в обновления заимствованных у сторонних разработчиков ПО компонентов путем их модификации в среде разработки стороннего разработчика ПО. Внедренные уязвимости программы в дальнейшем могут быть использованы нарушителем с целью выполнения компьютерных атак на информационную систему пользователя.

Нарушения в создании docker образов - наличие не обновлённого программного обеспечения с выявленными уязвимостями

5.4 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)

Данная угроза заключается в том, что обнаруженные ошибки ПО, которые могут стать причиной появления уязвимостей, не исправляют или исправляют несвоевременно вследствие неквалифицированных действий работников разработчика при определении и реализации процедур отслеживания и исправления обнаруженных ошибок ПО.

Угроза обусловлена недостатками в реализованных разработчиком ПО механизмах контроля доступа к объектам среды разработки ПО и контроля целостности объектов среды разработки ПО, а также мерах по разработке безопасного ПО, в частности: отсутствием мер, связанных с решением проблем в ПО в процессе эксплуатации, недостатками в процедурах, связанных с отслеживанием и исправлением обнаруженных ошибок ПО и обучением работников разработчика ПО в области разработки безопасного ПО

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.

# Заключен и рекомендации

## 3.1 Рекомендации по улучшению процесса разработки

* Организовать периодическое обучение работников и периодический анализ программы обучения работников;
* На этапе планирования учесть требования и оценить риски связанные с безопасностью
* На стадии разработки регулярно проводить статический анализ исходного кода программы;
* Внедрить на стадии разработки регистрацию событий, связанных с фактами изменения элементов конфигурации
* На этапе разработки обязательно отрабатывать результаты анализа SATS:

а) Убрать ложные срабатывания, устранить уязвимости, проверить исправления

б) Проверить версии зависимостей на наличие уязвимостей.

* На этапе тестирования проверять приложения специальными инструментами, призванными в автоматизированном режиме выявлять типичные уязвимости
* На этапе тестирования необходимо устранить уязвимости, имеющие критический и высокий уровень опасности
* Внедрить практику экспертизы исходного кода программы сторонними экспертами в области ИБ;
* Ввести в организации определение, документирование и соблюдение политики информационной безопасности;
* Ввести в практику регистрацию выявленных уязвимостей, отслеживание их устранения.
* Использование системы управления конфигурацией ПО;
* Защита от несанкционированного доступа к элементам конфигурации;
* Перед выпуском релиза рекомендуется устранить уязвимости, имеющие уровень опасности средний.
* К следующему релизу рекомендуется устранить уязвимости с уровнем опасности низкий и инфо.

## 3.2 ссылки на стандарты