SkyV Discovery

Система сбора информации посредством дискаверинга

ТЕХНИЧЕСКАЯ АРХИТЕКТУРА

Версия 0.01

Москва, 2026 г.

Аннотация

Настоящий документ описывает техническую архитектуру программного обеспечения SkyV Discovery. Описание сформировано по предоставленным исходным кодам серверной и клиентской частей, конфигурациям контейнерного развертывания и руководству пользователя.

SkyV Discovery предназначен для управления агентами дискаверинга, запуска сетевого и агентского сбора, контроля доступности и состояния агентов, работы с результатами сбора и администрирования параметров доступа.

1. Общие сведения

1.1. Назначение решения

SkyV Discovery является веб-системой для централизованного управления процессами дискаверинга в корпоративной инфраструктуре. Решение обеспечивает настройку источников и сценариев сбора, взаимодействие с агентами, запуск операций по запросу или по расписанию, контроль выполнения и предоставление результатов пользователям в зависимости от их прав доступа.

В состав функциональных сценариев входят:

- настройка и ведение реестра агентов инвентаризационного, NMAP- и SNMP-сбора;
- мониторинг состояния сетевого оборудования;
- запуск сетевого сканирования адресов, подсетей, портов и сервисов;
- опрос устройств по SNMP, включая работу с профилями MIB и параметрами подключения;
- получение инвентаризационных данных от программных агентов, включая сведения об операционной системе, оборудовании, программном обеспечении, сети, сервисах и процессах;
- контроль доступности, статуса, версии, очереди сообщений и диагностической информации агентов;
- управление учетными записями, ролями, группами агентов, настройками подключения и защищенными учетными данными;
- журналирование действий, просмотр уведомлений, поиск, фильтрация и экспорт результатов.

1.2. Область описания и источник данных

Архитектура описывает компоненты, прямо представленные в предоставленных архивах исходного кода: сервер управления агентами на Go, веб-консоль управления агентами на Vue, дополнительный веб-клиент на React, конфигурации Docker Compose, PostgreSQL, Kafka и Nginx.

Исходные коды самих исполняемых агентов дискаверинга и серверной части API, используемой дополнительным React-клиентом, в составе предоставленных архивов отсутствуют. Поэтому документ фиксирует их интерфейсные границы и назначение, но не приписывает им внутреннюю реализацию, не подтвержденную исходным кодом.

2. Обобщенная техническая архитектура

Архитектура построена по клиент-серверной модели. Пользователи работают через браузер; Nginx предоставляет статические ресурсы веб-интерфейса и проксирует запросы к серверу управления. Сервер управления хранит реестр агентов и параметры доступа в PostgreSQL, а также обращается к агентам по HTTP/JSON или gRPC. Для работы с сообщениями агентов в контейнерной конфигурации предусмотрен Apache Kafka.

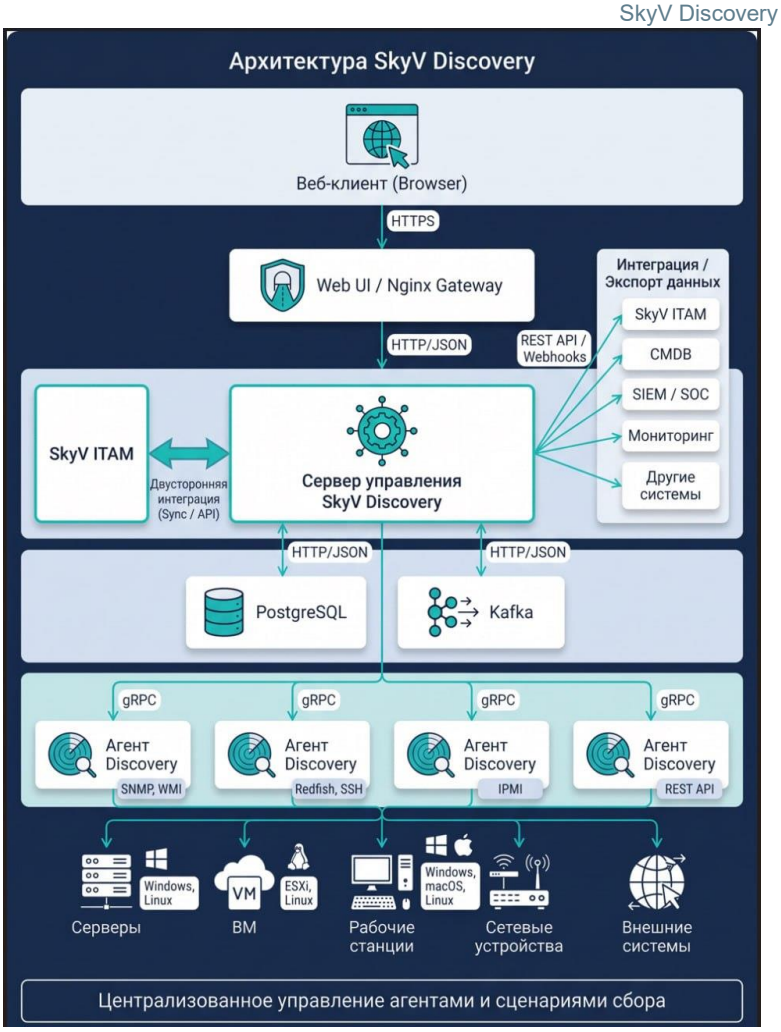

Рисунок 1 - Обобщенная техническая архитектура SkyV Discovery

2.1. Состав основных компонентов

Компонент	Назначение	Технологическая реализация
Веб-консоль управления агентами	Рабочее место администратора и оператора: вход в систему, реестр типов и экземпляров агентов, параметры подключения, конфигурация, проверка доступности, просмотр состояния, сообщений брокера и метрик.	Vue 3, TypeScript, Vite, Pinia, Vue Router, Axios. Производственная сборка размещается в Nginx.
Дополнительный веб-клиент	Интерфейс работы с устройствами, группами, настройками, диагностикой, метриками, получателями уведомлений и операциями импорта/экспорта. Использует отдельный прикладной API-контракт.	React 18, JavaScript/JSX, Redux Toolkit / RTK Query, React Router, Ant Design, Nginx.
Сервер управления агентами	Единая точка API для аутентификации, реестра агентов, групп, типов, учетных данных, диагностики, конфигурации, запуска операций и проксирования запросов к агентам.	Go 1.26.4; HTTP REST через gRPC-Gateway; gRPC и Protocol Buffers.
Nginx gateway	Публикация веб-интерфейса и маршрутизация HTTP-запросов к API, health-check и документации API.	Nginx в контейнере. Для frontend-конфигурации используются заголовки X-Content-Type-Options и X-Frame-Options.

Компонент	Назначение	Технологическая реализация
PostgreSQL	Постоянное хранение реестра пользователей, ролей, типов и групп агентов, экземпляров агентов, параметров соединения и зашифрованных секретов.	PostgreSQL 16, доступ из Go через pgx.
Apache Kafka	Проверка состояния очередей и чтение последних сообщений агентов при использовании брокера сообщений.	Apache Kafka 3.9.0; доступ из Go через kafka-go.
Агенты дискаверинга	Выполнение сетевого сканирования, SNMP-опроса и агентского инвентаризационного сбора в сегментах сети, где они развернуты.	Внешние по отношению к серверу управления компоненты. Доступ по HTTP/JSON или gRPC; поддерживается TLS по конфигурации.

2.2. Веб-клиенты

В предоставленных исходных кодах присутствуют две самостоятельные реализации пользовательского интерфейса. Базовая контейнерная конфигурация сервера управления собирает и запускает веб-консоль на Vue. Дополнительный React-клиент обслуживает сценарии работы с устройствами и метриками через API с путем /v1/backend. Реализация данного прикладного API не входит в состав переданных архивов и должна быть развернута совместно с этим клиентом в целевом контуре.

3. Логические уровни и взаимодействие компонентов

3.1. Уровень представления

Пользовательский интерфейс запускается в браузере. Веб-консоль управления агентами реализует аутентификацию, навигацию по типам и группам агентов, управление конфигурацией и подключением, а также отображение метрик и сообщений брокера. Маршрутизация в приложении выполняется на стороне клиента, а Nginx возвращает index.html для неизвестных маршрутов.

В веб-консоли доступны роли администратора и наблюдателя, а в пользовательских сценариях SkyV Discovery также выделяется оператор дискаверинга. Проверка полномочий выполняется как в интерфейсе, так и на сервере API.

3.2. Прикладной уровень

Сервер управления агентами запускает HTTP-сервер и gRPC-сервер. HTTP-интерфейс строится на базе gRPC-Gateway и предоставляет REST API в пространстве /api/v1. Для разработки и интеграции доступны OpenAPI/Swagger-описание, health-check и gRPC reflection при включении соответствующей настройки.

Серверная часть содержит следующие логические модули:

Модуль	Функции
Аутентификация и авторизация	Вход пользователя, выдача и обновление JWT-токенов, получение информации о текущем пользователе, проверка доступа к административным операциям.
Реестр агентов	Управление типами, группами и экземплярами агентов; хранение адресов, портов, выбранного транспорта, базового пути, признака TLS и тайм-аутов.
Конфигурация и секреты	Передача конфигурации и учетных данных агентам, работа с профилями подключения, защищенное хранение секретов.

Модуль	Функции
Операции дискаверинга	Запуск NMAP-сканирования, SNMP-сканирования подсети и отдельного устройства; работа с MIB-профилями, инвентаризационными агентами, режимами, статусами, фильтрами пакетов и VMware-сбором.
Наблюдаемость и диагностика	Проверка health, получение обзора экземпляра агента, диагностики, статусов брокера, очередей, сообщений и метрик.
Проксирование запросов	Передача отдельных разрешенных запросов к API агента через центральный сервер с сохранением модели доступа и контроля.

3.3. Транспортный уровень и агенты

Для каждого экземпляра агента в реестре фиксируются хост, HTTP- и gRPC-порты, базовый путь API, режим использования TLS и тайм-аут. Фабрика транспорта выбирает HTTP или gRPC по настройке экземпляра. В исходном коде предусмотрены клиенты для следующих типов агентов:

Тип агента	Поддерживаемые операции
Агент инвентаризации	Получение и изменение конфигурации, реестр программных агентов, группы, статистика, метрики, справочники режимов и статусов, фильтры пакетов, запуск сбора VMware, управление расписанием и брокером сообщений.
Агент NMAP	Запуск сетевого сканирования по набору сетей, получение параметров портов и конфигурации сканирования.
Агент SNMP	Запуск сканирования подсети и отдельного устройства, управление и просмотр профилей MIB, получение конфигурации.
Агент Mon	Мониторинг сетевого оборудования, сбор и визуализация данных.
Агент мониторинга	Поддерживается сервером управления как расширяемый тип для получения конфигурации и диагностических данных. Использование этого типа определяется составом конкретной поставки.

Взаимодействие с агентами допускает передачу Bearer-токена. При использовании TLS клиентские сертификаты центра сертификации могут задаваться через параметры окружения. В HTTP-режиме используются JSON-запросы, в gRPC-режиме - контракты Protocol Buffers.

3.4. Типовые потоки данных

Сценарий	Последовательность взаимодействий
Вход в систему	Браузер -> Nginx -> REST API -> модуль аутентификации -> PostgreSQL. После проверки пароля клиент получает токен доступа и передает его в заголовке Authorization при последующих запросах.
Создание и настройка агента	Веб-консоль -> REST API -> реестр агентов в PostgreSQL. Параметры подключения и секреты сохраняются централизованно; конфигурация передается агенту после проверки доступа.
Запуск NMAP- или SNMP-сканирования	Оператор -> API сервера управления -> транспортный адаптер -> соответствующий агент -> разрешенные сетевые адреса и устройства. Агент возвращает идентификатор задания и состояние выполнения через свой API.
Агентский инвентаризационный сбор	Программный агент или сервер инвентаризации передает сведения о доступных объектах и параметрах. Сервер управления получает статус, статистику и метрики через интерфейс агента; результаты отображаются в реестрах SkyV Discovery.

Сценарий	Последовательность взаимодействий
Контроль очередей	Сервер управления -> Apache Kafka или иной брокер, заданный в конфигурации -> чтение статуса, задержки и ограниченного числа последних сообщений для диагностического просмотра.

4. Хранение и обработка данных

4.1. Данные в PostgreSQL

Схема базы данных сервера управления содержит сущности ролей и пользователей, типов и групп агентов, экземпляров агентов, параметров соединения, типов секретов и зашифрованных значений. Структура обеспечивает разделение публичных параметров подключения и конфиденциальной информации.

Категория данных	Примеры хранимых сведений	Особенности
Пользователи и роли	логин, хэш пароля, роль, дата создания, дата блокировки	Роли используются для ограничения доступа; пароль хранится не в открытом виде.
Типы и группы агентов	код типа, наименование, описание, признак доступности, группа	Поддерживает организацию экземпляров по функциональному типу и логическому назначению.
Экземпляры агентов	имя, включенность, адрес, HTTP/gRPC-порт, транспорт, базовый путь, TLS, тайм-аут	Определяют, куда и как сервер управления направляет запросы.
Секреты и учетные данные	токены агентов, лицензии, ссылки на сертификаты	Значения шифруются до сохранения в базе данных.
Технические журналы и метрики	статусы, health, диагностические сведения, сообщения брокера	Получаются по API агентов и брокера. Длительное хранение метрик во времени может выполняться внешней системой мониторинга.

4.2. Защита конфиденциальных данных

Для хранения секретов используется симметричное шифрование AES-GCM с 32-байтовым ключом, задаваемым в параметрах окружения. Пароли пользователей хэшируются алгоритмом Argon2id с солью и настраиваемыми параметрами памяти, числа итераций, параллелизма и длины ключа.

5. Интерфейсы и протоколы

Интерфейс	Назначение	Технология / особенности
Web UI	Работа пользователей в браузере.	HTTP/HTTPS; статические файлы Vue или React размещаются в Nginx.
REST API сервера управления	Интеграция веб-консоли и внешних клиентов: аутентификация, пользователи, агенты, конфигурация, диагностика, сканирования, брокер.	HTTP, JSON, gRPC-Gateway. Базовый путь: /api/v1.
gRPC API сервера управления	Высокопроизводительный интерфейс на основе контрактов Protocol Buffers.	gRPC; порт по умолчанию 9090.

Интерфейс	Назначение	Технология / особенности
API агентов	Передача конфигурации, запуск сканирования, опрос состояния, получение метрик и диагностик.	HTTP/JSON или gRPC; Bearer token; TLS может быть включен для конкретного агента.
PostgreSQL	Хранение реестра и параметров решения.	Протокол PostgreSQL; порт 5432 в внутренней контейнерной сети.
Kafka	Диагностический доступ к сообщениям и расчет задержки очередей.	Kafka protocol; порт 9092 в внутренней контейнерной сети.
Prometheus	Экспорт технических метрик сервера при включении настройки.	Pull-модель; порт задается конфигурацией окружения.

6. Безопасность и разграничение доступа

- Аутентификация пользователей реализована на основе JWT. В токен включаются идентификатор пользователя, логин, роль, тип токена и сроки действия.
- Access token и refresh token имеют отдельные сроки действия, задаваемые конфигурацией. Значения по умолчанию в исходном коде: 15 минут для access token и 168 часов для refresh token.
- Доступ к административным операциям ограничивается серверными interceptor-проверками роли, а также ограничениями в пользовательском интерфейсе.
- Секреты агентов не хранятся в открытом виде; ключ шифрования не должен включаться в исходный код или передаваться через интерфейс пользователя.
- Для каналов HTTP и gRPC предусмотрено использование TLS с доверенным центром сертификации, задаваемым в конфигурации клиента.
- При сканировании сети следует включать только согласованные диапазоны, адреса и методы опроса. Это соответствует назначению SkyV Discovery как системы сбора информации без изменения конфигурации обнаруживаемых устройств.

7. Контейнерное развертывание

Эталонная docker-compose конфигурация серверной части включает следующие сервисы:

Сервис	Назначение в развертывании	Особенности
hub-postgres	База данных реестра и параметров системы.	Образ postgres:16-alpine; отдельный постоянный volume для данных; health-check pg_isready.
kafka	Брокер сообщений для агентных очередей.	Образ apache/kafka:3.9.0; конфигурация single-node для поставочного контура.
hub-api	Сервер управления агентами.	Собирается из Go исходного кода; публикует HTTP 8080 и gRPC 9090; запускает миграции БД при старте.
web	Веб-консоль управления агентами.	Собирается из Vue исходного кода; результат размещается в Nginx.
gateway	Единая входная точка для web и API.	Маршрутизирует /api/, /health, /docs, /openapi и web-запросы; в dev-конфигурации опубликован на порту 8090.

Сборка Go-сервиса выполняется в многостадийном Dockerfile: образ golang:1.26-alpine используется для компиляции статического бинарного файла, а runtime-слой основан на alpine:3.21. Веб-консоль собирается в Node.js 22 и публикуется из Nginx.

8. Технологический стек

Уровень	Используемые технологии
Серверная разработка	Go 1.26, gRPC, gRPC-Gateway, Protocol Buffers, pgx, sqlx, goose, gorm, amqp091-go, JWT, Argon2id, AES-GCM.
Веб-клиент	React 18, JavaScript/JSX, Redux Toolkit / RTK Query, React Router, Ant Design, xlsx, Nginx.
Данные и обмен	PostgreSQL 16, Apache Kafka 3.9, REST/JSON, gRPC, HTTP/HTTPS.
Инфраструктура	Docker, Docker Compose, Nginx, Alpine Linux, Node.js 22
Наблюдаемость	Health-check API, структурированное логирование, интеграция с Prometheus при включении.

9. Контакты разработчика

Разработчик: Компания «Скайфолл Лабс».

Официальный сайт: skyflabs.ru

Электронная почта службы поддержки: support@skyflabs.ru

Телефон: +7 (495) 109-47-17.