



БУДУЩЕЕ  
В НАШИХ  
РУКАХ

# Поймай меня, если сможешь: как обнаружить ROOT и Frida

Бужинская Таисия



## Бужинская Таисия

Инженер-программист

- 3 года опыта в Android-разработке
- Активно использовала ROOT-права на разных устройствах
- Портировала кастомные прошивки и делилась результатом с сообществом

## Что мы знаем о безопасности мобильных приложений

---

Зачем нужен ROOT — о правах в Linux-based ОС

SELinux — применение в Android

ROOT в Android — задача с множеством решений

Шалость удалась — о ROOT глазами хакера

Trusted Execution Environment — помощник в защите данных

Hardware Attestation — надежная проверка Android-устройства

eFUSE — как физики помогли сделать устройства безопаснее

Android Verified Boot — от кнопки включения до экрана блокировки

Все о FRIDA — использование и обнаружение

# Безопасность мобильных приложений

## Ожидание



Как же сильны мои лапищи:

- Бизнес-логика на сервере
- Мои данные независимы от других приложений
- Проверки перед публикацией в сторе

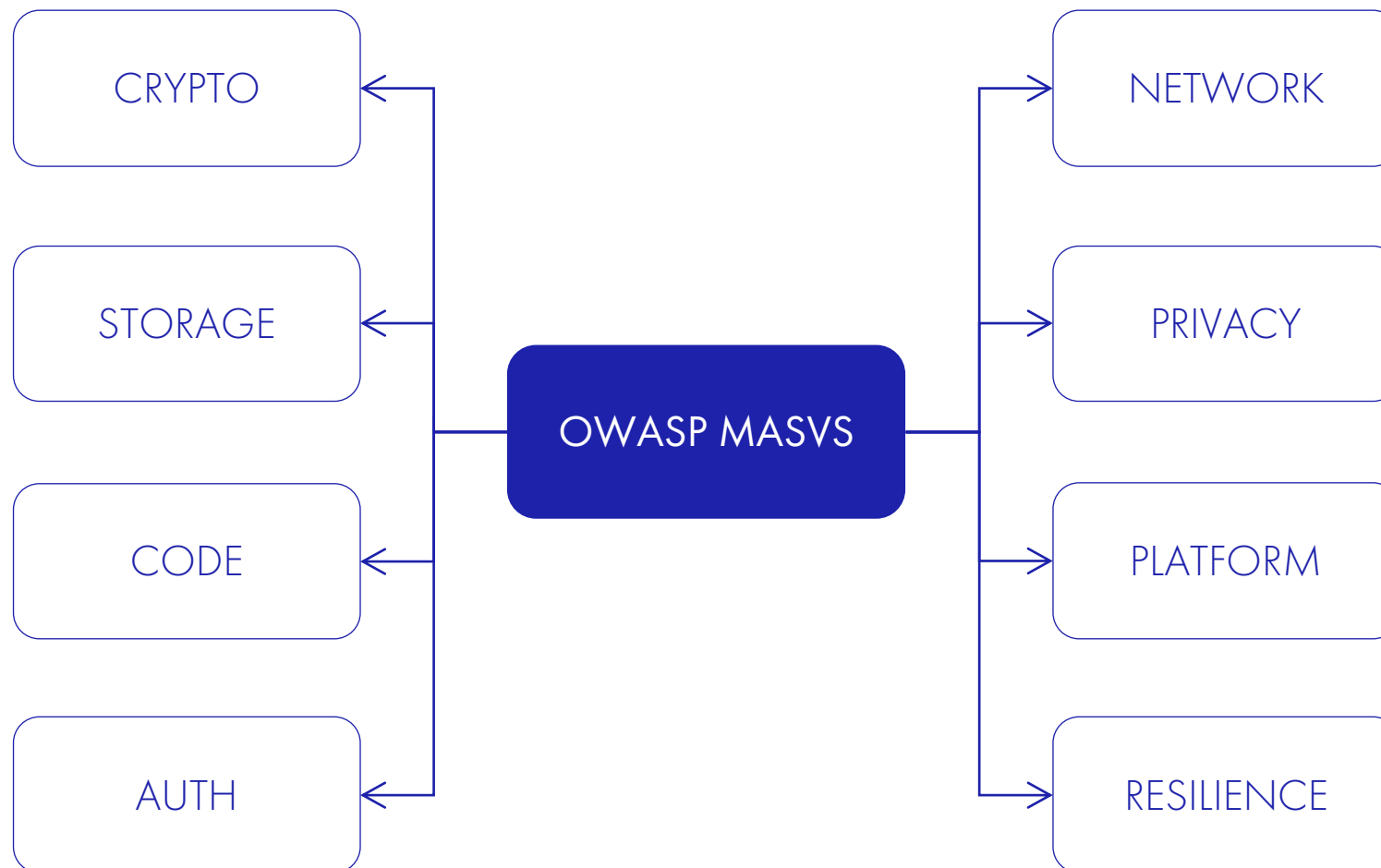
## Реальность



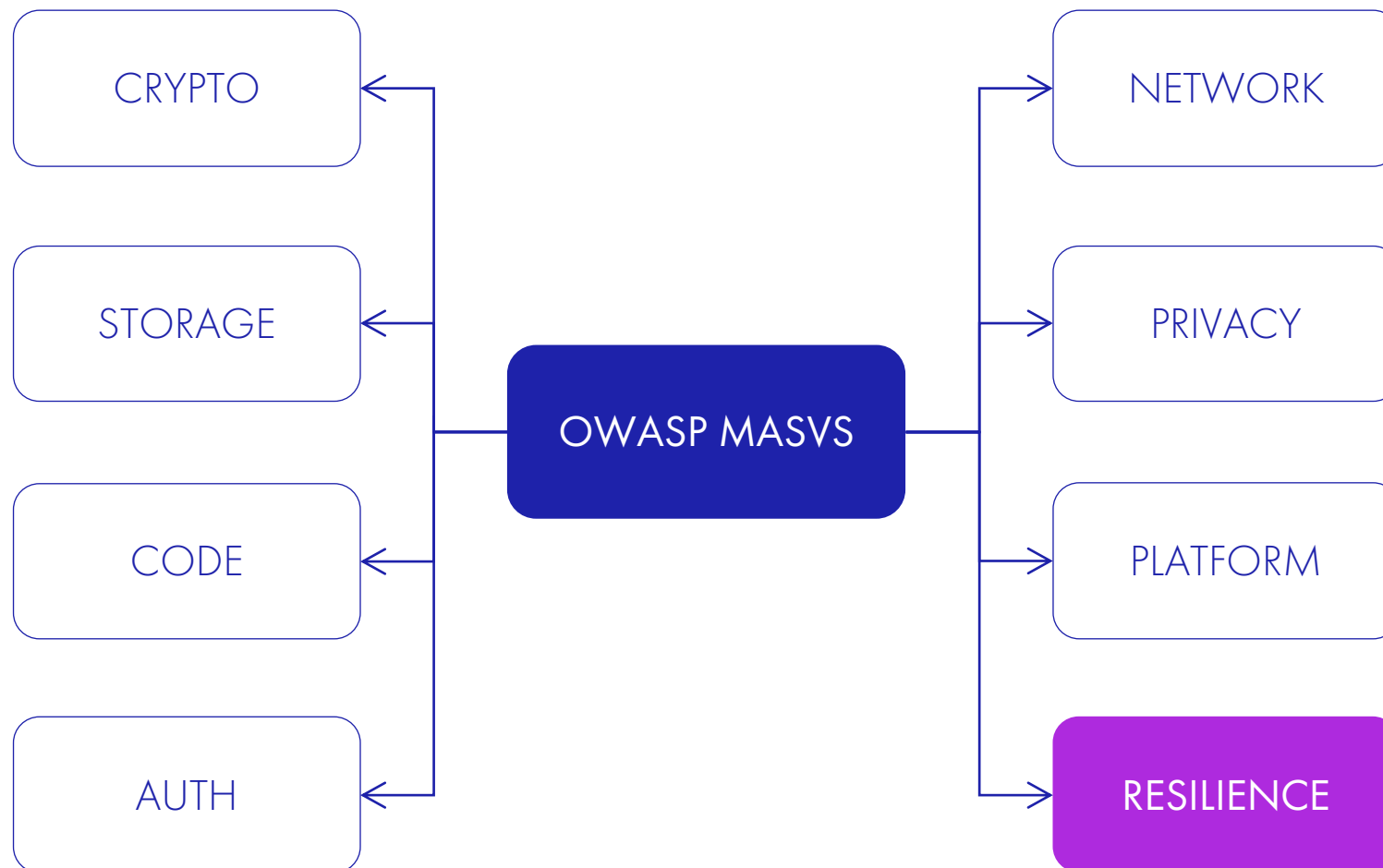
Нужно что-то делать, пока не поздно:

- Проверка на ROOT
- Защита от перехвата трафика
- Защита от декомпиляции

# OWASP



# OWASP



Что мы знаем о безопасности мобильных приложений

Зачем нужен ROOT — о правах в Linux-based ОС

---

SELinux — применение в Android

ROOT в Android — задача с множеством решений

Шалость удалась — о ROOT глазами хакера

Trusted Execution Environment — помощник в защите данных

Hardware Attestation — надежная проверка Android-устройства

eFUSE — как физики помогли сделать устройства безопаснее

Android Verified Boot — от кнопки включения до экрана блокировки

Все о FRIDA — использование и обнаружение



# Discretionary Access Control (DAC)

Для каждого объекта в системе есть **Permission**.

**Permission** отвечает на вопрос: **Имеет ли субъект S право R на объект O?**

**Permission** задает ограничения для 3 категорий пользователей:

01

Владелец (Owner)

02

Группа (Group)

03

Остальные (Others)



# Discretionary Access Control — категории пользователей

## 01 Владелец (Owner)

- Создатель объекта становится его владельцем
- Владелец определяет права доступа к своим объектам
- У каждого объекта есть владелец



# Discretionary Access Control — категории пользователей

## 02 Группа (Group)

- Состоит из совокупности пользователей
- Участники не могут изменять права доступа к объекту



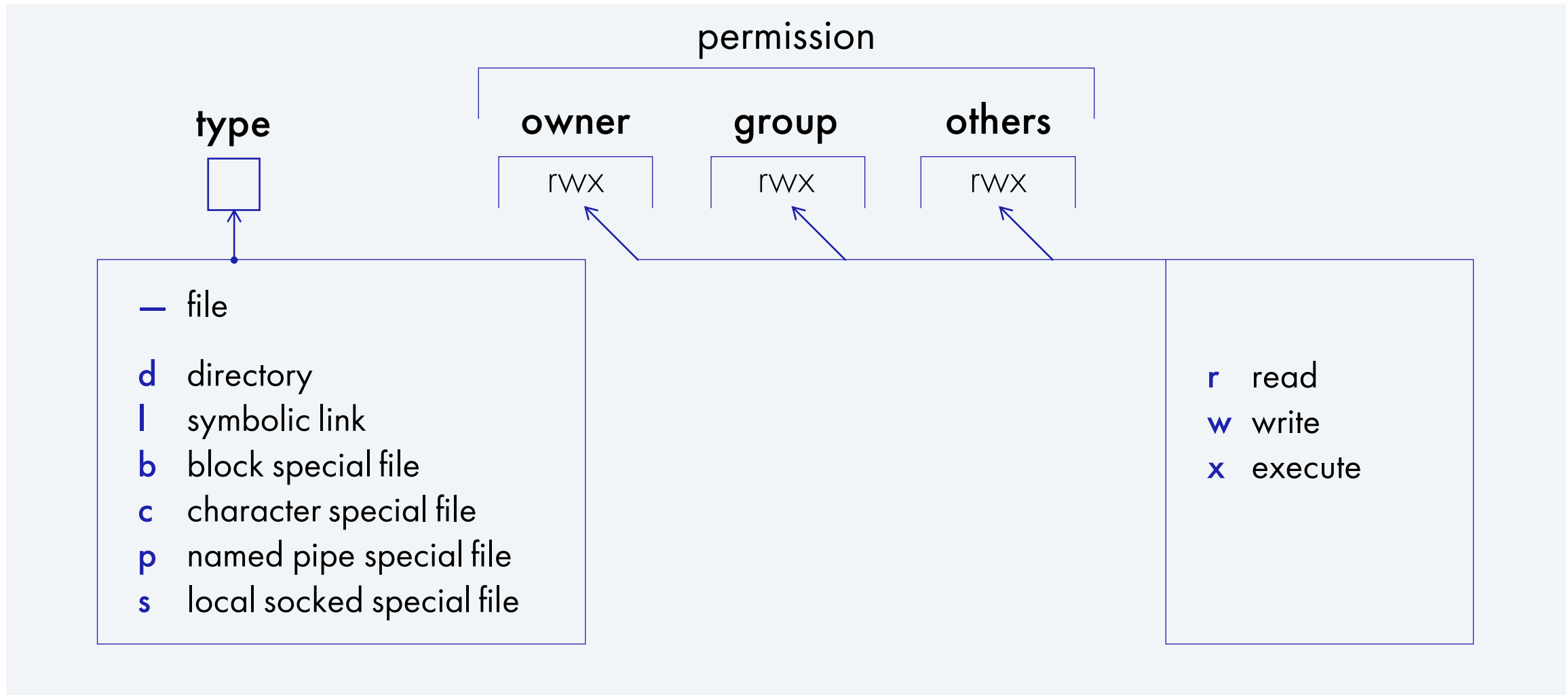
# Discretionary Access Control — категории пользователей

## 03 Остальные (Others)

- Пользователи, не вошедшие в предыдущие категории



# Permission в Discretionary Access Control



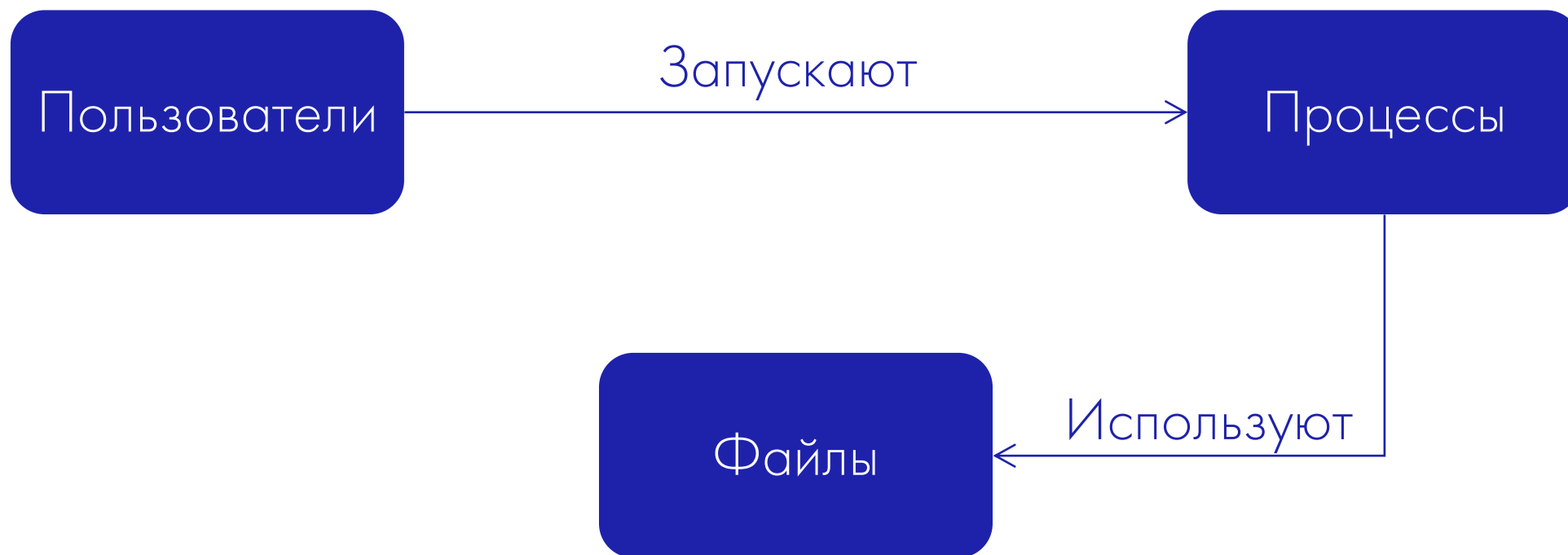


# Особенности Discretionary Access Control



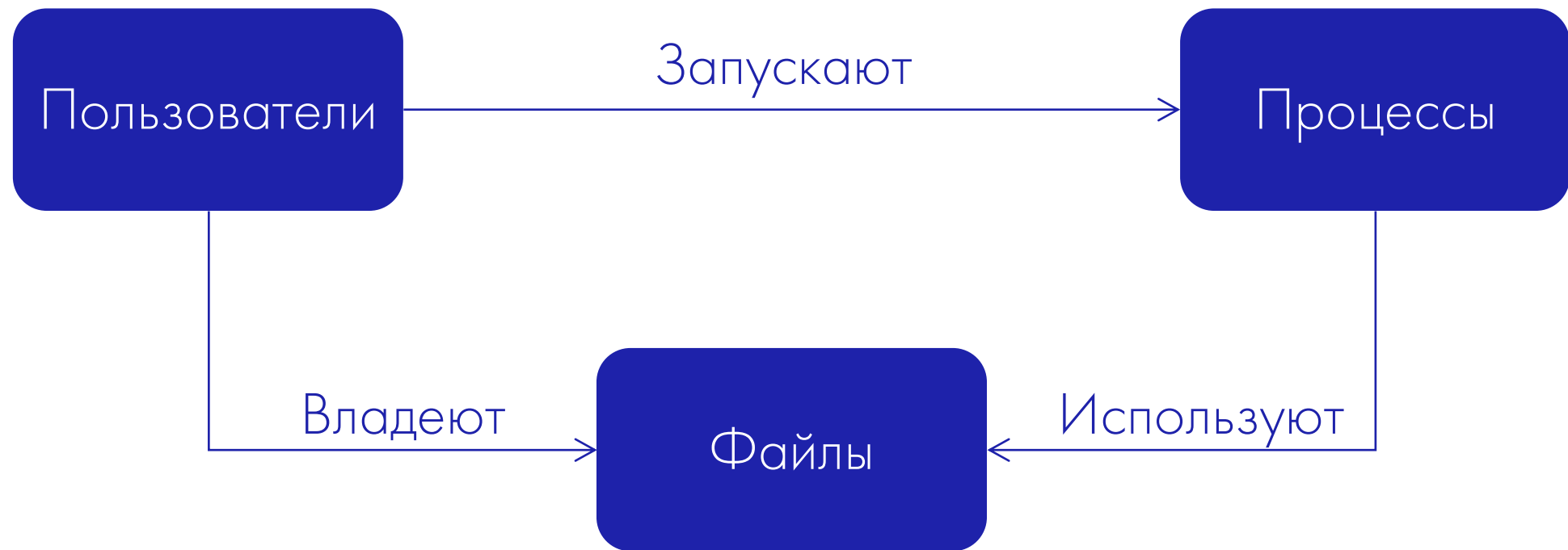


# Особенности Discretionary Access Control



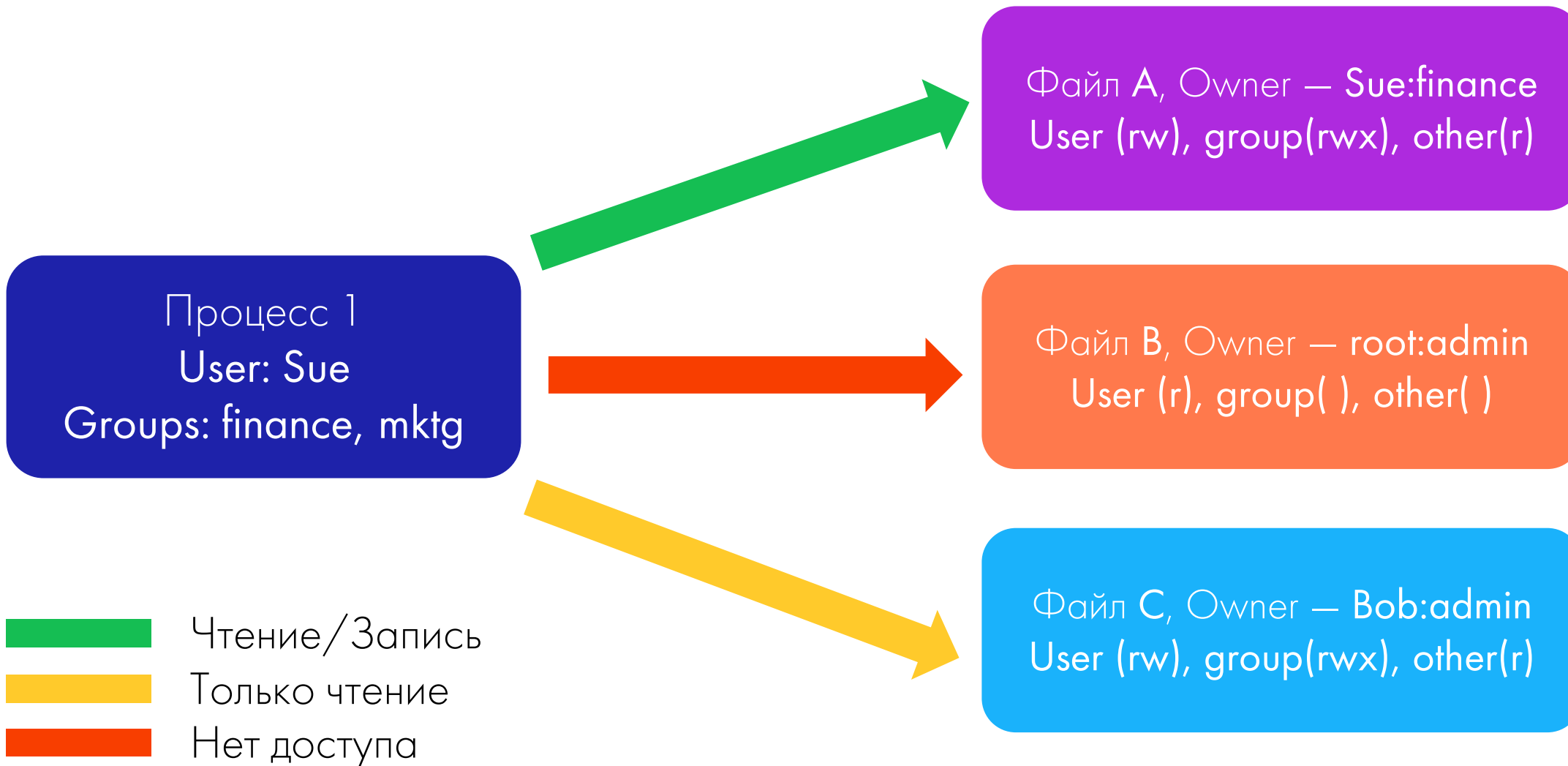


# Особенности Discretionary Access Control



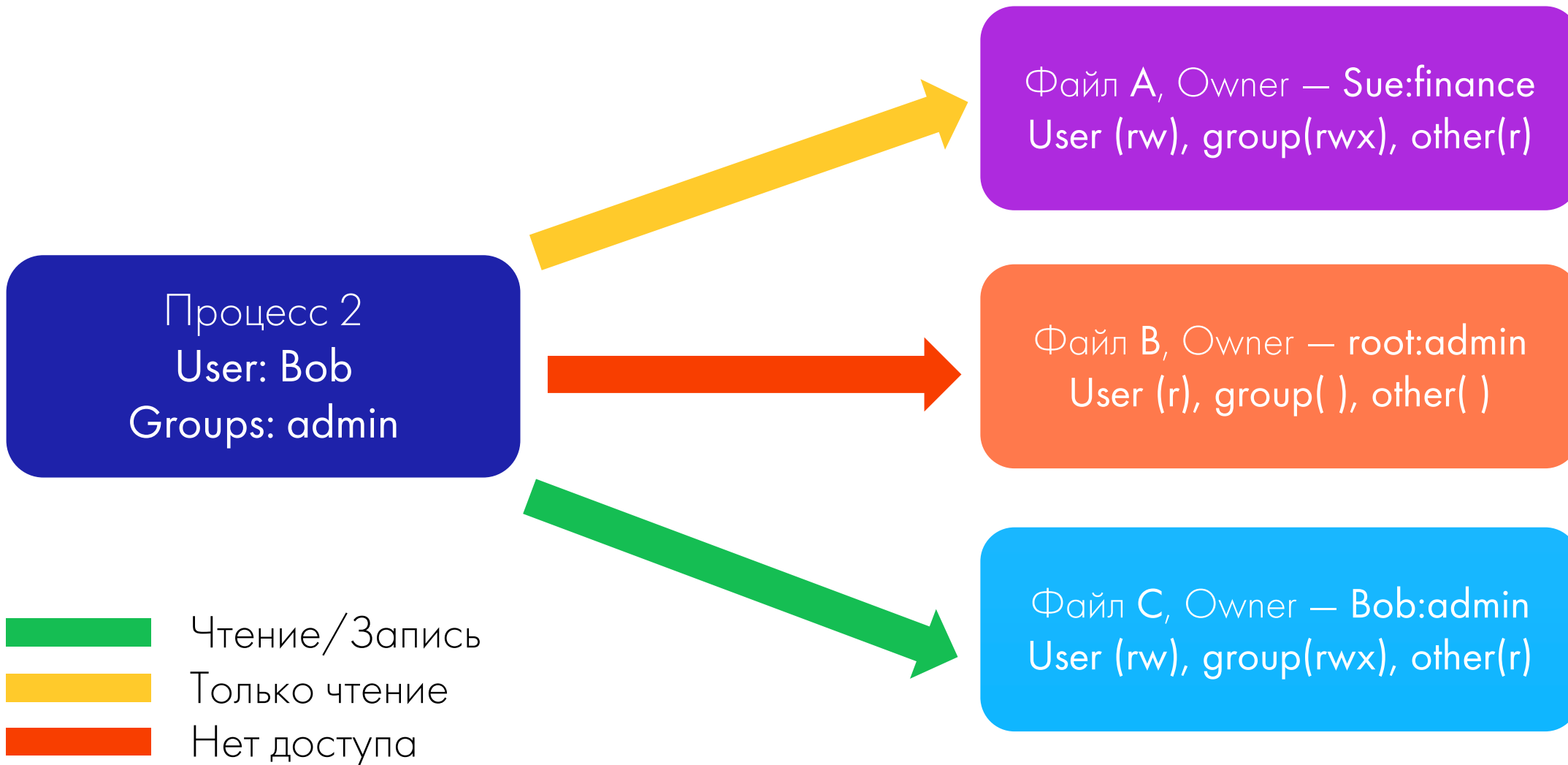


# Взаимодействие процессов с файлами в DAC





# Взаимодействие процессов с файлами в DAC





# Пример использования DAC в Android — Permissions



Permissions под капотом

[habr.com/p/741574](https://habr.com/p/741574)





# Недостатки Discretionary Access Control (DAC)

01

Не учитывается важность объекта и необходимости доступа процесса к нему

02

Владелец объекта решает, являются ли данные объекта секретными или будут доступными остальным пользователям

03

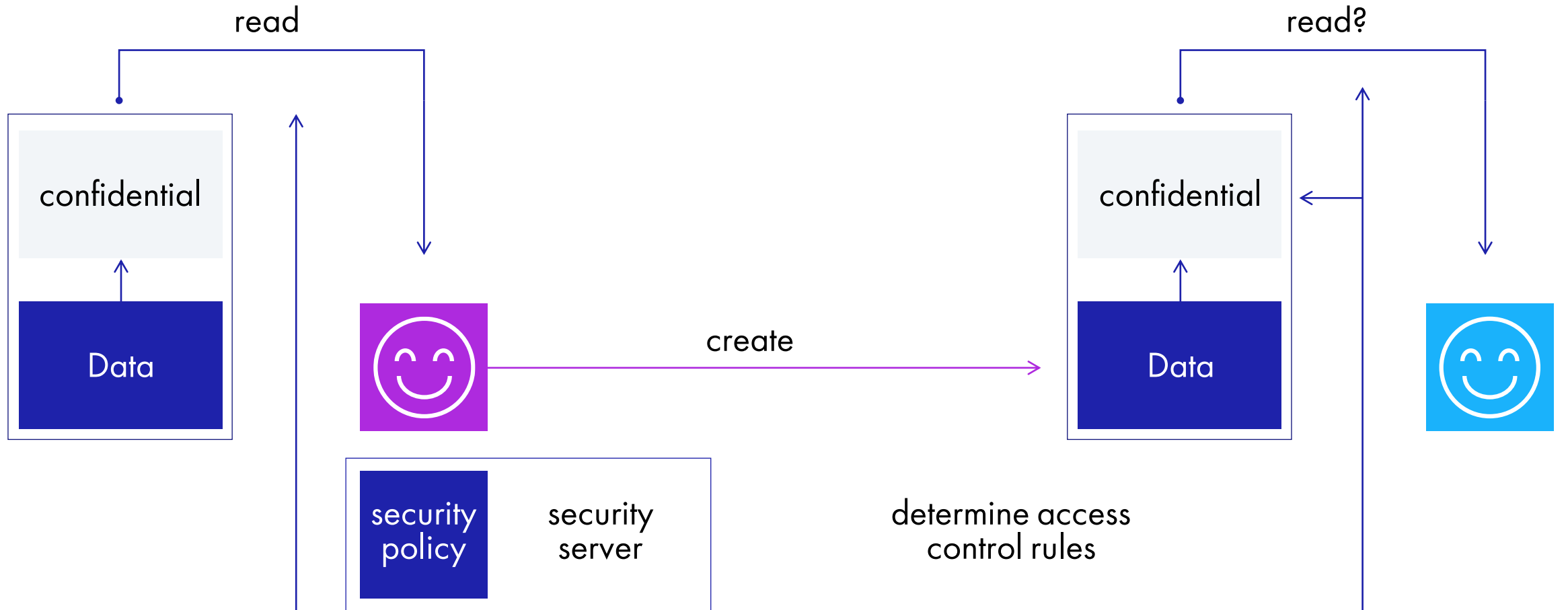
Нельзя задать ограничения доступа для конкретных приложений



**ROOT не имеет ограничений прав доступа**



# Mandatory Access Control (MAC)





## Что мы узнали?

# 01

Для контроля доступа в Discretionary Access Control используются Permissions

# 02

Discretionary Access Control имеет существенные недостатки, для преодоления которых используется Mandatory Access Control

# 03

Android Permissions основаны на Discretionary Access Control

Что мы знаем о безопасности мобильных приложений

Зачем нужен ROOT — о правах в Linux-based ОС

SELinux — применение в Android

---

ROOT в Android — задача с множеством решений

Шалость удалась — о ROOT глазами хакера

Trusted Execution Environment — помощник в защите данных

Hardware Attestation — надежная проверка Android-устройства

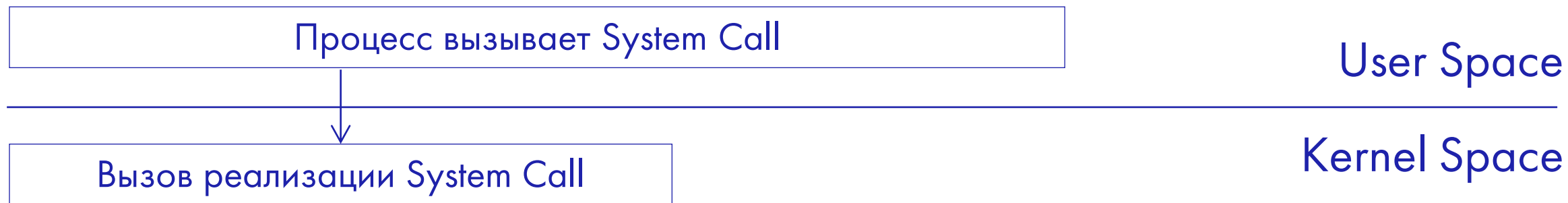
eFUSE — как физики помогли сделать устройства безопаснее

Android Verified Boot — от кнопки включения до экрана блокировки

Все о FRIDA — использование и обнаружение

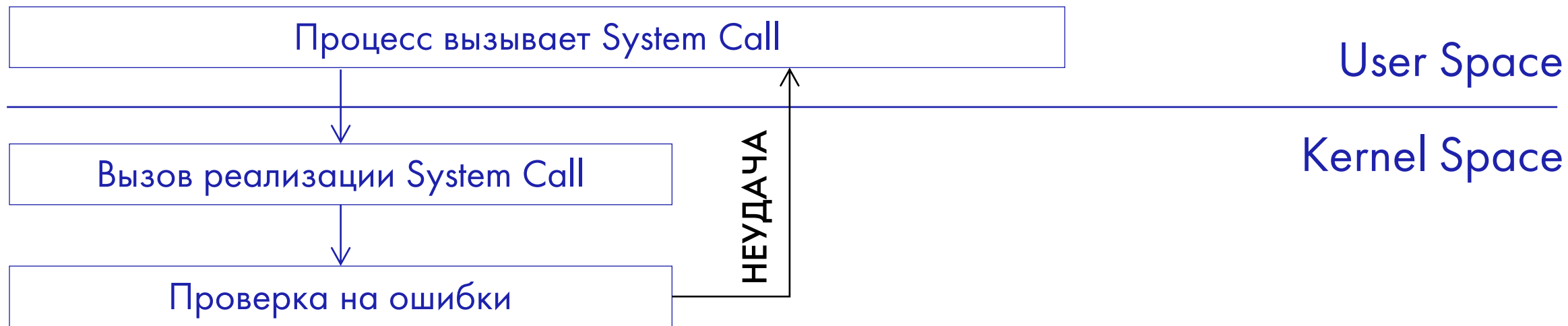


# Процесс исполнения системного вызова



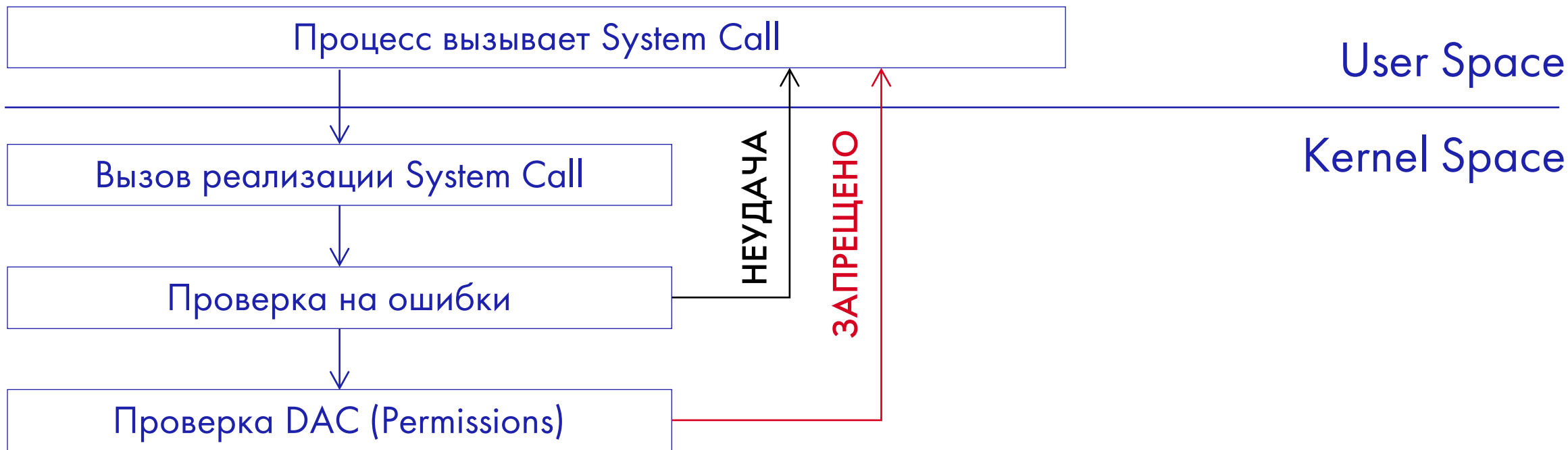


# Процесс исполнения системного вызова



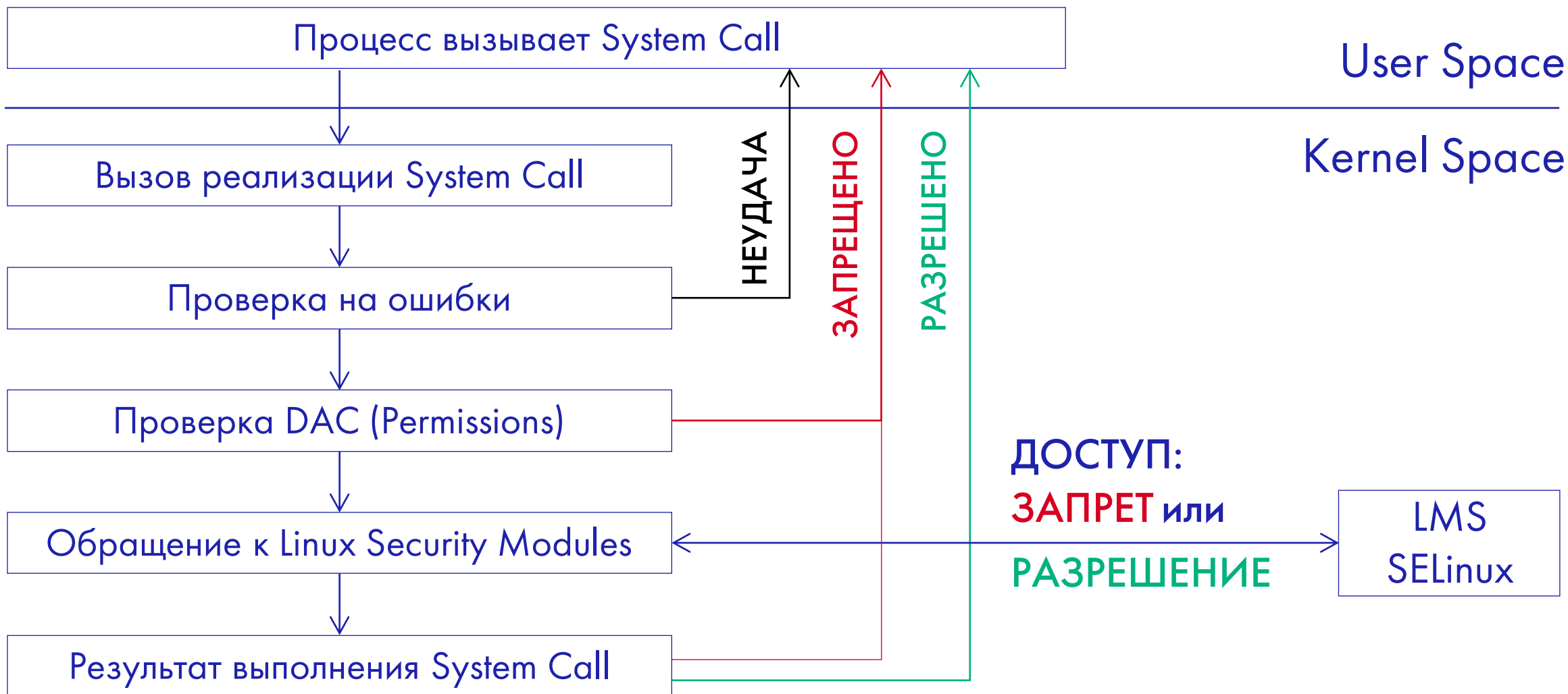


# Процесс исполнения системного вызова





# Процесс исполнения системного вызова



## Контекст SELinux



Каждый процесс и файл помечаются контекстом SELinux.

Контекст SELinux описывается в виде **user:role:type:level**, где:

- **user** — имя пользователя

# Контекст SELinux



Каждый процесс и файл помечаются контекстом SELinux.

Контекст SELinux описывается в виде **user:role:type:level**, где:

- **user** — имя пользователя
- **role** — имя роли



## Контекст SELinux

Каждый процесс и файл помечаются контекстом SELinux.

Контекст SELinux описывается в виде **user:role:type:level**, где:

- **user** — имя пользователя
- **role** — имя роли
- **type** — имя домена для процессов, а также имя типа для файлов и каталогов



# Контекст SELinux

Каждый процесс и файл помечаются контекстом SELinux.

Контекст SELinux описывается в виде **user:role:type:level**, где:

- **user** — имя пользователя
- **role** — имя роли
- **type** — имя домена для процессов, а также имя типа для файлов и каталогов
- **level** — уровень доступа, задается от s0 до s15



# Особенности реализации SELinux в Android

В Android реализация SELinux имеет следующие ограничения:

## 01

доступен только  
1 пользователь — **u**

## 02

доступны только  
2 роли:

- **object\_r** — для файлов
- **r** — для остального

## 03

level всегда **s0**



# Установка приложений — что под «капотом»

```
public class PackageManagerService
```

Class declared by `com.android.server.pm`

---

Keep track of all those APKs everywhere.

Internally there are three important locks:

- `#mLock` is used to guard all in-memory parsed package details and other related state. It is a fine-grained lock that should only be held momentarily, as it's one of the most contended locks in the system.
- `#mInstallLock` is used to guard all `installd` access, whose operations typically involve heavy lifting of application data on disk. Since `installd` is single-threaded, and its operations can often be slow, this lock should never be acquired while already holding `#mLock`. Conversely, it's safe to acquire `#mLock` momentarily while already holding `#mInstallLock`.
- `#mSnapshotLock` is used to guard access to two snapshot fields: the snapshot itself and the snapshot invalidation flag. This lock should never be acquired while already holding `#mLock`. Conversely, it's safe to acquire `#mLock` momentarily while already holding `#mSnapshotLock`.



# Установка приложений — что под «капотом»

```
/** Starts PackageManagerService. */
public static PackageManagerService main(Context context,
                                         Installer installer,
                                         @NonNull DomainVerificationService domainVerificationService,
                                         boolean factoryTest) {
    ...
    try (PackageManagerTracedLock installLock = mInstallLock.acquireLock()) {
        // writer
        synchronized (mLock) {
            ...
            SELinuxMMAC.readInstallPolicy();
            ...
        }
        ...
    }
    ...
}
```

Использование `installd` с помощью `mInstallLock`

Определение контекста SELinux для установленного приложения



## Запуск приложения — что под «капотом»

```
// Utility routine to specialize a zygote child process.
static void SpecializeCommon(JNIEnv * env, jstring managed_se_info, ...) {
    ...
    auto se_info = extract_fn(managed_se_info);
    ...
    const char * se_info_ptr = se_info.has_value() ? se_info.value().c_str() : nullptr;
    if (selinux_android_setcontext(uid, is_system_server, se_info_ptr, nice_name_ptr) == -1) {
        fail_fn(CREATE_ERROR("selinux_android_setcontext(%d, %d, \"%s\", \"%s\") failed", uid,
            is_system_server, se_info_ptr, nice_name_ptr));
    }
    ...
}
```

Чтение информации о контексте SELinux запускаемого приложения

Установка контекста SELinux для дочернего процесса



# Контекст SELinux и подпись приложения



# Что мы узнали?



## 01

**SELinux** — это  
Mandatory Access  
Control

## 02

**SELinux** — один  
из ключевых элементов  
безопасности Android

## 03

**SELinux** используется  
в Android  
для реализации  
Application Sandbox





## ROOT, встроенный в прошивку

Для образа Android есть 3 варианта сборки:

01

`user` —  
релизный вариант

02

`userdebug` —  
отладочный вариант

03

`eng` —  
отладочный вариант

В отладочных вариантах встроен **ROOT**, по умолчанию включен **adb** и есть специальный инструментарий для отладки



## ROOT, встроенный в прошивку

В процессе сборки отладочного образа, в политики SELinux включается контекст **u:r:su:s0**

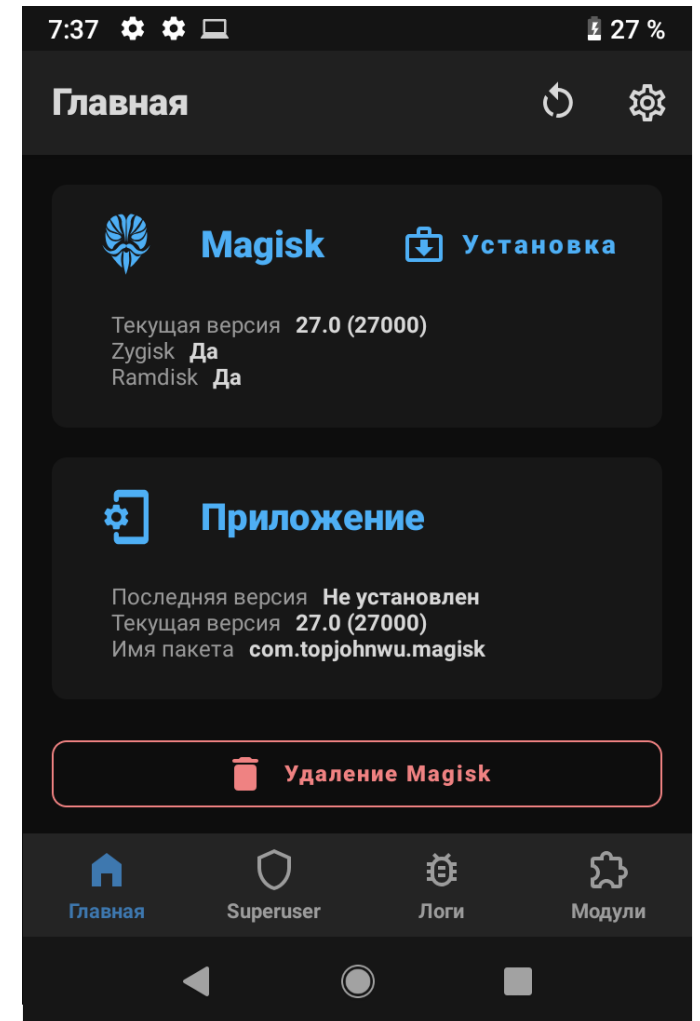
```
userdebug_or_eng(`  
    ...  
    # allow init to execute services marked with seclabel u:r:su:s0 in userdebug/eng  
    allow init su:process transition;  
    dontaudit init su:process noatsecure;  
    allow init su:process { siginh rlimitinh };  
`)
```

# Magisk



При установке Magisk вносит множество изменений в образ Android на устройстве:

- Добавляет в политики SELinux собственные типы и домены: **magisk** и **magisk\_file**
- Устанавливает приложение по управлению ROOT в качестве системного



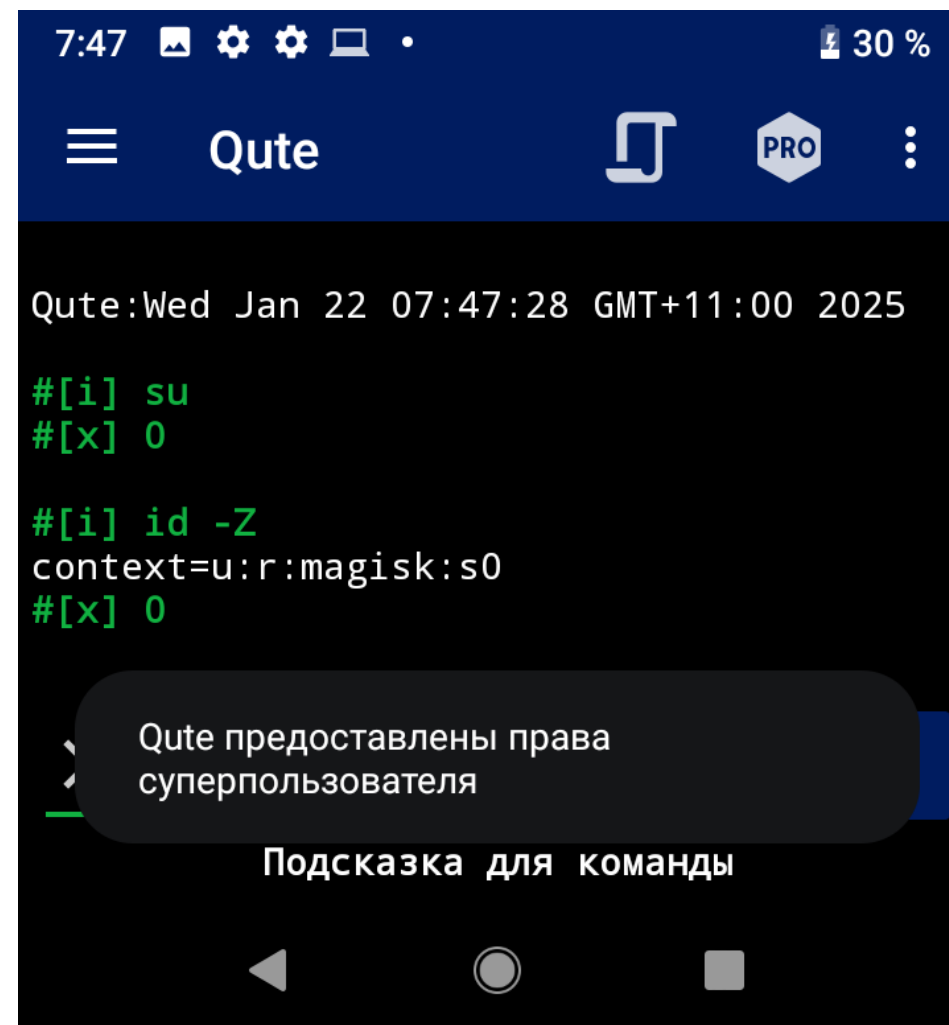
# Magisk



`u:r:magisk:s0` — основной контекст SELinux для Magisk

В данном контексте работают:

- все процессы с ROOT
- magiskd



# KernelSU



Использует особенности Generic Kernel Image

Поддерживает устройства со следующими параметрами:

- версия Android  $\geq 12$
- версия ядра Linux  $\geq 5.10$
- архитектура процессора – arm64-v8a и x86-64



# Зачем нужен Generic Kernel Image?

Раньше процесс создания ядра под конкретное устройство выглядел следующим образом:

01

Бралась последняя  
версия ядра  
Linux Long Term  
Supported (LTS)



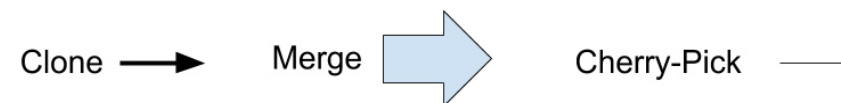
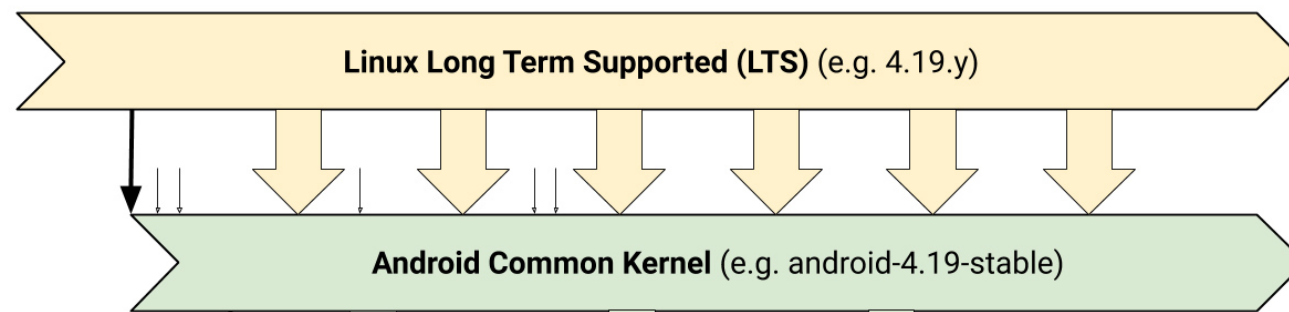
**Linux Long Term Supported (LTS)** (e.g. 4.19.y)



# Зачем нужен Generic Kernel Image?

## 02

Google на базе Linux LTS делал Android Common Kernel

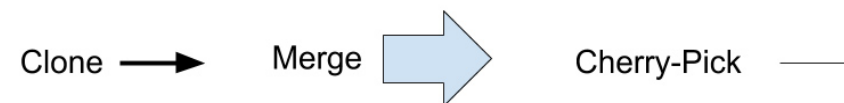
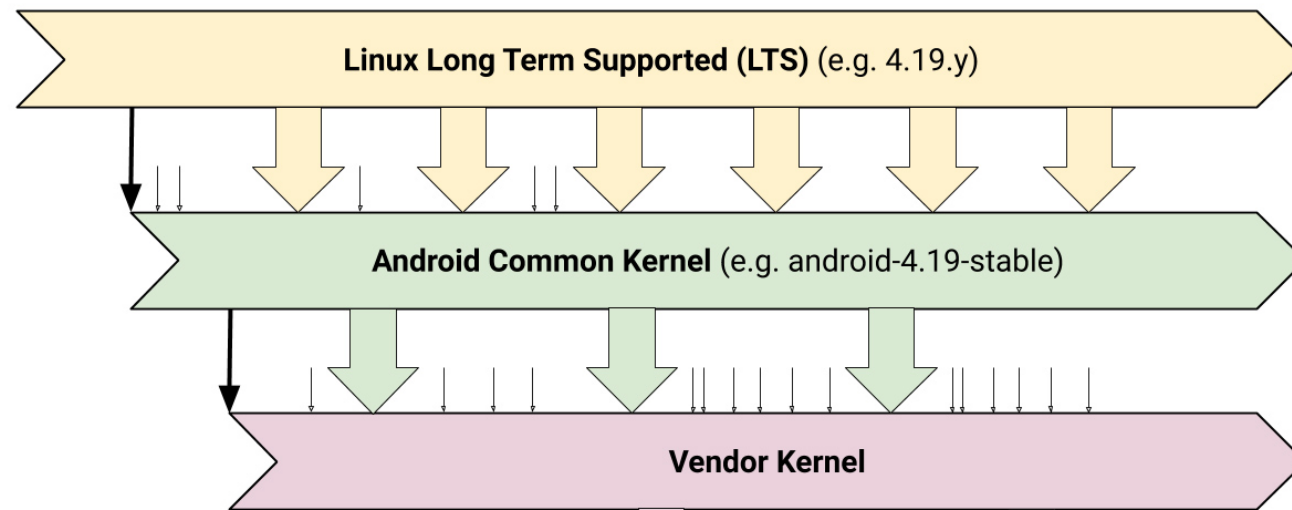




# Зачем нужен Generic Kernel Image?

## 03

System-on-Chip  
Vendor вносил  
изменения  
в Android Common  
Kernel

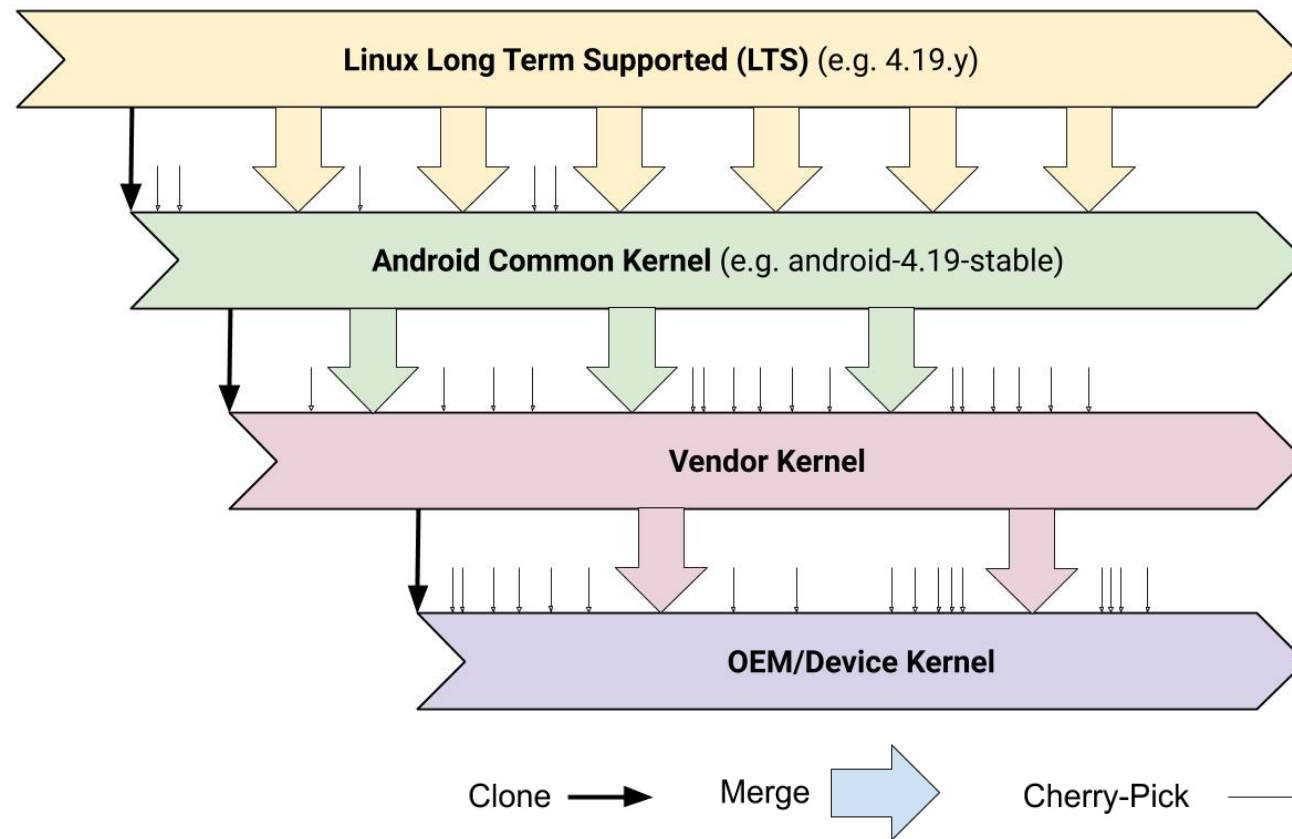




# Зачем нужен Generic Kernel Image?

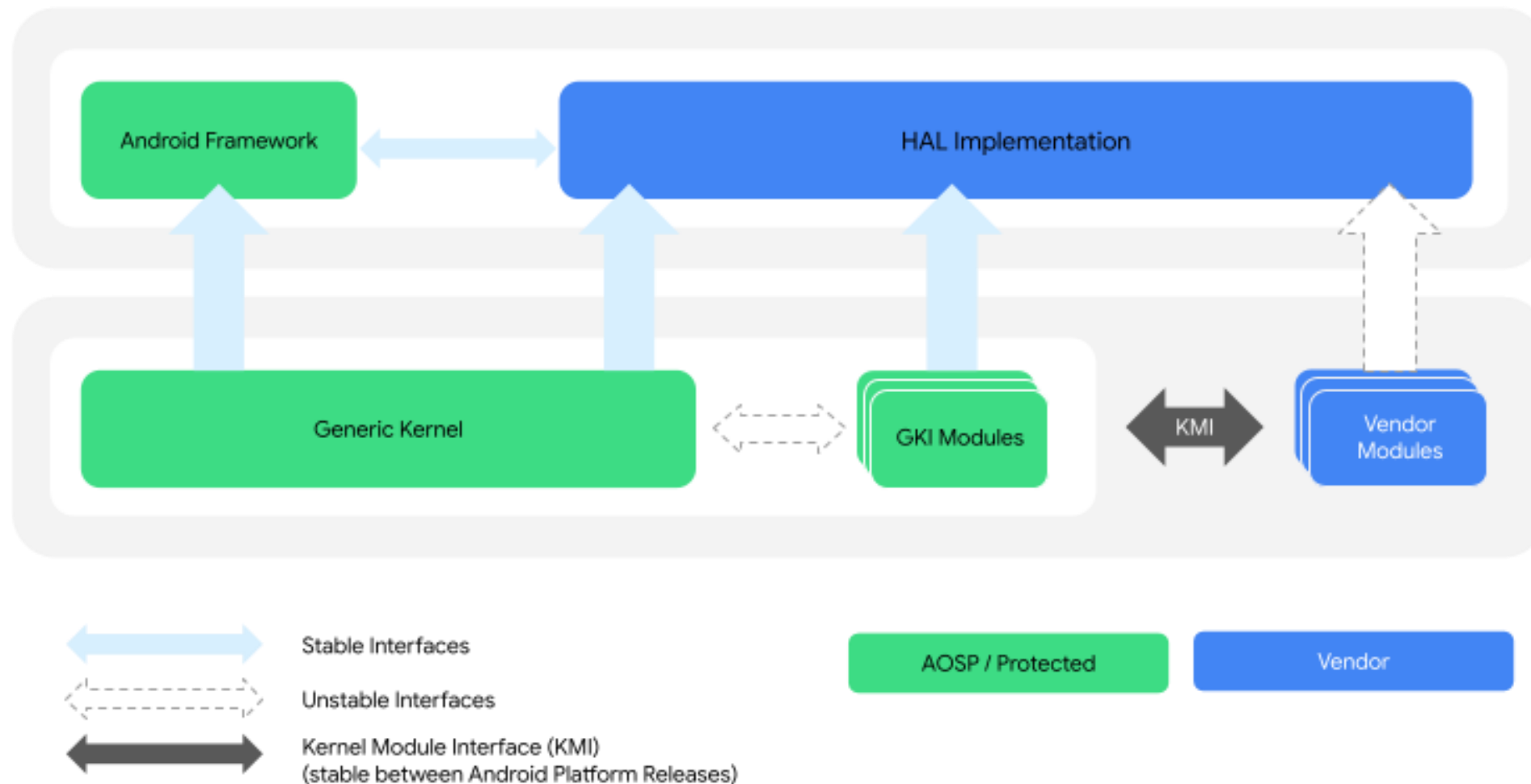
## 04

Производитель устройства вносит изменения в Vendor Kernel





# Generic Kernel Image — решение против фрагментации





# KernelSU — варианты эксплуатации

## 01

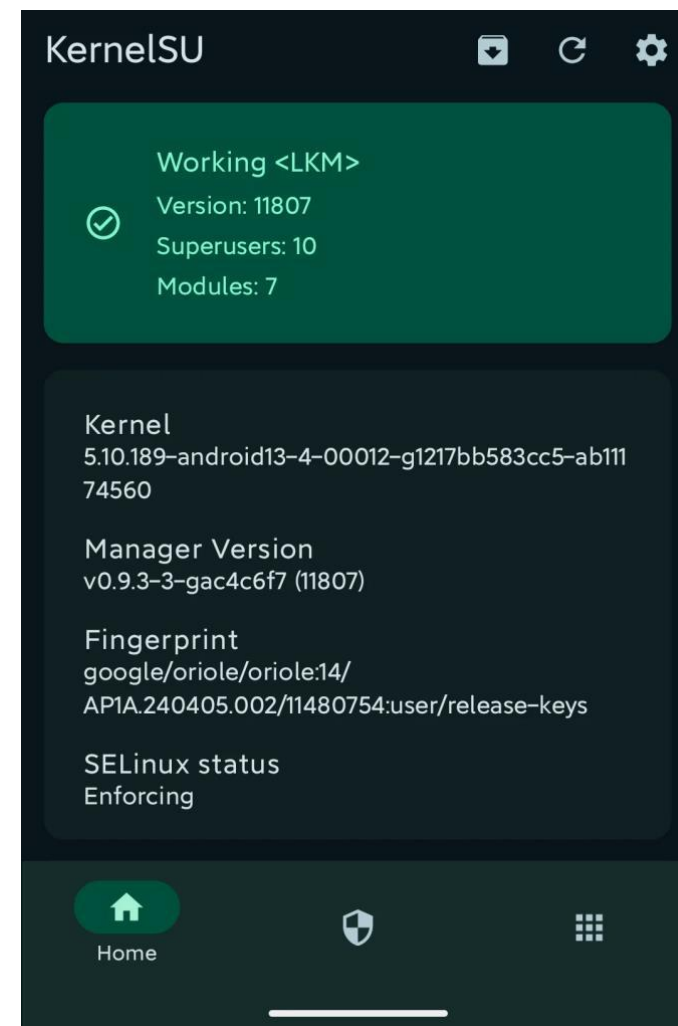
### GKI —

используется ядро  
от KernelSU

## 02

### LKM —

используется  
специальный  
модуль ядра  
без изменений  
оригинального  
ядра устройства



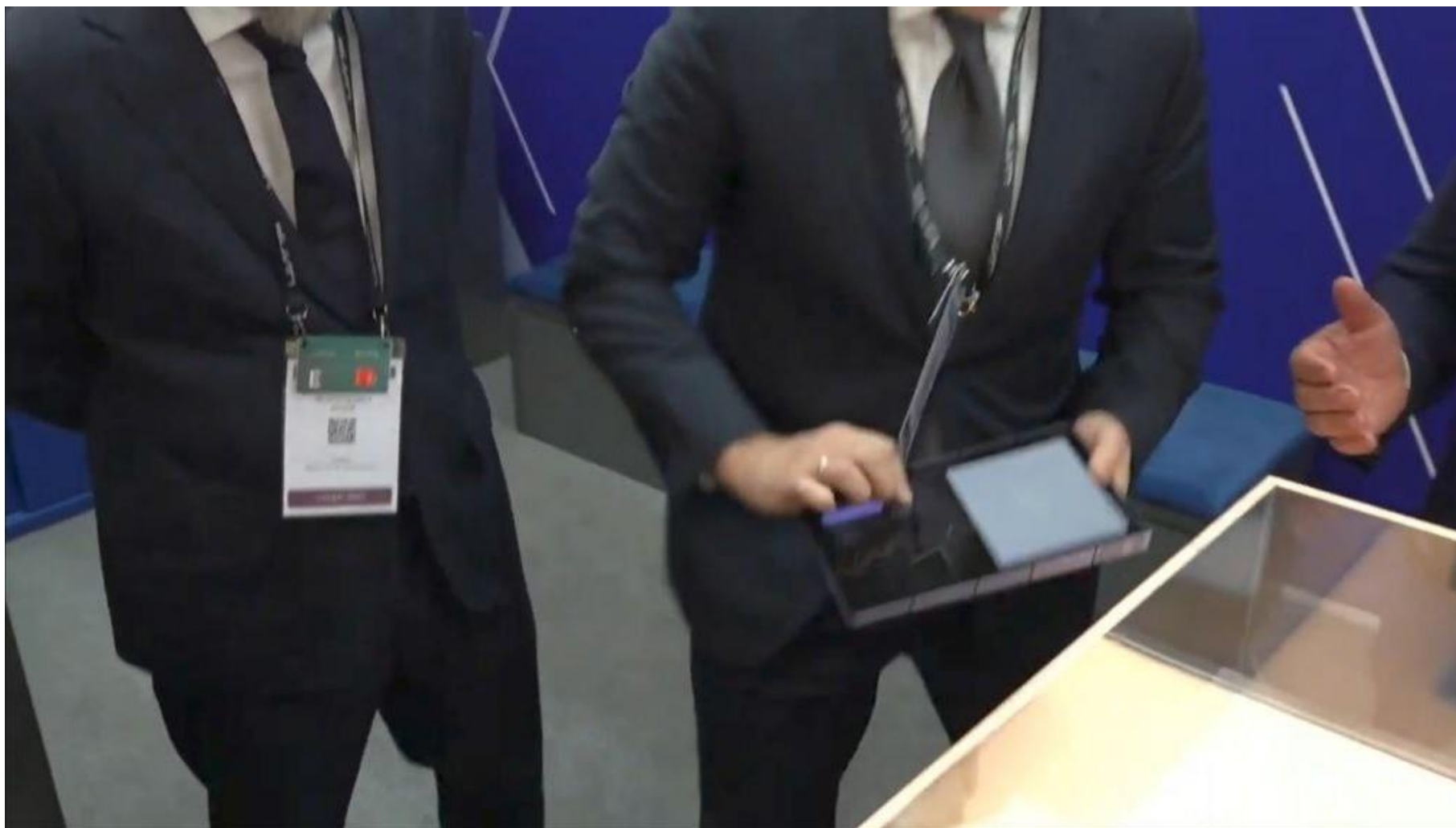
ROOT В ANDROID — ЗАДАЧА С МНОЖЕСТВОМ РЕШЕНИЙ

# Устройства с QuadraOS — неубиваемые



ROOT В ANDROID — ЗАДАЧА С МНОЖЕСТВОМ РЕШЕНИЙ

# Устройства с QuadraOS — неубиваемые



ROOT В ANDROID — ЗАДАЧА С МНОЖЕСТВОМ РЕШЕНИЙ

## Устройства с KvadraOS — неубиваемые

На устройствах с KvadraOS  
невозможно установить ROOT  
и другие модификации ОС Android



Что мы знаем о безопасности мобильных приложений

Зачем нужен ROOT — о правах в Linux-based ОС

SELinux — применение в Android

ROOT в Android — задача с множеством решений

Шалость удалась — о ROOT глазами хакера

---

Trusted Execution Environment — помощник в защите данных

Hardware Attestation — надежная проверка Android-устройства

eFUSE — как физики помогли сделать устройства безопаснее

Android Verified Boot — от кнопки включения до экрана блокировки

Все о FRIDA — использование и обнаружение



# Что можно делать с ROOT? Все что угодно!

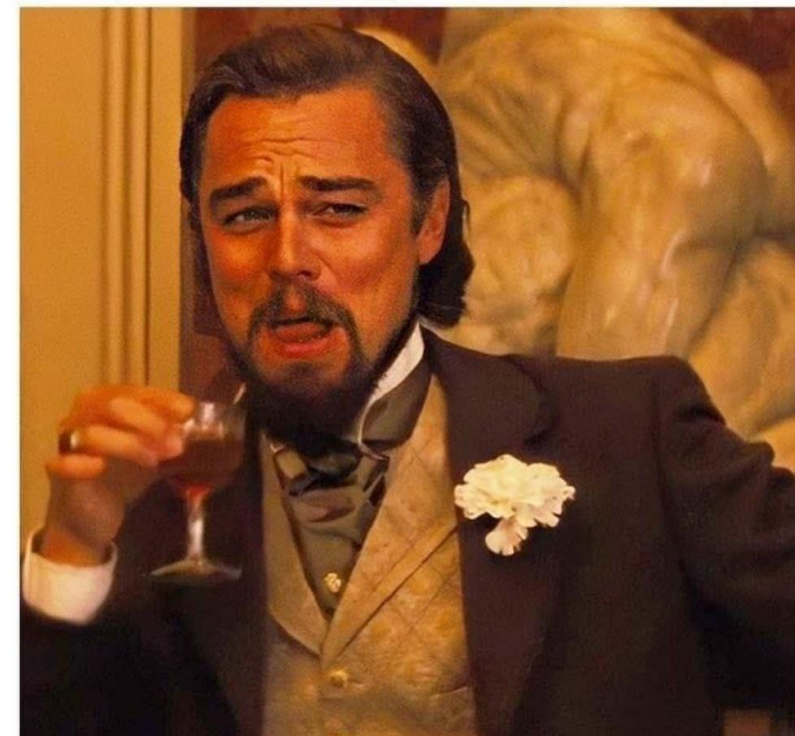
## 01

Модификация приложений в реальном времени:

- FRIDA
- LuckyPatcher
- LSPosed

App: your device is rooted and unsupported

me with magisk hide:



## 02

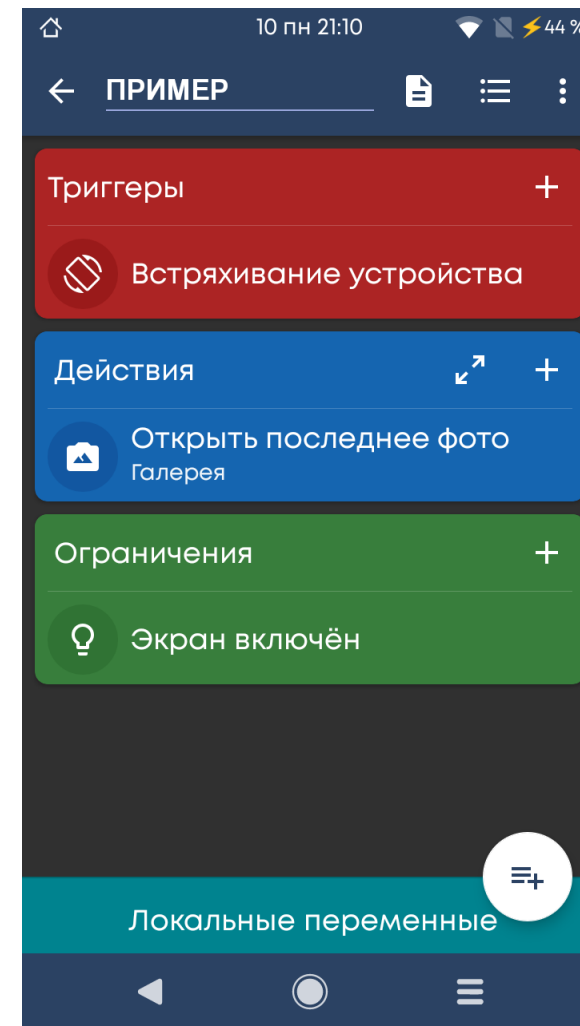
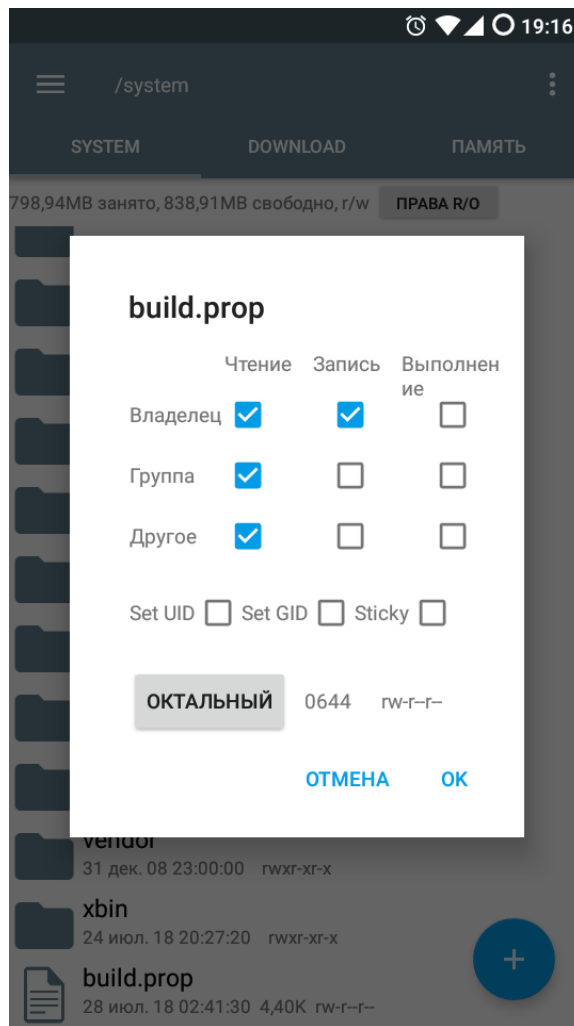
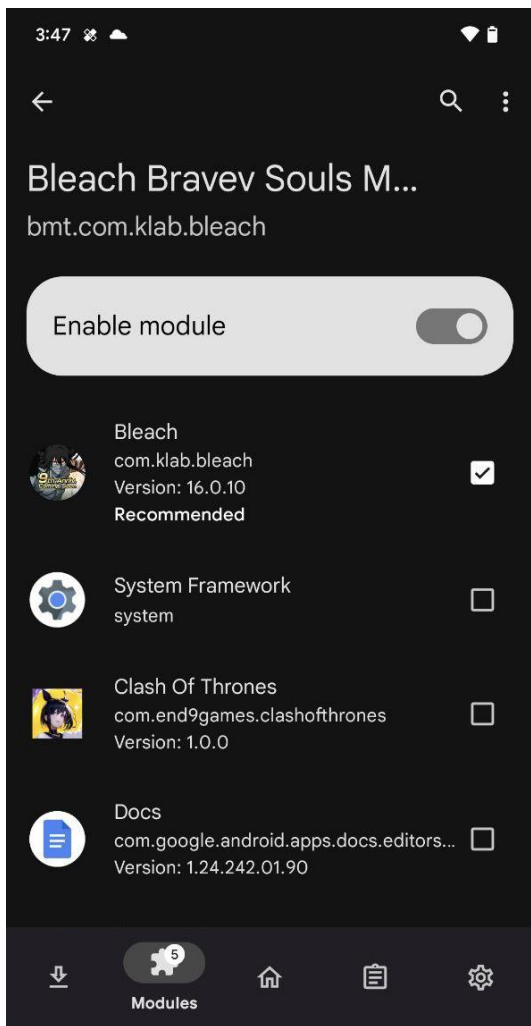
Изменение системных файлов и разделов

## 03

Использование низкоуровневых API в приложениях от сторонних разработчиков



# Приложения, использующие ROOT

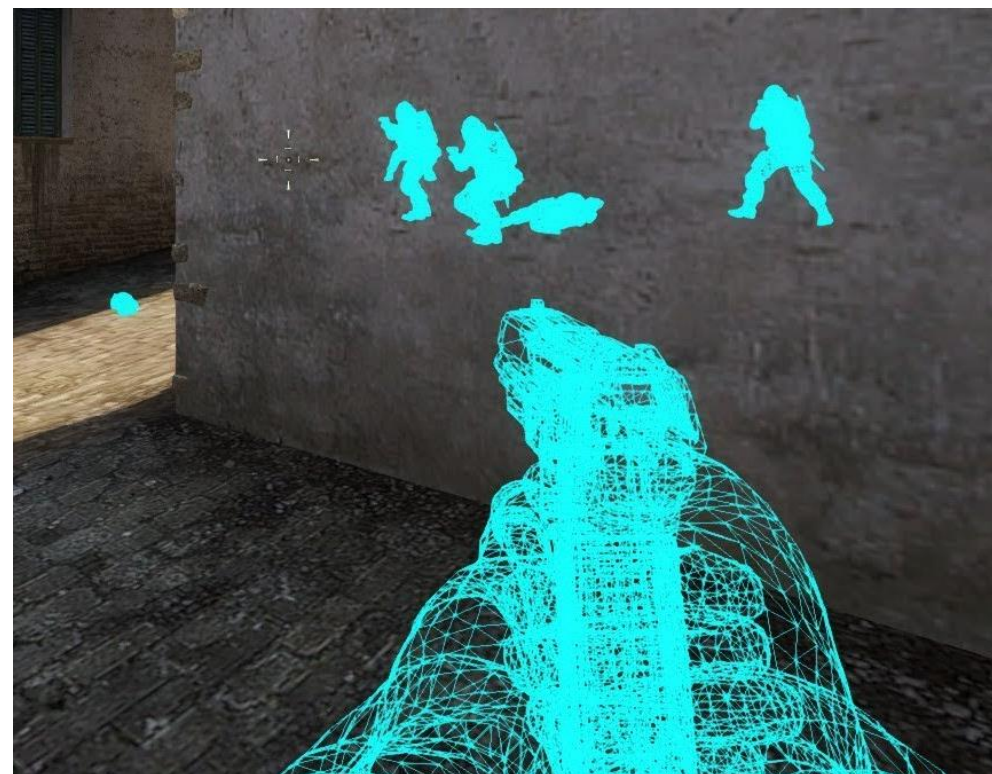
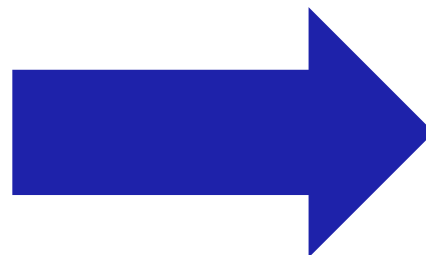
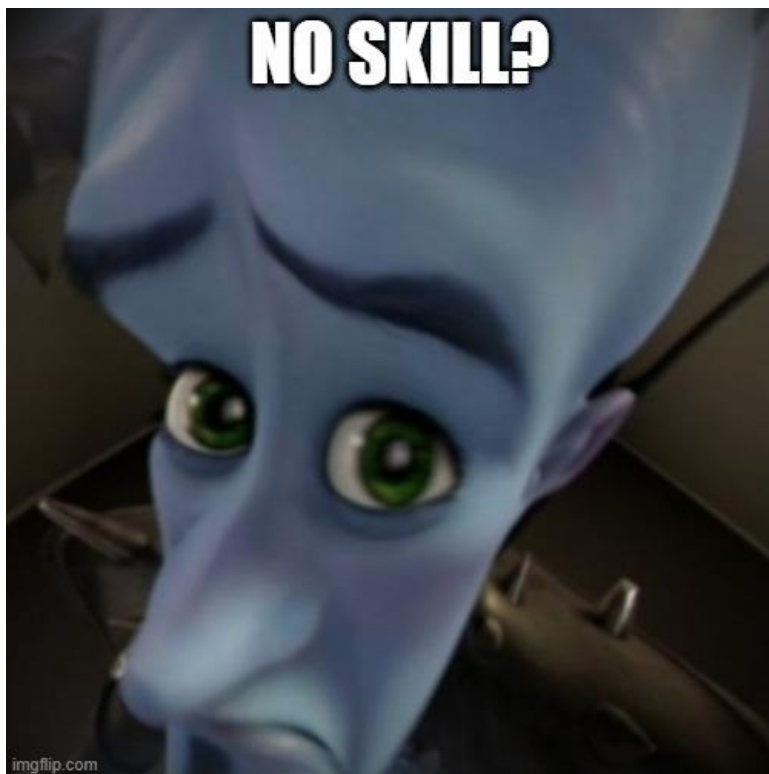




# Зачем люди ставят ROOT? И чем это грозит?

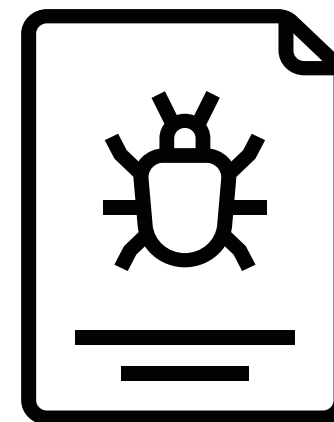
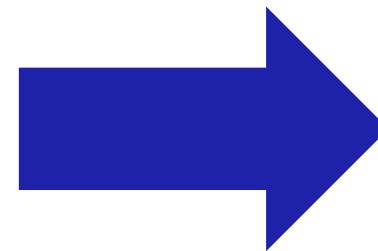
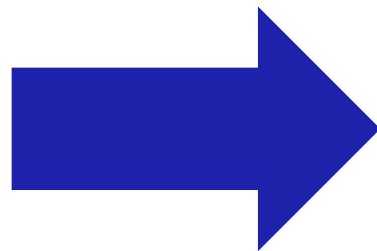


# Зачем люди ставят ROOT? И чем это грозит?



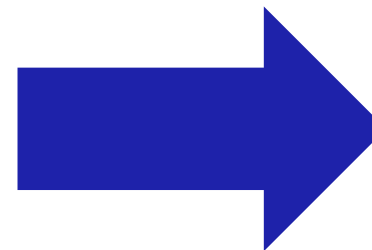
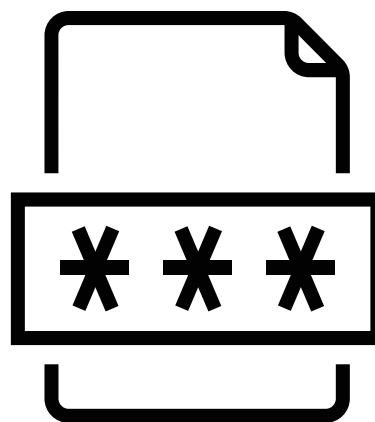
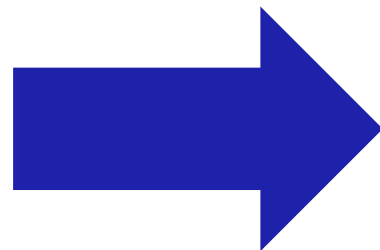
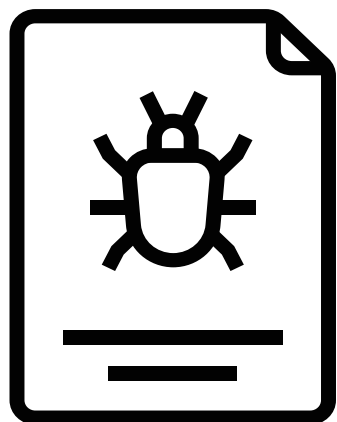


# Зачем люди ставят ROOT? И чем это грозит?





# Зачем люди ставят ROOT? И чем это грозит?



Что мы знаем о безопасности мобильных приложений

Зачем нужен ROOT — о правах в Linux-based ОС

SELinux — применение в Android

ROOT в Android — задача с множеством решений

Шалость удалась — о ROOT глазами хакера

Trusted Execution Environment — помощник в защите данных

---

Hardware Attestation — надежная проверка Android-устройства

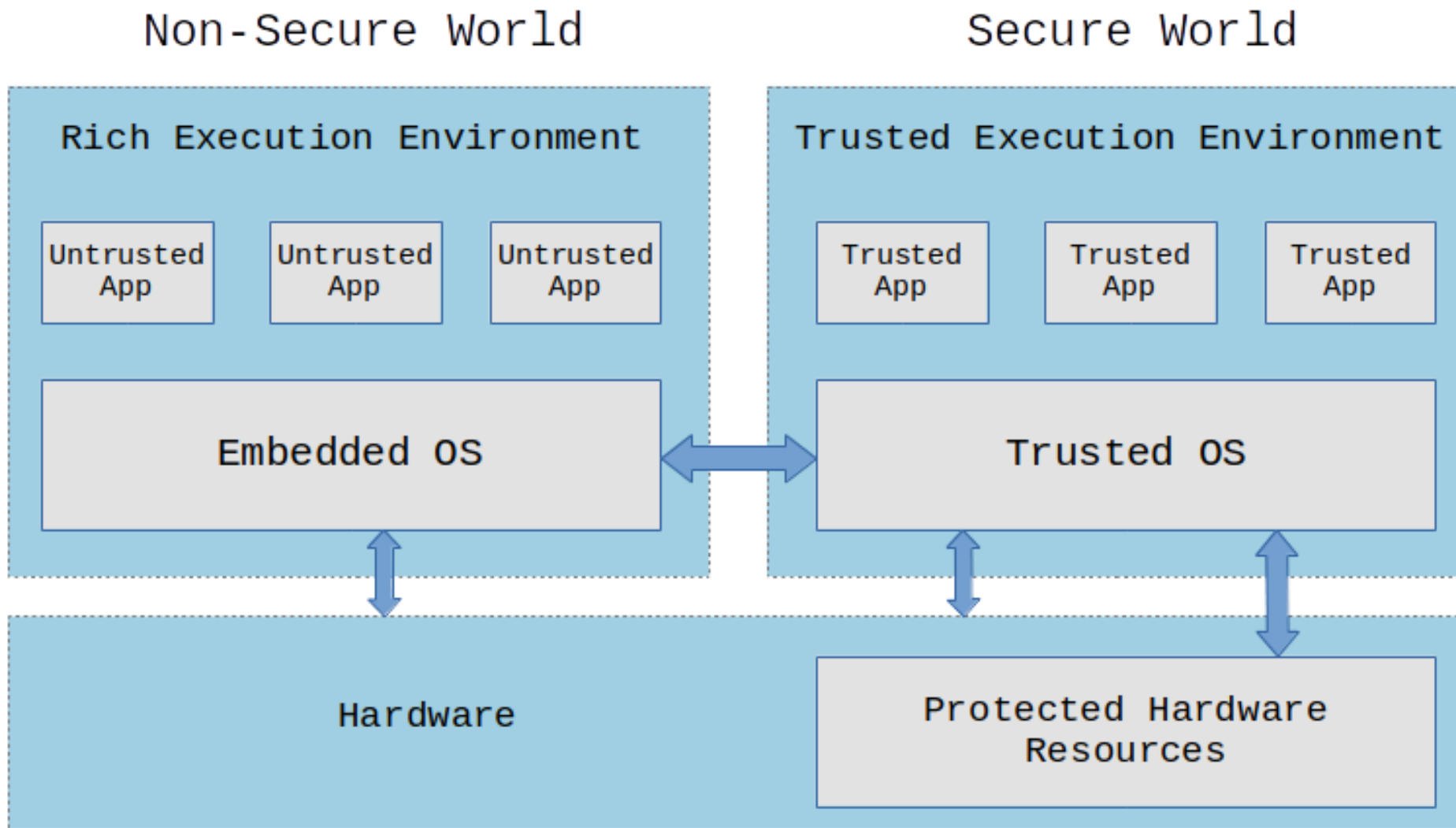
eFUSE — как физики помогли сделать устройства безопаснее

Android Verified Boot — от кнопки включения до экрана блокировки

Все о FRIDA — использование и обнаружение

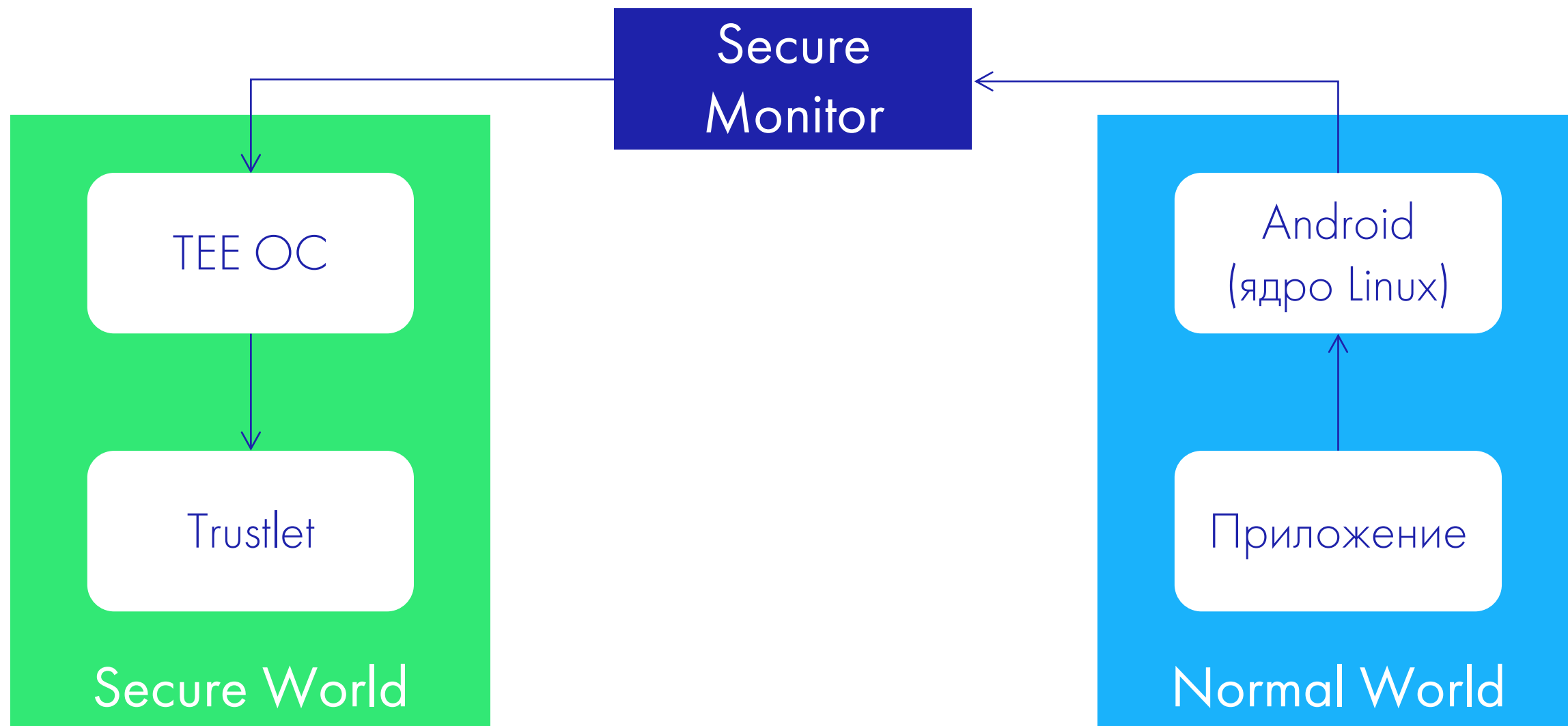


# Trusted Execution Environment (TEE)





# Взаимодействие Trusted Execution Environment и Android





# Trusted Execution Environment и сторонние приложения

## 01

Используйте для работы с криптографией **только те API**,  
которые работают с **Android Keystore**  
[developer.android.com/privacy-and-security/keystore](https://developer.android.com/privacy-and-security/keystore)



## 02

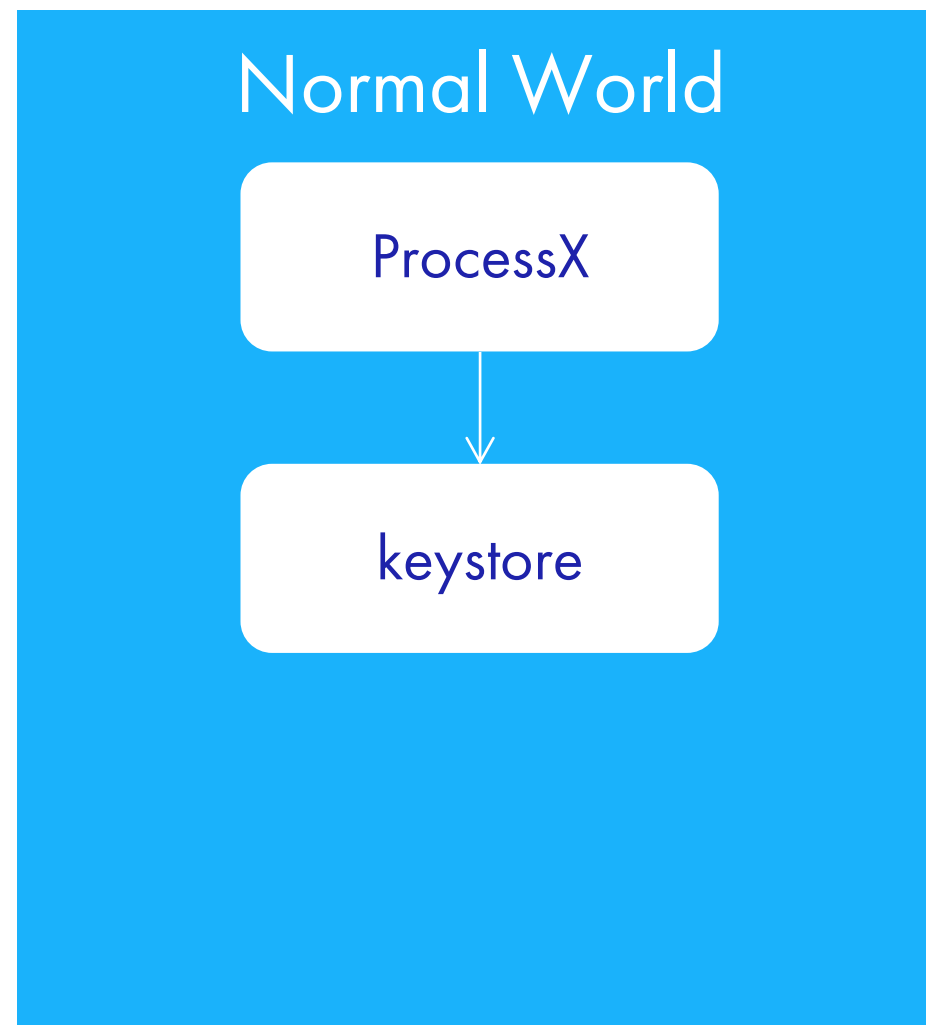
Используйте биометрическую аутентификацию  
[source.android.com/docs/security/features/biometric](https://source.android.com/docs/security/features/biometric)





# Процесс аутентификации с помощью биометрии

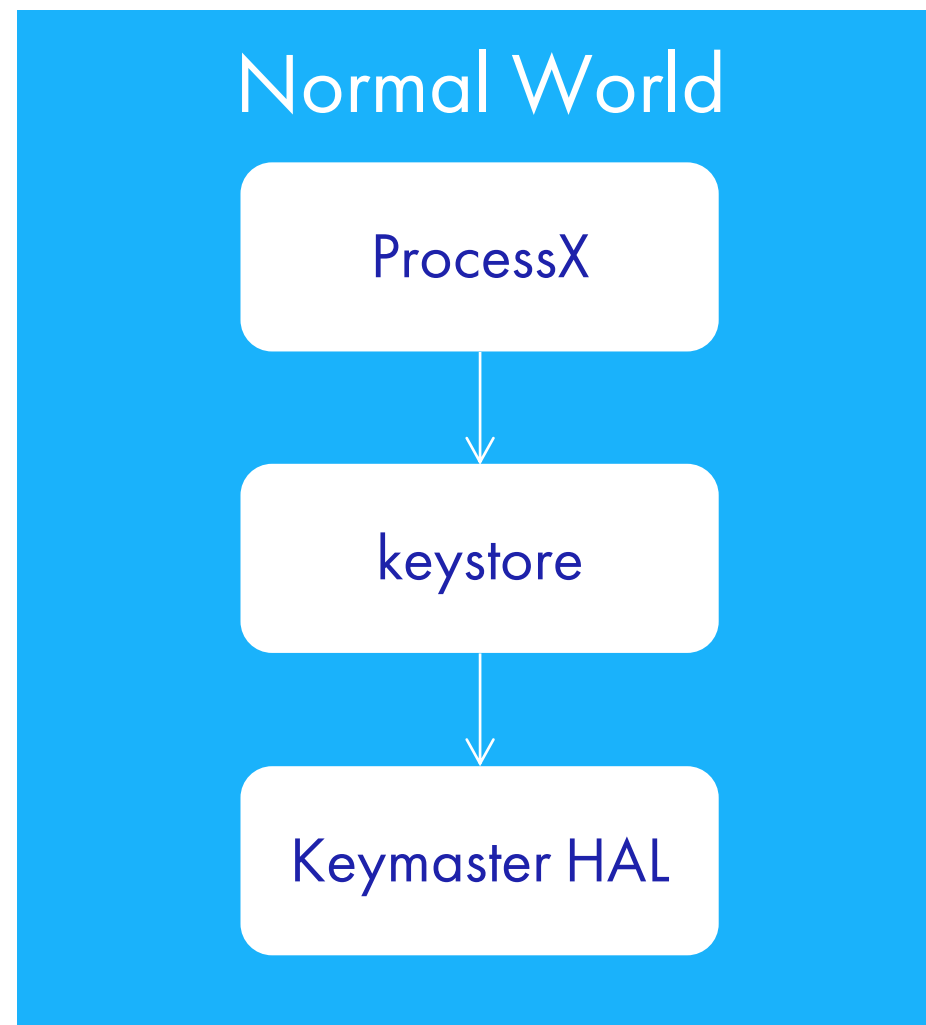
1. Процесс ProcessX вызывает системный процесс **keystore**





## Процесс аутентификации с помощью биометрии

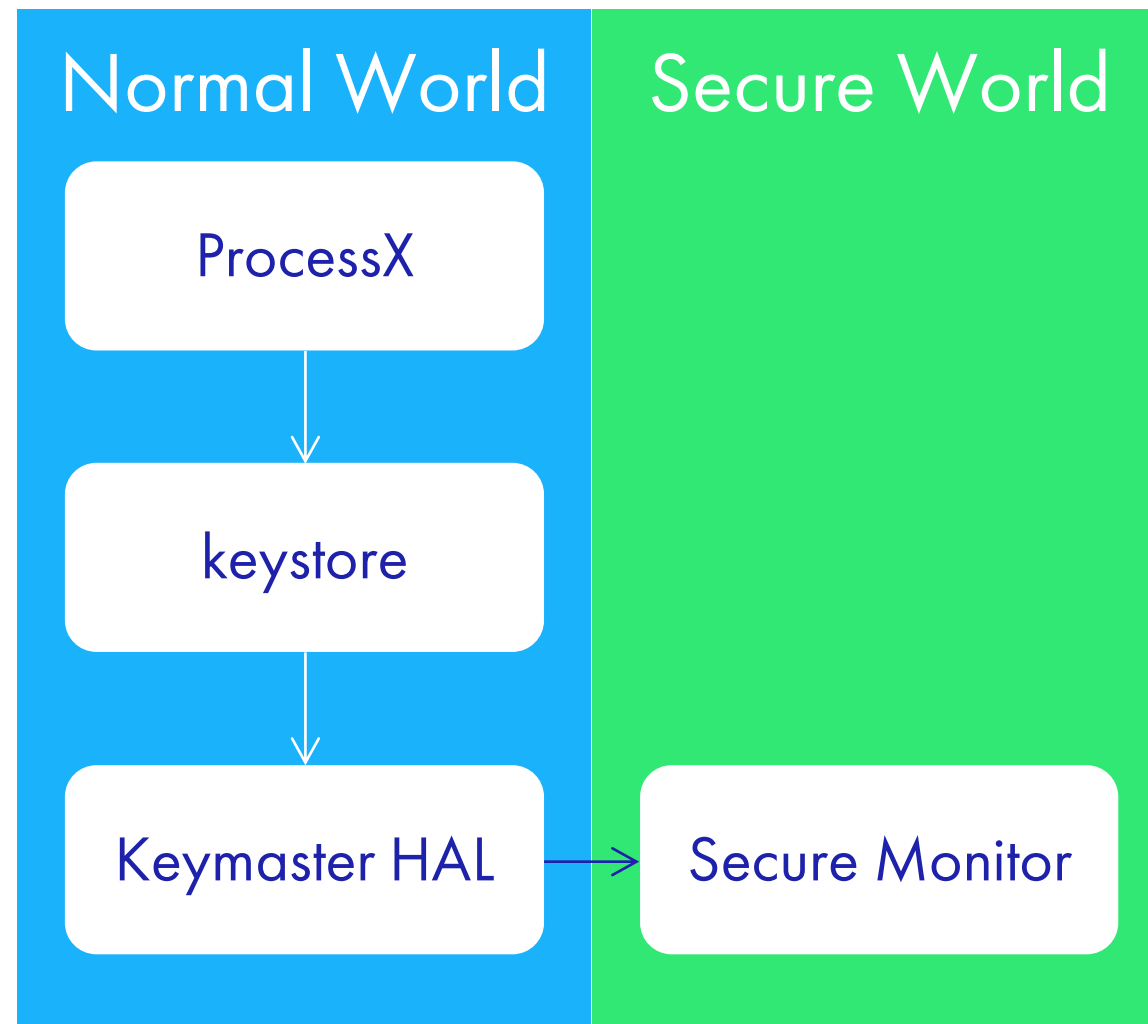
2. keystore вызывает библиотеку Keymaster HAL





## Процесс аутентификации с помощью биометрии

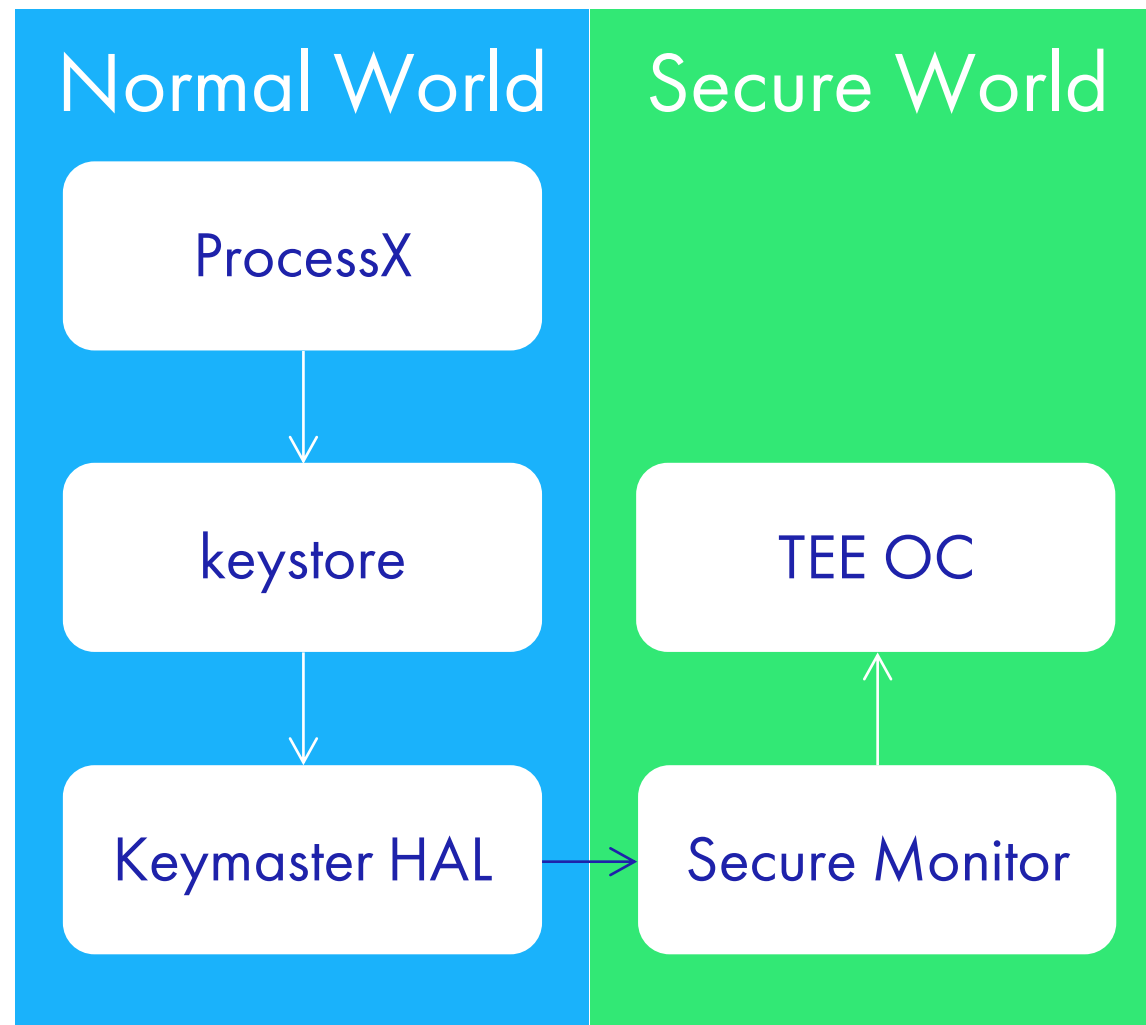
3. Keystore HAL, используя специальные интерфейсы ядра Linux обращается к **Secure Monitor**





# Процесс аутентификации с помощью биометрии

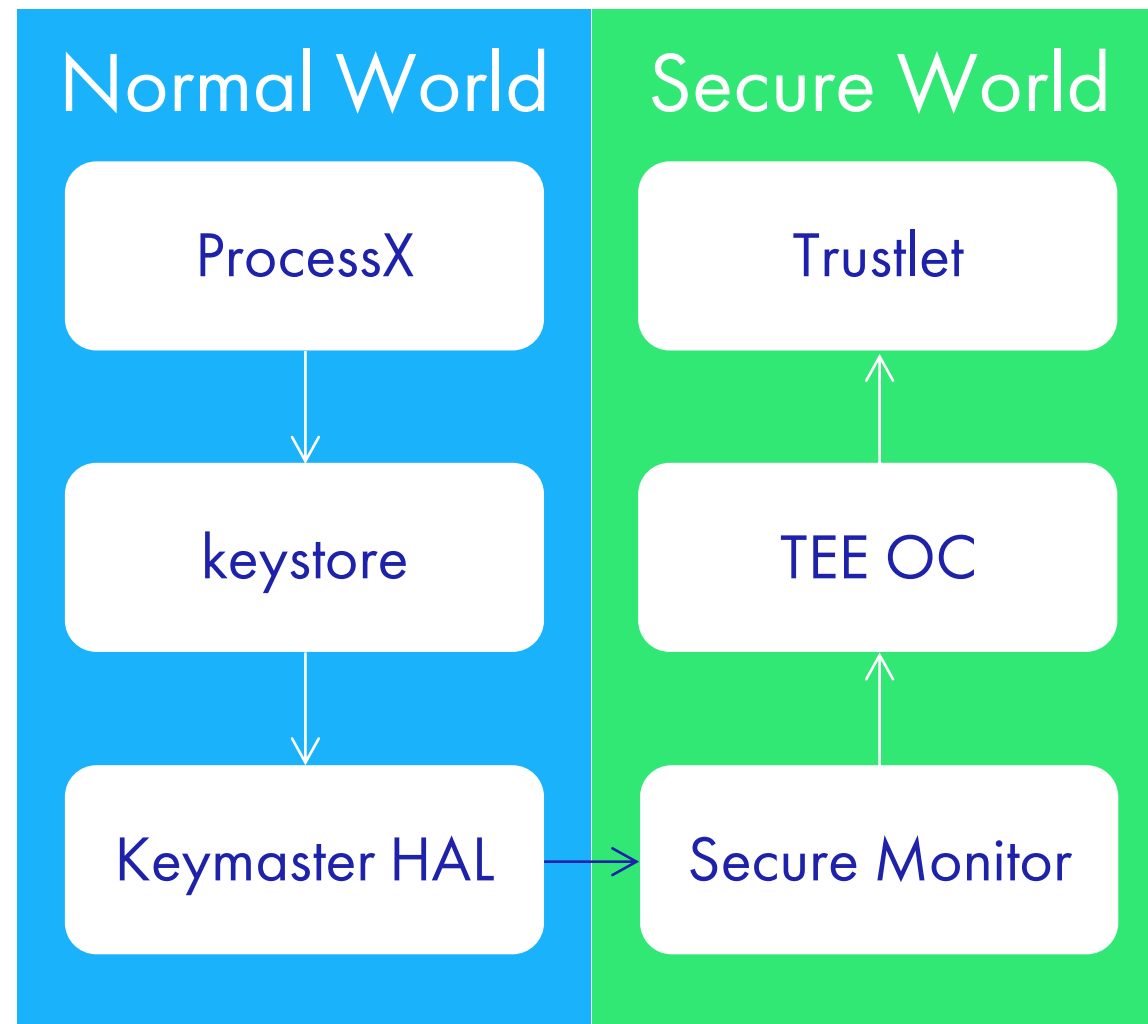
4. Secure Monitor  
обращается к TEE ОС





## Процесс аутентификации с помощью биометрии

5. TEE ОС запускает **Trustlet**, который будет работать с сканером и хранилищем биометрических данных



## KvadraOS и Trusted Execution Environment

- Есть поддержка работы с Trusted Execution Environment через API из Android SDK
- Биометрическая аутентификация работает по стандарту, который был описан ранее
- Не нужны изменения для поддержки KvadraOS





# Как можно проверить устройство на ROOT?

**ДАВАЙТЕ ДУМАТЬ**



**ПОДСКАЗЫВАЙТЕ**

Что мы знаем о безопасности мобильных приложений

Зачем нужен ROOT — о правах в Linux-based ОС

SELinux — применение в Android

ROOT в Android — задача с множеством решений

Шалость удалась — о ROOT глазами хакера

Trusted Execution Environment — помощник в защите данных

Hardware Attestation — надежная проверка Android-устройства

---

eFUSE — как физики помогли сделать устройства безопаснее

Android Verified Boot — от кнопки включения до экрана блокировки

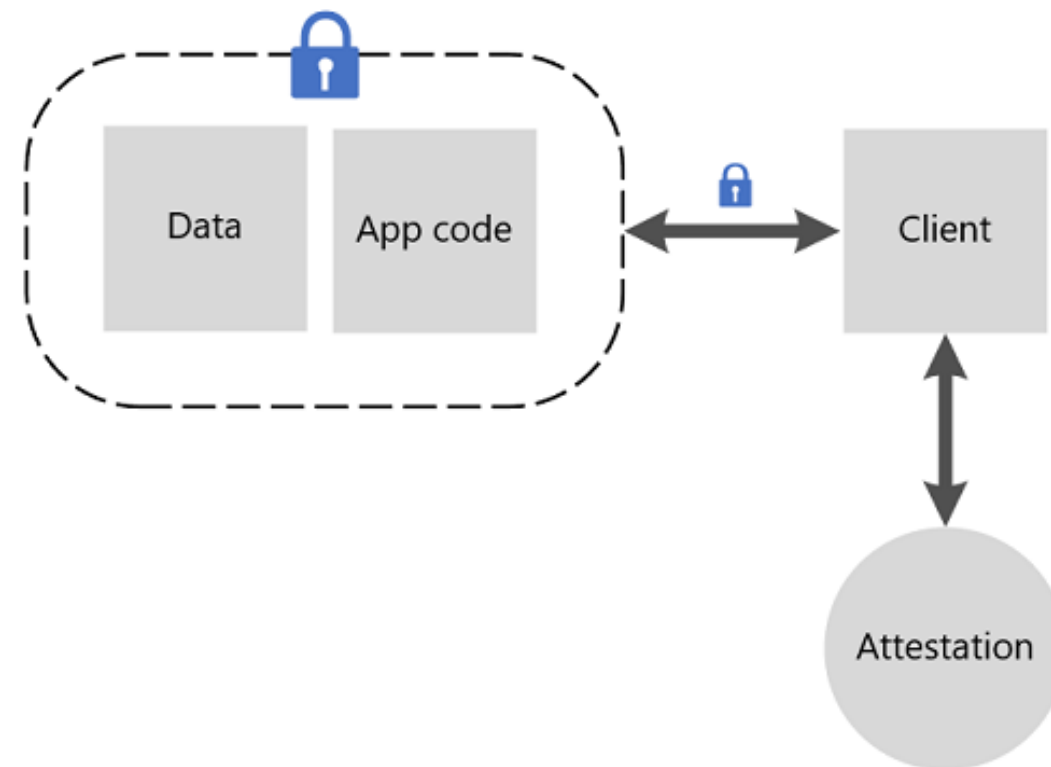
Все о FRIDA — использование и обнаружение



# Hardware Attestation

01

Подтверждение целостности образа ОС на устройстве





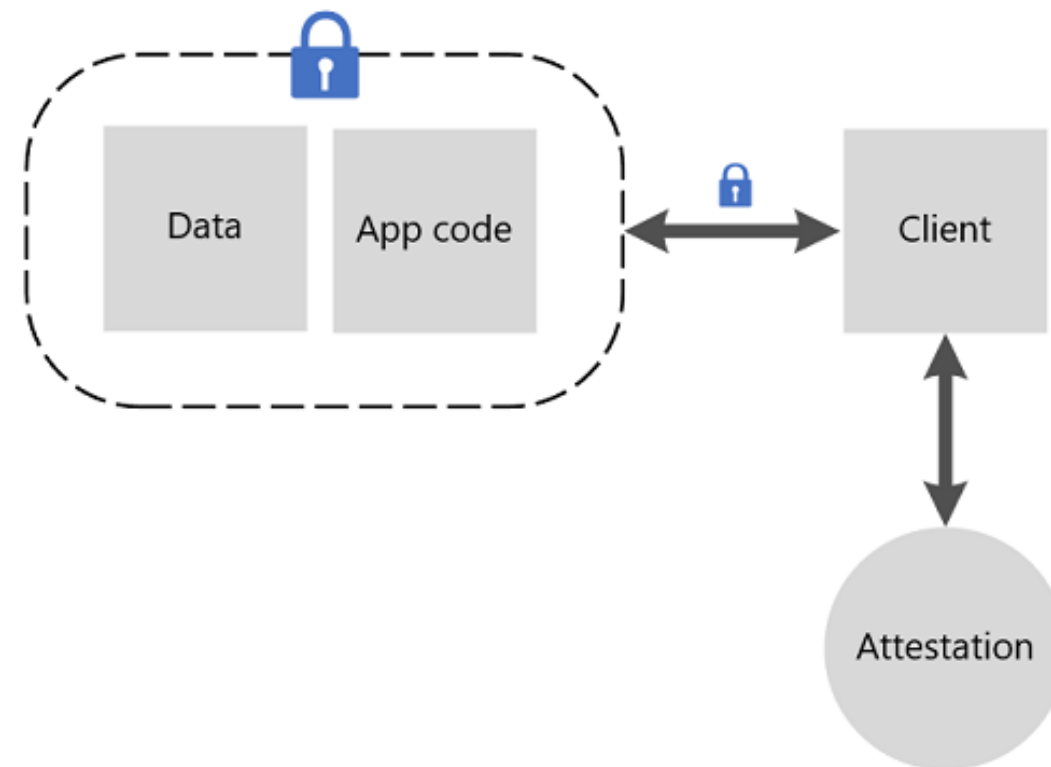
# Hardware Attestation

01

Подтверждение целостности образа ОС на устройстве

02

Проверка информации об устройстве осуществляется на сервере





# Hardware Attestation

01

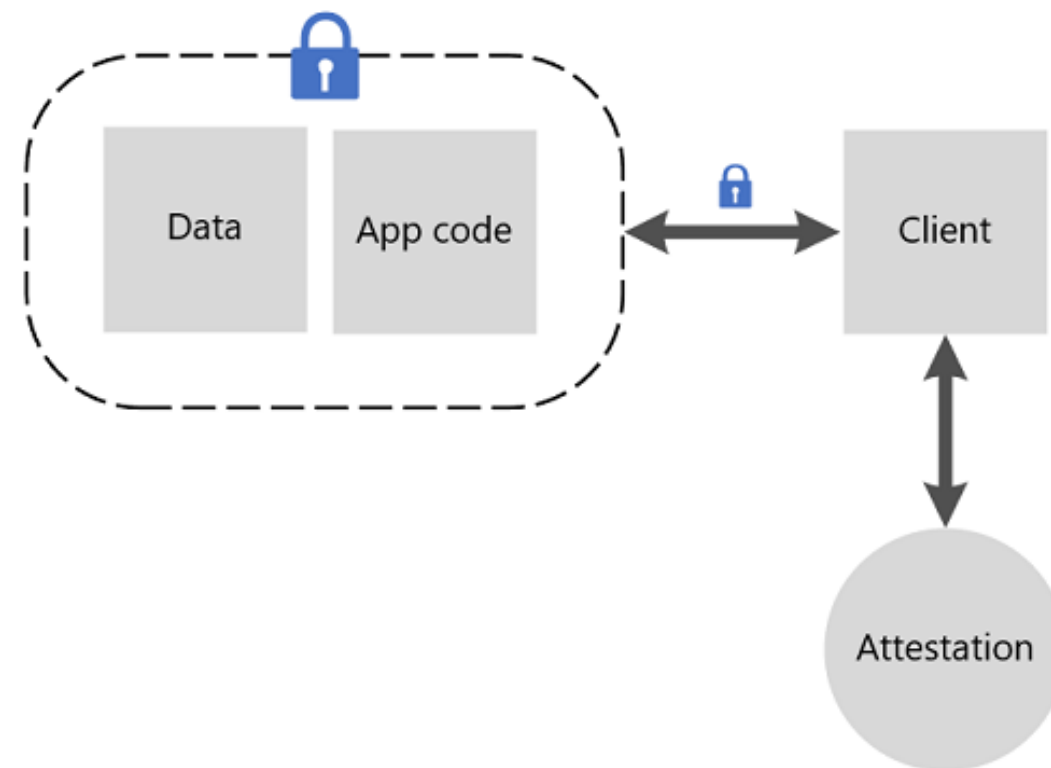
Подтверждение целостности образа ОС на устройстве

02

Проверка информации об устройстве осуществляется на сервере

03

Информация об устройстве — цепочка сертификатов





# Hardware Attestation

01

Подтверждение целостности образа ОС на устройстве

02

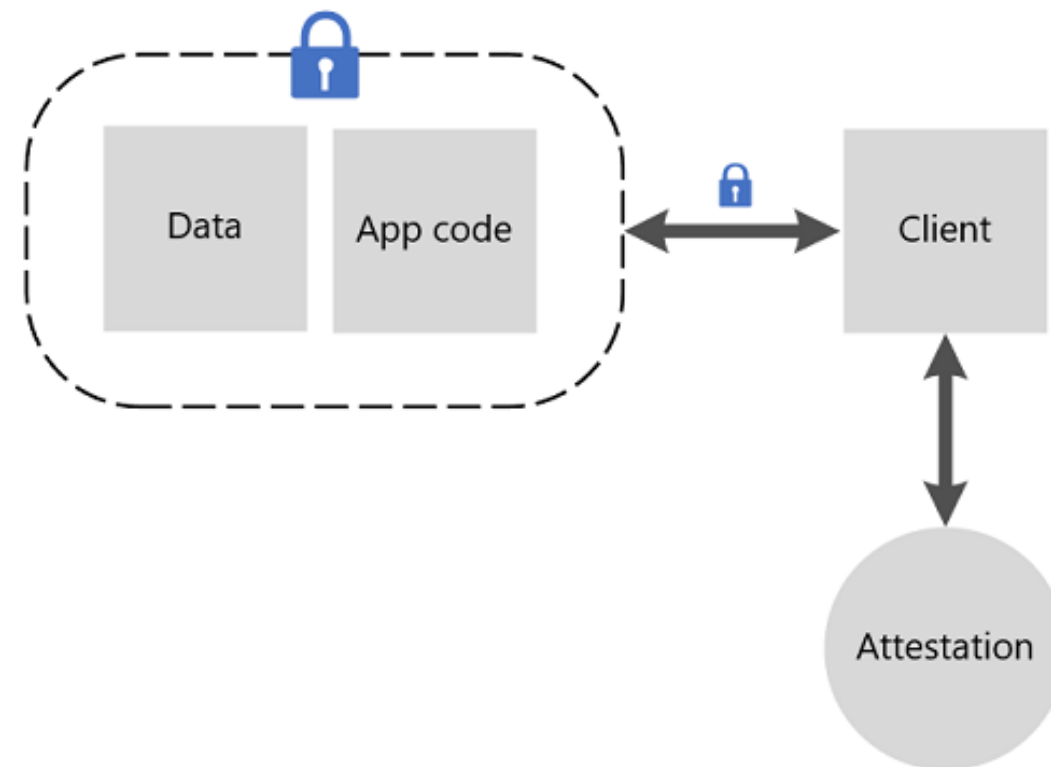
Проверка информации об устройстве осуществляется на сервере

03

Информация об устройстве — цепочка сертификатов

04

При создании цепочки используется Trusted Execution Environment





## В чем польза Hardware Attestation?

Вы можете быть уверены — устройство клиента является безопасным:

- Устройство реально существует = **не эмулятор**
- На устройстве **нет ROOT** и **других модификации Android**

**Небезопасные устройства не смогут пройти аттестацию,** так как подпись или данные сертификатов не будут совпадать с эталоном



# Кто поддерживает Hardware Attestation?

01 Samsung

[docs.samsungknox.com/dev/knox-attestation](https://docs.samsungknox.com/dev/knox-attestation)



02 Huawei

03 Xiaomi

04 Устройства с Google Mobile Services (GMS)



# Кто поддерживает Hardware Attestation?

01 Samsung

02 Huawei

[github.com/HMS-Core/hms-safetydetect-demo-kotlin/tree/master/SafetyDetect-SysIntegrity-Kotlin-Demo](https://github.com/HMS-Core/hms-safetydetect-demo-kotlin/tree/master/SafetyDetect-SysIntegrity-Kotlin-Demo)



03 Xiaomi

04 Устройства с Google Mobile Services (GMS)



# Кто поддерживает Hardware Attestation?

01 Samsung

02 Huawei

03 Xiaomi

[trust.mi.com/docs/miui-security-white-paper-global/2/1](https://trust.mi.com/docs/miui-security-white-paper-global/2/1)



04 Устройства с Google Mobile Services (GMS)



## Устройства *GMS* и *Hardware Attestation*

С *Android 8.0* все устройства с *GMS* должны поддерживать *Hardware Attestation* и *Android Verified Boot*.

Цепочка сертификатов должна быть подписана Google.

Root Of Trust:

Verified Boot Key: xdPHG8cNW0PgQJyp2bNMDbrB0vCaXeLIpLjwkPGSaWU=

Device Locked: true

Состояние Bootloader

Verified Boot State: VERIFIED

Verified Boot Hash: r5Ir4dRlyPyIrmVuDz4i0z3YL3nsgnhUB86oT0z00TA=

OS Version: 120000

Состояние Android Verified Boot



# Устройства GMS и Hardware Attestation

## 01

Библиотека по работе с цепочкой сертификатов для сервера  
[github.com/google/android-key-attestation](https://github.com/google/android-key-attestation)



## 02

Получение цепочки сертификатов для аттестации в Android-приложении  
[github.com/vvb2060/KeyAttestation](https://github.com/vvb2060/KeyAttestation)





## Что мы узнали?

# 01

Hardware Attestation — надежный способ проверить безопасность Android-устройств

# 02

Hardware Attestation основан на аппаратных мерах безопасности — на применении Trusted Execution Environment

# 03

Для GMS устройств можно реализовать собственный сервер для реализации Hardware Attestation

Что мы знаем о безопасности мобильных приложений

Зачем нужен ROOT — о правах в Linux-based ОС

SELinux — применение в Android

ROOT в Android — задача с множеством решений

Шалость удалась — о ROOT глазами хакера

Trusted Execution Environment — помощник в защите данных

Hardware Attestation — надежная проверка Android-устройства

eFUSE — как физики помогли сделать устройства безопаснее

---

Android Verified Boot — от кнопки включения до экрана блокировки

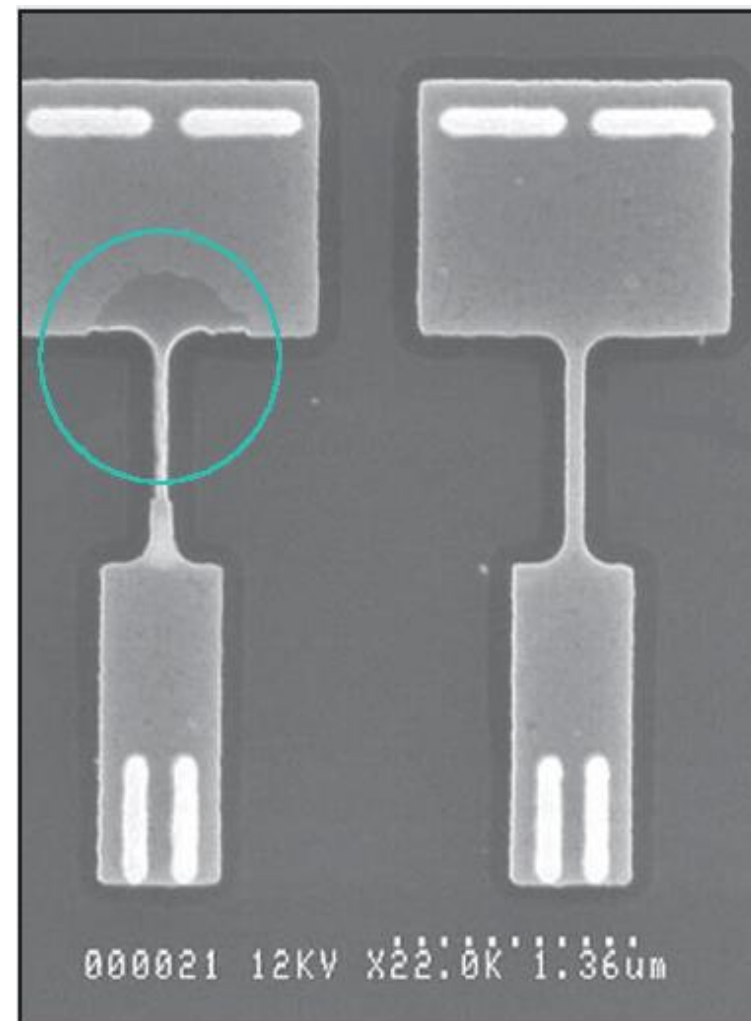
Все о FRIDA — использование и обнаружение



# eFUSE (Electronic fuse)

01

Основной компонент защиты  
от модификации образа ОС



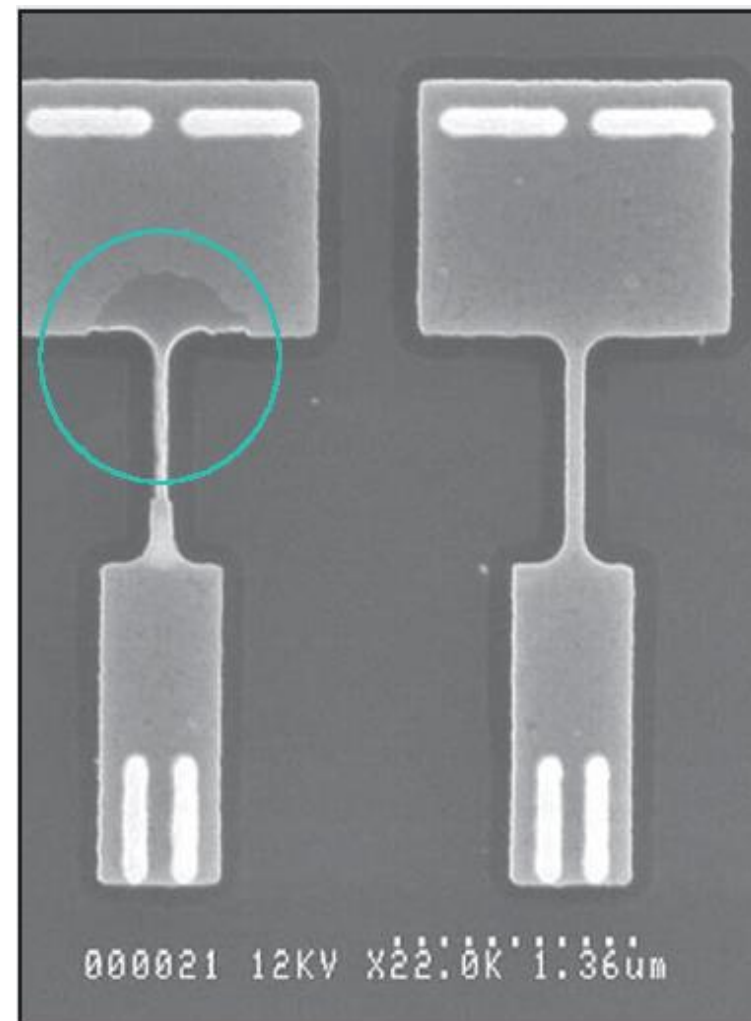
## eFUSE (Electronic fuse)

01

Основной компонент защиты  
от модификации образа ОС

02

Дорожки eFUSE повреждаются  
при воздействии напряжения  
больше номинального



## eFUSE (Electronic fuse)

01

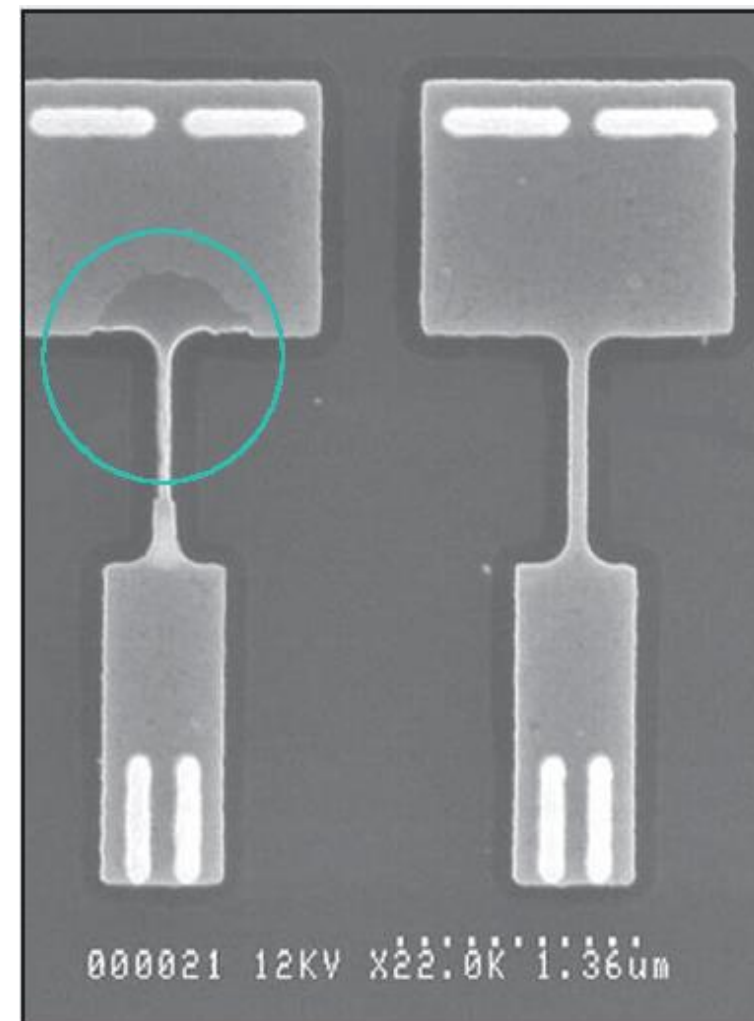
Основной компонент защиты от модификации образа ОС

02

Дорожки eFUSE повреждаются при воздействии напряжения больше номинального

03

Перегорание eFUSE не несет риска повреждения других устройств





## eFUSE (Electronic fuse)

01

Основной компонент защиты от модификации образа ОС

02

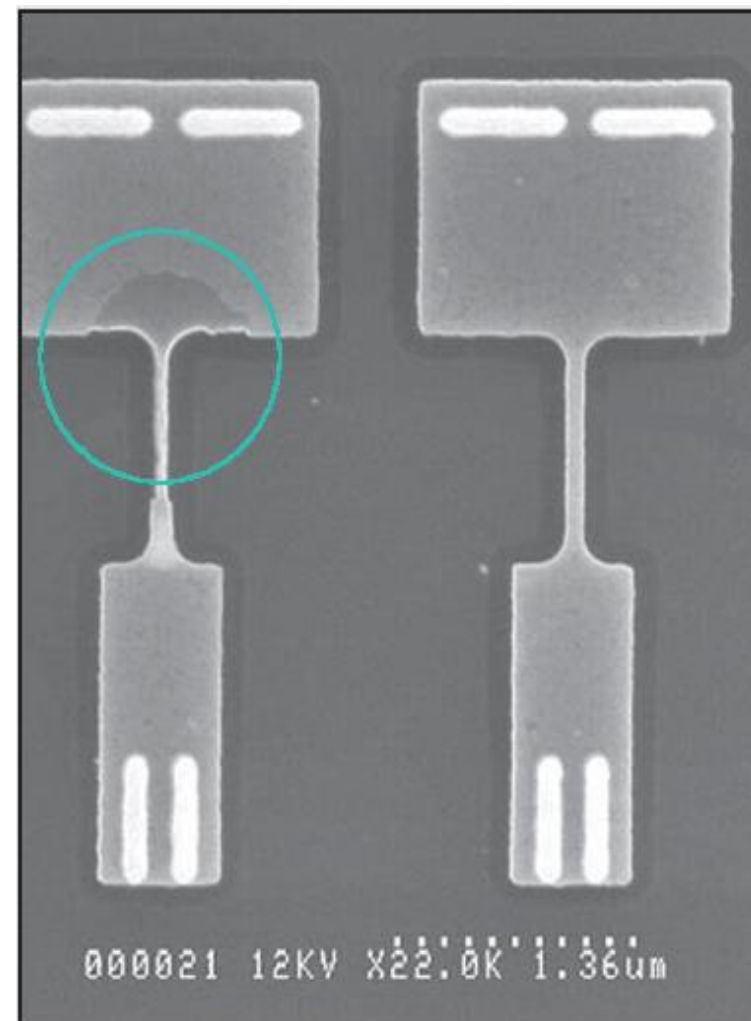
Дорожки eFUSE повреждаются при воздействии напряжения больше номинального

03

Перегорание eFUSE не несет риска повреждения других устройств

04

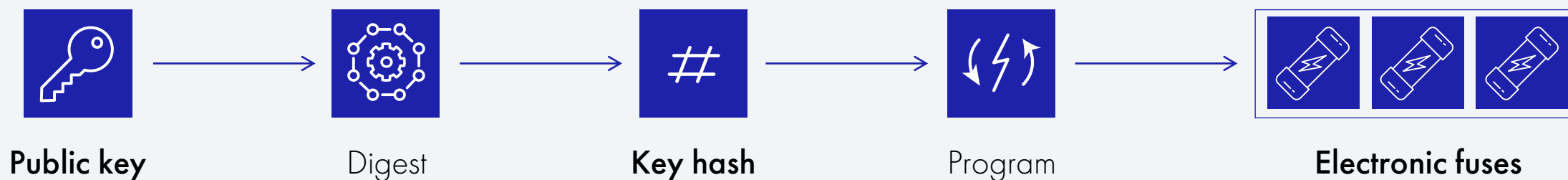
Для установки значений чаще всего используется программное решение





## eFUSE (Electronic fuse)

Информация записывается только 1 раз и затем не может быть изменена



Что мы знаем о безопасности мобильных приложений

Зачем нужен ROOT — о правах в Linux-based ОС

SELinux — применение в Android

ROOT в Android — задача с множеством решений

Шалость удалась — о ROOT глазами хакера

Trusted Execution Environment — помощник в защите данных

Hardware Attestation — надежная проверка Android-устройства

eFUSE — как физики помогли сделать устройства безопаснее

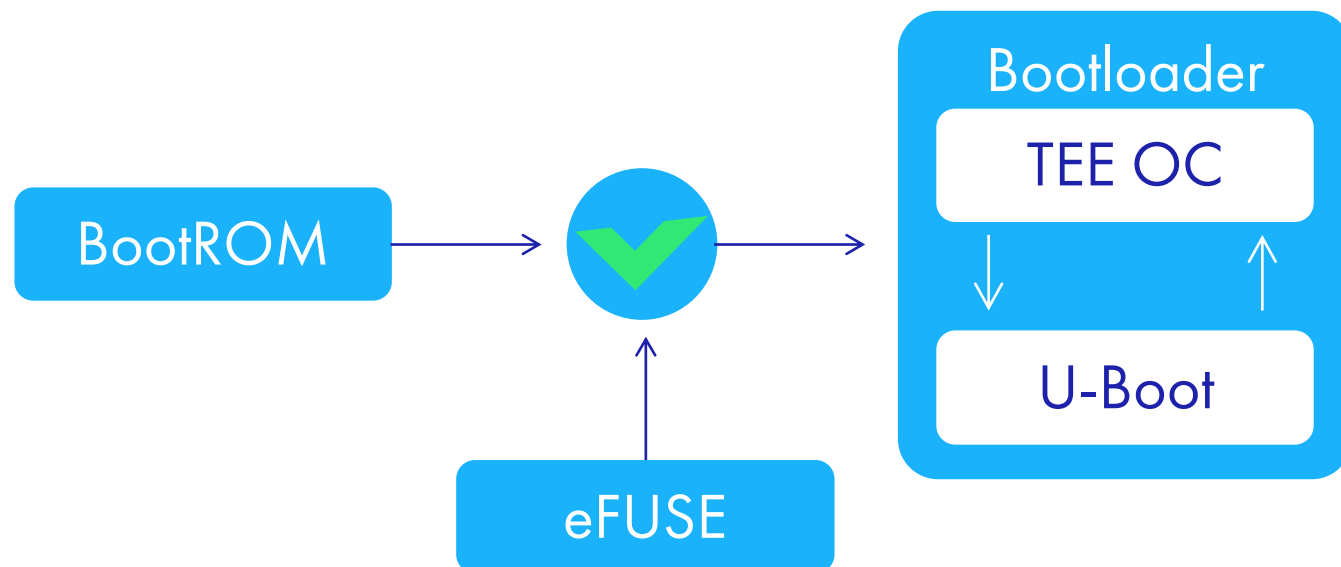
Android Verified Boot — от кнопки включения до экрана блокировки

---

Все о FRIDA — использование и обнаружение

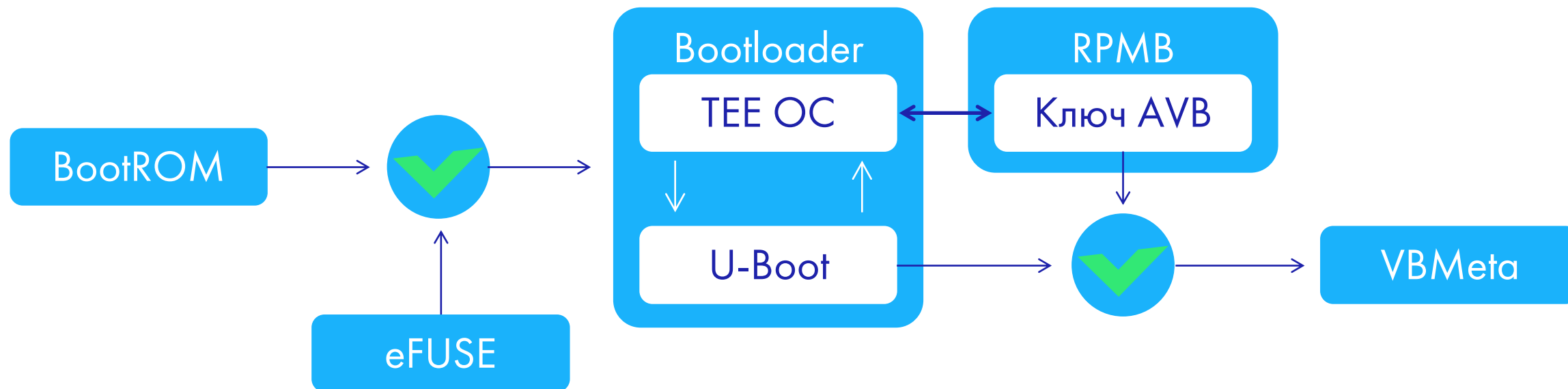


# Процесс запуска устройства при Android Verified Boot



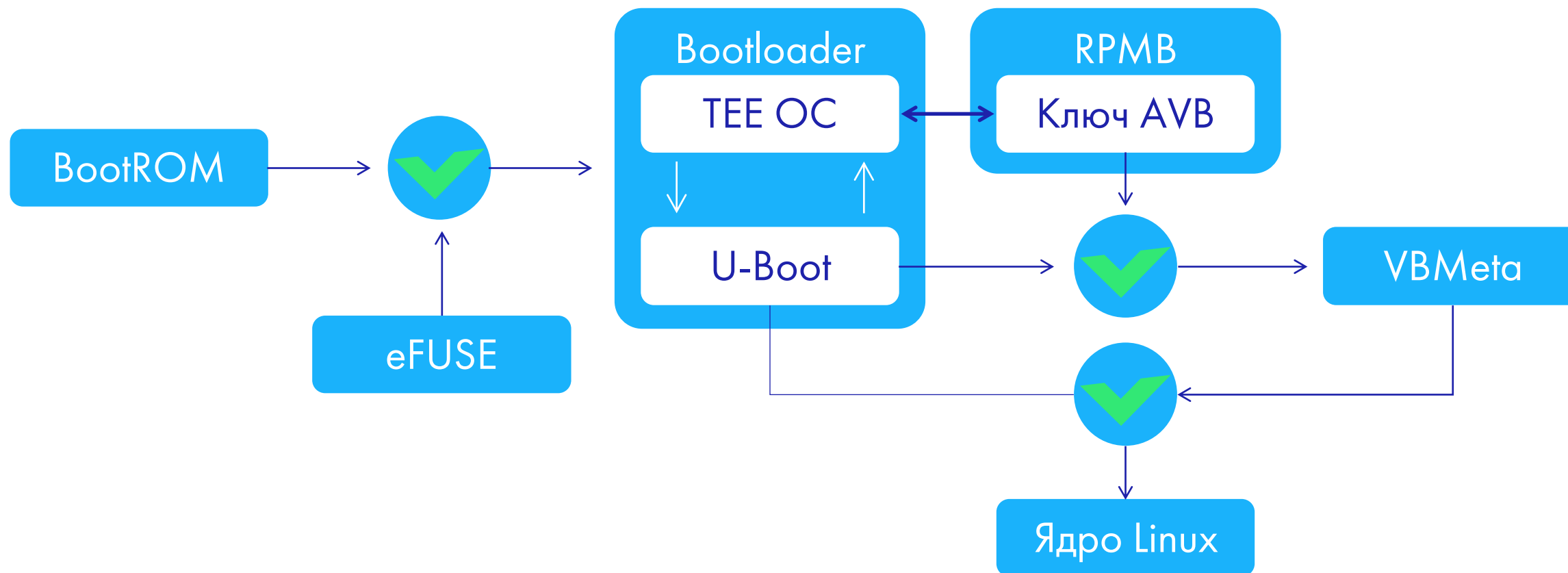


# Процесс запуска устройства при Android Verified Boot



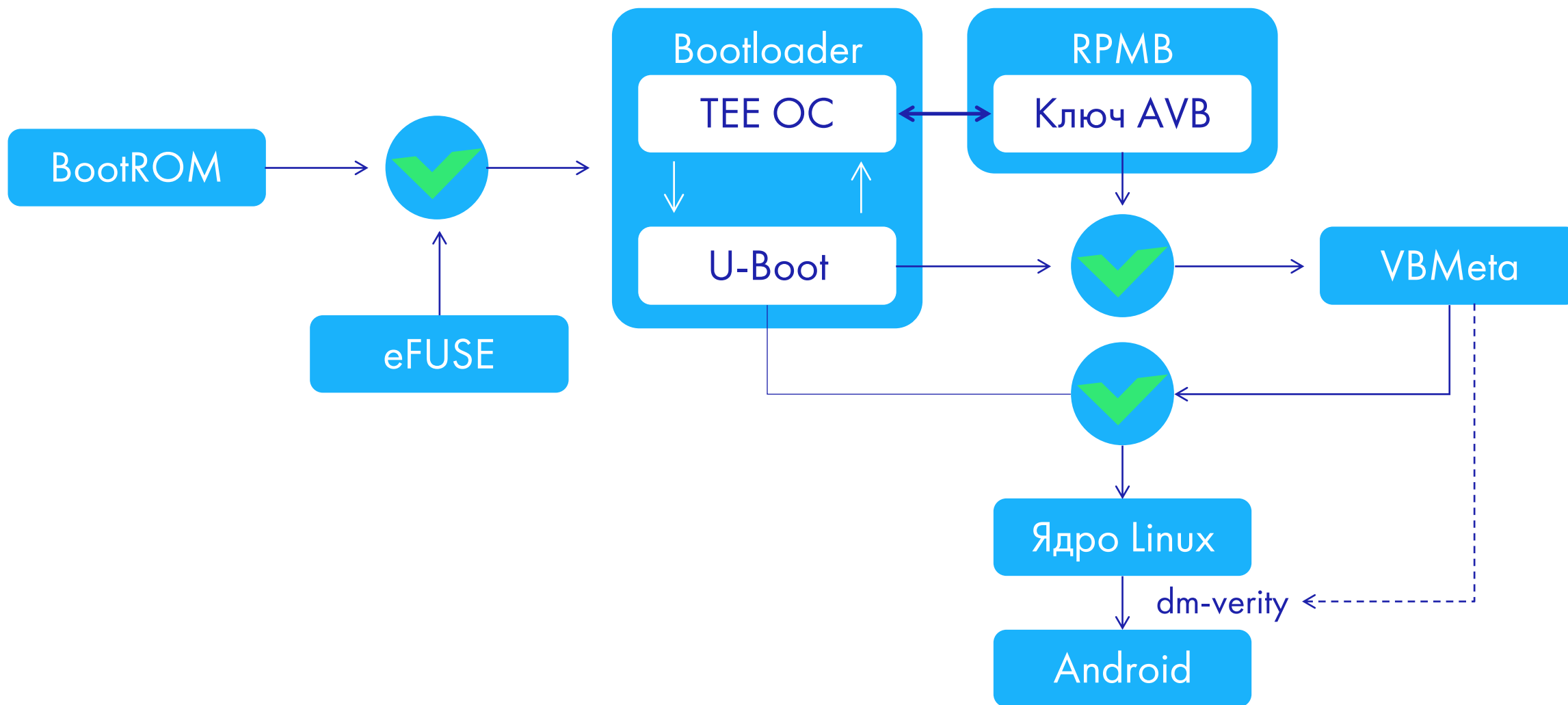


# Процесс запуска устройства при Android Verified Boot





# Процесс запуска устройства при Android Verified Boot





## Что мы узнали?

### 01

При использовании Android Verified Boot устройство включается только с образом Android от производителя

### 02

Android Verified Boot основан на аппаратных мерах безопасности — на применении Trusted Execution Environment и eFUSE

### 03

Android Verified Boot перестает работать при разблокированном Bootloader

## KvadraOS и Android Verified Boot

- Android Verified Boot активируется во время изготовления устройства
- Устройства попадают к пользователям с включенным Android Verified Boot
- Запрещена разблокировка Bootloader — нельзя отключить Android Verified Boot



Что мы знаем о безопасности мобильных приложений

Зачем нужен ROOT — о правах в Linux-based ОС

SELinux — применение в Android

ROOT в Android — задача с множеством решений

Шалость удалась — о ROOT глазами хакера

Trusted Execution Environment — помощник в защите данных

Hardware Attestation — надежная проверка Android-устройства

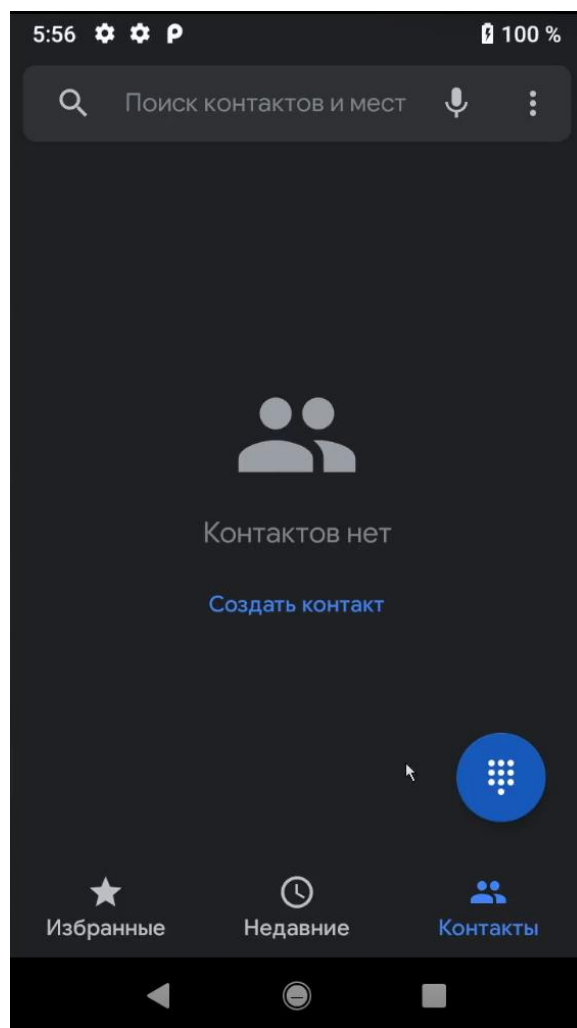
eFUSE — как физики помогли сделать устройства безопаснее

Android Verified Boot — от кнопки включения до экрана блокировки

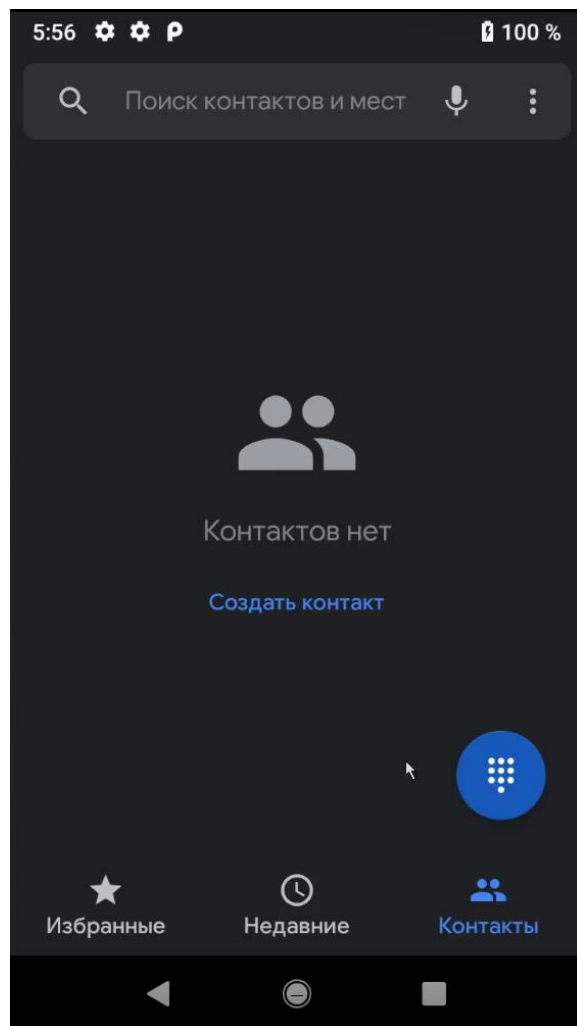
Все о FRIDA — использование и обнаружение

---

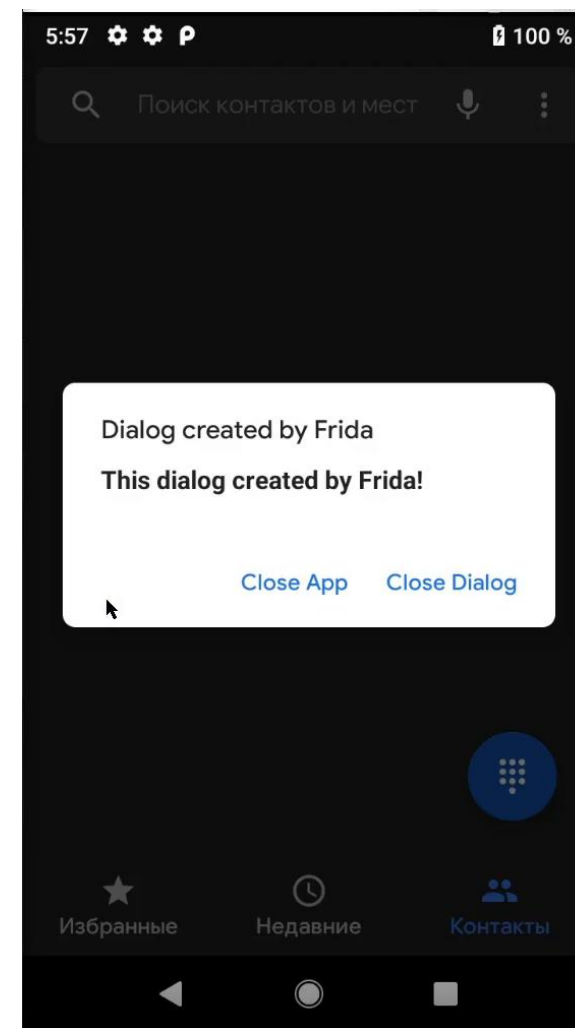
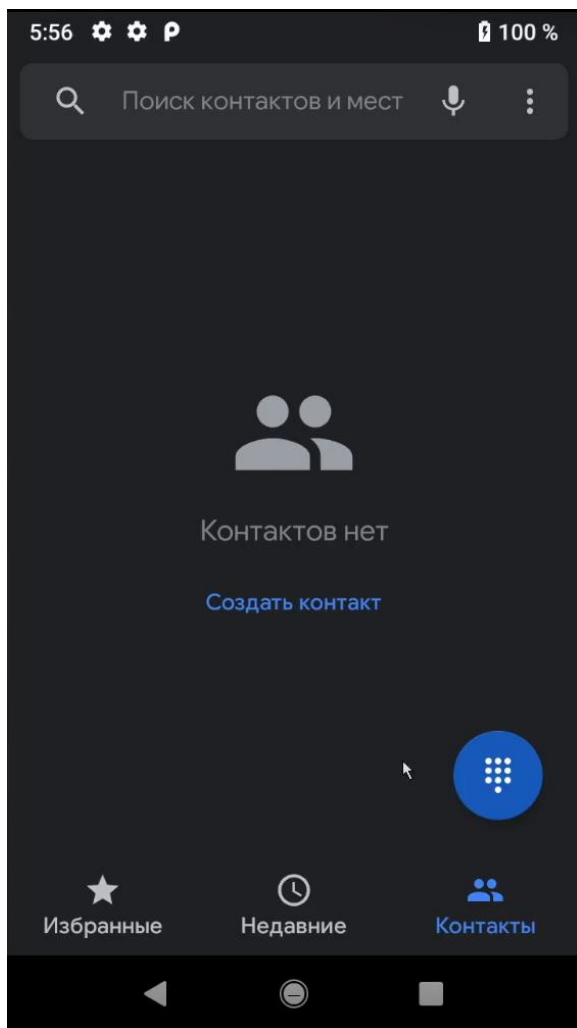
# Пример работы FRIDA



# Пример работы FRIDA



# Пример работы FRIDA





# FRIDA — способы внедрения

Поддерживается 2 способа внедрения:

## 01

Требуется ROOT:

FRIDA устанавливается в системный раздел, например, в `/data/local/tmp`

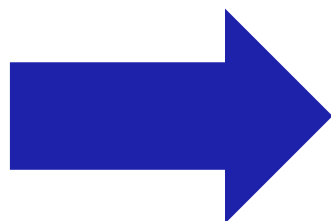
## 02

Не требуется ROOT:

злоумышленник модифицирует APK — добавляется библиотека `libfridagadget.so`, которая обеспечивает работоспособность FRIDA



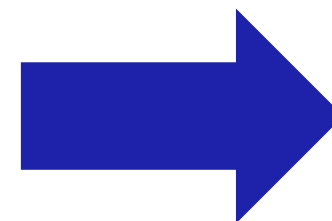
# FRIDA для SSL Unpinning и совершения MITM-атаки



**Мобильное приложение**

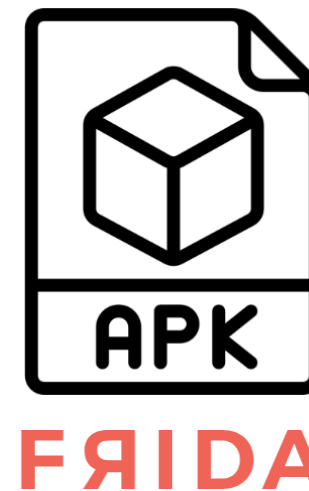
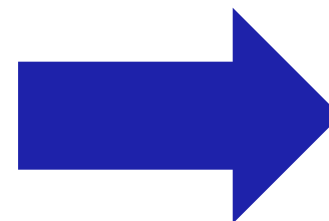
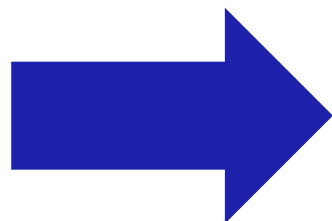
Скачать на Android

Интернет-банк





# FRIDA для SSL Unpinning и совершения MITM-атаки





# FRIDA для SSL Unpinning и совершения MITM-атаки



```
1. mitmproxy (python3.7)
Flows
GET http://api.icndb.com/jokes/random?firstName=John&lastName=Doe
<- 200 application/json 178b 280ms
>> GET http://api.icndb.com/jokes/random?firstName=John&lastName=Doe
<- 200 application/json 147b 198ms
GET http://api.icndb.com/jokes/random?firstName=John&lastName=Doe
<- 200 application/json 178b 143ms
GET http://api.icndb.com/jokes/random?firstName=John&lastName=Doe
<- 200 application/json 168b 139ms
GET http://api.icndb.com/jokes/random?firstName=John&lastName=Doe
<- 200 application/json 194b 140ms
GET http://api.icndb.com/jokes/random?firstName=John&lastName=Doe
<- 200 application/json 137b 183ms
GET http://api.icndb.com/jokes/random?firstName=John&lastName=Doe
<- 200 application/json 194b 292ms
GET http://api.icndb.com/jokes/random?firstName=John&lastName=Doe
<- 200 application/json 142b 149ms
GET http://api.icndb.com/jokes/random?firstName=John&lastName=Doe
<- 200 application/json 153b 272ms
GET http://api.icndb.com/jokes/random?firstName=John&lastName=Doe
<- 200 application/json 159b 164ms
GET http://api.icndb.com/jokes/random?firstName=John&lastName=Doe
<- 200 application/json 184b 136ms
[3/22] [*:8080]
```





# FRIDA для SSL Unpinning и совершения MITM-атаки

## 01

Скрипт FRIDA для SSL Unpinning

[codeshare.frida.re/@akabe1/frida-multiple-unpinning](https://codeshare.frida.re/@akabe1/frida-multiple-unpinning)



## 02

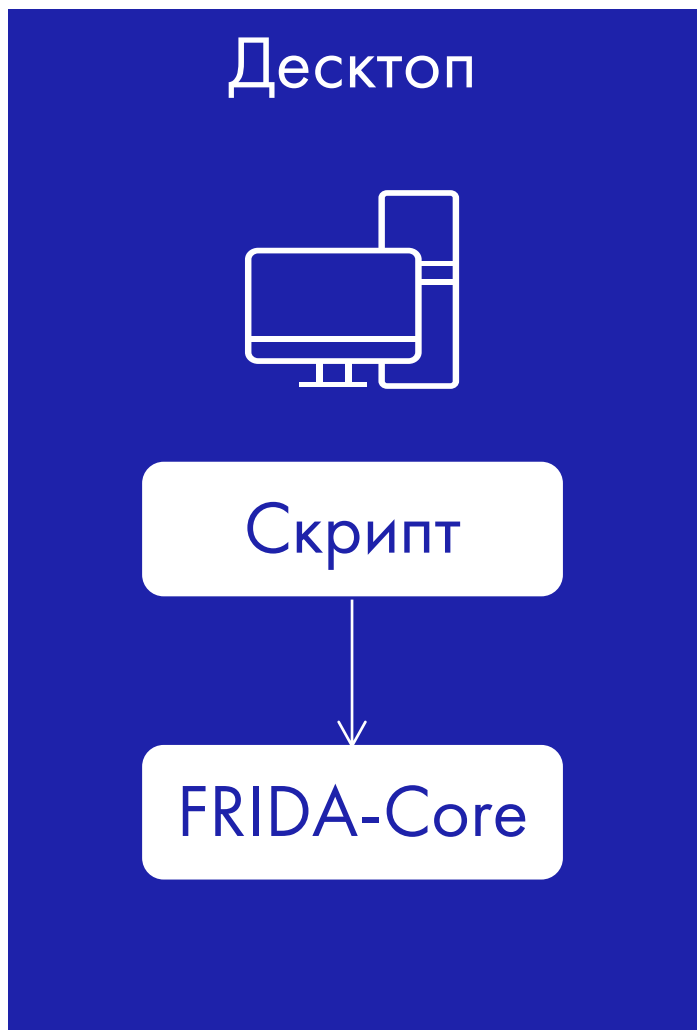
OWASP: SSL Unpinning Techniques in Android

[mas.owasp.org/MASTG/techniques/android/MASTG-TECH-0012](https://mas.owasp.org/MASTG/techniques/android/MASTG-TECH-0012)



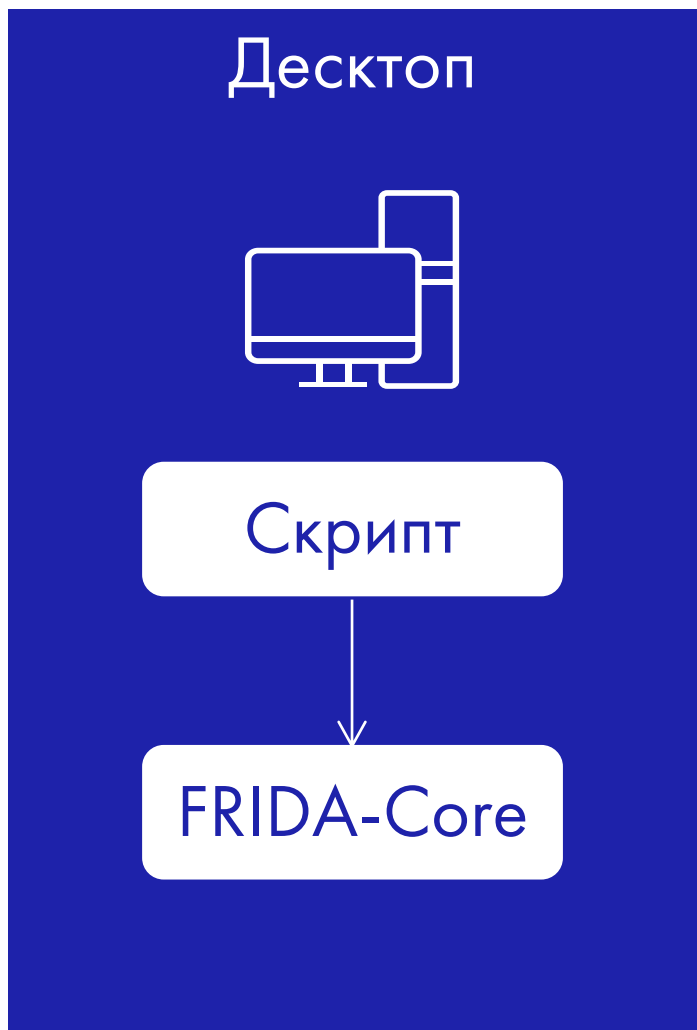


# Из чего состоит FRIDA?



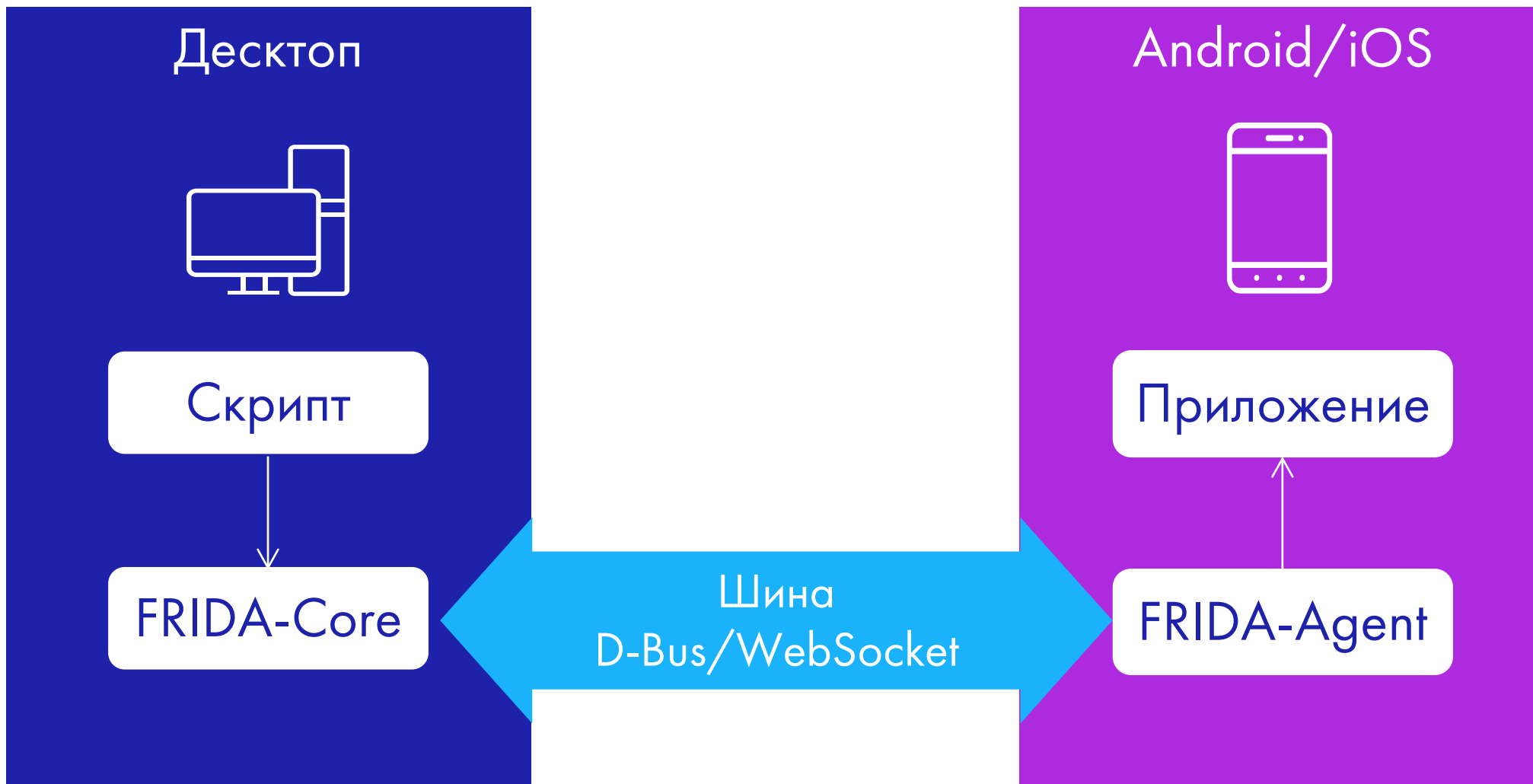


# Из чего состоит FRIDA?





# Из чего состоит FRIDA?





# Метод обнаружения FRIDA от @aimardcr

## 01

Detecting Frida the «Right» way

[medium.com/@aimardcr/detecting-frida-the-right-way-7cb3227edafb](https://medium.com/@aimardcr/detecting-frida-the-right-way-7cb3227edafb)



## 02

Исходный код проекта

[github.com/aimardcr/AndroidNativeGuard](https://github.com/aimardcr/AndroidNativeGuard)





# Метод обнаружения FRIDA от @aimardcr

```

for (int i = 1; i < 65535; i++) {
    addr.sin_port = htons(i);
    if (SecureAPI::connect(fd, (const struct sockaddr *)&addr, sizeof(addr)) == 0) {
        char req[1024]; char res[1024];
        sprintf(req,
            "GET /ws HTTP/1.1\r\nUpgrade: websocket\r\nConnection: "
            "Upgrade\r\nSec-WebSocket-Key: "
            "CpxD2C5REVLHvsUC9YAoqg==\r\nSec-WebSocket-Version: 13\r\nHost: "
            "%s:%d\r\nUser-Agent: Frida/16.1.7\r\n\r\n",
            inet_ntoa(addr.sin_addr), ntohs(addr.sin_port));
        SecureAPI::write(fd, req, SecureAPI::strlen(req));

        if (SecureAPI::read(fd, res, sizeof(res)) > 0) {
            if (SecureAPI::strstr(res, "tyZqL/Y8dNFFyopTrHadWzvbvRs=")) {
                LOGI("FridaDetect::detectFridaListener found fingerprint on port %d "
                    "(WebSocket)", ntohs(addr.sin_port));
                SecureAPI::close(fd);
                return true;
            }
        }
    }
}
    
```

Перебор всех TCP-портов

```

"GET /ws HTTP/1.1\r\nUpgrade: websocket\r\nConnection: "
"Upgrade\r\nSec-WebSocket-Key: "
"CpxD2C5REVLHvsUC9YAoqg==\r\nSec-WebSocket-Version: 13\r\nHost: "
"%s:%d\r\nUser-Agent: Frida/16.1.7\r\n\r\n",
    
```

Handshake для установления WebSocket соединения

Sec-WebSocket-Accept

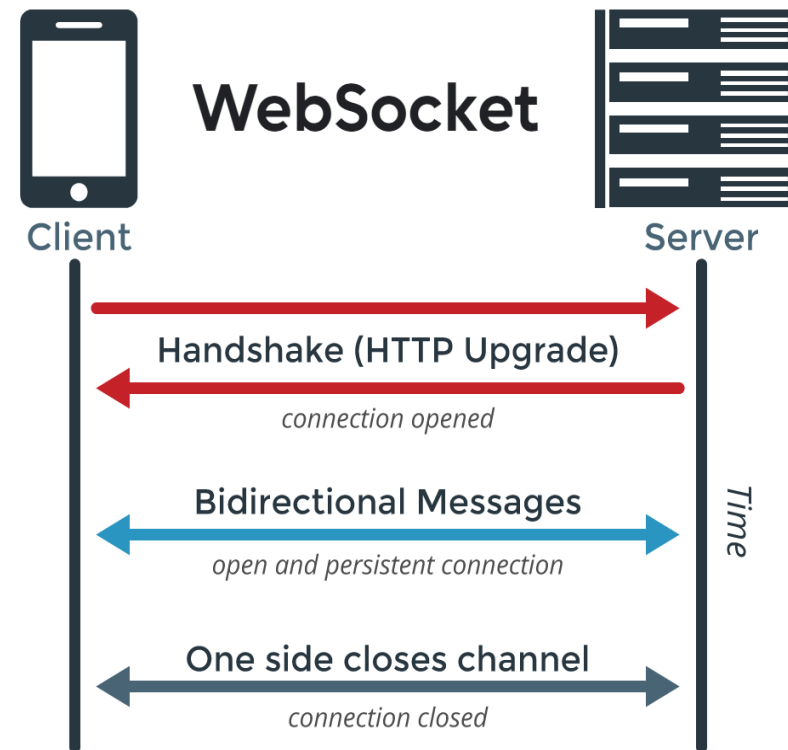


## В чем ошибается @aimardcr?

Вот так любой WebSocket-сервер генерирует Sec-WebSocket-Accept в ответе:

01

К Sec-WebSocket-Key в конец добавляем  
258EAF55-E914-47DA-95CA-C5AB0DC85B11





## В чем ошибается @aimardcr?

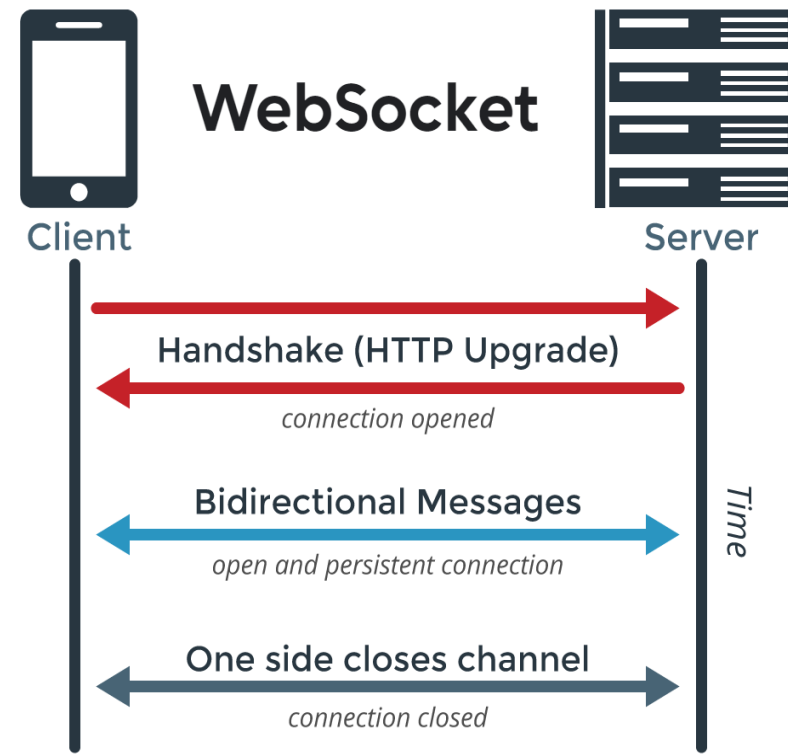
Вот так любой WebSocket-сервер генерирует Sec-WebSocket-Accept в ответе:

01

К Sec-WebSocket-Key в конец добавляем  
258EAF55-E914-47DA-95CA-C5AB0DC85B11

02

Применяем SHA-1 к строке, полученной  
на 1 шаге





## В чем ошибается @aimardcr?

Вот так любой WebSocket-сервер генерирует Sec-WebSocket-Accept в ответе:

01

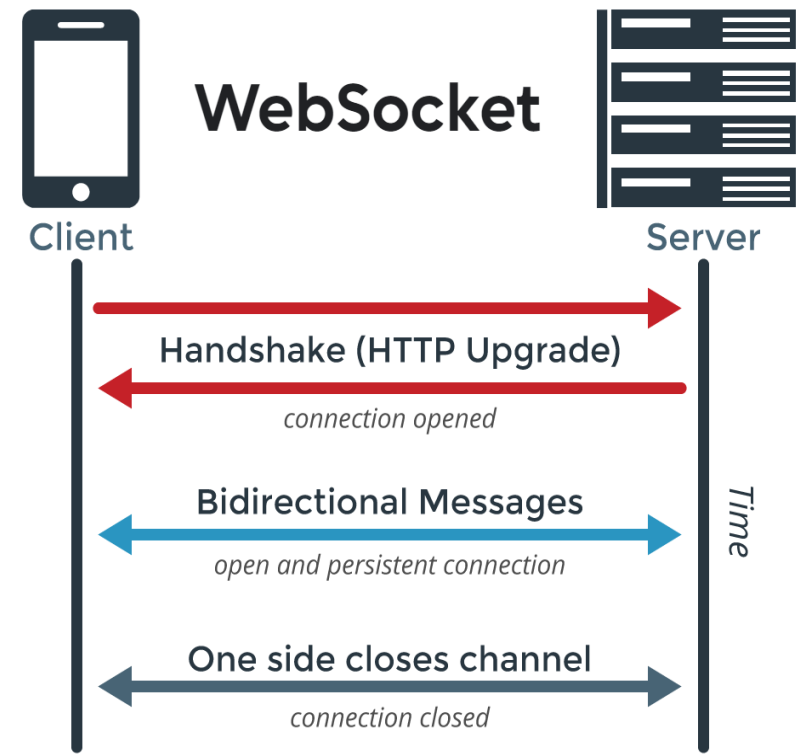
К Sec-WebSocket-Key в конец добавляем  
258EAF5A5-E914-47DA-95CA-C5AB0DC85B11

02

Применяем SHA-1 к строке, полученной  
на 1 шаге

03

Применяем Base64 к SHA-1 хэшу,  
полученному на 2 шаге






# Что стоит перенять от решения @aimardcr

## 01

Использовать **C++** и **Android NDK** для создания библиотек с функционалом, которому нужна повышенная защита от реверс-инжиниринга

## 02

Использовать **альтернативные реализации libc** для защиты от перехвата системных вызовов с помощью FRIDA

@aimardcr использует  **musl libc**



# А как правильно искать FRIDA?

01

Сначала обходим TCP порт 27042 на localhost (используется FRIDA по умолчанию), если не обнаружено — то обходим остальные порты



## А как правильно искать FRIDA?

01

Сначала обходим **TCP порт 27042** на **localhost** (используется FRIDA по умолчанию), если не обнаружено — то обходим остальные порты

02

Отправляем **Handshake-заголовок** для установления WebSocket соединения, созданный FRIDA-Core



## А как правильно искать FRIDA?

01

Сначала обходим **TCP порт 27042** на **localhost** (используется FRIDA по умолчанию), если не обнаружено — то обходим остальные порты

02

Отправляем **Handshake-заголовок** для установления **WebSocket** соединения, созданный **FRIDA-Core**

03

Получаем в ответ **заголовок о успешном установлении WebSocket** соединения и проверяем его



## А как правильно искать FRIDA?

01

Сначала обходим **TCP порт 27042** на **localhost** (используется FRIDA по умолчанию), если не обнаружено — то обходим остальные порты

02

Отправляем **Handshake-заголовок** для установления **WebSocket** соединения, созданный **FRIDA-Core**

03

Получаем в ответ **заголовок о успешном установлении WebSocket соединения** и проверяем его

04

Отправляем **D-Bus сообщение** для получения параметров устройства, на котором запущена **FRIDA-Agent**



## А как правильно искать FRIDA?

01

Сначала обходим **TCP порт 27042** на **localhost** (используется FRIDA по умолчанию), если не обнаружено — то обходим остальные порты

02

Отправляем **Handshake-заголовок** для установления **WebSocket** соединения, созданный **FRIDA-Core**

03

Получаем в ответ **заголовок о успешном установлении WebSocket соединения** и проверяем его

04

Отправляем **D-Bus сообщение** для получения параметров устройства, на котором запущена **FRIDA-Agent**

05

Проверяем ответ на **наличие ключевых слов** от **FRIDA**



# Handshake — заголовок для установления соединения

```
GET /ws HTTP/1.1
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: DSG+dXjvwBEDiiM7jVuXeQ==
Sec-WebSocket-Version: 13
Host: lolcathost
User-Agent: Frida/16.2.1
```

FRIDA задает lolcathost  
в качестве хоста

```
47 45 54 20 2F 77 73 20 48 54 54 50 2F 31 2E 31
0D 0A 55 70 67 72 61 64 65 3A 20 77 65 62 73 6F
63 6B 65 74 0D 0A 43 6F 6E 6E 65 63 74 69 6F 6E
3A 20 55 70 67 72 61 64 65 0D 0A 53 65 63 2D 57
65 62 53 6F 63 6B 65 74 2D 4B 65 79 3A 20 44 53
47 2B 64 58 6A 76 77 42 45 44 69 69 4D 37 6A 56
75 58 65 51 3D 3D 0D 0A 53 65 63 2D 57 65 62 53
6F 63 6B 65 74 2D 56 65 72 73 69 6F 6E 3A 20 31
33 0D 0A 48 6F 73 74 3A 20 6C 6F 6C 63 61 74 68
6F 73 74 0D 0A 55 73 65 72 2D 41 67 65 6E 74 3A
20 46 72 69 64 61 2F 31 36 2E 32 2E 31 0D 0A 0D
0A
```

```
GET /ws HTTP/1.1
??Upgrade: webso
cket??Connection
: Upgrade??Sec-W
ebSocket-Key: DS
G+dXjvwBEDiiM7jV
uXeQ==??Sec-WebS
ocket-Version: 1
3??Host: lolcath
ost??User-Agent:
Frida/16.2.1??
?
```



# Ответ о успешном WebSocket-соединении

```
HTTP/1.1 101 Switching Protocols  
Date: Tue, 14 May 2024 12:46:15 GMT  
Upgrade: websocket  
Connection: Upgrade  
Sec-WebSocket-Accept: la1xpP1iLDC92EGUqLTq++oxhJs=
```

```
48 54 54 50 2F 31 2E 31 20 31 30 31 20 53 77 69  
74 63 68 69 6E 67 20 50 72 6F 74 6F 63 6F 6C 73  
0D 0A 44 61 74 65 3A 20 54 75 65 2C 20 31 34 20  
4D 61 79 20 32 30 32 34 20 31 32 3A 34 36 3A 31  
35 20 47 4D 54 0D 0A 55 70 67 72 61 64 65 3A 20  
77 65 62 73 6F 63 6B 65 74 0D 0A 43 6F 6E 6E 65  
63 74 69 6F 6E 3A 20 55 70 67 72 61 64 65 0D 0A  
53 65 63 2D 57 65 62 53 6F 63 6B 65 74 2D 41 63  
63 65 70 74 3A 20 6C 61 31 78 70 50 31 69 4C 44  
43 39 32 45 47 55 71 6C 54 71 2B 2B 6F 78 68 4A  
73 3D 0D 0A 0D 0A
```

```
HTTP/1.1 101 Swi  
tching Protocols  
??Date: Tue, 14  
May 2024 12:46:1  
5 GMT??Upgrade:  
websocket??Conne  
ction: Upgrade??  
Sec-WebSocket-Ac  
cept: la1xpP1iLD  
C92EGUqLTq++oxhJ  
s=????
```



# D-Bus сообщение к FRIDA-Agent

```
1A 51 8F 3B 1A 51 DB 3A 75 51 CF 3B 1A 51 F5 49
7F 7E BC 49 73 35 BB 14 52 3E A9 4F 49 34 A9 48
73 3E B4 3B 1A 51 D8 3A 69 51 CC 3B 1A 51 A8 5E
34 37 A8 52 7E 30 F4 73 75 22 AE 68 7F 22 A9 52
75 3F EB 0D 1A 51 D2 3A 7D 51 DB 4E 1A 51 D9 3A
69 51 DE 3B 1A 51 8A 52 74 36 DA 3B 1A 51 C4 3B
1A 51 82 F8 EC 48 BC 3E 80 49 BC 3F EC 48 BC 3E
EE 48 BC 3E 8A 48 BC 3E ED 49 D3 3E F9 48 BC 3E
C3 3A D9 11 8A 3A D5 5A 8D 67 F4 51 9F 3C EF 5B
9F 3B D5 51 82 48 BC 3E EE 49 CF 3E FA 48 BC 3E
9E 2D 92 58 9E 21 D8 5F C2 00 D3 4D 98 1B D9 4D
9F 21 D3 50 DD 7E BC 3E E4 49 DB 3E EC 48 BC 3E
EF 49 CF 3E F9 48 BC 3E BD 3D D9 4C 95 1B C5 4D
98 2D D1 6E 8D 3A DD 53 89 3C D9 4C 9F 48 BC 3E
```

```
?Q?;?Q?:uQ?;?Q?I
~?Is5??R>?OI4?H
s>;?Q?:iQ?;?Q?^
47?R~0?su"?h "?R
u????Q?:}Q?N?Q?:
iQ?;?Q?Rt6?;?Q?;
?Q???H?>?I???H?>
?H?>?H?>?I?>?H?>
?:???:?Z?g?Q?<?[
?;?Q?H?>?I?>?H?>
?-?X?!?_???M???M
?!?P?~?>?I?>?H?>
?I?>?H?>=?L???M
?-?n?:?S?<?L?H?>
```



# Ответ с данными об устройстве от FRIDA-Agent

## Данные в JSON

```

{
  "arch": "arm",
  "os": {
    "version": "9",
    "id": "android",
    "name": "Android"
  },
  "platform": "linux",
  "api-level": 28,
  "access": "full"
}

```



## D-Bus ответ

```

?????g?a{sv}??
?????u??????
?????arch?s?
???arm?????
os?a{sv}??H??
?????version?
?s????9?????
id?s????andr
oid????name?s?
???Android????
platform?s????
linux?????
api-level?x???
?????????
access?s?????
full?

```



## Что мы узнали?

# 01

**FRIDA** —  
мощный инструмент  
для внедрения  
произвольного  
кода в другие  
приложения

# 02

**FRIDA** может  
использоваться  
на устройствах  
без ROOT

# 03

**FRIDA** можно  
обнаружить —  
используем знания  
о принципах работы  
данного инструмента



01

У ROOT в Android нет единственного воплощения

02

Android стал защищеннее благодаря аппаратным средствам защиты

03

Множество вещей под капотом в Android используют концепций и инструментарий из Linux

# Все ссылки из презентации



[drive.yadro.com/s/Z4Wf9sqt8QaA4y7](https://drive.yadro.com/s/Z4Wf9sqt8QaA4y7)



БУДУЩЕЕ  
В НАШИХ  
РУКАХ