Программный комплекс механизма управления доступом на основе риск-ориентированной атрибутивной модели

А.В. Шитов, Н.Е. Стельмах, Ш. Г. Магомедов

***Аннотация*—Целью статьи является разработка программного комплекса реализации механизма управления доступом на основе атрибутивной модели на основе риска, динамически обрабатывающей запросы от целевых сервисов, а также разработка интеграции в уже существующие системы для показания гибкости решения.**

В статье проводится анализ моделей управления доступом, в частности, на основе риска. Обсуждаются преимущества динамических моделей управления доступом. Отдельное внимание уделяется риск-ориентированной модели на основе нечеткой логики, рассматриваются ее отличительные особенности и объясняется ее выбор в качестве основы для разрабатываемого механизма управления доступом.

Представлена программная реализация механизма управления доступом на основе риск-ориентированной, разработанная на базе библиотеки Py-ABAC и написанная на языке программирования Python. Создана интеграция в систему электронного обучения и тестирования Moodle, протестированная на уровне предпрод стенда.

Внимание уделяется деталям, связанным с использованием риск-ориентированной модели управления доступом на основе нечеткой логики, ее преимуществам, предлагается вариация программной реализации механизма управления доступом на основе данной модели.

***Ключевые слова*— Информационная безопасность, модели управления доступом, Python, Py-ABAC, риск-ориентированная модель управления доступом, нечеткая логика, moodle**

I. Введение

В современном мире все больше организаций задаются вопросом обеспечения информационной безопасности, особенно на фоне геополитических событий последних лет, которые привели к кратному росту кибератак [1].

Одним из важнейших направлений в обеспечении информационной безопасности является создание политик информационной безопасности и их применение. Политика позволяет, основываясь на ситуации, применять правовые, организационные или технические меры по обеспечению информационной безопасности той системы, для которой она была разработана.

Одной из ключевых политик является политика управления доступом, которая, согласно [2] является совокупностью правил, подлежащих реализации средством защиты информации и регламентирующих предоставление доступа между компонентами среды функционирования этого средства защиты информации.

Традиционные модели управления доступом, такие как DAC, MAC, RBAC [3] основываются на статичном доступе к ресурсам, который описывается набором некоторых правил, зависящих от конкретной модели. Данный подход обладает рядом преимуществ, таких как:

- относительная простота реализации;
- удобство администрирования;
- надежность;
- обеспечение необходимого уровня защищенности от уже известных угроз.

Однако, последнее указанное преимущество является спорным, поскольку с развитием технологий совершенствуются и злоумышленники, соответственно, количество и уровень угроз растет в режиме реального времени. Однако, набор правил, согласно которым функционируют модели статичен. Это говорит о том, что данные модели не смогут качественно обеспечить необходимую степень защищенности в контексте ранее неопределенных угроз информационной безопасности.

В последние годы все чаще стали фиксировать инциденты информационной безопасности, произошедшие из-за утечки данных. Виновниками при этом являются внутренние нарушители, причем как в частных, так и государственных учреждениях. [4]. С ростом количества пользователей и сложности структуризации информации внутри системы появляется проблема грамотного распределения доступа, с которой не справляются традиционные модели управления доступом.

Динамические модели управления доступом позволяют решить эту проблему, используя не только заранее зафиксированный набор правил, но и некоторые параметры, которые являются дополнительным звеном в процессе принятия решения. Примером данного параметра является риск.

Статья получена 08 мая 2024.

Шитов Артём Валерьевич, РТУ МИРЭА (e-mail: Bulbazavr41@yandex.ru, shitov.a.v2@edu.mirea.ru).

Стельмах Никита Евгеньевич, РТУ МИРЭА (e-mail: stelmach.stels@yandex.ru, stelmakh.n.e@edu.mirea.ru).

Шамиль Гасангусейнович Магомедов, РТУ МИРЭА (e-mail: : magomedov_sh@mirea.ru).

II. Обзор решений в области управления доступом на основе риск-ориентированных динамических моделей

Динамические модели на основе риска могут адаптироваться к изменяющимся рискам и угрозам и стремятся к тонкому уровню гранулярности, что обеспечивает более эффективное управление доступом и минимизацию риска. Был проведен анализ моделей управления доступом на основе риска.

Модель Risk Adaptable Access Control (RAdAC) была предложена Макгроу [5]. Она основана на оценке риска безопасности и оперативных потребностей для предоставления или отказа в доступе. Эта модель оценивает риск, связанный с каждым запросом на доступ, затем сравнивает его с политикой управления доступом. После этого система проверяет операционные потребности; если связанные операционные потребности и политика удовлетворены, доступ предоставляется. Однако автор не предоставил подробностей о том, как количественно оценить риск и операционные потребности. Кроме того, Кандала и другие [6] предложили подход, который определяет различные компоненты риска в модели RAdAC, используя подход управления доступом на основе атрибутов.

Динамическая и гибкая модель управления доступом на основе рисков была предложена Диепом и другими [7]. Эта модель использует оценку риска для определения величины риска в зависимости от результатов действий с точки зрения доступности, конфиденциальности и целостности.

Один из способов выразить модель ценности — это отношение между элементами. Другой способ - присвоить каждому элементу числовые значения. Это и есть числовое представление. В их работе они используют более поздний метод, чтобы объединить контекст и значение риска.

Существует множество факторов, которые влияют на процесс оценки риска. Для каждого действия значение риска зависит от результатов. Если стоимость результата (из-за действия) высока, то и риск высок. Риск также зависит от текущих параметров контекста. Например, в условиях низкой скорости интернет-соединения легко потерять сессию ftp-соединения. Это означает, что мы теряем доступность. Или если у нас беспроводное соединение, нас легко взломать.

Свойство ресурсов в действии также имеет важную роль в оценке риска. Но риск, который оно зависит от вида действия и контекста результата. Если предположить, что риск, создаваемый таким действия, такого как удаление большого видеофайла, меньше, чем риск копирования большого видеофайла с точки зрения потери доступности.

Однако в этой модели не было однозначного стандарта того, а были приведены абстрактные примеры, как оценивать величину риска для каждого состояния среды и для каждого результата действия, не использовался пользовательский контекст, и отсутствовали функции адаптации к риску.

Раджбхандари и Снеккенес [8] предложили подход, основанный на анализе рисков, для динамического принятия решений о доступе. Этот подход основан на предпочтениях или значениях выгоды, которые могут предоставить субъекты, а не на субъективной вероятности, использующей теорию игр. Простой сценарий конфиденциальности между пользователем и онлайновым книжным магазином представлен для того, чтобы дать первоначальное представление о концепции. Однако использование только выгод субъекта для определения решения о доступе недостаточно для разработки гибкой и масштабируемой модели управления доступом. Кроме того, в ней отсутствуют функции адаптации к риску.

Шарма и др. [9] предложили модель управления доступом на основе задач для оценки величины риска с помощью функций, основанных на действиях, которые пользователь хочет выполнить. Значение риска вычисляется в терминах различных действий и соответствующих результатов. Исходы и вероятность риска определяются вместе с уровнем чувствительности данных. Для оценки общей величины риска используются модели поведения предыдущих пользователей. Оцененное значение риска сравнивается с пороговым значением риска для принятия решения о доступе. Однако в ней отсутствуют функции адаптации к риску.

Модель управления доступом на основе контекстного риска была предложена Ли и др [10]. Модель собирает всю полезную информацию из окружающей среды и оценивает ее с точки зрения безопасности. Для оценки риска применяется метод многофакторного процесса оценки (MFEP). Значение риска основано на результатах действий с точки зрения доступности, конфиденциальности и целостности. Эта модель была оценена для управления контролем доступа в больнице. Однако в этой модели не учитывается поведение пользователей в прошлом, а также адаптация к риску.

Основная проблема, связанная с использованием риск-ориентированной модели, заключается в обеспечении оперативного, надежного и точного метода оценки риска, особенно в условиях отсутствия данных для количественного описания риска и оценки его воздействия [11]. Для количественной оценки рисков предлагается использовать подход на основе нечетких множеств [12].

III. Риск-ориентированная модель управления доступом на основе нечеткой логики

В отличие от прочих динамических моделей на основе риска, модель на основе нечеткой логики не требует жертвовать доступностью услуг ради безопасности при этом минимизируя расходуемые ресурсы системы.

Риск-ориентированная модель управления доступом на основе нечеткой логики является динамической моделью, которая функционирует в режиме реального времени и использует контекстную информацию для принятия решений о доступе к объекту. Эта модель выполняет расчет риска по каждому запросу на доступ к объекту и принимает решение динамически на основе полученного значения риска.

Предлагаемая модель позволит перейти от статических правил [13] к динамическим и при этом описываемым

атрибутивной моделью управления доступом. Риск в данном контексте выступает дополнительным атрибутом управления доступом, участвующем в процессе принятия решения о предоставлении доступа.

Основные компоненты модели:

- Агенты риска собирают информацию о различных факторах риска, таких как история доступа, тип устройства, местоположение, время доступа и другие.
- Модуль оценки риска использует нечеткую логику для оценки уровня риска на основе информации, полученной
- Модуль принятия решений использует результаты оценки риска для принятия решений о доступе к ресурсам.
- База данных политик содержит правила и политики доступа, которые используются для принятия решений о доступе.

Агенты риска собирают информацию о различных факторах риска. Далее модуль оценки риска использует нечеткую логику для оценки уровня риска на основе информации, полученной от сенсоров риска. Модуль оценки риска определяет уровень риска как низкий, средний или высокий. Модуль принятия решений использует результаты оценки риска и политики доступа для принятия решений о доступе к ресурсам.

Выдача доступа производится в соответствии с заданными политиками доступа.

В результате, риск-ориентированная модель на основе нечеткой логики позволяет найти баланс между безопасностью и доступностью услуг.

IV. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ МЕХАНИЗМА УПРАВЛЕНИЯ ДОСТУПОМ, РЕАЛИЗУЮЩЕГО МОДЕЛЬ

Проведя анализ существующих решений было выявлено, что на поприще программной реализации риск-ориентированных моделей управления доступом не представлено научных работ, описывающих реализацию подобных механизмов управления доступом. Близкой к теме данной работы является указанный в [14] фреймворк атрибутного управления доступом на основе стандарта XACML. Необходимость практической реализации обусловлена потребностью в универсальном решении в вопросе простого и удобного создания механизма управления доступом. Перед разработкой были выдвинуты функциональные и нефункциональные требования.

A. Функциональные требования

Аутентификация и авторизация:

- Реализация возможности регистрации и аутентификации пользователей.
- Гибкая система управления доступом на основе ролей пользователей.

Управление данными:

- Функционал CRUD (создание, чтение, обновление, удаление) данных в базе MongoDB.
- Применение политик риск-ориентированной модели управления доступом для безопасного доступа к данным.

Взаимодействие с файлами:

- Возможность загрузки и хранения файлов в базе данных MongoDB.
- Возможность редактирования хранящихся внутри MongoDB файлов

B. Нефункциональные требования

Безопасность:

- Гарантированная защита данных с использованием механизмов шифрования и аутентификации.
- Соблюдение стандартов безопасности при работе с чувствительной информацией.

Производительность:

- Высокая скорость обработки запросов и отзывчивость системы.
- Оптимизированные запросы к базе данных для уменьшения нагрузки.

Масштабируемость:

- Возможность горизонтального масштабирования для обработки роста нагрузки.
- Эффективное управление ресурсами для обеспечения стабильной работы при росте пользовательской базы.

Надежность:

- Гарантированная доступность приложения и целостность данных.
- Резервное копирование данных и мониторинг работоспособности системы.

Интеграция: Возможность интеграции приложения в уже существующие системы

C. Используемые технологии

Приложение реализовано на базе Flask, в качестве хранилища данных, к которому применяются политики управления доступом на основе Py-ABAC используется MongoDB. [15]

Рисунок 1. Схема работы приложения

Flask RESTful API

Выбор Flask в качестве фреймворка для создания API обосновывается его легкостью в изучении и использовании, а также его гибкостью и расширяемостью. Flask, как микрофреймворк, обладает минималистичной структурой, что увеличивает скорость развертывания приложения. Также Flask обладает возможностью простой и удобной интеграции сторонних библиотек.

Py-ABAC

Использование атрибутивной модели позволяет обеспечить гибкий подход к управлению доступом. В программной реализации риск-ориентированной модели используется Py-ABAC, конструкция которого основана на стандарте XACML [16] и пакете SDK Vakt [17] для Python ABAC. Данная библиотека отличается от

родительского стандарта рядом особенностей, которые позволили сделать выбор в ее пользу:

- в отличии от используемого в XACML собственного стандарта политик и запросов в Py-ABAC применяется формат Json, что упрощает разработку и позволяет удобнее хранить и обрабатывать данные;
- в то время как XACML является стандартом и представляет собой язык разметки, что позволяет реализовывать XACML-запросы на различных языках программирования, Py-ABAC может быть реализован только как Python приложение, что позволяет легко интегрировать его в Flask;
- в Py-ABAC политика реализует одно правило, которое может реализовывать либо разрешение на операции в соответствии с атрибутами, либо запрет. В XACML же одна политика реализует множество правил со своими атрибутами, что является избыточным в рамках данного проекта.

Рисунок 2. Архитектура ABAC

На приведенной выше диаграмме изображена стандартная архитектура ABAC, которая выглядит следующим образом:

- PEP или точка применения политики: фрагмент кода, который использует Py-ABAC для защиты приложений и данных. PEP должен проверить запрос пользователя, создать соответствующий запрос на доступ и отправить его PDP для оценки.
- PDP или точка принятия решения: часть, которая оценивает входящие запросы доступа на соответствие политикам и возвращает решение «Разрешить/Запретить». PDP может также использовать PIP для получения недостающих значений атрибутов во время оценки политики.
- PIP или точка информации о политике: соединяет PDP с внешними источниками значений атрибутов, например, LDAP или базами данных.
- PAP или точка администрирования политики: управляет созданием, обновлением и удалением политик, оцениваемых PDP.

В приведенной выше архитектуре во время запроса доступа к PDP участвуют следующие четыре элемента:

- **субъект:** это объект, который запрашивает доступ, также известный как субъект запроса. Субъектом может быть все, что запрашивает доступ, например, пользователь или приложение.
- **ресурс:** объект, доступ к которому запрашивается субъектом.
- **действие:** действие, выполняемое над ресурсом.
- **контекст:** этот элемент касается времени, местоположения или динамических аспектов сценария контроля доступа.

В Py-ABAC определяются политики, содержащие условия для одного или нескольких атрибутов этих четырех элементов. Если эти условия удовлетворены, решение о доступе возвращается PDP с использованием алгоритма оценки. Поддерживаются три различных алгоритма оценки:

- **AllowOverrides:** возвращается allow если какое-либо решение оценивается как allow; и возвращает значение deny, если все решения оцениваются как deny.
- **DenyOverrides:** возвращается deny если какое-либо решение оценивается как deny; возвращается allow, если все решения оцениваются как allow.
- **HighestPriority:** возвращает решение с наивысшим приоритетом, которое оценивается как allow или deny. Если существует несколько конфликтующих решений с одинаковым наивысшим приоритетом, то DenyOverrides алгоритм будет применяться среди этих решений с наивысшим приоритетом.

Политика состоит из uid, description, rules, targets, effect и priority полей. Схема JSON задается:

```
{
   "uid": <string>,
   "description": <string>,
   "rules": <rules_block>,
   "targets": <targets_block>,
   "effect": <string>,
   "priority": <number>
}
```

где <rules_block> и <targets_block> — блоки JSON. По сути, поля "targets" и "rules" используются для определения условий атрибутов элементов управления доступом. Если эти условия удовлетворены, применяется политика и значение поля "effect" возвращается методом PDP. Таким образом, "effect" это возвращаемое решение политики, которое может быть либо "allow" или "deny". Поле "uid" представляет собой строковое значение, которое однозначно идентифицирует политику. Как следует из названия, в этом "description" поле хранится описание политики, "priority" предоставляет числовое значение, указывающее вес политики, когда ее решение конфликтует с другой политикой в рамках HighestPriority алгоритма оценки. По умолчанию это поле установлено 0 для всех политик.

Ключ определяет атрибут в нотации ObjectPath, а значение является условным выражением. Использование нотации ObjectPath дает Py-ABAC мощную возможность определять условия для вложенных атрибутов. Это <condition_expression> снова

блок JSON, определяющий требования, которым должно соответствовать значение атрибута. В качестве примера ниже показан блок условий для требования, чтобы податрибут «firstName» для «name» в элементе управления доступом субъекта был «Max»:

```
{
   "subject": {
      "$.name.firstName": {
         "condition": "Eq",
         "value": "Max"
      }
```

Иногда условий для одного атрибута недостаточно, и требуются ограничения для нескольких атрибутов, связанных логическими отношениями, такими, как И или ИЛИ. В Py-ABAC это достигается за счет использования встроенных структур данных JSON объектов и массивов в качестве неявных логических операторов. Объект неявно представляет собой оператор AND, который будет иметь значение true, только если все включенные пары ключ-значение будут оценены как true. Аналогично, массив неявно является оператором ИЛИ, который будет иметь значение true, если хотя бы один из его членов будет оценен как true.

PDP – этот компонент является точкой принятия решения о политике, экземпляр которой создается через PDP класс. Это основная точка входа Py-ABAC для оценки политики. Как минимум, для создания объекта требуется объект Storage PDP. У него есть один метод, is_allowed который при передаче AccessRequest объекта дает вам логический ответ: разрешен доступ или нет

Хранилище — это компонент, который обеспечивает интерфейс для реализации сохраняемости политики. Таким образом, он используется при создании PAP с открытыми методами. Он также используется PDP для получения политик для оценки.

D. Производительность

В Py-ABAC для определения условий атрибутов во время создания политики используется концепция “targets” и “rules”, заимствованная из стандарта XACML. С концептуальной точки зрения, "targets" указывает, к каким экземплярам элементов управления доступом применяется политика. Так называемые цели политики. "Rules", с другой стороны, определяют условия для атрибутов целей.

Благодаря использованию “targets” и “rules” при грамотной настройке достигается высокая производительность так как даже если в базе данных хранятся тысячи политик, для проверки определенной транзакции используется только несколько политик, зависящих от конкретной цели.

Масштабируемость

Данная реализация позволяет хранить неограниченное количество политик. Это достигается за счет того, политики находятся не в памяти, а в базе данных. СУБД при этом так же можно использовать любую, отличие будет заключаться в подходе к взаимодействию. При этом стоит отметить, что “бутылочным горлышком” может быть вызов слишком большого количества политик для обработки одной транзакции - в данном случае процесс принятия решения займет кратно больше времени.

E. Перспективы развития

В рамках данной работы предлагаются следующие перспективы развития проекта:

- создание универсального обработчика для любой выбранной базы данных. Py-ABAC позволяет написать свой собственный класс для взаимодействия с базой данных;
- разработка графического и/или веб интерфейса для удобного создания политик. Подразумевается не только создание/удаление, но и редактирование, осуществление мониторинга имеющихся политик;
- разработка инструмента автоматизированного тестирования с графическим и/или веб интерфейсом для проверки корректности создания политик и выявления конфликтов между политиками.

V. Пример работы приложения

Работа приложения разделяется на следующие этапы:

Пользовательский интерфейс: пользователь открывает веб-приложение в браузере и видит форму для ввода данных.

Отправка данных: пользователь вводит необходимую информацию и нажимает кнопку "Отправить". Далее JavaScript отправляет асинхронный запрос на сервер Flask через AJAX для обработки введенных данных.

Обработка на сервере: Flask принимает запрос и обрабатывает данные, которые конвертируются в JSON который будет обработан политикой.

Пример запроса:

```
{"subject":
{"id": "",
"attributes":
{"role": "Teacher",
"device_type": "Personal Laptop",
"connection_type": "VPN"}},
"resource":
{"id": "",
"attributes":
{"service": "Science"}},
"action":
{"id": "",
        "attributes": {
"method": "Write"}},
"context":
{"risk": "Low"}}
```

запрос пользователя сохраняется в логах с дополнительной меткой времени для последующего доступа и анализа.

Управление доступом: сервер применяет политики управления доступом на основе ABAC, определяющие, имеет ли пользователь право на доступ к этим данным.

```
{
    "uid": "9",
    "description": "Risk-Adaptive: Deny-overrides & Deny Biased"
                   "Deny to Delete on High risk",
    "effect": "deny",
    "rules": {
        "subject": {
            "$.role": {
                "condition": "AnyOf",
                "values": [{"condition": "Equals", "value": "editingteacher"},
                           {"condition": "Equals", "value": "teacher"},
                           {"condition": "Equals", "value": "student"}]
            },
            "$.device_type": {
                "condition": "Equals",
                "value": "Personal Laptop"
            },
            "$.connection_type": {
                "condition": "Equals",
                "value": "VPN"
            },
        },
        "resource": {"$.service": {"condition": "Equals", "value": "Science"}},
        "action": {"$.method": {
            "condition": "Equals",
            "value": "Delete"}},
        "context": {"$.risk": {"condition": "Equals", "value": "High"}}
    },
    "targets": {},
    "priority": 0
},
```


Рисунок 3. Пример политики для высокого риска

```
{
    "uid": "5",
    "description": "Risk-Adaptive: Deny-overrides & Deny Biased"
                   "Allow to read, write, del on Low risk",
    "effect": "allow",
    "rules": {
        "subject": {
            "$.role": {
                "condition": "AnyOf",
                "values": [{"condition": "Equals", "value": "editingteacher"},
                           {"condition": "Equals", "value": "teacher"},
                           {"condition": "Equals", "value": "student"}]
            },
            "$.device_type": {
                "condition": "Equals",
                "value": "Personal Laptop"
            },
            "$.connection_type": {
                "condition": "Equals",
                "value": "VPN"
            },
        },
        "resource": {"$.service": {"condition": "Equals", "value": "Science"}},
        "action": {"$.method": {"condition": "AnyOf",
                                "values": [{"condition": "Equals", "value": "Read"},
                                           {"condition": "Equals", "value": "Write"},
                                           {"condition": "Equals", "value": "Delete"}]}},
        "context": {"$.risk": {"condition": "Equals", "value": "Low"}}
    },
    "targets": {},
    "priority": 0
},
```


Рисунок 4. Пример политики для низкого риска

Для примера выделим две политики:

- политика для высокого риска, которая явно запрещает удаление при высоком риске
- политика для низкого риска, которая разрешает основные операции в базе данных на низком риске.

В зависимости от текущего риска системы будет применена одна из политик. Особенность риск-ориентированной модели в том, что показатель риска используется как атрибут для политики и является динамическим, то бишь рассчитывается в реальном времени. Таким образом, при грамотной настройке политик можно запрещать или разрешать определенные действия в зависимости от текущего риска. Что позволит в случае выявления аномалий пользователя или конкретных инцидентов информационной безопасности уменьшить последствия или полностью остановить действия злоумышленника.

Ответ пользователю: сервер генерирует ответ, сообщающий пользователю об успешном сохранении данных или ошибке, а информация об успешном сохранении данных отображается в пользовательском интерфейсе.

VI. Интеграция в стороннюю систему

Помимо применения данного приложения обособленно присутствует возможность интеграции в различные системы и сервисы. В качестве примера для демонстрации была выбрана система электронного обучения и тестирования Moodle. Интеграция может проводиться следующим образом:

в качестве отдельного плагина;

как модификация уже существующих плагинов.

Для тестовой реализации был выбран второй вариант. Была проведена модификация плагина Database, который позволяет в пределах Moodle создать SQL базу данных и форму точки доступа для пользователей. Суть интеграции заключается в проверке каждой операции удаления, добавления и изменения файлов на предмет соответствия заданным политикам Py-ABAC. Проверка осуществляется посредством POST запросов на сервер Flask. Добавлена функция sendPostRequest в двух вариациях для реализации Delete и Write операций. Для определенных участков кода в которых реализованы нужный функционал добавлена дополнительная проверка, которая посредством POST запроса к веб сервису на Flask проверят действие на соответствии политикам.

В момент, когда идет попытка создать, изменить или удалить запись отсылается POST запрос, который принимает Flask веб сервис, после обработки запроса возвращается или логическая переменная, которая разрешает или запрещает выполнение операции. Данные об операции заносятся в лог файл веб сервиса Flask.

Данные проверки можно встроить в любой участок PHP кода исходного плагина или использовать при разработки своего. В каталоге данного плагина были модифицированы следующие файлы: edit.php, lib.php, locallib.php, view.php

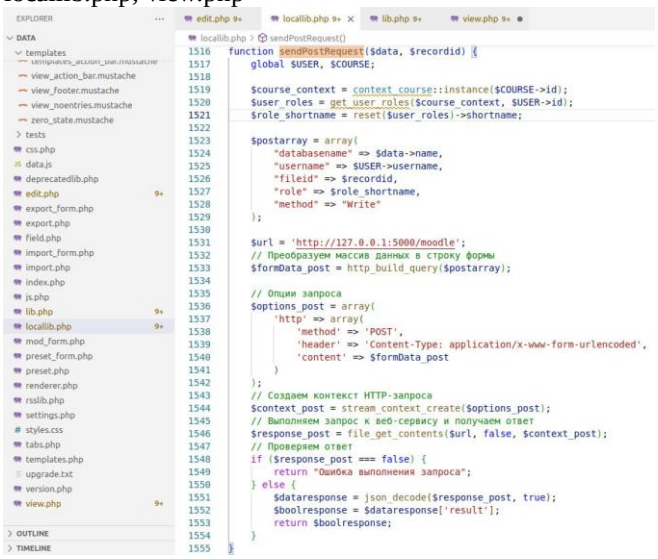

Рисунок 5. Файловая структура плагина

При нажатии на соответствующую кнопку на вебсайте на веб-сервис Flask приходит Json следующего содержания:

```
127.0.0.1 - - [27/Apr/2024 23:19:47] "POST /moodle HTTP/1.1" 200 -
Поле 'databasename': 'Science'
Поле 'amp;username': 'admin'
Поле 'amp;fileid': '0'
Поле 'amp;role': 'student'
Поле 'amp;method': 'Write'
127.0.0.1 - - [27/Apr/2024 23:19:51] "POST /moodle HTTP/1.1" 200 -
Поле 'databasename': 'Science'
Поле 'amp;username': 'admin'
Поле 'amp;fileid': '0'
Поле 'amp;role': 'student'
Поле 'amp;method': 'Write'
127.0.0.1 - - [27/Apr/2024 23:19:58] "POST /moodle HTTP/1.1" 200 -
Поле 'databasename': 'Science'
Поле 'amp;username': 'admin'
Поле 'amp;fileid': '0'
Поле 'amp;role': 'student'
Поле 'amp;method': 'Write'
127.0.0.1 - - [27/Apr/2024 23:20:01] "POST /moodle HTTP/1.1" 200 -
Поле 'databasename': 'Science'
Поле 'amp;username': 'admin'
Поле 'amp;fileid': '0'
Поле 'amp;role': 'student'
Поле 'amp;method': 'Write'
127.0.0.1 - - [27/Apr/2024 23:22:55] "POST /moodle HTTP/1.1" 200 -
Поле 'databasename': 'Science'
Поле 'amp;username': 'admin'
Поле 'amp;fileid': '202'
Поле 'amp;role': 'student'
Поле 'amp;method': 'Delete'
```

Рисунок 6. Пример JSON

Далее данные из этого Json используются для формирования другого Json, который идет на проверку. Риск, использующийся в процессе принятия решения, поступает посредством отдельного Request запроса на другой веб сервис, который рассчитывает риск системы на основе анализа логов. После проверки создается запись в логах о транзакции и возвращается ответ в Moodle.

```
71  student,Personal Laptop,VPN,Science,Write,High,None,False,1713809468
72  student,Personal Laptop,VPN,Science,Write,High,None,False,1714245882
73  student,Personal Laptop,VPN,Science,Write,Low,None,True,1714245903
74  student,Personal Laptop,VPN,Science,Delete,Low,204,True,1714245908
75  student,Personal Laptop,VPN,Science,Write,High,None,False,1714246039
76  student,Personal Laptop,VPN,Science,Write,High,None,False,1714246042
77  student,Personal Laptop,VPN,Science,Write,High,0,False,1714246922
78  student,Personal Laptop,VPN,Science,Write,High,0,False,1714246943
79  student,Personal Laptop,VPN,Science,Write,High,0,False,1714246963
80  student,Personal Laptop,VPN,Science,Delete,High,202,False,1714247522
81  student,Personal Laptop,VPN,Science,Write,High,0,False,1714249092
82  student,Personal Laptop,VPN,Science,Write,High,0,False,1714249147
83  student,Personal Laptop,VPN,Science,Write,High,0,False,1714249187
84  student,Personal Laptop,VPN,Science,Write,High,0,False,1714249191
85  student,Personal Laptop,VPN,Science,Write,High,0,False,1714249198
86  student,Personal Laptop,VPN,Science,Write,High,0,False,1714249201
87  student,Personal Laptop,VPN,Science,Write,High,0,False,1714249375
88  student,Personal Laptop,VPN,Science,Delete,High,202,False,1714249617
89
```

Рисунок 7. Пример логов транзакции

Далее будет описан пример работы интеграции.

В соответствии с политиками в одном из курсов необходимо создать базу данных Science. Вне рамок курса у пользователя необходимо задать роль в контексте всей системы, но текущая реализация получает роли на уровне контекста курсов.

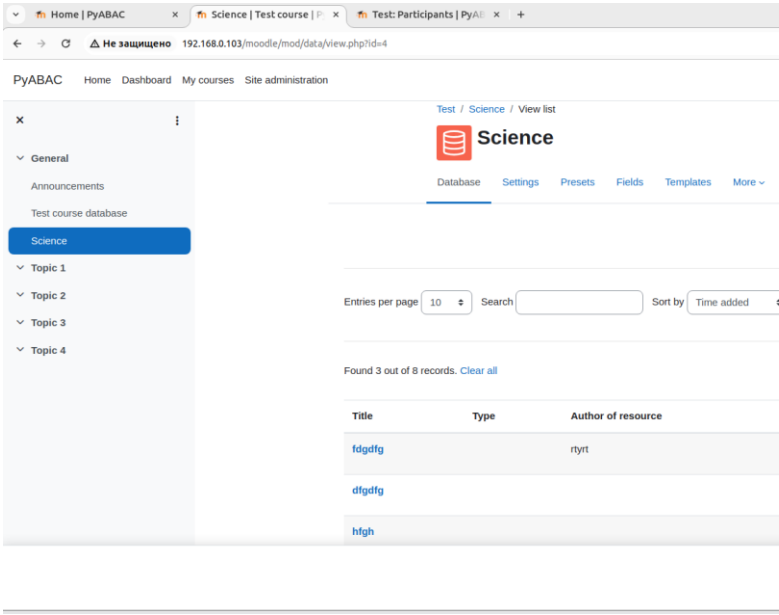

Рисунок 8. Главная страница плагина

В базе данных создано некоторое количество записей. Текущий риск системы высокий. На момент создания скриншотов риск задается статически, но в рамках дальнейшего развития проекта будет реализован сервис расчета риска, который будет передавать риск в реальном времени посредством POST запросов. На рисунке ниже представлен код Python который обрабатывает запросы со стороны веб сервиса управления доступом. Текущий пользователь - Admin User, роль в рамках курса студент.

```python
@app.route('/moodle', methods=['POST'])  # Маршрут для обработки данных из формы
def moodle_form():
    if request.method == 'POST':
        # Запись в дебаг лог.
        with open('temp_data.txt', mode='w', newline='') as file:
            exit = []
            for key, value in request.form.items():
                print(f"Поле '{key}': '{value}'")
                exit.append(f"Поле '{key}': '{value}'")
            json.dump(request.form, file)
        #
        role = str(request.form.get('amp;role'))
        device_type = "Personal Laptop"
        connection_type = "VPN"
        service = str(request.form.get('databasename'))
        method = str(request.form.get('amp;method'))
        file_id = str(request.form.get('amp;fileid'))
        risk = "High"
        decision = policy_handler.policy_proc(role, device_type, connection_type, service, method, risk)

        list_to_log = [role,device_type,connection_type,service,method,risk,file_id,str(decision[0])]
        policy_handler.logger(list_to_log) # Функция отправления записи в лог

        return jsonify({'result': decision[0]})
```

Рисунок 9. Код обработки запросов

Для корректной обработки политик требуется задавать роли во вкладке курса “Participants”. Так как текущая роль у пользователя - студент, ему запрещено как изменять, так и удалять файлы в базе данных. При попытке сохранить эту запись будет получено уведомление о том, что доступ запрещен. При попытке удалить запись также появится окно схожего содержания.

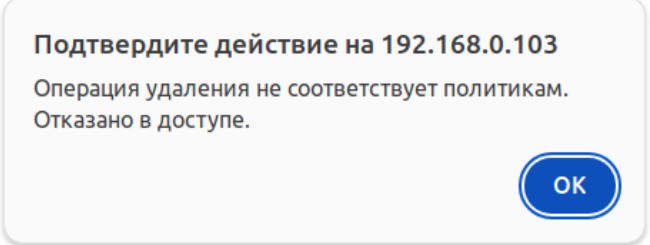

Рисунок 10. Пример текста уведомления

В этот момент на стороне Moodle функция sendPostRequest отправляет запрос на веб-сервис политик.

```php
function sendPostRequest($data, $recordid) {
    global $USER, $COURSE;

    $course_context = context_course::instance($COURSE->id);
    $user_roles = get_user_roles($course_context, $USER->id);
    $role_shortname = reset($user_roles)->shortname;

    $postarray = array(
        "databasename" => $data->name,
        "username" => $USER->username,
        "fileid" => $recordid,
        "role" => $role_shortname,
        "method" => "Write"
    );

    $url = 'http://127.0.0.1:5000/moodle';
    // Преобразуем массив данных в строку формы
    $formData_post = http_build_query($postarray);

    // Опции запроса
    $options_post = array(
        'http' => array(
            'method' => 'POST',
            'header' => 'Content-Type: application/x-www-form-urlencoded',
            'content' => $formData_post
        )
    );
    // Создаем контекст HTTP-запроса
    $context_post = stream_context_create($options_post);
    // Выполняем запрос к веб-сервису и получаем ответ
    $response_post = file_get_contents($url, false, $context_post);
    // Проверяем ответ
    if ($response_post === false) {
        return "Ошибка выполнения запроса";
    } else {
        $dataresponse = json_decode($response_post, true);
        $boolresponse = $dataresponse['result'];
        return $boolresponse;
    }
}
```

Рисунок 11. Функция sendPostRequest

Демонстрация трафика с помощью Wireshark

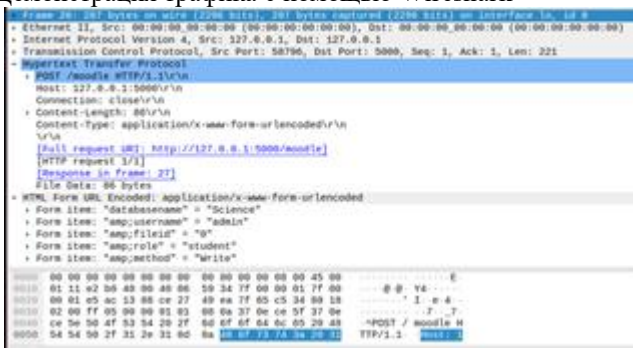

Рисунок 12. Пример запроса в Wireshark

На следующем рисунке продемонстриророван ответ на запрос.

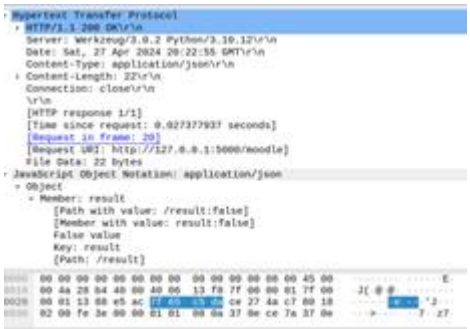

Рисунок 13. Ответ на запрос

После обработки запроса сервис вернет логическую переменную, которая будет использована в конструкции if/else.

VII. Заключение

В данной работе был проведен анализ существующих динамических моделей управления доступом, выявлены недостатки существующих решений и предоставлено новое решение, которое отличается гибкостью и легкостью механизма управления доступом.

Была представлена архитектура приложения, разработанного на базе Flask с использованием MongoDB и технологии управления доступом на основе Py-ABAC. Рассмотренный пример работы приложения продемонстрировал эффективное взаимодействие между клиентом и сервером через AJAX, сохранение и доступ к данным в базе MongoDB, а также важность применения системы ABAC для обеспечения безопасности данных.

Рассмотренное в статье приложение на базе Flask с применением MongoDB и ABAC является примером передовых технологий, способствующих созданию современных, безопасных и масштабируемых проектов.

Также на основе данного приложения была разработана интеграция в СДО moodle для плагина database, которая позволяет обеспечить доступ к базе данных внутри moodle.

Главным преимуществом данного решения является то, что его можно встроить в любой участок существующей системы, поскольку механизм управления доступом расположен в отдельном веб-сервисе.

Библиография

[1] Ptsecurity, «Кибербезопасность в 2023–2024 гг.: тренды и прогнозы. Часть пятая». Режим доступа: https://www.ptsecurity.com/ru-ru/research/analytics/kiberbezopasnost-v-2023-2024-gg-trendy-i-prognozy-chast-pyataya/ (дата обращения: 13.03.2024).

[2] НАЦИОНАЛЬНЫЙ СТАНДАРТ РОССИЙСКОЙ ФЕДЕРАЦИИ "Защита информации ФОРМАЛЬНАЯ МОДЕЛЬ УПРАВЛЕНИЯ ДОСТУПОМ" от 2021-06-01 № ГОСТ Р 59453.1-2021. Режим доступа: https://docs.cntd.ru/document/1200179191 (дата обращения: 15.03.2024)

[3] Kashmar, N., Adda, M., & Atieh, M. From Access Control Models to Access Control Metamodels: A Survey. //Advances in Biochemical Engineering/Biotechnology, 2019 – С. 892–911.

[4] Ma K., Yang G., Xiang Y. RCBAC: A risk-aware content-based access control model for large-scale text data. //Journal of Network and Computer Applications, 2020

[5] R. McGraw. Risk-Adaptable Access Control (RAdAC) //inPrivilege Manag. Work. NIST–National Inst. Stand. Technol. Technol. Lab., 2009 - С. 1–8.

[6] S. Kandala, R. Sandhu, and V. Bhamidipati, An Attribute Based Framework for Risk-Adaptive Access Control Models //Proc. 6th Int. Conf. Availability, Reliab. Secur., 2011 - С. 236–241.

[7] N. N. Diep, L. X. Hung, Y. Zhung, S. Lee, Y. Lee, and H. Lee, Enforcing Access Control Using Risk Assessment //Fourth Eur. Conf. Univers. Multiservice Networks, 2007 - С. 419–424.

[8] L. Rajbhandari and E. A. Snekkenes, Using game theory to analyze risk to privacy: An initial insight //Priv. Identity Manag. Life, Springer Berlin Heidelb., 2011 - С. 41–51.

[9] M. Sharma, Y. Bai, S. Chung, and L. Dai, Using risk in access control for cloud-assisted ehealth //14th Int. Conf. High Perform. Comput. Commun. IEEE, 2012 - С. 1047–1052.

[10] S. Lee, Y. W. Lee, N. N. Diep, S. Lee, Y. Lee, and H. Lee, Contextual Risk-based access control //Proc. 2007 Int. Conf. Secur. Manag., p. 2007 – С.406–412.

[11] Магомедов Ш.Г., Козачок А.В., Тарланов А.Т. Риск-ориентированная атрибутивная модель управления доступом для организаций высшего образования // Правовая информатика № 1, 2023 – С.72-82

[12] Petrović Dejan V., Miloš Tanasijević, Saša Stojadinović, Jelena Ivaz, Pavle Stojković, Fuzzy Model for Risk Assessment of Machinery Failures //Symmetry. 2020 - Vol. 12, No. 4, p. 525.

[13] Козачок А.В. Спецификация модели управления доступом к разнокатегорийным ресурсам компьютерных систем // Вопросы кибербезопасности. 2018. № 4 (28). С. 2-8.

[14] Д. Н. Колегов, О. В. Брославский, Н. Е. Олексов. О фрэймворке атрибутного управления доступом ABAC Engine //ПДМ. Приложение, 2017 - № 10, 115–120

[15] Py-ABAC. «Py-ABAC's documentation». Режим доступа: https://py-abac.readthedocs.io/en/latest/index.html (дата обращения: 27.02.2024).
[16] OASIS, «eXtensible Access Control Markup Language (XACML) Version 3.0». Режим доступа: https://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html (дата обращения: 20.02.2024).
[17] Github, «Attribute-based access control (ABAC) SDK for Python». Режим доступа: https://github.com/kolotaev/vakt (дата обращения: 27.02.2024).

Software Complex for Risk-Oriented Attribute-Based Access Control Mechanism

A.Shitov, N.Stelmakh, S. Magomedov

***Abstract*—The article aims to develop a software complex implementing a risk-based attribute access control mechanism that dynamically processes requests from target services, as well as integrating it into existing systems to demonstrate the flexibility of the solution.**

The article analyzes access control models, particularly risk-based ones. The advantages of dynamic access control models are discussed. Special attention is paid to the risk-oriented model based on fuzzy logic, its distinctive features are considered, and its selection as the basis for the developed access control mechanism is explained.

A software implementation of the risk-oriented access control mechanism is presented, developed using the Py-ABAC Python library. Integration with the Moodle learning and testing system has been created and tested at the pre-production level.

Attention is focused on the details related to the use of the risk-oriented access control model based on fuzzy logic, its advantages, and a variation of the software implementation of the access control mechanism based on this model is proposed.

Keywords—Information Security, Access Control Models, Python, Py-ABAC, Risk-Oriented Access Control Model, Fuzzy Logic, Moodle.

REFERENCES

[1] Ptsecurity, «Kiberbezopasnost' v 2023–2024 gg.: trendy i prognozy. Chast' pjataja». — online; accessed: 13.03.2024. — URL: https://www.ptsecurity.com/ru-ru/research/analytics/kiberbezopasnost-v-2023-2024-gg-trendy-i-prognozy-chast-pyataya/

[2] NACIONAL"NYJ STANDART ROSSIJSKOJ FEDERACII. Zashhita informacii FORMAL"NAJa MODEL" UPRAVLENIJa DOSTUPOM" ot 2021-06-01 # GOST R 59453.1-2021. — online; accessed: 15.03.2024. — URL: https://docs.cntd.ru/document/1200179191

[3] Kashmar, N., Adda, M., & Atieh, M. From Access Control Models to Access Control Metamodels: A Survey. //Advances in Biochemical Engineering/Biotechnology, 2019 – S. 892–911.

[4] Ma K., Yang G., Xiang Y. RCBAC: A risk-aware content-based access control model for large-scale text data. //Journal of Network and Computer Applications, 2020

[5] R. McGraw. Risk-Adaptable Access Control (RAdAC) //inPrivilege Manag. Work. NIST–National Inst. Stand. Technol. Technol. Lab., 2009 - S. 1–8.

[6] S. Kandala, R. Sandhu, and V. Bhamidipati, An Attribute Based Framework for Risk-Adaptive Access Control Models //Proc. 6th Int. Conf. Availability, Reliab. Secur., 2011 - S. 236–241.

[7] N. N. Diep, L. X. Hung, Y. Zhung, S. Lee, Y. Lee, and H. Lee, Enforcing Access Control Using Risk Assessment //Fourth Eur. Conf. Univers. Multiservice Networks, 2007 - S. 419–424.

[8] L. Rajbhandari and E. A. Snekkenes, Using game theory to analyze risk to privacy: An initial insight //Priv. Identity Manag. Life, Springer Berlin Heidelb., 2011 - S. 41–51.

[9] M. Sharma, Y. Bai, S. Chung, and L. Dai, Using risk in access control for cloud-assisted ehealth //14th Int. Conf. High Perform. Comput. Commun. IEEE, 2012 - S. 1047–1052.

[10] S. Lee, Y. W. Lee, N. N. Diep, S. Lee, Y. Lee, and H. Lee, Contextual Risk-based access control //Proc. 2007 Int. Conf. Secur. Manag., p. 2007 – S.406–412.

[11] Magomedov Sh.G., Kozachok A.V., Tarlanov A.T. Risk-orientirovannaja atributivnaja model' upravlenija dostupom dlja organizacij vysshego obrazovanija // Pravovaja informatika # 1, 2023 – S.72-82

[12] Petrović Dejan V., Miloš Tanasijević, Saša Stojadinović, Jelena Ivaz, Pavle Stojković, Fuzzy Model for Risk Assessment of Machinery Failures //Symmetry. 2020 - Vol. 12, No. 4, p. 525.

[13] Kozachok A.V. Cpecifikacija modeli upravlenija dostupom k raznokategorijnym resursam komp'juternyh sistem // Voprosy kiberbezopasnosti. 2018. # 4 (28). S. 2-8.

[14] D. N. Kolegov, O. V. Broslavskij, N. E. Oleksov. O frjejmvorke atributnogo upravlenija dostupom ABAC Engine //PDM. Prilozhenie, 2017 - # 10, 115–120

[15] Py-ABAC. «Py-ABAC's documentation». — online; accessed: 27.02.2024. — URL: https://py-abac.readthedocs.io/en/latest/index.html

[16] OASIS, «eXtensible Access Control Markup Language (XACML) Version 3.0». ». — online; accessed: 20.02.2024. — URL: https://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html

[17] Github, «Attribute-based access control (ABAC) SDK for Python». ». — online; accessed: 27.02.2024. — URL: https://github.com/kolotaev/vakt

A. V. Shitov is with the MIREA – Russian Technological University, Moscow, Russia (e-mail: Bulbazavr41@yandex.ru, shitov.a.v2@edu.mirea.ru).

N. E. Stelmakh is with the MIREA – Russian Technological University, Moscow, Russia (e-mail: stelmach.stels@yandex.ru, stelmakh.n.e@edu.mirea.ru).

S. G. Magomedov is with the MIREA – Russian Technological University, Moscow, Russia (e-mail: magomedov_sh@mirea.ru).