ORES

Научно-практический электронный журнал
Оригинальные исследования (ОРИС), *ISSN 2222-0402*
www.ores.su
original-research.ru

УДК 004.492.3

ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ ВЕДЕНИЯ БАЗЫ ЗНАНИЙ УЯЗВИМОСТЕЙ ПО

Трифонов Иван Сергеевич
Студент НИЯУ МИФИ, г. Москва;
Email: kortess1999@mail.ru

Красникова Светлана Анатольевна
Старший преподаватель НИЯУ МИФИ, г. Москва;
Email: sakrasnikova@mephi.ru

Салиева Аделина Рустамовна
Студент НИЯУ МИФИ, г. Москва;
Email: adelina_salieva@mail.ru

Аннотация

В данной статье рассматривается проблема обеспечения подхода к безопасному программированию. На основе изучения существующих рекомендаций, нормативных актов, международных стандартов создания, защищённого программных продуктов, автором было предложено улучшение для статических анализаторов кода, используемых в рамках жизненных циклов ПО. Значительное внимание уделяется архитектуре информационной системе, которая устраняет выявленные недостатки. Далее, в результате анализа данной архитектуры, автором была разработана модель базы знаний уязвимостей ПО, что будет использована при разработке информационной системы по данной архитектуре.

Ключевые слова: статические анализаторы кода, безопасное программирование, классификация уязвимостей.

DESIGNING AN INFORMATION SYSTEM FOR MAINTAINING A KNOWLEDGE BASE OF SOFTWARE VULNERABILITIES

Ivan S. Trifonov
Student NRU MEPhI, Moscow.
Email: kortess1999@mail.ru

Svetlana A. Krasnikova
Senior lecturer NRU MEPHI, Moscow;
Email: sakrasnikova@mephi.ru

Adelina R. Salieva
Student NRU MEPhI, Moscow.
Email: adelina_salieva@mail.ru

ABSTRACT

This article discusses the problem of providing an approach to secure programming. Based on the study of existing recommendations, regulations, international standards for the creation of protected software products, the author proposed an improvement for static code analyzers used within software life cycles. Considerable attention is paid to the architecture of the information system, which eliminates the identified shortcomings. Further, as a result of the analysis of this architecture, the author developed a knowledge base model of software vulnerabilities that will be used in the development of an information system for this architecture.

Keywords: static code analyzers, secure programming, vulnerability classification.

Введение

Анализ программных комплексов, систем и приложений на предмет наличия уязвимостей информационной безопасности становится всё более актуальной задачей ввиду значительного роста как числа систем, так и их активных пользователей, возрастающего риска и потенциального ущерба из-за использования программного (ПО) и аппаратного обеспечения с уязвимостями, которые не были распознаны до внедрения системы в эксплуатацию.

Классификация и сканирование уязвимостей на ранних этапах разработки продукта стали неотъемлемой частью мира информационной безопасности. Современные решения позволяют эффективно находить уязвимости в коде, однако не способны автоматически исправлять код. Для решения этой задачи разрабатывается система, частью которой является база знаний уязвимостей ПО.

1) Цель исследования

В рамках данной работы проведён анализ подходов к организации разработки ПО на принципах безопасного программирования с учётом отечественной и международной нормативно-методической базы и выполнено проектирование информационной системы, поддерживающей ведение базы знаний уязвимостей ПО, включая рекомендации по решению выявленных проблем.

2) Материалы и методы исследования

Нормативная база организации безопасного программирования

Отечественные стандарты

В Российской Федерации организация разработки безопасного ПО основывается на нормативной базе, которая составляет основой ориентир при работе с национальными проектами, и методических указаниях и рекомендациях различного уровня, которые устанавливают правила контроля разработки ПО, начиная с самого разработчика безопасного ПО и заканчивая государственными организациями, регулирующих работу того или иного сектора.

Основу нормативной базы по организации разработки безопасного программного обеспечения составляют:

- ГОСТ Р ИСО/МЭК 25010-2015 Информационные технологии. Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов [1];
- ГОСТ Р 56939–2016 Защита информации. Разработка безопасного программного обеспечения. Общие требования [2].

Первый из них, в контексте безопасности ПО, закрепляет понятия оценки защищённости, при оценке качества ПО, что активно используется в других нормативных актах [16]. Второй же составляет базу разработки безопасного ПО с внедрением дополнительных процессов обеспечения защищённости на различных этапах жизненного цикла проекта. Например, описывается:

- работа с требованиями по безопасности;
- выделение обязательных и желательных процедур обеспечения защищённости;
- проведение статического и динамического анализа, а также моделирования на проникновение.

Также внимания заслуживает проект ГОСТ-а по безопасному компилятору C/C++ в качестве примера введение нормативного контроля на используемые при разработке программные средства и введения профилактики основных уязвимостей кода конкретных языков программирования [3].

Проекты ФСТЭК в области реализации подхода разработки безопасного ПО

Федеральная служба по техническому и экспортному контролю помимо нормативных актов выпускает множество проектов методик и руководств для реализации подхода разработки безопасного ПО, которые в будущем могут быть выпущены в качестве национального стандарта. Так, данные труды призваны формализировать и повысить качество исполняемых процедур при разработке программных продуктов и некоторые из них будут рассмотрены далее.

Проект методики реализации ФСТЭК [5] предлагает расширение имеющегося ГОСТ Р 56939 и 58412 для установления регламента процесса разработки ПО с уделением дополнительного внимания к рабочим инструментам программиста и к ролевой модели команды в процессе создания проекта. Добавляются новые роли с указанием их зон ответственности и обязанностей в ходе жизненного цикла продукта.

Помимо методики реализации, ФСТЭК также выпустил и методику тестирования [6] обновлений безопасности программных, программно-аппаратных средств. Данный документ включает множество методов по сверке идентичности, проверки подлинности, антивирусному контролю обновлений и др. Методика содержит порядок и содержание работ по тестированию программного обеспечения, в том числе с открытым исходным кодом, предназначенного для устранения уязвимостей.

Проект руководства по оценке безопасности разработки программного обеспечения [4] приводит формализацию данной процедуры, а также предлагает подход к оценке мер при разработке ПО с учётов взаимосвязей с другими нормативными актами. В рамках этого руководства приводятся как список документов, необходимых при приёмке, так и обязанности и ответственность сторон.

Ведомственные и корпоративные регламенты в области безопасности ПО

Множество компаний и государственных организаций, которые занимаются контролем за безопасностью информационных систем, имеют собственные методические указания или справочники по обеспечению их безопасности.

Так, примером одного из таких документов является «Профиль защиты прикладного программного обеспечения автоматизированных систем и приложений кредитных организаций и некредитных финансовых организаций» [7]. Это методический документ, выпущенный Центральным банком России, и определяет требования к безопасности в автоматизированных системах. Данный документ содержит методические указания по созданию в типовых структурных элементах программных продуктов, а также указания по организации этапов жизненного цикла для обеспечения гибкой безопасной разработке и тестированию.

Международный стандарт OWASP

Помимо отечественных национальных стандартов, существует множество международных стандартов и проектов по реализации подхода разработки безопасного ПО. Примером является «Стандарт подтверждения безопасности приложений OWASP» (ASVS) [9]. Данный стандарт описывает требования к обеспечению защищённости при разработке, путём методических указаний к построению архитектур программных продуктов, хранению данных, авторизации пользователей в системе и др. Структурно данный стандарт аналогичен методическим указаниям Центрально банка России, рассмотренного ранее, и предлагает комплексное решение проблем безопасности ПО на этапе разработки требований.

Также, существует проект «Проактивная защита: Топ-10 требований OWASP» [8], направленный на разработчиков безопасного ПО для обеспечения безопасности продукта на ранних этапах создания. Для этого формулируются 10 рекомендаций, которые касаются как архитектурных подходов, так и конкретных решений при разработке проекта [10].

Анализ уязвимостей

Несмотря на значительный прогресс сообщества в создании нормативной базы и утверждённых подходах к созданию защищённых программных комплексов, немаловажной в данном вопросе является задача анализа существующих уязвимостей. Индустрия сделала большой скачок в систематизации имеющейся информации о проблемах безопасности, что в свою очередь является значительным подспорьем в создании технических средств для определения наличия тех или иных уязвимостей.

Классификаторы уязвимостей

Современные классификаторы уязвимостей невозможно себе представить без стандартизированного подхода к организации определения и хранения найденных угроз информационной безопасности. До разработки единых стандартов классификации уязвимостей существовали проблемы с поддержкой множества различных баз данных уязвимостей с различной структурой, количеством и наполнением. Так, при использовании нескольких баз невозможно определить, если они ссылаются на одну и ту же угрозу безопасности. Это стало одной из причин создания единого стандарта классификации уязвимостей Common Vulnerabilities and Exposures (CVE), который поддерживается множеством современных реестров.

Стандарт Common Vulnerabilities and Exposures

Создание единого стандарта классификации уязвимостей Common Vulnerabilities and Exposures (CVE) в 1999 году от некоммерческой компании MITRE положило начало унификации уязвимостей информационной безопасности между базами хранения.

Ввиду решения проблем идентификации уязвимостей, а также его открытости, данный стандарт получил широкое распространение и поддержку со стороны производителей ПО, которые решают следующие задачи:

- поиск и сбор информации об уязвимостях ПО;
- их классификация;
- выделение CVE-идентификаторов;
- обновление поступающей информации в двух официальных каталогах – реестре уязвимостей CVE List и базе данных уязвимостей National Vulnerability Database (NVD).

Это позволило накопить информацию о более чем 98 тысячах записей уязвимостей и добавить поддержку CVE-идентификаторов в различные реестры.

Реестры уязвимостей

Используя полученные идентификаторы CVE, позволили сосуществовать множеству реестров, которые могут различаться по предоставляемому функционалу и описанию уязвимостей. Так, можно выделить несколько примеров:

- БДУ ФСТЭК России: база данных уязвимостей одной из основных организаций, занимающаяся информационной безопасностью на территории Российской Федерации [18].
- MITRE CVE List: база данных, заполняемая самой компанией MITRE.
- NVD: база данных национального института технологий и стандартов США [14].
- OSVDB: некоммерческий проект, представлявший независимую базу данных уязвимостей, с открытым исходным кодом [15].

В результате порой расходящегося наполнения данных баз на рынке появились и различные агрегаторы уязвимостей.

Агрегаторы уязвимостей

Агрегаторы уязвимостей обеспечивают автоматизированный сбор доступной информации с представлением дополнительных функций поиска и фильтрации информации. Примерами данных агрегаторов являются Vulners и CVEDetails.

Сканеры безопасности

Сканеры безопасности в общем случае можно разделить на:

– Сканеры сети;
– Анализаторы кода.

Сканеры системы работают с уже готовой системой и проводят проверку в 4 этапа [13]:

1. Обнаружение активных IP-адресов, портов, операционной системы и приложений [11].
2. Составление отчёта о безопасности [19].
3. Активные зондирующие проверки – проверка «цифрового слепка» ПО с имеющейся базой.
4. Имитация атак – необязательный этап, который эксплуатирует уязвимости ПО, однако может повредить проверяемый узел [17].

Примерами таких сканеров являются: Nmap, Nessus и Owasp Zed Attack Proxy и другие [12].

Минусом данного подхода к сканированию является то, что данные проверки производятся на поздних этапах разработки системы, что затрудняет превентивное включение в разрабатываемое программное обеспечение подходов к реализации идеи безопасного программирования.

Исправить данный недостаток призваны статические анализаторы кода. Данный подход позволяет сканировать код на предмет уязвимостей на этапе его написания.

3) Результаты и их обсуждение

Проектирование архитектуры ИС ведения базы знаний уязвимостей ПО

Рассмотрим общую архитектуру разрабатываемой системы, которая представлена в нотации C4 на верхнем уровне на рисунке 1. С разрабатываемой системой планируется работать программистам, которые пишут код и имеют потребность в получении готовых решений проблем информационной безопасности.

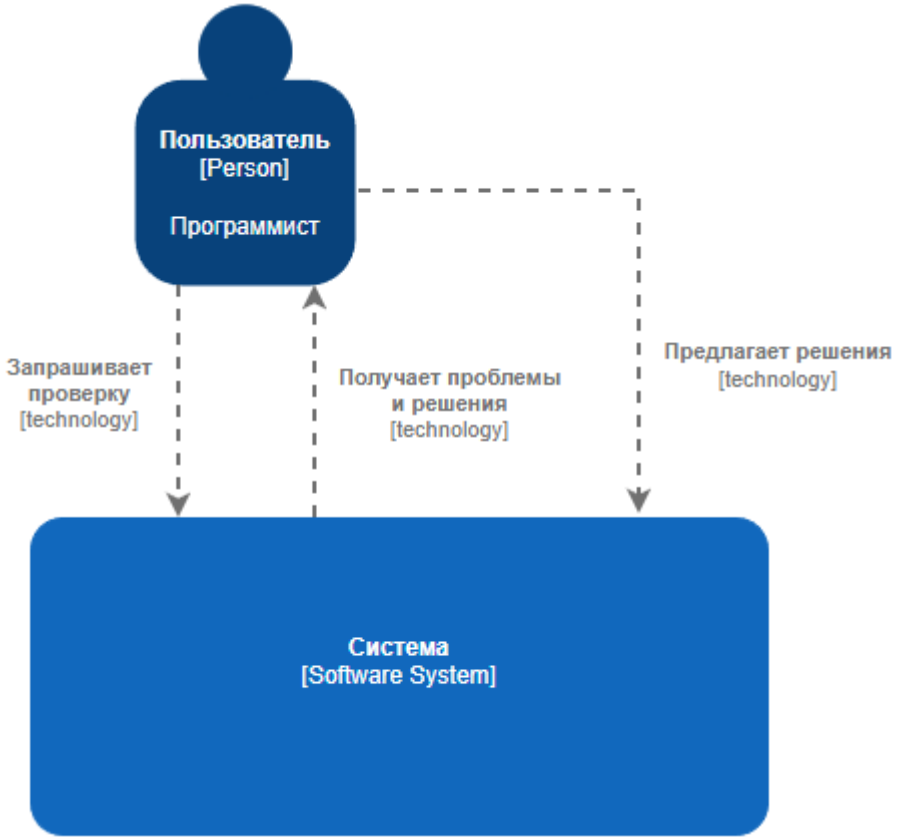

Рисунок 1 – Диаграмма C4 1 уровня разрабатываемой системы.

Исходя из данных запросов к системе, можно углубить детализацию компонентов системы, учитывая информацию о том, что программистам будет удобнее работать с системой, когда она встроена в средства разработки IDE. Также присутствует ограничение на то, что пользователей может быть множество. С учётом данных требований к архитектуре диаграмма второго уровня представлена на рисунке 2.

Формат расширения в IDE будет в данном случае лучшим вариантом клиентского приложения, которое устанавливается на стороне пользователя. В результате своей работы приложение посылает запросы к API серверу с запросами на получение и добавление решений пользователей. Ввиду некоммерческого характера работы системы необходимости в дополнительной авторизации пользователя нет необходимости.

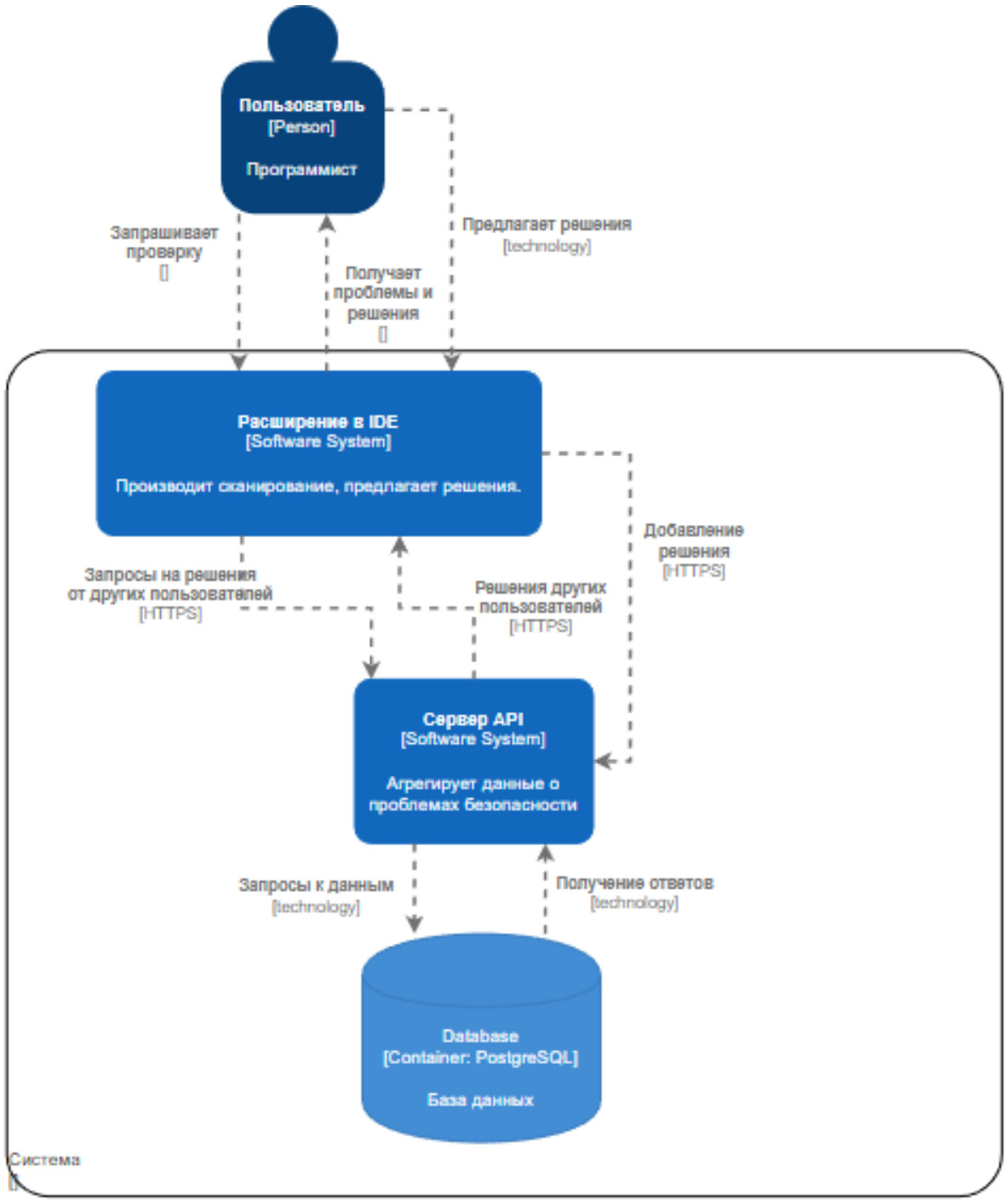

Рисунок 2 - Диаграмма С4 2 уровня разрабатываемой системы.

Далее, для уточнения наполнения запросов в системе раскроем содержание блока «Расширение в IDE», которое изображено на рисунке 3.

В ходе детализации получаем три основных компонента: «Сканер безопасности», «Анализатор решений» и «Анализатор выявленных проблем». Общая бизнес-логика взаимодействия данных компонентов состоит в том, что после получения запроса от пользователя системы «Сканер безопасности», который будет один из списка поддерживаемых системой, выполняет проверку на наличие уязвимостей кода, после чего полученные проблемы в «Анализаторе выявленных проблем» преобразовываются в шаблоны, которые в последствии отправляются с запросами на сервер с целью получения решений.

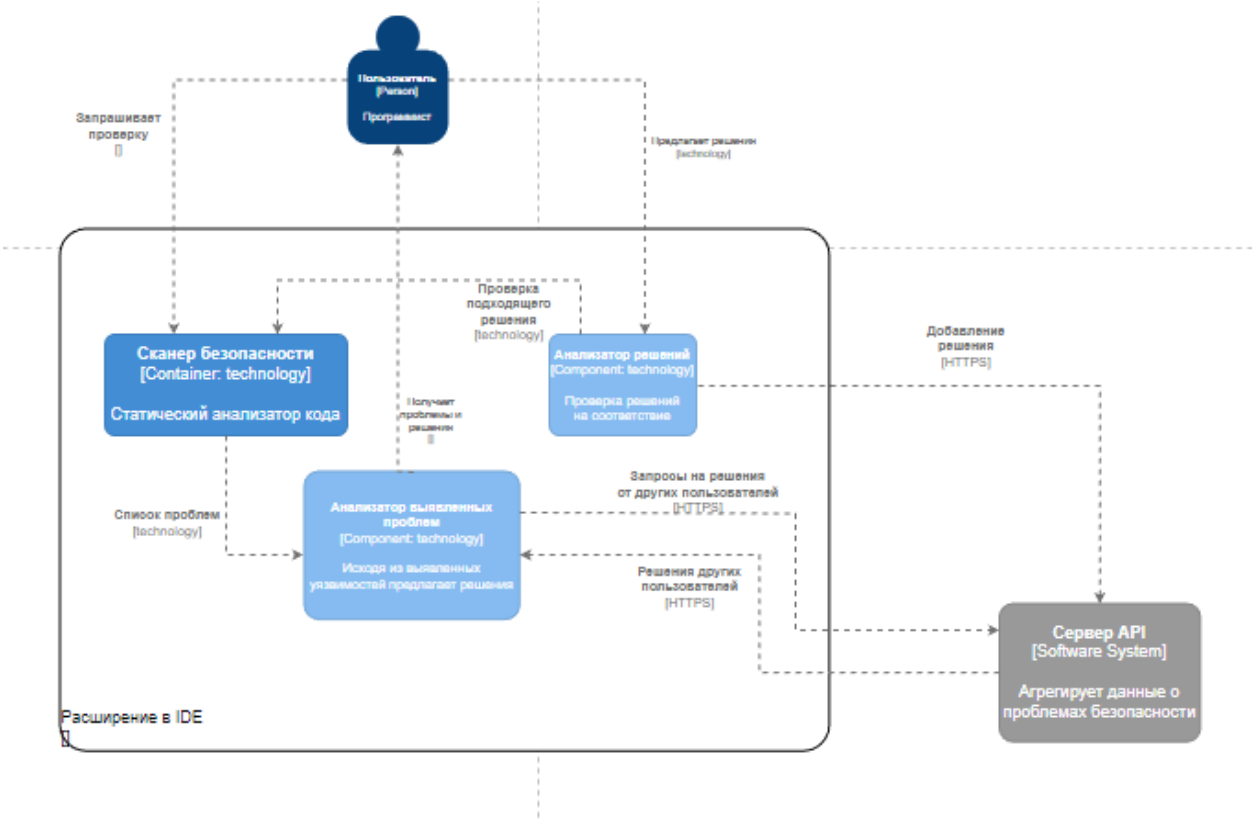

Рисунок 3 - Диаграмма C4 3 уровня с детализацией Расширения в IDE.

Также, в ходе развития системы планируется масштабирование путём увеличения количества поддерживаемых сканеров безопасности, что позволит обеспечить обхват большего количества платформ, языков программирования и потенциально практически безгранично развивать систему при наличии соответствующей программной и аппаратной поддержки.

Важно отметить, что при данном подходе, необходимо, чтобы сканеры безопасности были с открытым исходным кодом, либо была иная возможность анализа и доработки существующего сканера для поддержки его в рамках данной архитектуры.

Получаем, что в рамках данного запроса на получение решения проблемы, должен быть указан сканер и шаблон сообщения. Тогда в ответ анализатор получает от сервера API ограниченный список решений от различных пользователей и возможно рейтингами этих решений для улучшения подбора решений среди пользователей.

Далее, рассматривая функционал добавления решения от пользователя, сначала «Анализатор решений» получает команду от пользователя на добавление решения. После этого проверяется, что добавление этого решения привело к избавлению от уязвимости и в случае положительного результата посылается запрос на сохранение данного решения по проблеме на «Сервер API». Тогда в запрос будет входить: идентификатор проблемы по сканеру, какой-либо идентификатор пользователя для того, чтобы фиксировать отсутствие дублирующих запросов на сохранение решений и само решение, которое может быть оформлено в некоторую структуру данных, для чего на этапе проектирования предлагается формат хранения решений JSON.

Проектирование схемы базы данных

Учитывая проведённый ранее анализ предметной области и архитектуру построения системы, была разработана физическая схема данных, представленная на рисунке 4.

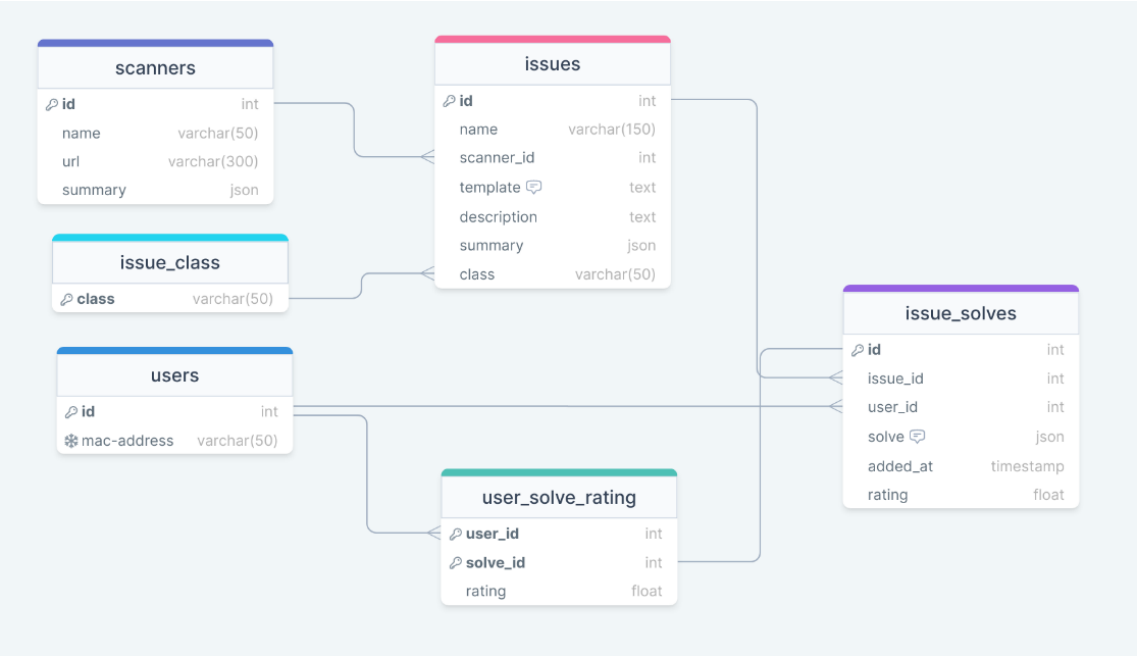

Рисунок 4 – Схема БД

Данная отношение сущностей позволяет хранить и масштабировать функционал со временем, так как в таблицах «scanners» «issues» и «issue_solves», в которых хранятся данные о сканерах, выявляемых ими уязвимостях и решениях соответственно, уже предусмотрено хранение дополнительных данных, необходимых для программной настройки сканеров, проблем и решений в форматах данных JSON. Также присутствует система рейтингов, отслеживающая уникальность оценки каждого пользователя, с логическими ограничениями на дублирование оценок и решений пользователей системы.

Важной особенностью данной модели хранения данных является то, что было отказано в хранении подробной информации об уязвимостях, так как это значительно увеличивает объём хранимых данных без функционального использования в процессе. Также использование готовых сканеров безопасности позволяет значительно повысить функционал и масштабируемость системы, снимая часть сложностей в поддержке системы для актуализации проблем безопасности.

Также, стоит отметить, что на данном этапе идентификация пользователей происходит по MAC-адресу устройства, на котором установлено расширение. В случаях масштабирования системы или при переходе на коммерческую модель распространения системы будет необходима доработка сущности пользователя.

Заключение

В ходе работы были рассмотрены существующие подходы к организации безопасного программирования с учётом отечественного и зарубежного опыта, проанализированы классификаторы уязвимостей и сканеры безопасности. На основе проведённого анализа были разработаны требования к информационной системе ведения базы знаний уязвимостей ПО. Проектирование системы включало последовательную детализацию системы в нотации C4 от контекстного представления системы к выбору основных технологий и их детализации через компоненты. Была создана физическая схемы данных для обеспечения хранения необходимой информации. Дальнейшая реализация системы будет основана на основе результатов, достигнутых в рамках данной работы, и. выполнена в формате расширения к IDE для удобства взаимодействия с пользователями(программистами). Будущее развитие системы может предполагать как переход на коммерческую основу распространения, что поможет интегрировать платные статические анализаторы и требует доработки сущности пользователя, либо введение

системы в эксплуатацию в качестве государственного некоммерческого сервиса с повышением защищённости и повышенными требованиями к доступности. В текущем исполнении система будет полезна компаниям или иным объединениям разработчиков, которые хотят стандартизировать подходы к решениям типовых проблем безопасности.

Список литературы:

1. ГОСТ Р ИСО/МЭК 25010–2015 Информационные технологии. Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов. М., 2015. 29 с.
2. ГОСТ Р 56939-2016 Защита информации. Разработка безопасного программного обеспечения. Общие требования. М., 2016. 19 с.
3. ГОСТ Р проект Безопасный компилятор языков Си/Си++. Общие требования. М. 13 с.
4. ГОСТ Р проект Руководство по оценке безопасности разработки программного обеспечения. М., 147 с.
5. ГОСТ Р проект Руководство по реализации мер по разработке безопасного программного обеспечения. М, 150 с.
6. Федеральная служба по техническому экспортному контролю. Методика тестирования обновлений безопасности программных, программно-аппаратных средств: метод. документ. М.; Москва, 2022. 16 с.
7. Центральный банк Российской Федерации. Профиль защиты прикладного программного обеспечения автоматизированных систем и приложений кредитных организаций и некредитных финансовых организаций : метод. документ. М.; Москва, 2021. 215 с.
8. Anton K., Manico J., Bird J. OWASP proactive controls for developers v 3.0. 10 critical security areas that software developers must be aware of. OWASP, 2018 – 39 p.
9. Holguera C., Müller B., Schleier S., Willemsen J. OWASP Standart. Mobile application security verification standart. Version v1.5.0. OWASP, 2023 – 49 p.
10. OWASP Proactive Controls: список обязательных требований для разработчиков ПО [Электронный ресурс]: Хабр // URL: https://habr.com/ru/company/jetinfosystems/blog/440202/ (Дата обращения: 20.12.2022).
11. Сетевые сканеры безопасности: возможности, принцип работы и передовые решения [Электронный ресурс]: Комсомольская правда // URL: https://www.kp.ru/guide/setevye-skanery-bezopasnosti.html (дата обращения: 15.10.2022).
12. Сканирование на уязвимости: обзор продуктов, которые есть на рынке [Электронный ресурс]: Хабр // URL: https://habr.com/ru/company/cloud4y/blog/651831/ (дата обращения: 10.10.2022).
13. Сканеры безопасности [Электронный ресурс]: StudFilles // URL: https://studfile.net/preview/4396380/ (дата обращения: 15.10.2022).

14. Классификаторы безопасности *[Электронный ресурс]:* Safe surf *// URL https://safe-surf.ru/specialists/article/5228/607311/ (дата обращения: 16.10.2022).*
15. Меряем уязвимости: классификаторы и метрики компьютерных брешей *[Электронный ресурс]:* Хакер *// URL:* https://xakep.ru/2009/05/15/48221/ *(дата обращения: 15.10.2022).*
16. Уязвимости информационных систем *[Электронный ресурс]:* Русский стандарт *// URL:* https://intelbit.ru/upload/iblock/119/r1u0umnn6sh70s6vh1w9qrudtql4xmtw.pdf(дата *обращения: 10.10.2022).*
17. Классификация уязвимостей. *[Электронный ресурс]:* Studfile *// URL: https://studfile.net/preview/1464207/page:2/ (дата обращения: 15.10.2022).*
18. Банк данных угроз безопасности информации *[Электронный ресурс]:* БДУ уязвимостей *// URL:* https://bdu.fstec.ru/vul *(дата обращения: 15.10.2022).*
19. Сканеры уязвимостей [Электронный ресурс]: Wikipedia // URL https://ru.wikipedia.org/wiki/Сканеры_уязвимостей (дата обращения: 16.10.2022).

References:

1. GOST R ISO/IEC 25010–2015 Information technologies. System and software engineering. Requirements and quality assessment of systems and software (SQuaRE). Quality models of systems and software products. M., 2015. 29 p.
2. GOST R 56939-2016 Information security. Development of secure software. General requirements. M., 2016. 19 p.
3. GOST R Project Safe compiler of C/C++ languages. General requirements. M. 13 p.
4. GOST R Draft Guidelines for assessing the security of software development. M., 147 p.
5. GOST R Draft Guidelines for the implementation of measures to develop secure software. M, 150 p.
6. Federal Service for Technical Export Control. Methodology for updating the security of software, firmware and hardware: method. document. M.; Moscow, 2022. 16 p.
7. Central Bank of the Russian Federation. Protection Profile of Application Software for Security Systems and Applications of Credit Institutions and Non-Bank Financial Institutions: Method. document. M.; Moscow, 2021. 215 p.
8. Anton K., Maniko J., Bird J. OWASP Proactive Controls for Developers v 3.0. 10 Critical Security Areas Software Developers Should Be Aware of. OWASP, 2018 - 39 p.
9. Holgera K., Müller B., Schleyer S., Willemsen J. OWASP Standard. Mobile Application Security Testing Standard. Version v1.5.0. OWASP, 2023 - 49 p.
10. OWASP Proactive Controls: a list of mandatory requirements for software protection [Electronic resource]: Habr // URL: https://habr.com/ru/company/jetinfosystems/blog/440202/ (Date of access: 12/20/2022).
11. Network security scanners: capabilities, principles of operation and advanced solutions [Electronic resource]: Komsomolskaya Pravda // URL: https://www.kp.ru/guide/setevye-skanery-bezopasnosti.html (date of access: 10/15/2022).).

12. Scanning for vulnerabilities: an overview of products that are on the market [Electronic resource]: Habr // URL: https://habr.com/ru/company/cloud4y/blog/651831/ (date of access: 10/10/2022).
13. Security scanners [Electronic resource]: StudFiles // URL: https://studfile.net/preview/4396380/ (date of access: 10/15/2022).
14. Safety classifiers [Electronic resource]: Safe surfing // URL https://safe-surf.ru/specialists/article/5228/607311/ (date of access: 10/16/2022).
15. We measure vulnerabilities: classifiers and metrics of computer gaps [Electronic resource]: Hacker // URL: https://xakep.ru/2009/05/15/48221/ (Accessed: 10/15/2022).
16. Vulnerabilities of information systems [Electronic resource]: Russian standard // URL: https://intelbit.ru/upload/iblock/119/r1u0umnn6sh70s6vh1w9qrudtql4xmtw.pdf (date of access: 10.10.2022).
17. Classification of vulnerabilities. [Electronic resource]: Studfile // URL: https://studfile.net/preview/1464207/page:2/ (date of access: 10/15/2022).
18. Data bank of information security threats [Electronic resource]: BDU of vulnerabilities // URL: https://bdu.fstec.ru/vul (date of access: 10/15/2022).
19. Vulnerability Scanners [Electronic resource]: Wikipedia // URL https://ru.wikipedia.org/wiki/Vulnerability_Scanners (Accessed: 10/16/2022).