Санкт-Петербургский государственный университет

МУРАВЬЕВ Кирилл Ильич

Выпускная квалификационная работа

Использование ПК под управлением ОС Windows в качестве HID для Android-устройств

Уровень образования: бакалавриат

Направление *02.03.03 «Математическое обеспечение и администрирование информационных систем»*

Основная образовательная программа *СВ.5006.2019 «Математическое обеспечение и администрирование информационных систем»*

Научный руководитель:
доцент кафедры системного программирования, к.т.н., Ю. В. Литвинов

Консультант:
архитектор ПО ООО «Цифровая корпоративная защита», Н. М. Тимофеев

Рецензент:
разработчик ПО ООО «Цифровая корпоративная защита» А. А. Губанов

Санкт-Петербург
2023

Saint Petersburg State University

Kirill Muravev

Bachelor's Thesis

Usage of a Windows PC as a HID for Android devices

Education level: bachelor

Speciality *02.03.03 "Software and Administration of Information Systems"*

Programme *CB.5006.2019 "Software and Administration of Information Systems"*

Scientific supervisor:
C.Sc., System Programming chair docent Y.V. Litvinov

Consultant:
software architect at OOO "Digital Corporate Protection" N.M. Timofeev

Reviewer:
software developer at OOO "Digital Corporate Protection" A.A. Gubanov

Saint Petersburg
2023

Оглавление

Введение

В задачах компьютерной криминалистики важную роль играет информация, собранная с цифровых устройств. При этом мобильные устройства — смартфоны — часто могут предоставить больше данных, чем флеш-накопители, ноутбуки и компьютеры, поскольку:

- согласно подсчётам[1,2], смартфон есть почти у каждого человека, тогда как лишь половина населения имеет компьютер;
- функциональность смартфонов широко используется для коммуникации, геолокации, финансовых операций; для управления паролями и авторизации на различных сервисах;
- практически все они оснащены фотокамерой.

Но устройства, исследуемые экспертами-криминалистами, почти всегда защищены от неавторизованного доступа. Разработчики мобильных операционных систем предоставляют различные механизмы защиты, поэтому «взлом», поиск уязвимостей и эксплоитов в ОС — трудоёмкая задача, не имеющая стандартного решения. Однако для авторизации на устройстве достаточно знать код разблокировки экрана: PIN-код, текстовый пароль или графический ключ. Распространённые мобильные ОС используют один из этих способов разблокировки в качестве вспомогательного, даже если на устройстве включена биометрическая защита. Далее в работе будут рассматриваться только текстовые и числовые (PIN) пароли, поскольку они вводятся с клавиатуры.

Для ввода пароля в ОС Android может использоваться не только экранная клавиатура, но и физическая, подключаемая по Bluetooth или USB. Bluetooth-соединение требует разблокировать смартфон для подключения (сопряжения) клавиатуры, а следовательно не может быть использовано, если пароль неизвестен. В свою очередь возможность

[1]Количество пользователей смартфонов: https://statista.com/statistics/330695/number-of-smartphone-users-worldwide (дата обращения: 25.05.2023).

[2]Количество пользователей компьютеров: https://statista.com/statistics/748551/worldwide-households-with-computer (дата обращения: 25.05.2023).

подключения клавиатуры по USB можно эксплуатировать: подключить к смартфону устройство, которое будет распознано как клавиатура, но не являться таковой. Эмулируя клавиатуру, можно автоматически отправлять на смартфон предварительно записанные сигналы клавиш и выполнять скрипты. Описанный подход далее будем называть BadUSB. Он может быть использован для перебора паролей экранной блокировки методом «грубой силы», однако не позволяет обойти стандартную защиту от этого метода: ограниченное количество попыток ввода пароля и необходимость ожидания между попытками. Отметим также, что BadUSB можно адаптировать для ввода графического ключа путём эмуляции движений USB-мыши; однако безопасности графического ключа посвящены отдельные исследования (см. [3, 7]), и это остаётся за рамками работы.

Все публично известные реализации BadUSB используют дополнительное оборудование: например, другой смартфон, как в проект Android-PIN-Bruteforce[3]. Но поскольку основным рабочим инструментом эксперта-криминалиста является современный ноутбук, возникла потребность эмулировать клавиатуру стандартными средствами — по возможности не модифицируя аппаратное обеспечение ноутбука и используя возможности ОС.

В данной работе описывается метод, позволяющий сконфигурировать ПК так, чтобы эмулировать USB-клавиатуру. В качестве доказательства осуществимости реализуется приложение, использующее алгоритм «грубой силы» для разблокировки экрана Android-смартфона.

[3]Проект Android-PIN-Bruteforce для Kali Nethunter: https://github.com/urbanadventurer/Android-PIN-Bruteforce (дата обращения: 25.05.2023).

1. Постановка задачи

Цель данной работы — реализовать приложение для ПК, позволяющее разблокировать экран Android-смартфона, эмулируя клавиатуру через USB-соединение.

Для достижения этой цели были поставлены следующие задачи.

1. Провести обзор технологий, которые могут быть использованы для эмуляции клавиатуры со стороны ПК.
2. Описать требования к приложению со стороны аппаратного обеспечения.
3. Спроектировать приложение, позволяющее автоматически отправлять на смартфон сигналы нажатия клавиш и перебирать пароли экранной блокировки.
4. Реализовать спроектированное приложение.
5. Провести тестирование реализации на смартфонах с портами micro-USB и USB Type-C.

2. Обзор

Помимо эмуляции клавиатуры, существуют различные методы снятия блокировки Android-смартфона без знания пароля. Рассмотрим те из них, которые предоставляют доступ к информации, содержащейся в устройстве, а не уничтожают её.

Важно отметить, что с развитием ОС некоторые методы извлечения информации становятся неприменимы. Так, начиная с Android 6.0, файловая система смартфона полностью шифруется (см. седьмую главу [10]), поэтому практика “chip-off”, т.е. выпаивание микросхемы памяти и прямое считывание с неё, теперь не даёт результата, если ключ шифрования неизвестен.

2.1. Android Debug Bridge

Программное управление и отладка Android-устройств выполняется с использованием протокола Android Debug Bridge. Официальная реализация этого протокола описана в [1] и представляет собой консольную утилиту adb, состоящую из:

- программы-клиента, запускаемой на компьютере эксперта и принимающей его команды;
- «демона» на подключённом устройстве, исполняющего команды;
- сервера на компьютере, обеспечивающего коммуникацию клиента и демона.

Чтобы запустить демона на устройстве с Android 4.2.2 и новее, необходимо выполнение двух условий. Во-первых, в его настройках должна быть разрешена «отладка по USB»; во-вторых, компьютер должен быть авторизован на устройстве, т.е. его RSA-ключ (adb_key.pub) должен содержаться в файле «белого списка ключей» /data/misc/adb/adb_keys[4].

[4]Эта процедура была добавлена в https://android.googlesource.com/platform/system/core/+/d5fcafaf41f8ec90986c813f75ec78402096af2d (дата обращения: 25.05.2023).

Уникальный RSA-ключ генерируется при установке утилиты, поэтому стандартный ключ компьютера эксперта не будет присутствовать в «белом списке».

Предположим, что стандартный ключ был подменён на один из ключей «белого списка», извлечённый, например, с ноутбука, ранее авторизованного на устройстве. Тогда, получив adb-доступ, для разблокировки смартфона можно эмулировать ввод с клавиатуры командой adb shell input text и таким образом перебирать пароли[5]. Отметим, что сбросить пароль или узнать его, сравнивая известный хэш по методу Rainbow Table, по-прежнему невозможно, поскольку соответствующие файлы защищены с помощью привилегий пользователей Linux и недоступны для чтения без root-прав (см. [11]).

2.2. BadUSB

BadUSB — общее название подхода, при котором USB-устройство реализует не предусмотренную его классом функциональность. Идея в том, что такое устройство не требует настройки и может начать работу сразу при подключении, поскольку для многих стандартных классов (например, Human Interface Device, HID) универсальный драйвер поставляется вместе с ОС и загружается автоматически. В частности, ОС Android поддерживает[6] подключение USB-устройств по протоколу USB On-The-Go, начиная с версии Android 3.1. Таким образом макетная плата или смартфон могут имитировать клавиатуру при подключении к ПК. Подробное описание подхода можно найти в докладе [9], а классы USB описаны в [2].

Большинство USB-устройств реализуют одну или несколько строго определённых производителем функций, а для того, чтобы добавить свою функциональность (т.е. реализовать BadUSB), требуется модифицировать прошивку микроконтроллера. Такая модификация не всегда

[5]Пример реализации этого подхода доступен в репозитории https://github.com/pancholiurvish/ALSPC (дата обращения: 25.05.2023).

[6]Объявление о включении USB Host API в Android: https://developer.android.com/about/versions/android-3.1-highlights.html (дата обращения: 25.05.2023).

выполнима, поэтому выпускаются продукты, специально предназначенные для этого: например, Rubber Ducky[7]. Примеры его использования для снятия экранной блокировки Android можно найти в следующих репозиториях:

- проект Android-USB-Rubber-Duck[8] эксплуатирует уязвимость, специфичную для Android 5 с патчем безопасности версии LMY48I;
- проект Ducky_Droid[9] реализует метод перебора паролей со словарём.

Ещё одну известную реализацию подхода BadUSB предоставляет проект Kali Nethunter[10]. Это модифицированная прошивка Android и программа для неё, позволяющая смартфону эмулировать USB-клавиатуру и, аналогично Rubber Ducky, выполнять скрипты на устройствах, к которым он подключается. Упомянутый во введении проект Android-PIN-Bruteforce[11] использует эту возможность для снятия экранной блокировки Android.

Ввиду различий в аппаратном обеспечении, описанных в следующей главе, ни один из упомянутых проектов не может быть адаптирован для работы на ПК.

2.3. USB Data Role Swap

Далее для описания устройств, работающих с USB, будем использовать английские термины «хост» (host) и «девайс» (device), поскольку русский термин «устройство» может относиться к обоим из них.

[7]Реализация BadUSB в форме продукта Rubber Ducky: https://shop.hak5.org/products/usb-rubber-ducky (дата обращения: 25.05.2023).

[8]Репозиторий Android-USB-Rubber-Duck: https://github.com/aluech/Android-USB-Rubber-Duck (дата обращения: 25.05.2023).

[9]Репозиторий Ducky_Droid: https://github.com/gcher90/Ducky_Droid (дата обращения: 25.05.2023).

[10]Проект Kali Nethunter: https://kali.org/docs/nethunter (дата обращения: 25.05.2023).

[11]Проект Android-PIN-Bruteforce для Kali Nethunter: https://github.com/urbanadventurer/Android-PIN-Bruteforce (дата обращения: 25.05.2023).

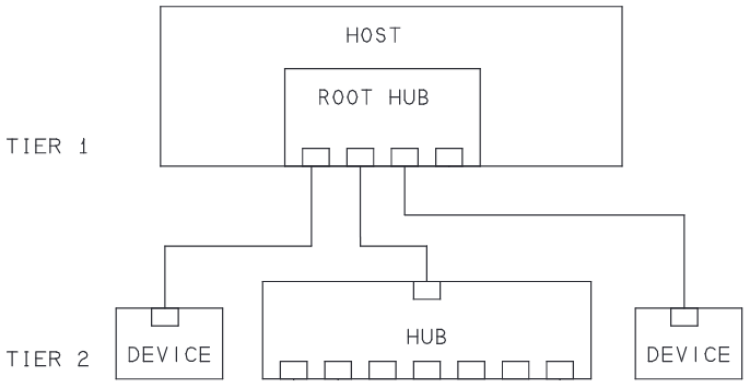

Рис. 1: Физическая топология USB. Источник: [2].

2.3.1. Роли USB-устройств

Логическая топология USB любой версии представляет собой звезду, в которой существуют устройства двух ролей: единственный хост и подключённые к нему девайсы, причём девайсы не могут коммуницировать между собой. Логическая топология отличается от физической (её пример приведён на рисунке 1) только тем, что концентраторы (hub) для неё прозрачны, и считается, что девайсы подключены к хосту напрямую.

Роль устройства определяется в первую очередь его аппаратным обеспечением, для которого используются термины «хост-контроллер» и «девайс-контроллер». Одно физическое устройство может иметь контроллеры обоих типов, тогда на логическом уровне оно будет представлять собой два независимых устройства. В такие устройства, как флешнакопитель, клавиатура и микрофон, устанавливаются контроллеры типа девайс.

Контроллеры, устанавливаемые в Android-смартфоны, тоже являются контроллерами типа девайс, но дополнительно реализуют расширенную спецификацию USB On-The-Go (USB OTG) [13], что позволяет подключать к ним перечисленную периферию. Согласно этой спецификации, смартфон считается “Targeted Host”: он не реализует роль хоста полностью, но может обслуживать ограниченное определяемое производителем подмножество периферийных устройств. Если два USB OTG-устройства подключены друг к другу, хост выбирается случайным об-

разом. Впоследствии, например, по запросу от ОС, одно из устройств может потребовать (assume) или освободить (release) роль хоста. При подключении к «стандартному» хосту Targeted Host всегда получает роль девайса. Стандартным хостом исторически считается ПК.

Фундаментальная часть протокола USB реализуется в прошивке соответствующего контроллера и не может быть модифицирована. Поэтому заключаем, что для эмуляции клавиатуры со смартфона нет препятствий (с точки зрения ролей USB), тогда как для эмуляции с ПК нужен другой подход.

2.3.2. USB Power Delivery

Протокол USB OTG позволяет девайсу брать на себя часть функций хоста, но не наоборот. Последнее может быть полезно, например, для прямой передачи файлов между ноутбуками[12] или чтобы заряжать ноутбук от док-станции (подаваемое на порт напряжение также регламентируется спецификацией USB и зависит от роли устройства).

Поэтому в спецификациях USB Type-C [14] и USB Power Delivery (USB-PD) [15], во-первых, разделены роли питания (power role, используются термины Source и Sink) и роли потока данных (data role, используются термины DFP[13] и UFP[14]); во-вторых, введено понятие Dual Role-устройства, чей контроллер полнофункционально реализует оба набора ролей; в-третьих, разработаны два алгоритма для независимой смены этих ролей. Для нашей задачи ключевое значение имеет смена роли потока данных.

Первый алгоритм описан в разделе 4.5.1.4.1 [14] и рекомендован для устройств, которые не реализуют спецификацию USB-PD. В этом случае роль потока данных определяется ролью питания, а роль питания выбирается при соединении устройств и не может быть изменена впоследствии. В соответствии с этим алгоритмом, при соединении ноутбука

[12]Передача файлов в Apple Target Disk Mode: https://support.apple.com/guide/mac-help/transfer-files-mac-computers-target-disk-mode-mchlp1443/mac (дата обращения: 25.05.2023).

[13]DFP — Downstream Facing Port, т.е. хост.

[14]UFP — Upstream Facing Port, т.е. девайс.

(который «предпочитает»[15] быть Source) и смартфона (который «предпочитает» быть Sink) питание будет подаваться с ноутбука на смартфон. Следовательно, ноутбук окажется в роли хоста, и этот алгоритм для достижения нашей цели не подходит.

Второй алгоритм описан в разделе 4.5.1.4.2 [14] и использует механизмы USB-PD. Отметим, что для его использования оба устройства — ноутбук и смартфон — должны реализовывать спецификацию USB-PD, а это условие не только ужесточает аппаратные требования потенциального решения, но и сужает границы его применимости. Основное преимущество данного алгоритма в том, что он может быть запущен в любой момент любым из двух устройств путём отправки USB-PD сообщения DR_Swap[16]. Теоретически мы можем использовать его, чтобы присвоить ноутбуку роль «девайс», однако практические эксперименты, описанные далее, показали, что этого недостаточно.

2.3.3. USB Alternate Mode

После того как определены роли устройств, хост начинает процесс перечисления девайсов (Enumeration, описан в [2]). В ответ на запрос хоста девайс отправляет набор дескрипторов, соответствующих классам USB, которые он реализует. Особо отметим класс-заглушку Billboard: его единственная задача — избежать «тихого» отказа и сообщить хосту, что девайс должен работать в режиме USB Alternate Mode и не реализует другую функциональность.

USB Alternate Mode — общее название для всех протоколов, которые используют коннектор USB Type-C, но не являются USB. Известные примеры — Intel Thunderbolt и DisplayPort. Подключение по этим протоколам устанавливается в следующем порядке: хост отправляет USB-PD сообщение Enter Mode; затем девайс отвечает списком доступных альтернативных режимов (Alternate Modes); далее хост выбирает режим и требует начать конфигурацию. Следовательно, если хост не способен обрабатывать альтернативные режимы, этот процесс не начнётся,

[15] См. в [14] состояния Try.SRC и Try.SNK соответственно.

[16] DR_Swap — Data Role Swap.

а девайс будет сконфигурирован как Billboard-заглушка.

2.3.4. Практические результаты

В ходе данной работы было изучено поведение контроллеров USB на нескольких ноутбуках, смартфонах и стационарных ПК. Для наблюдения за трафиком USB использовался следующий стек:

- ноутбук Lenovo T470 с контроллером USB, реализующим спецификации USB-PD и Intel Thunderbolt Alternate Mode;
- ОС Windows 10;
- утилита WireShark со сниффером USBPcap;
- диспетчер устройств Windows (далее «ДУ»).

Также для наблюдения использовались другие ноутбуки, в том числе без поддержки USB-PD и с ОС Linux, однако результаты оказались идентичными, поэтому их конфигурацию описывать не будем.

Процесс выглядел следующим образом: на наблюдающем ноутбуке запускались утилита WireShark и диспетчер устройств Windows; затем изучаемое устройство и ноутбук соединялись кабелем micro-USB или USB Type-C[17]; далее отслеживался трафик в WireShark или появление новых устройств в ДУ.

Пять проверенных Android-смартфонов разных производителей, и с micro-USB, и с USB Type-C, ожидаемо отображались в ДУ, а в WireShark можно было наблюдать ход процедуры перечисления. Из девяти ноутбуков (MacBook Air 2019, Lenovo Yoga 920, Lenovo ThinkPad 470, другие Lenovo, ASUS и HP) и одного ПК (материнская плата ROG Strix Z270G) с портами Type-C, а также одного ПК (Raspberry Pi Model B) с micro-USB, аналогичный результат был получен только при наблюдении за MacBook Air 2019 и Lenovo ThinkPad 470. На соединение с остальными устройствами наблюдающий ноутбук никак не реагировал:

[17]Использовались как пассивные кабели Type-A — Type-C, так и активные Type-C — Type-C с поддержкой Thunderbolt.

на обоих концах кабеля оказывались устройства в роли хост[18], и подключения не происходило.

Lenovo ThinkPad 470 был определён как USB Billboard-девайс с контроллером Texas Instruments TPS69582. В соответствующей документации [12] пояснено, что Billboard — единственный класс USB, который реализован в прошивке контроллера. Код прошивки закрыт, а официальная утилита для изменения конфигурации прошивки не позволяет добавить реализацию других классов, в том числе необходимого HID. Таким образом, хотя этому ноутбуку возможно назначить USB роль девайса, однако даже его невозможно использовать для эмуляции клавиатуры напрямую.

MacBook Air 2019 был определён как USB-девайс с VendorID=05AC и ProductID=165F, что позволяет надеяться на успех в эмуляции клавиатуры. Однако согласно требованиям, проектируемое приложение обязано работать на ноутбуках различных производителей и не может зависеть от аппаратного обеспечения Apple, поэтому дальнейшее изучение не производилось.

2.4. Bluetooth HID profile

Из предыдущей главы делаем вывод, что с использованием лишь встроенного в ноутбук USB-оборудования задача не может быть решена. В то же время единственным протоколом, который позволяет подключить к смартфону внешнее устройство без дополнительной конфигурации, остаётся USB OTG. Чтобы работа начиналась автоматически и не требовала разрешения со стороны Android, будем адаптировать идею BadUSB. Следовательно, остаётся определить, какое именно дополнительное оборудование, подключаемое по USB и способное реализовать класс USB HID, выбрать в качестве основы реализации.

Ввиду распространённости (в сравнении с макетными платами), дешевизны (в сравнении с Raspberry Pi) и наличия общепринятой реализации USB HID (в виде Bluetooth HID profile), будем использовать

[18]До внедрения USB Type-C такое соединение могло привести к возгоранию. Теперь же, согласно спецификации [14], питание просто не подаётся.

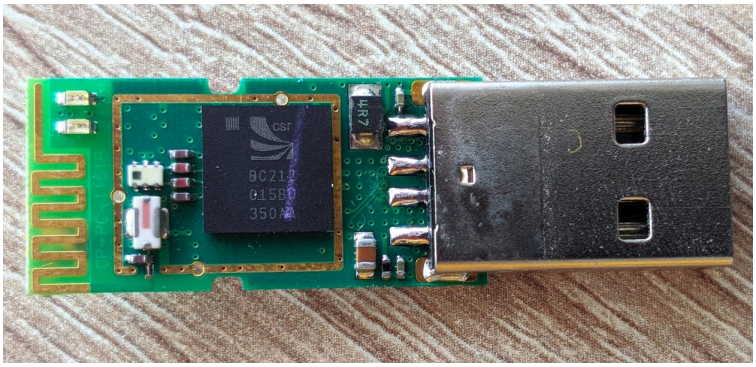

Рис. 2: Пример Bluetooth-приёмника с интерфейсом USB. Фото автора.

Bluetooth-приёмник. Для примера на рисунке 2 изображён D-Link DBT-120.

Спецификация [5] протокола Bluetooth вводит два логических компонента: «хост» и «контроллер», общение между которыми обеспечивается посредством Host Controller Interface (HCI). Контроллер реализуется аппаратно, он непосредственно управляет радио-оборудованием и отвечает за целостность передаваемых данных. Хост — программное обеспечение, которое реализует «верхние уровни» протокола, руководит подключениями и формирует данные для передачи. Описанная архитектура изображена на рисунке 3. Оба компонента могут быть как реализованы в одном чипе, так и разделены, во втором случае HCI может быть реализован поверх USB-соединения.

Подключаемый по USB Bluetooth-приёмник всегда содержит реализацию контроллера, а хост может быть представлен в одном из двух перечисленных ниже вариантов.

1. В виде Bluetooth-стека ОС, такого как BlueZ в Linux. Тогда ОС использует приёмник аналогично встроенной сетевой карте.

2. Реализован на отдельном чипе внутри приёмника. Тогда приёмник определяется ОС как USB устройство, класс которого зависит от реализованного на чипе профиля Bluetooth. Например, приёмник может быть определён как USB-клавиатура.

Многие Bluetooth-приёмники, в том числе тот, что изображён на рисунке 2, используют чип Cambridge Silicon Radio 8510, который реа-

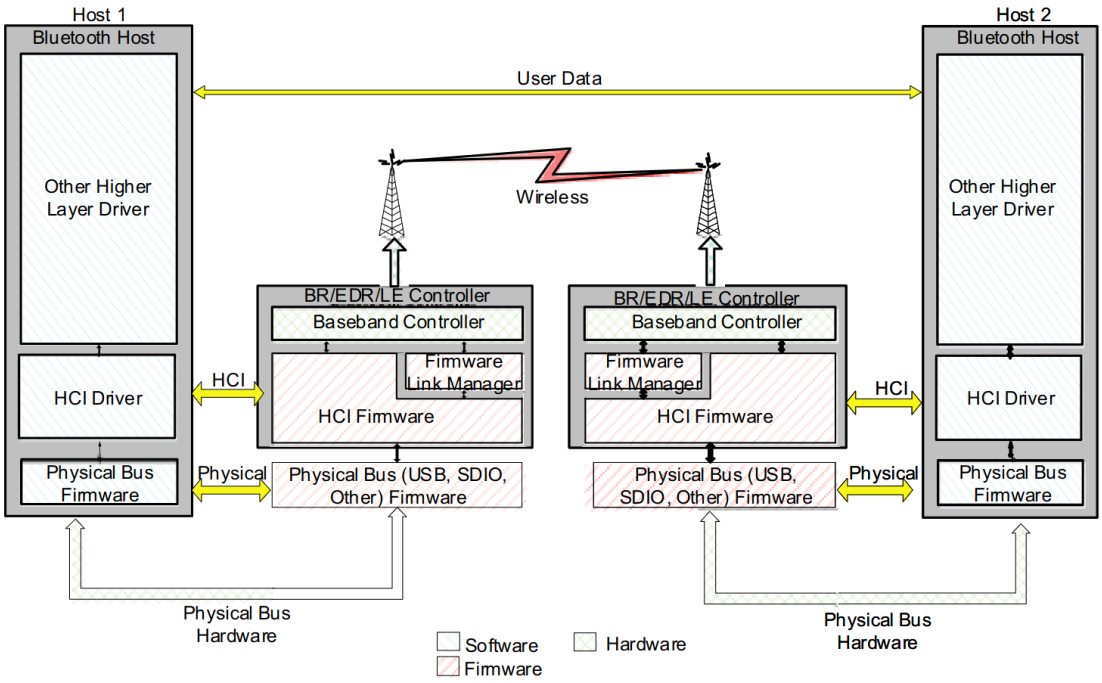

Рис. 3: Архитектура Bluetooth-устройств. Источник: [5].

лизует сразу оба варианта (далее будем называть их режимами HCI и HID соответственно) и может переключаться между ними при получении команды от ОС. Согласно сведениям о чипе [4], его также можно сконфигурировать таким образом, чтобы он автоматически запускался в выбранном режиме. Эту возможность мы и будем использовать для нашего решения.

Существует несколько проектов[19,20,21], в том числе коммерческих[22,23], позволяющих использовать ПК как Bluetooth-клавиатуру, однако все они требуют предварительного сопряжения со смартфоном, что не соответствует рассматриваемому сценарию использования.

[19]Проект hidclient: https://github.com/benizi/hidclient (дата обращения: 25.05.2023).

[20]Проект xkbdbthid: https://www.mulliner.org/bluetooth/xkbdbthid.php (дата обращения: 25.05.2023).

[21]Проект btkbdd: https://github.com/lkundrak/btkbdd (дата обращения: 25.05.2023).

[22]Приложение across: https://download.acrosscenter.com (дата обращения: 25.05.2023).

[23]Приложение Type2Phone: https://www.houdah.com/type2Phone/ (дата обращения: 25.05.2023).

3. Метод

Опишем предполагаемый сценарий использования приложения.

1. Пользователь подключает к ПК Bluetooth-приёмник и запускает приложение.

2. Приложение настраивает сопряжение между ПК и приёмником. Если приёмник находится в режиме HCI, приложение конфигурирует его так, чтобы в следующий раз он загружался в HID.

3. Пользователь подключает сконфигурированный Bluetooth-приёмник к смартфону. Приложение соединяется с приёмником.

4. Приложение подбирает пароль экранной блокировки и сохраняет его на ПК.

Для реализации этого сценария требуется следующее. Во-первых, использованный в приёмнике чип должен реализовывать режимы HCI и HID, а также обладать хранилищем, в которое можно записать ключ сопряжения с ПК. Во-вторых, на шаге 2 приложение должно сохранять MAC-адрес приёмника и соединяться с ним при обнаружении на шаге 3. В-третьих, Bluetooth-стек ОС должен позволить приложению передавать «сырые» байты (raw bytes), кодирующие сигналы клавиш.

3.1. Эксперименты с разными приёмниками

Были опробованы три приёмника, использующих чип Cambridge Silicon Radio 8510. Помимо изображенного на рисунке 2 D-Link DBT-120 (далее “D-Link”), это Buro BU-BT40A (далее “Buro”) и устройство неизвестного производителя (далее “NoName”), изображённые на рисунке 4. Можно отметить, что на приёмниках D-Link и Buro виден оригинальный чип Cambridge Silicon Radio 8510. В свою очередь хотя NoName и определяется на ПК как устройство CSR, однако на самом деле таким чипом не обладает и, по-видимому, является контрафактным.

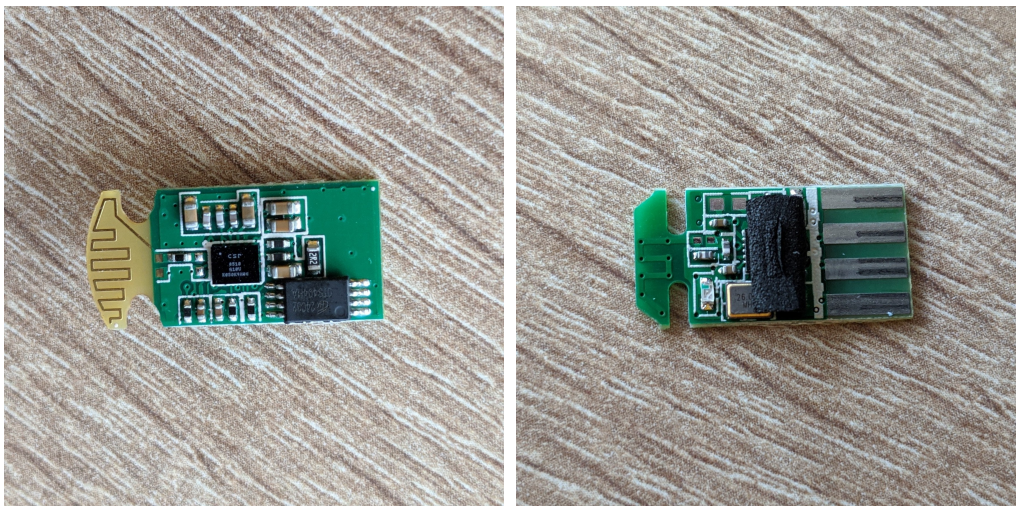

Рис. 4: Bluetooth-приёмники Buro (слева) и NoName (справа). Фото автора.

Утилита BlueSuite[24], поставляемая производителем чипа и позволяющая изменять конфигурацию приёмника, не смогла соединиться с NoName, что позволяет предположить, что контрафактный чип не идентичен оригинальному. Вследствие чего сменить режим его работы с HCI на HID либо затруднительно, либо невозможно.

Также упоминаются[25] другие чипы, потенциально пригодные для реализации. Однако устройства, использующие их, достаточно редки, а следовательно не удовлетворяют требованию универсальности.

3.2. Требования к аппаратному обеспечению

Таким образом, для работы проектируемого приложения требуются:

- ПК, оснащённый встроенным или внешним Bluetooth-модулем;
- подключаемый по USB Bluetooth-приёмник, использующий оригинальный чип Cambridge Silicon Radio 8510;
- кабель USB On-The-Go для подключения Bluetooth-приёмника к смартфону.

[24]Набор приложений Qualcomm (CSR) BlueSuite версий 2.5 и 2.6.6 для ОС Windows.

[25]Упоминание чипа Broadcom BCM2046: https://www.tonymacx86.com/threads/d-link-dbt-120-bluetooth-2-0-usb-adapter-works-otb.6330/post-255317 (дата обращения: 25.05.2023).

4. Архитектура

Автору не удалось получить документацию на программное обеспечение чипа Cambridge Silicon Radio 8510. Некоторая информация об имеющихся регистрах памяти и их назначении доступна в [6] и соответствует информации, отображаемой в BlueSuite, но попытка модифицировать их значения может привести к необратимому нарушению работоспособности Bluetooth-приёмника[26]. Кроме того, адреса и набор регистров, доступных для модификации утилитой bccmd[27] из пакета BlueZ, отличаются. Поэтому было принято решение на данный момент отказаться от автоматической настройки сопряжения между приёмником и ПК. Процедура ручного сопряжения устройств описана далее в разделе 5.3.

Спроектированное приложение состоит из четырёх основных компонентов, приведённых на диаграмме 5. Опишем их назначение.

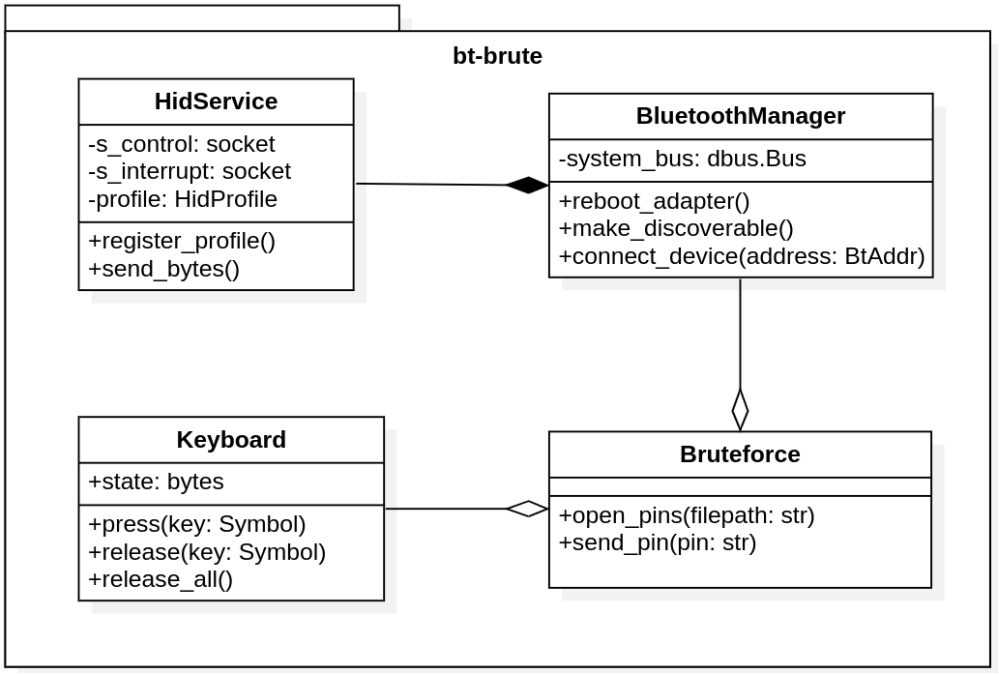

Рис. 5: Диаграмма основных классов приложения.

[26]Подробнее в разделе 6.2. См. также https://github.com/wmertens/textblade-dongler/issues (дата обращения: 25.05.2023).

[27]bccmd — неофициальный инструмент для работы с чипами CSR. Объявлен устаревшим и исключён из пакета, начиная с версии BlueZ 5.56.

- BluetoothManager управляет Bluetooth-модулем ПК посредством системной шины D-Bus. В том числе позволяет перезагрузить его, инициировать подключение к Bluetooth-приёмнику.

- HidService, включающий в себя объект HidProfile, реализует Bluetooth Service Discovery Protocol (SDP). Для этого публикуется SDP-запись, сходная с SDP-записями Bluetooth-клавиатур. Также HidService управляет сокетом типа AF_BLUETOOTH, ожидает подключения устройства (смартфона или Bluetooth-приёмника), позволяет отправить состояние клавиатуры на устройство.

- Keyboard представляет собой состояние виртуальной клавиатуры, то есть список идентификаторов одновременно нажатых клавиш (не более шести). Также конвертирует символ (такой как цифра или “enter”) в код клавиши в соответствии с десятой секцией [8].

- Bruteforce обрабатывает словари PIN, в том числе пользовательские. Также позволяет ввести PIN на устройстве, последовательно изменяя состояние Keyboard и используя HidService.

Пример совместной работы компонентов и справка по использованию приложения приведены на рисунке 6.

Из приведённого описания неясно, как приложение определяет, верен ли введённый PIN. Это неизбежное ограничение: в рамках выбранного подхода BadUSB принципиально отсутствует возможность определить корректность PIN, поскольку клавиатура не получает обратную связь от смартфона. Таким образом, в результате работы программы смартфон будет разблокирован ровно один раз, а чтобы узнать корректный пароль, требуется участие наблюдателя.

Опишем предлагаемый вариант подробнее. ОС Android защищена от перебора паролей экранной блокировки, поэтому через каждые несколько попыток приложение должно ожидать истечения «таймаута», прежде чем продолжать перебор. Число попыток между периодами ожидания и его длительность зависит от модели смартфона и версии ОС, их можно настроить в конфигурации приложения. Во время ожидания

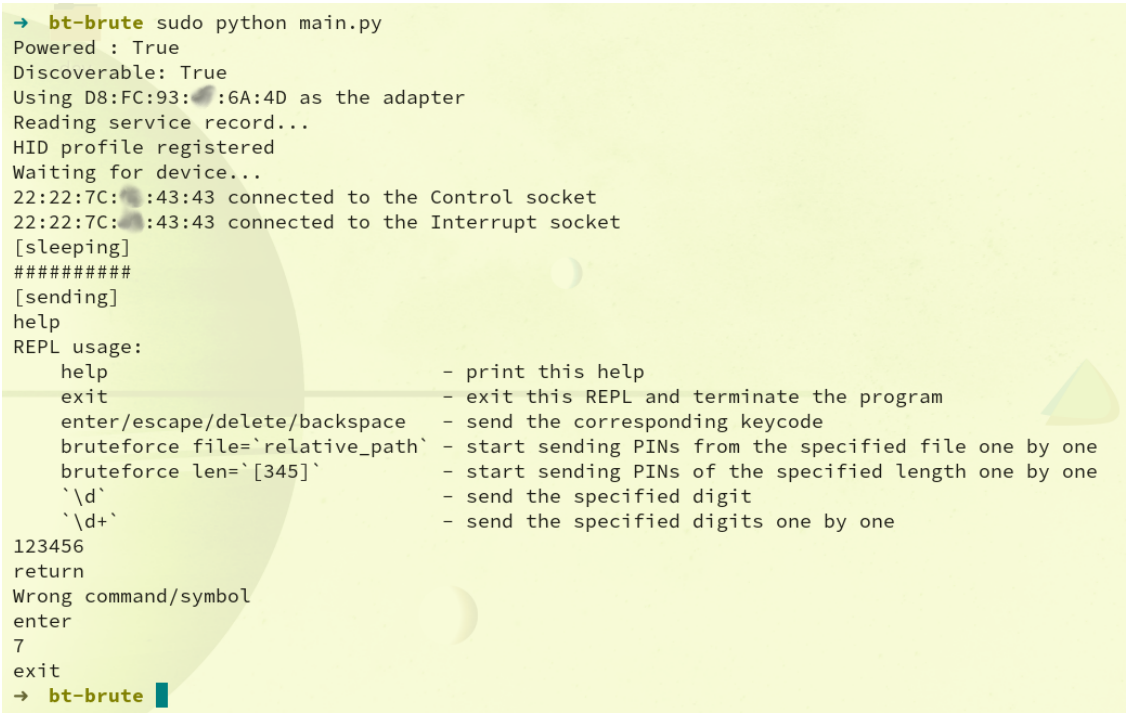

Рис. 6: Снимок экрана: функционал приложения и его консольный интерфейс в виде REPL.

приложение сохраняет для последнего введённого PIN его позицию в используемом файле и текущее время (назовём это контрольной точкой), а также отправляет звуковое уведомление. Таким образом, если ведётся видеозапись процесса или наблюдатель, привлечённый уведомлением, обнаружил смартфон разблокированным, время снятия блокировки можно соотнести с подходящей контрольной точкой, и тем самым выявить корректный PIN с точностью до нескольких (порядка пяти) попыток.

Второй вопрос — как гарантировать, что отправляемые символы достигают смартфона. Разберём несколько возможных проблем. Предположение первое: на смартфоне настроена другая раскладка клавиатуры, поэтому символы интерпретируются некорректно. Однако это не вызовет проблем в работе приложения, поскольку коды для цифр и специальных клавиш (таких как “enter”) не зависят от раскладки, а другие символы (буквы) не используются. Предположение второе: нарушено Bluetooth-соединение. В таком случае первая же попытка отправить данные на соответствующий сокет вызовет ошибку вида “Connection

reset by peer". После восстановления соединения можно продолжить работу с последней контрольной точки. Предположение третье: нарушено USB-соединение. Поскольку для работы USB-клавиатуры не требуется никакого специального ПО, подобная проблема может быть вызвана физическим отключением Bluetooth-приёмника от смартфона или выключением смартфона. В таком случае приёмник теряет питание, а значит Bluetooth-соединение также будет разорвано. Решение остаётся прежним.

В заключение отметим, что перебор PIN с учётом периодов ожидания — весьма длительный процесс. Так, для полного перебора всех четырёхзначных PIN на Samsung Galaxy S5 требуется почти 17 часов[28]. Для более длинных паролей рекомендуется использовать только словари, ограниченные наиболее вероятными комбинациями.

[28]Значение взято из документации к проекту Android-PIN-Bruteforce.

5. Особенности реализации

5.1. Коммуникация по шине D-Bus

D-Bus — шина межпроцессного общения в ОС Linux. «Процесс-демон» BlueZ, управляющий Bluetooth-модулем ПК, предоставляет свой интерфейс в виде объекта на этой шине, обладающего свойствами, некоторые из которых приведены на рисунке 7. Изменение значений этих свойств ведёт к изменению поведения модуля, в частности запись “True” в свойство “Discoverable” позволяет другим Bluetooth-устройствам обнаружить ПК. Разработанное приложение использует для этого библиотеку python-dbus, предоставляющую интерфейс на языке Python к стандартной реализации libdbus.

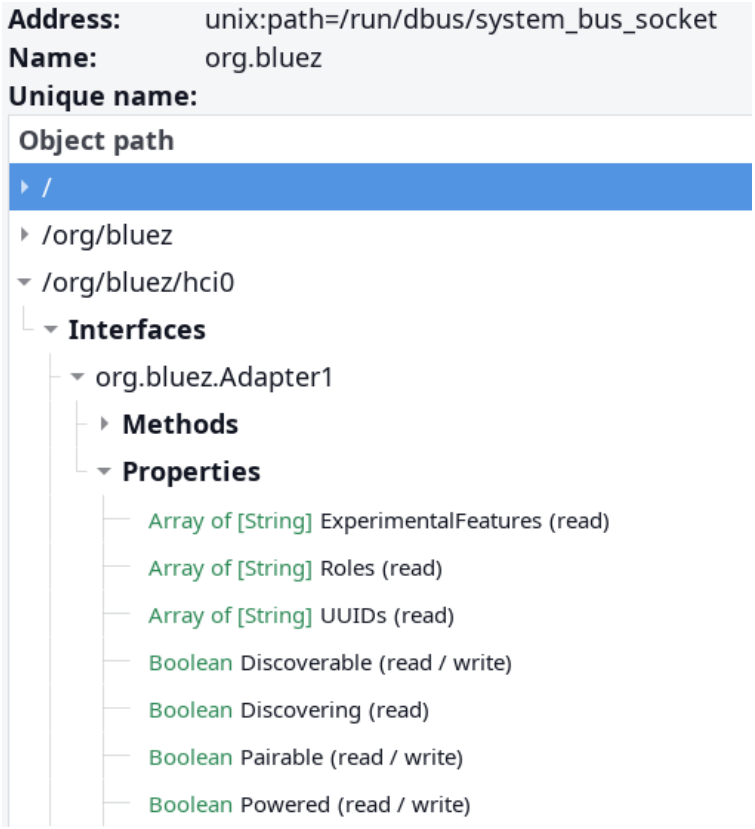

Рис. 7: Снимок экрана приложения D-Feet. Свойства объекта Bluetooth-модуля на шине D-Bus.

5.2. Смена режима работы Bluetooth-приёмника

В документе [4] производитель описывает механизм работы загрузочных режимов (HCI и HID, описанных выше) в чипе Cambridge Silicon Radio 8510. Выбор режима определяется значением в регистре PSKEY_INITIAL_BOOTMODE. Используя утилиту PSTool в составе BlueSuite[29], действительно можно изменить это значение. Вопреки ожиданиям, такая модификация привела к полной неработоспособности приёмника Buro.

Альтернативный подход был применён к приёмнику D-Link. С помощью утилиты DFUWizard из того же набора BlueSuite, его прошивка была заменена на GenericCSR.dfu, предоставляемую компанией Apple[30]. В данной прошивке нужное значение уже записано в указанный регистр. После процедуры приёмник был определён Диспетчером устройств Windows как USB-клавиатура.

Для того, чтобы временно (до переподключения) вернуть приёмник в режим HCI, была использована утилита hid2hci, после чего с ним снова можно было работать в утилите PSTool. Отметим, что хотя hid2hci также предусматривает обратную смену режима, ни один из предлагаемых ею методов не был завершён успешно.

5.3. Сопряжение с Bluetooth-приёмником

Было настроено сопряжение между приёмником D-Link, находящимся в режиме HCI, и Bluetooth-модулем ПК. Затем ключ сопряжения был извлечён из папки /var/lib/bluetooth (см. фрагмент содержимого соответствующего файла 8). Ключ и адрес Bluetooth-модуля были преобразованы в соответствии с правилом для регистра PSKEY_LINK_KEY_BD_ADDR0, описанном в [6], и записаны в память приёмника.

После перезагрузки в режим HID ключ сопряжения был принят. Теперь соединение между ПК и приёмником могло быть установлено без

[29]Набор приложений Qualcomm (CSR) BlueSuite версий 2.5 и 2.6.6 для ОС Windows.

[30]Bluetooth Firmware Updater от Apple: https://support.apple.com/kb/DL534 (дата обращения: 25.05.2023).

```
[General]
Name=BlueZ HID
...
[LinkKey]
Key=233B781BAF80215B5247EA18B116FF16
Type=0
PINLength=0
...
```

Рис. 8: Ключ сопряжения с приёмником D-Link.

дополнительных действий, то есть конфигурация приёмника окончена.

К сожалению, однако, данная процедура нарушила работу приёмника, что описано в следующей главе. Возможно, что сопряжение не вызвало бы проблем у другого экземпляра той же модели, однако получить его не удалось.

6. Тестирование

Поскольку приложение реализовано в виде прототипа (proof-of-concept), цель тестирования — удостовериться, что предложенный метод работоспособен и несёт практическую пользу. Сформулируем отдельные критерии этого:

1. ПК, управляемый приложением, опознаётся смартфоном как Bluetooth-клавиатура;
2. Bluetooth-приёмник опознаётся смартфоном как USB-клавиатура;
3. ПК подключается к Bluetooth-приёмнику;
4. приложение выполняет перебор PIN-кодов по словарю;
5. перебираемые PIN-коды вводятся на смартфон, к которому подключён Bluetooth-приёмник;
6. при вводе корректного PIN-кода смартфон разблокируется.

Основной сценарий тестирования соответствует сценарию, описанному в главе 3, с тем отличием, что сопряжение с Bluetooth-приёмником выполняется вручную. В сценарии проверяются все критерии.

Дополнительный сценарий позволяет проверить критерии 1, 4 и 6 и выглядит следующим образом. Запустить приложение; вручную разблокировать смартфон, включить Bluetooth и подтвердить сопряжение; обратно заблокировать смартфон и запустить процесс перебора. Дополнительный сценарий был разработан, чтобы протестировать часть приложения, не зависящую от конкретного аппаратного обеспечения Bluetooth-приёмника.

6.1. Условия тестирования

Для тестирования использовались:

- ноутбук с Bluetooth-модулем Intel Dual Band Wireless-AC 7260;

- ОС Linux (дистрибутив EndeavourOS) с BlueZ версии 5.66, Python версии 3.11.3;
- смартфон LG G710EAW с портом USB Type-C и ОС Android 13 (PixelExperience);
- смартфон LG D855 с портом micro-USB и ОС Android 6;
- Bluetooth-приёмник D-Link DBT-120, использующий чип Cambridge Silicon Radio 8510 с прошивкой GenericCSR.dfu;
- два работоспособных кабеля USB On-The-Go.

6.2. Результаты

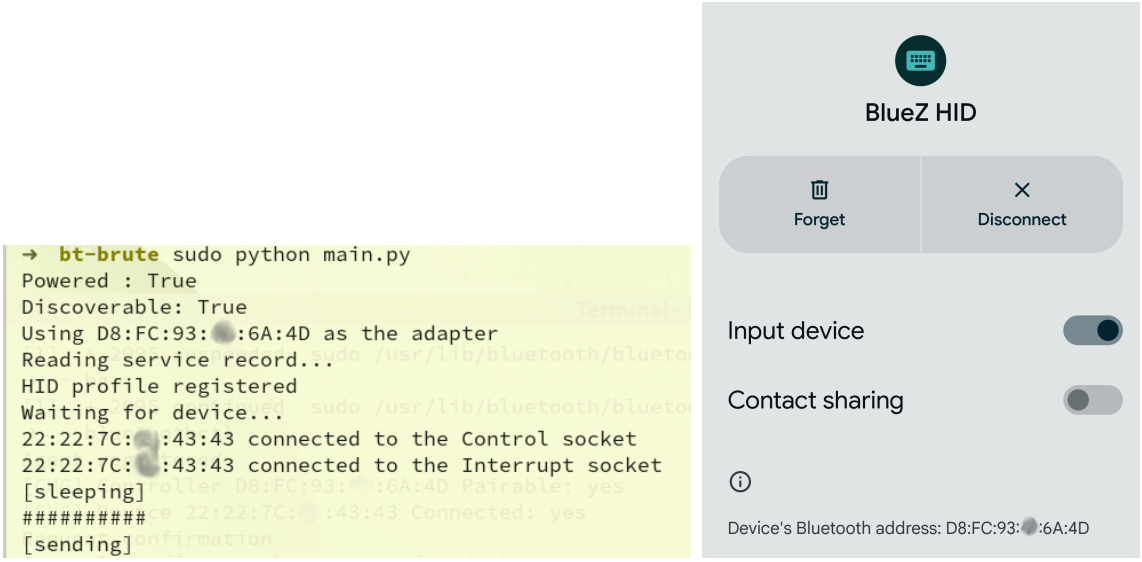

Рис. 9: Снимки экрана: работа приложения, подключение к смартфону.

Выполнение основного сценария было нарушено после сохранения ключа сопряжения в памяти Bluetooth-приёмника. Ключ был записан успешно, критерий 2 и 3 удовлетворены. Однако соединение с приёмником стало нестабильным, в ОС он опознаётся с заметной задержкой, вызывая зависание программы lsusb, а на вызов hid2hci не реагирует. Сбросить ключ сопряжения или заново загрузить в приёмник прошивку GenericCSR.dfu не удалось. Таким образом, критерий 5 удовлетворён не был, поскольку соединение обрывается при попытке передать данные.

Дополнительный сценарий был выполнен успешно. Иллюстрации работы приложения — подключение к смартфону в качестве Bluetooth-клавиатуры и ввод символов — приведены на рисунках 9 и 10.

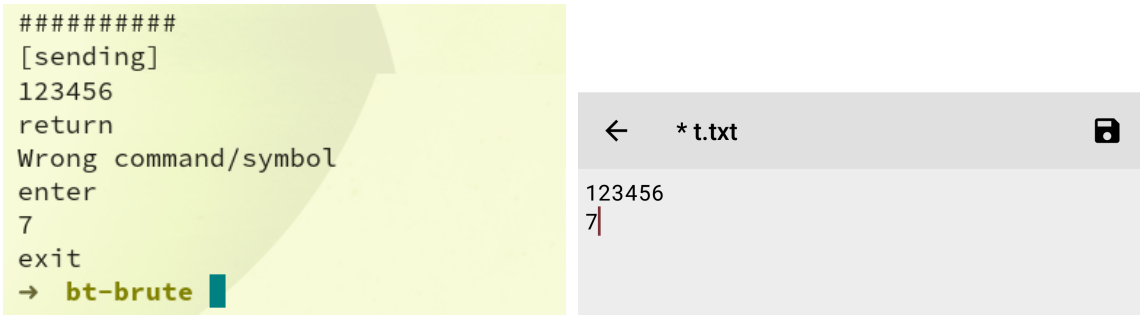

Рис. 10: Снимки экрана: работа приложения, ввод символов.

Два других приёмника, NoName и Buro, не могли быть использованы для тестирования. Первый, как было описано в разделе 3.1, не удалось перевести в режим HID. Второй после соответствующей настройки перестал обнаруживаться любым ПО как USB-устройство, то есть полностью потерял работоспособность.

Результаты тестирования на обоих смартфонах идентичны. Это объясняется тем, что протестировать работу с приёмником через USB не удалось, а от Bluetooth-стека требовались лишь базовые возможности, реализуемые каждым устройством.

Заключение

В ходе выполнения работы были достигнуты нижеследующие результаты.

1. Проведён обзор четырёх технологий для эмуляции клавиатуры. Доказано, что поставленную цель невозможно достичь без использования дополнительного оборудования. В качестве основы реализации выбран стек Bluetooth.

2. Сформулированы требования к аппаратному обеспечению ПК и Bluetooth-приёмника.

3. Спроектировано консольное приложение, позволяющее автоматически отправлять на смартфон сигналы нажатия клавиш и перебирать пароли экранной блокировки по словарю. Приложение использует установленное Bluetooth-соединение с приёмником, подключаемым к смартфону через USB On-The-Go. Используются язык Python, стек BlueZ для установки соединения и передачи данных, D-Bus для коммуникации с BlueZ.

4. В соответствии с проектом реализовано приложение, разработка велась в Intellij IDEA Community Edition. Предусмотрено дальнейшее расширение приложения для сохранения состояния при отключении смартфона.

5. Проведено тестирование на двух смартфонах (с портом micro-USB и с USB Type-C) с оптимизированными словарями. Основной сценарий не выполнен ввиду проблем с аппаратным обеспечением. В дополнительном сценарии корректные пароли были подобраны в обоих случаях.

Код реализованного приложения закрыт.

Список литературы

[1] Android Debug Bridge (adb) // Android Developers. — 2023. — URL: https://developer.android.com/studio/command-line/adb (дата обращения: 25.05.2023).

[2] Axelson Jan. USB complete: the developer's guide. — Lakeview research LLC, 2015.

[3] Bier Agnieszka, Kapczyński Adrian, Sroczyński Zdzisław. Pattern Lock Evaluation Framework for Mobile Devices: Human Perception of the Pattern Strength Measure // Man-Machine Interactions 5 / Ed. by Aleksandra Gruca, Tadeusz Czachórski, Katarzyna Harezlak et al. — Cham : Springer International Publishing, 2018. — P. 33–42.

[4] BlueCore. — Understanding and Using Bootmodes Application Note. — Cambridge Silicon Radio, 2003. — URL: https://web.archive.org/web/20150701021108/http://www.ie.ksu.edu.tw/data/bluetooth/28/docs/bcore-an-019P.pdf (дата обращения: 25.05.2023).

[5] Bluetooth Core Specification : Rep. / Bluetooth SIG ; Executor: Bluetooth Special Interest Group : 2023. — URL: https://www.bluetooth.com/specifications/specs/core-specification-5-4/ (дата обращения: 25.05.2023).

[6] Datalogic Scanning Inc. — DLBTM Bluetooth Radio Module Users Guide, 2009.

[7] Does the layout of the Android unlock pattern affect the security and usability of the password? / Lei Zhang, Yajun Guo, Xiaowei Guo, Xiaowei Shao // Journal of Information Security and Applications. — 2021. — Vol. 62. — P. 103011.

[8] HID Usage Tables for Universal Serial Bus (USB) : Rep. / USB Implementers Forum ; Executor: USB Implementers Fo-

rum : 2023. — URL: https://www.usb.org/document-library/hid-usage-tables-14 (дата обращения: 25.05.2023).

[9] Nohl Karsten, Krißler Sascha, Lell Jakob. BadUSB // Security Research Labs. — 2014. — URL: https://www.srlabs.de/blog-post/usb-peripherals-turn (дата обращения: 25.05.2023).

[10] Practical mobile forensics / Heather Mahalik, Rohit Tamma, Satish Bommisetty, Oleg Skulkin. — Packt Publishing Ltd, 2016.

[11] Storage // Android Open Source Project. — 2023. — URL: https://source.android.com/docs/core/storage (дата обращения: 25.05.2023).

[12] Texas Instruments. — TPS65981, TPS65982, and TPS65986 Firmware User's Guide. — Texas Instruments, 2016.

[13] On-The-Go and Embedded Host Supplement to the USB Revision 3.0 Specification : Rep. / USB Implementers Forum ; Executor: USB Implementers Forum : 2012. — URL: https://www.usb.org/sites/default/files/documents/usb_otg_and_eh_3-0_release_1_1_10may2012.pdf (дата обращения: 25.05.2023).

[14] Universal Serial Bus Type-C Cable and Connector Specification : Rep. / USB Implementers Forum ; Executor: USB Implementers Forum : 2022. — URL: https://www.usb.org/document-library/usb-type-cr-cable-and-connector-specification-release-22 (дата обращения: 25.05.2023).

[15] Universal Serial Bus Power Delivery Specification : Rep. / USB Implementers Forum ; Executor: USB Implementers Forum : 2023. — URL: https://www.usb.org/document-library/usb-power-delivery (дата обращения: 25.05.2023).