# Руководство по установке и интеграции

## Руководство по установке и интеграции

!!! note “Примечание” Данное Руководство включает разделы, описывающие установку и интеграцию системы с использованием CLI (командной строки) и специальных скриптов, а также необходимые для интеграции настройки в интегрируемых инструментах. Руководство также содержит описание требований к инфраструктуре и настройки мониторинга. ## Требования к инфраструктуре

### Требования к аппаратному обеспечению

Серверные компоненты Stingray должны быть установлены на выделенном сервере, предназначенном исключительно для эксплуатации серверных компонент Stingray.

Для работы системы обязательно наличие поддержки виртуализации, а именно необходим процессор с поддержкой технологии виртуализации Intel Virtualization Technology (VT, VT-x, vmx) или AMD Virtualization (AMD-V, SVM).

#### Минимальные технические характеристики серверного оборудования

* 6-ядерный CPU 2 ГГц.
* Оперативная память: 24 Гб.
* Свободное дисковое пространство 100 GB (+ пространство для размещения прикладных систем и баз данных).

#### Рекомендуемые технические характеристики серверного оборудования

* 8-ядерный CPU 3 ГГц.
* Оперативная память: 32 Гб.
* Свободное дисковое пространство 500 GB (+ пространство для размещения прикладных систем и баз данных).

Для развертывания платформы Stingray на базе клиентской инфраструктуры требуется следующая минимальная аппаратная или виртуальная конфигурация оборудования:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование | Кол-во | CPU | RAM, Гб | HDD, Гб |
| Stingray (минимальная конфигурация) | 1 | 6-ядерный процессор (2 ГГц) с поддержкой технологии виртуализации Intel Virtualization Technology (VT, VT-x, vmx) или AMD Virtualization (AMD-V, SVM) | 24 | 200 |
| Stingray (рекомендуемая конфигурация) | 1 | 8-ядерный процессор (3 ГГц) с поддержкой технологии виртуализации Intel Virtualization Technology (VT, VT-x, vmx) или AMD Virtualization (AMD-V, SVM) | 32 | 500 |

Минимальная конфигурация рассчитана из следующего количества сканирующих агентов:

* Два сканирующих агента Android.
* Два сканирующих агента iOS.

При увеличении количества сканирующих агентов необходимо пересмотреть конфигурацию оборудования из расчета 1 ядро и 4 Гб ОЗУ на каждого дополнительного сканирующего агента при их параллельной работе. Также можно исходить из таблицы ниже.

|  |  |  |  |
| --- | --- | --- | --- |
| Количество сканирующих агентов | Ядер процессора (физических) | Общее количество потоков | RAM, Гб |
| 2 Android, 2 iOS | 6\* | 12 | 24 |
| 4 Android, 4 iOS | 8\* | 16 | 24 |
| 6 Android, 6 iOS | 12\* | 24 | 32 |

\* При виртуализации количество vCPU должно соответствовать общему количеству потоков.

### Архитектура и ОС

Система поддерживает следующие типы ОС и ПО для полноценного функционирования.

|  |  |  |
| --- | --- | --- |
| Операционная система | Архитектура | Платформа |
| Linux | 64-bit | Ubuntu Server 18.04.6 x64 |
| Linux | 64-bit | Centos / RHEL 7 and higher |

### Сетевой доступ

Установленные для эксплуатации Stingray технические средства должны быть совместимы между собой и поддерживать сетевой протокол TCP/IP. Для первоначальной настройки и установки платформы Stingray и сопутствующих пакетов желателен доступ к следующим ресурсам:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | Адрес источника | Адрес приемника | Тип подключения | Порты | Назначение |
| 1 | Сетевой адрес Stingray | <https://download.docker.com> | Внешний | 80, 443 | Установка docker |
| 2 | Сетевой адрес Stingray | <http://archive.ubuntu.com><http://security.ubuntu.com> | Внешний | 80, 443 | Установка сопутствующих пакетов |
| 3 | Сетевой адрес Stingray | <https://cr.yandex/> | Внешний | 80, 443 | Авторизация в хранилище docker и загрузка docker-образов |
| 4 | Сетевой адрес Stingray | <https://container-registry.api.cloud.yandex.net> | Внешний | 80, 443 | Авторизация в хранилище docker и загрузка docker-образов |
| 5 | CI/CD система, в которой осуществляется процесс сборки приложения | Сетевой адрес Stingray | Внутренний | 80, 443 | Загрузка артефакта сборки (мобильного приложения) для анализа в Stingray |
| 6 | Сетевой адрес системы Stingray | Backend мобильного приложения | Внутренний | 80, 443 | Сетевая доступность backend для корректной работы мобильного приложения |
| 7 | Рабочее место пользователя Stingray | Сетевой адрес Stingray | Внутренний | 80, 443 | Работа пользователей с графическим интерфейсом системы |
| 8 | Рабочее место администратора Stingray | Сетевой адрес Stingray | Внутренний | 80, 443, 22 | Администрирование системы Stingray |

## Установка Stingray

!!! note “Примечание” Все действия, описанные в данном разделе, необходимо производить от пользователя root.

### Подготовка инфраструктуры

1. Необходимо убедиться, что CPU имеет поддержку технологии аппаратной виртуализации (Intel Virtualization Technology (VT, VT-x, vmx) или AMD Virtualization (AMD-V, SVM)), выполнив команды:

* === “Ubuntu” sudo apt-get install cpu-checker kvm-ok

1. Установить требуемые пакеты:

* === “Ubuntu Server 16” sudo apt-get install qemu-kvm libvirt-bin ubuntu-vm-builder bridge-utils === “Ubuntu Server 18/20” sudo apt install qemu-kvm libvirt-clients libvirt-daemon-system bridge-utils virt-manager === “RHEL/CentOS” sudo yum install qemu-kvm libvirt libvirt-python libguestfs-tools virt-install

1. Установите docker и docker-compose, если это не было сделано заранее. Рекомендации по установке можно найти на официальном сайте:
   * <https://docs.docker.com/install/linux/docker-ce/ubuntu/>
   * [https://docs.docker.com/compose/install/](https://docs.docker.com/install/linux/docker-ce/ubuntu/)
2. Создайте группы и пользователей.

* groupadd --system --gid 171 kvm  
   groupadd --gid 1717 emulator  
   useradd --uid 1717 --gid emulator --groups kvm emulator
* Если группа kvm уже существует, то вместо первой строки выполнить следующее:
* groupmod --gid 171 kvm  
   chgrp kvm /dev/kvm

#### Установка при наличии доступа к внешнему репозиторию docker-образов YCR

1. Авторизуйте docker на доступ к репозиторию YCR docker-образов:

* cat stingray-numeric\_id.json | docker login --username json\_key --password-stdin cr.yandex
* Ключ stingray-numeric\_id.json для подключения к Yandex Container Registry (YCR) docker-образам компании Stingray Technologies предоставляется при покупке лицензии Stingray.

1. Загрузите специальный docker-образ для подготовки конфигурационных файлов командой:

* docker pull cr.yandex/crp8p3a3l1ri2431n3ce/stingray/wizard:release-x
* !!! note “Примечание” Версия релиза указывается в формате release-x, где x — это текущая версия. Пожалуйста, уточняйте эту информацию у вендора или на официальном сайте.

#### Установка без наличия доступа к внешнему репозиторию docker-образов YCR

1. При отсутствии доступа к внешнему репозиторию docker-образов, образы поставляются в виде выгруженных tar-архивов. Для доступа к данным архивам необходимо запросить их у поставщика продукта.
2. После того, как архивы загружены и перенесены на сервер Stingray необходимо их импортировать в docker. Для этого выполните следующую команду для всех полученных архивов:

* docker load -i <archive\_name>.tar

### Настройка системы

1. Создайте директорию, где будут располагаться конфигурационные файлы Stingray, к примеру директорию /opt/stingray.
2. При необходимости использования HTTPS соединения создайте директорию, где будут располагаться файл полной цепочки сертификатов (например, /opt/certs) и закрытого ключа, скопируйте файлы сертификата и закрытого ключа и назовите их в соответствии с требованиями:
   * ***fullchain.pem*** — полная цепочка сертификатов;
   * ***privkey.pem*** — приватный ключ.
3. Запустите docker-контейнер для подготовки конфигурации.

* Пример запуска контейнера с двумя подключенными volumes для файлов конфигурации и с сертификатами (при доступе по HTTPS):
* docker run -i -t -v /opt/stingray:/opt/docker-files -v /opt/certs:/opt/nginx/certs cr.yandex/crp8p3a3l1ri2431n3ce/stingray/wizard:release-x
* !!! note “Примечание” Версия релиза указывается в формате release-x, где x — это текущая версия. Пожалуйста, уточняйте эту информацию у вендора или на официальном сайте.
* Пример запуска контейнера с одним volume для файлов конфигурации (при доступе по HTTP):
* docker run -i -t -v /opt/stingray:/opt/docker-files cr.yandex/crp8p3a3l1ri2431n3ce/stingray/wizard:release-x
* !!! note “Примечание” Версия релиза указывается в формате release-x, где x — это текущая версия. Пожалуйста, уточняйте эту информацию у вендора или на официальном сайте.

После запуска контейнера в интерактивном режиме необходимо заполнить ряд параметров.

|  |  |  |
| --- | --- | --- |
| Параметр | Описание | Значение по умолчанию |
| STINGRAY\_DOMAIN | Домен, на котором будет располагаться система для корректной настройки маршрутизации и обращения UI к нужному серверу | saas.stingray-mobile.ru |
| IP\_EXTERNAL | IP сервера, на который устанавливается Stingray | 0.0.0.0 |
| USE\_SSL | Параметр, определяющий, будет ли проходить соединение через протокол http или https. При указании «1» приложение конфигурируется для использования 443 порта и проводит настройку для соединения по HTTPS (копируются из второго volume с сертификатами) | 0 |
| POSTGRES\_USER | Пользователь, с которым будет запущена база данных Postgres | stingray |
| POSTGRES\_PASSWORD | Пароль пользователя, с которым будет запущена база данных Postgres | P@ssw0rd |
| RABBITMQ\_DEFAULT\_USER | Пользователь, с которым будет запущен брокер RabbitMQ | stingray |
| RABBITMQ\_DEFAULT\_PASS | Пароль пользователя, с которым будет запущен брокер RabbitMQ | P@ssw0rd |
| STINGRAY\_DEBUG | Запустить сервер с расширенным выводом ошибок | 0 |
| STINGRAY\_DOCKER\_LOGIN | Флаг, определяющий сетевую доступность от сервера Stingray до внешнего хранилища docker-образов. Данная функциональность необходима для динамического создания сканирующих агентов и загрузки последней актуальной версии при обновлении системы. При выставлении значения «0» — необходимо при обновлении системы вручную загрузить последнюю версию образа | 0 |
| STINGRAY\_LANGUAGE\_CODE | Язык системы по умолчанию. Влияет на язык swagger и язык по умолчанию для вновь создаваемых языков | ru |
| STINGRAY\_TIME\_ZONE | Временная зона для корректного отображения времени | Europe/Moscow |
| STINGRAY\_ACCESS\_TOKEN\_LIFETIME | Время жизни access\_token в минутах | 60 |
| STINGRAY\_REFRESH\_TOKEN\_LIFETIME | Время жизни refresh\_token в минутах | 1440 |
| STINGRAY\_CI\_TOKEN\_LIFETIME | Время жизни токена для интеграции в CI/CD | 525600 |
| STINGRAY\_COMPANY\_NAME | Название компании | Company Name |
| STINGRAY\_COMPANY\_DESCRIPTION | Описание компании | Company Description |
| STINGRAY\_SUPERUSER\_USERNAME | Имя пользователя с ролью Супер администратора | admin |
| STINGRAY\_SUPERUSER\_PASSWORD | Пароль пользователя с ролью Супер администратора | admin |
| STINGRAY\_CREATE\_SAMPLES | Флаг, определяющий, нужно ли создавать сущности по умолчанию, которые задаются далее (проекты/профили, пользователей, сканирующие агенты/компания) | 1 |
| STINGRAY\_ADMIN\_USERNAME | Имя пользователя с ролью Администратора компании | company\_admin |
| STINGRAY\_ADMIN\_PASSWORD | Пароль пользователя с ролью Администратора компании | 123 |
| STINGRAY\_ADMIN\_FIRSTNAME | Имя Администратора | FirstName |
| STINGRAY\_ADMIN\_LASTNAME | Фамилия Администратора | LastName |
| STINGRAY\_ENGINE\_NAME\_ANDROID | Имя агента для Android Engine | stingray-engine |
| STINGRAY\_ENGINE\_NAME\_IOS | Имя агента для iOS Engine | stingray-engine-ios |
| STINGRAY\_PROJECT\_NAME\_ANDROID | Имя проекта, создаваемого по умолчанию для Android проекта | Project Name Android |
| STINGRAY\_PROJECT\_DESCRIPTION\_ANDROID | Описание проекта, создаваемого по умолчанию для Android проекта | Project Description Android |
| STINGRAY\_PROJECT\_NAME\_IOS | Имя проекта, создаваемого по умолчанию для iOS проекта | Project Name iOS |
| STINGRAY\_PROJECT\_DESCRIPTION\_IOS | Описание проекта, создаваемого по умолчанию для iOS проекта | Project Description iOS |
| STINGRAY\_PROFILE\_NAME\_ANDROID | Имя профиля, создаваемого по умолчанию для Android проекта | Profile Name Android |
| STINGRAY\_PROFILE\_DESCRIPTION\_ANDROID | Описание профиля, создаваемого по умолчанию для Android проекта | Profile Description Android |
| STINGRAY\_PROFILE\_NAME\_IOS | Имя профиля, создаваемого по умолчанию для iOS проекта | Profile Name iOS |
| STINGRAY\_PROFILE\_DESCRIPTION\_IOS | Описание профиля, создаваемого по умолчанию для iOS проекта | Profile Description iOS |
| STINGRAY\_AUDIT\_USE | Включение или выключение аудита событий в системе | 1 |
| STINGRAY\_AUDIT\_MAX\_LENGTH | Максимальное количество записей в одном файле. После превышения заданного количества к концу файла добавляется постфикс (.1, .2 и т. д.), а новая информация записывается в стандартный файл без постфикса | 1000 |
| STINGRAY\_AUDIT\_FILE\_COUNT | Количество файлов, которое будет храниться в системе. При превышении количества файлов старые удаляются | 10 |

В результате выполнения в директории /opt/stingray будут созданы все необходимые файлы для запуска.

#### Список контейнеров

|  |  |
| --- | --- |
| Имя контейнера | Описание |
| stingray-nginx | Входная точка для обращений к backend и UI. Выполняет функции reverse-proxy |
| stingray-backend | Backend приложения, отвечает за основную логику обработки пользовательских запросов и выдачу результатов |
| stingray-rabbitmq | Менеджер очередей сканирования, управляет очередью сканирования |
| stingray-postgres | База данных |
| stingray-ui | Пользовательский интерфейс |
| stingray-redis | Redis для промежуточного хранения оперативной информации |
| engine-android | Сканирующий модуль для Android-проектов. Название контейнера может быть произвольным |
| engine-ios | Сканирующий модуль для iOS-проектов. Название контейнера может быть произвольным |
| stingray-knowledgebase | Контейнер, содержащий в себе информацию по устранению уязвимостей и документацию |
| stingray-maintenance | Осуществляет управление контейнерами - проверку статусов, перезагрузку, запуск и остановку |

## Запуск Stingray

!!! note “Примечание” В этом разделе описан процесс первого запуска версии системы Stingray. Процесс запуска Stingray после обновления версии описан в разделе «[Обновление системы](./obnovlenie_sistemy.md)». Процесс запуска Stingray после перезагрузки сервера без обновления версии Stingray описан в разделе «[Перезагрузка сервера без обновления Stingray](./perezagruzka_servera_bez_obnovleniya_stingray.md)».

Для запуска приложения в папке, указанной при генерации конфигурации (в примере ***/opt/stingray***) выполните команды:

docker-compose pull  
docker pull cr.yandex/crp8p3a3l1ri2431n3ce/stingray/android\_api27:release-x  
docker pull cr.yandex/crp8p3a3l1ri2431n3ce/stingray/android\_api30:release-x  
docker pull cr.yandex/crp8p3a3l1ri2431n3ce/stingray/ios:release-x  
docker-compose up -d

!!! note “Примечание” Версия релиза указывается в формате release-x, где x — это текущая версия. Пожалуйста, уточняйте эту информацию у вендора или на официальном сайте.

При первом запуске системы, в случае если команда docker-compose не может загрузить образ из репозитория, необходимо вручную загрузить контейнеры:

docker pull cr.yandex/crp8p3a3l1ri2431n3ce/stingray/stingray:release-x  
docker pull cr.yandex/crp8p3a3l1ri2431n3ce/stingray/android\_api27:release-x  
docker pull cr.yandex/crp8p3a3l1ri2431n3ce/stingray/android\_api30:release-x  
docker pull cr.yandex/crp8p3a3l1ri2431n3ce/stingray/ios:release-x  
docker pull cr.yandex/crp8p3a3l1ri2431n3ce/stingray/stingray-ui:release-x  
docker pull cr.yandex/crp8p3a3l1ri2431n3ce/stingray/stingray-knowledgebase:release-x

!!! note “Примечание” Версия релиза указывается в формате release-x, где x — это текущая версия. Пожалуйста, уточняйте эту информацию у вендора или на официальном сайте. ## Остановка Stingray

Для остановки приложения в папке, указанной при генерации конфигурации (в примере ***/opt/stingray***) выполните команду:

docker exec stingray-maintenance django-admin maintenance engines preserve  
docker-compose stop

!!! note “Примечание” Команда preserve сохраняет состояние контейнеров для дальнейшего их восстановления в ранее сохраненном состоянии. ## Обновление системы

### Обновление при наличии доступа к внешнему репозиторию docker-образов YCR

1. Остановите Stingray согласно инструкциям в разделе «[Остановка Stingray](./ostanovka_stingray.md)».
2. Обновите специальный docker-образ для подготовки конфигурационных файлов командой:

* docker pull cr.yandex/crp8p3a3l1ri2431n3ce/stingray/wizard:release-x
* !!! note “Примечание” Версия релиза указывается в формате release-x, где x — это текущая версия. Пожалуйста, уточняйте эту информацию у вендора или на официальном сайте.

1. Запустите docker-контейнер с параметром update.

* docker run -i -t -v /opt/stingray:/opt/docker-files cr.yandex/crp8p3a3l1ri2431n3ce/stingray/wizard:release-x update
* !!! note “Примечание” Версия релиза указывается в формате release-x, где x — это текущая версия. Пожалуйста, уточняйте эту информацию у вендора или на официальном сайте.

1. После завершения копирования новых конфигурационных файлов необходимо выполнить команду обновления образов из директории с конфигурационными файлами (в примере /opt/stingray):

* docker-compose pull  
   docker pull cr.yandex/crp8p3a3l1ri2431n3ce/stingray/android\_api27:release-x  
   docker pull cr.yandex/crp8p3a3l1ri2431n3ce/stingray/android\_api30:release-x  
   docker pull cr.yandex/crp8p3a3l1ri2431n3ce/stingray/ios:release-x  
   docker-compose up -d  
   docker exec stingray-maintenance django-admin maintenance engines recreate
* !!! note “Примечание” Команда recreate пересоздает контейнеры в их ранее сохраненном состоянии, используя новые версии образов.
* !!! note “Примечание” Версия релиза указывается в формате release-x, где x — это текущая версия. Пожалуйста, уточняйте эту информацию у вендора или на официальном сайте.
* !!! note “Примечание” При скачивании нового образа старый образ не удаляется. Чтобы накопившиеся старые образы не занимали много места, рекомендуется их удалять, например, с помощью следующих команд:
* docker image prune  
    
   Эта команда удалит все docker образы без тегов (у которых тег `<none>`). Следует учитывать, что она не удалит образы с предыдущими версиями. Например, если была установлена версия Stingray 2.7, а вместо нее поставили новую версию 2022.X, то старые образы не будут удалены, так как тег у старого образа будет 2.7, а не `<none>`.  
    
   docker image prune -a  
    
   Эта команда удалит docker образы без тегов (у которых тег <none>) и docker образы, которые не используются ни одним контейнером. Но в случае, если, например, ещё ни один engine контейнер для какого-нибудь нового образа не создавался (а такое может быть, например, если версия для iOS ещё не использовалась), то эта команда удалит соответствующий образ. Далее, когда возникнет необходимость создать контейнер из этого образа, то это сделать уже не удастся, так как такого образа уже не будет.  
    
   docker image rm image\_id  
    
   Эта команда предназначена для индивидуального удаления образов.

1. В случае возникновения ошибок возможна загрузка образов вручную:

* docker pull cr.yandex/crp8p3a3l1ri2431n3ce/stingray/stingray:release-x  
   docker pull cr.yandex/crp8p3a3l1ri2431n3ce/stingray/android\_api27:release-x  
   docker pull cr.yandex/crp8p3a3l1ri2431n3ce/stingray/android\_api30:release-x  
   docker pull cr.yandex/crp8p3a3l1ri2431n3ce/stingray/ios:release-x  
   docker pull cr.yandex/crp8p3a3l1ri2431n3ce/stingray/stingray-ui:release-x  
   docker pull cr.yandex/crp8p3a3l1ri2431n3ce/stingray/stingray-knowledgebase:release-x
* !!! note “Примечание” Версия релиза указывается в формате release-x, где x — это текущая версия. Пожалуйста, уточняйте эту информацию у вендора или на официальном сайте.
* После загрузки образов запустите систему согласно инструкциям в предыдущем пункте данного раздела.

1. Если осуществляется переход с версии Stingray 2.х на версию Stingray 2022.X, для корректной работы вновь установленной версии необходимо однократное выполнение команды:

* docker exec stingray-maintenance django-admin maintenance engines fill\_id
* Эта команда обеспечивает корректное взаимодействие всех компонентов системы после обновления версии. Повторное выполнение этой команды не имеет смысла, но при этом Stingray продолжит корректно функционировать.

### Обновление при отсутствии доступа к внешнему репозиторию docker-образов YCR

1. Остановите Stingray согласно инструкциям в разделе «[Остановка Stingray](./ostanovka_stingray.md)».
2. При отсутствии доступа к внешнему репозиторию docker-образов, образы поставляются в виде выгруженных tar-архивов. Для доступа к данным архивам необходимо запросить их у поставщика продукта.
3. После того, как архивы загружены и перенесены на сервер Stingray необходимо их импортировать в docker. Для этого выполните следующую команду для всех полученных архивов:

* docker load -i <archive\_name>.tar

1. Запустите специальный конфигуратор (Wizard) с параметром update.

* docker run -i -t -v /opt/stingray-docker-compose:/opt/docker-files cr.yandex/crp8p3a3l1ri2431n3ce/stingray/wizard:release-x update
* !!! note “Примечание” Версия релиза указывается в формате release-x, где x — это текущая версия. Пожалуйста, уточняйте эту информацию у вендора или на официальном сайте.

1. После загрузки образов запустите систему согласно инструкциям в разделе в пунктах 4 и 6 раздела «[Обновление при наличии доступа к внешнему репозиторию docker-образов YCR](#docker-gcp)». ## Перезагрузка сервера без обновления Stingray

В случае, если необходимо перезагрузить сервер, на котором установлен Stingray, например, для обновления операционной системы, без обновления версии Stingray, перед перезагрузкой сервера в папке, указанной при генерации конфигурации (в примере /opt/stingray), выполните команду:

docker exec stingray-maintenance django-admin maintenance engines preserve

После перезагрузки сервера там же выполните команду:

docker exec stingray-maintenance django-admin maintenance engines restore

. ## Интеграции системы ### Интеграции через REST API

Для интеграции с системой посредством API были созданы средства инструментальной поддержки, позволяющие значительно упростить процесс интеграции.

В частности, существует и доступен по ссылке: <https://github.com/Dynamic-Mobile-Security/mdast-cli> скрипт для работы с системой, написанный на языке программирования Python.

Данный скрипт предназначен для встраивания анализа безопасности мобильных приложений в непрерывный процесс разработки (CI/CD) и предоставляет полноценный интерфейс для запуска сканирований и интеграций с различными системами дистрибуции. Более подробная информация о данном скрипте приведена в разделе «[Системы CI/CD](./sistemy_ci_cd.md)».

Для работы с API Stingray существует PyPi пакет mdast\_cli\_core, ознакомиться с ним можно по ссылке: <https://github.com/Dynamic-Mobile-Security/mdast-cli-core>.

### Системы CI/CD

Для встраивания в процесс CI/CD предусмотрен специальный [скрипт](https://github.com/Dynamic-Mobile-Security/mdast-cli), в процессе выполнения которого собранное приложение отправляется в систему для анализа. На выходе формируется JSON файл с подробными результатами.

#### Варианты установки

##### DockerHub

Можно установить пакет, используя docker image:

docker pull mobilesecurity/mdast\_cli:release-version

!!! note “Примечание” Версия релиза указывается в формате release-version, где release-version — это текущая версия. Пожалуйста, уточняйте эту информацию у вендора, на официальном сайте или на [странице в Docker Hub](https://hub.docker.com/repository/docker/mobilesecurity/mdast_cli).

##### Пакетный менеджер pip

Возможно установить пакет, используя pip:

pip install mdast\_cli

При таком способе возможно запускать сканирование без указания интерпретатора python при помощи команды mdast\_cli, пример:

mdast\_cli -h

Во всех примерах ниже будет использован именно такой подход.

##### Исходный код

Также поддерживается запуск при помощи загрузки исходных файлов и запуска непосредственно основного скрипта:

python3 mdast\_cli/mdast\_scan.py -h

При таком способе запуска необходимо дополнительно установить пакеты, указанные в файле requirements.txt.

#### Варианты запуска

На данный момент поддерживается несколько вариантов запуска:

* анализ приложения, файл которого расположенного локально;
* анализ приложений из [Google Play](https://play.google.com/store/apps);
* анализ приложений из [Appstore](https://www.apple.com/app-store/);
* анализ приложений из [Firebase](https://firebase.google.com/);
* анализ приложений из системы [AppCenter](https://appcenter.ms/);
* анализ приложений из системы [Nexus Repository 3.x](https://help.sonatype.com/repomanager3).

#### Параметры запуска

Параметры запуска зависят от расположения файла приложения, отправляемого на анализ. Так же, существуют обязательные параметры, которые необходимо указывать при любом виде запуска:

* url — сетевой адрес Stingray (путь до корня без последнего «/»), при использовании cloud версии — <https://saas.stingray-mobile.ru>;
* profile\_id — id профиля, для которого проводится анализ;
* testcase\_id — id того тест-кейса, который будет воспроизведен во время анализа; возможен запуск нескольких тест-кейсов, для этого их id перечисляются через пробел. Это необязательный параметр, если он не задан, то будет запущено сканирование в ручном режиме и через 30 секунд после запуска остановлено, а данные отправлены на анализ;
* token — CI/CD токен для доступа, более подробная информация приведена в разделе «[Интеграции](./integracii.md)» Руководства пользователя;
* distribution\_system — способ загрузки приложения, возможные опции: file, google\_play, appstore, firebase, appcenter, nexus. Более подробно про них описано ниже в соответствующих разделах;
* company\_id — идентификатор компании, в рамках которой будет осуществлено сканирование;
* architecture\_id — опциональный параметр. Определяет идентификатор архитектуры операционной системы, на которой будет произведено сканирование;
* nowait — опциональный параметр, определяющий необходимость ожидания завершения сканирования. Если данный флаг установлен — скрипт не будет дожидаться завершения сканирования, а выйдет сразу же после запуска. Если флаг не установлен — скрипт будет ожидать завершения процесса анализа и формировать отчет;
* summary\_report\_json\_file\_name — опциональный параметр. Определяет имя JSON файла, в который выгружается информация по сканированию в формате JSON. При отсутствии параметра информация сохраняться в JSON не будет;
* pdf\_report\_file\_name — опциональный параметр. Определяет имя PDF файла в который выгружается информация по сканированию в формате PDF. При отсутствии параметра PDF-отчет сохраняться не будет.

#### Локальный запуск

Данный вид запуска подразумевает, что файл приложения для анализа располагается локально, рядом (на одной системе) со скриптом. Для выбора этого способа при запуске необходимо указать параметр distribution\_system file. В этом случае обязательным параметром необходимо указать путь к файлу file\_path.

##### Для запуска анализа локального файла

mdast\_cli --distribution\_system file \  
 --file\_path "/files/demo/apk/demo.apk" \  
 --url "https://saas.stingray-mobile.ru" \  
 --profile\_id 1 \  
 --testcase\_id 4 \  
 --company\_id 1   
 --architecture\_id 1 \  
 --token "token\_value"

В результате будет запущен автоматизированный анализ приложения demo.apk с профилем с id 1 и будет запущен тест-кейс с id 4.

##### Запуск без ожидания завершения сканирования

mdast\_cli --distribution\_system file \  
 --file\_path "/files/demo/apk/demo.apk" \   
 --url "https://saas.stingray-mobile.ru" \  
 --profile\_id 1 \  
 --testcase\_id 4 \  
 --company\_id 1 \   
 --architecture\_id 1 \  
 --token "token\_value" \  
 --nowait

В результате будет запущен автоматизированный анализ приложения demo.apk с профилем id 1 (тест-кейс id 4). Сразу после этого скрипт завершится и отчет генерироваться не будет.

##### Генерация Summary отчета в формате JSON

mdast\_cli --distribution\_system file \  
 --file\_path "/files/demo/apk/demo.apk" \   
 --url "https://saas.stingray-mobile.ru" \  
 --profile\_id 1 \  
 --testcase\_id 4 \  
 --company\_id 1 \  
 --architecture\_id 1 \  
 --token "token\_value" \  
 --summary\_report\_json\_file\_name json-scan-repot.json

В результате будет запущен автоматизированный анализ приложения demo.apk с профилем id 1 (тест-кейс id 4) и по завершении сканирования будет выгружен JSON-отчет с суммарным количеством дефектов и краткой статистикой по сканированию. ### Система дистрибуции AppCenter

#### Параметры запуска

Параметры запуска зависят от расположения файла apk, отправляемого на анализ. Также существуют обязательные параметры, которые необходимо указывать при любом виде запуска:

* url — сетевой адрес Stingray (путь до корня без последнего «/»), при использовании cloud версии — <https://saas.stingray-mobile.ru>;
* profile\_id — id профиля, для которого проводится анализ;
* testcase\_id — id того тест-кейса, который будет воспроизведен во время анализа; возможен запуск нескольких тест-кейсов, для этого их id перечисляются через пробел. Это необязательный параметр, если он не задан, то будет запущено сканирование в ручном режиме и через 20 секунд после запуска остановлено, а данные отправлены на анализ;
* token — CI/CD токен для доступа, более подробная информация приведена в разделе «[Интеграции](./integracii.md)» Руководства пользователя;
* distribution\_system — способ загрузки приложения, возможные опции: file, google\_play, appstore, firebase, appcenter, nexus;
* company\_id — идентификатор компании, в рамках которой будет осуществлено сканирование;
* architecture\_id — опциональный параметр. Определяет идентификатор архитектуры операционной системы, на которой будет произведено сканирование;
* nowait — опциональный параметр, определяющий необходимость ожидания завершения сканирования. Если данный флаг установлен — скрипт не будет дожидаться завершения сканирования, а выйдет сразу же после запуска. Если флаг не установлен — скрипт будет ожидать завершения процесса анализа и формировать отчет;
* summary\_report\_json\_file\_name — опциональный параметр. Определяет имя JSON файла, в который выгружается информация по сканированию в формате JSON. При отсутствии параметра информация сохраняться в JSON не будет;
* pdf\_report\_file\_name — опциональный параметр. Определяет имя PDF файла в который выгружается информация по сканированию в формате PDF. При отсутствии параметра PDF-отчет сохраняться не будет.

Для загрузки приложения из системы дистрибуции AppCenter при запуске необходимо указать параметр distribution\_system appcenter. Также необходимо указать обязательные параметры:

* appcenter\_token — API токен для доступа. Как его получить, можно узнать [здесь](https://docs.microsoft.com/en-us/appcenter/api-docs/);
* appcenter\_owner\_name — владелец приложения. Как узнать имя владельца, можно прочитать [здесь](https://intercom.help/appcenter/en/articles/1764707-how-to-find-the-app-name-and-owner-name-from-your-app-url) или в [официальной документации](https://docs.microsoft.com/en-us/appcenter/api-docs/#find-your-app-center-app-name-and-owner-name);
* appcenter\_app\_name — имя приложения в системе AppCenter. Как его узнать, можно прочитать [здесь](https://docs.microsoft.com/en-us/appcenter/api-docs/#find-your-app-center-app-name-and-owner-name);
  + appcenter\_release\_id или appcenter\_app\_version:
  + appcenter\_release\_id — идентификатор загружаемого релиза в системе AppCenter для конкретного приложения. Возможно выставить значение latest — тогда будет загружен последний доступный релиз приложения ([официальная документация](https://openapi.appcenter.ms/#/distribute/releases_getLatestByUser));
  + appcenter\_app\_version — при указании данного параметра будет найдена и скачана конкретная версия приложения по коду его версии (указанной в Android Manifest) (поле version в [документации](https://openapi.appcenter.ms/#/distribute/releases_list)).

#### Примеры запуска

##### AppCenter по id релиза

Для запуска анализа приложения по известному имени, владельцу и ID релиза необходимо выполнить следующую команду:

mdast\_cli --distribution\_system appcenter --appcenter\_token 18bc81146d374ba4b1182ed65e0b3aaa   
 --appcenter\_owner\_name test\_org\_or\_user --appcenter\_app\_name demo\_app --appcenter\_release\_id 710   
 --url "https://saas.stingray-mobile.ru" --profile\_id 2 --testcase\_id 3 --company\_id 1 --architecture\_id 1  
 --token "eyJ0eXA4OiJKA1QiLbJhcGciO5JIU4I1NiJ1..."

В результате у владельца (пользователя или организации test\_org\_or\_user) будет найдено приложение demo\_app с ID релиза 710. Данная версия релиза будет загружена и передана на анализ безопасности в Stingray.

Для загрузки релиза с последней версией необходимо указать параметр appcenter\_release\_id latest. Тогда команда будет выглядеть следующим образом:

mdast\_cli --distribution\_system appcenter --appcenter\_token 18bc81146d374ba4b1182ed65e0b3aaa   
 --appcenter\_owner\_name "test\_org\_or\_user" --appcenter\_app\_name "demo\_app" --appcenter\_release\_id latest   
 --url "https://saas.stingray-mobile.ru" --profile\_id 2 --testcase\_id 3 --company\_id 1 --architecture\_id 1  
 --token "eyJ0eXA4OiJKA1QiLbJhcGciO5JIU4I1NiJ1..."

и будет загружен последний доступный релиз для данного приложения.

##### AppCenter по версии приложения

Для запуска анализа приложения по известному имени, владельцу и версии приложения (version\_code в Android Manifest) необходимо выполнить следующую команду:

mdast\_cli --distribution\_system appcenter --appcenter\_token 18bc81146d374ba4b1182ed65e0b3aaa   
 --appcenter\_owner\_name "test\_org\_or\_user" --appcenter\_app\_name "demo\_app" --appcenter\_app\_version 31337   
 --url "https://saas.stingray-mobile.ru" --profile\_id 2 --testcase\_id 3 --company\_id 1 --architecture\_id 1   
 --token "eyJ0eXA4OiJKA1QiLbJhcGciO5JIU4I1NiJ1..."

В результате у владельца (пользователя или организации test\_org\_or\_user) будет найдено приложение demo\_app и найден релиз, в котором была указана версия приложения 31337. Данная версия релиза будет загружена и передана на анализ безопасности в Stingray. ### Система дистрибуции Nexus Repository

#### Параметры запуска

Параметры запуска зависят от расположения файла apk, отправляемого на анализ. Так же, существуют обязательные параметры, которые необходимо указывать при любом виде запуска:

* url — сетевой адрес Stingray (путь до корня без последнего «/»), при использовании cloud версии — <https://saas.stingray-mobile.ru>;
* profile\_id — id профиля, для которого проводится анализ;
* testcase\_id — id того тест-кейса, который будет воспроизведен во время анализа; возможен запуск нескольких тест-кейсов, для этого их id перечисляются через пробел. Это необязательный параметр, если он не задан, то будет запущено сканирование в ручном режиме и через 20 секунд после запуска остановлено, а данные отправлены на анализ;
* token — CI/CD токен для доступа, более подробная информация приведена в разделе «[Интеграции](./integracii.md)» Руководства пользователя;
* distribution\_system — способ загрузки приложения, возможные опции: file, google\_play, appstore, firebase, appcenter, nexus. Более подробно про них описано ниже в соответствующих разделах;
* company\_id — идентификатор компании, в рамках которой будет осуществлено сканирование;
* architecture\_id — опциональный параметр. Определяет идентификатор архитектуры операционной системы, на которой будет произведено сканирование;
* nowait — опциональный параметр, определяющий необходимость ожидания завершения сканирования. Если данный флаг установлен — скрипт не будет дожидаться завершения сканирования, а выйдет сразу же после запуска. Если флаг не установлен — скрипт будет ожидать завершения процесса анализа и формировать отчет;
* summary\_report\_json\_file\_name — опциональный параметр. Определяет имя JSON файла, в который выгружается информация по сканированию в формате JSON. При отсутствии параметра информация сохраняться в JSON не будет;
* pdf\_report\_file\_name — опциональный параметр. Определяет имя PDF файла в который выгружается информация по сканированию в формате PDF. При отсутствии параметра PDF-отчет сохраняться не будет.

Чтобы скачать приложение с maven-репозитория необходимо знать, где оно находится, а также его group\_id, artifact\_id и version. Для загрузки мобильного приложения в Nexus можно использовать сниппеты для Android-пакетов (\*.apk) или для iOS (\*.ipa). Необходимо в качестве агрумента параметра distribution\_system указать nexus, а также использовать следующие обязательные параметры:

* nexus\_url — URL сервера Nexus, на котором находится мобильное приложение;
* nexus\_login — имя пользователя Nexus с правами доступа к репозиторию, в котором находится мобильное приложение;
* nexus\_password — пароль учетной записи Nexus;
* nexus\_repo\_name — имя Nexus-репозитория, в котором находится мобильное приложение;
* nexus\_group\_id — group\_id мобильного приложения, загруженного с maven;
* nexus\_artifact\_id — artifact\_id мобильного приложения, загруженного с maven;
* nexus\_version — версия мобильного приложения, загруженного с maven. ### Интеграция с Firebase

#### Сбор необходимых параметров

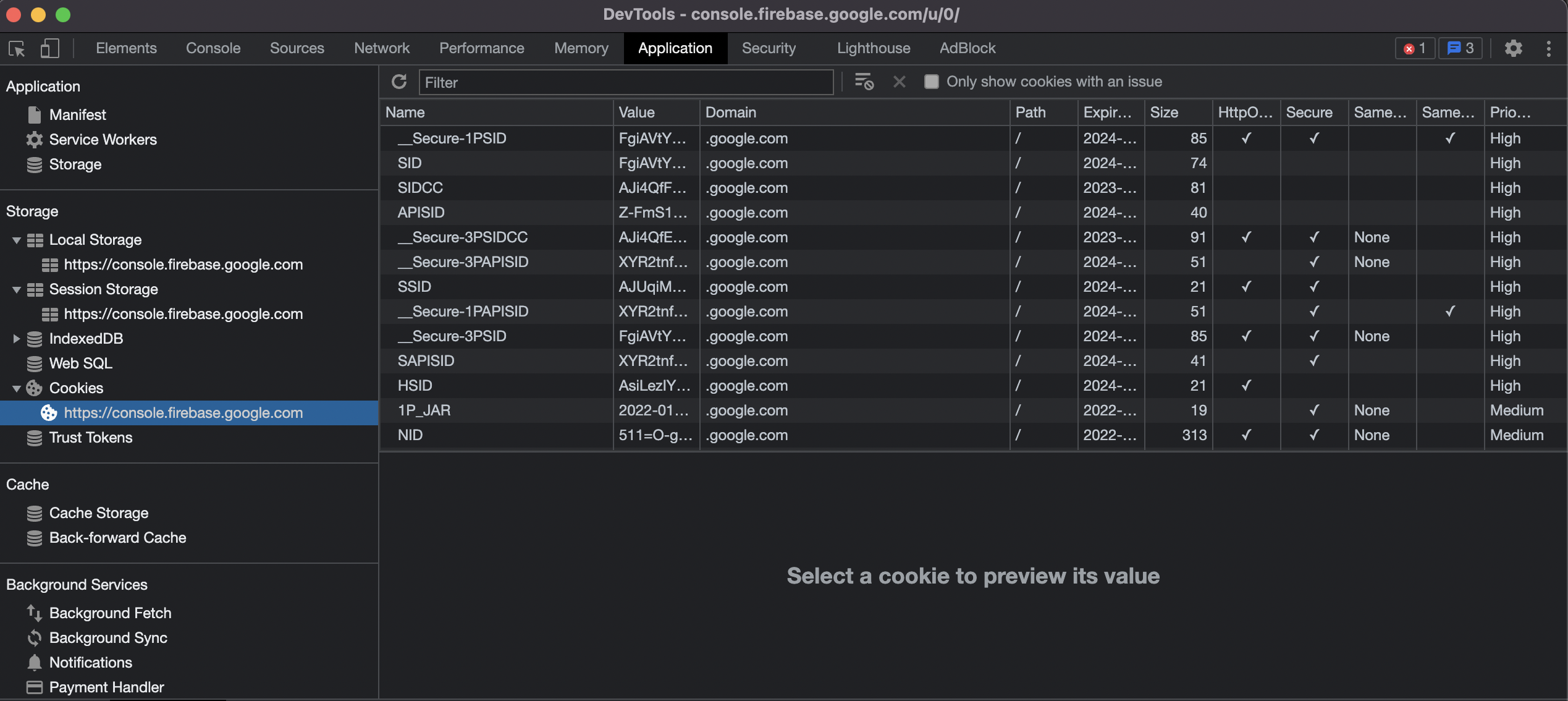
Система Stingray предлагает возможность интеграции с популярным набором сервисов [Firebase](https://firebase.google.com/) от компании Google, который используется на всех этапах жизненного цикла разработки программного обеспечения для мобильных устройств.

Интеграция с Firebase производится с помощью скрипта [mdast\_scan.py](https://github.com/Dynamic-Mobile-Security/mdast-cli). Для запуска скрипта в режиме интеграции с Firebase необходимо указать параметр --distribution\_system firebase.

Чтобы скачать приложение с Firebase и просканировать его, необходимо собрать информацию о значениях cookies для Google SSO аутентификации, а также о значениях параметров проекта в Firebase.

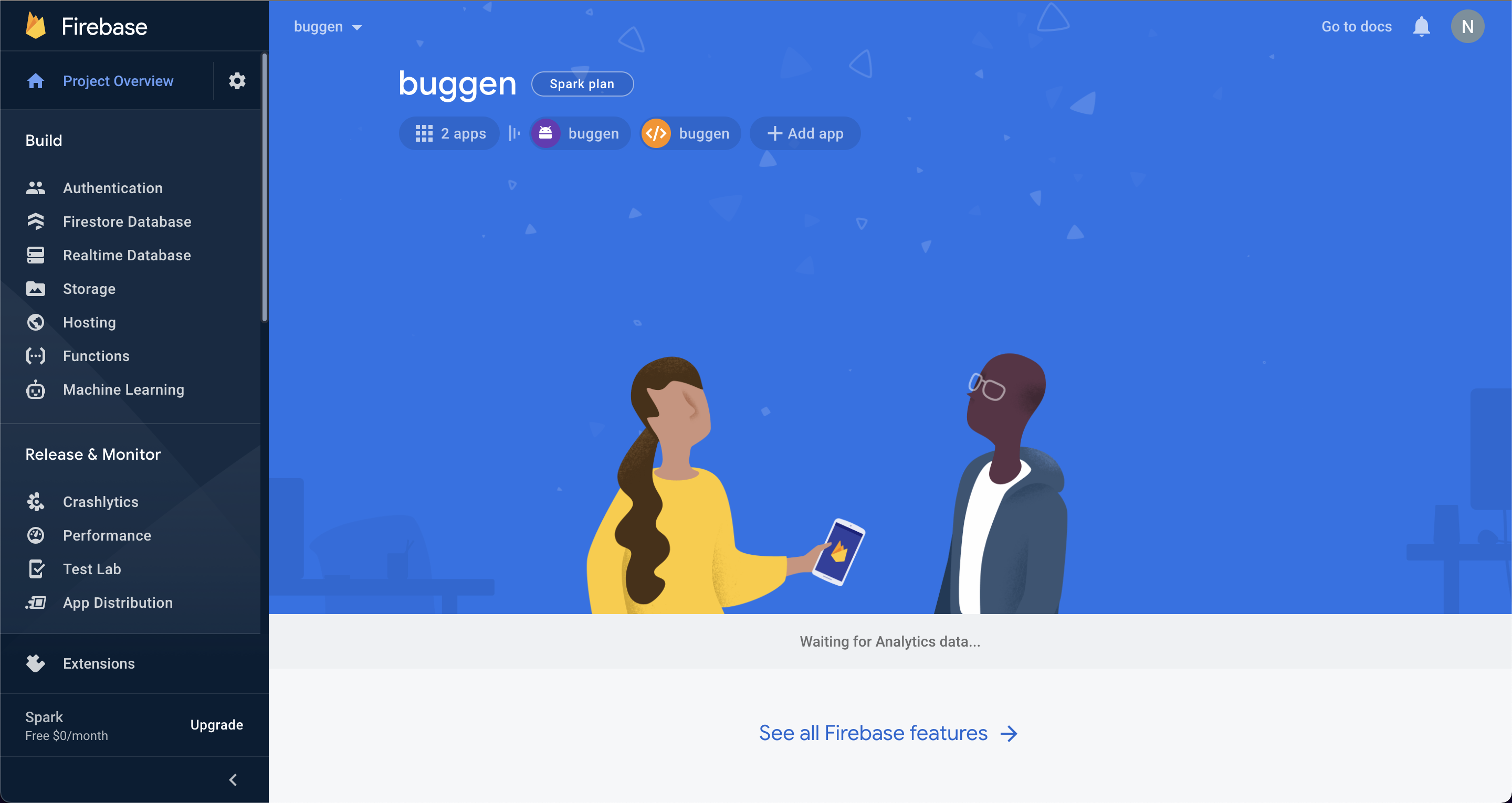
!!! note “Примечание” Не используйте личные учетные записи, так как параметры, используемые в процессе аутентификации в Firebase, привязаны к вашему аккаунту. Мы рекомендуем создать специальную техническую учетную запись для интеграции.

В браузере авторизуйтесь в Firebase с использованием Google SSO и соберите информацию о неоходимых cookies. Например, при использовании браузера Google Chrome или Microsoft Edge откройте DevTools, нажав F12 или использовав специальную комбинацию клавиш, которая применяется для этого в вашем браузере, перейдите на вкладку **Application**, а затем, выбрав в меню слева пункт **Cookies**, скопируйте значения следующих cookies: **SID**, **SSID**, **APISID**, **SAPISID** и **HSID**. Данная информация впоследствии будет использоваться в качестве параметров запуска скрипта [mdast\_scan.py](https://github.com/Dynamic-Mobile-Security/mdast-cli).

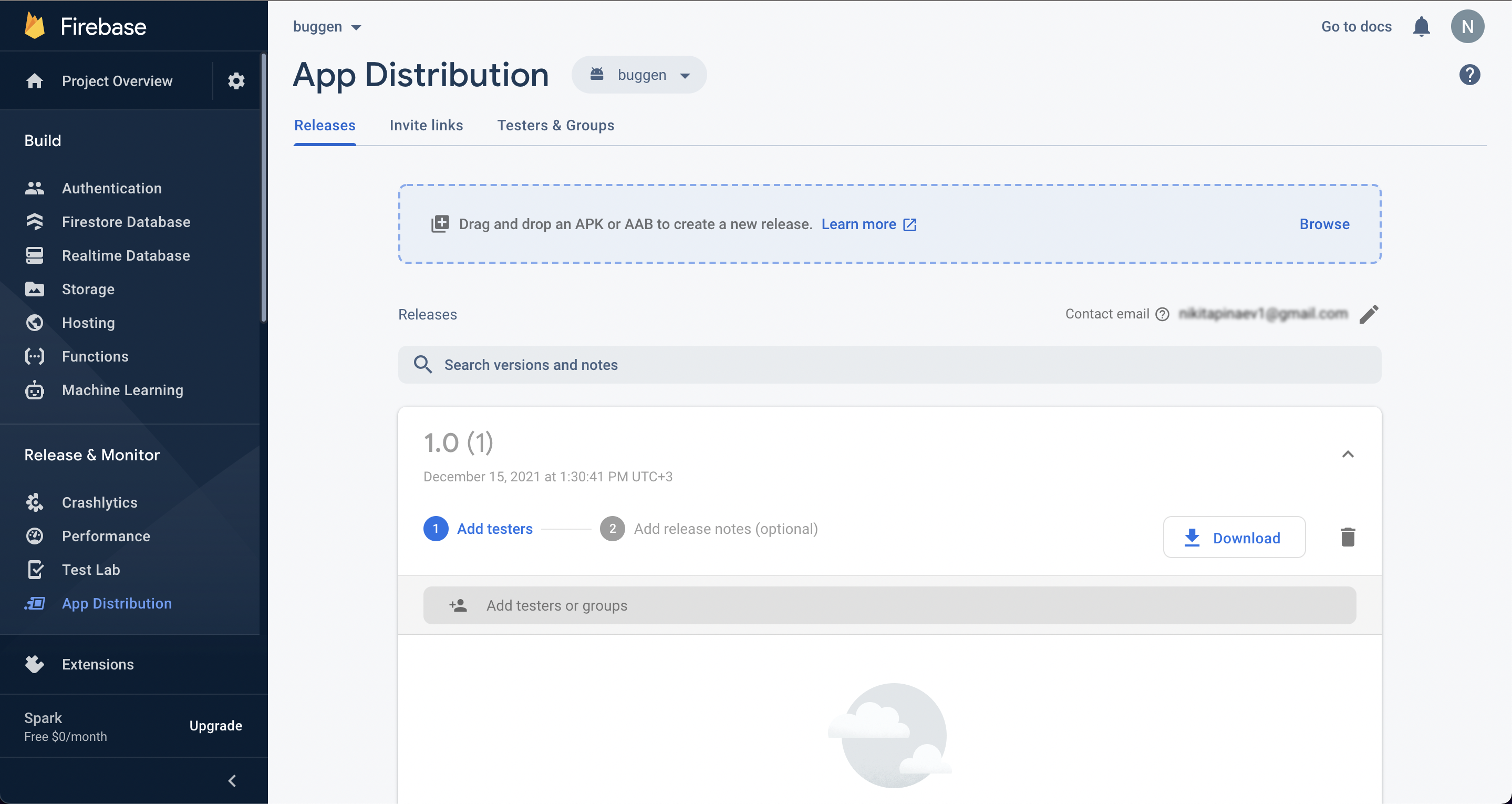


Далее необходимо получить следующие параметры проекта: **project\_id**, **app\_id**, **app\_code** и **api\_key**. Для этого перейдите на страницу проекта, адрес которой имеет следующий вид:

https://console.firebase.google.com/u/0/project/{project\_id}/overview

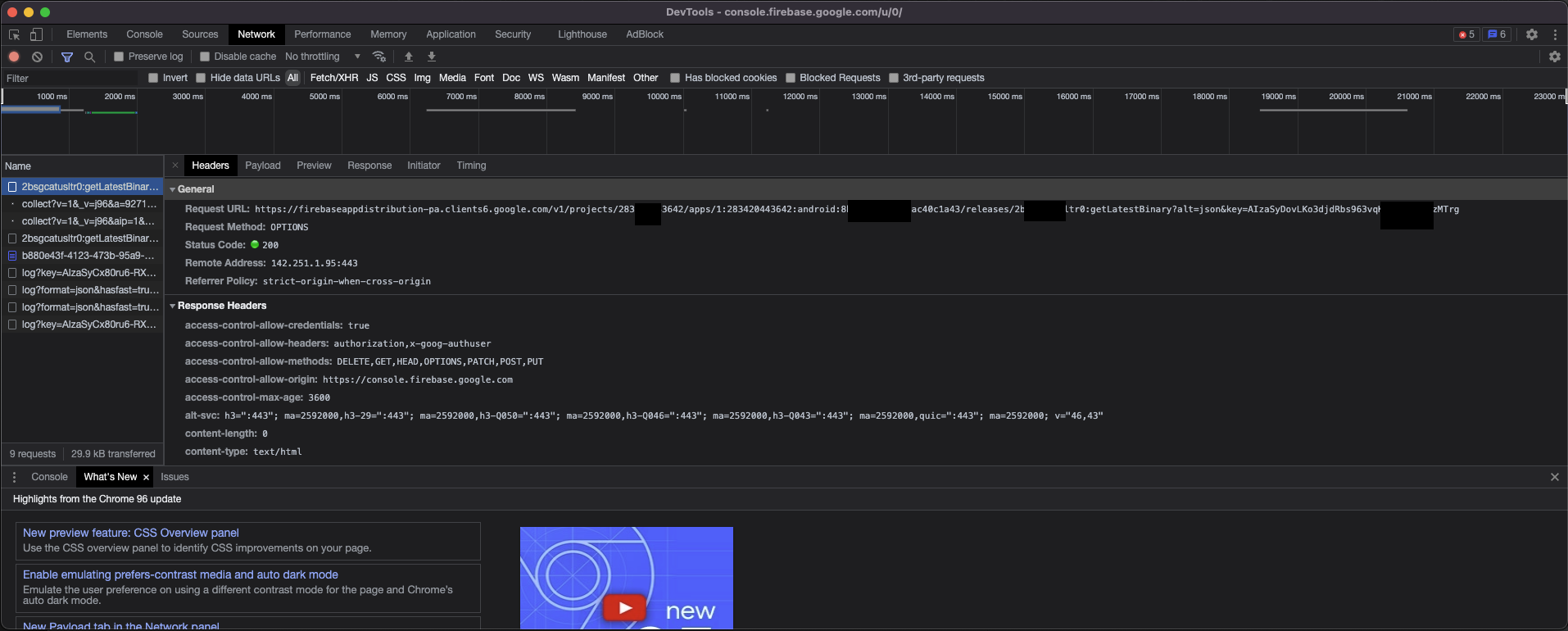


Слева в меню в разделе **Release & Monitor** выберите пункт **App Distribution**. Предварительно открыв DevTools (F12), выберите необходимый релиз из списка и нажмите кнопку **Download**.



В окне DevTools на вкладке **Network** отследите запрос, содержащий URL следующего вида:

https://firebaseappdistribution-pa.clients6.google.com/v1/projects/{project\_id}/apps/{app\_id}/releases/{app\_code}:getLatestBinary?alt=json&key={api\_key}



Из данного URL получаем остальные необходимые параметры.

В результате все необходимые параметры для запуска скрипта [mdast\_scan.py](https://github.com/Dynamic-Mobile-Security/mdast-cli) собраны. В зависимости от того, какое приложение скачивается, необходимо с помощью параметра firebase\_app\_extension указать расширение файла: для Android приложения — apk , а для iOS — ipa.

Приведем полный список собранных параметров:

* firebase\_SID\_cookie — SID;
* firebase\_HSID\_cookie — HSID;
* firebase\_SSID\_cookie — SSID;
* firebase\_APISID\_cookie — APISID;
* firebase\_SAPISID\_cookie — SAPISID;
* firebase\_project\_id — {project id};
* firebase\_app\_id — {application id};
* firebase\_app\_code — {application code};
* firebase\_api\_key — {api key}.
* firebase\_app\_extension — расширение файла: apk для Android или ipa для iOS.

Также возможно задать название скачиваемого файла, для этого надо задать опциональный параметр firebase\_file\_name.

#### Пример запуска скрипта

Чтобы запустить сканирование приложения, скачанного с Firebase, необходимо выполнить следующую команду:

python mdast\_cli/mdast\_scan.py \  
 --profile\_id 468 \  
 --architecture\_id 2 \  
 --distribution\_system firebase \  
 --firebase\_project\_id 2834204\*\*\*\* \  
 --firebase\_app\_id 1:283\*\*\*3642:android:8b0a0\*\*\*56ac40c1a43 \  
 --firebase\_app\_code 2b\*\*\*sltr0 \  
 --firebase\_api\_key AIzaSyDov\*\*\*\*\*qKdbj-geRWyzMTrg \  
 --firebase\_SID\_cookie FgiA\*\*\*\*\*ZiQakQ-\_C-5ZaEHvbDMFGkrgriAByQ9P9fv7LfRrYJ5suXgrCwIQBoOjA. \  
 --firebase\_HSID\_cookie AsiL\*\*\*\*OjPI \  
 --firebase\_SSID\_cookie A\*\*\*\*dwcZk1Z-1pE \  
 --firebase\_APISID\_cookie Z-FmS1aPB\*\*\*\*djK/AjmG0h2Hc-GG9g2Ac \  
 --firebase\_SAPISID\_cookie XYR2tnf\*\*\*\*0zOt/AEvVZ8JVEuCnE6pxm \  
 --url "https://saas.mobile.appsec.world" \  
 --company\_id 1 \   
 --token 2fac9652a2fbe4\*\*\*\*9f44af59c3381772f \  
 --firebase\_file\_name your\_app\_file\_name \  
 --firebase\_file\_extension apk

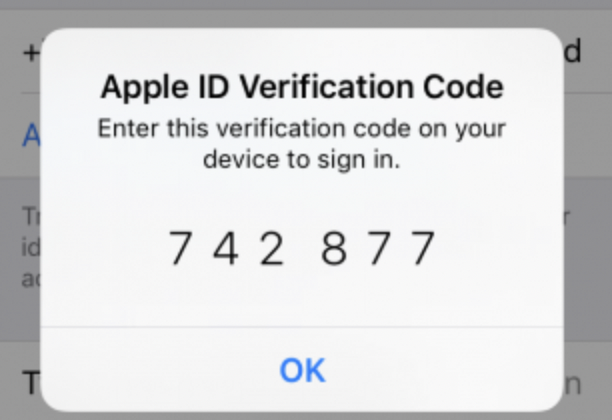
. ### Интеграция с AppStore

#### Сбор необходимых параметров

Интеграция с AppStore производится с помощью скрипта [mdast\_scan.py](https://github.com/Dynamic-Mobile-Security/mdast-cli). Чтобы скачать приложение с AppStore, необходимо располагать учетными данными аккаунта iTunes.

Для успешной авторизации с использованием аккаунта iTunes необходимо получить и сохранить код двухфакторной аутентификации.

При первом запуске скрипта с указанием адреса электронной почты и пароля вы столкнетесь с ошибкой авторизации (это ожидаемое и нормальное в данном случае поведение), при этом на ваше устройство придет код двухфакторной аутентификации.



Необходимо сохранить полученный код. Теперь при запуске скрипта пароль и код двухфакторной аутентификации необходимо передавать в следующем виде:

--appstore\_password2FA password2FA, где:

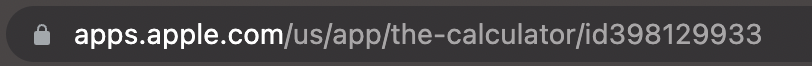
* password — пароль;
* 2FA — код двухфакторной аутентификации.

Например, если для входа в аккаунт используется пароль P@ssword, а полученный код двухфакторной аутентификации — 742877, передаваемый параметр будет выглядеть следующим образом: --appstore\_password2FA P@ssword742877.

Повторное получение кода двухфакторной аутентификации не потребуется в течение следующих 6 месяцев.

Для запуска сканирования, необходимо указать или appstore\_bundle\_id или appstore\_app\_id.

Чтобы получить параметр appstore\_app\_id, в браузере перейдите на страницу приложения в AppStore и скопируйте параметр из URL.



В данном примере appstore\_app\_id — 398129933.

При запуске скрипта указываются следующие обязательные параметры:

* distribution\_system appstore — признак того, что приложение скачивается из AppStore;
* appstore\_bundle\_id или appstore\_app\_id;
  + appstore\_bundle\_id — бандл имя приложения в AppStore;
  + appstore\_app\_id — идентификатор приложения в AppStore, который можно получить из ссылки приложения в магазине AppStore, взяв числовое значение id нужного приложения: apps.apple.com/app/id**{appstore\_app\_id}**;
* appstore\_apple\_id — электронная почта аккаунта iTunes;
* appstore\_password2FA — пароль и код двухфакторной аутентификации аккаунта iTunes.

Также используя параметр appstore\_file\_name, можно указать имя, с которым файл будет сохранен (данный параметр не является обязательным).

!!! note “Примечание” Если вы авторизовались и потеряли код двухфакторной аутентификации, сессия будет оставаться активной в течение одного дня (достаточно appstore\_app\_id и пароля). Вы не сможете завершить сессию с помощью данного скрипта. Таким образом, чтобы скрипт продолжал корректно работать, необходимо вновь авторизоваться спустя 24 часа с использованием пароля и кода двухфакторной аутентификации.

!!! note “Примечание” Если при запуске сканирования возникает ошибка, связанная с неправильным apple\_id, например: «Вход в App Store». Чтобы открыть «Telegram», войдите с Apple ID, с которым была произведена покупка», обратитесь в службу технической поддержки компании Stingray для согласования Apple ID, с помощью которого будет выполняться интеграция с AppStore, вам будет предложено решение этой проблемы.

#### Пример запуска скрипта

Чтобы запустить сканирование приложения, скачанного с AppStore, необходимо выполнить следующую команду:

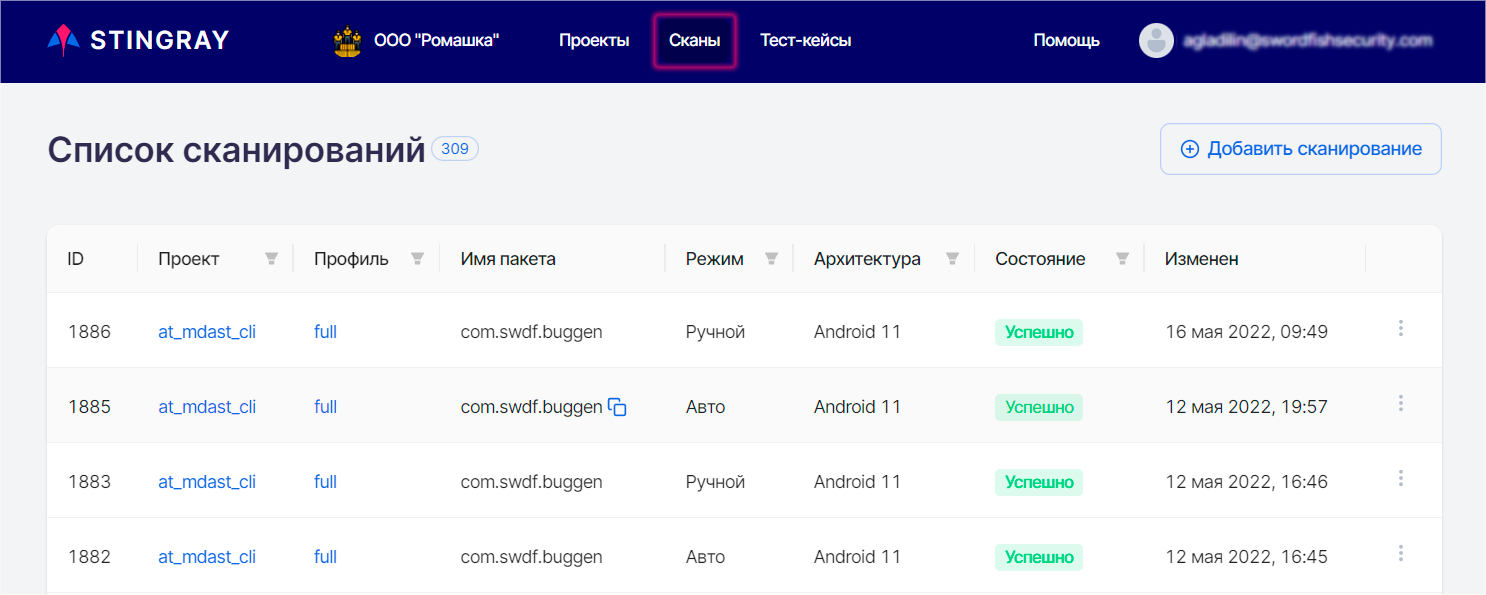
python mdast\_cli/mdast\_scan.py \  
 --architecture\_id 3 \  
 --profile\_id 1246 \  
 --distribution\_system appstore \  
 --appstore\_app\_id 564177498 \  
 --appstore\_apple\_id ubet\*\*\*\*\*\*@icloud.com \  
 --appstore\_password2FA pass\*\*\*\*\*\*\*31\*\*\*\*\*\*454 \  
 --url "https://saas.mobile.appsec.world" \  
 --company\_id 2 \  
 --token 5d5f6\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*2d9f --appstore\_file\_name my\_b3st\_4pp

. ### Интеграция c Burp Suite

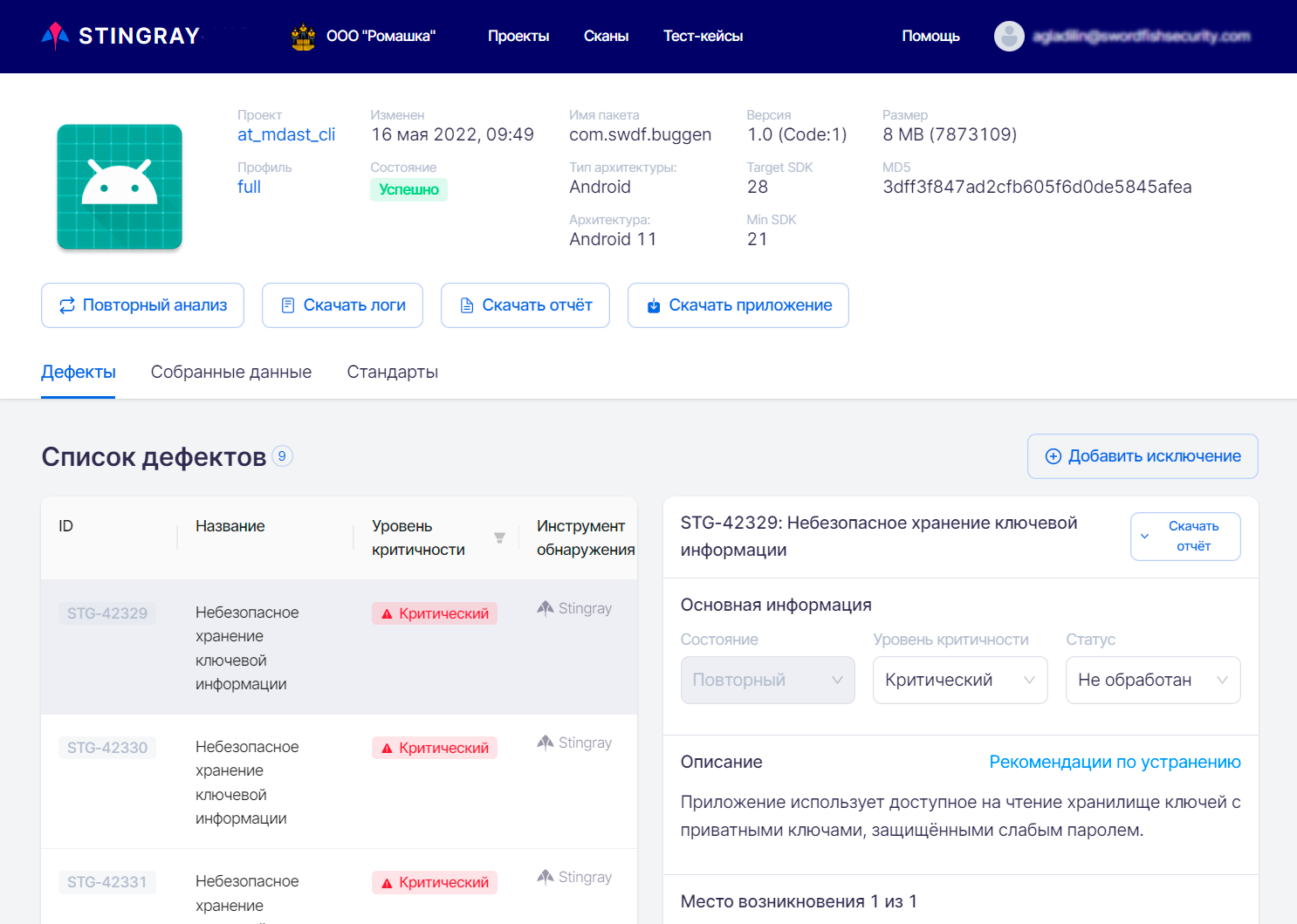
Stingray обеспечивает возможность передачи данных в различные инструменты анализа защищенности приложений, включая очень популярный и универсальный — [Burp Suite](https://portswigger.net/burp). Такая интеграция призвана облегчить более тщательный анализ результатов сканирования.

#### Получение результатов работы модуля Сетевая активность

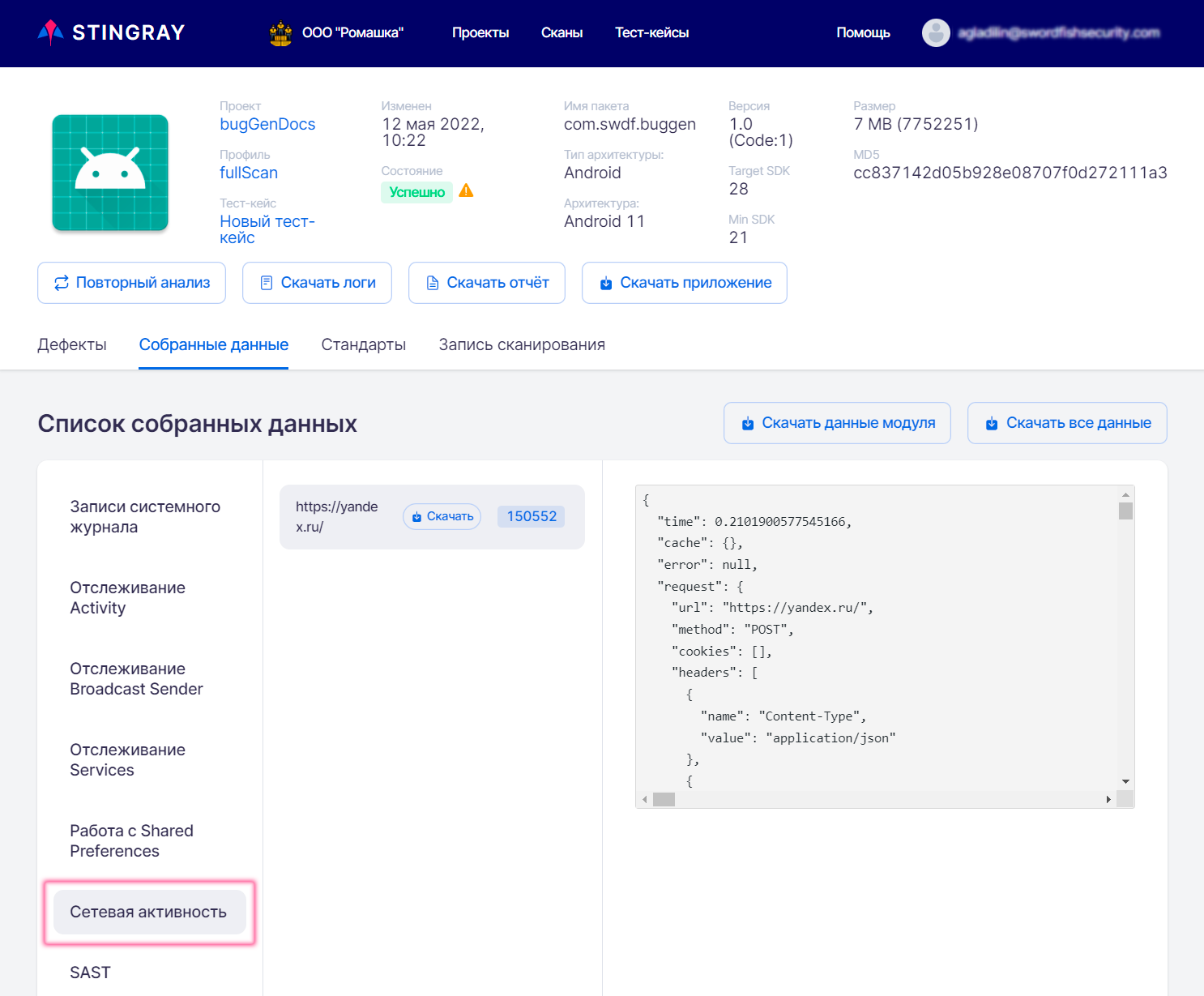
Чтобы передать собранные в результате сканирования данные в Burp Suite, необходимо перейти на страницу **Сканы**.



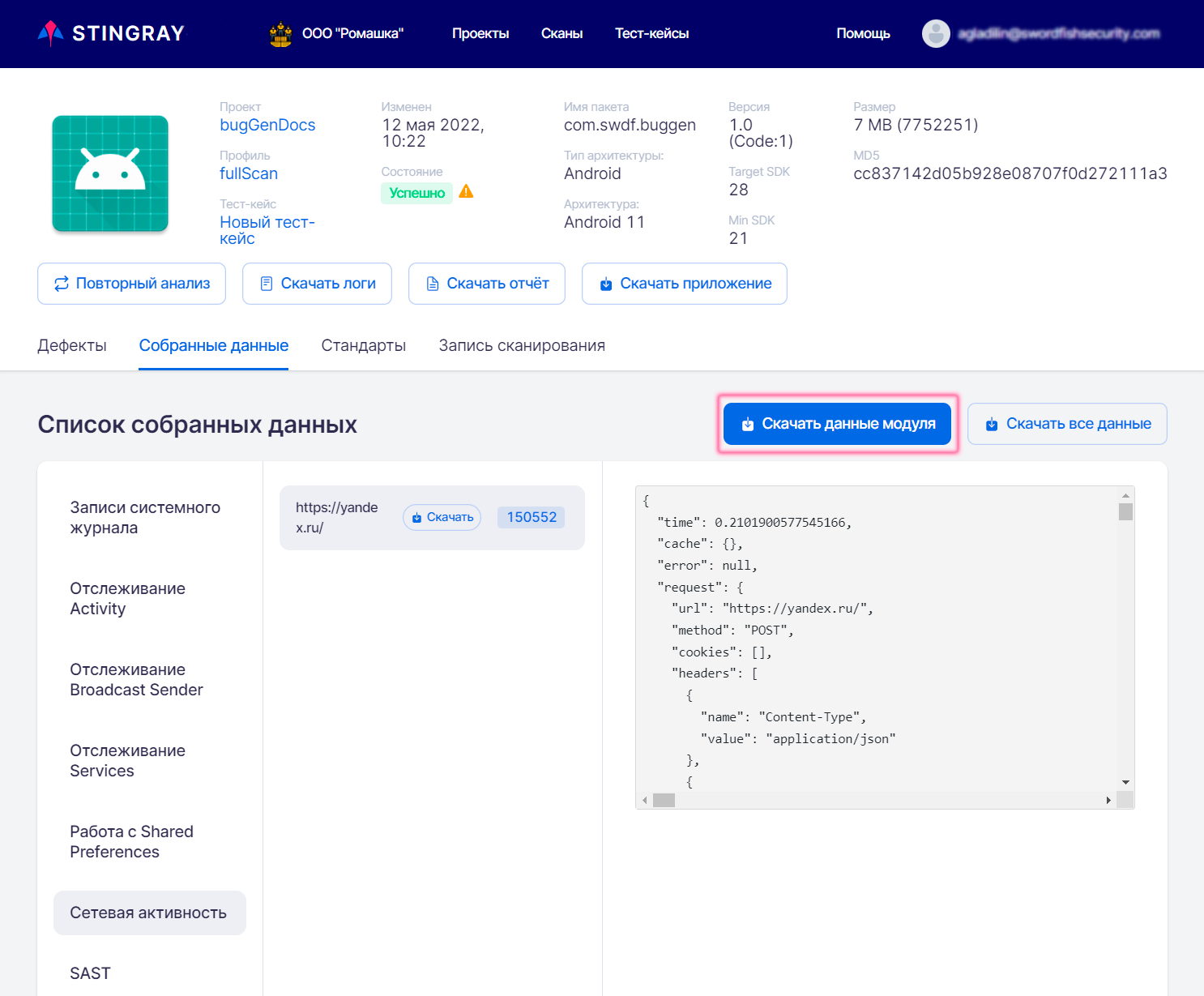
Нажав на **ID** соответствующего сканирования, перейти на страницу с результатами.



Перейти на вкладку **Собранные данные** и выберите модуль **Сетевая активность**.



Нажмите кнопку **Скачать данные модуля**, чтобы скачать zip-архив с результатами работы модуля.



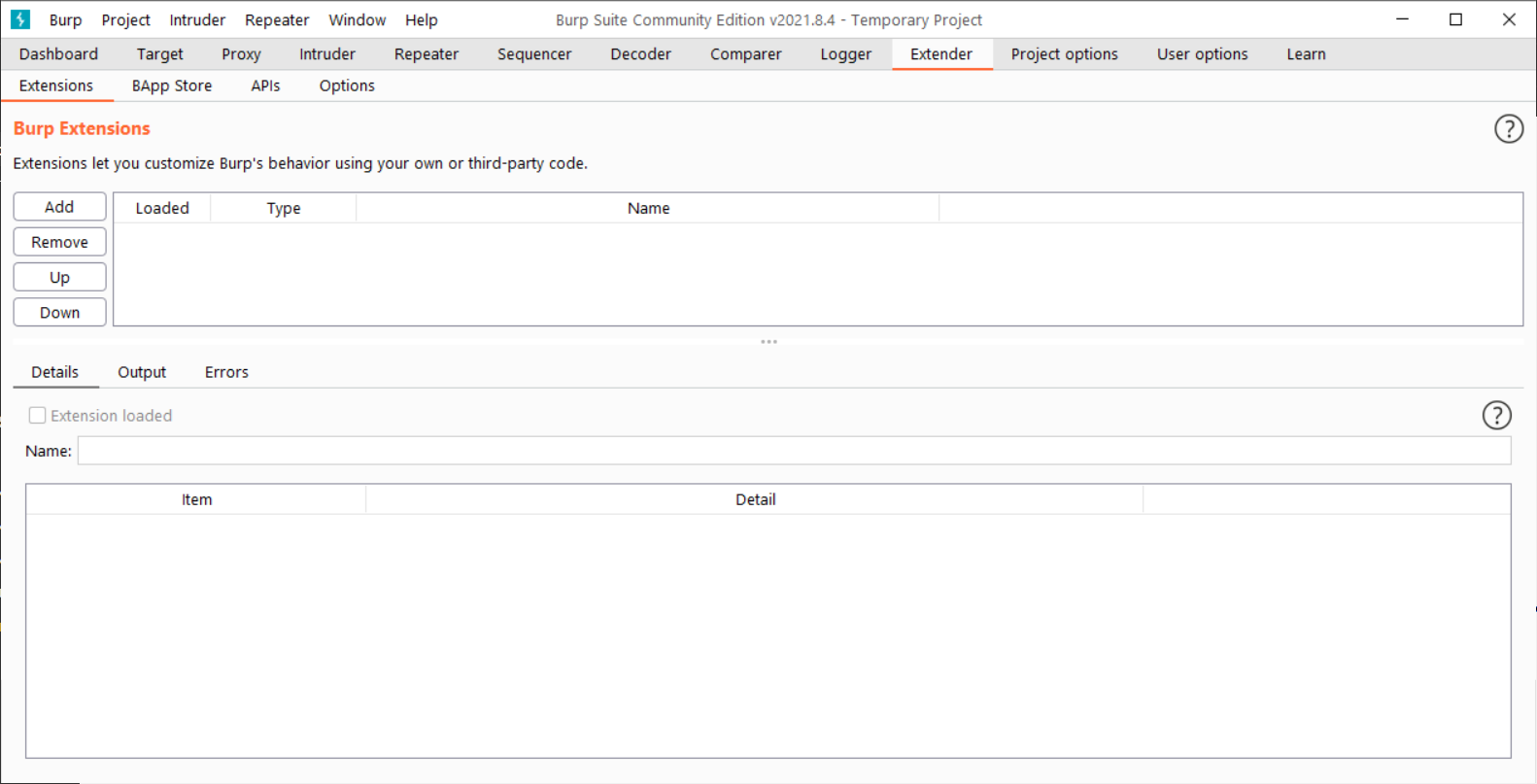
Скачанный архив содержит три файла в различных форматах.

#### Импорт данных в Burp Suite

Для импорта данных в Burp Suite необходимо использовать специальный плагин [burp-har-loader](https://github.com/Dynamic-Mobile-Security/burp-har-importer).

Скачайте плагин и запустите Burp Suite.

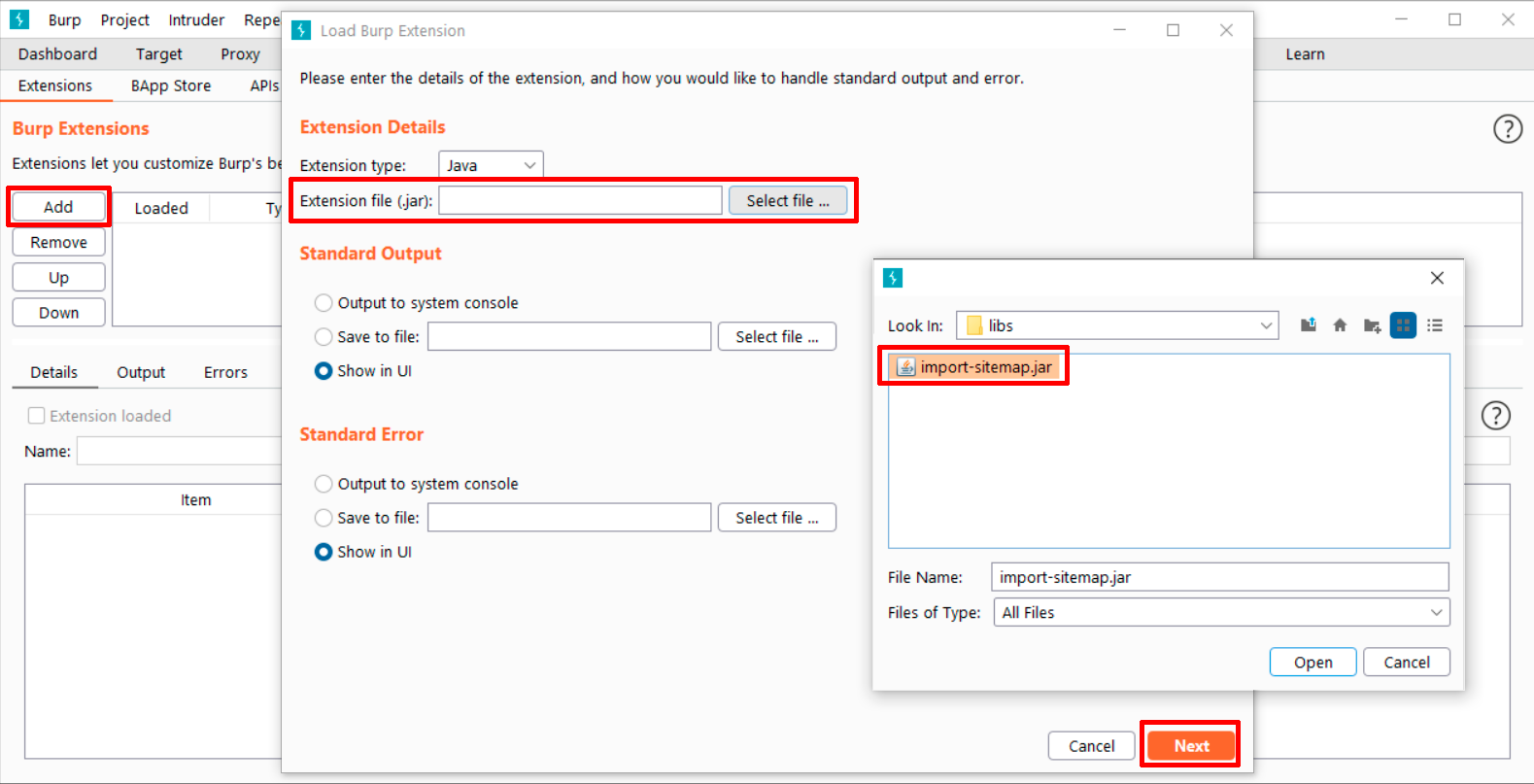
Перейдите на страницу **Extender**.



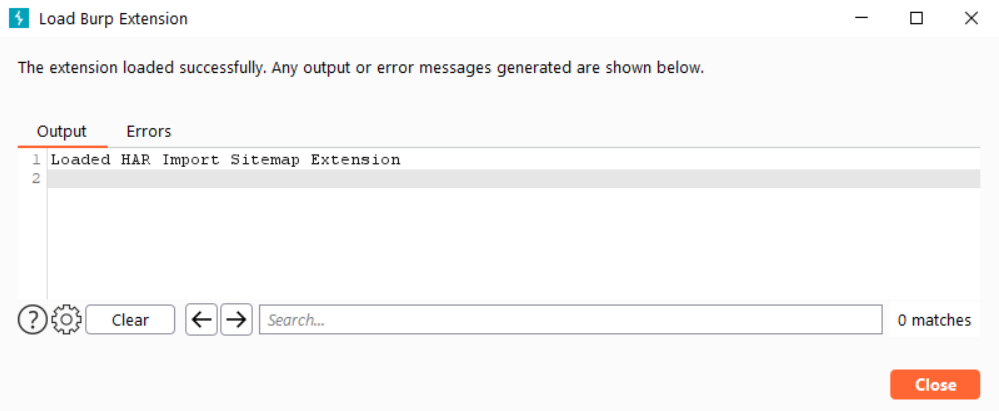
Нажав кнопку **Add**, откройте окно добавления плагина в Burp Suite.

В поле **Extension file (.jar)** укажите файл плагина.

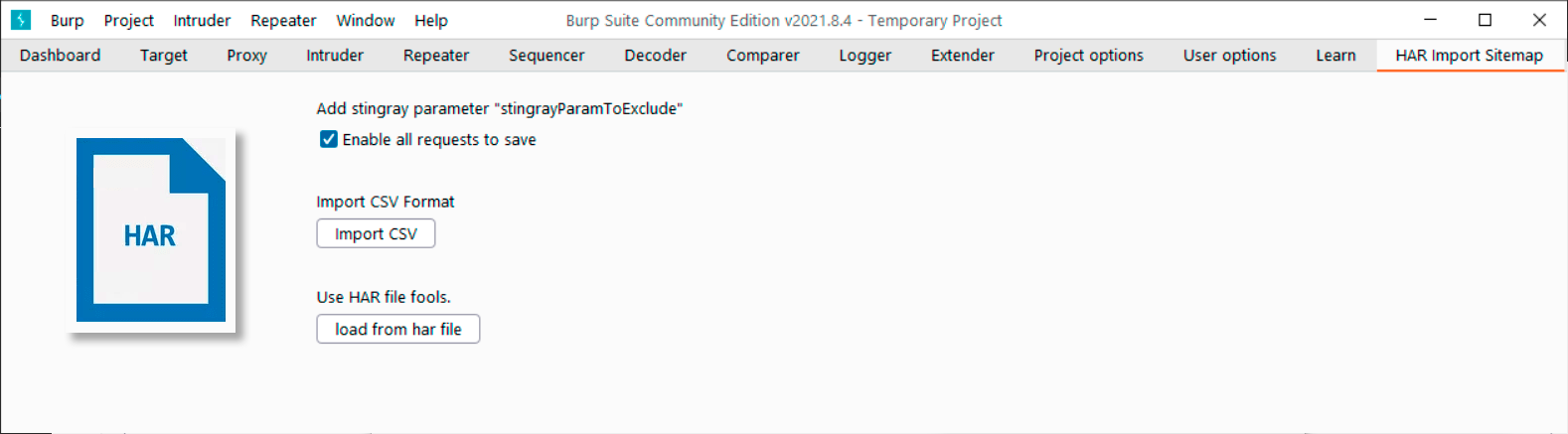
Нажмите кнопку **Next**, чтобы перейти в следующее диалоговое окно.



Убедившись, что в следующем окне отображается необходимый плагин, нажмите кнопку **Close**.

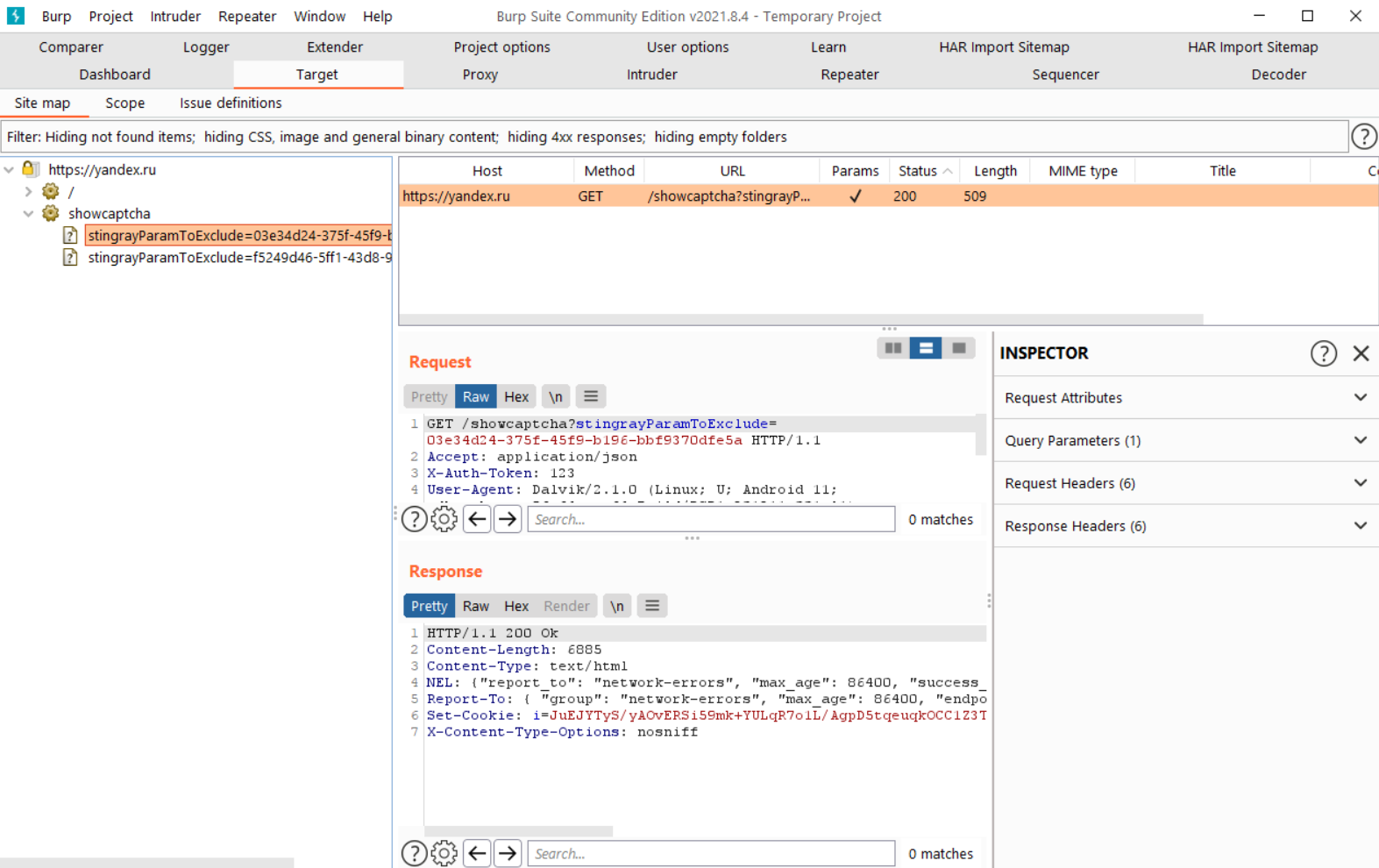


В результате успешного добавления плагина в строке меню Burp Suite появится новый пункт **HAR Import Sitemap**.



Нажав пункт меню **HAR Import Sitemap**, переходим на страницу плагина. Реализована возможность импорта данных в двух форматах: csv и har. Процесс экспорта данных из Stingray подробно описан выше в разделе «[Получение результатов работы модуля Сетевая активность](#Xdd76b4e93e1d1e1eefc363df7625b205566acf0)».

Выполнив импорт в удобном формате, можно перейти на вкладку **Target**, где теперь импортированы в Sitemap все запросы, которые были собраны во время работы приложения. Таким образом, можно запустить автоматическое сканирование или проанализировать вручную.



### Интеграция с Netsparker

Stingray обеспечивает возможность передачи данных в различные инструменты анализа защищенности приложений, включая популярный инструмент динамического и интерактивного анализа — [Netsparker](https://www.netsparker.com/). Такая интеграция позволяет расширить список ссылок, анализируемых Netsparker, и существенно повысить качество результатов последующего сканирования.

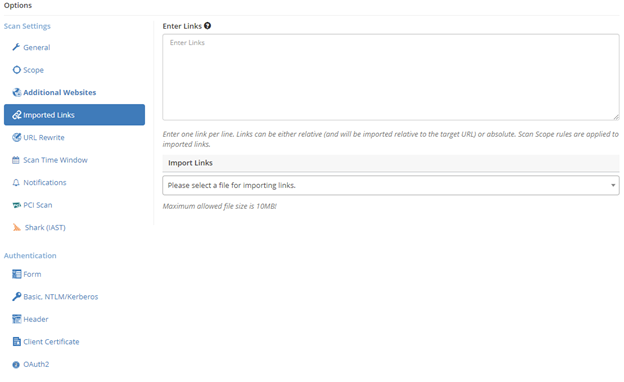
Передача данных осуществляется в формате har. Чтобы передать данные в Netsparker, необходимо предварительно в Stingray получить результаты работы модуля **Сетевая активность**. Данный процесс подробно описан в разделе «[Интеграция c Burp Suite](#c-burp-suite) / [Получение результатов работы модуля Сетевая активность](#Xdd76b4e93e1d1e1eefc363df7625b205566acf0)».

Скачанный архив содержит три файла в различных форматах, включая har — его мы и будем импортировать в Netsparker.

#### Импорт данных в Netsparker

В главном меню Netsparker выберите **Scans** > **New Scan**.

На странице **New Scan** выберите **Imported Links**.

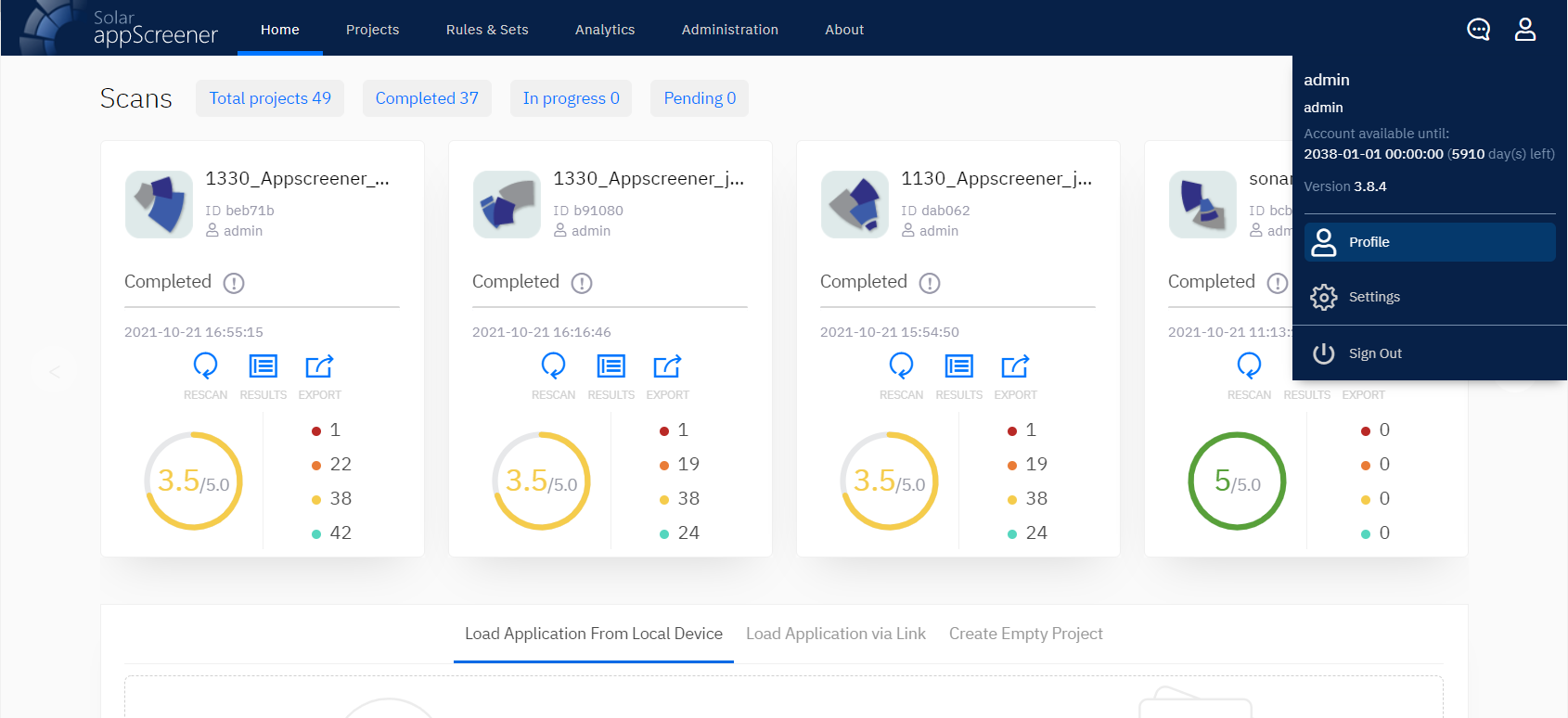


Укажите импортируемый файл в поле **Import Links**. ### Интеграция с AppScreener

#### Создание токена AppScreener

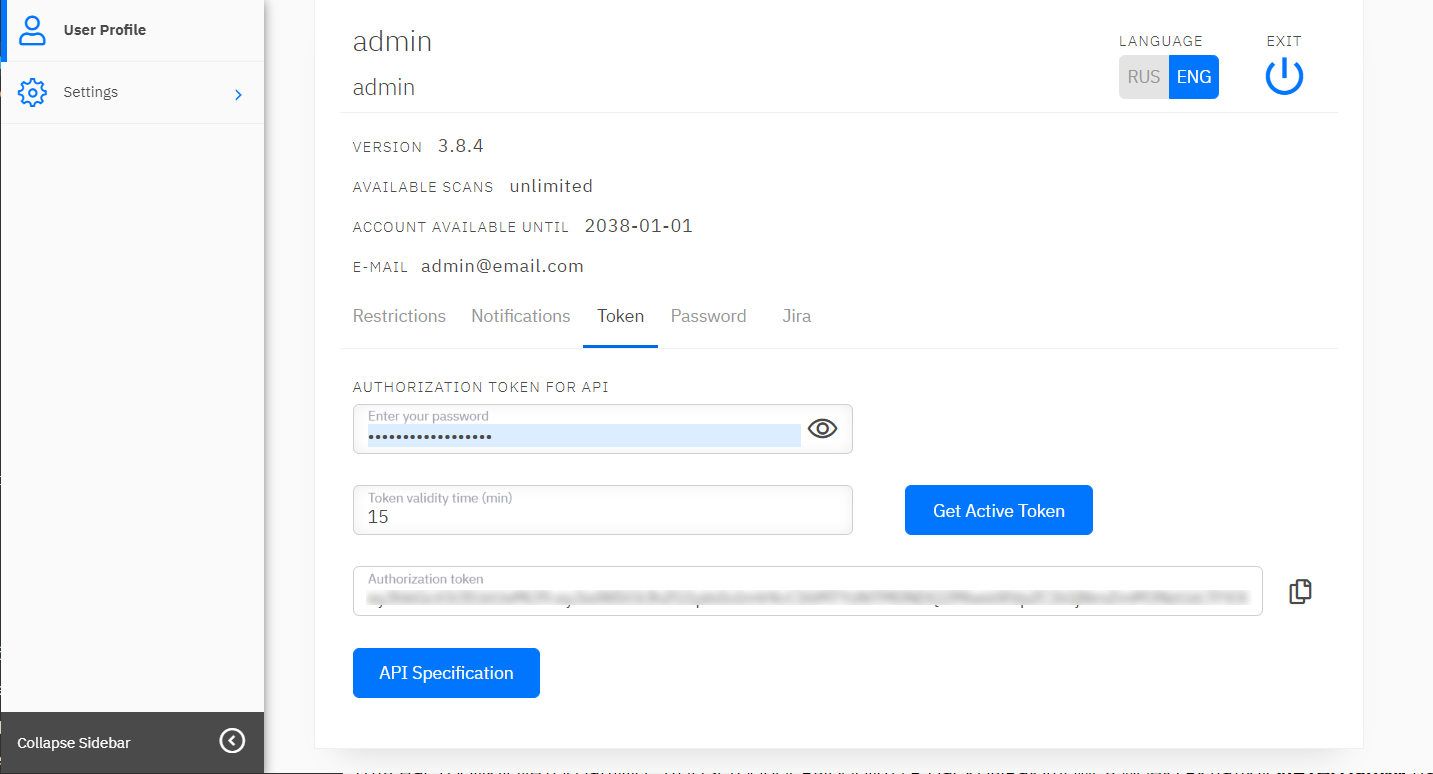
Для интеграции с AppScreener понадобится токен, порядок создания которого приведен ниже.

В пользовательском интерфейсе AppScreener перейдите в профиль пользователя (**Profile**).



Откройте вкладку **Token**. Введите пароль Администратора AppScreener в поле **Enter your password**, укажите время действия токена в минутах в поле **Token validity time (min)** и нажмите кнопку **Get Active Token**.

Скопируйте сгенерированный токен, нажав иконку .



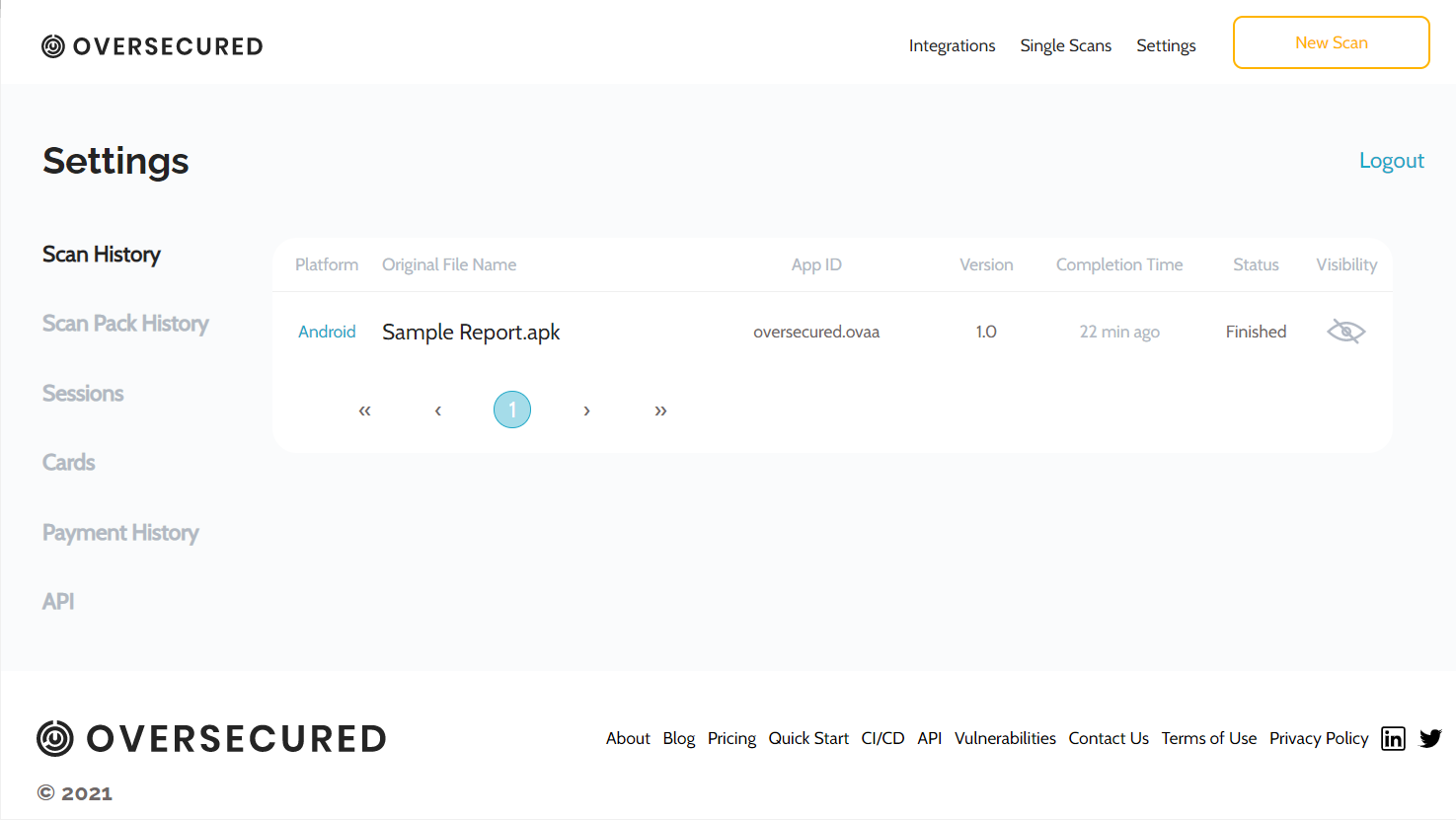
#### Настройка интеграции с AppScreener

Настройка интеграции с AppScreener подробно описана в разделе «[AppScreener](#appscreener)» Руководства пользователя. ### Интеграция с Oversecured

#### Создание токена Oversecured

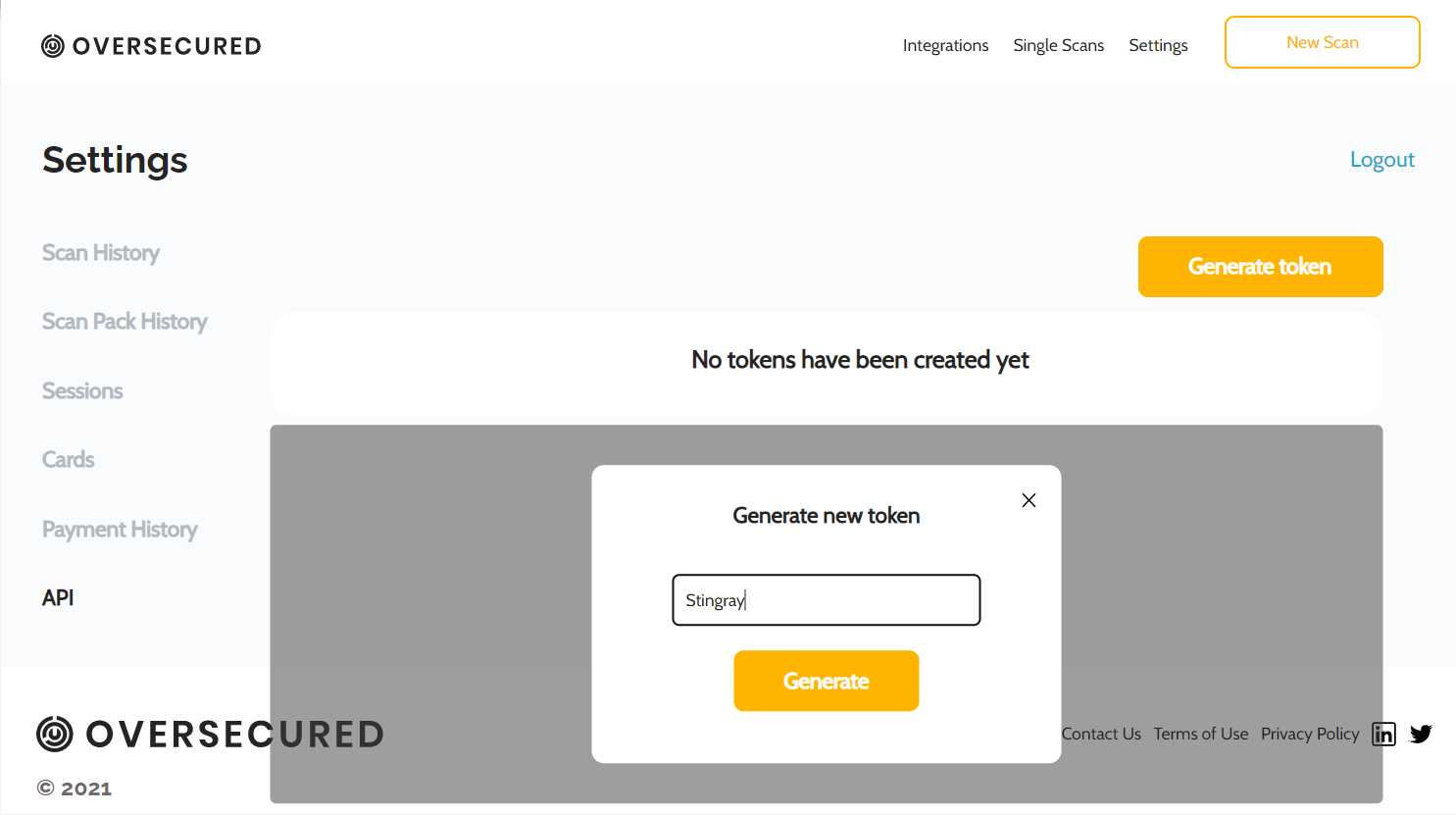
Для интеграции с Oversecured понадобится токен, порядок создания которого приведен ниже.

В Oversecured перейдите на страницу настроек (**Settings**).

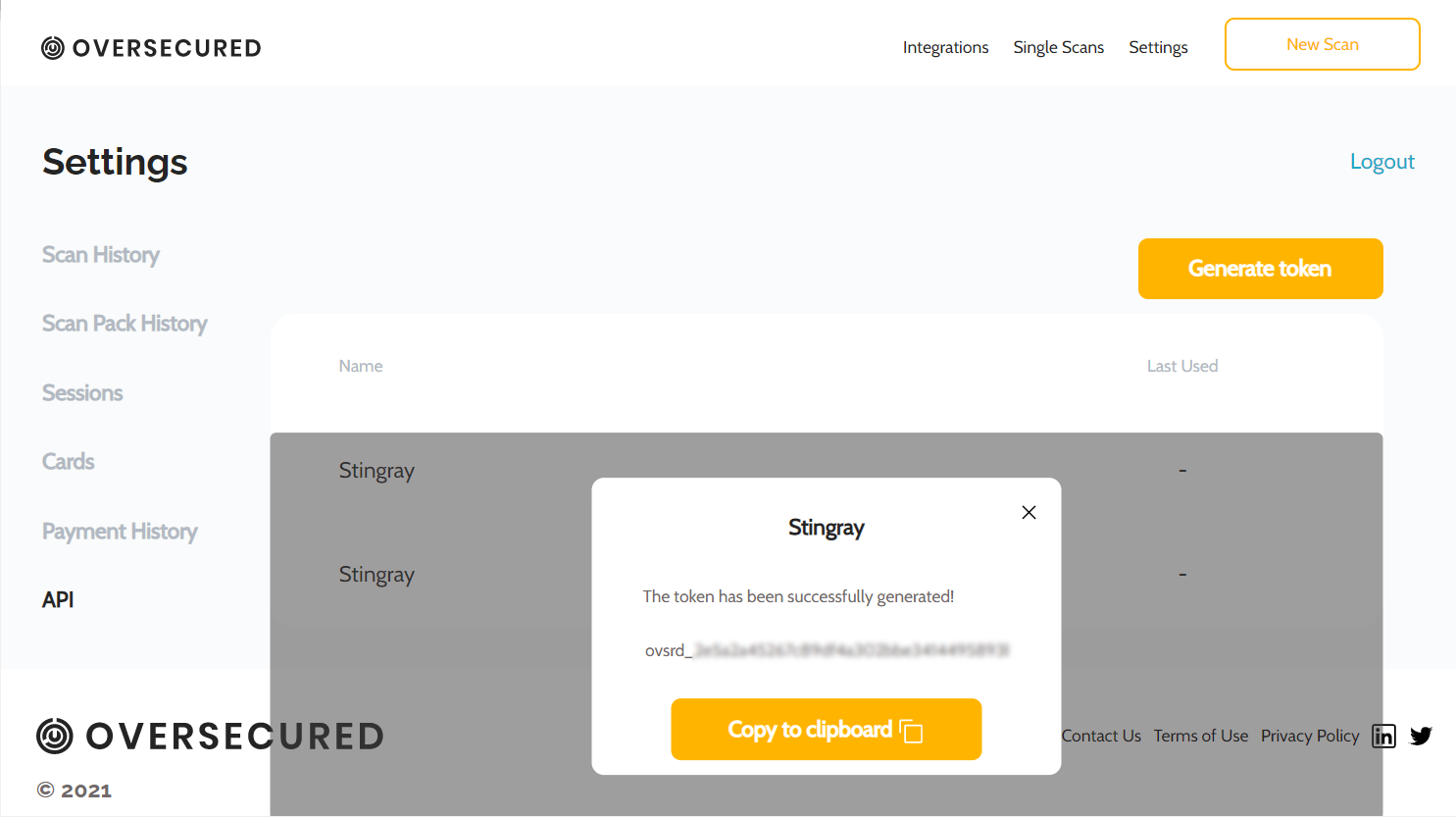


В меню слева выберите пункт **API** и нажмите кнопку **Generate token**.

В открывшемся окне **Generate new token** введите имя создаваемого токена и нажмите кнопку **Generate**.



Нажмите кнопку **Copy to clipboard**, чтобы скопировать токен в буфер обмена.



#### Настройка интеграции с Oversecured

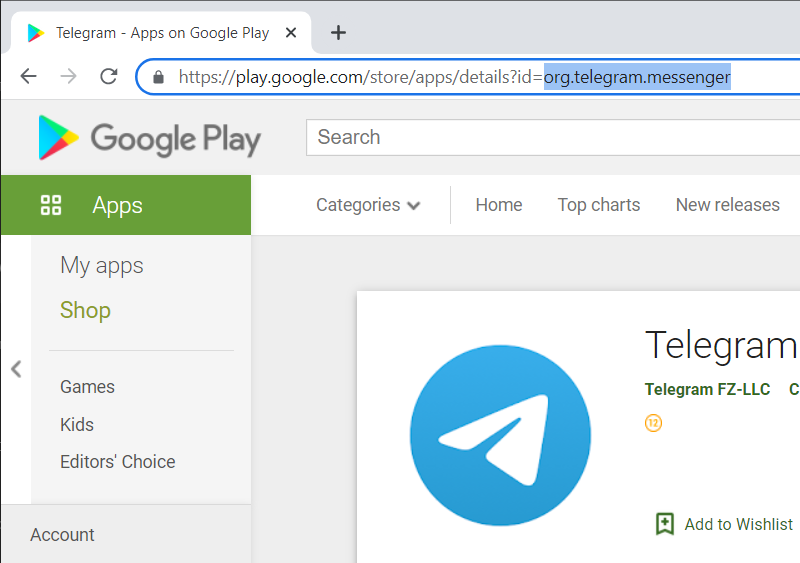
Настройка интеграции с Oversecured подробно описана в разделе «[Oversecured](#oversecured)» Руководства пользователя. ### Интеграция с Google Play

Интеграция с Google Play осуществляется с помощью скрипта [mdast\_scan.py](https://github.com/Dynamic-Mobile-Security/mdast-cli).

#### Сбор необходимых параметров

!!! note “Примечание” Поскольку передача учетных данных осуществляется в незащищенном виде, для интеграции с Google Play следует использовать специально выделенный для этих целей сервисный Google-аккаунт. Двухфакторная аутентификация для аккаунта должна быть отключена.

При предварительном запуске скрипта обязательна передача следующих параметров:

* google\_play\_package\_name — имя скачиваемого пакета. Чтобы узнать имя пакета приложения, можно открыть его страницу в Google Play — имя пакета является частью URL (параметр id);
* 
* google\_play\_email — электронная почта аккаунта Google;
* google\_play\_password — пароль аккаунта Google;
* distribution\_system — для Google Play указываем значение google\_play.

!!! note “Примечание” Если на этом этапе необходимо скачать приложение, добавьте параметр --google\_play\_download\_with\_creds.

Результатом будут следующие сообщения:

20/04/2022 15:03:36 - INFO Google Play - Google Play integration, trying to login  
20/04/2022 15:03:36 - INFO Google Play - Logging in with email and password, you should copy token after  
20/04/2022 15:03:38 - INFO Google Play - gsfId: 36\*\*\*\*\*\*\*\*\*\*2297818, authSubToken: Jg\*\*\*\*\*\*\*\*\*\*RjqFD\_pGqcTFjodc\_mBULuauit8o1uB4-AKFaFKHr6wb9serwzgwLBIRvA.  
20/04/2022 15:03:38 - INFO Google Play - You should copy these parameters and use them for next scans instead of email and password:  
20/04/2022 15:03:38 - INFO Google Play - "--google\_play\_gsfid 36\*\*\*\*\*\*\*\*\*\*2297818 --google\_play\_auth\_token Jgju\_\*\*\*\*\*\*\*\*\*\*FD\_pGqcTFjodc\_mBULuauit8o1uB4-AKFaFKHr6wb9serwzgwLBIRvA."

В последней строке указаны два параметра --google\_play\_gsfid и --google\_play\_auth\_token, которые необходимо скопировать. В дальнейшем они будут использоваться для скачивания приложения и запуска сканирования. Использование этих параметров вместо электронной почты и пароля позволит избежать дальнейших проверок безопасности в браузере.

#### Пример запуска скрипта

После получения необходимых параметров можно, запустив скрипт, скачать приложение и запустить его ручное сканирование.

python mdast\_cli/mdast\_scan.py \  
 --profile\_id 1337 \  
 --architecture\_id 1 \  
 --distribution\_system google\_play \  
 --url "https://saas.mobile.appsec.world" \  
 --company\_id 1 \  
 --token 5d5f6c98\*\*\*\*\*\*\*\*\*487a68ee20d4562d9f \  
 --google\_play\_package\_name com.instagram.android \  
 --google\_play\_gsfid 432\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*43 \  
 --google\_play\_auth\_token JAgw\_2h\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*8KRaYQ.  
 --google\_play\_file\_name best\_apk\_d0wnl04d3r

В результате приложение будет скачано в папку downloaded\_apps под именем best\_apk\_d0wnl04d3r.apk, а также запустится ручное сканирование.

!!! note “Примечание” Более подробная информация о параметрах скрипта приведена в разделе «[Системы CI/CD](./sistemy_ci_cd.md)». ## Настройка мониторинга ### Настройка журналов аудита

#### Журналы аудита

Журналы аудита располагаются в директории, в которой расположены конфигурационные файлы, созданные после запуска Wizard (см. раздел «[Установка Stingray](./ustanovka_stingray.md)» Руководства по установке и интеграции), в общем случае (по умолчанию):

<path\_to\_config\_directory>/logs/audit.log

#### Управление журналами аудита

Управление журналами аудита производится во время первоначальной установки системы и включает в себя следующие пункты:

* Настройка включения аудита.
* Настройка количества записей в одном файле лога.
* Настройка количества хранимых файлов лога.

##### События аудита

События аудита логируются при каждом изменении базы данных в случае любого изменения:

2021-02-07 21:54:38.364420 Process ID: 39 Event ID: 7, Event Name: Update Record,  
Host: http://localhost:4200, User: admin, Args: {'table': 'User', 'id': 1,  
'fields': {'language': {'before': 'en', 'after': 'ru'}}}

Параметры, представленные в каждой записи:

* Время события — время произошедшего события, в формате YYYY-MM-DD h:m:s (год-месяц-день часы:минуты:секунды).
* Идентификатор процесса — идентификатор процесса, от которого произошло событие, внутри системы.
* Идентификатор события — цифровой идентификатор события.
* Имя события — описание события в человеко-читаемом формате.
* Хост — имя хоста, от которого пришел запрос.
* Имя пользователя — пользователь, от имени которого пришел запрос.
* Аргументы запроса — переданные аргументы запроса, определяющие состояние до и после обновления/изменения/удаления и несущие другой информативный характер.

Перечень всех событий и их описание:

|  |  |  |
| --- | --- | --- |
| ID | Название | Описание |
| 1 | Login Success | Событие успешного входа в систему |
| 2 | Login Fail | Неуспешная попытка входа в систему |
| 3 | Logout | Событие выхода из системы |
| 4 | Bad Request | Событие, определяющее неправильное обращение к системе. В информации события указывается код ошибки, которую вернул сервер. Включает в себя события попытки доступа неавторизованного пользователя к ресурсам (код ошибки 401), попытки доступа к ресурсам, для которых у пользователя нет прав (код ошибки 403), а также неверно сформированные запросы (код ошибки 400) и т. д. |
| 5 | Server Error | Событие об ошибке сервера с информацией о причине возникновения ошибки |
| 6 | New Record | Создание нового элемента в системе. Список возможных элементов, к которым применяется аудит, указан в таблице ниже |
| 7 | Update Record | Изменение элемента в системе. Логируется старое и новое значение элемента. Список возможных элементов, к которым применяется аудит, указан в таблице ниже |
| 8 | Delete Record | Удаление элемента в системе. Список возможных элементов, к которым применяется аудит, указан в таблице ниже |
| 9 | New Related Record | Событие создания новой связанной сущности. В системе присутствует иерархия сущностей и при явном добавлении одного элемента может создаваться несколько дочерних.Для разделения событий создания основных и вложенных сущностей используются различные типы событий |
| 10 | Delete Related Record | Событие удаления связанной сущности. В системе присутствует иерархия сущностей и при явном удалении одного элемента может удаляться несколько дочерних. Для разделения событий удаления основных и вложенных сущностей используются различные типы событий |
| 11 | Change Password | Событие смены пароля пользователей |
| 12 | Dast Action | Событие совершения операций со сканированиями (запуск, остановка, завершение сканирования) |
| 13 | Testcase Action | Событие совершения операций с автотестами (запуск, остановка, завершение записи автотеста) |

Список элементов, к которым применяются события создания/удаления/модификации:

|  |  |  |
| --- | --- | --- |
| № | Модель в БД | Описание |
| 1 | stingray\_user | Пользователи системы |
| 2 | stingray\_project | Проекты |
| 3 | stingray\_profile | Профили |
| 4 | stingray\_dast | Сканирования |
| 5 | stingray\_dastIssue | Уязвимости, выявленные в ходе сканирования |
| 6 | stingray\_DastResult | Собранные в результате проведения сканирования данные |
| 7 | stingray\_ProjectIssue | Сущность, связывающая уязвимости между сканированиями |
| 8 | stingray\_TestCase | Автотесты, записываемые внутри системы |
| 9 | stingray\_Rule | Правила поиска уязвимостей |
| 10 | stingray\_RuleModule | Связка правил поиска уязвимостей с модулями для поиска |
| 11 | stingray\_RuleExpression | Связка правил поиска уязвимостей со строками поиска |
| 12 | stingray\_Expression | Связка правил поиска уязвимостей с возможными местами поиска |
| 13 | stingray\_Injection | Строки поиска для определения чувствительной информации |
| 14 | stingray\_Settings | Настройки системы |
| 15 | stingray\_RequirementGroup | Стандарты и категории внутри стандартов |
| 16 | stingray\_Requirement | Требования внутри стандартов |