Инженерный вестник Дона, №10 (2025)
ivdon.ru/ru/magazine/archive/n10y2025/10449

Метод противодействия несанкционированному повышению привилегий в ОС Android на основе технологии аппаратной виртуализации

Г.Ю. Пагуба [1], Д.Н. Пермяков [1], Н.Н. Самарин [2], Н.В. Соболев[1]

[1]Петербургский политехнический университет Петра Великого
[2]Московский технический университет связи и информатики

Аннотация: В работе предложен метод противодействия несанкционированному повышению привилегий в операционной системе Android. Предложенный метод подразумевает использование технологии аппаратной виртуализации архитектуры ARM для контроля доступа к служебным структурам данных ядра операционной системы, хранящим идентификационную информацию задач.
Ключевые слова: информационная безопасность, повышение привилегий, Android, гипервизор, защита информации, аппаратная виртуализация, контроль доступа, целостность структур языка, обеспечение информационной безопасности

Введение

По данным источника [1], операционная система Android является наиболее распространённой операционной системой для мобильных устройств, занимая 76% рынка на период первого квартала 2025 года.

Под управлением данной операционной системы работают мобильные устройства различных видов: от смартфонов до банковских терминалов.

По данным Лаборатории Касперского [2], в первом квартале 2025 года было зафиксировано 12 миллионов атак на мобильные устройства с применением вредоносного программного обеспечения. В таблице 1 представлено соотношение имени образца вредоносного программного обеспечения и процента его использования в первом квартале 2025 года для десяти наиболее распространённых образцов. Как можно заметить на данной таблице, почти все имена образцов вредоносного программного обеспечения содержат строку «Android OS», что по стандарту именования Лаборатории Касперского говорит о предназначении данных образцов для функционирования в операционной системе Android.

Таблица 1

Соотношение имен образцов вредоносного программного обеспечения и процентов их использования в первом квартале 2025 года

Имя образца вредоносного программного обеспечения	**Процент использования в атаках в первом квартале 2025 года**
Trojan.AndroidOS.Fakemoney.v	26,41
DangerousObject.Multi.Generic.	19,30
Trojan-Banker.AndroidOS.Mamont.db	15,99
Trojan-Banker.AndroidOS.Mamont.da	11,21
Trojan-Banker.AndroidOS.Mamont.bc	7,61
Backdoor.AndroidOS.Triada.z	4,71
Trojan.AndroidOS.Triada.hf	3,81
Trojan.AndroidOS.Triada.fe	3,48
Trojan.AndroidOS.Triada.gn	2,68

Одним из ключевых механизмов работы современного вредоносного программного обеспечения для ОС Android является механизм несанкционированного повышения привилегий процесса вредоносного программного обеспечения до привилегий системы или суперпользователя (root) посредством эксплуатации уязвимостей в различных компонентах системы с целью обхода механизма песочницы операционной системы и получения доступа к чувствительной пользовательской информации.

Таким образом, актуальной задачей является противодействие несанкционированному повышению привилегий в операционной системе Android. Для решения этой задачи в рамках данной работы предлагается использование технологии аппаратной виртуализации архитектуры ARM [3].

1. Описание предлагаемого метода

Задача противодействия несанкционированному повышению привилегий в операционной системе Android может быть решена за счет контроля доступа к структурам ядра, содержащих информацию,

определяющую привилегии пользовательских задач операционной системы (в том числе, процессов и потоков).

Контроль доступа к данным структурам ядра предлагается осуществлять из специального программного средства – гипервизора 1 типа, называемого далее средством противодействия.

1.1. Выбор служебных структур ядра для контроля доступа

По данным источника [4], идентификационная информация задач операционной системы Android хранится в служебных структурах ядра, представленных на таблице 2.

Таблица 2

Служебные структуры ядра ОС Android, хранящие идентификационную информацию задач

Имя структуры	**Файл исходных кодов ядра ОС Android**	**Описание структуры**
task_struct	/include/linux/sched.h	Содержит описание текущего состояния задачи и указатели на структуру cred
task_security_struct	/security/selinux/include/objsec.h	Содержит информацию, описывающую привилегии задачи в контексте механизма SeLinux (мандатной модели контроля доступа)
cred	/include/linux/cred.h	Содержит идентификаторы задачи, такие как UID, GID и другую вспомогательную идентификационную информацию

1.2. Осуществление контроля доступа к выбранным служебным структурам

Реализация контроля доступа к служебным структурам ядра операционной системы Android заключается в модификации таблиц трансляции механизма двухуровневой трансляции адресного пространства оперативной памяти.

На рис. 1 представлена блок-схема контроля доступа к выбранным служебным структурам ядра.

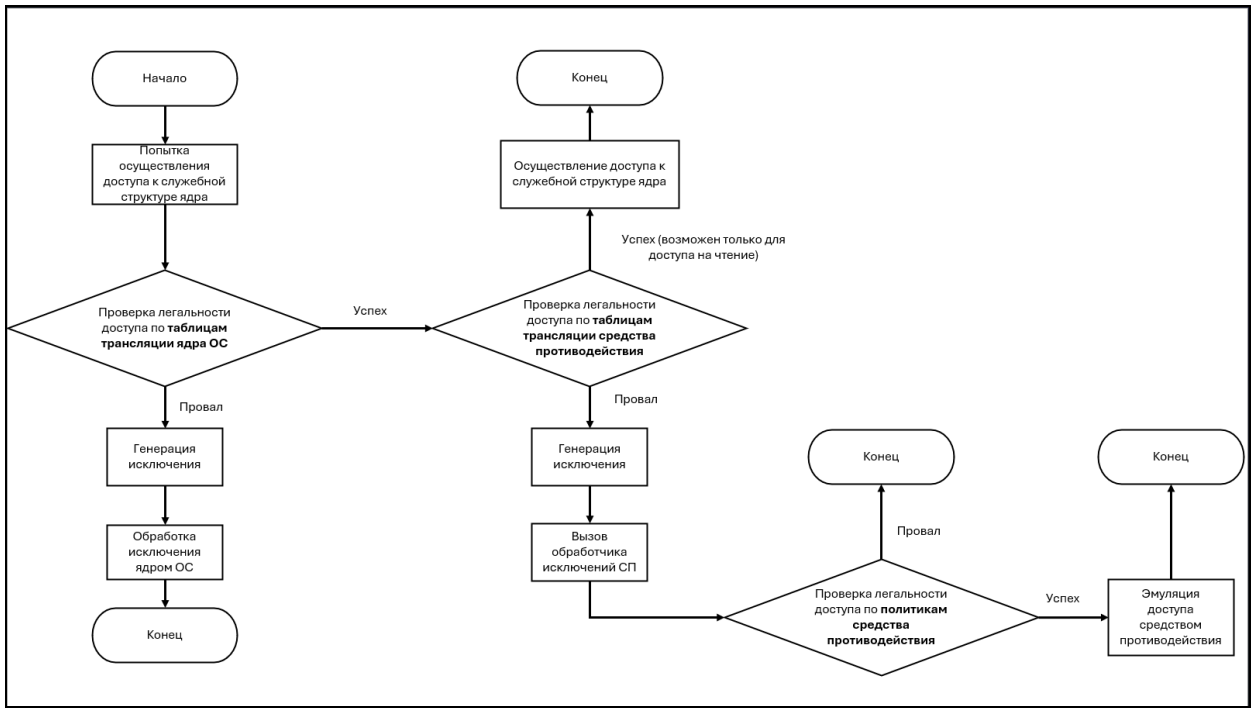

Рис. 1. – Блок-схема контроля доступа к служебным структурам ядра операционной системы Android

Проверка легальности доступа с использованием политик средства противодействия осуществляется следующим образом:

1. Если происходит попытка доступа на исполнение, доступ блокируется.

2. Если происходит попытка модификации структуры task_struct и данная попытка не является перезаписью указателей task_struct.real_cred или task_struct.cred или поля addr_limit – максимального адреса, доступного в

пространстве пользователя, то попытка разрешается.

3. Если происходит попытка модификации структуры task_struct и данная попытка является перезаписью указателя task_struct.real_cred или task_struct.cred, то попытка разрешается в случае, если модифицируемая структура task_struct совпадает со структурой task_struct текущей исполняемой задачи.

4. Если происходит попытка модификации структуры cred, происходит анализ её текущего состояния на предмет удовлетворения модели безопасности операционной системы и на его основе выносится вердикт о разрешении доступа.

5. Если происходит попытка модификации структуры task_security_struct, то вердикт о запрете попытки выносится на основе анализа состояния связанной с ней структуры cred.

1.3. Динамическая инструментация функций создания служебных структур ядра

Для обеспечения контроля доступа к служебным структурам ядра операционной системы Android, необходим их перенос в области памяти, недоступные для модификации из ядра и доступные для чтения и модификации из средства противодействия.

Перенос данных структур предлагается осуществлять за счет динамической инструментации функций ядра, осуществляющих создание данных структур посредством поиска и модификации в ядре операционной системы структуры kallsyms, содержащей адреса всех экспортируемых функций ядра.

Kallsyms [5] – структура данных ядра операционной системы Android, входящая в состав образа ядра и содержащая адреса (или базовый адрес и сдвиги), статус экспортирования и имена всех его функций.

Формат структуры kallsyms для ядра версии 6.6 представлен на таблице 3.

Таблица 3

Формат структуры kallsyms для ядра версии 6.6 операционной системы Android

Имя		**Тип**	**Описание**
1		**2**	**3**
kallsyms_num_syms		unsigned long int	Количество символов ядра, доступных в таблице kallsyms.
kallsyms_names		unsigned char []	Массив, содержащий преобразованные имена символов ядра, адреса которых доступны для получения из структуры kallsyms.
kallsyms_markers		unsigned long int []	Массив, содержащий информацию для быстрого поиска адреса символа по его имени.
kallsyms_token_table		unsigned char []	Массив символьных или строковых литералов, содержащихся в именах символов ядра, адреса которых доступны для получения из структуры kallsyms.
kallsyms_token_index		unsigned short []	Массив, содержащий соотношение между литералами в массивах kallsyms_markers и kallsyms_token_table.
kallsyms_addresses	kallsyms_relative_base	unsigned long int [] (kallsyms_addresses) или unsigned long int (kallsyms_relative_base) и int [] (kallsyms_offsets)	Массив, содержащий адреса (kallsyms_addresses) или сдвиги (kallsyms_offsets) от базового адреса (kallsyms_relative_base) символов ядра, адреса которых доступны для получения из структуры kallsyms.
	kallsyms_offsets		

Имя	Тип	Описание
1	2	3
kallsyms_seqs_of_names	unsigned int []	Массив, содержащий преобразованные целые числа. Индекс каждого элемента данного массива соотносится с алфавитным порядком имен символов, а значение с элементами массивов kallsyms_addresses и kallsyms_names.

1.4. Пример работы предлагаемого метода

В качестве примера работы предлагаемого метода рассмотрим реализацию функции setuid (__sys_setuid), являющуюся частью программного кода ядра ОС Android. Исходный код данной функции представлен на листинге 1.

Как можно заметить, данная функция вызывает функцию prepare_creds, отвечающую за создание новой структуры cred с измененным UID.

Функция prepare_creds подлежит инструментации средством противодействия до вызова.

После вызова инструментированной версии данной функции будет произведён вызов средства противодействия (с помощью ассемблерной инструкции HVC). Средство противодействия выделит память для структуры cred и вернёт указатель на выделенную область памяти вызывающему коду.

В случае успешного выполнения логики функции __sys_setuid, в завершение вызывается функция commit_creds, отвечающая за перезапись указателей task_struct.real_cred и task_struct.cred текущей исполняемой задачи. Программный код, осуществляющий перезапись создаст исключения отказа страницы (Page Fault) и уровень исключений будет повышен до EL2. После прохождения стандартных проверок по таблицам трансляции будет реализована проверка по политикам средства противодействия.

Листинг 1

Исходный код функции setuid.

```
long __sys_setuid(uid_t uid)
{
    struct user_namespace *ns = current_user_ns();
    const struct cred *old;
    struct cred *new;
    int retval;
    kuid_t kuid;
    kuid = make_kuid(ns, uid);
    if (!uid_valid(kuid))
    {
        return -EINVAL;
    }
    new = prepare_creds();
    if (!new)
    {
        return -ENOMEM;
    }
    old = current_cred();
    retval = -EPERM;
    if (ns_capable_setid(old->user_ns, CAP_SETUID))
    {
        new->suid = new->uid = kuid;
        if (!uid_eq(kuid, old->uid))
        {
            retval = set_user(new);
            if (retval < 0)
            {
                goto error;
            }
        }
    }
    else if (!uid_eq(kuid, old->uid) && !uid_eq(kuid, new->suid))
    {
        goto error;
    }
    new->fsuid = new->euid = kuid;
    retval = security_task_fix_setuid(new, old, LSM_SETID_ID);
    if (retval < 0)
    {
        goto error;
    }
    return commit_creds(new);
error:
    abort_creds(new);
    return retval;
}
```

В случае, если процесс, вызывающий системный вызов setuid, имеет возможность (capability) CAP_SETUID [7], то есть данный запрос удовлетворяет стандартной модели безопасности ОС Android, то средство

противодействия выполнит его вместо ядра операционной системы, иначе данный запрос будет заблокирован.

2. Программная реализация предлагаемого метода

2.1. Требования к программной реализации предлагаемого метода

По результатам анализа существующих средств противодействия несанкционированному повышению привилегий, представленных в предыдущей работе авторов, к программной реализации предлагаемого метода были выдвинуты следующие требования:

1. Эффективность. Программная реализация должна противодействовать несанкционированному повышению привилегий.

2. Надёжность. Программная реализация не должна противодействовать санкционированному повышению привилегий.

3. Оптимальное расходование вычислительных ресурсов. Программная реализация не должна создавать заметных пользователю замедлений ядра операционной системы, а также должна рационально тратить процессорное время и оперативную память.

4. Модульность. Программная реализация должна иметь модульную архитектуру, где каждый модуль должен решать отдельную задачу.

5. Простота портирования. Исходные коды программной реализации должны быть легко портированы между различными микропроцессорами и системами на чипе архитектуры ARM.

6. Простота внедрения в состав программного обеспечения операционной системы Android. Программное средство должно легко внедряться в состав программного обеспечения операционной системы Android для различных устройств, в том числе иметь возможность загрузки в составе цепочки доверенной загрузки стандарта ATF [8] при условии установленного пользователем корня доверия.

2.2. Описание программного средства, реализующего предлагаемый метод

Архитектура программного средства, реализующего программный метод, включает следующие модули:

1. Модуль инициализации. Реализует функционал по инициализации других модулей и записи конфигурационных значений в системные регистры архитектуры ARM, такие как HCR_EL2.

2. Модуль динамической инструментации программного кода ядра ОС. Реализует функционал поиска структуры kallsyms в бинарном файле ядра ОС и инструментации функций, создающих служебные структуры ядра.

3. Модуль информирования средства противодействия. Является полезной нагрузкой, встраиваемой в код ядра операционной системы, и содержит механизмы передачи информации о служебных структурах ядра средству противодействия.

4. Модуль контроля доступа к служебным структурам ядра ОС. Реализует функционал обработки исключений и анализа попыток доступа к служебным структурам ядра на основе встроенных политик.

5. Модуль исполнения запросов. Реализует исполнение запросов ядра, которые прошли проверку валидности.

6. Модуль журналирования. Реализует функционал вывода диагностических и отладочных сообщений по протоколу UART.

На рис. 2 представлена структурная схема взаимодействия модулей программного средства, реализующего предлагаемый метод.

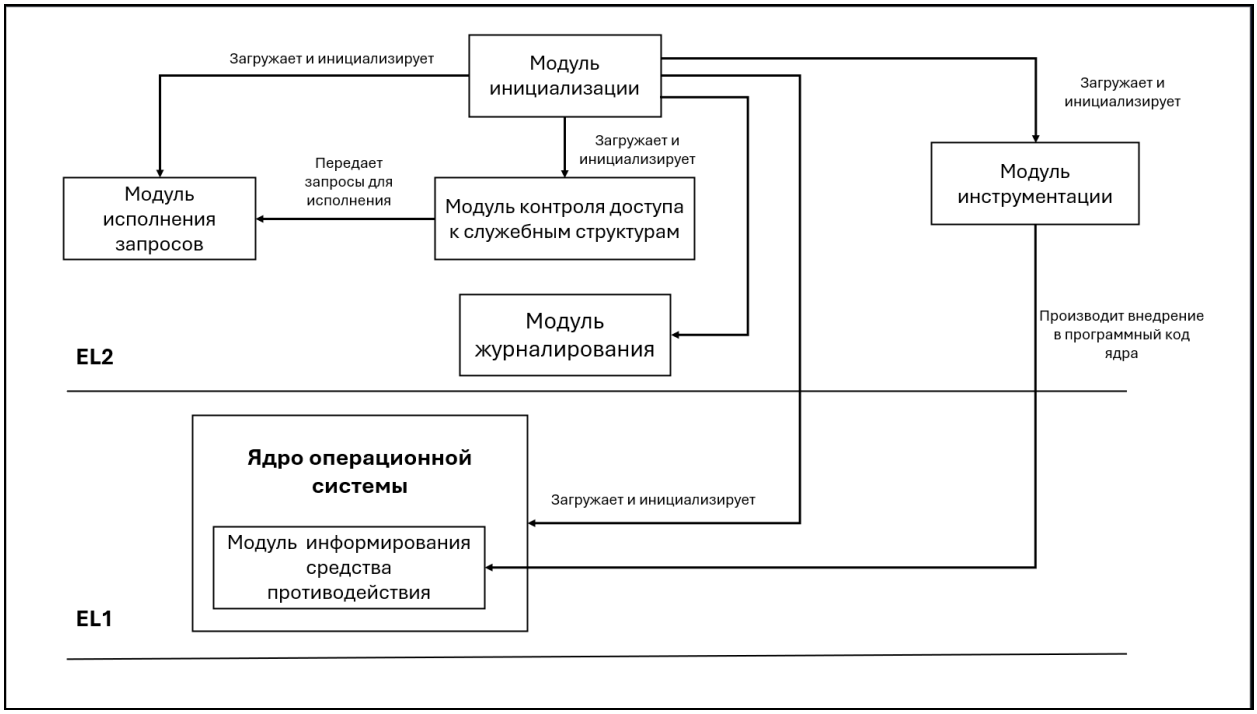

Рис. 2. – Структурная схема взаимодействия модулей программного средства, реализующего предлагаемый метод

Программное средство, реализующее предлагаемый метод, имеет следующую последовательность инициализации:

1. Получение управления от загрузчика ядра операционной системы.

2. Установка обработчика исключения отказа страницы (Page Fault) и других вспомогательных обработчиков.

3. Поиск базового адреса ядра операционной системы.

4. Поиск в программном коде ядра функций, реализующих функционал создания выделенных служебных структур ядра.

5. Динамическая инструментация найденных методов.

6. Настройка таблиц трансляции второго уровня.

7. Передача управления ядру операционной системы.

2.2.1. Модуль инициализации

Модуль инициализации является первым модулем, исполнение которого начинается после получения управления средством противодействия от загрузчика операционной системы. Основной задачей

данного модуля является подготовка устройства для работы остальных модулей и ядра операционной системы. Данная задача может быть разбита на следующие подзадачи:

1. Установка таблиц трансляции.
2. Настройка перехвата доступа к регистрам.
3. Инициализация других модулей средства противодействия.
4. Распаковка и загрузка ядра и рамдиска.
5. Понижение уровня исключений и передача управления ядру.

Для успешной работы предлагаемого метода необходима корректная настройка таблиц трансляции гипервизора. Таблицы трансляции гипервизора настраиваются таким образом, чтобы создать отдельный регион оперативной памяти, доступный только для чтения, в котором будут храниться экземпляры служебных структур ядра. Адреса настроенных таблиц трансляции записываются в регистр VTTBR_EL2.

Для предотвращения модификации программного кода средства противодействия или таблиц трансляции, модуль инициализации производит настройку платформенных регистров, таких как HCR_EL2 и устанавливает перехват доступа к регистру SCTLR_EL1 с целью блокировки попыток выключения MMU.

Распаковка и загрузка ядра операционной системы и рамдиска происходят по следующим этапам:

1. Релокация программного кода средства противодействия на другой базовый адрес.
2. Распаковка сжатого образа, содержащего ядро операционной системы и рамдиск на адрес, на котором находился программный код средства противодействия.
3. Проверка целостности ядра и рамдиска.
4. Передача управления ядру операционной системы.

Для передачи управления ядру операционной системы, в регистр ELR_EL2 записывается адрес RESET вектора ядра, а в регистр SPSR_EL2 значение, указывающее на то, что уровень исключений необходимо понизить до EL1.

2.2.2. Модуль динамической инструментации

Алгоритм работы модуля динамической инструментации имеет следующие основные шаги:

1. Получение от модуля инициализации базового адреса ядра.
2. Поиск в образе ядра структуры kallsyms.
3. Поиск в структуре kallsyms методов для инструментации.
4. Инструментация найденных методов.
5. Возвращение модулю инициализации статуса инструментации.

Поиск структуры kallsyms осуществляется посредством поиска заранее заданных шаблонов в образе ядра.

Инструментация методов ядра осуществляется за счет использования библиотек keystone [9] и capstone [10], позволяющих дизассемблировать программный код ядра и ассемблировать внедряемую в программный код ядра полезную нагрузку во время исполнения средства противодействия.

2.2.3. Модуль информирования средства противодействия

Модуль информирования средства противодействия является фрагментом программного кода, встраиваемого в функции ядра операционной системы.

Данный модуль содержит функционал по отправке запросов средству противодействия по созданию нового экземпляра служебной структуры ядра посредством исполнения ассемблерной инструкции HVC (Hypercall) и передачи в качестве аргументов структуры, описывающей запрос. Содержимое структуры, описывающей запрос, представлено на листинге 2.

Листинг 2

Содержимое структуры, описывающей запрос модуля информирования средства противодействия

```
typedef struct _CREATE_REQUEST
{
// Хранит тип создаваемой структуры для средства противодействия
char struct_type;
// Хранит размер создаваемой структуры
unsigned int struct_sz;
// Хранит указатель на структуру со значениями, которыми необходимо
инициализировать создаваемую структуру
void * default_values;
} CREATE_REQUEST, *PCREATE_REQUEST;
```

В задачи модуля информирования средства противодействия также входит подмена указателя на созданную структуру в контексте ядра операционной системы.

2.2.4. Модуль контроля доступа к служебным структурам ядра и модуль исполнения запросов ядра

Модуль контроля доступа реализует проверку доступа по принципу, описанному в подраздел 2.2 данной работы. Политики проверки валидности хранятся в виде бинарного файла и загружаются модулем контроля доступа на этапе инициализации.

Модуль исполнения запросов ядра исполняет запросы на модификацию служебных структур ядра. Алгоритм его работы состоит из следующей последовательности шагов:

1. Получение запроса на изменение служебной структуры от ядра (Начало исполнения обработчика исключений).

2. Получение таблиц трансляции средства противодействия.

3. Поиск в таблицах трансляции записи, соответствующей странице, в

которой расположена изменяемая структура.

4. Изменение настроек доступа к странице в таблицах трансляции.

5. Инвалидация буфера ассоциативной трансляции.

6. Исполнение операции, запрошенной ядром операционной системы.

7. Восстановление настроек доступа к странице в таблицах трансляции.

8. Инвалидация буфера ассоциативной трансляции.

9. Восстановление процессорного контекста в состояние до генерации исключения.

10. Передача управления коду ядра (Исполнение инструкции eret).

3. Экспериментальные исследования разработанного программного средства

3.1. Состав используемого программно-аппаратного стенда

Для проведения экспериментальных исследований программного средства, реализующего предлагаемый метод, был разработан программно-аппаратный стенд.

Разработанный программно-аппаратный стенд имеет следующий состав:

1. Устройство подготовки тестовых данных и обработки диагностической информации в виде ПК с операционной системой Ubuntu 22.04.

2. Преобразователь протоколов USB <-> UART на базе интегральной микросхемы CH340.

3. Устройство тестирования средства противодействия в виде одноплатного микрокомпьютера Raspberry Pi 5 с операционной системой Android 15 (версия ядра 6.6.47 с патчами безопасности от 05.09.24).

Структурная схема используемого программно-аппаратного стенда представлена на рис. 3.

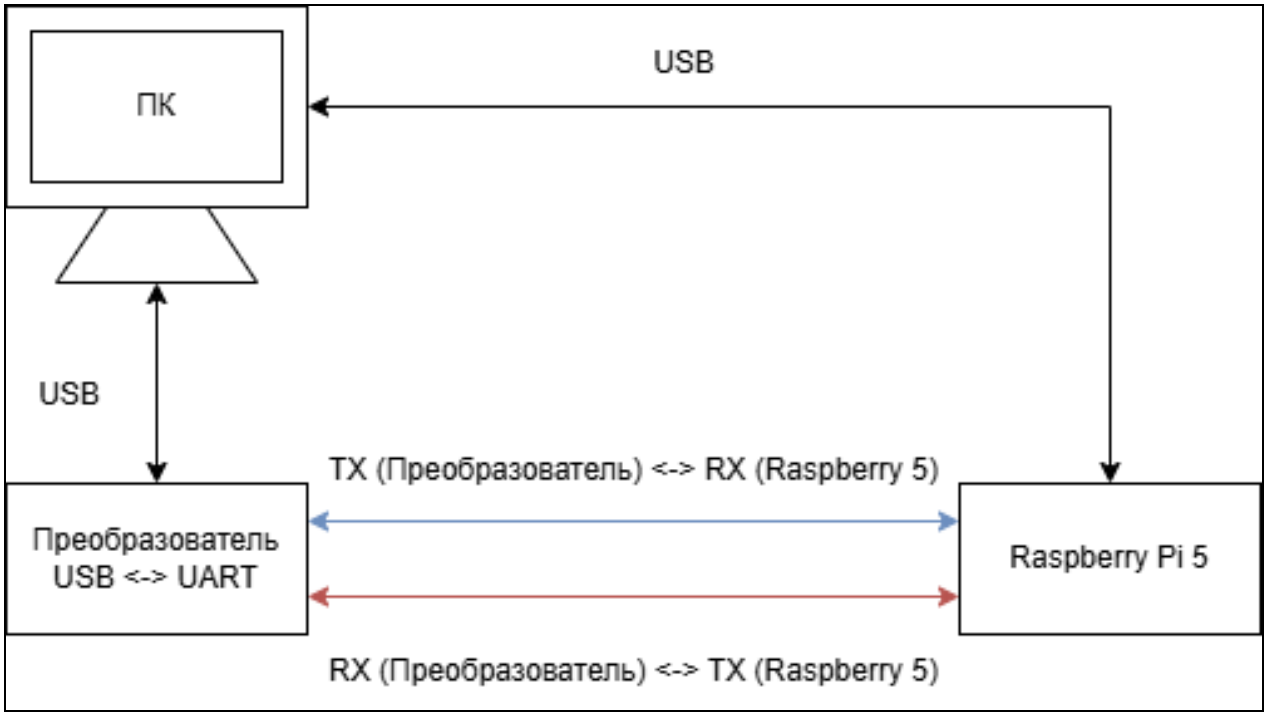

Рис. 3. – Структурная схема используемого программно-аппаратного стенда

3.2. Выбор уязвимостей для проведения экспериментальных исследований

Для проведения экспериментальных исследований программного средства, реализующего предложенный метод, были выбраны уязвимости в ядре операционной системы Android, непосредственная эксплуатация которых приводит к повышению привилегий злоумышленника до уровня ядра или системного пользователя. Критериями отбора уязвимостей являлись наличие готового эксплоита и описательной информации об уязвимости в сети Интернет.

На таблице 4 представлены CVE-идентификаторы и краткие описания выбранных уязвимостей, а также ссылки на коммиты, содержащие их исправления. Стоит отметить, что на этапе отбора уязвимостей приоритет отдавался уязвимостям, находящимся не в драйверах вендоров устройств, а в исходном коде ядра, поддерживаемом Linux Foundation или Google. Таким

образом, были выделены уязвимости в механизме межпроцессного взаимодействия Binder [11].

Таблица 4

Идентификаторы, краткие описания и коммиты с исправлениями выделенных уязвимостей

Идентификатор уязвимости в системе CVE	**Краткое описание уязвимости**	**Ссылка на коммит, устраняющий уязвимость**
CVE-2019-2215 [12]	Уязвимость заключается в ошибочном сохранении ссылки на поле wait структуры binder_thread после её освобождения	https://android.googlesource.com/kernel/msm/+/f227b9244fe32e2ce1d16fde4a9b4a41d76341cd
CVE-2020-0041 [13]	Уязвимость заключается в записи за границы массива из-за повреждения транзакции при обработке вредоносного объекта типов BINDER_TYPE_PTR и BINDER_TYPE_FDA	https://android.googlesource.com/device/google/crosshatch-kernel/+/6ea51c778c8950ae18e7eb28940f231147539034
CVE-2022-20421 [14]	Уязвимость заключается в наличии состояния гонки при взаимодействии со структурой binder_proc, в результате которого возможен доступ к данной структуре после освобождения	https://android.googlesource.com/kernel/common/+/ed25d753966a2cb3c6b488f63d6c7aa521949fd2
CVE-2023-20938 [15]	Уязвимость заключается в некорректной валидации пользовательских данных, в том числе обработке объектов, содержащих поле offsets_size без выравнивания	https://android.googlesource.com/kernel/common/+/1f56867149da18c624d298c84f3dd3ddcf41734d

3.3. Подготовка ядра операционной системы Android для проведения экспериментальных исследований

Для проведения экспериментальных исследований в ядро операционной системы Android были внесены изменения.

Для каждой из тестируемых уязвимостей был составлен обратный патч – файл с изменениями исходного кода, вносящий требуемую уязвимость в дерево исходных кодов ядра операционной системы. Далее обратные патчи были применены, а ядро операционной системы подверглось пересборке.

Таким образом было получено ядро операционной системы Android версии 6.6.47 с уязвимостями CVE-2019-2215, CVE-2020-0041, CVE-2022-20421 и CVE-2023-20938 в драйвере системы межпроцессного взаимодействия Binder.

Пересобранное ядро было установлено на целевое устройство и использовано для дальнейших экспериментальных исследований программного средства, реализующего предлагаемый метод.

3.4. Результаты экспериментальных исследований

Экспериментальные исследования программного средства, реализующего предложенный метод, проводились следующим образом:

1. Эксплуатация всех уязвимостей, выделенных ранее.
2. Фиксация результатов тестирования.
3. Установка на целевое устройство средства противодействия.
4. Эксплуатация всех уязвимостей, выделенных ранее.
5. Фиксация результатов тестирования.

Результаты экспериментальных исследований представлены в табл. 5.

Символ «+» означает факт успешной эксплуатации уязвимости, а символ «–» – отсутствие данного факта.

Таблица 5

Результаты экспериментальных исследований программного средства.

№	Идентификатор уязвимости в системе CVE	Краткое описание уязвимости	Успешность эксплуатации без установленного средства противодействия	Успешность эксплуатации с установленным средством противодействия
1	CVE-2019-2215	Уязвимость заключается в ошибочном сохранении ссылки на поле wait структуры binder_thread после её освобождения.	+	–
2	CVE-2020-0041	Уязвимость заключается в записи за границы массива из-за повреждения транзакции при обработке вредоносного объекта типов BINDER_TYPE_PTR и BINDER_TYPE_FDA.	+	–
3	CVE-2022-20421	Уязвимость заключается в наличии состояния гонки при взаимодействии со структурой binder_proc, в результате которого возможен доступ к данной структуре после освобождения.	+	–
4	CVE-2023-20938	Уязвимость заключается в некорректной валидации пользовательских данных, в том числе обработке объектов, содержащих поле offsets_size без выравнивания.	+	–

Как видно на табл. 5, внедрение средства противодействия в состав программного кода операционной системы Android привело к тому, что все 4 уязвимости в программном коде драйвера системы межпроцессного взаимодействия Binder не были успешно проэксплуатированы.

Заключение

Таким образом, в данной работе был предложен метод противодействия несанкционированному повышению привилегий в операционной системе Android с применением технологии аппаратной виртуализации архитектуры ARM.

Основой предложенного метода является использование сочетания динамической инструментации ядра операционной системы и двухуровневой трансляции адресного пространства архитектуры ARM для контроля доступа к ключевым структурам ядра операционной системы Android.

Также, было разработано программное средство, реализующее предложенный метод и проведены его экспериментальные исследования.

В качестве направления дальнейших исследований можно выделить использование технологии доверенной виртуализации, предоставляемой технологией ARM TrustZone для обеспечения контроля доступа не только к ключевым структурам ядра операционной системы Android, но и контроля доступа к структурам ядра доверенной среды исполнения.

Литература (References)

1. CounterPoint. Global Smartphone OS Market Share. URL: counterpointresearch.com/insights/global-smartphone-osmarket-share/. — (дата обращения (date accessed): 04.06.2025).

2. IT Threat Evolution Q2 2024 Mobile Statistics. URL: securelist.com/malware-report-q1-2025-mobile-statistics/116676/. — (дата обращения (date accessed 05.06.2025).

3. Armv8-A virtualization. URL:developer.arm.com//media/Arm%20Developer%20Community/PDF/Learn%20the%20Architecture/Armv8-A%20virtualization.pdf?revision=a765a7df1a00-434d-b241-357bfda2dd31. — (дата обращения (date accessed 12.06.2025).

4. Credentials in Linux – The Linux Kernel documentation. URL: kernel.org/doc/html/v6.6/security/credentials.html. – (дата обращения (date accessed 15.06.2025).

5. proc_kallsyms(5) – Linux manual page. URL: man7.org/linux/man-pages/man5/proc_kallsyms.5.html. – (дата обращения (date accessed 18.06.2025).

6. Vmlinux-to-elf. README.md. URL: github.com/marin-m/vmlinux-to-elf/blob/master/README.md. — (дата обращения (date accessed 18.06.2025).

7. Capabilities (7) – Linux manual page. URL: man7.org/linux/man-pages/man7/capabilities.7.html. — (дата обращения (date accessed 20.06.2025).

8. Trusted Firmware-A Documentation. URL: trustedfirmware-a.readthedocs.io/en/latest/. — (дата обращения (date accessed 22.06.2025).

9. Keystone – The Ultimate Assembler. URL: keystone-engine.org/. — (дата обращения (date accessed 24.06.2025).

10. Capstone – The Ultimate Disassembler. URL: capstone-engine.org/. — (дата обращения (date accessed): 24.06.2025).

11. Binder. URL: developer.android.com/reference/android/os/Binder. — (дата обращения (date accessed): 01.07.2024).

12. NVD – CVE-2019-2215.URL: nvd.nist.gov/vuln/detail/cve-2019-2215 — (дата обращения (date accessed): 03.07.2025).

13. NVD – CVE-2020-0041. URL: nvd.nist.gov/vuln/detail/cve-2020-0041— (дата обращения (date accessed): 03.07.2025).

14. NVD – CVE-2022-20421. URL: nvd.nist.gov/vuln/detail/cve-2022-20421 — (дата обращения (date accessed): 03.07.2025).

15. NVD – CVE-2023-20938. URL: nvd.nist.gov/vuln/detail/cve-2023-20938 — (дата обращения (date accessed): 03.07.2025).

Дата поступления: 7.08.2025

Дата публикации: 25.09.2025