

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт
Факультет приборостроения,
информационных технологий и
электроники

Кафедра
«Информационная безопасность
систем и технологий»

Специальность 100503 –

Информационная безопасность
автоматизированных систем

Специализация –

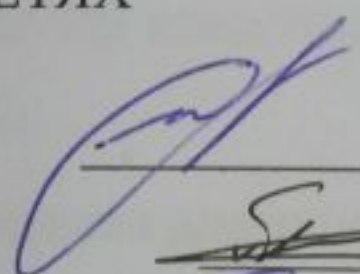
Защищенные автоматизированные
системы управления

ДИПЛОМНЫЙ ПРОЕКТ

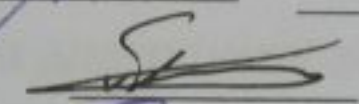
на тему

АППАРАТНО-ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ ИССЛЕДОВАНИЯ
СПОСОБОВ СКРЫТОЙ ПЕРЕДАЧИ ИНФОРМАЦИИ В КОМПЬЮТЕРНЫХ
СЕТЯХ

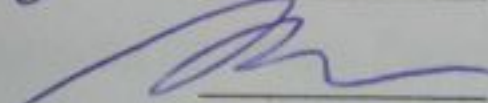
Студент

 Исаев Роман Сергеевич

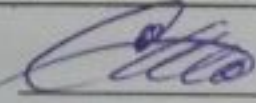
Руководитель

 Печилейский С. В.

Нормконтроллер

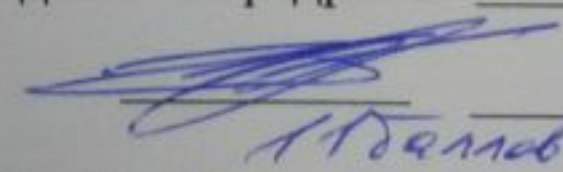
 Иванов А. П.

Рецензент заместитель ген. директора по спецвопросам АО «ПНИЭИ»

 Сомкин С. А.

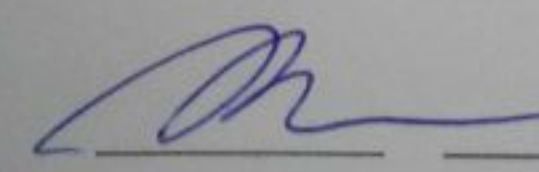
Работа допущена к защите (протокол заседания кафедры от 20.06/17 № 14)

Заведующий кафедрой

 Зефиоров С. Л.

Работа защищена с отметкой отлично (протокол заседания ГЭК
от 23.06/17 № 10)

Секретарь ГЭК

 Иванов А. П.

Пенза 2017

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Политехнический институт
Факультет приборостроения,
информационных технологий и
электроники

Кафедра
«Информационная безопасность
систем и технологий»

Специальность 100503 –

Информационная безопасность
автоматизированных систем

Специализация –

Защищенные автоматизированные
системы управления

ДИПЛОМНЫЙ ПРОЕКТ

на тему

АППАРАТНО-ПРОГРАММНЫЙ КОМПЛЕКС ДЛЯ ИССЛЕДОВАНИЯ
СПОСОБОВ СКРЫТОЙ ПЕРЕДАЧИ ИНФОРМАЦИИ В КОМПЬЮТЕРНЫХ
СЕТЯХ

Студент _____ Исаев Роман Сергеевич

Руководитель _____ Печилейский С. В.

Нормконтроллер _____ Иванов А. П.

Рецензент заместитель ген. директора по спецвопросам АО «ПНИЭИ»
_____ Сомкин С. А.

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой _____ Зефиоров С. Л.

Работа защищена с отметкой _____ (протокол заседания ГЭК
от _____ № _____)

Секретарь ГЭК _____ Иванов А. П.

Пенза 2017

РЕФЕРАТ

Пояснительная записка содержит 83 с., 43 рис., 1 табл., 12 источников, 4 прил.

СТЕГАНОГРАФИЯ, СКРЫТЫЙ КАНАЛ ПЕРЕДАЧИ, ПРОТОКОЛ МЕЖСЕТЕВОГО ВЗАИМОДЕЙСТВИЯ, ЭТАЛОННАЯ МОДЕЛЬ МЕЖСЕТЕВОГО ВЗАИМОДЕЙСТВИЯ.

Объектом исследования являются протоколы межсетевого взаимодействия и их недокументированные возможности.

Предметом исследования являются скрытые каналы передачи информации и возможность их реализации посредством стандартных протоколов межсетевого взаимодействия.

Целью дипломного проекта является исследование способов скрытой передачи информации с использованием стандартных протоколов межсетевого взаимодействия и разработка аппаратно-программного комплекса, предназначенного для демонстрации и изучения способов скрытой передачи информации.

В процессе работы проведен обзор известных способов скрытой передачи информации и их эффективности.

В процессе проектирования созданы диаграммы вариантов использования, диаграммы развертывания, диаграммы пакетов. Разработана программная часть для выбранной аппаратной части комплекса.

В результате дипломного проектирования был разработан аппаратно-программный комплекс для исследования способов скрытой передачи информации в компьютерных сетях.

СОДЕРЖАНИЕ

Введение	6
1 Теоретический раздел	8
1.1 Проблема скрытых каналов передачи информации	8
1.2 Эталонная модель взаимодействия открытых систем (OSI) . .	9
1.3 Набор сетевых протоколов TCP/IP	11
1.4 Классификация способов скрытой передачи информации . . .	13
1.5 Способы организации скрытых каналов передачи информации путем модификации пакетов	15
1.6 Гибридные способы организации скрытых каналов передачи информации	17
1.7 Емкость и эффективность применения скрытых каналов передачи	19
2 Практический раздел	21
2.1 Разработка требований к аппаратно-программному комплексу	21
2.2 Аппаратное обеспечение	22
2.2.1 Выбор компонентов аппаратной части комплекса . . .	23
2.2.2 Разработка аппаратного модуля управления	27
2.3 Программное обеспечение	31
2.3.1 Определение программных компонентов комплекса . .	31
2.3.2 Выбор функционала программной части	32
2.3.3 Разработка программной части, реализующей скрытые каналы передачи информации	38
2.3.4 Разработка программной части для аппаратного модуля управления	41
2.3.5 Проверка разработанной программной части на рабо- тоспособность	45
2.4 Методика по работе с аппаратно-программным комплексом для исследования способов скрытой передачи информации в компьютерных сетях	46

2.4.1 Состав аппаратно-программного комплекса для исследования способов скрытой передачи информации	46
2.4.2 Состав узлов, участвующих в передаче информации по скрытому каналу	47
2.4.3 Подключение компьютера для перехвата и анализа трафика	49
2.4.4 Выполнение скрытой передачи информации с помощью протокола ICMP	50
2.4.5 Выполнение скрытой передачи информации с помощью протокола TCP	52
2.4.6 Выполнение скрытой передачи информации с помощью протокола UDP (способ №1)	53
2.4.7 Выполнение скрытой передачи информации с помощью протокола UDP (способ №2)	55
2.4.8 Выполнение приема скрыто передаваемой информации с помощью протокола ICMP	57
2.4.9 Выполнение приема скрыто передаваемой информации с помощью протокола TCP	59
2.4.10 Выполнение приема скрыто передаваемой информации с помощью протокола UDP (способ №1)	61
2.4.11 Выполнение приема скрыто передаваемой информации с помощью протокола UDP (способ №2)	63
Заключение	66
Список использованных источников	67
Приложение А Исходный код программной части комплекса для выполнения скрытой передачи информации с помощью протокола ICMP	68
Приложение Б Исходный код программной части комплекса для выполнения скрытой передачи информации с помощью протокола TCP	73

Приложение В Исходный код программной части комплекса для выполнения скрытой передачи информации с помощью протокола UDP (способ №1)	76
Приложение Г Исходный код программной части комплекса для выполнения скрытой передачи информации с помощью протокола UDP (способ №2)	79

ВВЕДЕНИЕ

В настоящее время телекоммуникационные системы неразрывно связаны с ЭВМ. Связь осуществляется с использованием стандартных протоколов межсетевого взаимодействия. Однако, эти протоколы имеют особенности, позволяющие реализовывать различного рода угрозы информационной безопасности.

Угроза утечки информации – одна из угроз, которая может быть реализована путем использования стандартных протоколов нестандартным образом, давая при этом потенциальную возможность организации скрытого канала передачи информации.

Угроза скрытого управления – угроза, позволяющая скрыто передавать управляющие команды, например, на зараженные вредоносными программами компьютеры. Таким образом, по команде злоумышленника может быть осуществлено любое негативное воздействие, например, уничтожение или кража информации.

Разумеется, все эти угрозы являются актуальными потому, что с помощью реализующих их методов можно обойти такие средства защиты, как межсетевой экран или системы предотвращения утечек информации, осуществляющие анализ трафика.

Целью дипломного проекта является исследование способов скрытой передачи информации с использованием стандартных протоколов межсетевого взаимодействия и разработка аппаратно-программного комплекса, предназначенного для демонстрации и изучения способов скрытой передачи информации.

В теоретическом разделе приводится определение стандартных протоколов межсетевого взаимодействия сетевого и транспортного уровней модели TCP/IP. Рассматриваются особенности данных протоколов и каким образом можно организовать с их помощью скрытые каналы передачи информации. Также рассматривается классификация и способы организации скрытых каналов передачи информации.

В практическом разделе описываются требования к разрабатываемому аппаратно-программному комплексу. Рассматривается разработка аппаратно-программного комплекса. Рассматриваются процесс выбора аппаратной части комплекса, процесс разработки программной части комплекса. Приводятся результаты проверки корректности работы реализованных скрытых каналов передачи информации, а также методика по работе с разработанным аппаратно-программным комплексом для исследования способов скрытой передачи информации.

1 Теоретический раздел

1.1 Проблема скрытых каналов передачи информации

Задача надежной защиты информации является одной из древнейших и нерешенных до сегодняшнего дня. Существует несколько наук, занимающихся решением данной проблемы – криптография и стеганография. Основные задачи криптографии заключаются в разработке методов преобразования (шифрования) информации с целью ее защиты от незаконных пользователей [1]. Задача стеганографии, напротив, заключается в том, чтобы посторонние вообще не узнали о факте передачи информации.

Пока криптография защищает сообщения от несанкционированного перехвата, разработкой средств и методов скрытия факта передачи сообщения занимается стеганография. С помощью стеганографии можно прятать и предварительно зашифрованные тексты. Поэтому стеганография потенциально обеспечивает не только информационную безопасность, но и анонимность и приватность, которая становится все более востребованной в современном обществе.

С одной стороны, стеганография предназначена для защиты от несанкционированного доступа к информации, но в то же время, например, применительно к компьютерным сетям (далее – сетевая стеганография), сам фактор скрытности актуализирует различного рода угрозы. Одна из таких угроз – угроза утечки информации из различных информационных систем. Так как в настоящее время степень компьютеризации в организациях стремительно увеличивается, для обеспечения информационной безопасности этих систем разрабатываются различные руководящие документы (например, приказы ФСБ, методики ФСТЭК), и в соответствии с ними разрабатываются программные и аппаратно-программные средства защиты информации. Однако нигде утечка информации с применением сетевой стеганографии, как таковая, не рассматривается. Соответственно, разрабатываемые средства защиты информации (СЗИ) не обеспечивают защиту от данной угрозы.

Вторая угроза связана с угрозой скрытого управления вредоносными программами, внедренными в технические средства информационных систем. К вредоносным программам относится такое ПО, как компьютерные вирусы, эксплойты, трояны и т.д. Трояны – разновидность вредоносных программ, проникающих в компьютер под видом полезного программного обеспечения. Трояны осуществляют различные несанкционированные действия на компьютере: сбор, хищение информации, выполнение деструктивных действий на компьютере. Однако, ввиду того, что троянские программы часто состоят из клиентской и серверной частей, взаимодействие между ними может осуществляться по скрытым каналам.

Третья угроза связана с угрозой обхода различного рода средств защиты, таких как межсетевые экраны или системы предотвращения утечек информации, выполняющих анализ трафика. Данная угроза является самой опасной, так как она актуализирует, две вышеназванные угрозы. На момент написания работы стандартные средства защиты, как межсетевые экраны, средства обнаружения вторжений или системы предотвращения утечек информации не предоставляют средств защиты от утечки информации по скрытым каналам передачи. Также, ввиду того, что имеется множество различных способов реализации скрытых каналов передачи, задача по созданию универсального средства защиты от утечки информации по скрытым каналам передачи представляется трудно выполнимой.

1.2 Эталонная модель взаимодействия открытых систем (OSI)

Изучение любой предметной области начинается с изучения относящихся к ней фундаментальных вещей. В области компьютерных сетей к фундаментальным знаниям относится эталонная модель взаимодействия открытых систем (ЭМВОС). ЭМВОС – концептуальная модель, являющаяся стандартом взаимодействия телекоммуникационных систем. Модель OSI содержит 7 уровней и определяет, какие функции возлагаются на каждый уровень. Появление такой структуры было обусловлено следующими соображениями [2]:

- уровень должен создаваться по мере необходимости отдельного уровня абстракции;
- каждый уровень должен выполнять строго определенную функцию;
- выбор функций для каждого уровня должен осуществляться с учетом международных стандартов;
- границы между уровнями должны быть такими, чтобы поток данных между интерфейсами был минимальным;
- число уровней должно быть максимально сбалансированным, чтобы отдельные функции не объединялись в конкретном уровне, архитектура модели не было громоздкой.

В данную модель входят следующие уровни:

- физический уровень – занимается конкретной передачей необработанных битов по каналу связи. Принципиальными вопросами здесь выступают уровень напряжения бита, длительность импульса, направление передачи (симплекс, полудуплекс, дуплекс), способ синхронизации аппаратуры, начала и конца передачи, нахождение и исправление ошибок при передаче;
- канальный уровень (уровень передачи данных) предназначен для передачи первичных данных, разбиваемых на кадры. Кадры передаются последовательно с обработкой кадров подтверждения;
- сетевой уровень предназначен для управления операциями подсети. Основной задачей данного уровня является определение маршрутов пересылки пакетов от источника к получателю. Маршруты могут задаваться либо статически, либо динамически с помощью специализированных протоколов. Также задачей сетевого уровня является недопущение так называемого "затопления канала", то есть присутствия в сети большого количества одновременно присутствующих в сети пакетов;
- транспортный уровень предназначен для разбиения данных от сеансового уровня на небольшие фрагменты и их передачи сетевому уровню с гарантией корректной доставки этих фрагментов по назначению. Транспортный

уровень также определяет тип сервиса, предоставляемого сеансовому уровню и, в конечном счете, пользователям сети;

- сеансовый уровень позволяет пользователям различных компьютеров устанавливать сеансы связи друг с другом. При этом предоставляются различные типа сервисов, среди которых управление диалогом, синхронизация и так далее;

- уровень представления отвечает за преобразование протоколов и кодирование/декодирование данных. Запросы приложения с прикладного уровня здесь преобразуются в формат, подходящий для передачи по сети, а полученные данные преобразует обратным образом;

- прикладной уровень содержит набор популярных протоколов, необходимых пользователям. К протоколам прикладного уровня относятся такие протоколы, которые используются в наиболее популярных приложениях, например, протокол передачи гипертекста HTTP (HyperText Transfer Protocol), протокол защищенного удаленного доступа SSH (Secure Shell) и тому подобное.

1.3 Набор сетевых протоколов TCP/IP

Для того, чтобы понять, каким образом функционируют скрытые каналы передачи, важно понимать архитектуру сетевых протоколов. В области компьютерных сетей, протоколы межсетевого взаимодействия являются основой при передаче данных; передача данных через сеть требует применения специальных протоколов, разработанных для облегчения общения между узлами сети.

Набор интернет протоколов включает в себя набор протоколов межсетевого взаимодействия, которые служат основой для работы всей сети. Следует отметить, что методы сетевой стеганографии преимущественно ориентированы на манипуляции с данными протоколами. Понятие "набор протоколов", обычно ассоциируется с набором протоколов TCP/IP. Данный набор имеет четырехуровневую архитектуру для разделения протоколов по их задачам, поэто-

му его также называют "стек протоколов TCP/IP". Набор протоколов включает в себя следующие уровни [3]:

- канальный уровень – первый, базовый уровень. Включает в себя протоколы, используемые для определения топологии локальной сети. Протоколы канального уровня описаны в стандартах IEEE 802 подгруппы 802.3 являющихся основой семейства технологий пакетной передачи данных Ethernet;
- сетевой уровень – второй уровень стека. Включает в себя протоколы, необходимые для эффективной передачи данных между сетями. Примеры протоколов сетевого уровня: IPv4 и IPv6;
- транспортный уровень – третий уровень стека. Включает в себя протоколы, необходимые для открытия и поддержки соединений между определенными приложениями или процессами. Основными представителями являются протоколы TCP и UDP, используемые для передачи данных между конкретными узлами сети;
- прикладной уровень – четвертый уровень стека. К этому уровню относятся протоколы, необходимые для организации соединения между процессами на разных узлах. К протоколам прикладного уровня относятся протоколы HTTP, SSH, FTP и т.д.

Пакет готов для передачи тогда, когда все данные были соответствующим образом инкапсулированы в соответствии с уровнями, как показано на рисунке 1.

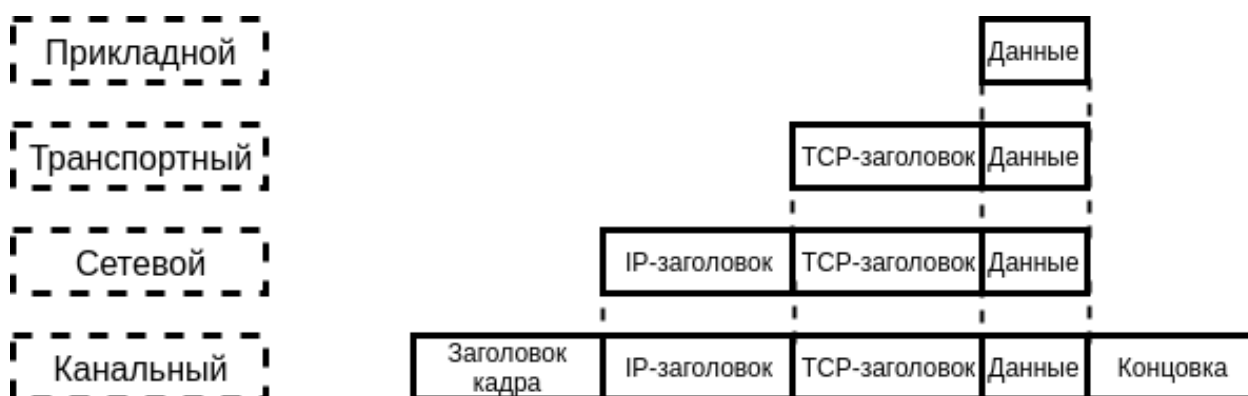


Рисунок 1 — Представление данных в соответствии с набором протоколов TCP/IP

Следует заметить, что законченный пакет не требует наличия всех четырех уровней, например, существует много типов пакетов, в которых не используется прикладной уровень. Аналогичным образом методы сетевой стеганографии подразумевают различные способы модификации пакетов, но обязательно пакеты должны содержать все 4 уровня.

Также многие из способов организации скрытых каналов передачи требуют использования протокола IPv4. IPv4 (Internet Protocol v.4) – адресный протокол сетевого уровня. IPv4 использует 32-битный адрес, состоящий из четырех десятичных чисел или октетов, разделенных точками (например, 192.168.1.1). Каждый октет IP-адреса определен в интервале $[0; 255]$, который переводится в 8 бит или 1 байт. Так как IPv4 имеет только 32 бита, теоретически максимально возможное количество адресов ограничено значением 2^{32} . В настоящее время испытывается дефицит IPv4-адресов, по этой причине началось постепенное внедрение протокола IPv6. Данный протокол призван решить проблему дефицита IP-адресов путем увеличения его размера. В протоколе IPv6 на один адрес отводится 128 бит, поэтому максимальное теоретическое количество адресов равно 2^{128} . Таким образом, существующие способы организации скрытых каналов передачи на базе протокола IPv4 требуют модификации для работы с протоколом IPv6.

1.4 Классификация способов скрытой передачи информации

Способы реализации сетевой стеганографии можно разделить на три группы [4]:

- способы, в которых для передачи применяются поля заголовков сетевых протоколов и поля полезной нагрузки пакетов;
- способы, в которых изменяется структура передачи пакетов, например, изменение очередности передачи пакетов или введение потерь пакетов при передаче;
- смешанные способы, использующие в одновременно способы из двух предыдущих групп.

В свою очередь, каждая из этих групп делится еще на несколько подгрупп, например, группа модификации полей пакетов включают в себя следующие методы:

- использование для скрытой передачи информации полей заголовков протоколов (TCP, UDP, ICMP и т.д.);
- изменение полезной нагрузки пакета. В данном случае возможно использование других методов компьютерной стеганографии, в зависимости от представления передаваемых данных;
- использование смешанных способов;

Способы модификации структуры передачи пакетов включают в себя следующие направления:

- способы, в которых изменяется порядок передачи пакетов;
- способы, в которых вводится задержка между передаваемыми пакетами;
- способы, в которых вводится преднамеренная потеря пакетов при отправлении.

Подробная схема классификации способов реализации скрытых каналов передачи с использованием сетевой стеганографии приведена на рисунке 2.

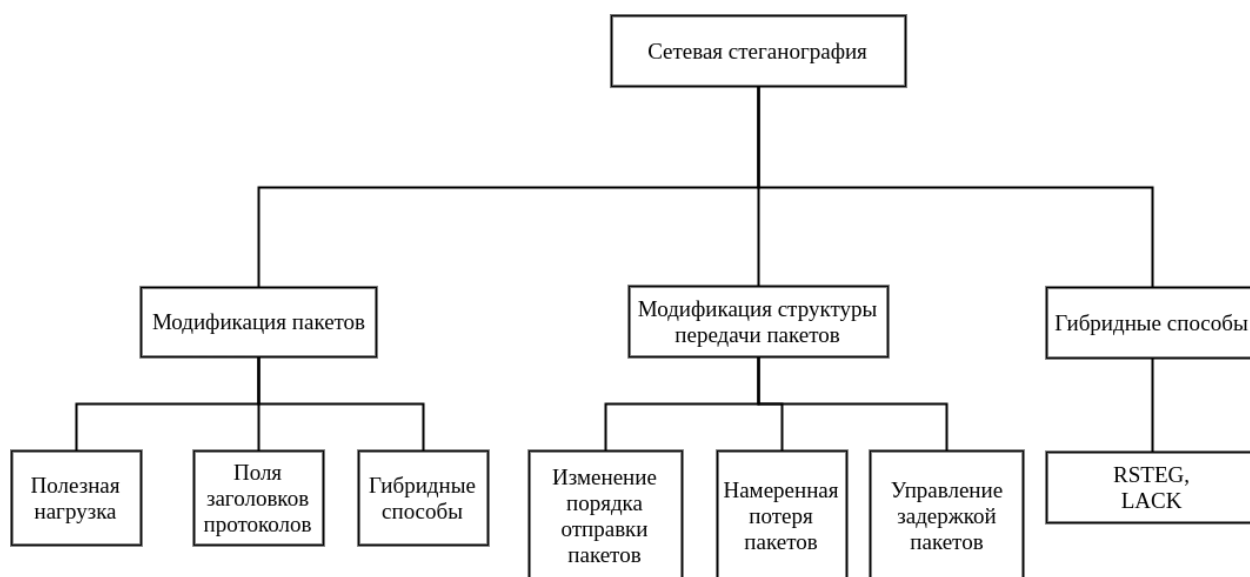


Рисунок 2 — Классификация способов реализации скрытых каналов передачи

Особенность методов скрытия информации в сетевых пакетах является предопределенность мест, где размещается скрываемая информация. Дополнительная защита может достигаться путем применения криптографии.

1.5 Способы организации скрытых каналов передачи информации путем модификации пакетов

Основная мысль при применении способов модификации пакетов заключается в использовании некоторых полей заголовков при внесении в них стеганограммы. Это возможно благодаря особенностям полей заголовков, которые не задействуются при передаче данных или изменяются определенным образом, не нарушая правил построения пакетов.

На рисунке 3 приведен формат заголовка пакетов протокола TCP. Данный протокол является одним из основных протоколов передачи данных в сети интернет. Механизм TCP предлагает функционал, позволяющий осуществлять передачу с гарантированной целостностью путем выполнения повторных запросов данных в случае их потери или устранения дублирования при получении двух копий одного пакета. Данный протокол работает на транспортном уровне стека TCP/IP и применяется при передаче информации, требовательной к целостности, например, данный протокол желателен при использовании протоколов прикладного уровня HTTP или HTTPS, с помощью которых может передаваться ценная информация, например, аутентифицирующая информация или информация о банковских операциях.

Выделенные на рисунке поля потенциально могут использоваться для размещения в них стеганограмм. Поле "Порт отправителя" имеет размер 2 байта, а поле "Порядковый номер" – 4 байта. Передача с помощью данных полей возможна один раз при установлении соединения. Таким образом, за один сеанс возможно передать 6 байт информации.

Октет (байт)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
[0-3]	Порт отправителя															Порт получателя																
[4-7]	Порядковый номер (SN)																															
[8-11]	Номер подтверждения (ACK SN)																															
[12-15]	Длина Заголовка		Зарезервировано						Флаги					Размер окна																		
[16-19]	Контрольная сумма															Указатель важности																
[20-23]	Опции (необязательное, но используется практически всегда)																															
[24-...]	Данные																															

Рисунок 3 — Поля заголовка протокола TCP, подходящие для размещения стеганограмм

На рисунке 4 приведен формат заголовка пакета UDP. Данный протокол является одним из ключевых протоколов стека TCP/IP. Ввиду своей простоты используется при передаче потоковых видео и аудиоданных, в сетевых компьютерных играх и т.д. UDP подразумевает, что проверка целостности либо ненужна, либо должна проверяться в самом приложении. Чувствительные ко времени приложения часто используют UDP, так как предпочтительнее отбросить пакеты, чем ждать задержавшиеся пакеты, что может быть невозможным в системах реального времени. При необходимости исправления ошибок можно заменить UDP на TCP.

Выделенные на рисунке поля потенциально могут использоваться для размещения в них стеганограмм. В протоколе имеется поле, которое хранит размер передаваемой полезной нагрузки (длина датаграммы). Таким образом, изменяя размер полезной нагрузки, также возможно использовать данное поле при скрытой передаче.

Октет (байт)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
[0-3]	Порт отправителя															Порт получателя																
[4-7]	Длина датаграммы															Контрольная сумма																
[8-...]	Данные																															

Рисунок 4 — Поля заголовка протокола UDP, подходящие для размещения стеганограмм

На рисунке 5 приведен формат заголовка пакета ICMP. Особенность данного протокола в том, что он не предназначен для передачи данных. В основном ICMP используется для передачи сообщений об ошибках и других исключительных ситуациях, возникших при передаче данных, например, запрашиваемая услуга недоступна, или хост, или маршрутизатор не отвечают. Также

на ICMP возлагаются некоторые сервисные функции. Также, при условии использования в сети протокола IPv6, протокол ICMPv6 является обязательным и без него корректное функционирование сети невозможно. По этим причинам реализация скрытого канала передачи на базе протокола ICMP является более интересной.

Октет (байт)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
[0-3]	Тип							Код								Контрольная сумма																
[4-7]	Идентификатор															Номер последовательности																
...	Данные																															

Рисунок 5 — Поля заголовка протокола ICMP, подходящие для размещения стеганограмм

1.6 Гибридные способы организации скрытых каналов передачи информации

Идея гибридных способов заключается в том, чтобы использовать совместно способы из группы "Модификация пакетов" и "Модификация структуры передачи пакетов". Основными представителями этой группы являются методы RSTEG и LACK.

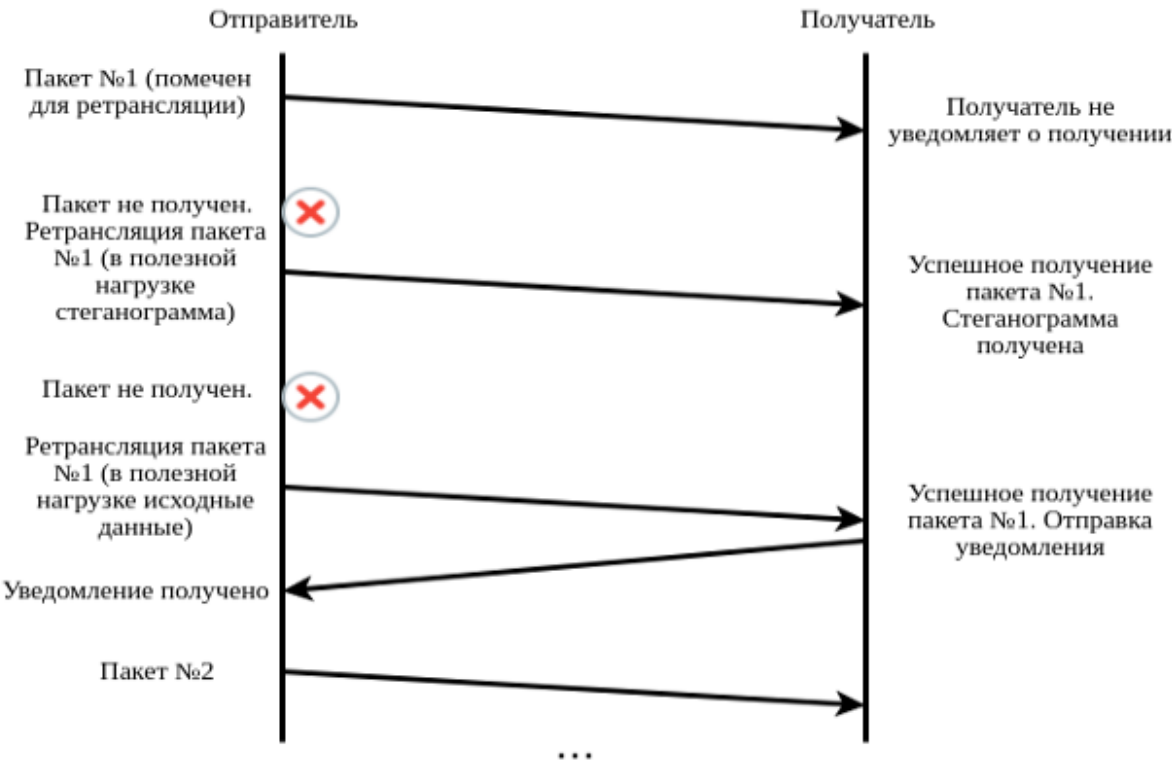


Рисунок 6 — Принцип функционирования метода RSTEG

Метод RSTEG (Retransmission Steganography) реализуется путем имитации потери пакета и его повторной отправкой с внедренной в него стеганогаммой. На рисунке 6 приведена схема, описывающая принцип функционирования метода RSTEG.

Отправитель, пересылая пакет, ждет подтверждения о доставке от получателя. В случае, если ответ не получен, пакет рассматривается отправителем как помеченный для ретрансляции и размещает в нем стеганогамму. После ретрансляции пакета, получатель также сообщает о его получении. Далее, при повторной ретрансляции выполняется отправка исходного пакета и при его получении получатель отправляет пакет с подтверждением о получении. Подобным образом выполняется дальнейшая передача.

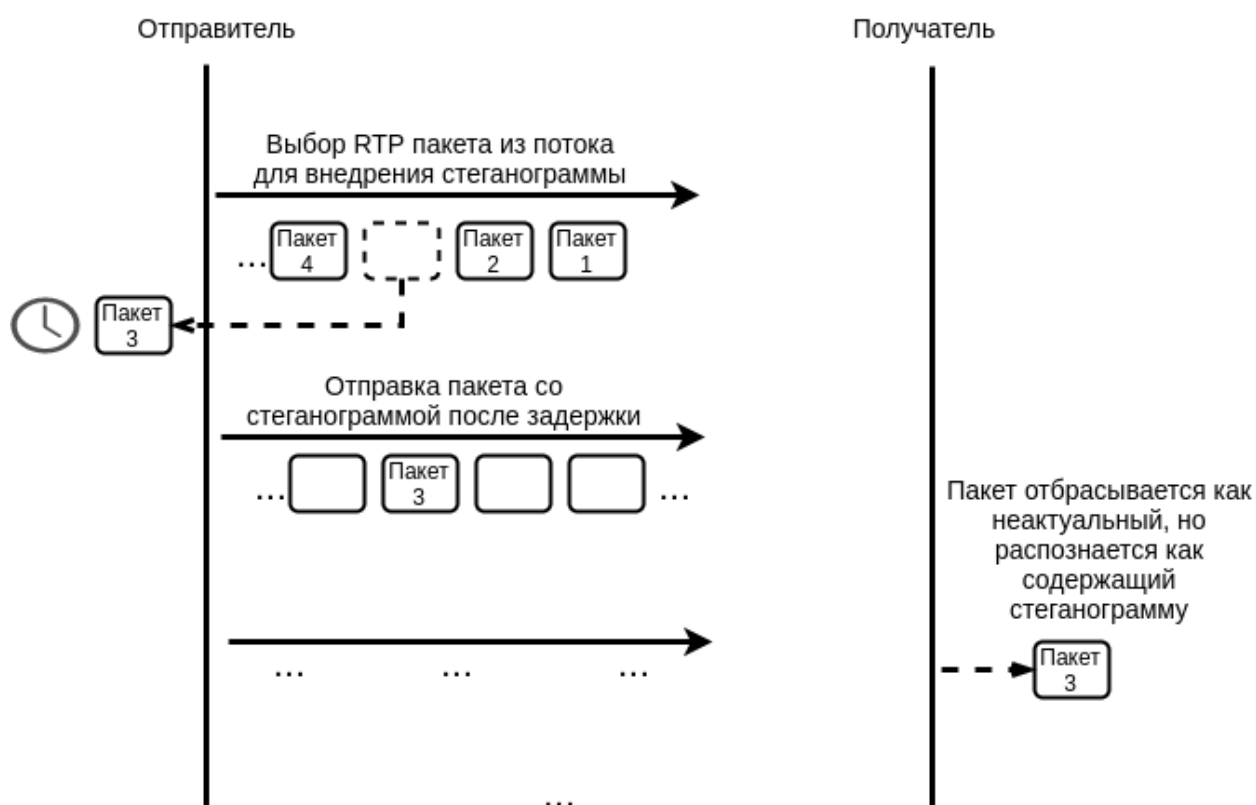


Рисунок 7 — Принцип функционирования метода LACK

Метод LACK (Lost Audio Packet Steganography) осуществляется путем преднамеренных задержек аудио пакетов. Реализуется при организации связи через IP-телефонию. Метод используется в связке с протоколами SIP и RTP. С помощью протокола SIP (Session Initiation Protocol) осуществляется установление и завершение пользовательского интернет-сеанса. С помощью протокола

RTP (Real-time Transport Protocol) осуществляется передача пакетов с аудиоданными в дуплексном режиме. На рисунке 7 изображен принцип функционирования метода LACK.

Скрытая передача с использованием метода LACK осуществляется следующим образом: из потока RTP-пакетов выбирается пакет в который записывается стеганограмма. Отправка данного пакета откладывается до тех пор, пока получатель не перестанет ожидать его получения. При получении пакета, получатель рассматривает его как неактуальный и в принимаемый поток не включает, но определяет его как пакет, содержащий стеганограмму.

1.7 Емкость и эффективность применения скрытых каналов передачи

Емкость или пропускная способность каналов изменяется в зависимости от способа их реализации. В каналах, реализованных путем модификации пакетов стандартных протоколов, пропускная способность определяется количеством бит или байт изменений, произведенных с пакетом. В каналах, реализованных путем модификации структуры передачи пакетов, пропускная способность ограничена объемом данных передаваемых через сеть.

В первых версиях стандарта "Критерии определения безопасности компьютерных систем" (так называемая "Оранжевая книга"), министерство обороны США заявляло, что с точки зрения безопасности, скрытые каналы с малой пропускной способностью представляют меньшую угрозу, нежели каналы с большой пропускной способностью [5]. При оценке эффективности скрытого канала передачи кажется, что более приоритетным критерием является пропускная способность канала, однако, большинство систем обнаружения вторжений и утечек, развернутых в сети, могут быть нацелены на выявление подозрительного трафика в сети. Если, например, злоумышленник решит использовать скрытый канал передачи с необычно большой скоростью передачи или пропускной способностью, существует риск быть обнаруженным какими-либо средствами мониторинга или привлечь внимание администраторов сети. В случае большой пропускной способности или скорости передачи по каналу, тем не менее, наиболее эффективной является передача данных на более медленных

и менее заметных скоростях. Несмотря на то, что данные по каналу в целом передаются медленно, решающим фактором здесь является его незаметность, следовательно, в перспективе канал будет наиболее долговечным и эффективным.

Таким образом, независимо от назначения скрытого канала, разработчики должны понимать, что скрытая передача данных на высоких скоростях скорее всего, в большинстве случаев не приведет к успеху.

2 Практический раздел

2.1 Разработка требований к аппаратно-программному комплексу

Для разработки аппаратно-программного комплекса как средства для исследования и изучения способов скрытой передачи информации было принято решение реализовать скрытые каналы передачи информации, основанные на модификации пакетов протоколов TCP, UDP и ICMP.

Для имитации скрытой передачи данных на передающей стороне должны осуществляться следующие действия:

- выбор способа организации скрытого канала, по которому будет осуществляться передача. Передачу осуществлять путем модификации пакетов протоколов TCP, UDP и ICMP. Выбор способа передачи осуществлять с помощью аппаратного модуля управления;
- выбор режима сжатия передаваемых данных. Сжатие осуществлять с помощью утилиты gzip. Включение и выключение режима выполнять с помощью кнопки "Сжатие" на аппаратном модуле управления;
- передачу выполнять после нажатия кнопки "Пуск" на аппаратном модуле управления, при этом подавать сигнал о статусе передачи.

Для имитации скрытой передачи данных на принимающей стороне должны осуществляться следующие действия:

- выбор способа организации скрытого канала, по которому будет осуществляться прием. Прием осуществлять путем извлечения данных из модифицированных пакетов протоколов TCP, UDP и ICMP. Выбор способа приема осуществлять с помощью аппаратного модуля управления;
- выбор режима сжатия принимаемых данных. Восстановление передаваемых данных осуществлять с помощью утилиты gzip. Включение и выключение режима выполнять с помощью кнопки "Сжатие" на аппаратном модуле управления;
- прием выполнять после нажатия кнопки "Пуск" на аппаратном модуле управления, при этом подавать сигнал о статусе передачи.

Для перехвата передаваемого трафика с целью анализа на промежуточном маршрутизаторе должны осуществляться следующие действия:

- маршрутизатор должен назначать передающему и принимающему узлам статические IP-адреса;
- весь трафик, генерируемый принимающей и передающей стороной зеркалировать на порт, к которому подключена наблюдающая сторона.

2.2 Аппаратное обеспечение

В подразделе 2.1 приведены требования к аппаратно-программному комплексу для исследования способов скрытой передачи информации в компьютерных сетях. Согласно приведенным требованиям, была составлена концептуальная структура аппаратно-программного комплекса для исследования способов скрытой передачи информации, представленная на рисунке 8.



Рисунок 8 — Структура аппаратно-программного комплекса для исследования способов скрытой передачи информации в компьютерных сетях

Далее, согласно приведенным требованиям была составлена диаграмма вариантов использования для разрабатываемого аппаратно-программного комплекса. Данная диаграмма приведена на рисунке 9.

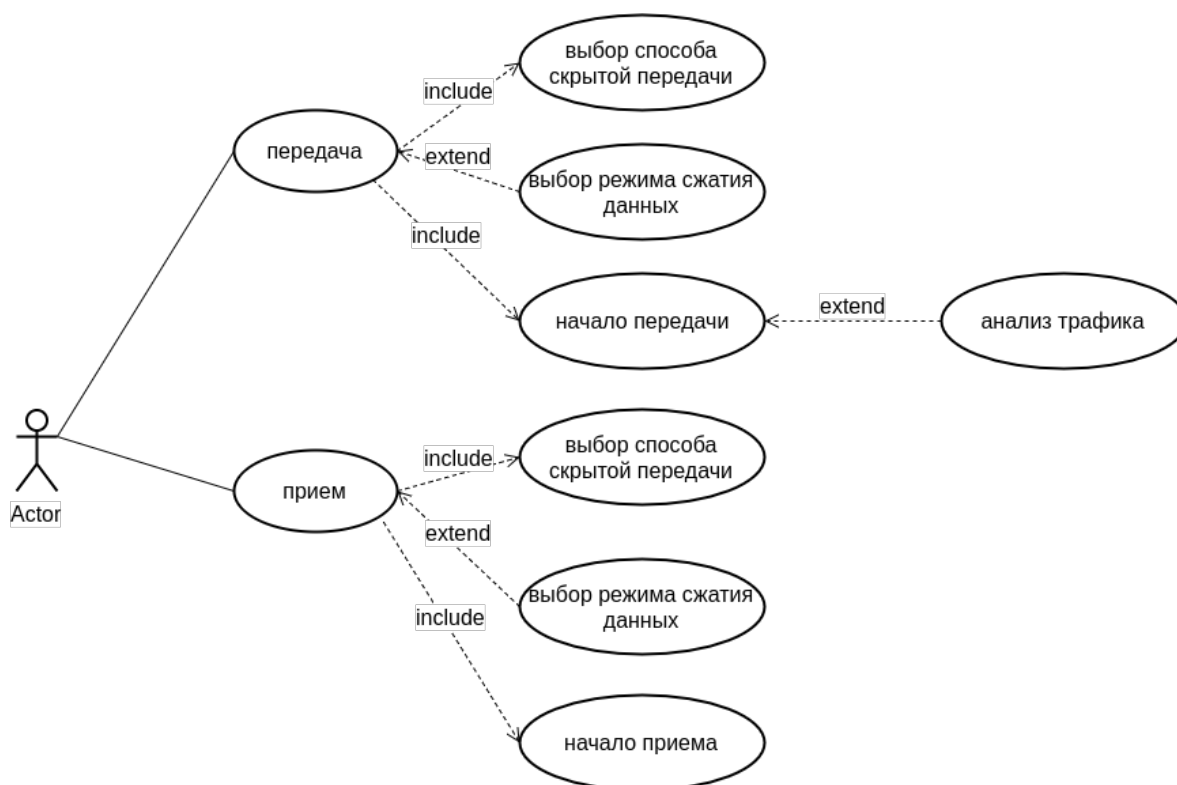


Рисунок 9 — Диаграмма вариантов использования для разрабатываемого аппаратно-программного комплекса

2.2.1 Выбор компонентов аппаратной части комплекса

Аппаратную часть разрабатываемого аппаратно-программного комплекса возможно реализовать несколькими способами:

- с использованием технологий виртуализации;
- с использованием реальных аппаратных компонентов.

При использовании виртуальных технологий подразумевается применение так называемых виртуальных машин, разворачиваемых и конфигурированных между собой специальным образом. На данный момент наиболее популярными инструментами виртуализации являются программные продукты VMWare и Virtualbox, являющиеся лидерами в своей отрасли. Данные решения позволяют развернуть в виртуальной среде требуемые операционные системы и настроить между ними необходимое сетевое взаимодействие. В плане экономической эффективности использование средств виртуализации является наиболее эффективным, так как не требует дополнительных аппаратных компонентов, однако при использовании данного способа реализации процесс обучения предстает в искусственном виде.

При выборе способа реализации с использованием реальных аппаратных компонентов, подходящих для реализации аппаратно-программного комплекса, имеется целый спектр подходящих устройств в зависимости от их назначения, начиная с персональных настольных компьютеров, и, заканчивая портативными одноплатными компьютерами малой мощности. Использование настольных компьютеров благодаря большой предоставляет максимальные возможности для реализации, но для изучения компьютерных сетей мощность настольных компьютеров будет избыточной. С точки зрения использования данного варианта в качестве сетевого устройства, настольные компьютеры будут иметь наибольшую функциональность и расширяемость, однако с точки зрения экономической выгоды, данное решение будет наименее эффективным.

Одноплатные компьютеры, с точки зрения экономической выгоды и приближенности к реальным условиям, являются наиболее оптимальным решением. Одноплатные компьютеры представляют из себя единую плату, на которой расположены центральный процессор, графический процессор, модули ОЗУ и т.д. По сравнению с настольными компьютерами являются менее мощными, но более энергоэффективными устройствами. По своему функционалу они практически не отличаются от более громоздких представителей, и имеют возможность установки и запуска на них операционных систем и программных средств, аналогичных тем, которые запускаются на настольных компьютерах. Главными представителями современных одноплатных компьютеров являются Raspberry Pi, Banana Pi, Orange Pi.

В качестве сетевого оборудования по техническому заданию требуется использовать компьютер с установленной на него сетевой операционной системой, например, OpenWRT или pfSense. pfSense – операционная система, основанная на системе FreeBSD, предназначенная для работы на сетевом оборудовании. Может быть установлена на обычный персональный компьютер.

OpenWRT – сетевая операционная система, основанная на ядре Linux, и предназначенная для установки на маломощных (домашних) маршрутизаторах. Поддерживает такие аппаратные платформы, как ARM, MIPS, x86, x86-64 и т.д.

Таким образом, согласно техническому заданию, в качестве готовых компонентов аппаратной части аппаратно-программного комплекта для исследования способов скрытой передачи информации выбраны следующие готовые решения: Raspberry Pi 3 Model B, маршрутизатор D-Link DIR-320nru.

Одноплатный компьютер Raspberry Pi 3 Model B – третье поколение одноплатных компьютеров семейства Raspberry Pi. Имеет размер, схожий с размером банковской карты. Изначально разрабатывался как обучающая платформа, однако позднее получил намного более широкое применение, начиная от медиацентров, разработанных на базе Raspberry Pi, и заканчивая портативными комплексами. Благодаря используемой аппаратной платформе, основанной на архитектуре ARM, имеет заметно низкое энергопотребление. Данная модель имеет следующие характеристики [6]:

- 4-ядерный ARMv8 процессор с частотой 1.2 ГГц;
- 802.11n Wireless LAN;
- Bluetooth 4.1;
- 1 Гб ОЗУ;
- 40-контактный GPIO-порт;
- 100-Mbit Ethernet;
- Full HDMI-порт;
- MicroSD-слот.

Преимуществом Raspberry PI 3 является наличие 40-контактного GPIO-порта, позволяющего подключать к компьютеру различные периферийные модули. Схема данного порта приведена на рисунке 10. По своей сути, эти контакты являются физическим интерфейсом между компьютером и внешним миром. На простейшем уровне их можно рассматривать как переключатели, которые можно включать или выключать. Из 40 контактов 26 – контакты GPIO. Остальные контакты выполняют функции питания или земли.

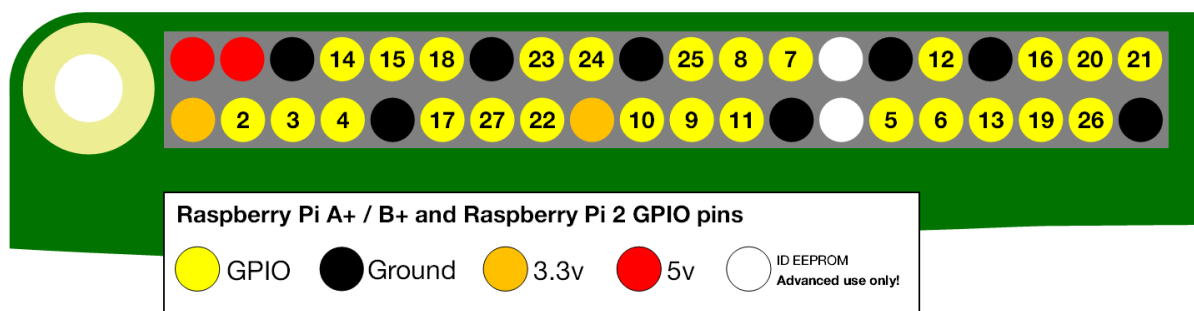


Рисунок 10 — Схема расположения контактов GPIO на одноплатном компьютере Raspberry Pi 3

В качестве устройства, выступающего в качестве маршрутизатора выбран маршрутизатор D-Link DIR320nru. Данное устройство выбрано по причине возможности установки на него операционной системы OpenWRT, требуемой по техническому заданию.

Данный маршрутизатор имеет следующие характеристики:

- интерфейс WAN 10/100BASE-TX для подключения DSL-модемов или выделенной линии;
- 4 порта 10/100BASE-TX Ethernet;
- WLAN интерфейс, поддерживающий стандарты IEEE 802.11b/g/n;
- USB 2.0 порт для подключения USB-модема или flash-накопителя;
- процессор MIPS 24K RT5350F 360 Mhz;
- 32 МБ ОЗУ;
- 8 МБ ПЗУ.

Ввиду того, что программное обеспечение данного маршрутизатора обладает весьма ограниченным функционалом и по техническому заданию требуется установка специально операционной системы, была осуществлена его прошивка под ОС OpenWRT. OpenWRT – гибкий дистрибутив GNU/Linux для встраиваемых систем. Вместо того, чтобы создавать статическое ПО под каждое устройство, OpenWRT предоставляет полноценную файловую систему с менеджером пакетов. Это дает возможность самостоятельно выбирать и настраивать функционал маршрутизатора, исключая, например, ненужный функционал, который предоставляет производитель. Данная операционная система

предоставляет следующий функционал, необходимый для функционирования разрабатываемого аппаратно-программного комплекса для исследования способов скрытой передачи информации:

- зеркалирование трафика с помощью утилиты iptables;
- перехват трафика с помощью утилиты tcpdump;
- удаленный доступ к маршрутизатору с помощью ssh;
- назначение статических IP-адресов конечным узлам аппаратно-программного комплекса для исследования способов скрытой передачи информации;
- расширение ПЗУ с помощью внешнего flash-накопителя.

Таким образом, выбрав готовые решения для аппаратно-программного комплекса для исследования способов скрытой передачи информации, также для демонстрационных целей требовалось разработать аппаратный модуль управления, позволяющий выполнять следующие манипуляции с передающим узлом:

- выбор способа скрытой передачи информации;
- включение/выключение режима сжатия передаваемых данных;
- начало передачи с индикация статуса передачи.

Для принимающего узла аппаратный модуль управления должен позволять выполнять следующие манипуляции:

- выбор способа скрытой передачи информации по которому будет осуществляться прием;
- включение/выключение режима сжатия принимаемых данных;
- начало передачи с индикация статуса приема.

2.2.2 Разработка аппаратного модуля управления

По причине наличия в компьютерной АЕ GPIO-порта, для демонстрационных целей был разработан аппаратный модуль управления, позволяющий выполнять передачу путем выбора способа передачи с помощью данного модуля. Схема соединения аппаратного модуля управления с портом GPIO приведена на рисунке 11.

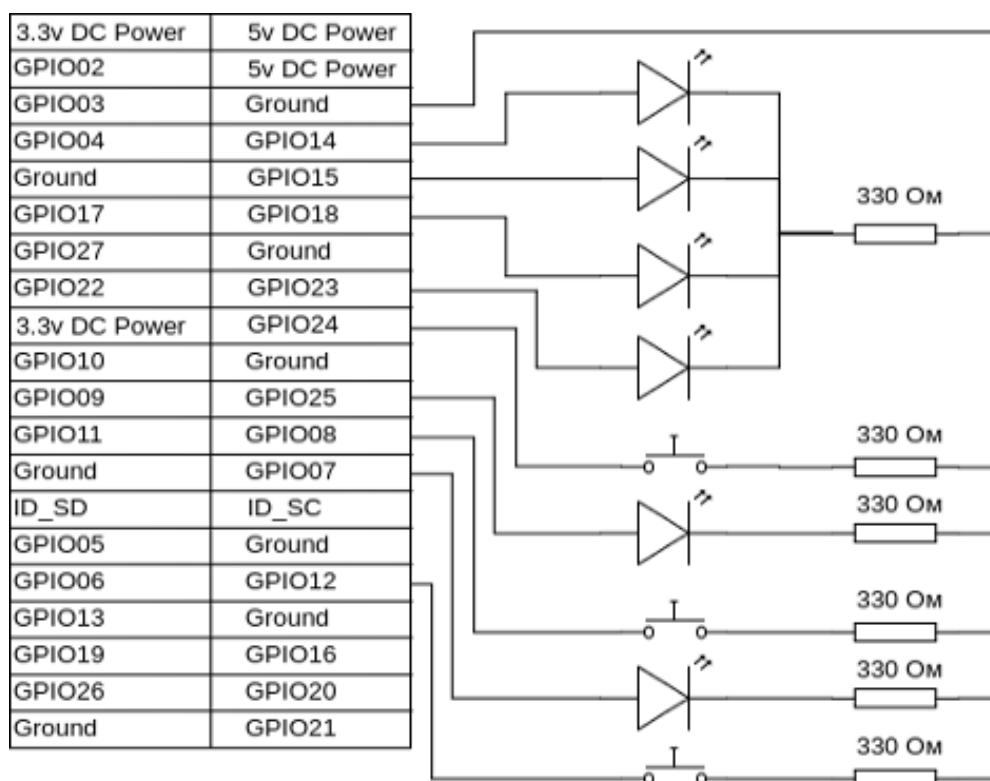


Рисунок 11 — Схема соединения аппаратного модуля управления с портом GPIO

Было собрано 2 аппаратных модуля управления согласно принципиальной схеме, приведенной на рисунке 12:

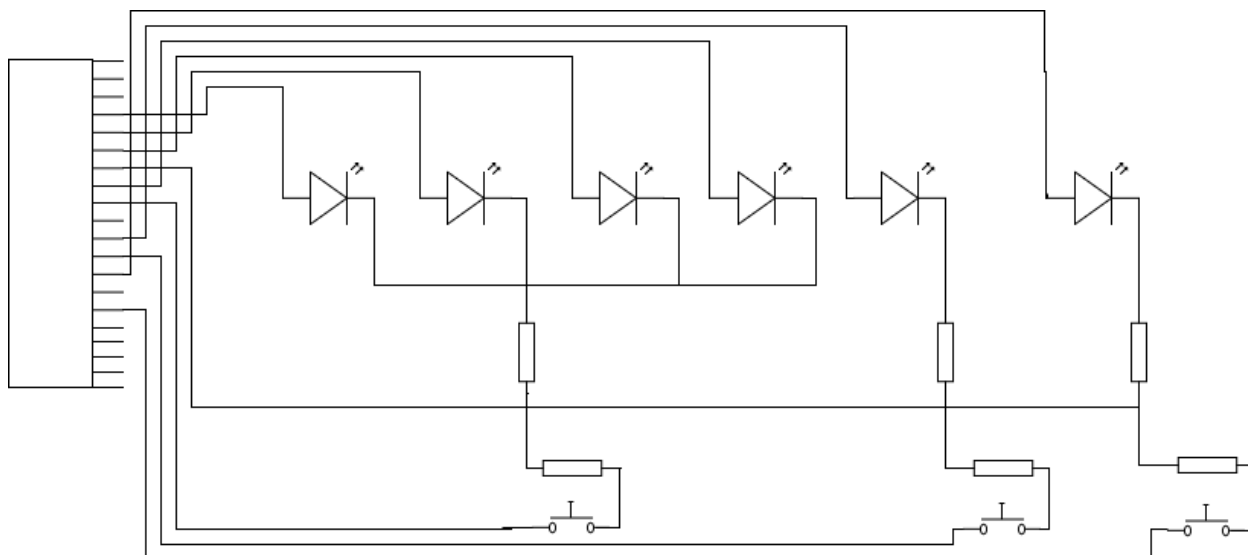


Рисунок 12 — Принципиальная схема аппаратного модуля управления

В ходе разработки схемы модуля требовалось рассчитать сопротивление резистора для светодиодов и кнопок. На аппаратном модуле управления передающего узла имеется 6 светодиодов и 3 кнопки, поэтому всего требуется

9 резисторов, но так имеется 4 светодиода, из которых в любой момент времени питание подается только на один светодиод, то можно ограничиться одним резистором на эти 4 светодиода.

Согласно рекомендациям по работе с Raspberry Pi 3, максимальное значение силы тока, проходящего через порт GPIO составляет 400 мА. На каждый контакт порта GPIO компьютера Raspberry Pi 3 рекомендованное значение силы тока составляет 50 мА. Однако, так как в нашем случае используются диоды низкой яркости, сила тока на контакт порта снижена до 15 мА.

Для проверки работоспособности диода можно можно подключить его с сопротивлением 1кОм к напряжению до 12 В, однако при таком сопротивлении имеющиеся диоды имеют недостаточную светимость. Значение требуемого сопротивления резистора вычисляется по формуле:

$$I = \frac{U_S - U_L}{R}, \quad (1)$$

где U_S – напряжение питания; U_L – прямое напряжение светодиода.

Имея значение напряжения питания, равным 5 В и значение прямого напряжения светодиода, равное 1.5 В и значение силы тока, равное 0.15 мА, рассчитывается требуемое сопротивление:

$$R = \frac{U_S - U_L}{I} = \frac{5 - 1.5}{0.015} = 233 \approx 330 \text{ Ом.}$$

Так как в данном случае можно выбрать резистор с большим номиналом, был выбран резистор с номиналом 330 Ом. С таким же сопротивлением был выбран резистор для кнопок.

На аппаратном модуле управления принимающего узла имеется 6 светодиодов и 3 кнопки, поэтому всего требуется 9 резисторов, но так имеется 4 светодиода, из которых в любой момент времени питание подается только на один светодиод, то можно ограничиться одним резистором на эти 4 светодиода.

Согласно рекомендациям по работе с Raspberry Pi 3, максимальное значение силы тока, проходящего через порт GPIO составляет 400 мА. На каждый

контакт порта GPIO компьютера Raspberry Pi 3 рекомендованное значение силы тока составляет 50 мА. Однако, так как в нашем случае используются диоды низкой яркости, сила тока на контакт порта снижена до 15 мА.

Для проверки работоспособности диода можно подключить его с сопротивлением 1кОм к напряжению до 12 В, однако при таком сопротивлении имеющиеся диоды имеют недостаточную светимость. Значение требуемого сопротивления резистора вычисляется по формуле:

$$I = \frac{U_S - U_L}{R},$$

где U_S – напряжение питания; U_L – прямое напряжение светодиода.

Имея значение напряжения питания, равным 5 В и значение прямого напряжения светодиода, равное 1.5 В и значение силы тока, равное 0.15 мА, рассчитывается требуемое сопротивление:

$$R = \frac{U_S - U_L}{I} = \frac{5 - 1.5}{0.015} = 233 \approx 330 \text{ Ом.}$$

В результате разработки модулей было собрано 2 аппаратных модуля управления, приведенных на рисунке 13.

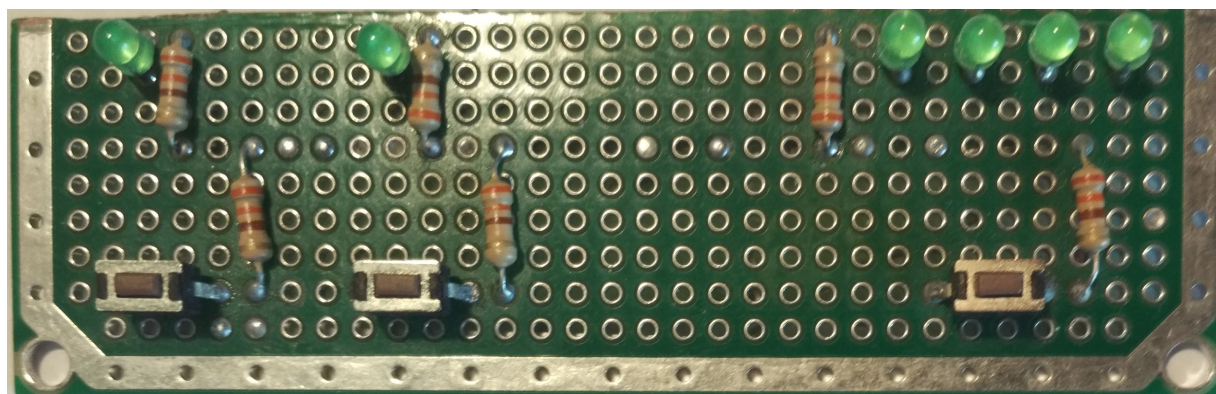


Рисунок 13 — Изображение аппаратного модуля управления

Подключение аппаратного модуля к Raspberry Pi 3 производится согласно схеме, приведенной на рисунке 11.

В ходе разработки аппаратной части аппаратно-программного комплекса для исследования способов скрытой передачи информации, приведенном

в пункте 2.2, была разработана структура аппаратного комплекса и описана в виде схематической структуры аппаратной части комплекса, диаграммы вариантов использования разрабатываемого аппаратно-программного комплекса, принципиальной схемы аппаратного модуля управления для одноплатного компьютера Raspberry Pi 3.

2.3 Программное обеспечение

2.3.1 Определение программных компонентов комплекса

После выбора готовых аппаратных решений и разработки требуемых аппаратных компонентов, выполняется разработка программной части для каждого аппаратного компонента. В результате была разработана диаграмма развертывания UML для аппаратно-программного комплекса для исследования способов скрытой передачи информации, приведенная на рисунке 14.

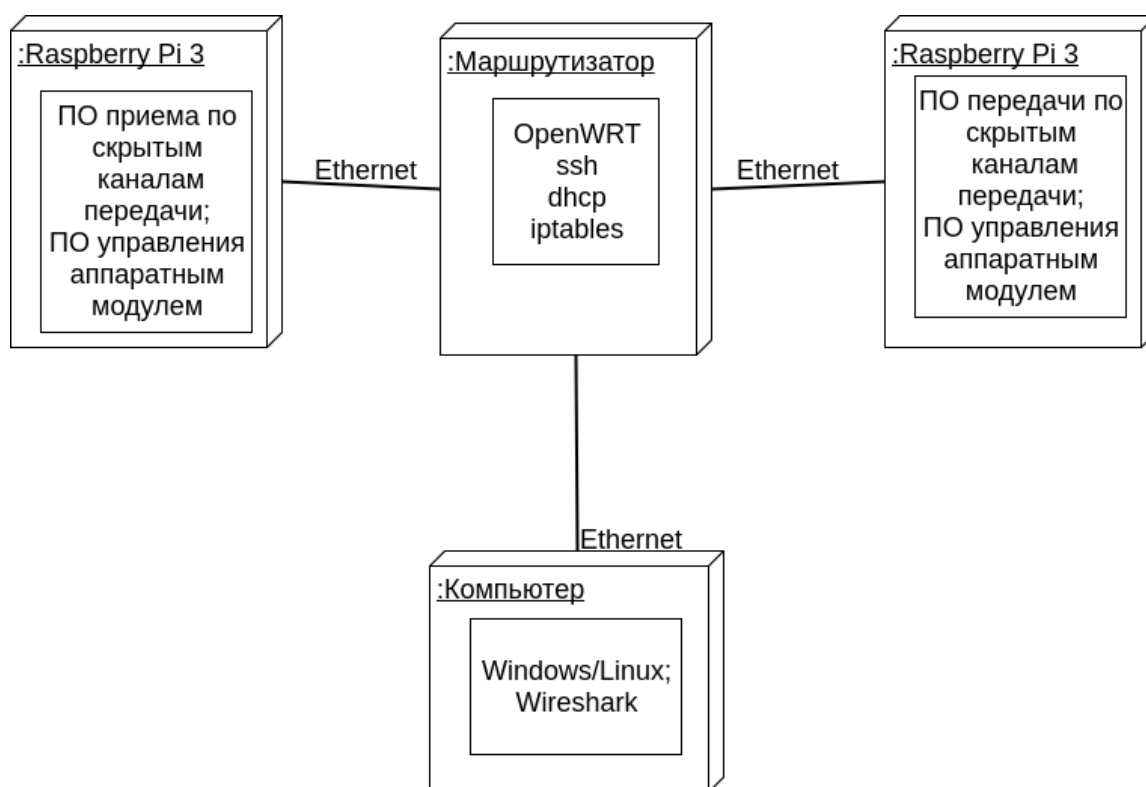


Рисунок 14 — Диаграмма развертывания для разрабатываемого аппаратно-программного комплекса

Аппаратно-программный комплекс для исследования способов скрытой передачи информации функционирует следующим образом: пользователь №1 на принимающем узле выставляет следующие параметры приема:

- выбирается один из четырех способов приема информации. Выбирается с помощью кнопки №1 на аппаратном модуле управления;
- по желанию пользователя включается режим сжатия передаваемой информации (кнопка №2);
- ожидание передачи после выбора параметров (кнопка №3).

Далее, после настройки принимающего узла, выполняется настройка передающего узла. Пользователь №1 на принимающем узле выставляет следующие параметры передачи:

- выбирается один из четырех способов передачи информации (должен совпадать с выбранным способом на принимающем узле). Выбирается с помощью кнопки №1 на аппаратном модуле управления;
- значение режима сжатия выставляется аналогичным значению на принимающем узле (кнопка №2);
- начало передачи информации по скрытому каналу передачи (кнопка №3).

По началу скрытой передачи, в зависимости от выбранного способа скрытой передачи информации и используемого протокола, передающий и принимающий узлы начинают генерировать трафик. Получаемый трафик от обеих устройств зеркалируется на узел, выступающий в роли наблюдателя или перехватчика. Просмотр трафика осуществляется с помощью программы Wireshark.

2.3.2 Выбор функционала программной части

Согласно предъявляемым требованиям к аппаратно-программному комплексу для исследования способов скрытой передачи информации было проведено описание функционала и поведения программной части и его представление в виде диаграммы вариантов использования UML. С помощью данной диаграммы возможно определить действующих лиц и их взаимодействие с системой, а также описать ожидаемую функциональность системы. Полученная диаграмма вариантов использования для разрабатываемого программного обеспечения для аппаратно-программного комплекта для исследования способов скрытой передачи информации приведена на рисунке 15.

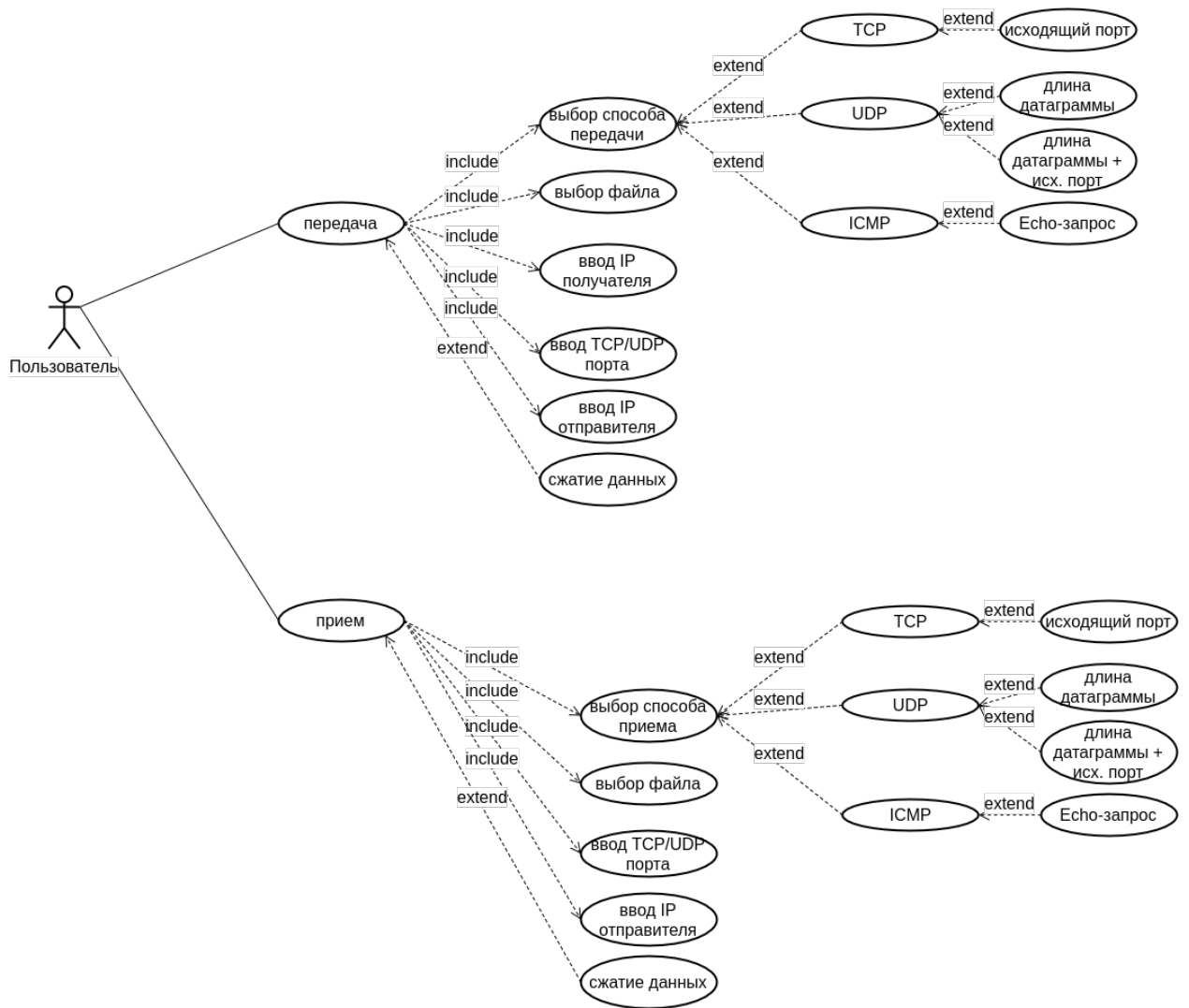


Рисунок 15 — Диаграмма вариантов использования программной части комплекса UML

Согласно приведенной рисунке 15 диаграмме вариантов использования, проведена первичная разработка структуры программной части аппаратно-программного комплекса для исследования способов скрытой передачи информации. Таким образом, программная часть комплекса включает в себя следующие основные модули:

- центральный модуль, отвечающий за выбор режима приема или передачи по скрытым каналам;
- модуль, отвечающий за прием и передачу информации по скрытому каналу передачи, реализованному с помощью протокола ICMP;
- модуль, отвечающий за прием и передачу информации по скрытому каналу передачи, реализованному с помощью протокола UDP;

- модуль, отвечающий за прием и передачу информации по скрытому каналу передачи, реализованному с помощью протокола TCP;
- модуль, отвечающий за взаимодействие программной части комплекса с аппаратным модулем управления.

Разработка программной части для выбранного оборудования возможна с помощью множества программных инструментов. Разработка под одноплатный компьютер Raspberry Pi 3 может вестись с использованием компилируемых языков программирования (например, C, C++, Go, и т.д.) или интерпретируемых языков программирования (Perl, Python, и т.д.). Язык C является наиболее известным представителем в своей области, ввиду его распространенности под различные архитектуры и платформы. Программный код, написанный на языке C обладает максимальной скоростью работы, однако основным недостатком языка является большое количество потенциально опасных элементов и конструкций, содержащихся в языке. Например, основными проблемами являются утечка памяти, ввиду отсутствия в языке механизма сборки мусора, наличие уязвимых функций (printf, strcpy и т.д.).

Язык C++ – компилируемый язык общего назначения со статической типизацией. Поддерживает такие парадигмы программирования, как объектно-ориентированное программирование и обобщенное программирование. Синтаксис языка унаследован от C и в большинстве своем является с ним совместимым. В ядре самого языка, также как и в его стандартной библиотеке отсутствует механизм сборки мусора, что накладывает на разработчика дополнительные задачи по контролю своевременного освобождения памяти.

Golang – C-подобный язык программирования высокого уровня. Разрабатывается как язык системного программирования для создания системных программ с акцентом на работу с современными распределенными системами и многоядерными процессорами. Программы на Go компилируются в объектный код и не требуют для исполнения каких-либо дополнительных инструментов. В языке по умолчанию имеется поддержка механизма сборки мусора, что позволяет разработчикам сосредоточить свое внимание на логике са-

мого программного средства. Компилятор поддерживает компиляцию и кросс-компиляцию под большинство современных архитектур и платформ [7].

Python – высокоуровневый язык программирования общего назначения, ориентированный на повышение производительности разработчика и читаемости кода. Синтаксис ядра Python минималистичен. В то же время стандартная библиотека включает большой объём полезных функций. Python поддерживает несколько парадигм программирования, в том числе структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное. Основные архитектурные черты – динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений и удобные высокоуровневые структуры данных [8].

Pcap – библиотека, позволяющая создавать программы анализа сетевых данных, поступающих на сетевую карту компьютера. Примером программного обеспечения, использующего библиотеку Pcap, служит программа Wireshark. Разнообразные программы мониторинга и тестирования сети, sniffеры используют эту библиотеку. Она предназначена для использования совместно с языками C/C++, а для работы с библиотекой на других языках, таких как Java, .NET, используют обёртки. Для Unix-подобных систем это библиотека libpcap, а для Microsoft Windows — WinPcap. Программное обеспечение сетевого мониторинга может использовать libpcap или WinPcap, чтобы захватить пакеты, путешествующие по сети, и (в более новых версиях) для передачи пакетов в сети. Libpcap и WinPcap также поддерживают сохранение захваченных пакетов в файл и чтение файлов, содержащих сохранённые пакеты. Программы, написанные на основе libpcap или WinPcap, могут захватить сетевой трафик, анализировать его. Файл захваченного трафика сохраняется в формате, понятном для приложений, использующих Pcap [9].

Wireshark – программа, позволяющая пользователю просматривать весь проходящий по сети трафик в режиме реального времени, переводя сетевую карту в неразборчивый режим (promiscuous mode). Это приложение, ко-

торое «знает» структуру самых различных сетевых протоколов, и поэтому позволяет разобрать сетевой пакет, отображая значение каждого поля протокола любого уровня. Поскольку для захвата пакетов используется `pcap`, существует возможность захвата данных только из тех сетей, которые поддерживаются этой библиотекой. Тем не менее, Wireshark умеет работать с множеством форматов входных данных, соответственно, можно открывать файлы данных, захваченных другими программами, что расширяет возможности захвата [10].

`Tcpdump` – утилита, основанная на библиотеке `Pcap`, позволяющая перехватывать и анализировать сетевой трафик, проходящий через компьютер, на котором запущена данная утилита. `Tcpdump` выводит описание содержимого пакетов на каком-либо интерфейсе в соответствии с заданным фильтром. С помощью фильтра возможно отбирать пакеты по таким параметрам, как IP-адрес, TCP/UDP-порт, флаги TCP-пакетов и тому подобное [11].

`Gpiozero` – библиотека, написанная на языке Python, позволяющая осуществлять работу с портом GPIO одноплатных компьютеров Raspberry Pi 3. Одной из самых интересных особенностей Raspberry Pi является наличие портов GPIO (general purpose input/output). Благодаря этому компьютер можно использовать для управления различными устройствами, а также управлять самим устройством, путем программирования отдельных пинов порта. В модели «В» платы присутствует 26-пиновый, а в модели «В+» и «2 В» — 40-пиновый разъем GPIO [12].

Для разработки программной части было принято решение использовать следующие программные решения:

- языки программирования Go, Python;
- библиотека анализа сетевого трафика `Pcap`;
- библиотека для работы с портом GPIO одноплатного компьютера Raspberry Pi 3 `gpiozero`;
- анализатор сетевого трафика Wireshark.

Для реализации программной части выбран язык Go ввиду ряда причин:

- С-подобный язык со статической типизацией. Язык обладает достаточно низким порогом вхождения, соответственно, пучаемый код является более простым для понимания. Также языком поддерживаются более совершенные и удобные языковые конструкции (например, конструкции `for`);
- автоматическое управление памятью с помощью сборщика мусора;
- встроенная поддержка средств параллельного программирования (`go routines`);
- обширная коллекция библиотек, как встроенных, так и внешних;
- поддержка множества разнообразных архитектур и операционных систем, что дает возможность сделать разрабатываемое программное обеспечение легко портируемым.

В качестве библиотеки анализа сетевого трафика для программной части, отвечающей за организацию скрытых каналов передачи информации выбрана библиотека `Rcar`. Причины выбора библиотеки:

- библиотека `Rcar` является кроссплатформенной. Имеется поддержка таких операционных систем, как Windows, Linux, MacOS, FreeBSD;
- для `Rcar` реализовано множество интерфейсов под разные языки программирования (в том числе, для языка `Go`);
- поддерживает требуемые функции фильтрации пакетов и извлечения из них данных.

Для реализации программного интерфейса для аппаратного модуля управления был выбран язык `Python` ввиду следующего ряда причин:

- наличие интерпретатора языка для одноплатного компьютера Raspberry Pi 3;
- удобный синтаксис, позволяющий писать короткий и эффективный программный код;
- наличие библиотеки `gpiozero`, написанной на языке `Python`;
- возможность использовать написанный код в качестве демона (фоновый процесс, запускаемого операционной системой).

В качестве анализатора сетевого трафика выбран программный инструмент Wireshark ввиду следующего ряда причин:

- наличие графического интерфейса (GUI);
- поддержка фильтрации перехватываемых пакетов. Возможность создания требуемых правил фильтрации;
- удобное представление структуры пакетов при анализе;
- наличие представления пакетов в шестнадцатеричном виде.

2.3.3 Разработка программной части, реализующей скрытые каналы передачи информации

Основная программная часть, в которой реализованы скрытые каналы передачи, реализована с помощью языка программирования Go. На рисунке 16 приведена структура проекта и основные используемые функции.

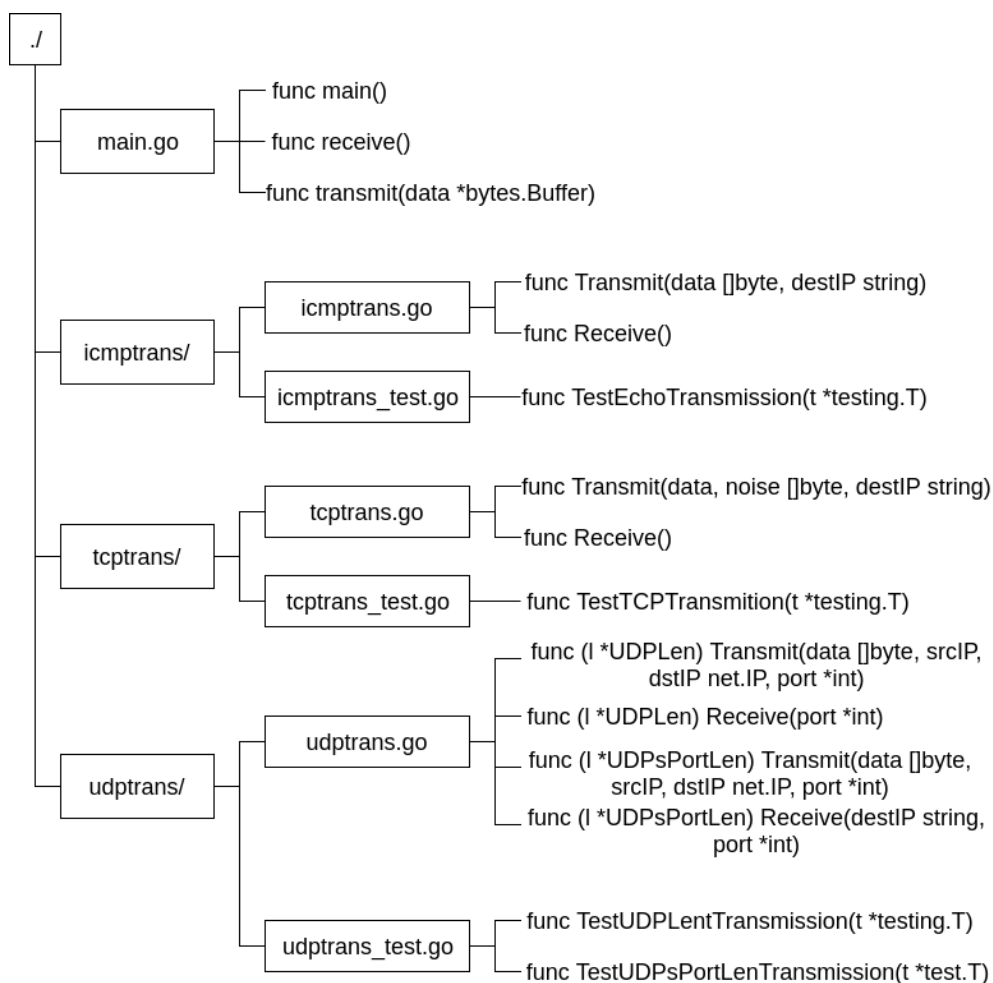


Рисунок 16 — Структура проекта, реализующего скрытые каналы передачи информации, и основные используемые функции

Основными модулями, реализующими функционал программной части, являются модуль `icmptrans`, реализующий скрытый канал передачи информации с помощью протокола ICMP, `tcptrans`, реализующий скрытый канал передачи информации с помощью протокола TCP, `udptrans`, реализующий скрытый канал передачи информации с помощью протокола UDP. Также для каждого модуля реализован модуль тестирования, предназначенный для проверки корректности выполнения программ. В имени данных модулей содержится суффикс `test`.

В каждом из трех модулей проекта имеются функции `Transmit` и `Receive`. Например, в модуле `icmptrans` данные функции отвечают за организацию скрытого канала передачи информации по протоколу ICMP. Передача реализована путем отправки последовательности байт в Echo-запросах. Исходные тексты модуля приведены в приложении А.

В модуле `tcptrans` с помощью функций `Transmit` и `Receive` реализованы скрытые прием и передача информации с помощью поля "Исходящий порт" в TCP-пакете. Исходные тексты модуля приведены в приложении Б.

В модуле `udptrans` с имеется две реализации функций `Transmit` и `Receive`. Каждая пара функций отвечает за реализацию одного способа скрытой передачи. Первый способ скрытой передачи реализован с помощью поля "Длина датаграммы" заголовка протокола UDP. Второй способ реализован с помощью полей "Длина датаграммы" и "Исходящий порт". Выбор набора функций зависит от структуры, выполняющей роль объекта. Исходные тексты модуля приведены в приложениях В и Г.

В результате разработки проекта была получена структура связей пакетов программной части и представлена в виде диаграммы пакетов UML. Данная диаграмма позволяет показать зависимость между пакетами программного обеспечения. Диаграмма приведена на рисунке 17.

Дополнительно, для реализации необходимых функций потребовалось импортировать следующие библиотеки(в контексте языка Go — пакеты):

- **gopacket** – пакет, предоставляющий набор инструментов по декодированию пакетов. Для работы в качестве зависимостей требует библиотека анализа сетевого трафика Pcap;
- **x/net/icmp** – пакет, предоставляющий базовые возможности по манипуляции сообщениями используемыми в протоколах ICMPv4 и ICMPv6;
- **x/net/ipv4** – пакет, реализующий функции IPv4-протокола;
- **I/O** – пакет, предоставляющий набор функций, реализующих базовые примитивы ввода-вывода. По своей сути является оболочкой для уже существующих функций, реализованных и используемых в операционных системах Linux и Windows.

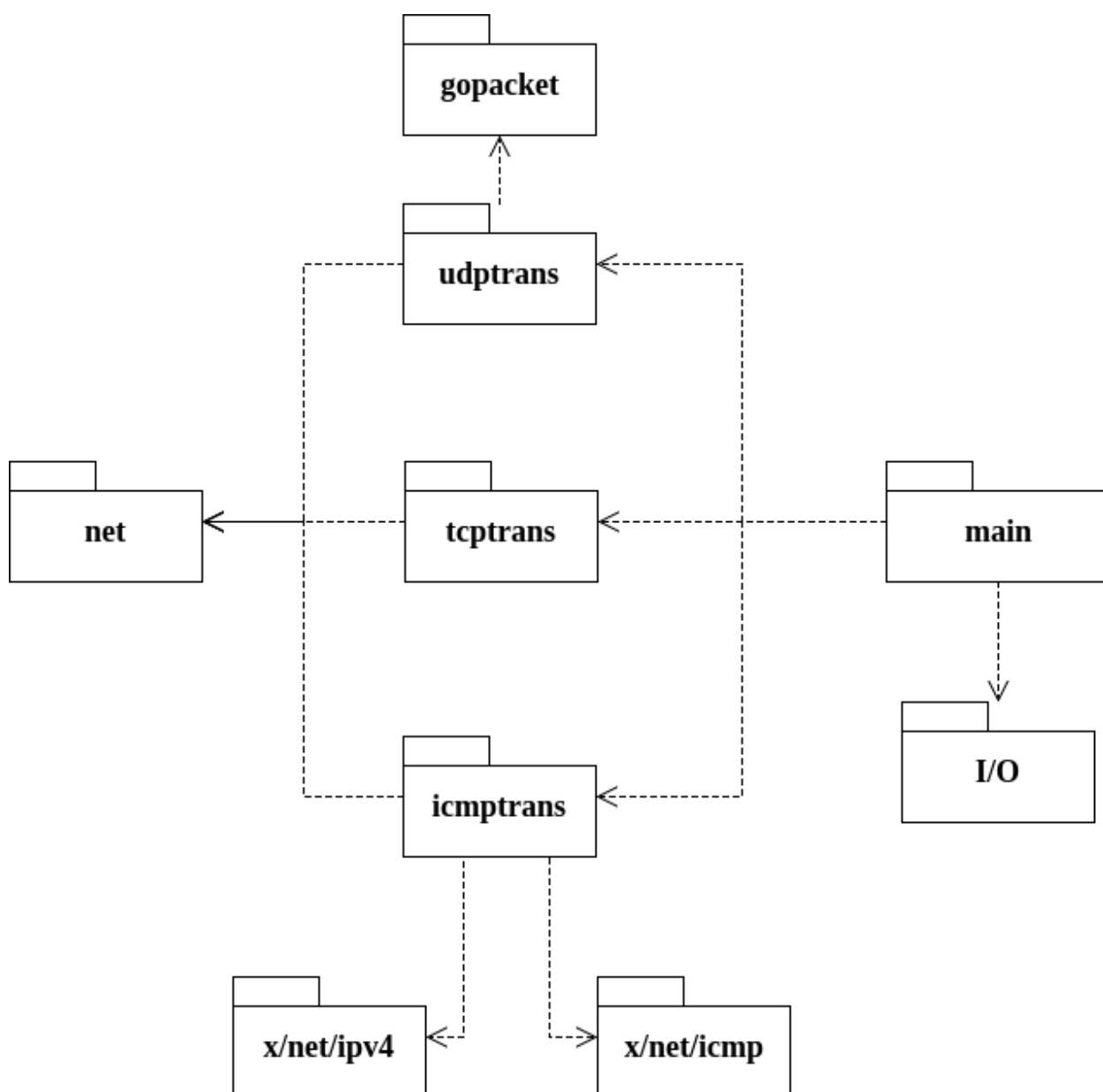


Рисунок 17 — Диаграмма пакетов UML

Таким образом, был разработан элемент программной части аппаратно-программного комплекса для исследования способов скрытой передачи информации, реализующий скрытые каналы передачи информации. На рисунке 18 приведен снимок с описанием используемых параметров в программной части, реализующей скрытые каналы передачи.

```
→ ~/coding/hidchan/hidchan -h [e0f0391]
Usage of /home/roman/coding/hidchan/hidchan:
-c    Use compression: true/false
-d string
      Destination IPv4
-i string
      File for transmtion
-mode string
      [T]ransmit or [R]eceive mode
-n string
      File name to save. Default: resData (default "resData")
-o string
      Saving directory (default ".")
-port int
      TCP/UDP port
-proto string
      [T]CP, [U]DP, [I]CMP
-s string
      Source IPv4
-usp string
      UDP[L]en, UDP[s]portLen,
```

Рисунок 18 — Описание рабочих параметров разрабатываемого программного обеспечения

2.3.4 Разработка программной части для аппаратного модуля управления

На рисунке 19 приведен фрагмент кода, отвечающего за взаимодействие программной части с аппаратным модулем управления. Код написан на языке Python и разработан таким образом, чтобы быть универсальным как для узла-отправителя, так и для узла-получателя.

```
import sys, signal, subprocess
from gpiozero import LED, Button
leds = [LED(14), LED(15), LED(18), LED(23), LED(25), LED(7)]
sw1 = Button(24); sw2 = Button(8); sw3 = Button(12)
```

Рисунок 19 — Фрагмент кода, отвечающего за назначение функций контактам GPIO-порта

На приведенном рисунке изображен фрагмент, в котором контактам GPIO-порта назначаются функции. Так, контактам GPIO14, GPIO15, GPIO18,

GPIO23, GPIO25, GPIO7 назначается функция светодиода, чтобы к ним можно было подключить светодиоды LED, а контактам GPIO24, GPIO8, GPIO12 назначается функция кнопки.

В рисунке 20 приведен фрагмент функции, описывающей поведение элементов аппаратного модуля управления в режиме приема, при котором светодиоды led1-led4 программируются для выбора способа передачи и загораются поочередно при нажатии на кнопку btn1. Светодиод led5 используется для включения режима сжатия передаваемых данных и включается при нажатии на кнопку btn2. Светодиод led6 используется для индикации статуса приема. Прием начинается по нажатии на кнопку btn4.

```
def receive():
    cmd = []
    if leds[0].is_lit is True:
        proto = 'I'
        cmd = ['hidchan', '-mode', 'R', '-proto', proto, '-s',
              '192.168.1.5', '-d', '192.168.1.10']
    elif leds[1].is_lit is True:
        proto = 'T'
        cmd = ['hidchan', '-mode', 'R', '-proto', proto, '-s',
              '192.168.1.5',
              '-d', '192.168.1.10']
        ...
        if leds[4].is_lit is True:
            compress = ['-c', 'g']
            cmd.append(compress)
    else:
        compress = ['-c', 'n']
        cmd.append(compress)
    hidchan = subprocess.Popen(cmd, stdout=subprocess.PIPE)
    leds[5].blink()
    hidchan.wait()
    leds[5].off()
    return
```

Рисунок 20 — Фрагмент кода, описывающий поведение элементов аппаратного модуля в режиме приема

На рисунке 21 приведена функция, описывающая поведение элементов аппаратного модуля управления в режиме приема, при котором светодиоды led1-led4 программируются для выбора способа передачи и загораются поочередно при нажатии на кнопку btn1. Светодиод led5 используется для включения режима сжатия передаваемых данных и включается при нажатии на кнопку btn2. Светодиод led6 используется для индикации статуса приема. Прием начинается по нажатии на кнопку btn4.

```
def transmit():
    cmd = []
    if leds[0].is_lit is True:
        proto = 'I'
        cmd = ['hidchan', '-mode', 'T', '-proto', proto, '-s',
              '192.168.1.5', '-d', '192.168.1.10', '-i', randomfile()]
    elif leds[1].is_lit is True:
        proto = 'T'
        port = '34442'
        cmd = ['hidchan', '-mode', 'T', '-proto', proto, '-s',
              '192.168.1.5', '-d', '192.168.1.10', '-i', randomfile()]
    if leds[4].is_lit is True:
        compress = ['-c', 'g']
    else:
        compress = ['-c', 'n']
    cmd.append(compress)
    hidchan = subprocess.Popen(cmd, stdout=subprocess.PIPE)
    leds[5].blink()
    hidchan.wait()
    leds[5].off()
    return

def switch_transmit_mode_led():
    for i in range(4):
        if leds[i].is_lit is True:
            leds[(i+1) % 4].on()
            leds[i].off()
            break
```

Рисунок 21 — Фрагмент кода, описывающий поведение элементов аппаратного модуля в режиме передачи

На рисунке 22 приведен фрагмент, описывающий режимы работы аппаратного модуля управления, такие как режим приема и режим передачи. В данном фрагменте выполняется выбор одной из функций, приведенных на рисунках 20 и 21.

```
if __name__ == '__main__':
    if len(sys.argv) != 2:
        print('error: working mode was not set')
        sys.exit()
    #receive or transmit mode
    sw1.when_pressed = switch_transmit_mode_led
    sw2.when_pressed = toggle_compression
    leds[0].on()

    if sys.argv[1] == '-R' or sys.argv[1] == '--receive':
        sw3.when_pressed = receive
    elif sys.argv[1] == '-T' or sys.argv[1] == '--transmit':
        sw3.when_pressed = transmit
    signal.pause()
```

Рисунок 22 — Фрагмент кода, описывающий режимы работы аппаратного модуля управления

Приведенный выше код отвечает за взаимодействие аппаратного модуля управления с компьютером через порт GPIO. В начале определяются контакты, к которым будут подключены 6 светодиодов и 3 кнопки. Группа из четырех светодиодов назначается для выбора режима передачи, пятый светодиод программируется для выбора режима сжатия при передаче. шестой светодиод программируется для индикации статуса активности приема или передачи. Кнопки запрограммированы для выбора режима передачи, режима сжатия и сигнала о начале приема или передачи. Данная программа является универсальной как для принимающего, так и для передающего узлов. Запускается в качестве демона, то есть программы, работающей в фоновом режиме и загружаемой автоматически включении устройства. Схема аппаратного модуля управления приведена на рисунке 11.

2.3.5 Проверка разработанной программной части на работоспособность

Для определения правильности работы реализованных алгоритмов было произведено тестирование на предмет корректности передачи с использованием скрытых каналов передачи. В качестве тестовых данных использовалась сгенерированная случайная последовательность размером 1000 байт. Результат тестирования приведен в таблице 1.

Таблица 1 — Результаты тестирования реализованных скрытых каналов передачи

Используемый протокол	Общее кол-во передач	Количество байт на пакет	Количество успешных передач	Количество неудачных передач	% успешных передач
ICMP	10	20	10	0	100
TCP	10	1	10	0	100
UDP №1	10	1	9	1	90
UDP №2	10	2	7	3	70

На основании полученных результатов составлена сводная диаграмма зависимости скорости скрытой передачи информации от способа передачи, приведенная на рисунке 23.

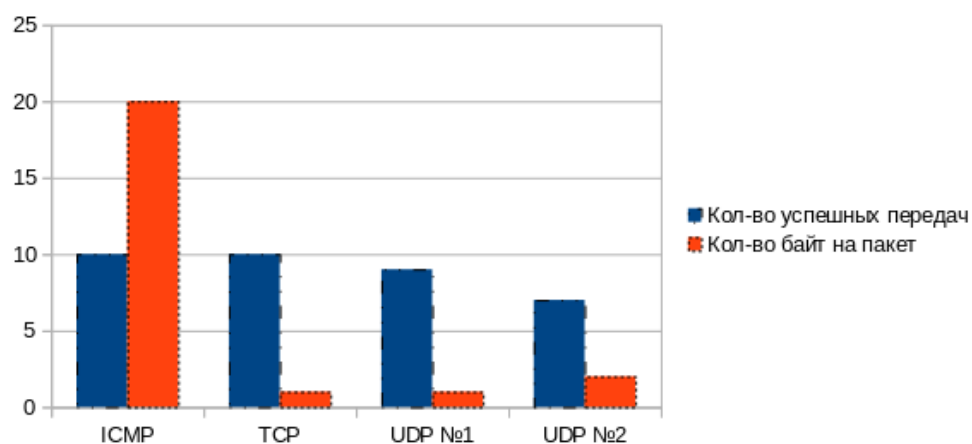


Рисунок 23 — Диаграмма зависимости скорости передачи информации и количества успешных передач от способа передачи

2.4 Методика по работе с аппаратно-программным комплексом для исследования способов скрытой передачи информации в компьютерных сетях

Согласно заданию на дипломное проектирование, требуется разработать методические указания по работе с аппаратно-программным комплексом для исследования способов скрытой передачи информации в компьютерных сетях.

Аппаратно-программный комплекс для исследования способов скрытой передачи информации в компьютерных сетях рассчитан для студентов, обучающихся по специальностям, связанным с информационной безопасностью. Комплекс призван дать студентам представление о том, что такое скрытые каналы передачи информации и помочь получить навыки в поиске таких каналов.

Ввиду специфики разрабатываемого аппаратно-программного комплекса, он может быть использован для изучения скрытых каналов передачи информации на таких дисциплинах, как "Безопасность сетей ЭВМ" и "Сети и системы передачи информации" кафедры "Информационная безопасность систем и технологий" Пензенского государственного университета.

По завершению работы с аппаратно-программным комплексом студенты должны усвоить тот факт, что любая система может содержать в себе недокументированные возможности, которые могут быть использованы злоумышленниками в корыстных целях.

2.4.1 Состав аппаратно-программного комплекса для исследования способов скрытой передачи информации

В состав аппаратно-программного комплекса входят следующие компоненты:

- маршрутизатор D-Link DIR320nru;
- одноплатный компьютер Raspberry Pi 3, выполняющий функцию узла, передающего информацию по скрытым каналам передачи;
- одноплатный компьютер Raspberry Pi 3, выполняющий функцию узла, принимающего информацию из скрытых каналов передачи.

2.4.2 Состав узлов, участвующих в передаче информации по скрытому каналу

В состав аппаратно-программного комплекса входят оконечные устройства, являющиеся одноплатными компьютерами Raspberry Pi 3 и организующие передачу по скрытому каналу. Управление устройствами осуществляется с помощью аппаратного модуля управления, подключаемого к порту GPIO одноплатного компьютера Raspberry Pi 3. Внешний вид данного элемента аппаратно-программного комплекса с аппаратным модулем управления приведен на рисунке 24.

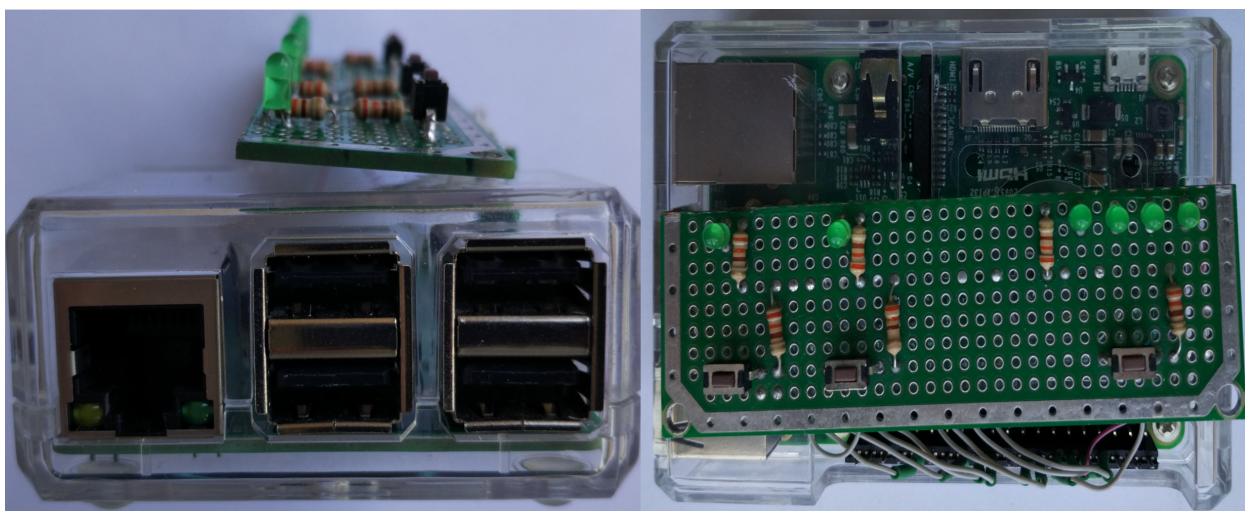


Рисунок 24 — Внешний вид конечного устройства с аппаратным модулем управления

Аппаратная и программная реализация оконечных устройств является идентичной. В зависимости от требуемого режима работы устройства (прием или передача) изменяется поведение запрограммированных кнопок аппаратного модуля управления. На рисунке 25 приведена схема аппаратного модуля управления для работы в режиме передачи. Как видно по рисунку, модуль состоит из 6 диодов и 3 кнопок. Первые четыре светодиода предназначены для выбора скрытого канала передачи информации и загораются поочередно при нажатии на кнопку снизу. Каждый светодиод означает выбор соответствующего скрытого канала передачи. Предусмотрено 4 способа организации передачи по скрытому каналу (способы перечислены в порядке, соответствующем расположению на модуле):

- передача с помощью протокола ICMP;
- передача с помощью протокола TCP;
- передача с помощью протокола UDP (вариант №1);
- передача с помощью протокола UDP (вариант №2).

Светодиод №5 является индикатором включения режима сжатия передаваемой информации (горит – режим сжатия активен; не горит – не активен). Сжатие осуществляется при помощи утилиты GZIP, которая сжимает передаваемые данные по алгоритму сжатия без потерь DEFLATE. Переключение режимов осуществляется при нажатии на кнопку "Сжатие".

Скрытая передача данных начинается при нажатии на кнопку "Передача", при этом светодиод, расположенный над этой кнопкой, начинается мигать, чем сигнализирует о том, что в данный момент выполняется передача. По завершению передачи светодиод перестает мигать, чем сигнализирует об окончании процесса передачи.

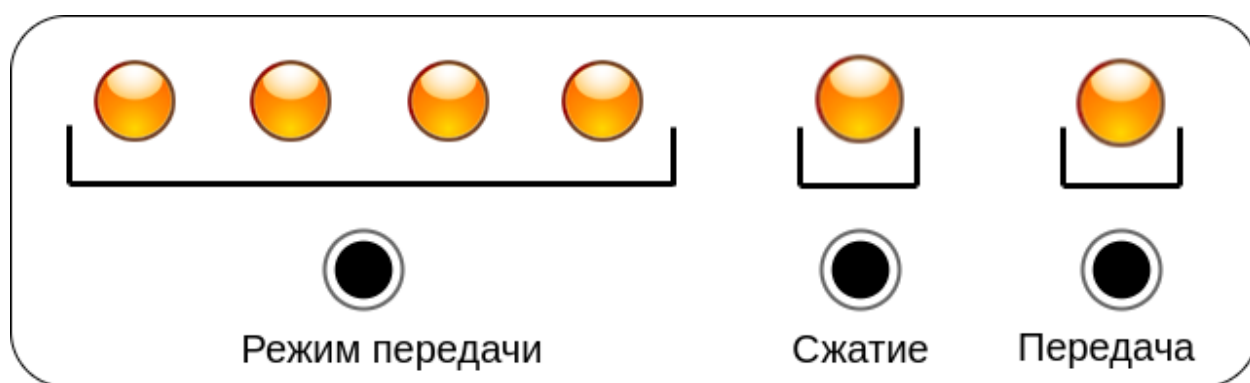


Рисунок 25 — Аппаратный модуль управления передачей

На рисунке 26 приведена схема аппаратного модуля управления для работы в режиме приема. Как видно по рисунку, модуль состоит из 6 диодов и 3 кнопок. Первые четыре светодиода предназначены для выбора скрытого канала передачи информации и загораются поочередно при нажатии на кнопку снизу. Каждый светодиод означает выбор соответствующего скрытого канала, по которому будет ожидать прием данных. Предусмотрено 4 способа организации приема по скрытому каналу (способы перечислены в порядке, соответствующем расположению на модуле):

- передача с помощью протокола ICMP;
- передача с помощью протокола TCP;
- передача с помощью протокола UDP (вариант №1);
- передача с помощью протокола UDP (вариант №2).

Светодиод №5 является индикатором включения режима сжатия принимаемой информации (горит – режим сжатия активен; не горит – не активен). При включенном режиме сжатия принимаемые данные восстанавливаются после приема. Восстановление принимаемых данных осуществляется при помощи утилиты GZIP, которая сжимает передаваемые данные по алгоритму сжатия без потерь DEFLATE. Переключение режимов осуществляется при нажатии на кнопку "Сжатие".

Ожидание приема скрыто передаваемых данных начинается при нажатии на кнопку "Передача", при этом светодиод, расположенный над этой кнопкой, начинается мигать, чем сигнализирует о том, что в данный момент устройство находится в режиме ожидания приема. По завершению приема светодиод перестает мигать, чем сигнализирует об окончании процесса передачи.

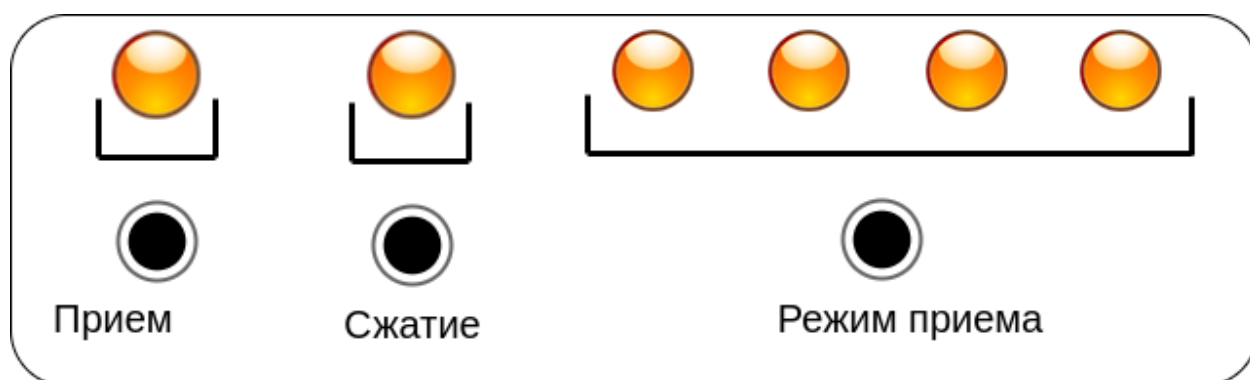


Рисунок 26 — Аппаратный модуль управления приемом

2.4.3 Подключение компьютера для перехвата и анализа трафика

В состав аппаратно-программного комплекса входят маршрутизатор и 2 одноплатных компьютера, выполняющих скрытую передачу информации, однако в состав комплекса не входит элемент, предназначенный для анализа перехватываемого трафика. Поэтому дополнительно требуется подключить дополнительный компьютер. Данный компьютер будет принимать зеркалируемый

трафик и представлять его в читабельном виде с помощью программы анализа сетевого трафика Wireshark.

Подключение компьютера к аппаратно-программному комплексу осуществляется посредством WiFi-соединения. Для подключения к сети необходимо на данном компьютере подключиться к точке доступа с именем Stegano_АПК и ввести пароль Stegano_АПК.

Подключившись к сети аппаратно-программного комплекса для исследования способов скрытой передачи информации, требуется запустить программу анализа сетевого трафика Wireshark. В программе выбрать интерфейс, с которого будет сниматься трафик, то есть интерфейс беспроводного модуля.

В текстовое поле "Filter" программы Wireshark можно ввести правила отбора перехватываемых пакетов. Например для выбранных протоколов предусмотрены следующие параметры фильтрации:

- IP – ip.addr, ip.dst, ip.src;
- TCP – tcp.flags.syn, tcp.srcport, tcp.dstport;
- UDP – udp.length, udp.dstport, udp.srcport.

2.4.4 Выполнение скрытой передачи информации с помощью протокола ICMP

В подразделе 2.4.4 описывается методика по настройке режимов скрытой передачи информации с помощью протокола TCP и перехвату получаемого трафика. Настройка режима на устройстве, выполняющем передачу по скрытому каналу выполняется с помощью модуля управления, подключенного к одноплатному компьютеру Raspberry Pi 3. Выбор режима выполняется посредством нажатия на кнопку "Режим передачи". При нажатии на кнопку загорается соответствующий светодиод, означающий выбранный способ передачи информации. На рисунке 27 приводится снимок, показывающий, какой из светодиодов должен быть выставлен на устройстве передачи по скрытому каналу.

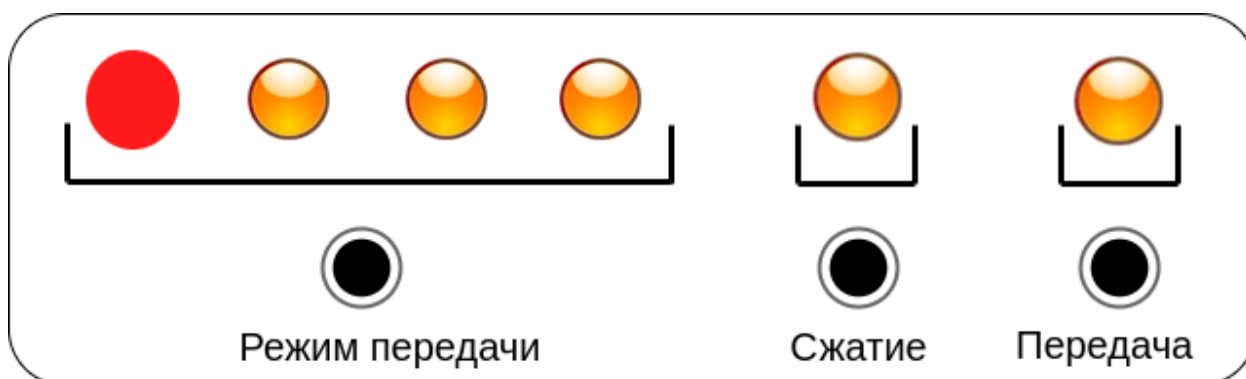


Рисунок 27 — Состояние аппаратного модуля при выбранном режиме скрытой передачи с помощью протокола ICMP

Режим сжатия данных предназначен для ускорения передачи по скрытому каналу включается по нажатии на кнопку "Сжатие". Для начала передачи необходимо нажать на кнопку "Передача" на модуле управления передающего устройства.

При нажатии на кнопку "Передача" начинается передача по выбранному скрытому каналу передачи. Во время передачи светодиод равномерно мигает. По окончании передачи светодиод гаснет.

Все пакеты, передаваемые конечными устройствами по сети зеркалируются на компьютер, подключенный через WiFi. На рисунке 28 приведен результат передачи файла с исходными текстами main.go.

70	0.708327286	192.168.1.10	192.168.1.1	ICMP	62 Echo (ping) request
71	0.708330872	192.168.1.10	192.168.1.1	ICMP	62 Echo (ping) request
72	0.708334549	192.168.1.10	192.168.1.1	ICMP	62 Echo (ping) request
73	0.708338209	192.168.1.10	192.168.1.1	ICMP	62 Echo (ping) request

```

Checksum: 0xaad8 [correct]
[Checksum Status: Good]
Identifier (BE): 4137 (0x1029)
Identifier (LE): 10512 (0x2910)
Sequence number (BE): 2 (0x0002)
Sequence number (LE): 512 (0x0200)
▶ [No response seen]
▼ Data (20 bytes)
  Data: 65734461746122290a0970726f746f203d20666c
  [Length: 20]
0000  84 c9 b2 a7 61 96 e8 03  9a f8 f0 ac 08 00 45 00  ....a... ..E.
0010  00 30 ce 53 40 00 40 01  e9 1d c0 a8 01 0a c0 a8  .0.S@.@. ....
0020  01 01 08 00 aa d8 10 29  00 02 65 73 44 61 74 61  ..... ) ..esData
0030  22 29 0a 09 70 72 6f 74  6f 20 3d 20 66 6c        ")..prot o = fl

```

Рисунок 28 — Пример трафика, получаемого при скрытой передаче с помощью протокола ICMP

Передача по данному скрытому каналу осуществляется в ICMP-пакетах, содержащих Echo-запроса. Для просмотра фрагмента скрыто передаваемых данных необходимо выбрать ICMP-пакет, содержащий в себе Echo-запрос. Скрыто передаваемые данные передаются в поле ICMP Payload. По-умолчанию данное поле может содержать произвольные последовательности байт, например, на рисунке 29 приведена последовательность байт, которую передает утилита Ping.

```
26:ce:06:00:00:00:00:00:10:11:12:13:
14:15:16:17:18:19:1a:1b:1c:1d:1e:1f:
20:21:22:23:24:25:26:27:28:29:2a:2b:
2c:2d:2e:2f:30:31:32:33:34:35:36:37
```

Рисунок 29 — Последовательность байт, передаваемая утилитой Ping

2.4.5 Выполнение скрытой передачи информации с помощью протокола TCP

В подразделе 2.4.5 описывается методика по настройке режимов скрытой передачи информации и перехвату получаемого трафика. Настройка режима на устройстве, выполняющем передачу по скрытому каналу выполняется с помощью модуля управления, подключенного к одноплатному компьютеру Raspberry Pi 3. Выбор режима выполняется посредством нажатия на кнопку "Режим передачи". При нажатии на кнопку загорается соответствующий светодиод, означающий выбранный способ передачи информации. На рисунке 30 приводится снимок, показывающий, какой из светодиодов должен быть выставлен на устройстве передачи по скрытому каналу.

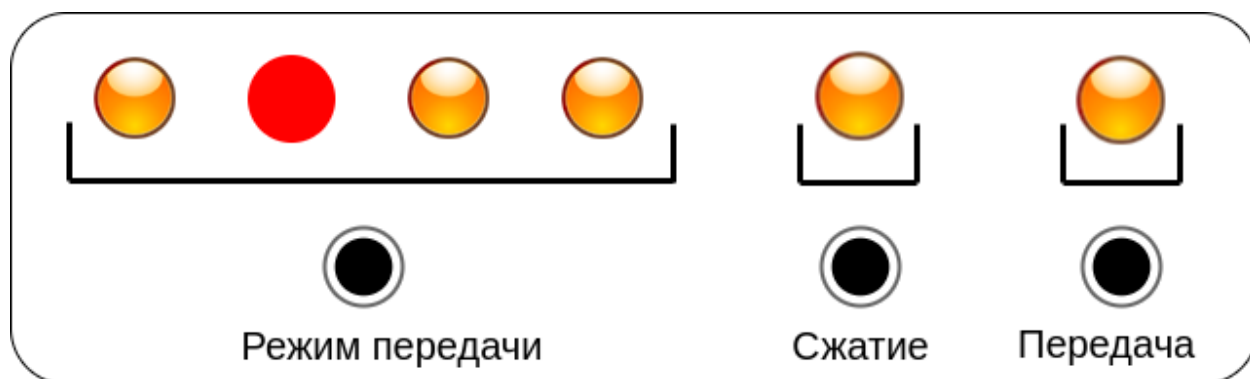


Рисунок 30 — Состояние аппаратного модуля при выбранном режиме скрытой передачи с помощью протокола TCP

Режим сжатия данных предназначен для ускорения передачи по скрытому каналу включается по нажатии на кнопку "Сжатие". Для начала передачи необходимо нажать на кнопку "Передача" на модуле управления передающего устройства.

При нажатии на кнопку "Передача" начинается передача по выбранному скрытому каналу передачи. Во время передачи светодиод равномерно мигает. По окончании передачи светодиод гаснет.

Все пакеты, передаваемые конечными устройствами по сети зеркалируются на компьютер, подключенный через WiFi. На рисунке 31 приведен результат передачи файла с исходными текстами main.go.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	74	53724 → 27116 [SYN] Seq=0 Win=43690 Len=0 MS
2	0.000011568	127.0.0.1	127.0.0.1	TCP	74	27116 → 53724 [SYN, ACK] Seq=0 Ack=1 Win=436
3	0.000019943	127.0.0.1	127.0.0.1	TCP	66	53724 → 27116 [ACK] Seq=1 Ack=1 Win=43776 Le
4	0.000054118	127.0.0.1	127.0.0.1	TCP	75	53724 → 27116 [PSH, ACK] Seq=1 Ack=1 Win=437
5	0.000056846	127.0.0.1	127.0.0.1	TCP	66	27116 → 53724 [ACK] Seq=1 Ack=10 Win=43776 l
6	0.000062852	127.0.0.1	127.0.0.1	TCP	66	53724 → 27116 [FIN, ACK] Seq=10 Ack=1 Win=43
7	0.000094922	127.0.0.1	127.0.0.1	TCP	74	60838 → 27101 [SYN] Seq=0 Win=43690 Len=0 MS
8	0.000099748	127.0.0.1	127.0.0.1	TCP	74	27101 → 60838 [SYN, ACK] Seq=0 Ack=1 Win=436
9	0.000102061	127.0.0.1	127.0.0.1	TCP	66	60838 → 27101 [ACK] Seq=1 Ack=1 Win=43776 Le
10	0.000111347	127.0.0.1	127.0.0.1	TCP	75	60838 → 27101 [PSH, ACK] Seq=1 Ack=1 Win=437
11	0.000112849	127.0.0.1	127.0.0.1	TCP	66	27101 → 60838 [ACK] Seq=1 Ack=10 Win=43776 l
12	0.000115811	127.0.0.1	127.0.0.1	TCP	66	60838 → 27101 [FIN, ACK] Seq=10 Ack=1 Win=43
13	0.000124014	127.0.0.1	127.0.0.1	TCP	74	53382 → 27120 [SYN] Seq=0 Win=43690 Len=0 MS
14	0.000127148	127.0.0.1	127.0.0.1	TCP	74	27120 → 53382 [SYN, ACK] Seq=0 Ack=1 Win=436
15	0.000129181	127.0.0.1	127.0.0.1	TCP	66	53382 → 27120 [ACK] Seq=1 Ack=1 Win=43776 Le

Рисунок 31 — Пример трафика, получаемого при скрытой передаче с помощью протокола TCP

2.4.6 Выполнение скрытой передачи информации с помощью протокола UDP (способ №1)

В подразделе 2.4.6 описывается методика по настройке режимов скрытой передачи информации с помощью протокола UDP и перехвату получаемого трафика. Настройка режима на устройстве, выполняющем передачу по скрытому каналу выполняется с помощью модуля управления, подключенного к одноплатному компьютеру Raspberry Pi 3. Выбор режима выполняется посредством нажатия на кнопку "Режим передачи". При нажатии на кнопку загорается соответствующий светодиод, означающий выбранный способ передачи информации. На рисунке 32 приводится снимок, показывающий, какой из светодиодов должен быть выставлен на устройстве передачи по скрытому каналу.

В данном режиме передача реализована с помощью протокола UDP и использует поле заголовка "Длина датаграммы". Несмотря на то, что это поле имеет размер 2 байта, для скрытой передачи используется только старший байт, так как в это поле не может быть нулевым и минимальное значение, которое оно должно иметь – 7 байт. Таким образом, в старший байт записывается один скрыто передаваемый байт. В соответствии с полученным значением данного поля формируется случайная последовательность данных соответствующего размера, заполняющая поле "Данные".

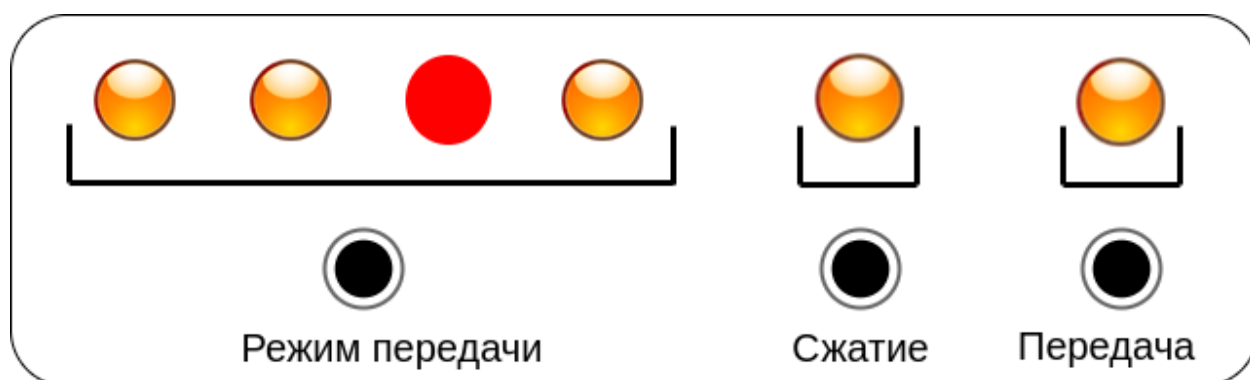


Рисунок 32 — Состояние аппаратного модуля при выбранном режиме скрытой передачи с помощью протокола UDP

Режим сжатия данных предназначен для ускорения передачи по скрытому каналу включается по нажатии на кнопку "Сжатие". Для начала передачи необходимо нажать на кнопку "Передача" на модуле управления передающего устройства.

При нажатии на кнопку "Передача" начинается передача по выбранному скрытому каналу передачи. Во время передачи светодиод равномерно мигает. По окончании передачи светодиод гаснет.

Все пакеты, передаваемые конечными устройствами по сети зеркалируются на компьютер, подключенный через WiFi. На рисунке 33 приведен результат передачи файла с исходными текстами main.go.

No.	Time	Source	Destination	Protocol	Length	Info
1601	13.56911973	192.168.1.121	192.168.1.5	UDP	60	20257 → 40400 Len=4
1618	13.61452553	192.168.1.121	192.168.1.5	UDP	1194	26751 → 40400 Len=24832
1635	13.65272423	192.168.1.121	192.168.1.5	UDP	1194	26751 → 40400 Len=24832
1652	13.69104972	192.168.1.121	192.168.1.5	UDP	1194	26751 → 40400 Len=24832
1669	13.72913978	192.168.1.121	192.168.1.5	UDP	1194	26751 → 40400 Len=24832
1686	13.76726821	192.168.1.121	192.168.1.5	UDP	1194	26751 → 40400 Len=24832
1703	13.80556273	192.168.1.121	192.168.1.5	UDP	1194	26751 → 40400 Len=24832

▶ Frame 1686: 1194 bytes on wire (9552 bits), 1194 bytes captured (9552 bits) on interface 0

▶ Ethernet II, Src: Raspberr_12:ac:1d (b8:27:eb:12:ac:1d), Dst: AsustekC_b7:fe:cf (38:d5:47:b7:fe:cf)

▶ Internet Protocol Version 4, Src: 192.168.1.121, Dst: 192.168.1.5

▼ User Datagram Protocol, Src Port: 26751, Dst Port: 40400

Source Port: 26751

Destination Port: 40400

Length: 24840

Checksum: 0x1a37 [unverified]

[Checksum Status: Unverified]

[Stream index: 1]

▼ Data (24832 bytes)

Data: e4f0caf3375a876d53b09be45b205404db1207ecb041c919...

[Length: 24832]

Рисунок 33 — Пример трафика, получаемого при скрытой передаче с помощью протокола UDP (вариант №1)

2.4.7 Выполнение скрытой передачи информации с помощью протокола UDP (способ №2)

В подразделе 2.4.7 описывается методика по настройке режимов скрытой передачи информации с помощью протокола UDP и перехвату получаемого трафика. Настройка режима на устройстве, выполняющем передачу по скрытому каналу выполняется с помощью модуля управления, подключенного к одноплатному компьютеру Raspberry Pi 3. Выбор режима выполняется посредством нажатия на кнопку "Режим передачи". При нажатии на кнопку загорается соответствующий светодиод, означающий выбранный способ передачи информации. На рисунке 34 приводится снимок, показывающий, какой из светодиодов должен быть выставлен на устройстве передачи по скрытому каналу.

В режиме передачи с помощью протокола UDP (способ №2) передача реализована с помощью протокола UDP. Для передачи используются поля заголовка "Длина датаграммы" и "Порт отправителя". Несмотря на то, что поле "Длина датаграммы" имеет размер 2 байта, для скрытой передачи используется только старший байт, так как в это поле не может быть нулевым и минимальное значение, которое оно должно иметь – 7 байт. Таким образом, в старший байт записывается один скрыто передаваемый байт. В соответствии с полученным

значением данного поля формируется случайная последовательность данных соответствующего размера, заполняющая поле "Данные". Поле "Порт отправителя" имеет размер 2 байта, но для того, чтобы избежать передачи нулевого байта в младшем байте (так как порт 0 зарезервирован), для передачи используется только старший байт.

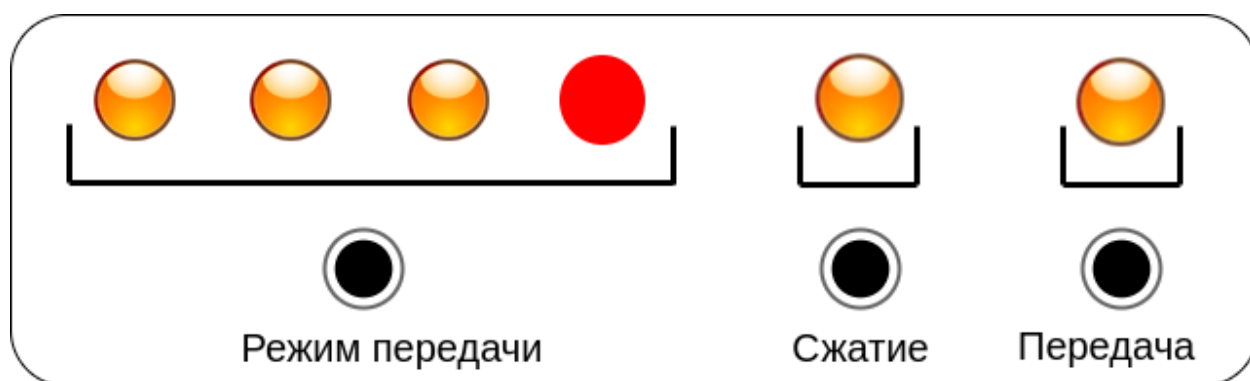


Рисунок 34 — Состояние аппаратного модуля при выбранном режиме скрытой передачи с помощью протокола UDP (вариант №2)

Режим сжатия данных предназначен для ускорения передачи по скрытому каналу включается по нажатии на кнопку "Сжатие". Для начала передачи необходимо нажать на кнопку "Передача" на модуле управления передающего устройства.

При нажатии на кнопку "Передача" начинается передача по выбранному скрытому каналу передачи. Во время передачи светодиод равномерно мигает. По окончании передачи светодиод гаснет.

Все пакеты, передаваемые конечными устройствами по сети зеркалируются на компьютер, подключенный через WiFi. На рисунке 35 приведен результат передачи файла с исходными текстами main.go.

Для фильтрации пакетов в программе Wireshark можно использовать следующие параметры:

- `udp.port` – параметр выбирающий те пакеты, в которых порт отправителя или получателя соответствует этому значению;
- `udp.srcport` – параметр, выбирающий те пакеты, в которых порт отправителя соответствует этому значению;

– `udp.dstport` – параметр, выбирающий те пакеты, в которых порт получателя соответствует этому значению.

11	0.374131233	127.0.0.2	127.0.0.1	UDP	29930	19971	→	40400	Len=29930
12	0.411372435	127.0.0.2	127.0.0.1	UDP	29482	29559	→	40400	Len=29440
13	0.447197594	127.0.0.2	127.0.0.1	UDP	26410	24860	→	40400	Len=26368
14	0.482582932	127.0.0.2	127.0.0.1	UDP	2602	26069	→	40400	Len=2560

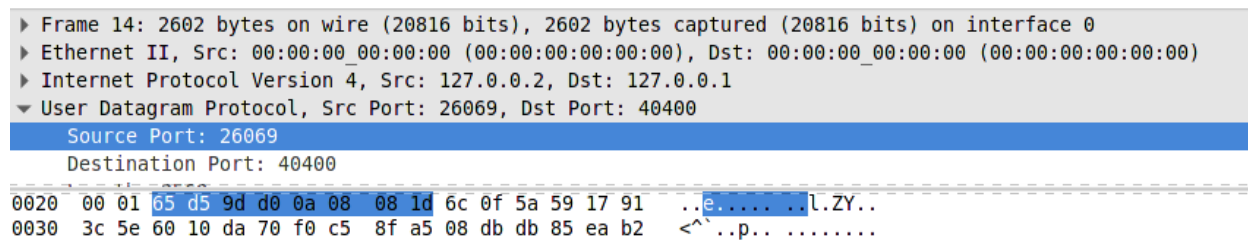


Рисунок 35 — Пример трафика, получаемого при скрытой передаче с помощью протокола UDP (вариант №2)

2.4.8 Выполнение приема скрыто передаваемой информации с помощью протокола ICMP

В подразделе 2.4.8 описывается методика по настройке режима приема скрыто передаваемой информации с помощью протокола ICMP и перехвату получаемого трафика. Настройка режима на устройстве, выполняющем передачу по скрытому каналу, выполняется с помощью модуля управления, подключенного к одноплатному компьютеру Raspberry Pi 3. Выбор режима выполняется посредством нажатия на кнопку "Режим приема". При нажатии на кнопку загорается соответствующий светодиод, означающий выбранный способ передачи информации. В данном случае возможно выбрать только один из реализованных режимов скрытой передачи информации. Переключение режимов посредством нажатия на кнопку "Режим передачи" осуществляется до тех пор, пока не будет выбран нужный режим. На рисунке 36 приводится рисунок, показывающий, какой из светодиодов должен быть выставлен на устройстве передачи по скрытому каналу.

Режим сжатия данных предназначен для ускорения передачи по скрытому каналу включается по нажатии на кнопку "Сжатие". Для начала передачи необходимо нажать на кнопку Прием" на модуле управления передающего устройства.

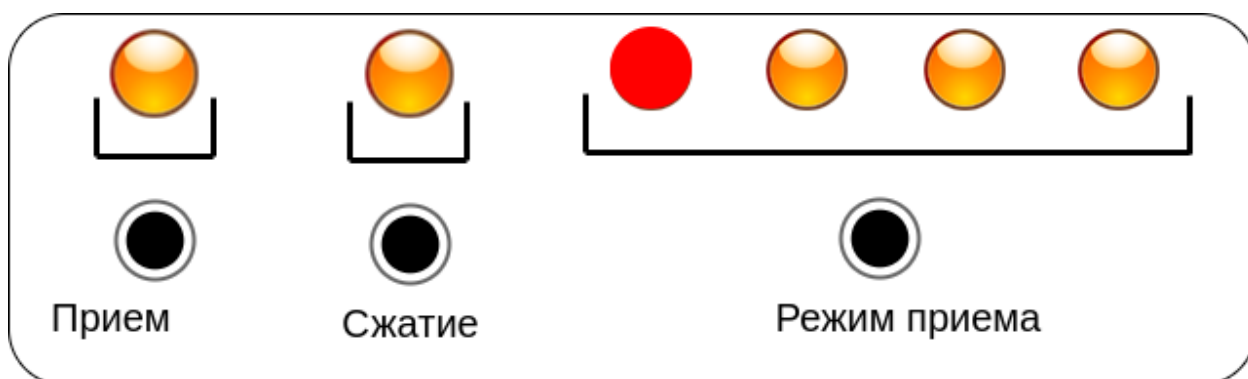


Рисунок 36 — Состояние аппаратного модуля при выбранном режиме приема скрыто передаваемой информации с помощью протокола ICMP

При нажатии на кнопку "Прием" начинается ожидание приема по выбранному скрытому каналу передачи. Во время ожидания приема светодиод равномерно мигает. По окончании приема светодиод гаснет. Для успешной скрытой передачи по данному каналу настройка параметров приема и начало приема должны выполняться в первую очередь.

Все пакеты, передаваемые конечными устройствами по сети зеркалируются на компьютер, подключенный через WiFi. На рисунке 37 приведен результат передачи файла с исходными текстами main.go, являющегося модулем разрабатываемой программной части, реализующей скрытую передачу.

- icmp.type – параметр, соответствующий полю "Тип" пакета ICMP;
- icmp.code – параметр, соответствующий полю "Код" пакета ICMP;
- icmp.ext.data – параметр, соответствующий полю "Данные" пакета ICMP.
- ip.addr – параметр, соответствующий полю "Адрес источника" или "Адрес получателя" IP-пакета;
- ip.src – параметр, соответствующий полю "Адрес источника";
- ip.dst – параметр, соответствующий полю "Адрес получателя".

70	0.708327286	192.168.1.10	192.168.1.1	ICMP	62	Echo (ping) request
71	0.708330872	192.168.1.10	192.168.1.1	ICMP	62	Echo (ping) request
72	0.708334549	192.168.1.10	192.168.1.1	ICMP	62	Echo (ping) request
73	0.708338209	192.168.1.10	192.168.1.1	ICMP	62	Echo (ping) request

Checksum: 0xaad8 [correct]
[Checksum Status: Good]
Identifier (BE): 4137 (0x1029)
Identifier (LE): 10512 (0x2910)
Sequence number (BE): 2 (0x0002)
Sequence number (LE): 512 (0x0200)

► [No response seen]

▼ Data (20 bytes)

Data: 65734461746122290a0970726f746f203d20666c
[Length: 20]

0000	84 c9 b2 a7 61 96 e8 03	9a f8 f0 ac 08 00 45 00	a... ..E.
0010	00 30 ce 53 40 00 40 01	e9 1d c0 a8 01 0a c0 a8	.0.S@.@.
0020	01 01 08 00 aa d8 10 29	00 02 65 73 44 61 74 61	) ..esData
0030	22 29 0a 09 70 72 6f 74	6f 20 3d 20 66 6c	")..prot o = fl

Рисунок 37 — Пример трафика, получаемого при скрытой передаче с помощью протокола ICMP

Передача по данному скрытому каналу осуществляется в ICMP-пакетах, содержащих Echo-запрос. Для просмотра фрагмента скрыто передаваемых данных необходимо выбрать ICMP-пакет, содержащий в себе Echo-запрос. Скрыто передаваемые данные передаются в поле "ICMP Payload".

2.4.9 Выполнение приема скрыто передаваемой информации с помощью протокола TCP

В подразделе 2.4.9 описывается методика по настройке режимов приема скрыто передаваемой информации и перехвату получаемого трафика. Настройка режима на устройстве, выполняющем передачу по скрытому каналу выполняется с помощью модуля управления, подключенного к одноплатному компьютеру Raspberry Pi 3. Выбор режима приема выполняется посредством нажатия на кнопку "Режим приема". При нажатии на кнопку загорается соответствующий светодиод, означающий выбранный способ приема скрыто передаваемой информации. На рисунке 38 приводится рисунок, показывающий, какой из светодиодов должен быть выставлен на устройстве передачи по скрытому каналу.

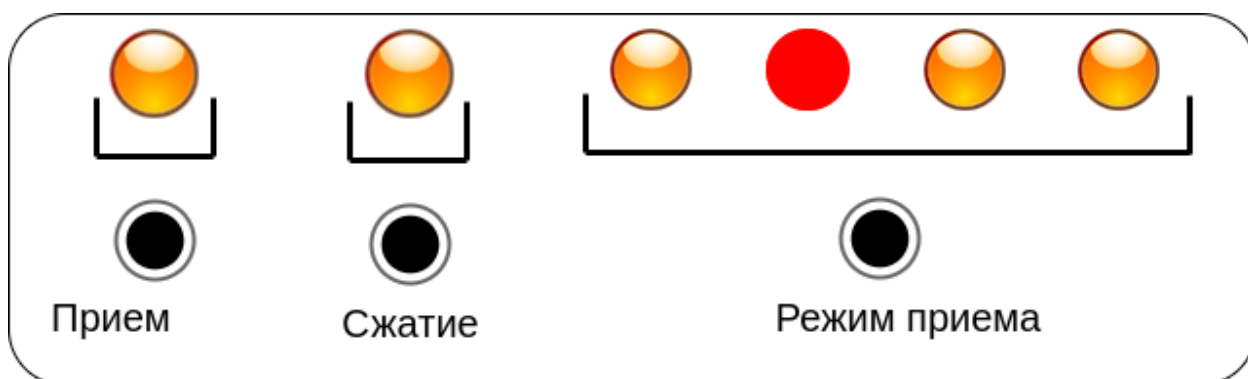


Рисунок 38 — Состояние аппаратного модуля при выбранном режиме скрыто передаваемой информации с помощью протокола TCP

Режим сжатия данных предназначен для ускорения передачи по скрытому каналу включается по нажатии на кнопку "Сжатие". Для начала ожидания приема информации по скрытому каналу необходимо нажать на кнопку "Прием" на модуле управления передающего устройства.

При нажатии на кнопку "Прием" начинается ожидание приема по выбранному скрытому каналу передачи. Во время приема светодиод равномерно мигает. По окончании приема светодиод гаснет. Для успешной скрытой передачи по данному каналу настройка параметров приема и начало приема должны выполняться в первую очередь.

Все пакеты, передаваемые конечными устройствами по сети зеркалируются на компьютер, подключенный через WiFi. На рисунке 39 приведен результат передачи файла с исходными текстами `main.go`.

Для фильтрации пакетов в программе Wireshark можно использовать следующие параметры:

- `tcp.port` – параметр выбирающий те пакеты, в которых порт отправителя или получателя соответствует этому значению;
- `tcp.srcport` – параметр, выбирающий те пакеты, в которых порт отправителя соответствует этому значению;
- `tcp.dstport` – параметр, выбирающий те пакеты, в которых порт получателя соответствует этому значению.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	127.0.0.1	127.0.0.1	TCP	74	53724 → 27116 [SYN] Seq=0 Win=43690 Len=0 MS
2	0.000011568	127.0.0.1	127.0.0.1	TCP	74	27116 → 53724 [SYN, ACK] Seq=0 Ack=1 Win=436
3	0.000019943	127.0.0.1	127.0.0.1	TCP	66	53724 → 27116 [ACK] Seq=1 Ack=1 Win=43776 Le
4	0.000054118	127.0.0.1	127.0.0.1	TCP	75	53724 → 27116 [PSH, ACK] Seq=1 Ack=1 Win=437
5	0.000056846	127.0.0.1	127.0.0.1	TCP	66	27116 → 53724 [ACK] Seq=1 Ack=10 Win=43776 l
6	0.000062852	127.0.0.1	127.0.0.1	TCP	66	53724 → 27116 [FIN, ACK] Seq=10 Ack=1 Win=43
7	0.000094922	127.0.0.1	127.0.0.1	TCP	74	60838 → 27101 [SYN] Seq=0 Win=43690 Len=0 MS
8	0.000099748	127.0.0.1	127.0.0.1	TCP	74	27101 → 60838 [SYN, ACK] Seq=0 Ack=1 Win=436
9	0.000102061	127.0.0.1	127.0.0.1	TCP	66	60838 → 27101 [ACK] Seq=1 Ack=1 Win=43776 Le
10	0.000111347	127.0.0.1	127.0.0.1	TCP	75	60838 → 27101 [PSH, ACK] Seq=1 Ack=1 Win=437
11	0.000112849	127.0.0.1	127.0.0.1	TCP	66	27101 → 60838 [ACK] Seq=1 Ack=10 Win=43776 l
12	0.000115811	127.0.0.1	127.0.0.1	TCP	66	60838 → 27101 [FIN, ACK] Seq=10 Ack=1 Win=43
13	0.000124014	127.0.0.1	127.0.0.1	TCP	74	53382 → 27120 [SYN] Seq=0 Win=43690 Len=0 MS
14	0.000127148	127.0.0.1	127.0.0.1	TCP	74	27120 → 53382 [SYN, ACK] Seq=0 Ack=1 Win=436
15	0.000129181	127.0.0.1	127.0.0.1	TCP	66	53382 → 27120 [ACK] Seq=1 Ack=1 Win=43776 Le

Рисунок 39 — Пример трафика, получаемого при скрытой передаче с помощью протокола TCP

2.4.10 Выполнение приема скрыто передаваемой информации с помощью протокола UDP (способ №1)

В подразделе 2.4.10 описывается методика по настройке режимов приема скрыто передаваемой информации с помощью протокола UDP и перехвату получаемого трафика. Настройка режима на устройстве, выполняющем передачу по скрытому каналу, выполняется с помощью модуля управления, подключенного к одноплатному компьютеру Raspberry Pi 3. Выбор режима выполняется посредством нажатия на кнопку "Режим приема". При нажатии на кнопку загорается соответствующий светодиод, означающий выбранный способ передачи информации. На рисунке 40 приводится снимок, показывающий, какой из светодиодов должен быть выставлен на устройстве передачи по скрытому каналу.

В режиме приема с помощью протокола UDP (способ №1) прием реализован с помощью протокола UDP и используется поле заголовка "Длина датаграммы". Несмотря на то, что это поле имеет размер 2 байта, для скрытой передачи используется только старший байт, так как в это поле не может быть нулевым и минимальное значение, которое оно должно иметь – 7 байт. Таким образом, в старший байт записывается один скрыто передаваемый байт. В соответствии с полученным значением данного поля формируется случайная последовательность данных соответствующего размера, заполняющая поле "Данные".

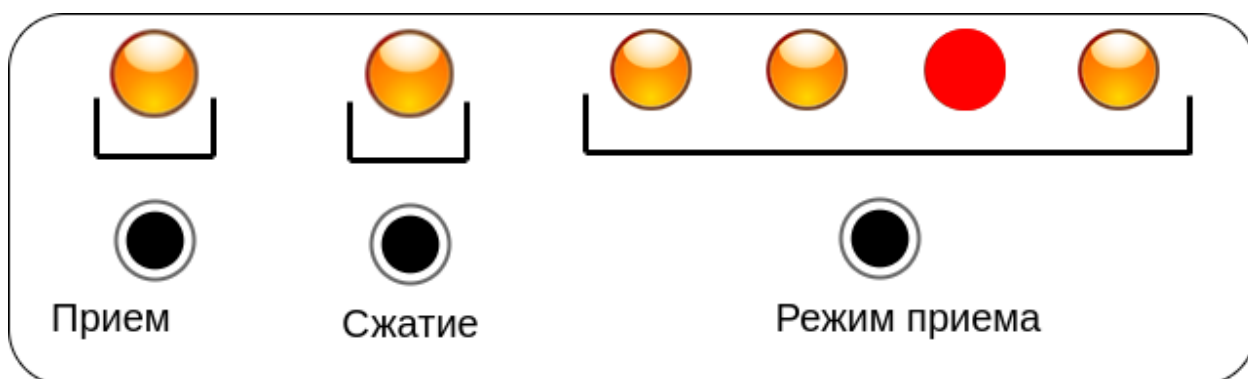


Рисунок 40 — Состояние аппаратного модуля при выбранном режиме приема с помощью протокола UDP

Режим сжатия данных предназначен для ускорения передачи по скрытому каналу включается по нажатии на кнопку "Сжатие". Для начала ожидания приема информации по скрытому каналу необходимо нажать на кнопку "Прием" на модуле управления передающего устройства.

При нажатии на кнопку "Прием" начинается ожидание приема по выбранному скрытому каналу передачи. Во время приема светодиод равномерно мигает. По окончании приема светодиод гаснет. Для успешной скрытой передачи по данному каналу настройка параметров приема и начало приема должны выполняться в первую очередь.

Все пакеты, передаваемые конечными устройствами по сети зеркалируются на компьютер, подключенный через WiFi. На рисунке 41 приведен результат передачи файла с исходными текстами main.go.

Для фильтрации пакетов в программе Wireshark можно использовать следующие параметры:

- `udp.port` – параметр выбирающий те пакеты, в которых порт отправителя или получателя соответствует этому значению;
- `udp.srcport` – параметр, выбирающий те пакеты, в которых порт отправителя соответствует этому значению;
- `udp.dstport` – параметр, выбирающий те пакеты, в которых порт получателя соответствует этому значению.

No.	Time	Source	Destination	Protocol	Length	Info
1601	13.56911973	192.168.1.121	192.168.1.5	UDP	60	20257 → 40400 Len=4
1618	13.61452553	192.168.1.121	192.168.1.5	UDP	1194	26751 → 40400 Len=24832
1635	13.65272423	192.168.1.121	192.168.1.5	UDP	1194	26751 → 40400 Len=24832
1652	13.69104972	192.168.1.121	192.168.1.5	UDP	1194	26751 → 40400 Len=24832
1669	13.72913978	192.168.1.121	192.168.1.5	UDP	1194	26751 → 40400 Len=24832
1686	13.76726821	192.168.1.121	192.168.1.5	UDP	1194	26751 → 40400 Len=24832
1703	13.80556273	192.168.1.121	192.168.1.5	UDP	1194	26751 → 40400 Len=24832

▶ Frame 1686: 1194 bytes on wire (9552 bits), 1194 bytes captured (9552 bits) on interface 0

▶ Ethernet II, Src: Raspberr_12:ac:1d (b8:27:eb:12:ac:1d), Dst: AsustekC_b7:fe:cf (38:d5:47:b7:fe:cf)

▶ Internet Protocol Version 4, Src: 192.168.1.121, Dst: 192.168.1.5

▼ User Datagram Protocol, Src Port: 26751, Dst Port: 40400

Source Port: 26751

Destination Port: 40400

Length: 24840

Checksum: 0x1a37 [unverified]

[Checksum Status: Unverified]

[Stream index: 1]

▼ Data (24832 bytes)

Data: e4f0caf3375a876d53b09be45b205404db1207ecb041c919...

[Length: 24832]

Рисунок 41 — Пример трафика, получаемого при скрытой передаче с помощью протокола UDP (вариант №1)

2.4.11 Выполнение приема скрыто передаваемой информации с помощью протокола UDP (способ №2)

В подразделе 2.4.11 описывается методика по настройке режимов приема скрыто передаваемой информации с помощью протокола UDP и перехвату получаемого трафика. Настройка режима на устройстве, выполняющем передачу по скрытому каналу, выполняется с помощью модуля управления, подключенного к одноплатному компьютеру Raspberry Pi 3. Выбор режима выполняется посредством нажатия на кнопку "Режим приема". При нажатии на кнопку загорается соответствующий светодиод, означающий выбранный способ передачи информации. На рисунке 42 приводится снимок, показывающий, какой из светодиодов должен быть выставлен на устройстве передачи по скрытому каналу.

В режиме приема с помощью протокола UDP (способ №2) прием реализован с помощью протокола UDP и используется поле заголовка "Длина датаграммы". Несмотря на то, что это поле имеет размер 2 байта, для скрытой передачи используется только старший байт, так как в это поле не может быть нулевым и минимальное значение, которое оно должно иметь – 7 байт. Таким образом, в старший байт записывается один скрыто передаваемый байт.

В соответствии с полученным значением данного поля формируется случайная последовательность данных соответствующего размера, заполняющая поле "Данные". Поле "Порт отправителя" имеет размер 2 байта, но для того, чтобы избежать передачи нулевого байта в младшем байте (так как порт 0 зарезервирован), для передачи используется только старший байт.

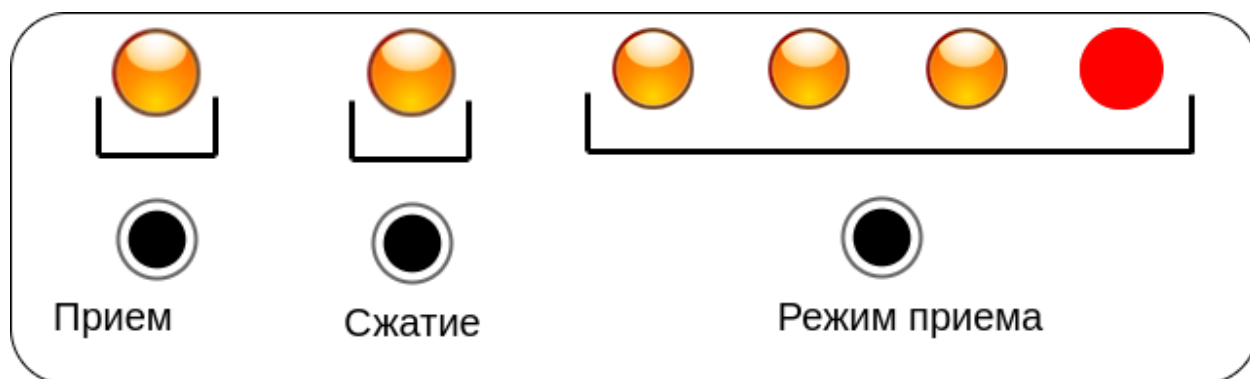


Рисунок 42 — Состояние аппаратного модуля при выбранном режиме приема с помощью протокола UDP (вариант №2)

Режим сжатия данных предназначен для ускорения передачи по скрытому каналу включается по нажатии на кнопку "Сжатие". Для начала ожидания приема информации по скрытому каналу необходимо нажать на кнопку "Прием" на модуле управления передающего устройства.

При нажатии на кнопку "Прием" начинается ожидание приема по выбранному скрытому каналу передачи. Во время приема светодиод равномерно мигает. По окончании приема светодиод гаснет. Для успешной скрытой передачи по данному каналу настройка параметров приема и начало приема должны выполняться в первую очередь.

Все пакеты, передаваемые конечными устройствами по сети зеркалируются на компьютер, подключенный через WiFi. На рисунке 43 приведен результат передачи файла с исходными текстами main.go.

Для фильтрации пакетов в программе Wireshark можно использовать следующие параметры:

- `udp.port` – параметр выбирающий те пакеты, в которых порт отправителя или получателя соответствует этому значению;

– `udp.srcport` – параметр, выбирающий те пакеты, в которых порт отправителя соответствует этому значению;

– `udp.dstport` – параметр, выбирающий те пакеты, в которых порт получателя соответствует этому значению.

11	0.374131233	127.0.0.2	127.0.0.1	UDP	29338	19371	→	40400	Len=29338
12	0.411372435	127.0.0.2	127.0.0.1	UDP	29482	29559	→	40400	Len=29440
13	0.447197594	127.0.0.2	127.0.0.1	UDP	26410	24860	→	40400	Len=26368
14	0.482582932	127.0.0.2	127.0.0.1	UDP	2602	26069	→	40400	Len=2560

```

▶ Frame 14: 2602 bytes on wire (20816 bits), 2602 bytes captured (20816 bits) on interface 0
▶ Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
▶ Internet Protocol Version 4, Src: 127.0.0.2, Dst: 127.0.0.1
▼ User Datagram Protocol, Src Port: 26069, Dst Port: 40400
  Source Port: 26069
  Destination Port: 40400
0020  00 01 65 d5 9d d0 0a 08 08 1d 6c 0f 5a 59 17 91  ..e.....l.ZY..
0030  3c 5e 60 10 da 70 f0 c5 8f a5 08 db db 85 ea b2  <^..p.. .....
.....

```

Рисунок 43 — Пример трафика, получаемого при скрытой передаче с помощью протокола UDP (вариант №2)

ЗАКЛЮЧЕНИЕ

В рамках дипломного проекта разработан аппаратно-программный комплекс для исследования способов скрытой передачи информации в компьютерных сетях. Указанный комплекс обеспечивает возможность ознакомления студентов с таким понятием, как скрытые каналы передачи информации и обеспечивает возможность выполнения лабораторных работ и практикумов по таким дисциплинам, как "Безопасность сетей ЭВМ" и "Сети и системы передачи информации", изучаемых на кафедре ИБСТ Пензенского государственного университета.

В исследовательском разделе приведено определение стандартных протоколов межсетевого взаимодействия сетевого и транспортного уровней модели ТСР/ІР. Рассмотрены особенности данных протоколов и каким образом можно организовать с их помощью скрытые каналы передачи информации. Также рассмотрены классификация и способы организации скрытых каналов передачи информации.

В практическом разделе описаны требования к разрабатываемому аппаратно-программному комплексу. Рассмотрена разработка аппаратно-программного комплекса: процесс выбора аппаратной части комплекса, процесс разработки программной части комплекса. Приведены результаты проверки корректности работы реализованных скрытых каналов передачи информации, а также методика по работе с разработанным аппаратно-программным комплексом для исследования способов скрытой передачи информации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1 Криптографические методы защиты информации // Е.А. Новиков, Ю.А. Шитов, 2008 – С. 5
- 2 Computer networks // Andrew S. Tanenbaum, David J. Wetherall, 2011 – С. 41
- 3 Techniques of network steganography and covert channels // James P. Black, 2013
- 4 Применение сетевой стеганографии для защиты данных, передаваемых по открытым каналам Интернет // О.Ю. Пескова, Г. Ю. Халабурда, МПСС (Санкт-Петербург), 2012 – С. 348
- 5 Department of Defense Trusted Computer System Evaluation Criteria // U.S. DOD, DOD 5200.28-STD, 1983
- 6 Raspberry Pi 3 Model B [Электронный ресурс]. – Режим доступа: <https://www.raspberrypi.org/products/raspberry-pi-3-model-b/>
- 7 Go Programming For Beginners [Электронный ресурс]. – Режим доступа: <https://thenewstack.io/the-new-stack-intros-go-programming-for-beginners>
- 8 Learn Python – Programming Tutorials For Beginners [Электронный ресурс]. – Режим доступа: <https://www.programiz.com/python-programming>
- 9 Packet capture library (libpcap) [Электронный ресурс]. – Режим доступа: <https://wiki.wireshark.org/libpcap>
- 10 Wireshark User's Guide [Электронный ресурс]. – Режим доступа: https://www.wireshark.org/docs/wsug_html_chunked/index.html
- 11 A tcpdump Tutorial and Primer with Examples [Электронный ресурс]. – Режим доступа: <https://danielmiessler.com/study/tcpdump>
- 12 gpiozero – Gpiozero 1.3.2 documentation [Электронный ресурс]. – Режим доступа: <https://gpiozero.readthedocs.io/en/stable>

Приложение А
(обязательное)

Исходный код программной части комплекса для выполнения скрытой
передачи информации с помощью протокола ISMP

```

func (l *ICMPEcho) Transmit(data []byte, destIP string) error {

c, err := icmp.ListenPacket("ip4:icmp", "0.0.0.0")
    if err != nil {
        return err
    }
    defer c.Close()
    //send data size first
    {
        bs := make([]byte, 4)
        dataSize := len(data)
        if dataSize > 50000 {
            return errors.New("upload size exceeded (max 50
                kB)")
        }
        binary.LittleEndian.PutUint32(bs, uint32(len(data)))
        wm := icmp.Message{
            Type: ipv4.ICMPTypeEcho,
            Code: 0,
            Body: &icmp.Echo{
                ID: os.Getpid() & 0xffff, Seq: 2,
                Data: bs,
            },
        }
        wb, err := wm.Marshal(nil)
        if err != nil {
            return err
        }
        log.Println(wb)
        if _, err := c.WriteTo(wb, &net.IPAddr{IP: net.ParseIP(
            destIP)}); err != nil {
            return err
        }
    }
}

var blocksize int
if len(data) < 20 {
    blocksize = len(data)
}

```

```

} else {
    blocksize = 20
}

for i := 0; i < len(data); i = i + blocksize {
    wm := icmp.Message{
        Type: ipv4.ICMPTypeEcho,
        Code: 0,
        Body: &icmp.Echo{
            ID: os.Getpid() & 0xffff, Seq: 2,
            Data: data[i : i+blocksize],
        },
    }
    wb, err := wm.Marshal(nil)
    if err != nil {
        return err
    }
    if _, err := c.WriteTo(wb, &net.IPAddr{IP: net.ParseIP(
        destIP)}); err != nil {
        time.Sleep(time.Second)
        i = i - blocksize
        continue
    }
}

if i := len(data) % blocksize; i != 0 {
    wm := icmp.Message{
        Type: ipv4.ICMPTypeEchoReply,
        Code: 0,
        Body: &icmp.Echo{
            ID: os.Getpid() & 0xffff, Seq: 2,
            Data: data[len(data)-i:],
        },
    }
    wb, err := wm.Marshal(nil)
    if err != nil {
        return err
    }
}

```

```

        if _, err := c.WriteTo(wb, &net.IPAddr{IP: net.ParseIP(
            destIP)}); err != nil {
            return err
        }
    }
    return nil
}

func (l *ICMPEcho) Receive() (result []byte, err error) {
    c, err := icmp.ListenPacket("ip4:icmp", "0.0.0.0")
    c.SetReadDeadline(time.Now().Add(5 * time.Minute))
    if err != nil {
        return []byte{}, err
    }
    defer c.Close()
    rb := make([]byte, 1500)
    //first packet contains size of transmitting data in bytes
    n, _, err := c.ReadFrom(rb)
    if err != nil {
        return []byte{}, err
    }
    rm, err := icmp.ParseMessage(1, rb[:n])
    if err != nil {
        return []byte{}, err
    }
    size, _ := rm.Body.Marshal(1)
    dataSize := int(binary.LittleEndian.Uint32(size[4:]))
    log.Println("dataSize: ", dataSize)
    if dataSize > 1024*20 {
        return []byte{}, errors.New("upload size exceeded (max
            20KiB)")
    }
    //data upload
    var i int
    for i < dataSize {
        n, _, err = c.ReadFrom(rb)
        if err != nil {
            return []byte{}, err
        }
    }
}

```

```

    }

    rm, err = icmp.ParseMessage(1, rb[:n])
    if err != nil {
        return []byte{}, err
    }
    tmp, _ := rm.Body.Marshal(1)
    tmp = tmp[4:]
    result = bytes.Join([][]byte{result, tmp}, []byte{})
    i = i + len(tmp)
}
return result, nil
}

```

Приложение Б
(обязательное)

Исходный код программной части комплекса для выполнения скрытой
передачи информации с помощью протокола TSP

```

func (d *TCPDict) Transmit(data, noise []byte, destIP string) error {
    dstaddrs, err := net.LookupIP(destIP)
    if err != nil {
        log.Fatal(err)
    }
    if len(d.Dict) == 0 {
        for i := 0; i < cap(d.Dict); i++ {
            d.Dict[i] = uint16(i) + 27000
        }
    }
    data = bytes.Join([][]byte{data, []byte{0x55, 0xaa}}, []byte{})
    for _, value := range data {
        tmp, err := net.Dial("tcp4", dstaddrs[0].String()+":"+
            strconv.Itoa(int(d.Dict[value])))
        if err != nil {
            return err
        }
        _, err = tmp.Write(noise)
        if err != nil {
            return err
        }
        tmp.Close()
    }
    return nil
}

func (d *TCPDict) Receive() ([]byte, error) {
    var listens []net.Listener
    var result []byte
    resBytes := make(chan byte)

    for _, i := range d.Dict {
        tmp, err := net.Listen("tcp4", ":"+strconv.Itoa(int(i))
        )
        defer tmp.Close()
        if err != nil {
            return []byte{}, err
        }
    }

```

```

        listens = append(listens, tmp)
    }

    for key, val := range listens {
        go handleConnection(val, key, resBytes)
    }

    for {
        result = append(result, <-resBytes)
        tmpLen := len(result)
        if tmpLen >= 2 && bytes.Equal([]byte{0x55, 0xaa},
            result[tmpLen-2:]) {
            for _, val := range listens {
                val.Close()
            }
            return result[:tmpLen-2], nil
        }
    }
}

func handleConnection(c net.Listener, port int, ch chan byte) {
    for {
        c.Accept()
        ch <- byte(port)
    }
}

```

Приложение В
(обязательное)

Исходный код программной части комплекса для выполнения скрытой
передачи информации с помощью протокола UDP (способ №1)

```

func (l *UDPLen) Transmit(data []byte, srcIP, dstIP net.IP, port *int
) error {
    destAddr, err := net.ResolveUDPAddr("udp4", strings.Join([]
        string{dstIP.String(), strconv.Itoa(*port)}, ":"))
    if err != nil {
        return err
    }
    sourceAddr, err := net.ResolveUDPAddr("udp4", strings.Join([]
        string{srcIP.String(), "0"}, ":"))
    if err != nil {
        return err
    }
    c, err := net.DialUDP("udp4", sourceAddr, destAddr)
    if err != nil {
        return err
    }
    defer c.Close()
    s := rand.NewSource(time.Now().UnixNano())
    r := rand.New(s)
    commonBuf := make([]byte, 0xff00)
    for i := 0; i < 0xff; i++ {
        commonBuf[i] = byte(r.Intn(0x100))
    }
    bs := make([]byte, 4)
    binary.LittleEndian.PutUint32(bs, uint32(len(data)))
    _, err = c.Write(bs)
    if err != nil {
        return err
    }
    for _, val := range data {
        size := uint(val)
        size = size << 8
        tmpBuf := commonBuf[0:size]
    tmpBuf = func(b []byte) []byte {
        tbuf := make([]byte, len(b))
        for i, v := range b {
            tbuf[i] = v ^ byte(r.Intn(0x100))
        }
    }

```

```

        return tbuf
    }(tmpBuf)
    _, err := c.Write(tmpBuf)
    if err != nil {
        return err
    }
    time.Sleep(time.Millisecond * 5)
}
return nil
}

func (l *UDPLen) Receive(port *int) ([]byte, error) {
    destAddr, err := net.ResolveUDPAddr("udp4", ":"+strconv.Itoa(
        int(*port)))
    if err != nil {
        return []byte{}, err
    }
    c, err := net.ListenUDP("udp4", destAddr)
    if err != nil {
        return []byte{}, err
    }
    defer c.Close()
    bs := make([]byte, 4)
    _, _, err = c.ReadFromUDP(bs)
    buf := make([]byte, binary.LittleEndian.Uint32(bs))
    commonBuf := make([]byte, 0xff00)
    for i := 0; i < len(buf); i++ {
        n, _, err := c.ReadFromUDP(commonBuf)
        if err != nil {
            return []byte{}, err
        }
        buf[i] = byte(n >> 8)
    }
    return buf, nil
}

```

Приложение Г
(обязательное)

Исходный код программной части комплекса для выполнения скрытой
передачи информации с помощью протокола UDP (способ №2)

```

func (l *UDPsPortLen) Transmit(data []byte, srcIP, dstIP net.IP, port
    *int) error {
    var (
        sourceAddr, destAddr *net.UDPAddr
    )
    conns := make([]*net.UDPConn, 0xff)
    destAddr, err := net.ResolveUDPAddr("udp4", strings.Join([]
        string{dstIP.String(), strconv.Itoa(*port)}, ":"))
    if err != nil {
        return err
    }
    sourceAddr, err = net.ResolveUDPAddr("udp4", strings.Join([]
        string{srcIP.String(), "0"}, ":"))
    if err != nil {
        return err
    }
    s := rand.NewSource(time.Now().UnixNano())
    r := rand.New(s)
    commonBuf := make([]byte, 0xff00)
    for i := 0; i < 0xff; i++ {
        commonBuf[i] = byte(r.Intn(0x100))
    }
    for i := range conns {
        sourceAddr, err = net.ResolveUDPAddr("udp4", strings.
            Join([]string{srcIP.String(), strconv.Itoa((i << 8)
                | r.Intn(0xff))}, ":"))
        conns[i], err = net.DialUDP("udp4", sourceAddr,
            destAddr)
        if err != nil {
            return err
        }
        defer conns[i].Close()
    }

    bs := make([]byte, 4)
    if len(data)%2 != 0 {
        data = append(data, byte(0x00))
    }

```

```

log.Println(data)
binary.LittleEndian.PutUint32(bs, uint32(len(data)))
if _, err = conns[r.Intn(0x100)].Write(bs); err != nil {
    return err
}
time.Sleep(40 * time.Millisecond)
for i := 0; i < len(data); i = i + 2 {
    size := uint(data[i+1])
    size = size << 8
    tmpBuf := commonBuf[0:size]
    tmpBuf = func(b []byte) []byte {
        tbuf := make([]byte, len(b))
        for j, v := range b {
            tbuf[j] = v ^ byte(r.Intn(0x100))
        }
        return tbuf
    }(tmpBuf)
    _, err := conns[data[i]].Write(tmpBuf)
    if err != nil {
        return err
    }
    time.Sleep(35 * time.Millisecond)
}
return nil
}

func (l *UDPsPortLen) Receive(destIP string, port *int) ([]byte, error)
{
    destAddr, err := net.ResolveUDPAddr("udp4", ":"+strconv.Itoa(
        int(*port)))
    if err != nil {
        return []byte{}, err
    }
    c, err := net.ListenUDP("udp4", destAddr)
    if err != nil {
        return []byte{}, err
    }
    defer c.Close()

```

```

dstaddrs, err := net.LookupIP(destIP)
if err != nil {
    log.Fatal(err.Error())
}
dstIP := dstaddrs[0].To4()
var workInterface net.Interface
if ifaces, err := net.Interfaces(); err != nil {
    log.Fatal(err.Error())
} else {
    for _, j := range ifaces {
        addrs, _ := j.Addrs()
        for _, k := range addrs {
            if strings.Split(k.String(), "/")[0] ==
                dstIP.String() {
                workInterface = j
                break
            }
        }
    }
}

bs := make([]byte, 4)
if _, _, err := c.ReadFromUDP(bs); err != nil {
    log.Fatal(err)
}
go fakeUDPReceive(c)
buf, err := receiveBytes(workInterface.Name, binary.
    LittleEndian.Uint32(bs), port)
if err != nil {
    log.Fatal(err.Error())
}
log.Println(buf)

return buf, nil
}

func fakeUDPReceive(c *net.UDPConn) {
    commonBuf := make([]byte, 0xff00)
    for {

```

```

        if _, _, err := c.ReadFromUDP(commonBuf); err != nil {
            log.Fatal(err.Error())
        }
    }
}

func receiveBytes(ifName string, bufLen uint32, port *int) (buf []byte,
err error) {
    buf = make([]byte, 0)
    handle, err := pcap.OpenLive(ifName, 500, false, pcap.
        BlockForever)
    if err != nil {
        return []byte{}, err
    }
    defer handle.Close()
    packetSource := gopacket.NewPacketSource(handle, handle.
        LinkType())
    handle.SetBPFFilter("udp and dst port 40400")
    for packet := range packetSource.Packets() {
        log.Println(len(buf), byte(packet.NetworkLayer().
            LayerPayload()[0]),
            byte(packet.NetworkLayer().LayerPayload()[4]))
        buf = append(buf, byte(packet.NetworkLayer().
            LayerPayload()[0]),
            byte(packet.NetworkLayer().LayerPayload()[4]))
        if uint32(len(buf)) >= bufLen {
            break
        }
    }
    return buf, err
}

```