Метод аутентификации на удаленном сервере для обеспечения непрерывности обработки данных в защищенных системах

П.А. Головняк, К.С. Зайцев, А.М. Пинчук

***Аннотация*. Сегодня наблюдается значительный рост числа платежей, выполняемых с использованием пластиковых карт. Банковские организации в своей работе вынуждены анализировать огромные объемы данных, в связи с чем в эксплуатацию внедряются программные продукты для обработки больших данных. Популярным инструментом для потоковой и батчевой обработки данных является фреймворк Apache Spark. Для анализа информации часто используется программное обеспечение запуска процессов по расписанию. Примером такого программного обеспечения является Apache Airflow. Запускаемые процессы в Airflow принято представлять в виде направленного графа без циклов, при этом вершинами графа являются задачи, которые являются реализацией определённого оператора Airflow. Целью статьи является разработка Apache Airflow оператора для запуска Spark-задач на сервере, отличном от того, на котором функционирует Airflow. При разработке подобного оператора возникает задача аутентификации на удалённом сервере. В статье предлагается решение проблемы аутентификации за счёт использования токенов доступа, предоставляемых хранилищем секретов Vault, а также веб-сервиса собственной разработки, с помощью которого выполняется менеджмент секретов и токенов, хранящихся в Vault. В результате работы получено решение, позволяющее выполнять запуск Spark-задач в Airflow на удалённом сервере, при этом используемые для аутентификации секреты хранятся в Vault, что повышает защищенность системы.**

***Ключевые слова* – аутентификация, Apache Airflow, Apache Spark, HashiCorp Vault, Spark-оператор, токен доступа.**

I. ВВЕДЕНИЕ

На конец первого квартала 2021 года количество банковских карт, эмитированных кредитными организациями и Банком России, достигло максимального значения за всю историю[1]. Число активных банковских карт составило 284 млн, что на 9 млн больше, чем в аналогичный период 2020 года. Рост числа банковских карт влечёт за собой увеличение числа мошеннических операций с их использованием. Так, по данным Банка России, рост числа операций по переводу денежных средств, совершенных без согласия клиентов в канале дистанционного банковского обслуживания, за второй квартал 2021 составил 52% по сравнению с аналогичным периодом 2020 года [2]. Наиболее распространёнными типами атак являются атаки с использованием методов социальной инженерии. Согласно статье 159.3 УК РФ, мошенничество с банковскими картами — это хищение чужого имущества, которое совершено с использованием принадлежащей другому человеку или поддельной расчётной, кредитной или иной карты посредством обмана сотрудника торговой, финансовой или иной организации.

В связи с ростом числа мошеннических операций руководство банков формирует специальные отделы, целью функционирования которых является как оперативная реакция на перевод денежных средств без согласия клиента, так и использование современных подходов для автоматизированного предотвращения мошенничества в различных платёжных каналах (кибермошенничества) [3]. Следует отметить, что мошенничество и кража личных данных наносят ущерб не только пострадавшему человеку, но и банкам и платёжным сетям, которые в некоторых случаях возмещают ущерб пострадавшему. Правительственные организации, такие как федеральная торговая комиссия США, задействуют существенные ресурсы для расследований, целью которых является возврат средств пострадавшим и обнаружение мошеннических компаний [4]. Более того, жертвы несут не только сиюминутные финансовые потери; владельцы карт могут запятнать свои кредитные скоринги, потратить много времени на

Статья получена 15 июня 2022.

Головняк Павел Алексеевич, Национальный Исследовательский Ядерный Университет МИФИ, магистрант, pgolovnyak@gmail.com

Зайцев Константин Сергеевич, Национальный Исследовательский Ядерный Университет МИФИ, профессор, KSZajtsev@mephi.ru

Пинчук Андрей Михайлович, ПАО Сбербанк, исполнительный директор, pichuk.a.m@sberbank.ru

разбирательства и попытки вернуть утерянные деньги.

Непрерывный рост числа банковских карт влечёт за собой рост общего числа банковских транзакций, которые должны быть обработаны.

Отделы противодействия кибермошенничеству современных банков, как правило, используют технологии, относящиеся к технологиям больших данных [5]. При обработке больших данных возникает необходимость построения конвейеров доставки и обработки данных (англ. pipelines). Для этого используются специальные программные продукты, например, такие как Apache Airflow, Apache Oozie и др.

Актуальность работы обусловлена тем, что набирающий популярность инструмент построения конвейеров для обработки данных Apache Airflow не позволяет запускать задачи в рамках технологии для обработки больших данных Apache Spark, в связи с чем у компаний возникает необходимость исследования исходного кода библиотеки и реализации собственного Spark-оператора. Одной из важных частей разработки Spark-оператора является аутентификация на сервере, имеющем доступ к Apache Spark.

В настоящей статье приводится решение, позволяющее реализовать Spark-оператор для Apache Airflow, рассматриваются программные инструменты, которые могут быть использованы для менеджмента учётных данных пользователей, используемых для аутентификации на сервере, имеющем доступ к Spark.

II. АНАЛИЗ ВОЗМОЖНОСТЕЙ APACHE AIRFLOW

Apache Airflow представляет собой платформу для программного создания, выполнения по расписанию и мониторинга рабочих процессов (pipelines – *англ.* конвейер) [6]. В рамках Airflow рабочие процессы представляются в виде направленного графа без циклов, вершины которого называются задачами (task – *англ.* задача). Планировщик Airflow (scheduler – *англ.* планировщик) отвечает за распределение задач по исполнителям (worker – *англ.* рабочий), при этом задачи в одном рабочем процессе могут быть зависимы друг от друга. Пользовательский интерфейс Airflow позволяет визуализировать последовательность обработки данных, отслеживать прогресс выполнения и устранять ошибки при необходимости.

Процессы в Airflow представляются в виде направленного графа без циклов, при этом каждая вершина графа является определённым оператором Airflow. Оператор – шаблон для задачи конкретного типа. В Airflow представлен ряд операторов доступных сразу поле установки: BashOperator – оператор для выполнения команд интерфейса командной строки Bash, PythonOperator – оператор для выполнения программного кода языка программирования Python, EmailOperator – оператор для отправки сообщения по электронной почте, SimpleHttpOperator – оператор для отправки HTTP запроса, SQLiteOperator – оператор для выполнения запроса в базу данных SQLite и др.

Планировщик Airflow предоставляет возможность реализовать собственный оператор. Язык разработки оператора – Python, поскольку Airflow также реализован на Python.

Для написания собственного Apache Airflow оператора необходимо реализовать класс-наследник родительского класса для всех операторов - BaseOperator. Реализуемый класс должен переопределять два метода родительского класса:

- метод __init__, в рамках которого выполняется инициализация необходимых для работы оператора переменных;
- метод execute – программный код, который будет исполняться при вызове оператора. Один из аргументов функции execute – переменная context типа dict (словарь), содержащая информацию о параметрах конфигурации текущего потока [7].

Следует отметить, что помимо функциональности, которая будет выполняться в рамках оператора, возможно также определить вид оператора в пользовательском интерфейсе Airflow.

Ещё одна важная особенность, которую необходимо учитывать при написании Airflow оператора – возможность параметризации оператора с помощью шаблонизатора Jinja [8]. Под шаблонизацией понимается определение какого-либо параметра оператора непосредственно во время выполнения кода. Для шаблонизации поля необходимо определить атрибут класса template_fields, при этом тип template_fields – список языка Python, содержащий все шаблонизируемые поля.

III. ФОРМУЛИРВОКА ТРЕБОВАНИЙ К РАЗРАБАТЫВАЕМУ SPARK-ОПЕРАТОРУ

Apache Spark – программная платформа (framework – англ. структура, каркас) для обработки больших данных, входящая в экосистему Hadoop [9]. Hadoop представляет собой программную платформу для одновременной обработки данных на нескольких узлах кластера.

Apache Spark, в отличие от классического обработчика ядра Hadoop, использует парадигму резидентных вычислений (англ. in-memory computing), согласно которой данные сохраняются в оперативной памяти узлов кластера, что позволяет получить существенный выигрыш в скорости работы в сравнении MapReduce, в котором выполняется сохранение результатов на диск после каждого этапа обработки данных. Более того, использование вычислений с сохранением результатов в оперативной памяти позволяет реализовывать алгоритмы машинного обучения на кластере [10].

Существует два способа запуска Spark-задачи на узле кластера. Первый из них – выполнение команды *spark-submit* в интерпретаторе командной строки. В случае, если исполняется JAR-файл (джава архив – англ. java archieve), то JAR должен представлять собой fat (uber) JAR, т. е. все зависимости проекта должны содержаться внутри исполняемого архива. Вызов команды *spark-submit* представлен на рисунке 7. Для запуска jar необходимо указать класс, который должен быть исполнен (ключ --class), а также путь до исполняемого JAR (<application-jar>). В случае запуска программы, написанной на языке Python, в аргументе команды <application-jar> должен быть указан путь до файла с расширением .py. Все зависимости, необходимые для программы на языке Python могут быть упакованы в .egg или в .zip архив и переданы через аргумент --py-files. Через ключ --conf передаются параметры запуска задачи на кластере, такие как количество памяти на исполнителя, количество памяти на мастер-процесс, порт для отображения пользовательского интерфейса исполняемой Spark-задачи и др [11].

```
spark-submit \
 --class <main-class> \
--conf <key>=<value> \
  ... # other options
<application-jar> \
[application-arguments]
```

Второй способ запуска Spark-задачи – запуск интерактивного интерпретатора командной строки для выполнения команд Spark. Для использования этого способа в командной строке необходимо выполнить команду spark-shell, которая предоставит доступ к REPL (англ. read-eval-print-loop – цикл чтения-вычисления-отображения) языка Scala.

В работе рассматривается архитектура серверов для работы с Airflow и Spark, представленная на рисунке 2.

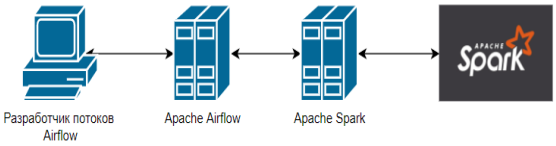

Рисунок 2: Рассматриваемая архитектура взаимодействия серверов

При запуске Spark задач на кластере, как правило, указывается достаточно большое количество Spark-параметров. Причём эти параметры, условно, можно разделить на несколько блоков:

- параметры исполнителей (экзекьюторов – англ. executor) задач на кластере;
- параметры драйвера – JVM процесса, отвечающего за координацию взаимодействия между исполнителями;
- параметры, связанные с загрузкой сторонних зависимостей во время исполнения потока.
- остальные параметры.

При запуске Spark-задачи необходимо извлечь конфигурационные параметры из файла, при этом для наглядности внутри файла параметры могут быть разбиты согласно классификации, представленной выше.

Помимо извлечения конфигурации и автоизации на Сервере, имеющем доступ к Apache Spark, разрабатываемый оператор должен поддерживать возможность сохранения обработанных данных в файл с целью дальнейшей обработки этого файла на Сервере 1.

Таким образом, разрабатываемый Spark-оператор должен реализовывать следующие функции:

- авторизация на Сервере 2 по учётным данным сотрудника;
- возможность сохранения полученных в результате выполнения Spark-задачи наборов данных в файл, который будет сохранён на Сервере с Airflow;
- извлечение параметров Spark из конфигурационного YAML-файла.

IV. ОСОБЕННОСТИ РЕАЛИЗАЦИИ SPARK-ОПЕРАТОРА

Метод __init__ разрабатываемого Spark-оператора принимает следующие аргументы:

- name – имя, которое будет отображаться в YARN в процессе выполнения Spark-задачи;
- run_point –код, исполняемый в рамках Spark-задачи (либо Scala/Java класс, либо программа, написанная на PySpark);
- run_args – аргументы исполняемой Spark-задачи;

- conn_id – идентификатор соединения к серверу, на котором будет исполнена задача (более подробно этот параметр будет описан в следующей главе);
- repos – список URL репозиториев для выгрузки сторонних зависимостей;
- fat_jar – JAR, который может быть загружен из репозиториев для выполнения Spark-задачи;
- py_deps – Python зависимости, которые могут быть задгружены из репозиториев;
- push-files – файлы, которые могут быть загружены в Classpath JVM-процесса, выполняющего Spark-задачу;
- deploy_type – способ запуска Spark-задачи. По умолчанию – client mode;
- conf_file – файл, из которого считываются параметры запускаемой Spark-задачи;
- env_dict – словарь с переменными окружения;
- timeout – время ожидания ответа сервера при выполнении команды после установления ssh-соединения;
- keepalive_interval – период отправки запросов исполнителей сообщений для проверки их работоспособности.

Параметр conf_file содержит путь до файла в формате YAML, в котором содержатся настройки конфигурации исполняемой Spark-задачи. В рамках оператора предлагается разбить параметры Spark-задачи на несколько блоков, среди которых следует отметить количество памяти и количество ядер, выделяемых на исполнителей и драйвер-процесс Spark.

В рамках Spark-оператора предлагается также возможность сохранения файла с данными, сформированными в процессе выполнения Spark-задачи на сервер с Apache Airflow. Для этого необходимо сохранить файл в определённую директорию на сервере, на котором исполняется Spark-задача. После этого, на заключительном этапе выполнения Spark-оператора, происходит рекурсивное копирование всех файлов с сервера на сервер.

V. МЕТОД АУТЕНТИФИКАЦИИ НА УДАЛЁННОМ СЕРВЕРЕ

Основная задача разрабатываемого метода аутентификации – предоставление доступа к серверу с Apache Spark по персональным учётным данным (логин, пароль). Для решения задачи необходимо иметь хранилище секретной информации. В рамках настоящей работы предлагается использовать программное обеспечение HashiCorp Vault.

Vault может быть использован для решения следующих задач:

- хранилище секретной информации;
- генерация ключей API для исполняемых программ;
- шифрование данных[12].

Аутентифкации в HashiCorp Vault реализуется с помощью токенов доступа. В рамках Vault все токены однозначно соотносятся с какой-либо информацией. Самая важная информация, связанная с токеном – множество из одной или более политик Vault. Политики определяют, к какой информации и какие права имеет владелец токена. Также с токеном связана служебная информация: время создания, дата последнего обновления и др. Важной особенностью Vault является возможность предоставления функционала через REST API, при этом для большинства популярных языков программирования существуют специальные библиотеки для упрощения работы с API[13].

Для реализации метода аутентификации предлагается разработать веб-сервис, содержащий, как минимум, следующий функционал:

1. Предоставление администраторам Airflow возможности заведения нового токена Vault для доступа к необходимым секретам. Эта часть может быть представлена в виде веб-страницы с формой ввода, основными полями которой будет название потока обработки данных, время жизни токена, список секретов, доступ к которым должен предоставляться по выпускаемому токену. При этом каждый секрет должен иметь уникальный идентификатор. Именно этот идентификатор указывается в аргументе conf_file при инициализации Spark-оператора (см. пред. главу).
2. После создания токена на стороне сервиса должна быть реализована строгая привязка названия потока обработки данных Airflow к списку доступных секретов.

Для реализации нового потока обработки данных разработчик должен предварительно сообщить администратору Airflow название потока и список секретов, необходимых для реализации. Затем администратор с помощью веб-сервиса, о котором говорилось ранее, выполняет регистрацию нового потока со списком доступных этому потоку секретов. Далее, в процессе выполнения Spark-оператора выполняется следующая последовательность действий:

1. Выполняется HTTP-запрос к разработанному ранее веб-сервису на REST API маршрут, по которому содержится информация обо всех доступных потоку секретах. На стороне

веб-сервиса проверяется, владеет ли поток обработки данных правами на чтение запрашиваемого секрета. Запрос также содержит заголовок X-Bender-Token, значением которого является токен из переменной окружения Airflow. Если токен некорректен, то пользователь получает информацию об ошибке 403. Подобная проверка необходима для невозможности запросов к REST API маршрутам без знания токена, т.е. не из окружения Airflow. Если представленный пользователем токен корректный, а также поток обработки данных имеет право на чтение запрашиваемого секрета, то клиент получает от веб-сервиса маршрут, по которому необходимо выполнить запрос к REST API Vault для чтения секрета.

2. Выполняется HTTP-запрос к разработанному ранее веб-сервису на REST API маршрут, по которому содержатся токены, предоставляемые потокам Airflow. При запросе также отправляется заголовок X-Bender-Token для возможности выполнения запросов только из окружения Airflow. На стороне веб-сервиса выполняется проверка корректности токена X-Bender-Token и возможности чтения конкретного секрета потоком Airflow. Если проверка завершается успешно, то клиент (Spark-оператор) получает токен Vault для запроса к Vault API.

3. Выполняется HTTP запрос к Vault API. При этом конкретный маршрут HTTP запроса был получен в пункте 1. При обработке запроса на стороне Vault выполняется проверка наличия заголовка X-Vault-Token. Токен для заполнения значения этого заголовка был получен на шаге 2. Если проверка токена в Vault завершается успешно, то пользователь получает значение секрета (логин и пароль для авторизации на сервере с доступом к Spark).

4. В рамках Spark-оператора устанавливается SSH-соединение с помощью учётных данных, полученных на шаге 3, с сервером для запуска команды *spark-submit*.

Предложенный алгоритм аутентификации представлен на рисунке 3.

Недостатком предложенного метода аутентификации является компрометация учётных данных при компрометации переменных окружения Apache Airflow, однако хранение секретных данных в переменных окружения является распространённой практикой, поскольку для их компрометации злоумышленник должен иметь доступ к системе.

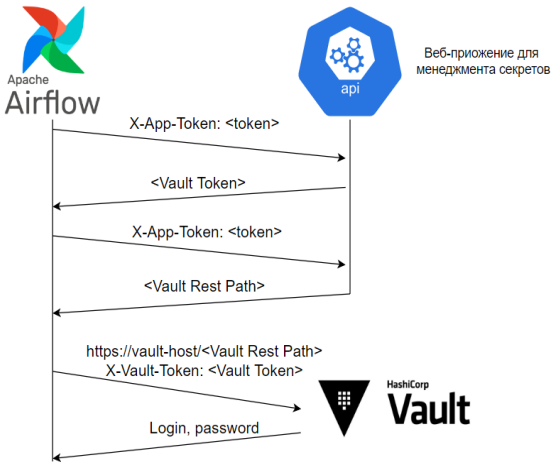

Рисунок 3: Алгоритм аутентификации на удалённом сервере

VI. ДИСКУССИЯ ПО ПРЕДЛОЖЕННОМУ АЛГОРИТМУ АУТЕНТИФКАЦИИ

Преимуществами предложенного метода аутентификации являются:

1. Контроль администраторами Airflow разрабатываемых потоков обработки данных. Контроль осуществляется за счёт регистрации названия потока на первоначальном этапе разработки, за счёт чего гарантируется, что поток имеет доступ только к определённым секретам.

2. Возможность использования нескольких экземпляров (*англ.* - instances) Airflow в рамках одной системы. В случае использования стандартных менеджеров учётных данных Airflow необходимо самостоятельно вносить информацию о секретах в каждый экземпляр Airflow, однако при использовании подобного механизма аутентификации все секреты содержатся в едином Vault. Это может быть удобно, например, при наличии Airflow на тестовом стенде.

3. В случае наличия политики паролей в организации, согласно которой пользователи периодически должны менять пароль, возможно расширение веб-сервиса страницей по смене пароля пользователя в Vault. В случае использования стандартного хранилища секретов Vault администраторы должны самостоятельно обновлять учётные данные в Airflow, поскольку отсутствует Airflow API для решения этой задачи.

4. Возможность журналирования всех доступов к секретам с помощью Vault.

Основным недостатком предложенного алгоритма является возможность компрометации переменной окружения Airflow и последующая компрометация секретов, используемых для аутентификации, однако использование

подобного недостатка злоумышленником возможно лишь в случае получения им доступа к системе[14].

Вторым недостатком предложенного метода аутентификации является необходимость разработки веб-сервиса для менеджмента токенов и секретов Vault, однако подобный сервис может быть расширен функционалом по смене пароля, что решит проблему необходимости периодической смены паролей.

VII. ЗАКЛЮЧЕНИЕ

В работе рассмотрена возможная реализация Spark-оператора Apache Airflow для запуска Spark-задач на удалённом сервере, при этом для оператора предложен метод аутентификации, для реализации которого необходимо хранилище секретов HashiCorp Vault. Отмечены достоинства и недостатки предложенного метода аутентифкации. В работе рассмотрены основные особенности функционирования Airflow, Spark и Vault, сформированы требования к Spark-оператору. В результате работы получен проект программного комплекса, позволяющего выполнять запуск Spark-задач в Airflow, проект веб-сервиса для менеджмента секретов, используемых для авторизации на удалённом сервере, а также для периодической смены паролей пользователей в Vault.

БЛАГОДАРНОСТИ

Авторы выражают благодарность Высшей инжиниринговой школе НИЯУ МИФИ за помощь в возможности опубликовать результаты выполненной работы.

БИБЛИОГРАФИЯ

[1] CBR [электронный ресурс] // Количество платежных карт, эмитированных кредитными организациями и Банком России, по типам карт [сайт] URL: https://old.cbr.ru/statistics/psrf/sheet013/ (дата обращения: 01.05.2022).

[2] Tadviser [online resource] //Bank card fraud [website] URL: https://www.tadviser.ru/index.php//index.php/Статья:Мошенничество_с_банковскими_картами (Дата обращения 01.05.2022)

[3] FINVERSIA [электронный ресурс] // Big-data банков и телекомов: кто и как внедряет большие данные [сайт] URL: https://www.finversia.ru/publication/big-data-bankov-i-telekomov-kto-i-kak-vnedryaet-bolshie-dannye-45951 (Дата обращения 02.05.2022)

[4] Tableau [online resource] // Identity Theft Reports - Federal Trade Commission [сайт]: https://public.tableau.com/profile/federal.trade.commission#!/vizhome/IdentityTheftReports/TheftTypesOverTime (дата обращения: 05.05.2022).

[5] Проект ГОСТ Р Информационные технологии. Большие данные. Обзор и словарь. Информационные технологии. Большие данные. – М., Стандартинформ, 2021. – 16 с.

[6] Apache [online resource] // Apache Airflow Documentation [website] URL: https://airflow.apache.org/docs/apache-airflow/2.2.5/ (дата обращения: 10.05.2022).

[7] Kotliar M., Kartashov A. V., Barski A. CWL-Airflow: a lightweight pipeline manager supporting Common Workflow Language //Gigascience. – 2019. – Т. 8. – №. 7. – С. giz084.

[8] Rubio D. Jinja templates in Django //Beginning Django. – Apress, Berkeley, CA, 2017. – С. 117-161.

[9] Spark [online resource] // Spark Overview [website] https://spark.apache.org/docs/2.4.0/ (дата обращения: 15.05.2022).

[10] Salloum S. et al. Big data analytics on Apache Spark //International Journal of Data Science and Analytics. – 2016. – Т. 1. – №. 3. – С. 145-164.

[11] Esmaeilzadeh A. et al. Efficient large scale nlp feature engineering with apache spark //2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC). – IEEE, 2022. – С. 0274-0280.

[12] HASHICORP [online resource] // Documentation [website] https://www.vaultproject.io/docs (дата обращения: 18.05.2022).

[13] Sabharwal N., Pandey S., Pandey P. Getting Started with Vault //Infrastructure-as-Code Automation Using Terraform, Packer, Vault, Nomad and Consul. – Apress, Berkeley, CA, 2021. – С. 131-150.

[14] Seh A. H. et al. Hybrid computational modeling for web application security assessment //CMC-Comput., Mater. Continua. – 2022. – Т. 70. – №. 1. – С. 46

Development of an authentication method for running Apache Spark tasks in the Apache Airflow

P.A. Golovnyak, K.S. Zaytsev, A.M. Pinchuk

Abstract - **Today there is a significant increase in the number of payments made using plastic cards. Banking organizations are forced to analyze huge amounts of data in their work, and therefore software products for processing big data are being put into operation. Apache Spark is a popular tool for streaming and batch data processing. To analyze information, software is often used to run processes on a schedule. An example of such software is Apache Airflow. It is customary to represent the processes launched in Airflow in the form of a directed graph without cycles, while the vertices of the graph are tasks that are the implementation of a certain Airflow operator. The purpose of the article is to develop an Apache Airflow operator to run Spark tasks on a server other than the one on which Airflow operates. When developing such an operator, the task of authentication on a remote server arises. The article proposes a solution to the authentication problem by using access tokens provided by the Vault secret repository, as well as a web service of its own development, with the help of which the secrets and tokens stored in the Vault are managed. As a result of the work, a solution was obtained that allows you to run Spark tasks in Airflow on a remote server, while the secrets used for authentication are stored in Vault, which increases the security of the system.**

Keywords — **Apache Airflow, Apache Spark, HashiCorp Vault, Spark-operator, authentication, access token**

REFERENCES

[1] CBR [online resource] // Number of payment cards issued by credit institutions and the Bank of Russia, by card type [website] URL: https://old.cbr.ru/statistics/psrf/sheet013/ (Date of request: 01.05.2022).

[2] Tadviser [online resource] //Bank card fraud [website] URL: https://www.tadviser.ru/index.php//index.php/Статья:Мошенничество_с_банковскими_картами (Date of request 01.05.2022)

[3] FINVERSIA [online resource] // Big-data of banks and telekcoms: who and how inject big fata [website] URL: https://www.finversia.ru/publication/big-data-bankov-i-telekomov-kto-i-kak-vnedryaet-bolshie-dannye-45951 (Date of request 02.05.2022)

[4] Tableau [online resource] // Identity Theft Reports - Federal Trade Commission [сайт]: https://public.tableau.com/profile/federal.trade.commission#!/vizhome/IdentityTheftReports/TheftTypesOverTime (Date of request 05.05.2022).

[5] Project GOST R Information technology. Big data. Overview and dictionary. Information technology. Big data. – M., Standartinform, 2021. – 16 c.

[6] Apache [online resource] // Apache Airflow Documentation [website] URL: https://airflow.apache.org/docs/apache-airflow/2.2.5/ (Date of request 10.05.2022).

[7] Kotliar M., Kartashov A. V., Barski A. CWL-Airflow: a lightweight pipeline manager supporting Common Workflow Language //Gigascience. – 2019. – Т. 8. – №. 7. – С. giz084.

[8] Rubio D. Jinja templates in Django //Beginning Django. – Apress, Berkeley, CA, 2017. – С. 117-161.

[9] Spark [online resource] // Spark Overview [website] https://spark.apache.org/docs/2.4.0/ (Date of request: 15.05.2022).

[10] Salloum S. et al. Big data analytics on Apache Spark //International Journal of Data Science and Analytics. – 2016. – Т. 1. – №. 3. – С. 145-164.

[11] Esmaeilzadeh A. et al. Efficient large scale nlp feature engineering with apache spark //2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC). – IEEE, 2022. – С. 0274-0280.

[12] HASHICORP [online resource] // Documentation [website] https://www.vaultproject.io/docs (Date of request: 18.05.2022).

[13] Sabharwal N., Pandey S., Pandey P. Getting Started with Vault //Infrastructure-as-Code Automation Using Terraform, Packer, Vault,

Nomad and Consul. – Apress, Berkeley, CA, 2021. – С. 131-150.

[14] Seh A. H. et al. Hybrid computational modeling for web application security assessment //CMC-Comput., Mater. Continua. – 2022. – Т. 70. – №. 1. – С. 469-489.