ОБЕСПЕЧЕНИЕ БЕЗОПАСНОСТИ СЕРВЕРНОЙ ИНФРАСТРУКТУРЫ НА БАЗЕ ФРЕЙМВОРКА KTOR ДЛЯ АНАЛИЗА ОПЕРАТИВНОЙ ОБСТАНОВКИ

Кочеров Р.С., студент гр.230501

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Супрун Ф.Н. – преподаватель

Аннотация. Одной из перспективных платформ для построения высоконагруженных сервисов является асинхронный фреймворк Ktor, созданный на языке Kotlin. Он позволяет создавать масштабируемые сервисы, но требует тщательного проектирования защитных механизмов на всех уровнях: от приёма сетевых запросов до хранения данных в базах. В данной статье рассматривается комплексный подход к обеспечению безопасности Ktor‑сервера для органов внутренних дел Республики Беларусь с акцентом на моделирование угроз, реализацию концепции нулевого доверия, защиту от SQL‑инъекций, управление токенами доступа и противодействие DDoS‑атакам.

Ключевые слова. Безопасность, Ktor, серверная защита, модель угроз, Zero Trust, SQL-инъекции, Exposed, JWT, аутентификация, авторизация, DDoS.

В современном мире происходит активное внедрение цифровых технологий во все сферы деятельности государства, включая правоохранительные органы. Этот процесс получил название цифровая трансформация - переход от традиционных бумажных и ручных методов работы к автоматизированным информационным системам, основанным на компьютерной обработке данных. Для органов внутренних дел Республики Беларусь цифровая трансформация означает создание и использование информационно-аналитических систем - программных комплексов, которые собирают, хранят, обрабатывают и анализируют большие объёмы информации о правонарушениях, преступлениях, участниках процессов и оперативной обстановке.

Однако цифровизация порождает новые уязвимости. Информация, циркулирующая в таких системах, часто относится к категории ограниченного доступа: это персональные данные граждан (фамилии, адреса, паспортные данные, биометрические сведения), служебная информация о расследованиях, оперативные сводки. Поэтому одновременно с внедрением систем необходимо обеспечить их защиту от несанкционированного доступа, утечек, искажения или уничтожения данных. В Республике Беларусь действует Закон «О персональных данных», а также ведомственные нормативные акты, которые устанавливают обязательные требования к обработке и защите такой информации. Соблюдение этих требований является не просто технической задачей, а юридической обязанностью.

Прежде чем строить защиту, необходимо знать какие угрозы могут нарушить работу сервера или завладеть полезной информацией. В информационной безопасности выделяется такое понятие, как моделью угроз - систематизированное описание всех возможных опасностей, их источников, способов реализации и потенциального ущерба. Модель угроз разрабатывается для конкретной системы с учётом её архитектуры, данных и условий эксплуатации.

Для ведомственной информационной системы, работающей в органах внутренних дел можно выделить следующие риски:

Внешние атаки на сетевой периметр. Сетевой периметр – *это* граница между внутренней защищённой сетью органов и внешней средой. Злоумышленник, находящийся снаружи, может попытаться проникнуть внутрь через уязвимости в сетевых сервисах, например, через веб-приложение сервера.

Попытки несанкционированного доступа через уязвимости веб‑приложений. Веб-приложение – это программа, работающая на сервере и взаимодействующая с пользователями через браузер или мобильное приложение. Если в коде есть ошибки, злоумышленник может использовать их, чтобы получить доступ к данным или выполнить запрещённые команды.

Компрометация учётных записей сотрудников. Компрометация означает, что злоумышленник каким-то образом завладел логином и паролем сотрудника и теперь может действовать от его имени.

Превышение должностных полномочий. Даже легальный пользователь может намеренно или случайно выполнить операции, на которые у него нет.

Случайные или преднамеренные утечки информации. Это может быть как злой умысел сотрудника, который копирует данные на флешку, так и случайная ошибка - отправка отчёта не по адресу.

Особую опасность представляют внутренние угрозы, то есть исходящие от самих сотрудников или обслуживающего персонала. Внутренние угрозы считаются более сложными для предотвращения, потому что человек уже имеет легальный доступ к системе и знает её особенности. Кроме того, в больших ведомствах работает много людей с разными уровнями допуска, что увеличивает вероятность ошибок или злоупотреблений.

В ответ на перечисленные риски современная практика информационной безопасности предлагает подход Zero Trust. Концепция «нулевого доверия» пересматривает рабочие процессы на основе

отсутствия доверия к любому пользователю перед началом каждой операции. Для этого система должна автоматически проводить процедуру аутентификации и авторизации пользователя, прежде чем предоставить ему доступ к какому-либо приложению, базе данных или другим ресурсам компании. Кроме того, в процессе работы с приложениями и данными статус авторизации каждого пользователя постоянно перепроверяется [1].

В архитектуре Ktor-сервера принцип нулевого доверия реализуется через многоступенчатую проверку каждого запроса к API. Каждый запрос, прежде чем получить доступ к бизнес-логике или к базе данных, проходит следующие этапы:

1. Аутентификация – проверка, действительно ли пользователь тот, за кого себя выдаёт с помощью пароля, логина и токена.

2. Авторизация – проверка, имеет ли аутентифицированный пользователь право выполнять именно эту операцию.

3. Валидация входных данных – проверка, что данные, присланные пользователем, имеют правильный формат, не содержат вредоносного кода и находятся в допустимых пределах.

Только после успешного прохождения всех трёх этапов запрос передаётся на обработку. Ни один запрос не получает доступа к данным или функциям без полного подтверждения своих полномочий. Такой подход особенно важен для распределённых систем, где сотрудники работают с удалённых автоматизированных рабочих мест, планшетов патрульных экипажей или мобильных терминалов.

SQL-инъекция – это один из самых распространённых и опасных видов атак на веб-приложения, работающие с реляционными базами данных. Реляционная база данных – это способ организации данных в виде таблиц, связанных между собой. Для управления такими базами данных используется язык SQL (Structured Query Language – структурированный язык запросов). Программа формирует SQL-команды, чтобы найти, добавить, изменить или удалить данные.

Уязвимость SQL-инъекции возникает, если разработчик напрямую вставляет в текст SQL-команды данные, введённые пользователем. Например, представьте, что программа строит запрос так:

```
"SELECT * FROM users WHERE login = '" + userInput + "'".
```

Если злоумышленник в поле ввода напишет `' OR '1'='1`, то получится запрос:

```
SELECT * FROM users WHERE login = '' OR '1'='1'.
```

Условие `'1'='1'` всегда истинно, поэтому база данных вернёт всех пользователей, и злоумышленник может войти в систему без пароля. Ещё более опасные варианты позволяют удалять таблицы, читать чужие данные или выполнять произвольные команды на сервере.

В Ktor-сервере для работы с базами данных используется библиотека Exposed. Exposed основана на параметризованных запросах. Вместо того чтобы вставлять пользовательский ввод прямо в текст SQL-команды, программист пишет шаблон запроса, где вместо данных ставятся специальные метки-параметры. Затем значения параметров передаются драйверу базы данных отдельно от текста команды. Драйвер – это программа-посредник между приложением и базой данных, которая отвечает за безопасную подстановку данных. В результате даже если злоумышленник попытается внедрить вредоносный код, драйвер интерпретирует его как обычную строку данных, а не как часть SQL-команды. В проекте категорически запрещено формировать SQL-запросы через конкатенацию строк, то есть склеивание кусочков текста с помощью оператора + или аналогичных методов. Такой подход считается грубейшим нарушением стандартов безопасной разработки.

Для контроля качества кода используется статический анализатор – программа, которая проверяет исходный код без его запуска, ищет потенциально опасные конструкции. В проекте применяется инструмент detekt с набором правил безопасности. Если анализатор находит запрещённый способ формирования запроса, он блокирует сборку проекта, то есть не позволяет создать рабочую версию программы, пока ошибка не исправлена.

Для подтверждения личности пользователя и разграничения его прав доступа в системе используется стандарт JSON Web Tokens.

JWT состоит из трёх частей, разделённых точками:

1. Заголовок – содержит информацию о том, как подписан токен.

2. Полезная нагрузка – набор утверждений о пользователе: его идентификатор, роль, уникальный номер токена, время выдачи и истечения срока действия.

3. Подпись – результат криптографического преобразования заголовка и полезной нагрузки с использованием секретного ключа [2].

В Ktor интеграция с JWT выполняется через официальный плагин. Токены подписываются асимметричным алгоритмом RS256. Асимметричный алгоритм означает, что используются два ключа: закрытый и открытый. Закрытый ключ хранится на сервере в защищённом хранилище секретов. Этим ключом сервер подписывает токены. Открытый ключ используется для проверки подписи - его можно распространять свободно, потому что с его помощью нельзя подделать подпись, а только проверить её подлинность. Если злоумышленник попытается изменить полезную нагрузку токена, подпись станет недействительной, и сервер отклонит такой токен.

Проверка прав доступа происходит на каждом маршруте сервера. Система проверяет не только срок действия токена, но и наличие у роли пользователя разрешения на выполнение запрошенной операции. Рядовой сотрудник милиции не может просматривать документы для руководства, а аналитик не может вносить изменения в оперативные данные. Такой подход соответствует принципу наименьших привилегий – каждому пользователю выдаётся ровно тот минимум прав, который необходим ему для выполнения его служебных обязанностей, и не более того.

Если токен украден, то злоумышленник сможет выдавать себя за легального пользователя в течение всего срока действия токена. Чтобы минимизировать ущерб от такой кражи, в системе реализована двухуровневая схема с двумя видами токенов:

Токен доступа (access token) – короткоживущий, его срок действия составляет не более 15 минут. Он используется для каждого запроса к API. Если токен украден, злоумышленник сможет использовать его только в течение этого короткого окна.

Токен обновления (refresh token) – может действовать часы или дни. Он предназначен исключительно для получения нового токена доступа, когда старый истекает. Токен обновления хранится на сервере в связке с идентификатором устройства и отпечатком браузера. Это позволяет серверу убедиться, что запрос на обновление токена приходит с того же устройства, с которого выполнялся вход.

Для принудительного отзыва токенов, то есть немедленной отмены их действия до истечения срока, может использоваться высокопроизводительное хранилище Redis. Redis - это база данных, размещаемая в памяти, которая используется, в основном, в роли кеша, находящегося перед другой, центральной базой данных, вроде MySQL или PostgreSQL. Кеш, основанный на Redis, помогает улучшить производительность приложений. Он эффективно использует скорость работы с данными, характерную для памяти, и смягчает нагрузку центральной базы данных приложения [3]. В нашем случае в Redis хранится чёрный список идентификаторов токенов, которые были отозваны.

Доступность - одно из трёх основных свойств информационной безопасности. Оно означает, что система должна быть работоспособна и отвечать на запросы пользователей в любое необходимое время. Для правоохранительных органов недоступность информационной системы может иметь критические последствия: невозможно проверить человека по базам данных, зарегистрировать заявление, получить оперативную сводку.

Одним из самых серьёзных способов нарушить доступность является DDoS-атака (Distributed Denial of Service – распределённая атака типа «отказ в обслуживании»). Злоумышленник использует сеть из множества заражённых компьютеров и направляет от них огромное количество запросов на сервер жертвы. Сервер не справляется с нагрузкой, начинает тормозить, а затем полностью перестаёт отвечать. Для противодействия DDoS на уровне Ktor-приложения реализован плагин Rate Limiting. Этот плагин подсчитывает, сколько запросов приходит от одного IP-адреса или от одного пользователя за единицу времени, например, за секунду или минуту. Если количество превышает установленный лимит, последующие запросы отклоняются.

Валидация – это процесс проверки того, что данные, присланные пользователем, соответствуют ожидаемому формату, типу, диапазону значений и не содержат недопустимых символов. Валидация выполняет две функции: защита от ошибок - предотвращает попадание в базу данных некорректных данных; защита от атак - отсеивает попытки внедрения вредоносного кода. Необходимо обеспечить проверку соответствия JSON-схем, формата полей, диапазона значений. Некорректные или потенциально опасные запросы отклоняются ещё до того, как достигнут бизнес-логики. Это снижает риск эксплуатации уязвимостей на уровне приложения, поскольку вредоносные данные просто не доходят до мест, где они могли бы причинить вред.

Таким образом, при построении защищённого Ktor-сервера для информационных систем органов внутренних дел применяется архитектура на основе принципа нулевого доверия с многоступенчатой проверкой каждого запроса к API (аутентификация, авторизация, валидация). Для защиты от SQL-инъекций используются параметризованные запросы через библиотеку Exposed, категорически запрещена конкатенация строк, а качество кода контролируется статическим анализатором detekt. Аутентификация и разграничение доступа реализованы на основе JWT с асимметричной подписью RS256, применяется двухуровневая схема с короткоживущими токенами доступа (15 минут) и токенами обновления, принудительный отзыв токенов осуществляется через высокопроизводительное хранилище Redis. Доступность сервиса обеспечивается механизмом ограничения частоты запросов (Rate Limiting), а валидация входных данных по JSON-схемам позволяет отклонять некорректные и опасные запросы до попадания в бизнес-логику.

Список использованных источников:

1. Kaspersky [Электронный ресурс]. URL: https://www.kaspersky.ru/resource-center/definitions/zero-trust (дата обращения: 02.04.2026).

2. Wikipedia [Электронный ресурс]. URL: https://en.wikipedia.org/wiki/JSON_Web_Token (дата обращения: 02.04.2026).

3. Хабр [Электронный ресурс]. URL: https://habr.com/ru/companies/wunderfund/articles/685894 (дата обращения: 01.04.2026).