Санкт-Петербургский государственный университет

Программная инженерия
Кафедра системного программирования

Гордиенко Егор Александрович

Разработка инструмента для извлечения данных со смартфонов Motorola

Отчет о прохождении производственной практики

Научный руководитель:
ст. преп. Куликов Е. К.

Консультант:
архитектор ПО, ООО “Белкасофт” Тимофеев Н. М.

Санкт-Петербург
2021

Оглавление

Введение

Смартфоны стали частью нашей повседневной жизни. При этом существенно увеличилось и число совершаемых с их помощью преступлений [10]. Решением этой проблемы занимается цифровая криминалистика – отрасль судебной науки, которая изучает извлечение и исследование данных, обнаруженных на цифровых устройствах, зачастую в связи с компьютерными преступлениями. Она активно используется сотрудниками правоохранительных органов, следственных комитетов, МВД. В современном мире данные, хранящиеся на мобильных устройствах, могут служить одними из главных свидетельств виновности или невиновности подозреваемого в преступлении. Доказательством может являться информация из мессенджеров, браузеров, почтовых ящиков, медиафайлов, системных файлов и т.д. Криминалистам важно быстро и просто уметь извлекать улики с различных устройств, в том числе и смартфонов, работающих на ОС Android.

Смартфоны компании Motorola хоть и не являются самыми популярными среди мобильных устройств на ОС Android, но все равно занимают определенную долю рынка (2.5%) [7], и имеют своих пользователей, в том числе и среди преступников, которые понимают, что чем более распространено устройство, тем больше вероятность, что инструменты компьютерной криминалистики умеют извлекать из них данные. Кроме того, в продукте Belkasoft Evidence Center [9], разрабатываемом компанией Belkasoft, производящей ПО для цифровой криминалистики, поддержка моделей этих смартфонов еще не обеспечена, поэтому важно также уметь работать и с устройствами данного производителя.

Для извлечения как можно большего объема информации необходимо получить root-права, т.е. права суперпользователя, которые обеспечивают неограниченный доступ к файловой системе смартфона. Однако получение root-прав варьируется от устройства к устройству и не всегда является простой задачей. Нередко для этого приходится прибегать к различным уязвимостям. Большой простор для их поиска представляет bootloader – загрузчик, который отвечает за запуск операцион-

ной системы. При этом некоторые из уязвимостей своевременно устраняются, но многие остаются. Поэтому необходимо проанализировать, какие из них позволят получить root-доступ к смартфонам Motorola и установить, на каких моделях они применимы.

1. Цели и задачи

Целью данной работы является создание инструмента для снятия данных с ряда моделей Motorola.

Для достижения цели были поставлены следующие задачи:

- Изучение механизма bootloader и обзор его уязвимостей в различных моделях смартфонов Motorola
- Разработка инструмента, извлекающего данные со смартфона с использованием найденных уязвимостей
- Выбор ряда тестовых устройств и апробация инструмента на них

2. Обзор предметной области

2.1. Хранение данных

Постоянная память мобильных устройств состоит из логический частей – разделов. Их структура различается у разных поставщиков и платформ. Однако большинство смартфонов на базе ОС Android имеют схожую таблицу разделов, и обычно память устройства содержит следующие основные части [3] :

Boot: в этом разделе хранится загрузочный образ, который состоит из ядра Linux и RAM-диска корневой файловой системы.

Recovery: хранит минимальную версию ОС Android (ядро и RAM-диск) и предоставляет различные инструменты для работы на низком уровне (например, обновление ОС).

System: этот раздел содержит платформу Android, библиотеки, системные и предустановленные приложения. После загрузки ОС Android этот раздел монтируется по пути `/system`.

Userdata: в этом разделе хранятся пользовательские данные, включая данные установленных приложений, документы, изображения, аудио, видео и т. д. Он монтируется как путь `/data` (раздел данных) после загрузки системы.

SDcard: если в телефоне используется внешняя SD-карта, путь `/sdcard` указывает на отдельный раздел SD-карты, в противном случае путь `/sdcard` включается в раздел Userdata. Путь `/sdcard` содержит данные, включающие изображения и видео, снятые приложением камеры смартфона, и загруженные файлы.

2.2. Подходы к извлечению

Для получения данных можно воспользоваться разными методами. Существующие решения условно разделяют на две категории: физическое и логическое извлечение.

Физическое извлечение заключается в побитовом копировании всего раздела данных Android устройства. Например, исследователи могут напрямую подключать провода к CPU устройства и, используя протокол JTAG, управлять процессором, тем самым получая побитовое представление флеш-памяти. Также возможно извлечь микросхемы флэш-памяти NAND с печатной платы устройства и получить физические данные с помощью специальных аппаратных средств. При этом, учитывая сложность операции, велика вероятность повредить чип и информацию на нем. Кроме того, получение полного доступа ко внутренней памяти смартфона зависит от операционной системы и механизмов защиты, установленных производителем, требует индивидуального подхода к каждому устройству и углубленных знаний в области аппаратного обеспечения.

Логическое извлечение заключается во взаимодействии с операционной системой устройства через API. Этот подход не требует взаимодействия с внутренними компонентами устройства – достаточно USB соединения с компьютером. При этом имеются некоторые ограничения, например, невозможно восстановить то, что было удалено. Типичными данными, доступными посредством логического извлечения, являются журналы вызовов, SMS, MMS, изображения, видео, аудиофайлы, контакты, календари и данные приложений. Зачастую при этом подходе используется агент – очень небольшое приложение, которое временно устанавливается на устройство. Он действует как любое другое стороннее приложение, при этом не перезаписывая никаких данных, а после завершения операции автоматически удаляется. Логический способ позволяет быстро и без особых усилий со стороны пользователя получить важные данные со смартфона, поэтому он был выбран как основной в данной работе.

2.3. root-права и bootloader

В общем случае ОС Android предоставляет доступ только к единственному разделу, который монтируется по пути `/storage/device` и

содержит такие пользовательские данные, как фотографии и видеофайлы. Однако гораздо больший интерес представляют разделы userdata и sdcard, но доступ к ним по умолчанию ограничен. Для решения этой проблемы необходимо получить root-доступ, или права суперпользователя. Они позволяют снять ограничения производителя, манипулировать системными приложениями и иметь доступ к любому файлу системы ОС Android. Получение root-прав зависит от устройства, однако почти всегда требует разблокировки bootloader.

Bootloader (загрузчик) – это программа, которая загружает операционную систему смартфона. Поскольку bootloader является важным компонентом процесса загрузки, он хранится в энергонезависимой памяти (например, флэш-памяти). Загрузчики написаны поставщиками устройств и предназначены для оборудования, на котором они работают. Устройства Android имеют два состояния: locked (заблокировано), где целостность ОС гарантируется посредством безопасной / проверенной загрузки [4], и unlocked (разблокировано), когда нет гарантии безопасности. Разблокировка загрузчика позволяет поставить пользовательскую прошивку и обычно является первым шагом к получению root-прав. Motorola позволяет разблокировать bootloader через официальный сайт: пользователи должны зарегистрировать серийный номер своего устройства Motorola, чтобы получить специальный код. Однако разблокировка загрузчика в последних устройствах Motorola уничтожает криптографические ключи, которые используются для расшифровки раздела данных, и требует полного сброса настроек до заводских, прежде чем устройство можно будет использовать. Поскольку потеря пользовательских данных недопустима, для получения root-доступа необходимо разблокировать bootloader без восстановления заводских настроек, или вовсе обойтись без разблокировки.

2.4. Устройство загрузчика

На самом деле, bootloader состоит из нескольких частей. (Рис. 1) Сначала выполняется PBL (первичный загрузчик), или BootROM, ко-

торый находится в аппаратно защищенном от записи участке памяти и служит минимальной цели: разрешить устройству загрузиться, а также аутентифицировать и загрузить следующую часть загрузочной цепочки.

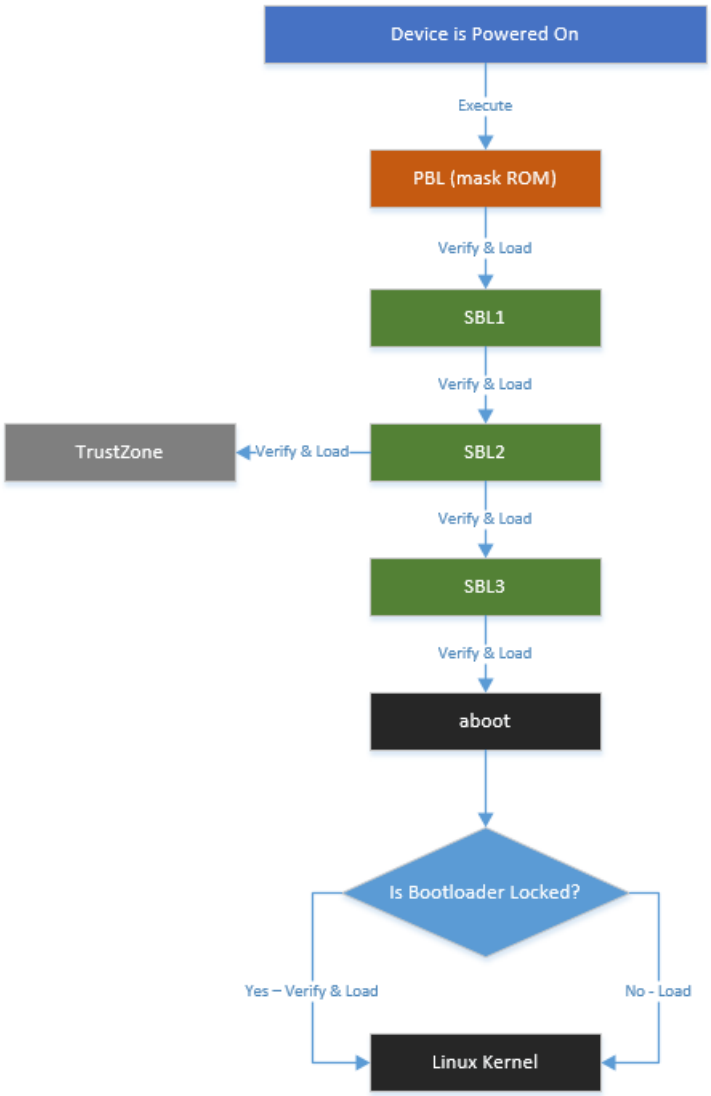

Рис. 1: Устройство bootloader

Затем выполняются два вторичных загрузчика, SBL1 и SBL2. Их основная обязанность – загрузить различные процессоры на однокристальных системах и настроить их так, чтобы они были готовы к работе. Также SBL2 отвечает за загрузку TrustZone – механизма, обеспечивающего безопасность и интегрированного во многие современные процессоры ARM. TrustZone управляет защищенной памятью, которую нельзя

прочитать из небезопасного места, независимо от уровня привилегий.

Далее в загрузочной цепочке загружается третий и последний вторичный загрузчик, SBL3. Этот загрузчик, среди прочего, проверяет и загружает загрузчик ОС Android – aboot. Именно aboot отвечает за блокировку bootloader и взаимодействует с интерфейсом fastboot.

Таким образом реализуется Secure Boot – цепочка доверия, когда каждый загрузчик проверяет сертификат следующего, пока очередь не дойдет до aboot и управление не передастся ядру Linux.

2.5. Разблокировка bootloader в Motorola с использованием уязвимости TrustZone

В статьях [8] и [1] авторы рассматривают возможность разблокировки загрузчика Motorola с использованием уязвимости TrustZone.

С помощью дизассемблирования полного образа bootloader было установлено, что для разблокировки загрузчика aboot взаимодействует с TrustZone с помощью специальной функции, вызывающей ассемблерные инструкции ARM, – SMC (Secure Monitor Call). В результате статического анализа дизассемблированного кода TrustZone оказалось, что механизм разблокировки работает с помощью QFuse. Это аппаратный компонент, который обеспечивает “одноразовую запись” в часть памяти. Каждый QFuse представляет собой отдельный бит, при этом нетронутый QFuse является нулем, а “сгоревший” – единицей. Операция выставления бита в QFuse необратима. Таким образом, необходимо выставить единственный бит в QFuse, к которому обращается TrustZone, и загрузчик будет считаться разблокированным.

Однако в защищенной памяти TrustZone есть глобальный флаг, который выставляется aboot и предотвращает все последующие вызовы для изменения битов QFuse. Для обхода этого ограничения используется уязвимость TrustZone, заключающаяся в том, что вызов SMC с определенным аргументом позволяет “незащищенному” ядру Linux получить значения определенной области защищенной памяти. Второй аргумент SMC отвечает за физический адрес, куда копируется защи-

щенная память. При этом не проверяется, что физический адрес отвечает незащищенной области, поэтому он может быть использован для перезаписи памяти в защищенном регионе, включая тот самый глобальный флаг.

Данная уязвимость позволяет разблокировать bootloader, не удаляя пользовательские данные. На момент публикации статей (2013, 2016 год) она работала на устройствах Motorola с чипсетами Qualcomm MSM8960 (Moto X, Atrix HD, Razr HD, Razr M – модели, выпускавшиеся в 2012-2014 годах).

2.6. Уязвимость в fastboot для обхода Secure Boot

В статье [6] авторы предлагают различные способы обхода безопасной загрузки с помощью интерфейса fastboot на ряде моделей Motorola, включая Moto G4, Moto G5, Nexus 6, при этом не разблокируя загрузчик.

Fastboot – это протокол Android, который позволяет компьютеру связываться с bootloader-ом устройства через USB. Клиент fastboot, работающий на хосте, позволяет пользователю прошивать или стирать разделы устройства с помощью инструментов командной строки, а также может загружать устройство Android с пользовательским образом восстановления, не меняя при этом разделов. Производители могут добавлять собственные команды в fastboot на свое усмотрение.

В загрузчике Motorola было найдено несколько аргументов, которые могут быть изменены через fastboot: `bootmode`, `console`, `fsg-id` и `carrier`. Последние три могут содержать произвольные значения (фиксированного размера), и в результате встраиваться в командную строку ядра Linux, которая используется несколькими сущностями ОС, включая процессы в пользовательском пространстве (например, `init`). Команда для аргумента `fsg-id` в интерфейсе fastboot выглядит следующим образом: `fastboot oem config fsg-id value`.

При включении смартфона aboot проверяет подлинность разделов boot и recovery, загружает ядро Linux и `initramfs` – начальную фай-

ловую систему в оперативной памяти. Также aboot подготавливает командную строку ядра, стартовый и конечный адрес `initramfs`, а затем передает управление непосредственно ядру Linux. Оно, в свою очередь, сопоставляет переданные физические адреса виртуальным, которые используются для загрузки `initramfs` в `rootfs`, корневую файловую систему Linux. Оттуда и запускаются первые процессы в пользовательском пространстве.

В ARM/64 оказалось возможным через аргумент `initrd` командной строки ядра контролировать физический адрес, откуда загружается `initramfs`. Этот аргумент перезаписывает начальные значения, находящиеся в aboot. Таким образом, с помощью fastboot можно изменить адрес так, чтобы он указывал на специально подделанный `initramfs`, который позволит, например, получить неограниченный root-доступ.

Для загрузки поддельного `initramfs` можно воспользоваться загрузочным механизмом aboot через USB с помощью команды `flash`, которая скачивает данные на устройство и затем пытается прошить его (что невозможно, так как загрузчик заблокирован). При этом данные будут находится по фиксированному адресу `SCRATCH_ADDR`, видному ядру Linux. Получить этот адрес для разных моделей можно с помощью дизассемблирования aboot и анализа функции `get_scratch_address`.

Создать поддельный `initramfs` можно с помощью извлечения и изменения предустановленных официальных архивов, находящихся в ROM-памяти устройств. Например, чтобы создать `initramfs` с правами суперпользователя необходимо выставить SELinux в `permissive` режим, изменить `adbd`, чтобы он оставался в режиме root, убрать параметр `dm-verity` и сделать некоторые другие модификации. Возможно также изменить `initramfs` так, чтобы он выполнял и некоторую другую необходимую логику.

В результате уязвимость в интерфейсе fastboot позволила, имея измененный архив `initramfs` с root-правами, загрузить его в память по фиксированному физическому адресу и заставить ядро Linux запуститься с этого адреса, тем самым получив root-доступ при заблокированном загрузчике. (Рис. 2) Данный способ позволяет обойти проверки безопас-

```
$ fastboot oem config fsg-id "a initrd=0xA2100000,1588598"
$ fastboot flash aleph initroot-cedric.cpio.gz
$ fastboot continue
$ adb shell
cedric:/ # id
uid=0(root) gid=0(root) groups=0(root), [...] context=u:r:
     kernel:s0 context=u:r:kernel:s0
cedric:/ # getenforce
Permissive
cedric:/ #
```

Рис. 2: Успешная эксплуатация уязвимости на Moto G5

ности Secure Boot и запустить на устройстве произвольный RAM-образ, в который можно добавить свою логику, например, получения ключей шифрования и извлечения раздела данных, при этом даже не доходя до этапа загрузки ОС Android.

Стоит отметить, что уязвимость была исправлена компанией Motorola в OTA-обновлении от мая 2017 года [2], однако есть значительное количество устройств, которые все еще работают с необновленными версиями загрузчика, либо поддержка которых была прекращена. Кроме того, если устройство имеет разблокированный bootloader, то можно установить более старую версию загрузчика, подверженную уязвимости.

2.7. Qualcomm Emergency Download Mode

Многие смартфоны, оснащенные чипсетами Qualcomm, имеют Emergency Download Mode (EDL), который можно запустить вместо загрузки PBL. Этот режим обычно включается путем короткого замыкания определенных контактов во время включения телефона, с помощью специальных USB-кабелей, либо через `adb`.

Для использования этого режима необходимо иметь специальные программы с цифровой подписью производителя, которые позволяют взаимодействовать с EDL. Они оказались в открытом доступе для смартфонов Moto G4 Plus, Nexus 6 и Nexus 6P. В статье [5] был проведен

анализ этих программ и было выявлено, что в них используется протокол Firehose, с помощью которого можно через USB-соединение отправлять команды, записанные в XML. Определенные теги `program` и `patch` позволяют прошить произвольные разделы устройства, в том числе и bootloader.

3. Реализация

3.1. Описание решения

Уязвимость с использованием TrustZone была найдена еще в 2013 году и работала на устройствах, которые на сегодняшний день устарели и не являются актуальными, при этом на более поздних смартфонах она была исправлена производителями. Поэтому для реализации было решено взять описанную раннее уязвимость oem-команд интерфейса fastboot. Кроме того, исследование режима Qualcomm EDL показало, что с помощью него возможно понизить версию загрузчика до подверженной уязвимости fastboot, избежав удаления пользовательских данных. Таким образом, круг поддерживаемых устройств включает в себя не только смартфоны с обновлением безопасности от мая 2017 года, но и имеющие более поздние версии программного обеспечения.

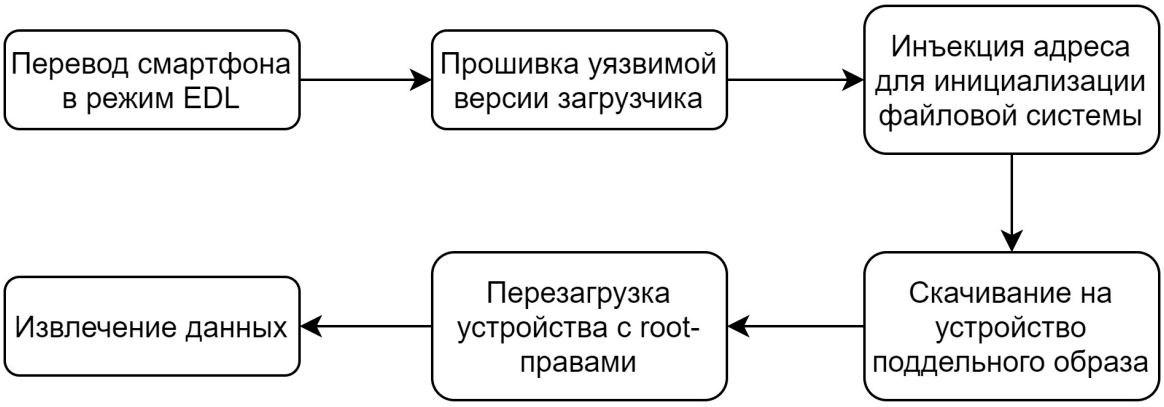

Рис. 3: Последовательность действий программы

Схема работы программы представлена на Рис. 3. Решение включает в себя перевод смартфона в EDL, понижение версии загрузчика и эксплуатацию уязвимости fastboot. Наконец, происходит извлечение логического образа данных с устройства на компьютер. Стоит отметить, что первые два этапа являются опциональными: если устройство имеет версию bootloader-a, подверженную уязвимости, то использование EDL не имеет необходимости.

3.2. Используемые инструменты и библиотеки

В качестве языка реализации был выбран Python 3, как удобный кроссплатформенный инструмент, имеющий встроенные библиотеки для вызова процессов (adb, sh) и работы с файловой системой (извлечение файлов с устройства). Для автоматизации процессов понижения версии загрузчика через EDL и использования уязвимости были написаны отдельные bash-скрипты (основу которых составляют материалы из статьи [6]), вызывающиеся внутри Python-программы.

Основными инструментами для работы с Android-устройством являются adb и fastboot. ADB (Android Debug Bridge) – утилита командной строки, позволяющая управлять устройством посредством USB-соединения. Она обеспечивает взаимодействие компьютера и смартфона, позволяя войти в командную строку устройства, перемещать файлы, устанавливать приложения, просматривать информацию об устройстве и т.д. Fastboot – также инструмент командной строки, который обеспечивает работу с внутренними компонентами системы. Основной функцией fastboot является возможность перепрошивки разделов. Кроме того, данный интерфейс является ключевым компонентом используемой уязвимости.

3.3. Детали реализации

3.3.1. Понижение версии загрузчика с помощью EDL

На данном шаге проверяется версия последнего обновления безопасности системы – если она старше мая 2017 года, то уязвимость fastboot работать не будет, и необходимо использовать режим EDL. Перевод устройства в этот режим осуществляется через adb посредством команды `adb reboot edl`. Кроме того, для работы с EDL необходимы специальные бинарные файлы с цифровой подписью производителя (обычно под названием `programmer.mbn`). Эти файлы используются утилитой qboot для выполнения команды `blank-flash`, которая затем переводит устройство в режим fastboot. Из этого режима становится возможным

прошить разделы устройства вне зависимости от того, разблокирован загрузчик или нет. Стоит отметить, что каждая часть в цепочке загрузчика имеет цифровую подпись и проверяется предыдущей частью. Однако производители устройств Motorola не делают проверку на актуальность версии загрузчика, что позволяет прошить устройство более старой, уязвимой версией aboot, имеющей корректную цифровую подпись.

3.3.2. Использование уязвимости fastboot

Через интерфейс fastboot возможно изменить встроенные производителем параметры произвольным образом. Так, в параметр `fsg-id` помещается произвольный код, который отвечает за физический адрес в памяти, откуда происходит загрузка начальной файловой системы. Образ данного инициализатора был изменен таким образом, чтобы устройство загружалось с root-правами. Так как по умолчанию загрузчик заблокирован, прошивать произвольные разделы невозможно, однако при попытке это сделать загружаемые файлы сохраняются по определенному физическому адресу. Данный адрес используется в конфигурации переменной `fsg-id`.

В процессе эксплуатации уязвимости возникала следующая проблема: после прошивки устройство входило в бесконечный цикл загрузки. Это было связано с тем, что образ загрузчика повреждался из-за того, что перед запуском ядра загрузчик помещал лишнюю информацию по адресу, куда загружался инициализатор. Решение проблемы было предложено в статье [6]: перед образом былы положены пустые данные в размере 32 Мб, чего хватило, чтобы образ инициализатора остался целостным. Чтобы физический адрес, помещаемый в параметр `fsg-id`, стал корректным, к нему был прибавлен размер добавленных пустых данных.

После выполения всех шагов, в результате успешного применения уязвимости устройство перезагружается с правами суперпользователя.

3.3.3. Извлечение данных с устройства

В созданном решении реализовано два способа извлечения данных. Первый из них полностью извлекает раздел userdata. Для этого сначала определяется настоящее название требуемого раздела, которое ищется в системной папке, содержащей соответствия физических и логических разделов. Затем с помощью команды `adb pull` извлекается image-файл с пользовательскими данными. Проблема такого подхода заключается в том, что многие современные смартфоны используют Full Disk Encryption – все разделы шифруются с помощью некоторого ключа, вшитого в устройство. Поэтому получить полный раздел userdata возможно только в зашифрованном состоянии.

Второй способ заключается в логическом снятии данных. С помощью командной строки устройства просматриваются все доступные файлы в каталоге `data` и происходит их копирование в виде tar-архивов на компьютер посредством исполнения команды `adb exec-out`. При этом данные извлекаются в сжатом, но незашифрованном виде.

3.4. Апробация на тестовом устройстве

В качестве тестового устройства использовался смартфон Motorola Nexus 6 32Gb (системное название – shamu) с версией Android 7.1.1, на котором было установлено обновление безопасности от октября 2017 года. Написанная программа была успешно протестирована несколько раз с разными версиями официальных заводских образов Android. Стоит также отметить, что использование уязвимости никак не влияет на пользовательские данные – они не удаляются и не изменяются. В результате удалось получить root-доступ (Рис. 4) и извлечь доступные файлы, в числе которых, например, есть базы данных контактов и вызовов, совершенных на устройстве (Рис. 5).

```
goegor@goegor-FX503VD:~/Documents/edlrooter$ adb shell
shamu:/ # id
uid=0(root) gid=0(root) groups=0(root),1004(input),1007(log),1011(adb),1015(sdca
rd_rw),1028(sdcard_r),3001(net_bt_admin),3002(net_bt),3003(inet),3006(net_bw_sta
ts),3009(readproc) context=u:r:su:s0
```

Рис. 4: root-права на тестовом смартфоне Nexus 6

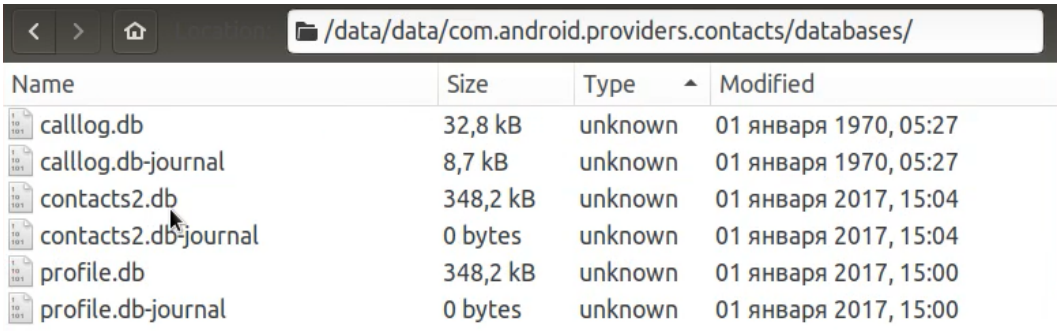

Рис. 5: Извлеченные файлы БД контактов и журнала вызовов

К сожалению, провести апробацию на других устройствах не вышло, поскольку не удалось получить доступ к необходимым моделям смартфонов. Однако написанная программа легко расширяема, что позволит обеспечить поддержку и других уязвимых устройств (в том числе Moto G5, G5 Plus, G4, G4 Play), имея необходимые образы.

4. Результаты

В рамках данной курсовой работы были выполнены следующие задачи:

- Выбран подход к извлечению данных, изучен механизм bootloader и проведен обзор его уязвимостей
- Разработана и реализована программа, эксплуатирующая уязвимость и извлекающая образ пользовательского раздела устройства
- Проведена апробация полученного результата на тестовом устройстве Motorola Nexus 6

Результаты данной работы в ближайшее время будут интегрированы в Belkasoft Evidence Center.

Автор выражает признательность компании Belkasoft за предоставление темы данной курсовой работы. А также отдельную благодарность ее сотрудникам, Никите Тимофееву и Михаилу Виноградову, за ценные советы при планировании исследования, рекомендации по оформлению курсовой и предоставленное оборудование.

Список литературы

[1] Bits Please! Unlocking the Motorola Bootloader. — 2016. — URL: http://bits-please.blogspot.com/2016/02/unlocking-motorola-bootloader.html (online; accessed: 08.12.2020).

[2] Google. Android Security Bulletin May 2017. — 2017. — URL: https://source.android.com/security/bulletin/2017-05-01#eop-in-motorola-bootloader (online; accessed: 08.12.2020).

[3] Google. Partitions overview. — 2020. — URL: https://source.android.com/devices/bootloader/partitions?hl=en (online; accessed: 08.12.2020).

[4] Google. Verifying Boot. — 2020. — URL: https://source.android.com/security/verifiedboot/verified-boot (online; accessed: 08.12.2020).

[5] Hadad Roee Hay & Noam. Exploiting Qualcomm EDL Programmers. — 2018. — URL: https://alephsecurity.com/2018/01/22/qualcomm-edl-1/ (online; accessed: 08.12.2020).

[6] Hay Roee. fastboot oem vuln: Android Bootloader Vulnerabilities in Vendor Customizations // 11th USENIX Workshop on Offensive Technologies (WOOT 17). — Vancouver, BC : USENIX Association, 2017. — URL: https://www.usenix.org/conference/woot17/workshop-program/presentation/hay.

[7] Mobile Vendor Market Share Worldwide. — 2020. — URL: https://gs.statcounter.com/vendor-market-share/mobile/worldwide (online; accessed: 08.12.2020).

[8] Rosenberg Dan. Unlocking the Motorola Bootloader. — 2013. — URL: http://blog.azimuthsecurity.com/2013/04/unlocking-motorola-bootloader.html (online; accessed: 08.12.2020).

[9] Домашняя страница продукта Belkasoft Evidence Center. — 2020. — URL: https://belkasoft.com/x (online; accessed: 08.12.2020).

[10] Число преступлений с использованием смартфонов в России резко увеличилось. — 2020. — URL: https://1prime.ru/telecommunications_and_technologies/20200914/832019974.html (online; accessed: 08.12.2020).