УДК 004

Модель информационной системы запуска недоверенного кода в веб-ориентированных средах

Фатеенков Данила Витальевич
Приамурский государственный университет имени Шолом-Алейхема
Студент

Аннотация
В рамках данной статьи описывается модель информационной системы, основным назначением которой является компиляция и запуск недоверенного кода в веб-ориентированных средах (клиент-серверных приложениях и сайтах). Представлены диаграммы, отражающие суть и принцип работы модулей, предполагающихся в ИС. Также описаны технологии, применяемые в разработке информационной системы и особенности работы самой системы.
Ключевые слова: изолированная среда, виртуализация, контейнеризация, онлайн-компилятор, обучающая система, Docker

Information system model for running untrusted code in web-oriented environments

Fateenkov Danila Vitalievich
Sholom-Aleichem Priamursky State University
Student

Abstract
This paper describes a model of an information system, the main purpose of which is to compile and run untrusted code in web-oriented environments (client-server applications and websites). Diagrams reflecting the essence and principle of operation of the modules assumed in the IS are presented. The technologies used in the development of the information system and the peculiarities of the system itself are also described.
Keywords: isolated environment, virtualisation, containerisation, online compiler, learning system, Docker

1. Введение

1.1 Актуальность

Темп развития информационных систем сложно недооценить на сегодняшний день. Каждый месяц появляются новые технологии и разработки в IT-сфере, направленные на упрощение разработки приложений или на улучшение уже существующих решений. С созданием новых технологий также появляются и методики поиска уязвимостей ИС для последующего нарушения работы модулей или систем в целом. Особо остро

вопрос безопасности стоит в системах, которые работают с недоверенным кодом. Недоверенным кодом считается любой код, который загружает пользователь и который не предполагается системой изначально. Недоверенный код является слабым звеном большинства ИС, которое разработчики не прорабатывают тщательно, что может привести к последующим нарушениям безопасности приложения. Самым популярным методом обеспечения защиты является виртуализация и запуск ИС в контейнере.

1.2 Обзор исследований

Н. В. Филиппов и Н. В. Киреева провели анализ безопасного запуска кода в изолированной среде web-приложений [1].

М. К. Солонько и К.А. Краснов рассмотрели программную платформу виртуализации Docker [2]. Также в статье рассмотрен процесс создания и запуска контейнера на конкретном примере.

Н. А. Афанасьева в своей работе рассмотрела аспекты использования SOAP протокола и REST стиля в разработке веб-сервисов и выделила различия между REST и SOAP веб-сервисами [3].

Можно выделить научную работу под авторством Л. К. Бабенко и А. С. Кириллова «Разработка автоматизированной системы обнаружения вредоносного программного обеспечения», в которой были раскрыты технические требования к проектируемой системе обнаружения вредоносного программного обеспечения, обусловленные предложенным ранее методом обнаружения и кластеризации [4].

1.3 Цель исследования

Цель – разработать модель информационной системы запуска недоверенного кода на стороне сервера веб-ориентированных приложений (сайты и клиент-серверные прикладные программы).

2 Материалы и методы

Для разработки используется Docker, Bash и PostgreSQL. ИС строится на RESTful технологиях и работает через HTTP-запросы. Представленные схемы и диаграммы выполнены по соответствующим им методологиям.

3 Результаты и обсуждения

В ходе проектирования архитектуры ИС было решено построить модель ИС по RESTful технологии, то есть разработать не отдельную программу или сайт, а представить разработчикам инструмент для интеграции эффективного и безопасного модуля для компиляции и запуска недоверенного кода на сервере сторонних систем. Для того, чтобы реализовать поставленную задачу, архитектура системы проектировалась по принципам работы API. Это значит, что ИС будет представлена в виде веб-ориентированного приложения, у которого будет присутствовать самый базовый интерфейс для отладки работы основных модулей, при этом будет

настроен доступ к модулям по средством отправки GET и POST запросов на определённые адресы системы. При этом разработчикам также будет предоставлена возможность развёртывания сайта в файловой системе API-сервиса.

Для дальнейшего развёртывания API-сервиса будет предложено разработчикам 2 возможных варианта использования системы: подключение к API через Rapid API или самостоятельное развёртывание на локальном сервере.

Rapid API – это платформа для разработчиков, предоставляющая доступ к тысячам API от различных провайдеров. В основе Rapid API лежит идея объединения и стандартизации доступа к разнообразным API, что позволяет разработчикам быстро интегрировать функциональность из различных сервисов в свои приложения.

Были выделены следующие цели, которые должен выполнять разрабатываемый API-сервис:

1. Предоставлять разработчикам открытую и бесплатную платформу, которую можно будет масштабировать под нужды ИС, для которой необходим функционал для запуска недоверенного кода.
2. Сделать возможным развёртывание собственных информационных систем в том же контейнере, где находится сам API-сервис. Такая возможность предназначена для разработчиков, которые только приступают к разработке собственных ИС.
3. Предоставить на выбор актуальные компиляторы самых популярных языков программирования в настоящее время.
4. Предоставить эффективные алгоритмы компиляции, изоляции запуска недоверенного кода. API-сервис должен быть гибким в настройке, чтобы он был применим к большому количеству веб-ориентированных и прикладных систем.

Актуальность разработки данной ИС можно объяснить не только постоянным ростом количества способов обхода защиты сайтов, но и также отсутствием на рынке информационных технологий (в российском сегменте) в настоящее время решений, которые могли бы выполнить описанные выше цели. Существующие сервисы по компиляции кода не предоставляют доступа к API, что делает невозможным их использование за рамками исходной системы. Некоторые сервисы могут предоставить модули для запуска недоверенного кода, но за определённую плату, что тоже не соответствует цели разрабатываемой ИС.

Запросы к API выполняются в соответствии со структурой HTTP протокола. Это значит, что передаваемая информация между пользователем и сервером, которая проходит через API-сервис, обрабатывается с помощью стандартных GET, POST, DELETE и др. запросов. Такой подход позволяет обеспечить независимость API от используемого веб-сервера.

Работа API-сервиса обеспечивается за счёт конечных точек (англ. endpoint), которые указывают на определённые функциональные модули API.

К функциональным модулям относятся: модуль запуска изолирующей среды, модуль запуска компиляторов, обработчики событий в ОС, СУБД.

API работает по принципу токенизации данных: каждому запуску изолирующей среды присваивается свой уникальный токен.

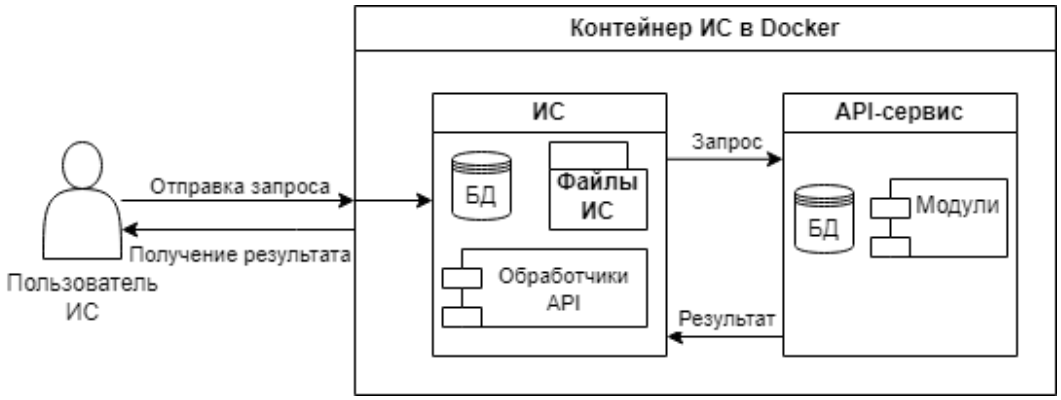

Рисунок 1. Общий принцип работы ИС

Разработанный API-сервис можно будет внедрить в большое количество веб-ориентированных решений:

1. Обучающие системы, такие как Stepik и др.
2. Сайты, предоставляющие функционал по компиляции и запуску пользовательского кода (онлайн компиляторы), например OnlineGDB.
3. Сайты, предназначенные для организации проведения чемпионатов по программированию, например CodeForces.

При проектировании ИС были собраны следующие исходники компиляторов и интерпретаторов:

1. GCC 13.3.0 – предназначен для компиляции и запуска кода, написанного на C и C++. На данный момент является последней версией GNU компиляторов, в которые входят C, C++, Ada, Fortran, Objective-C, Go и др., но будут установлены только C и C++.
2. CPython – базовый интерпретатор языка программирования Python. Является наиболее популярным интерпретатором в настоящее время, так как предустановлен в классический Python. Было решено задействовать 2 версии языка программирования: 3.12.4 (актуальная версия на момент публикации статьи) и 2.7.17.
3. PyPy – высокопроизводительная реализация JIT-кода для Python. Хорошо подходит для требовательных программ и ситуаций, когда могут возникать проблемы с производительностью запущенного кода. В API на старте будет присутствовать наиболее актуальная версия интерпретатора: 7.3.15 (актуальная версия на момент публикации статьи). Была выбрана последняя на момент написания версия PyPy, а именно 7.3.16, в рамках которой также выбраны версии Python 2.7 и 3.10.
4. PHP – язык программирования для запуска кода в веб-ориентированных средах. Интерпретатор языка предполагается стандартный,

так как является наиболее надёжным решением для большинства случаев. Предполагается доступ к последней версии, а именно к версии 8.3.8.

5. Assembly – низкоуровневый язык программирования, позволяющий записывать инструкции, которые процессор компьютера может выполнять напрямую. Существует несколько версий данного языка, в ИС предполагается использование NASM последней версии.

6. Java – популярный язык программирования, который часто используется в олимпиадном программировании. Предполагается использование последней версии OpenJDK.

Представлена только часть компиляторов и интерпретаторов, на старте в API предполагается возможность использования 21-го языка программирования. Также предполагается возможность загрузки в API исполняемых файлов и проектов с неопределённым количеством файлов.

Первым этапом разработки ИС стало создание изолирующей среды для запуска кода.

Изолирующая среда является ядром всего API-сервиса и представляет собой первый уровень защиты API-сервиса. Защита на данном уровне направлена на ограничение доступа к файловой системе ИС и API-сервиса для пользователя, который будет запускать код. В таких разработках необходимо учитывать риск попытки запуска вредоносного кода, который может навредить работе ИС (перезагрузить сервер, например). Предполагается, что изолирующая среда является единственным элементом во всём API, который нельзя менять.

Изолирующая среда не только защищает основную систему от несанкционированного доступа и вирусов, но и также предоставляет разработчику отладочную информацию о работе того или иного процесса. Изолирующая среда учитывает следующую информацию о работе процесса:

1. Время работы процесса, которое возвращается пользователю в миллисекундах.
2. Максимальное потребление памяти процессом, которое возвращается в килобайтах.
3. Код и сигнал завершения работы. Так как API проектируется с учётом дальнейших развёртывания и запуска системы на UNIX-подобных системах (Linux, Ubuntu, Debian и др.), то разработчику будет предоставляться информация о кодах завершения работы процесса, по которой можно будет определить причину прекращения процесса.
4. Максимально допустимое ограничение по времени и памяти. Пользователь сможет ограничивать ресурсы, чтобы процесс не использовал всю доступную память или не работал слишком долго.
5. Входные параметры, которые подаются на старте работы программы.
6. Ожидаемый и полученный результаты работы программы. Пользователь может определять ожидаемый (какой должен получиться по

мнению пользователя) результат и сверять его с полученными после завершения работы процесса.

7. Сообщения об ошибках. Если код не получилось скомпилировать, то пользователь сможет получить подробную информацию об ошибке компиляции (предоставляется самим компилятором или интерпретатором).

8. Язык программирования, который был использован для выбранного ID.

9. Время создания процесса и время завершения работы изолирующей среды.

10. Отладочная информация: настройки компилятора, опции командной строки, перенаправление вывода в особые потоки, максимальное количество потоков, токен и другие параметры изолирующей среды.

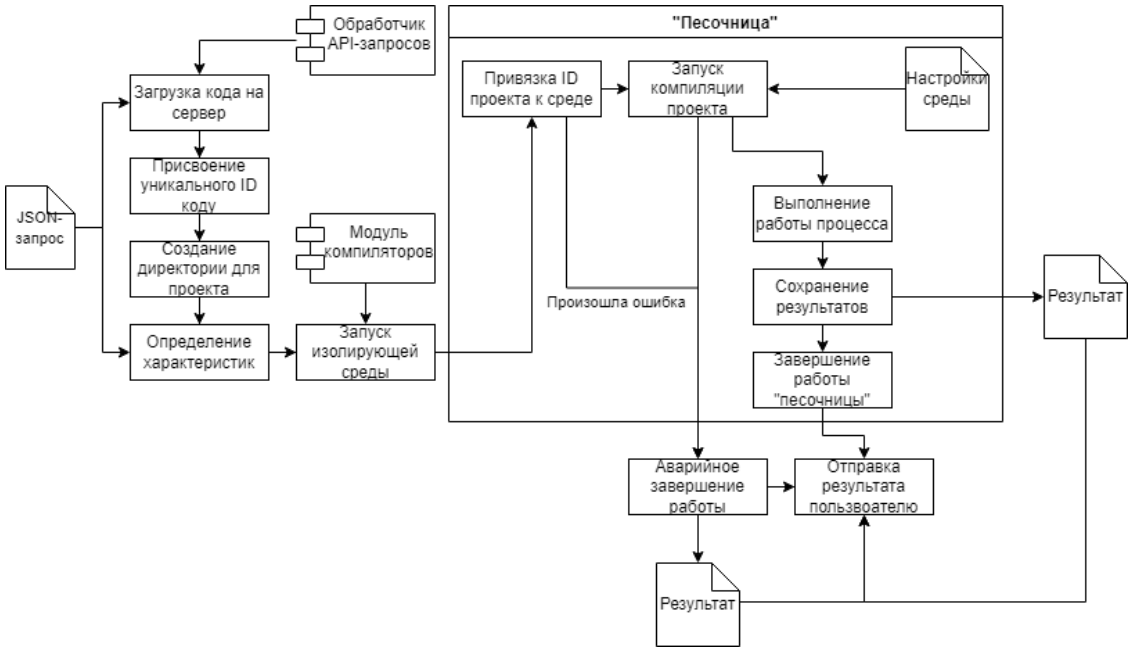

Рисунок 2. Схематичное представление работы изолирующей среды в проектируемой ИС

В API-сервисе также предусмотрена возможность работы с исполняемыми файлами. Пользователям ИС, в которых задействован рассматриваемый API, нужно учитывать сильную привязку сервиса к ОС, на которой он запущен. Это значит, что запуск файлов формата .exe будет невозможен, так как API работает только на UNIX-подобных системах.

В архитектуре API-сервиса предусмотрена возможность загрузки файлов в виде закодированного в Base64 архива, что позволяет пользователям запускать исполняемые файлы и проекты, представленные в виде некоторого количества модулей и файлов.

Сильную привязку к ОС было решено компенсировать добавлением ядра ОС Debian в проект API-сервиса. И хоть такое решение позволило запускать API на серверах, работающих под управлением Windows, проблема с форматом исполняемых файлов не была решена.

Работа с исполняемыми файлами также предусматривает запуск изолирующей среды, в которую помещаются все файлы проекта пользователя. Некоторые статусы ошибок становятся недоступными из-за невозможности их проверки в реальных условиях (например, ошибка компиляции невозможна, так как приложение уже заранее скомпилировано пользователем).

Рисунок 3. Диаграмма, описывающая процесс запуска исполняемого файла

API-сервис проектируется с учётом возможности работы с различными СУБД. Подход к выполнению запросов к базам данных несколько отличается от запуска стандартного кода, так как в большинстве случаев требует наличия базы данных, к которой пользователь будет выполнять запросы. Исходя из данной информации, возникают следующие задачи, с которыми невозможно столкнуться в стандартных сценариях работы API:

1. Необходимость временного хранения БД на стороне сервера.
2. Установка связи между клиентом и сервером во время работы SQL обработчиков, что может быть сложной задачей с учётом того, что выбранные СУБД работают по разным принципам.
3. Обеспечение безопасности при работе с создаваемыми пользователями БД.

На данный момент предусмотрена работа только с SQLite, так как данная СУБД работает с БД, которая представлена в файловом виде, что делает её удобной для передачи информации между клиентом и сервером.

В рамках проведённого исследования была описана модель API-сервиса, основной задачей которого является запуск недоверенного кода в изолирующей среде. Разработчики смогут использовать API для интеграции функционала в свои ИС и приложения.

API-сервис предлагает разработчикам гибкую архитектуру, которую можно настроить в зависимости от собственных предпочтений. На выбор разработчикам предложено более 20-и компиляторов и интерпретаторов. Конфигурационные файлы предлагают ряд настроек, которые позволят настроить систему под возможности сервера.

Следующие этапы разработки сервиса состоят, в первую очередь, из внедрения API в реальную систему, где предполагается работа с недоверенным кодом, а также разработка ИС на базе созданного API-сервиса (онлайн-компиляторы, системы обучения программированию и др.).

Библиографический список

1. Филиппов Н.В., Киреева Н.В. Анализ возможности безопасного запуска кода в изолированной среде web-приложений // XXIX Российская научно-техническая конференция: Материалы XXIX Российской научно-технической конференции профессорско-преподавательского состава, научных сотрудников и аспирантов университета с приглашением ведущих ученых и специалистов родственных вузов и организаций, Самара. Самара: Поволжский государственный университет телекоммуникаций и информатики. 2022. С. 51-52.
2. Солонько М.К., Краснов К.А. Docker-контейнеры // Телекоммуникации и информационные технологии. 2020. Т.7. № 1. С. 65-69.
3. Афанасьева Н. А. ВЕБ-СЕРВИСЫ: REST и SOAP // Вестник образовательного консорциума Среднерусский университет. Информационные технологии. 2021. № 2(18). С. 26-28.
4. Бабенко Л.К., Кириллов А.С. Разработка автоматизированной системы обнаружения вредоносного программного обеспечения // Известия ЮФУ. Технические науки. 2021. №7 (224).