

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ 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 10.05.03 –

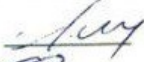
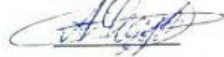
Специализация –

Информационная безопасность
автоматизированных систем
Защищенные автоматизированные
системы управления

ДИПЛОМНЫЙ ПРОЕКТ

на тему

ЗАЩИЩЕННЫЙ DNS-КЛИЕНТ

Студент		<u>Костюченков Станислав Андреевич</u>
Руководитель		<u>Липилин О. В.</u>
Нормоконтролёр		<u>Фатеев А. Г.</u>
Рецензент начальник группы АО «ПНИЭИ»		<u>Федулаев А. В.</u>

Работа допущена к защите (протокол заседания кафедры от 21.06.19 № 16)
Заведующий кафедрой  С. Л. Зефилов

Работа защищена с отметкой отлично
(протокол заседания ГЭК
от 25.06.19 № 22)
Секретарь ГЭК  А. П. Иванов

Пенза 2019

Проверка ВКР – Юбачев
 С. Л. Зефилов

Оглавление

Введение.....	4
1 Система доменных имен.....	8
2 Оценка рисков информационной безопасности DNS-клиента.....	12
2.1 Определение ценности активов.....	12
2.2 Идентификация угроз информационной безопасности.....	13
2.3 Идентификация уязвимостей.....	19
2.4 Идентификация сценариев инцидентов.....	21
2.5 Оценивание рисков DNS-клиента.....	23
3 Выбор способа обработки рисков.....	30
4 Выбор защитных мер.....	33
5 Разработка архитектуры DNS-клиента.....	42
6 Программная реализация функции DNS-запроса для прикладных программ.....	52
7 Программная реализация защищенного DNS-клиента.....	61
7.1 Реализация главной функции.....	61
7.2 Реализация функции подключения к именованному каналу.....	68
7.3 Реализация функции принятия DNS-запроса от программы.....	69
7.4 Реализация функции обработки принятого DNS-ответа.....	88
7.5 Реализация функции журналирования DNS-клиента.....	117
8 Проверка работоспособности DNS-клиента.....	120
9 Разработка руководства пользователя DNS-клиента.....	127
10 Вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие.....	128
Заключение.....	130
Список используемых источников.....	131
Приложение А Диаграмма вариантов использования.....	133

Приложение Б Руководство пользователя DNS-клиента.....	135
--	-----

Введение

DNS служба является критичным и важным компонентом сети Интернет. Она предназначена для преобразования доменных имен, удобных для пользователей, в IP-адреса и представляет собой иерархически организованную распределенную базу данных.

Центральная роль службы DNS делает ее, с одной стороны, желанной целью атакующего, поскольку нарушение работы DNS наносит огромный ущерб работе сети, а с другой мощным средством для проведения атак на другие сетевые программные средства. При разработке DNS протокола в 1980-х годах безопасность не была приоритетом.

Актуальность данной темы обусловлена, тем что система DNS является мощным средством для проведения атак на другое системное и прикладное ПО, и именно DNS-клиент принимает решение, какие DNS-ответы принять, а какие отбросить и не передавать прикладным программам, значит особое внимание безопасности необходимо уделить именно DNS-клиенту. Кроме того, в квалификационной работе будут реализованы защитные механизмы, которые отсутствуют в стандартном DNS-клиенте в ОС Windows 10, а именно:

- использование протокола TSIG;
- выбор случайного DNS-сервера;
- использование букв разного регистра в доменном имени при запросе;
- подпись файла hosts, содержащего локальную базу данных доменных имен, с помощью вычисления хеш-суммы с ключом;
- аутентификация DNS-клиента и DNS-сервера с помощью DNS-Cookies.

Целью дипломного проекта является разработка защищенного DNS-клиента. Защищенный DNS-клиент предназначен для безопасного выполнения DNS-запросов к DNS-серверам и получения DNS-ответов на них.

Требования для защищенного DNS-клиента следующие:

- DNS-клиент должен функционировать в операционной системе Windows 10;
- DNS-клиент должен поддерживать типы DNS-записей, описанные в RFC 1035 «Domain names – implementation and specification»:
 - узел IPv4 (тип A);
 - узел IPv6 (тип AAAA);
 - псевдоним (тип CNAME);
 - ответственный DNS-сервер (тип NS);
 - расположение службы (тип SRV);
 - почтовый сервер (тип MX);
 - указатель (тип PTR);
 - текст (тип TXT);
- DNS-клиент должен поддерживать тип DNS-записи OPT, описанный в RFC 1035 «Extension Mechanisms for DNS (EDNS(0))»;
- DNS-клиент должен поддерживать тип DNS-записи TSIG, описанный в RFC 2845 «Secret Key Transaction Authentication for DNS»;
- DNS-клиент должен поддерживать отправку DNS-запросов с использованием протоколов транспортного уровня TCP и UDP;
- в DNS-клиенте должны быть реализованы следующие защитные механизмы из RFC 5452 «Measures for Making DNS More Resilient against Forged Answers»:
 - DNS-клиент должен отклонить DNS-ответ, если секции «запрос» отправленного и принятого DNS-пакета не совпадают;
 - DNS-клиент должен отклонить DNS-ответ, если идентификатор DNS-ответа не совпадает с идентификатором DNS-запроса;
 - DNS-клиент должен отклонить DNS-ответ, если IP-адрес и порт DNS-сервера не совпадают в DNS-запросе и DNS-ответе;

- DNS-клиент должен отправлять DNS-запросы со случайного порта;
- идентификатор DNS-запроса должен быть случайным;
- при получении трёх некорректных DNS-ответов, инкапсулированных в UDP-датаграммы, DNS-клиент должен отправить повторно DNS-запрос по протоколу TCP;
- DNS-клиент должен игнорировать принятые DNS-ответы, которые содержат доменные имена, не относящиеся к доменному имени из секции «запрос»;
- в DNS-клиенте должно быть реализовано использование случайного регистра букв в имени домена при DNS-запросе по RFC «Use of Bit 0x20 in DNS Labels to Improve Transaction Identity»;
- в DNS-клиенте должен быть реализован механизм случайного выбора DNS-сервера;
- DNS-клиент должен иметь возможность подписывать файл hosts, содержащий локальную базу данных доменных имен;
- DNS-клиент должен поддерживать протокол TSIG по RFC 2845 «Secret Key Transaction Authentication for DNS» для проверки подлинности DNS-сервера и целостности DNS-ответов;
- DNS-клиент должен поддерживать механизм DNS Cookie по RFC 7873 «Domain Name System (DNS) Cookies» для аутентификации DNS-клиента и DNS-сервера;
- в DNS-клиенте должно быть реализовано ведение журнала аудита DNS-клиента.

В ходе дипломного проектирования будут рассмотрены следующие вопросы:

- описание DNS-клиента;
- описание угроз информационной безопасности DNS-клиента;
- описание уязвимостей DNS-клиента;

- оценка рисков информационной безопасности DNS-клиента;
- описание выбранных защитных мер DNS-клиента;
- разработка архитектуры DNS-клиента;
- реализация защищенного DNS-клиента;
- проверка работоспособности DNS-клиента;
- обзор вредных психофизиологических факторов, влияющих на психоэмоциональное состояние специалиста по защите информации и методов и средств физической культуры, снижающие их воздействие;
- разработка руководства пользователя DNS-клиента.

Служба DNS имеет иерархическую структуру. Иерархию образуют DNS-серверы, которые поддерживают распределенную базу отображений, а DNS-клиенты обращаются к серверам с запросами об отображении доменного имени на IP-адрес. Определение системы DNS дано в RFC 1034 «Domain Names – concepts and facilities»[1], RFC 1035 «Domain names – implementation and specification»[2].

DNS-клиентом является программное обеспечение, обеспечивающее получение ресурсных DNS-записей от DNS-сервера. DNS-клиент установлен на каждом узле Интернета, будь то клиентский компьютер, сервер приложений или маршрутизатор.

Запросы к DNS-серверам и их ответы обслуживаются протоколом DNS, что позволяет клиенту делать запросы относительно некоторого доменного имени, либо задавая тип записи, либо запрашивая все типы, относящиеся к данному имени. DNS-сообщения в открытом виде чаще всего передаются в дейтаграммах UDP с портом сервера 53, но в некоторых случаях, требующих повышенной надежности, служба DNS обращается по TCP.

Разрешение доменного имени - это процесс поиска служебной информации по доменному имени в системе доменных имен.

DNS-записи - это соответствие доменного имени служебной информации.

В таблице 1 показано описание типов записей, которые должны поддерживаться разрабатываемым DNS-клиентом.

Таблица 1 - Описание типов DNS-записей

Тип	Название	Описание
A	Узел IPv4	Соответствие доменного имени и адреса IPv4.
AAAA	Узел IPv6	Соответствие доменного имени и адреса IPv6.
CNAME	Псевдоним	Соответствие псевдонима и доменного имени.

Продолжение таблицы 1

Тип	Название	Описание
NS	Ответственный DNS-сервер	Имя DNS-сервера, ответственного за доменную зону.
SRV	Расположение службы	Содержит информацию об использовании на компьютерах определенных служб.
MX	Почтовый сервер	Содержит информацию о доменном имени почтового сервера.
PTR	Указатель	Соответствие адреса IPv4 или IPv6 доменному имени.
TXT	Текст	Запись может содержать любой текст.
OPT	Опции	Используется в протоколе EDNS0.
TSIG	Подпись транзакции	Используется для подписи DNS-записей по протоколу TSIG.

Для разрешения доменного имени прикладная программа обращается к DNS-клиенту, передавая ему доменное имя и тип записи в качестве параметров.

DNS-клиент посылает запрос DNS-серверу, который ищет имя и возвращает соответствующий IP-адрес DNS-клиенту, который, в свою очередь, передает этот адрес вызвавшей его прикладной программе. Имея IP-адрес, программа может установить TCP-соединение с адресатом или послать ему UDP-пакеты.

Формат DNS-сообщения показан на рисунке 1. На рисунке 2 показан формат запроса секции запросов. На рисунке 3 показан общий формат DNS-записи.

Принцип работы DNS показан на примере определения IP-адреса доменного имени rnzgu.ru. Принцип работы DNS показан на рисунке 4.

Сначала компьютер отправляет DNS-запрос с типом A DNS-серверу провайдера или домашнему DNS-серверу.

0	15	16	31
Идентификатор		Флаги	
Количество запросов		Количество записей ответов	
Количество записей с серверами имен		Количество записей дополнительных ответов	
Секция запросов			
Секция записей ответов			
Секция записей с серверами имен			
Секция записей дополнительных ответов			

Рисунок 1 - Формат DNS-сообщения

Имя домена (размер от 1 до 256 байт включительно)	
Тип запроса (2 байта)	Класс запроса (2 байта)

Рисунок 2 - Формат запроса

Имя домена (1 до 256 байт включительно)	
Тип записи (2 байта)	Класс записи (2 байта)
Время жизни (4 байта)	
Длина поля данных (2 байта)	
Данные записи	

Рисунок 3 - Формат DNS-записи

Далее DNS-сервер провайдера начинает отправлять DNS-запросы другим DNS-серверам в порядке убывания зон в имени. Для определения DNS-сервера, который хранит информацию о домене «ru», отправляется DNS-запрос одному из корневых серверов DNS. DNS-серверы возвращают в ответе записи типа NS

и А. Корневых серверов DNS всего 13. Далее узнается IP-адрес сервера, который хранит информацию о зоне pnzgu.ru. И наконец узнается IP-адрес самого домена «pnzgu.ru».

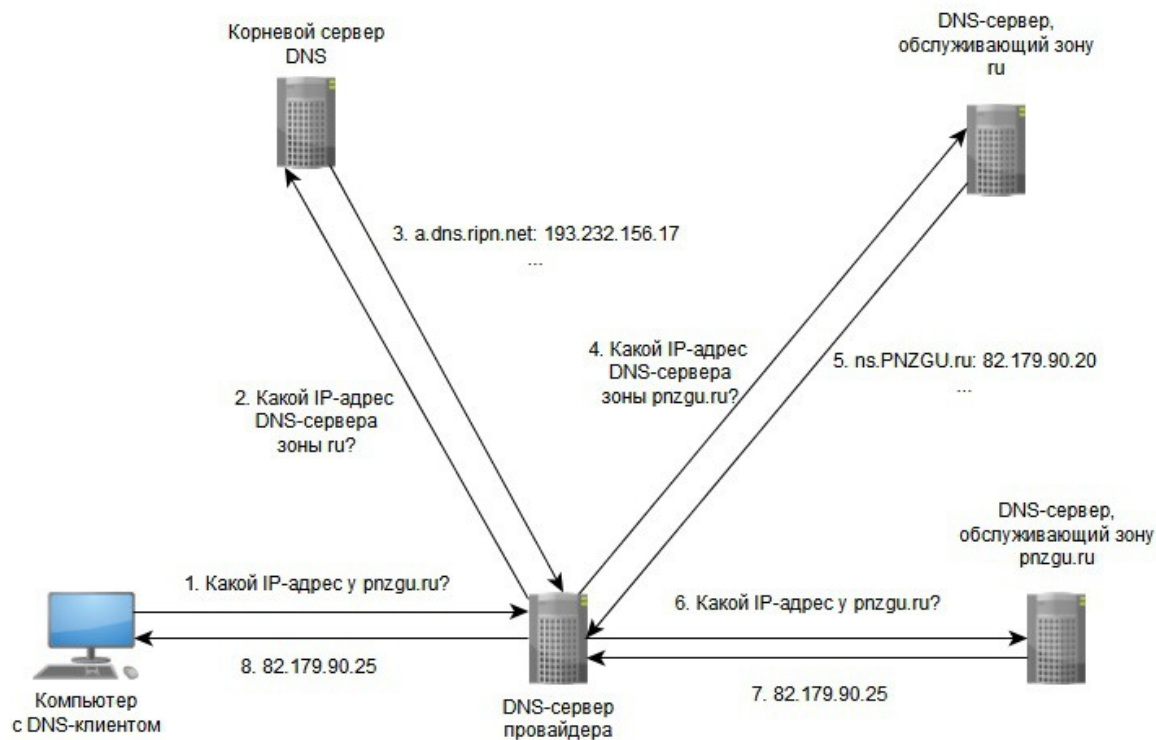


Рисунок 4 - Принцип работы DNS

Таким образом, истинное назначение системы DNS заключается в преобразовании доменных имен в записи ресурсов.

2 Оценка рисков информационной безопасности DNS-клиента

2.1 Определение ценности активов

Были идентифицированы вспомогательные и информационные активы DNS-клиента. Экспертным путем была определена ценность каждого актива. Результаты идентификации и определения ценности активов приведены в таблице 2. «ВА» означает вспомогательный актив, «ИА» - информационный актив

Таблица 2 - Результаты идентификации и определения ценности активов

№	Актив	Тип актива	Ценность
1	DNS-клиент	ВА	В
2	DNS-записи	ИА	В
3	Файл hosts, содержащий локальную базу данных доменных имен	ИА	В

Были определены значимые свойства ИБ для каждого информационного актива. Знак «+» означает, что свойство ИБ значимо для данного актива. Знак «-» означает, что свойство ИБ не значимо для данного актива. «Аут.» - аутентичность. Результаты идентификации, определения ценности и значимых свойств ИБ приведены в таблице 3.

Таблица 3 - Значимые свойства ИБ информационных активов

№	Информационный актив	Значимые свойства ИБ			
		К	Д	Ц	Аут.
1	DNS-записи	-	+	+	+
2	Файл hosts, содержащий локальную базу данных доменных имен	-	+	+	+

2.2 Идентификация угроз информационной безопасности

Были идентифицированы угрозы для DNS-клиента.

Следующие угрозы были взяты из Банка данных угроз безопасности информации ФСТЭК[3]:

- угроза внедрения кода или данных;
- угроза использования слабостей протоколов сетевого/локального обмена данными;
- угроза перехвата данных, передаваемых по вычислительной сети;
- угроза обнаружения открытых портов и идентификации привязанных к нему сетевых служб;
- угроза заражения DNS-кеша;
- угроза подмены содержимого сетевых ресурсов;
- угроза подмены субъекта сетевого доступа.

Была проанализирована статья «Удаление вируса Редактируем файл HOSTS»[4] и определена следующая угроза: угроза несанкционированного изменения файла hosts, содержащего локальную базу данных доменных имен.

Так как протокол DNS использует протокол UDP, то была идентифицирована угроза потери UDP-дейтаграмм в сети [5].

Описание угроз DNS-клиента представлено в таблице 4.

Таблица 4 - Описание угроз DNS-клиента

№	Угроза	Описание	Последствия реализации угрозы
1	УБИ.034: Угроза использования слабостей протоколов сетевого/локального обмена данными	Угроза заключается в возможности осуществления нарушителем несанкционированного доступа к передаваемой в системе защищаемой информации за счёт деструктивного воздействия на протоколы сетевого/локального обмена данными.	Нарушение конфиденциальности, целостности

Продолжение таблицы 4

№	Угроза	Описание	Последствия реализации угрозы
2	УБИ.006: Угроза внедрения кода или данных	Угроза заключается в возможности внедрения нарушителем в дискредитируемую информационную систему или IoT-устройство вредоносного кода, который может быть в дальнейшем запущен «вручную» пользователями, автоматически при выполнении определённого условия (наступления определённой даты, входа пользователя в систему и т.п.) или с использованием аутентификационных данных, заданных «по умолчанию», а также в возможности несанкционированного внедрения нарушителем некоторых собственных данных для обработки в дискредитируемую информационную систему, фактически осуществив незаконное использование чужих вычислительных ресурсов, и блокирования работы устройства при выполнении определенных команд.	Нарушение конфиденциальности, доступности, целостности

Продолжение таблицы 4

№	Угроза	Описание	Последствия реализации угрозы
3	УБИ.116: Угроза перехвата данных, передаваемых по вычислительной сети	Угроза заключается в возможности осуществления нарушителем несанкционированного доступа к сетевому трафику дискредитируемой вычислительной сети в пассивном (иногда в активном) режиме (т.е. «прослушивать сетевой трафик») для сбора и анализа сведений, которые могут быть использованы в дальнейшем для реализации других угроз, оставаясь при реализации данной угрозы невидимым (скрытным) получателем перехватываемых данных.	Нарушение конфиденциальности

Продолжение таблицы 4

№	Угроза	Описание	Последствия реализации угрозы
4	УБИ.098: Угроза обнаружения открытых портов и идентификации привязанных к нему сетевых служб	Угроза заключается в возможности определения нарушителем состояния сетевых портов дискредитируемой системы (т.н. сканирование портов) для получения сведений о возможности установления соединения с дискредитируемой системой по данным портам, конфигурации самой системы и установленных средств защиты информации, а также других сведений, позволяющих нарушителю определить по каким портам деструктивные программные воздействия могут быть осуществлены напрямую, а по каким – только с использованием специальных техник обхода межсетевых экранов.	Нарушение конфиденциальности
5	УБИ.019: Угроза заражения DNS-кеша	Угроза заключается в возможности перенаправления нарушителем сетевого трафика через собственный сетевой узел путём опосредованного изменения таблиц соответствия IP- и доменных имён, за счёт генерации лавины возможных ответов на DNS-запрос легальному пользователю.	Нарушение целостности

Продолжение таблицы 4

№	Угроза	Описание	Последствия реализации угрозы
6	Угроза несанкционированного изменения файла hosts, содержащего локальную базу данных доменных имен	Угроза заключается в возможности осуществления нарушителем несанкционированного доступа к защищаемой информации и изменения.	Нарушение конфиденциальности, целостности
7	УБИ.130: Угроза подмены содержимого сетевых ресурсов	<p>Угроза заключается в возможности осуществления нарушителем несанкционированного доступа к защищаемым данным пользователей сети или проведения различных мошеннических действий путём скрытной подмены содержимого хранящихся (сайты, веб-страницы) или передаваемых (электронные письма, сетевые пакеты) по сети данных.</p> <p>Данная угроза обусловлена слабостями технологий сетевого взаимодействия, зачастую не позволяющими выполнить проверку подлинности содержимого электронного сообщения.</p>	Нарушение конфиденциальности, целостности

Продолжение таблицы 4

№	Угроза	Описание	Последствия реализации угрозы
8	УБИ.131: Угроза подмены субъекта сетевого доступа	Угроза заключается в возможности осуществления нарушителем несанкционированного доступа к защищаемым данным пользователя сети или проведения мошеннических действий путем скрытной подмены в отправляемых дискредитируемым пользователем сетевых запросах сведений об отправителе сообщения. Данную угрозу можно охарактеризовать как «имитация действий сервера».	Нарушение конфиденциальности, целостности
9	Угроза потери UDP-дейтаграмм в сети	Передача по протоколу UDP не предусматривает установления соединения, поэтому протокол не гарантирует доставку UDP-дейтаграмм.	Нарушение доступности

Для каждой угрозы экспертным путем была определена степень вероятности возникновения. Результаты определения степени вероятности возникновения угроз представлены в таблице 5.

Таблица 5 - Результаты определения степени вероятности возникновения угроз

№	Угроза	Степень вероятности возникновения угрозы
1	УБИ.034: Угроза использования слабостей протоколов сетевого/локального обмена данными	В
2	УБИ.006: Угроза внедрения кода или данных	В

Продолжение таблицы 5

№	Угроза	Степень вероятности возникновения угрозы
3	УБИ.116: Угроза перехвата данных, передаваемых по вычислительной сети	Н
4	УБИ.098: Угроза обнаружения открытых портов и идентификации привязанных к нему сетевых служб	В
5	УБИ.019: Угроза заражения DNS-кеша	С
6	Угроза несанкционированного изменения файла hosts, содержащего локальную базу данных доменных имен	С
7	УБИ.130: Угроза подмены содержимого сетевых ресурсов	С
8	УБИ.131: Угроза подмены субъекта сетевого доступа	С
9	Угроза потери UDP-дейтаграмм в сети	С

2.3 Идентификация уязвимостей

На основе проведенного анализа протокола DNS, статьи «Кибератаки в подробностях: вмешательство в работу DNS»[6], статьи «Уязвимости сетевых протоколов»[7] были идентифицированы уязвимости DNS-клиента. Уязвимости приведены в таблице 6.

Таблица 6 - Описание уязвимостей DNS

№	Уязвимость	Описание
1	Передача DNS-сообщений по сети в открытом виде	В соответствии с RFC DNS-сообщения между DNS-клиентом и DNS-сервером передаются в виде UDP-дейтаграмм или TCP-пакетов в незашифрованном виде.

Продолжение таблицы 6

№	Уязвимость	Описание
2	Отсутствие проверки подлинности источника и целостности DNS-ответов	В структуре DNS-сообщения отсутствуют поля, связанные с цифровой подписью или контрольной суммой. DNS-сообщение состоит в соответствии с и состоит из следующих полей: ID запроса, флаги, количество запросов, количество ответов, количество NS-записей, количество дополнительных ответов, секция вопросы, секция ответы, секция NS-записи, секция дополнительных ответов. Злоумышленник может подделать DNS-ответы или изменить отправленные DNS-сообщения на пути от DNS-клиента к DNS-серверу.
3	Маленький диапазон выбора портов в протоколах TCP/IP	В стеке TCP/IP количество портов ограничено 65535. Перебор всех портов возможен за достаточно небольшое количество времени.
4	Отсутствие контроля целостности файла hosts	В операционной системе Windows отсутствует контроль целостности файла hosts.
5	Отсутствие защитного механизма от подмены IP-адреса	Злоумышленник может сформировать IP-пакет с любым IP-адресом отправителя.
6	Отсутствие установления соединения в протоколе UDP	Из-за отсутствия установления соединения протокол UDP не гарантирует доставку.

Была проведена оценка простоты использования уязвимостей с использованием критериев, приведенных в таблице 7. Результаты оценки показаны в таблице 8.

Таблица 7 - Критерии оценки простоты использования уязвимостей

Критерий	Описание
Низкая	Уязвимость существует (присуща) у актива (активу) в минимальной степени и/или со значительным трудом может быть использована
Средняя	Уязвимость существует (присуща) у актива (активу) частично и/или без значительных затруднений может быть использована
Высокая	Уязвимость существует (присуща) у актива (активу) в максимальной степени и/или легко может быть использована

Таблица 8 - Результаты оценки простоты использования уязвимостей

№	Уязвимость	Простота использования
1	Передача DNS-сообщений по сети в открытом виде	Высокая
2	Отсутствие проверки подлинности источника и целостности DNS-ответов	Высокая
3	Маленький диапазон выбора портов в протоколах TCP/IP	Средняя
4	Отсутствие контроля целостности файла hosts	Высокая
5	Отсутствие защитного механизма от подмены IP-адреса	Высокая
6	Отсутствие установления соединения в протоколе UDP	Средняя

2.4 Идентификация сценариев инцидентов

Были идентифицированы возможные сценарии инцидентов ИБ. Результаты показаны в таблице 9.

Таблица 9 - Сценарии инцидентов

№	Угроза	Уязвимость
1	УБИ.034: Угроза использования слабостей протоколов сетевого/локального обмена данными	Передача DNS-сообщений по сети в открытом виде
2	УБИ.034: Угроза использования слабостей протоколов сетевого/локального обмена данными	Отсутствие проверки подлинности источника и целостности DNS-ответов
3	УБИ.034: Угроза использования слабостей протоколов сетевого/локального обмена данными	Маленький диапазон выбора портов в протоколах TCP/IP
4	УБИ.034: Угроза использования слабостей протоколов сетевого/локального обмена данными	Отсутствие защитного механизма от подмены IP-адреса
5	УБИ.006: Угроза внедрения кода или данных	-
6	УБИ.116: Угроза перехвата данных, передаваемых по вычислительной сети	Передача DNS-сообщений по сети в открытом виде
7	УБИ.098: Угроза обнаружения открытых портов и идентификации привязанных к нему сетевых служб	Маленький диапазон выбора портов в протоколах TCP/IP
8	УБИ.019: Угроза заражения DNS-кеша	Отсутствие проверки подлинности источника и целостности DNS-ответов
9	УБИ.019: Угроза заражения DNS-кеша	Отсутствие контроля целостности файла hosts
10	УБИ.019: Угроза заражения DNS-кеша	Отсутствие защитного механизма от подмены IP-адреса
11	Угроза несанкционированного изменения файла hosts, содержащего локальную базу данных доменных имен	Отсутствие контроля целостности файла hosts

Продолжение таблицы 9

№	Угроза	Уязвимость
12	УБИ.130: Угроза подмены содержимого сетевых ресурсов	Отсутствие защитного механизма от подмены IP-адреса
13	УБИ.130: Угроза подмены содержимого сетевых ресурсов	Отсутствие проверки подлинности источника и целостности DNS-ответов
14	УБИ.130: Угроза подмены содержимого сетевых ресурсов	Отсутствие контроля целостности файла hosts
15	УБИ.131: Угроза подмены субъекта сетевого доступа	Отсутствие защитного механизма от подмены IP-адреса
16	Угроза потери UDP-дейтаграмм в сети	Отсутствие установления соединения в протоколе UDP

2.5 Оценивание рисков DNS-клиента

Для оценивания рисков DNS-клиента был использован метод "Матрица с заранее определенными значениями"[8]. Матрица для определения меры риска показана в таблице 10.

Таблица 10 - Матрица для определения меры риска

Степень вероятности возникновения угрозы		Низкая			Средняя			Высокая		
Простота использования		Н	С	В	Н	С	В	Н	С	В
Ценность актива	Н	1	2	3	2	3	4	3	4	5
	С	2	3	4	3	4	5	4	5	6
	В	3	4	5	4	5	6	5	6	7

Была определена мера риска каждого сценария инцидента. Результаты оценивания рисков показаны в таблице 11. Все активы DNS-клиента имеют высокую ценность, поэтому ценность активов не показана в таблице 11. Так как

угроза «Угроза внедрения кода или данных» не имеет уязвимостей, то было решено присвоить значению «простота использования» - средняя.

Таблица 11 - Результаты оценивания рисков ИБ

№	Угроза	Степень вероятности возникновения угрозы	Уязвимость	Простота использования	Мера риска
1	УБИ.034: Угроза использования слабостей протоколов сетевого/локального обмена данными	В	Передача DNS-сообщений по сети в открытом виде	В	7
2	УБИ.034: Угроза использования слабостей протоколов сетевого/локального обмена данными	В	Отсутствие проверки подлинности источника и целостности DNS-ответов	В	7
3	УБИ.034: Угроза использования слабостей протоколов сетевого/локального обмена данными	В	Маленький диапазон выбора портов в протоколах TCP/IP	С	6
4	УБИ.034: Угроза использования слабостей протоколов сетевого/локального обмена данными	В	Отсутствие защитного механизма от подмены IP-адреса	В	7

Продолжение таблицы 11

№	Угроза	Степень вероятности возникновения угрозы	Уязвимость	Простота использования	Мера риска
5	УБИ.006: Угроза внедрения кода или данных	В	-	В	7
6	УБИ.116: Угроза перехвата данных, передаваемых по вычислительной сети	Н	Передача DNS-сообщений по сети в открытом виде	В	5
7	УБИ.098: Угроза обнаружения открытых портов и идентификации привязанных к нему сетевых служб	В	Маленький диапазон выбора портов в протоколах TCP/IP	С	6
8	УБИ.019: Угроза заражения DNS-кеша	С	Отсутствие проверки подлинности источника и целостности DNS-ответов	В	6
9	УБИ.019: Угроза заражения DNS-кеша	С	Отсутствие контроля целостности файла hosts	В	6
10	УБИ.019: Угроза заражения DNS-кеша	С	Отсутствие защитного механизма от подмены IP-адреса	В	6

Продолжение таблицы 11

№	Угроза	Степень вероятности возникновения угрозы	Уязвимость	Простота использования	Мера риска
11	Угроза несанкционированного изменения файла hosts, содержащего локальную базу данных доменных имен	С	Отсутствие контроля целостности файла hosts	В	6
12	УБИ.130: Угроза подмены содержимого сетевых ресурсов	С	Отсутствие защитного механизма от подмены IP-адреса	В	6
13	УБИ.130: Угроза подмены содержимого сетевых ресурсов	С	Отсутствие проверки подлинности источника и целостности DNS-ответов	В	6
14	УБИ.130: Угроза подмены содержимого сетевых ресурсов	С	Отсутствие контроля целостности файла hosts	В	6
15	УБИ.131: Угроза подмены субъекта сетевого доступа	С	Отсутствие защитного механизма от подмены IP-адреса	В	6

Продолжение таблицы 11

№	Угроза	Степень вероятности возникновения угрозы	Уязвимость	Простота использования	Мера риска
16	Угроза потери UDP-дейтаграмм в сети	С	Отсутствие установления соединения в протоколе UDP	С	5

Для всех сценариев инцидентов был определен уровень риска с помощью критериев определения уровня риска из таблицы 12. Результаты определения уровня риска представлены в таблице 13.

Таблица 12 - Критерий определения уровня риска

Мера риска	Уровень риска
1-2	Низкий
3-4	Средний
5-7	Высокий

Таблица 13 - Результаты определения уровня риска

№	Угроза	Уязвимость	Уровень риска
1	УБИ.034: Угроза использования слабостей протоколов сетевого/локального обмена данными	Передача DNS-сообщений по сети в открытом виде	Высокий
2	УБИ.034: Угроза использования слабостей протоколов сетевого/локального обмена данными	Отсутствие проверки подлинности источника и целостности DNS-ответов	Высокий

Продолжение таблицы 13

№	Угроза	Уязвимость	Уровень риска
3	УБИ.034: Угроза использования слабостей протоколов сетевого/локального обмена данными	Маленький диапазон выбора портов в протоколах TCP/IP	Высокий
4	УБИ.034: Угроза использования слабостей протоколов сетевого/локального обмена данными	Отсутствие защитного механизма от подмены IP-адреса	Высокий
5	УБИ.006: Угроза внедрения кода или данных	-	Высокий
6	УБИ.116: Угроза перехвата данных, передаваемых по вычислительной сети	Передача DNS-сообщений по сети в открытом виде	Высокий
7	УБИ.098: Угроза обнаружения открытых портов и идентификации привязанных к нему сетевых служб	Маленький диапазон выбора портов в протоколах TCP/IP	Высокий
8	УБИ.019: Угроза заражения DNS-кеша	Отсутствие проверки подлинности источника и целостности DNS-ответов	Высокий
9	УБИ.019: Угроза заражения DNS-кеша	Отсутствие контроля целостности файла hosts	Высокий
10	УБИ.019: Угроза заражения DNS-кеша	Отсутствие защитного механизма от подмены IP-адреса	Высокий

Продолжение таблицы 13

№	Угроза	Уязвимость	Уровень риска
11	Угроза несанкционированного изменения файла hosts, содержащего локальную базу данных доменных имен	Отсутствие контроля целостности файла hosts	Высокий
12	УБИ.130: Угроза подмены содержимого сетевых ресурсов	Отсутствие защитного механизма от подмены IP-адреса	Высокий
13	УБИ.130: Угроза подмены содержимого сетевых ресурсов	Отсутствие проверки подлинности источника и целостности DNS-ответов	Высокий
14	УБИ.130: Угроза подмены содержимого сетевых ресурсов	Отсутствие контроля целостности файла hosts	Высокий
15	УБИ.131: Угроза подмены субъекта сетевого доступа	Отсутствие защитного механизма от подмены IP-адреса	Высокий
16	Угроза потери UDP-дейтаграмм в сети	Отсутствие установления соединения в протоколе UDP	Высокий

3 Выбор способа обработки рисков

Для каждого сценария реализации угрозы был выбран способ обработки риска. Результаты показаны в таблице 14.

Таблица 14 - Способы обработки риска

№	Угроза	Уязвимость	Способ обработки риска
1	УБИ.034: Угроза использования слабостей протоколов сетевого/локального обмена данными	Передача DNS-сообщений по сети в открытом виде	Снижение
2	УБИ.034: Угроза использования слабостей протоколов сетевого/локального обмена данными	Отсутствие проверки подлинности источника и целостности DNS-ответов	Снижение
3	УБИ.034: Угроза использования слабостей протоколов сетевого/локального обмена данными	Маленький диапазон выбора портов в протоколах TCP/IP	Снижение
4	УБИ.034: Угроза использования слабостей протоколов сетевого/локального обмена данными	Отсутствие защитного механизма от подмены IP-адреса	Снижение
5	УБИ.006: Угроза внедрения кода или данных	-	Снижение

Продолжение таблицы 14

№	Угроза	Уязвимость	Способ обработки риска
6	УБИ.116: Угроза перехвата данных, передаваемых по вычислительной сети	Передача DNS-сообщений по сети в открытом виде	Принятие
7	УБИ.098: Угроза обнаружения открытых портов и идентификации привязанных к нему сетевых служб	Маленький диапазон выбора портов в протоколах TCP/IP	Снижение
8	УБИ.019: Угроза заражения DNS-кеша	Отсутствие проверки подлинности источника и целостности DNS-ответов	Снижение
9	УБИ.019: Угроза заражения DNS-кеша	Отсутствие контроля целостности файла hosts	Снижение
10	УБИ.019: Угроза заражения DNS-кеша	Отсутствие защитного механизма от подмены IP-адреса	Снижение
11	Угроза несанкционированного изменения файла hosts, содержащего локальную базу данных доменных имен	Отсутствие контроля целостности файла hosts	Снижение
12	УБИ.130: Угроза подмены содержимого сетевых ресурсов	Отсутствие защитного механизма от подмены IP-адреса	Снижение

Продолжение таблицы 14

№	Угроза	Уязвимость	Способ обработки риска
13	УБИ.130: Угроза подмены содержимого сетевых ресурсов	Отсутствие проверки подлинности источника и целостности DNS-ответов	Снижение
14	УБИ.130: Угроза подмены содержимого сетевых ресурсов	Отсутствие контроля целостности файла hosts	Снижение
15	УБИ.131: Угроза подмены субъекта сетевого доступа	Отсутствие защитного механизма от подмены IP-адреса	Снижение
16	Угроза потери UDP-дейтаграмм в сети	Отсутствие установления соединения в протоколе UDP	Снижение

Было решено, что необходимо снижать уровень риска для каждого сценария реализации угрозы кроме реализации угрозы «УБИ.116: Угроза перехвата данных, передаваемых по вычислительной сети» с использованием уязвимости «Передача DNS-сообщений по сети в открытом виде», так как конфиденциальность не является значимым свойством ИБ для DNS-клиента. Так же при использовании защитных мер для снижения рисков угрозы «УБИ.116: Угроза перехвата данных, передаваемых по вычислительной сети» скорость разрешения доменных имен, а вследствие скорость работы прикладных программ будет значительно снижена.

4 Выбор защитных мер

Для выбора защитных мер были использованы следующие документы:

- RFC 1034 «Domain Names – concepts and facilities»;
- RFC 1035 «Domain names – implementation and specification»;
- RFC 5452 «Measures for Making DNS More Resilient against Forged Answers»[9];
- RFC 2845 «Secret Key Transaction Authentication for DNS»[10];
- RFC 7873 «Domain Name System (DNS) Cookies»[11];
- черновик RFC «Use of Bit 0x20 in DNS Labels to Improve Transaction Identity»[12].

Был проанализирован документ RFC 5452 «Measures for Making DNS More Resilient against Forged Answers». В нем представлены следующие защитные меры DNS-клиента:

- секции запроса отправленного и принятого DNS-пакета должны быть одинаковыми;
- проверка идентификатора DNS-пакета;
- проверка IP-адреса и порта DNS-сервера;
- порт отправления должен быть случайным;
- идентификатор DNS-пакета должен быть случайным;
- при многократном получении некорректных ответов необходимо попытаться отправить повторно DNS-запрос по TCP;
- использование протокола TSIG (RFC 2845 «Secret Key Transaction Authentication for DNS»);
- доменные имена в принятых DNS-записях должны относиться к доменному имени из секции запроса.

Был проанализирован документ RFC 7873 «Domain Name System (DNS) Cookies». В нем представлена защитная мера: аутентификация DNS-клиента и DNS-сервера с помощью DNS-Cookies.

Далее был проанализирован черновик RFC «Use of Bit 0x20 in DNS Labels to Improve Transaction Identity». В нем описана защитная мера: использование случайных прописных и строчных букв в имени домена при запросе.

Кроме вышеперечисленных защитных мер были выделены следующие:

- проверка принятых DNS-ответов на соответствие RFC 1034 «Domain Names – concepts and facilities», RFC 1035 «Domain names – implementation and specification»;
- контроль за переполнением буфера и заикливания при разборе DNS-ответов;
- повторная отправка DNS-сообщений;
- выбор случайного DNS-сервера;
- ведение журналов аудита DNS-клиента;
- подпись файла hosts, содержащего локальную базу данных доменных имен, с помощью вычисления хеш-суммы с ключом.

Было проведено описание идентифицированных защитных мер. Описание приведено в таблице 15.

Таблица 15 - Описание защитных мер

Защитная мера	Описание
Секции запроса отправленного и принятого DNS-пакета должны быть одинаковыми	Поля доменное имя, тип запроса, класс запроса должны одинаковыми в отправленном и принятом DNS-пакете.
Проверка идентификатора DNS-пакета	Идентификатор должен одинаковыми в отправленном и принятом DNS-пакете.
Проверка IP-адреса и порта DNS-сервера	Адрес источника в принятом DNS-пакете должен принадлежать одному из DNS-серверов, которому был отправлен запрос.

Продолжение таблицы 15

Защитная мера	Описание
Порт отправления должен быть случайным	Для того, чтобы злоумышленнику было сложнее обнаружить открытый порт, для каждого запроса используется случайный порт отправления.
Идентификатор DNS-пакета должен быть случайным	Для того, чтобы злоумышленнику было сложнее подобрать идентификатор, необходимо использовать случайный идентификатор в каждом запросе.
При многократном получении некорректных ответов необходимо попытаться отправить повторно DNS-запрос по TCP	Если DNS-клиент стал получать некорректные ответы, то возможно злоумышленник перебирает идентификатор запроса, отправляя UDP-дейтаграммы. Для получения корректного ответа необходимо установить соединение с DNS-сервером по TCP.
Доменные имена в принятых DNS-записях должны относиться к доменному имени из секции запроса	Для исключения ситуация когда в DNS-ответе находятся записи, которые никаким образом не имеют отношения к DNS-запросу. Например, при DNS-запросе получения IP-адреса pnzgu.ru в DNS-ответе находятся записи типа A: yandex.ru: 185.5.4.8.
Контроль за переполнением буфера и заикливания при разборе DNS-ответов	<p>При чтении полей DNS-сообщения необходимо убедиться в том, что при чтении данного поля не будет выход за пределы буфера DNS-сообщения. Осуществлять чтение полей необходимо строго в соответствии с RFC.</p> <p>При формировании DNS-записей для отправки прикладной программе необходимо проверять возможно ли записать данные в буфер DNS-сообщений или нет.</p>

Продолжение таблицы 15

Защитная мера	Описание
Использование протокола TSIG	<p>Для обеспечения целостности и аутентичности DNS-записей необходимо использовать протокол TSIG. При использовании протокола TSIG к запросу добавляется DNS-запись типа TSIG, которая содержит подпись DNS-сообщения. Подпись формируется с помощью хеш-функции с ключом.</p> <p>Когда DNS-клиент отправляет DNS-запрос, то вычисляется значение хеша от DNS-сообщения до добавления записи TSIG и следующих полей записи TSIG: наименование ключа, класс, время жизни, наименование алгоритма хеширования, значение времени, значение погрешности времени, код ошибки, длина других данных, другие данные.</p> <p>Когда DNS-клиент получает DNS-ответ, то происходит вычисление значения хеша от подписи отправленного DNS-запроса серверу, от принятого DNS-сообщения без записи TSIG и следующих полей записи TSIG: наименование ключа, класс, время жизни, наименование алгоритма хеширования, значение времени, значение погрешности времени, код ошибки, длина других данных, другие данные.</p>
Повторная отправка DNS-сообщений	<p>Для исключения случаев, когда из-за ошибок в сети DNS-сообщения не доходят до адреса по протоколу UDP, необходимо использовать повторную отправку DNS-сообщений.</p>

Продолжение таблицы 15

Защитная мера	Описание
Выбор случайного DNS-сервера	Данная защитная мера вносит меру неопределенности для злоумышленника который будет пытаться заразить DNS-кеш, так как для успешного принятия запроса необходимо, чтобы адрес должен принадлежать одному из DNS-серверов, которому был отправлен запрос.
Ведение журналов аудита DNS-клиента	Данная мера необходима для журналирования инцидентов безопасности DNS-клиента.
Подпись файла hosts, содержащего локальную базу данных доменных имен, с помощью вычисления хеш-суммы с ключом	Для исключения изменений файла hosts необходимо при каждом запуске DNS-клиента проверять его хеш-сумму.
Проверка принятых DNS-ответов на соответствие RFC 1034 «Domain Names – concepts and facilities», RFC 1035 «Domain names – implementation and specification»	Необходимо строго следовать стандартам RFC для правильной работы DNS-клиента.
Использование случайных прописных и строчных букв в имени домена при запросе	Данный механизм значительно снижает вероятность заражения DNS-кеша методом перебора параметров. При использовании данного механизма в доменном имени в запросе меняется регистр букв. При принятии DNS-сообщения проверяется доменное имя в секции запрос.

Продолжение таблицы 15

Защитная мера	Описание
Использование механизма DNS Cookie	<p>Данный механизм значительно снижает вероятность заражения DNS-кеша методом перебора параметров: идентификатора, имени домена, адреса DNS-сервера.</p> <p>При включении данного механизма к каждому DNS-сообщению добавляется запись типа OPT со значением DNS Client Cookie. Данные DNS Cookie должны совпадать в отправленном запросе и принятом ответе. Размер DNS Cookie 64 бита. Для каждого DNS-сообщения формируется новое значение DNS Cookie.</p>

Было произведено сопоставление угроз и выбранных защитных мер. Результаты сопоставления угроз и выбранных защитных мер приведены в таблице 16.

Таблица 16 - Сопоставление угроз и выбранных защитных мер

Угроза	Защитные меры
УБИ.006: Угроза внедрения кода или данных	<p>контроль за переполнением буфера при разборе DNS-ответов;</p> <p>проверка принятых DNS-ответов на соответствие RFC 1034 «Domain Names – concepts and facilities», RFC 1035 «Domain names – implementation and specification»;</p> <p>ведение журналов аудита DNS-клиента</p>

Продолжение таблицы 16

Угроза	Защитные меры
<p>УБИ.034: Угроза использования протоколов сетевого/локального обмена данными</p>	<p>секции запроса отправленного и принятого DNS-пакета должны быть одинаковыми; проверка идентификатора DNS-пакета; проверка IP-адреса и порта DNS-сервера; порт отправления должен быть случайным; идентификатор DNS-пакета должен быть случайным; при многократном получении некорректных ответов необходимо попытаться отправить повторно DNS-запрос по TCP; доменные имена в принятых DNS-записях должны относиться к доменному имени из секции запроса; ведение журналов аудита DNS-клиента; использование протокола TSIG (RFC 2845 «Secret Key Transaction Authentication for DNS»)</p>
<p>УБИ.098: Угроза обнаружения открытых портов и идентификации привязанных к нему сетевых служб</p>	<p>порт отправления должен быть случайным</p>

Продолжение таблицы 16

Угроза	Защитные меры
УБИ.019: Угроза заражения DNS-кеша	<p>секции запроса отправленного и принятого DNS-пакета должны быть одинаковыми;</p> <p>порт отправления должен быть случайным;</p> <p>идентификатор DNS-пакета должен быть случайным;</p> <p>проверка идентификатора DNS-пакета;</p> <p>при многократном получении некорректных ответов необходимо попытаться отправить повторно DNS-запрос по TCP;</p> <p>использование протокола TSIG (RFC 2845 «Secret Key Transaction Authentication for DNS»);</p> <p>доменные имена в принятых DNS-записях должны относиться к доменному имени из секции запроса;</p> <p>использование случайных прописных и строчных букв в имени домена при запросе</p> <p>выбор случайного DNS-сервера;</p> <p>аутентификация DNS-клиента и DNS-сервера с помощью DNS-Cookies;</p> <p>использование случайных прописных и строчных букв в имени домена при запросе</p>
Угроза несанкционированного изменения файла hosts, содержащего локальную базу данных доменных имен	<p>подпись файла hosts, содержащего локальную базу данных доменных имен, с помощью вычисления хеш-суммы с ключом</p>

Продолжение таблицы 16

Угроза	Защитные меры
УБИ.130: Угроза подмены содержимого сетевых ресурсов	<p>секции запроса отправленного и принятого DNS-пакета должны быть одинаковыми;</p> <p>порт отправления должен быть случайным;</p> <p>идентификатор DNS-пакета должен быть случайным;</p> <p>проверка идентификатора DNS-пакета;</p> <p>доменные имена в принятых DNS-записях должны относиться к доменному имени из секции запроса;</p> <p>при многократном получении некорректных ответов необходимо попытаться отправить повторно DNS-запрос по TCP;</p> <p>использование случайных прописных и строчных букв в имени домена при запросе;</p> <p>использование протокола TSIG (RFC 2845 «Secret Key Transaction Authentication for DNS»)</p>
УБИ.131: Угроза подмены субъекта сетевого доступа	использование протокола TSIG (RFC 2845 «Secret Key Transaction Authentication for DNS»)
Угроза потери UDP-дейтаграмм в сети	использование механизм повтора DNS-запросов через промежутки времени

Перечисленные выше меры защиты снижают вероятность возникновения всех идентифицированных угроз безопасности DNS-клиента.

5 Разработка архитектуры DNS-клиента

В соответствии с ТЗ DNS-клиент должен представлять собой консольное приложение, написанное на языке Си.

Была разработана архитектура DNS-клиента, архитектура показана на рисунке 5.

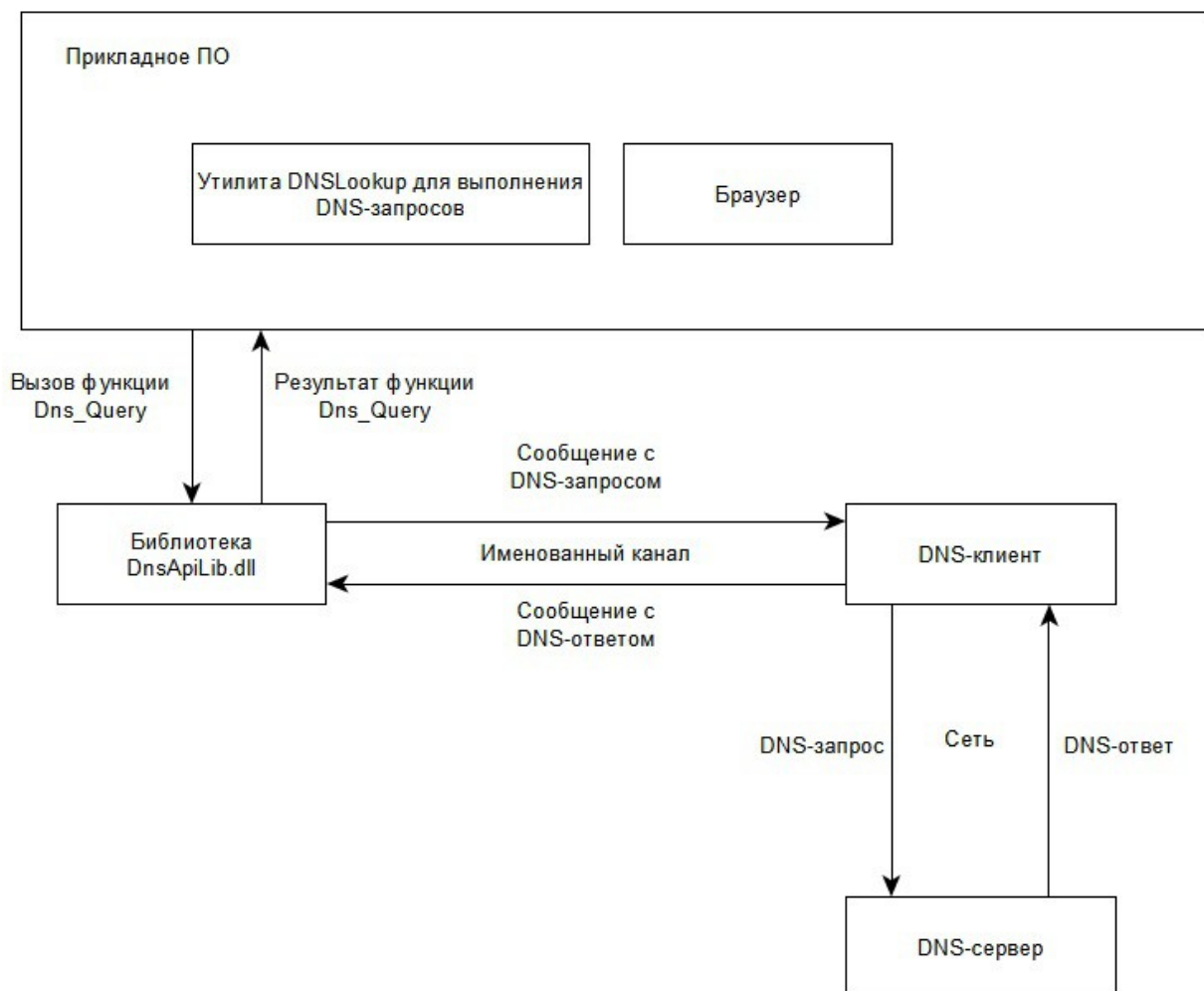


Рисунок 5 - Архитектура DNS-клиента

Для того, чтобы какому-либо прикладному ПО узнать IP-адрес доменного имени необходимо подключить библиотеку DnsApiLib.dll и вызвать функцию Dns_Query. Библиотека DnsApiLib.dll представляет собой динамическую библиотеку, в которой находится функция Dns_Query. Именованный канал — это канал для передачи сообщений между программами.

Утилита DNSLookup была разработана для тестирования DNS-клиента.
Утилита представляет собой консольное приложение.

Прототип функции Dns_Query:

```
/**  
 * Функция DNS-запроса. В ответе все символы в кодировке Unicode.  
 * @param[in] pszName Доменное имя  
 * @param[in] wType Тип запроса  
 * @param[in] Options Опции  
 * @param[in] pExtra Должно быть NULL или указатель типа PIP4_ARRAY
```

для задания DNS-серверов

```
 * @param[in] wszTsigKey Ключ для протокола TSIG  
 * @param[in] ppQueryResults Результаты DNS-запроса  
 * @return ERROR_SUCCESS - успех  
 *          ERROR_INVALID_PARAMETER - не все параметры заданы
```

или заданы некорректно

```
 *          DNS_ERROR_RCODE - см. GetLastError()  
 *          DNS_INFO_NO_RECORDS - DNS-записи в ответе
```

отсутствуют

```
 */
```

DNSAPI

DNS_STATUS

DNSAPI_CALL_TYPE

Dns_Query(
 In PCWSTR pszName,
 In WORD wType,
 In DWORD Options,
 _Inout_opt_ PIP4_ARRAY pExtra,
 _In_z_ PCWSTR wszTsigKey,
 _Outptr_result_maybenull_ PDNS_RECORD * ppQueryResults

);

В функции проверяются входные параметры, и после этого библиотека DnsApiLib.dll формирует сообщение для отправки по именованному каналу DNS-клиенту.

На протяжении работы DNS-клиента в нём открыт именованный канал для принятия запросов от прикладных программ. DNS-клиент после принятия сообщения по именованному каналу формирует контекст DNS-операции.

Контекст DNS-операции:

```
typedef struct
{
    // поля, относящиеся к I/O
    OVERLAPPED OverlappedSocket4;
    OVERLAPPED OverlappedSocket6;
    OVERLAPPED OverlappedPipe;
    // тип операции
    BYTE OperationType;
    // Информация о транспорте
    LPPER_HANDLE_DATA PipeInfo;
    LPPER_HANDLE_DATA Socket4Info;
    LPPER_HANDLE_DATA Socket6Info;
    // Протокол передачи (UDP или TCP)
    BYTE bProtocolSocket;
    // Контекст DNS-клиента
    PSDnsResolverContext DnsResolverContext;
    // поля, относящиеся непосредственно к DNS
    // версия DnsQuery
    BYTE bQueryVersion;
    // ID запроса
    WORD Xid;
```

```

// доменное имя
char* QueryName;
// Кодировка символов
BYTE CharSet;
// тип DNS-запроса
WORD QueryType;
// Опции
ULONG64 QueryOptions;
// Код статуса
DNS_STATUS QueryStatus;
// Код ошибки
DWORD dwCodeError;
// количество принятых некорректных DNS-ответов
BYTE bCountIncorrectUdpDnsResposes;
// Количество таймаутов DNS-запросов
BYTE bCountTimeouts;
// Количество неудачных присоединений DNS-суффиксов к доменам
BYTE bCountRetryDnsSuffix;
// Выбранные DNS-серверы через параметр в функциях DnsQuery и
DnsQueryEx
PDNS_ADDR_ARRAY pDnsServersCustom;
// таймер для повторных запросов
HANDLE hTimer;
// Клиентские DNS куки
BYTE* pClientCookie;
// TSIG
// Ключ
BYTE* pKeyTsig;
// длина ключа

```

```

WORD wLenKey;

// MAC запроса
BYTE* pRequestMac;

// Число ссылок
LONG m_lReferencesCount;

} PER_IO_OPERATION_DATA, *LPPER_IO_OPERATION_DATA;

```

В данной структуре хранится вся информация, которая необходима для выполнения DNS-запроса.

Была разработана диаграмма вариантов использования DNS-клиента. Диаграмма вариантов использования приведена в приложении А на рисунке А.1.

DNS-клиент является многопоточным приложением. Количество потоков зависит от количества ядер в системе. Операции чтения и записи выполняются асинхронно. Для эффективного выполнения асинхронных операций в DNS-клиенте используется модель «порта завершения» (completion port). Модель «порта завершения» - это специальный механизм в операционной системе Windows, с помощью которого приложение использует ограниченное число потоков для обработки асинхронных операций. После того, как асинхронная операция будет выполнена операционная система оповещает об этом специальную функцию и возвращает контекст операции. Далее контекст операции проходит этапы для выполнения DNS-запроса. Ниже представлена функция работы потоков операций чтения/записи.

```

DWORD
WINAPI
IOWorkerThread(
_In_ LPVOID CompletionPortID)
{
    HANDLE CompletionPort = (HANDLE)CompletionPortID;

    // Количество байт в IO операции
    DWORD dwBytesTransferred = 0;

```

```

LPPER_HANDLE_DATA PerHandleData = NULL;
LPPER_IO_OPERATION_DATA PerIoData = NULL;
// Код ошибки
BYTE bCodeReturn = 0;
LPOVERLAPPED pOverlapped = NULL;
while (TRUE)
{
    // Ожидание завершения ввода-вывода на любом из сокетов
или именованных каналов
    BOOL bSuccess = GetQueuedCompletionStatus(CompletionPort,
&dwBytesTransferred, (LPDWORD)&PerHandleData, &pOverlapped, INFINITE);
    if (!bSuccess)
    {
        if (!pOverlapped)
        {
            // ошибка IOCP
            return 1;
        }
    }
    // получаем контекст
    PerIoData = GetOperationContext(pOverlapped, PerHandleData);
    // Определяем тип операции
    switch (PerIoData->OperationType)
    {
        // подключение прикладной программы по имен. каналу
        case CLIENT_CONNECT_NAMEDPIPE:
        {
            break;
        }
    }
}

```

```

// было принято сообщение по именованному каналу от программы
case READ_DNS_QUESTION_IN_NAMEDPIPE:
{
    break;
}
// было принято DNS-сообщение из сети
case RECEIVE_DNS_RESPONSE:
{
    break;
}
}
return 0;
}

```

DNS-клиент имеет следующие кеши: кеш файла hosts, кеш DNS-ответов, кеш негативных DNS-ответов. За основу кеша был взят `hash_map`. Основная структура кеша показана на рисунке 6.

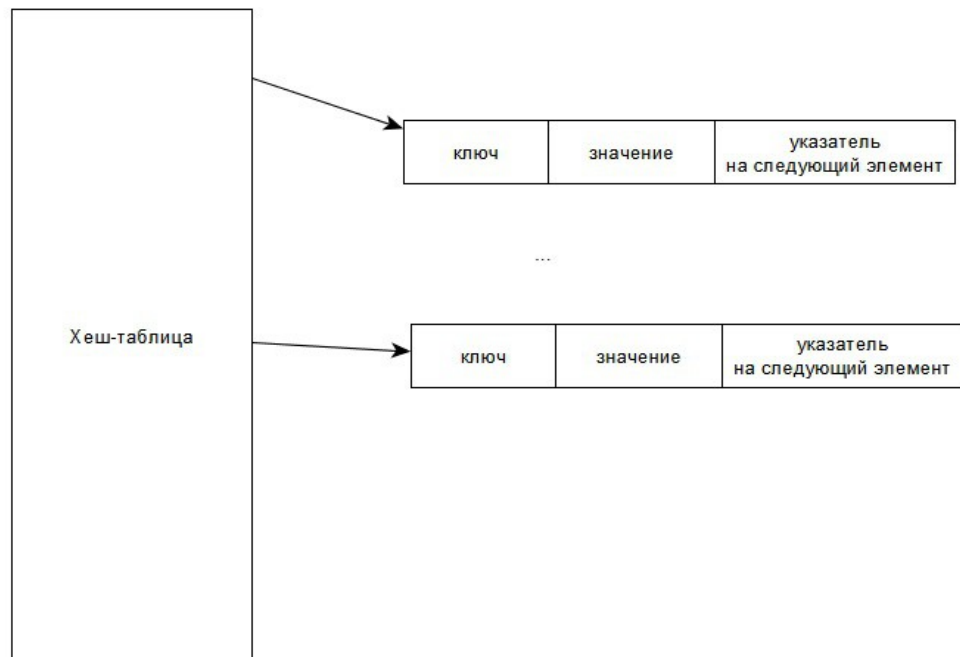


Рисунок 6 - Основная структура кеша

После принятия сообщения по именованному каналу DNS-клиент анализирует доменное имя и тип записи. Если доменное имя «loopback», «localhost» или содержит loopback-адрес, то происходит разрешение DNS-имени локально. В этом случае возвращается IP-адрес loopback интерфейса или доменное имя «localhost».

Если доменное имя — это Netbios имя, то происходит разрешение DNS-имени локально. В этом случае возвращается список IP-адресов всех интерфейсов кроме loopback.

Если доменное имя — это один из адресов интерфейса, то возвращается Netbios имя.

Далее производится поиск в кеше DNS-записей файла hosts. Если запись найдена, то она возвращается программе. После этого аналогично проверяется кеш DNS-записей, потом кеш негативных DNS-ответов.

Если ни в одном кеше нет информации по DNS-запросу, то формируется буфер DNS-запроса для отправки DNS-серверу.

DNS-клиент поддерживает UDP и TCP для отправки и принятия DNS-сообщений от DNS-сервера. Так как в основном DNS-сообщения отправляются по UDP, то в DNS-клиенте реализовано повторная отправка DNS-запросов через заданные промежутки времени: 1 секунда, 2 секунды, 4 секунды. Если DNS-ответ не получен через 1 секунду, то еще раз происходит отправка DNS-запроса и так далее. Если через 9 секунд DNS-клиент, ничего не получит, то он отправляет прикладной программе код ошибки «время ожидания ответа вышло».

На рисунке 7 показано, как происходит повтор DNS-сообщения. На рисунке 7 черная точка означает отправку. Сначала DNS-клиент отправляет DNS-запрос всем DNS-серверам на всех адаптерах компьютера. Далее только предпочтительным серверам, потом снова всем серверам, после осуществляется отправка альтернативным серверам. После отправки альтернативным серверам DNS-клиент ожидает ответ в течение 2 секунд.

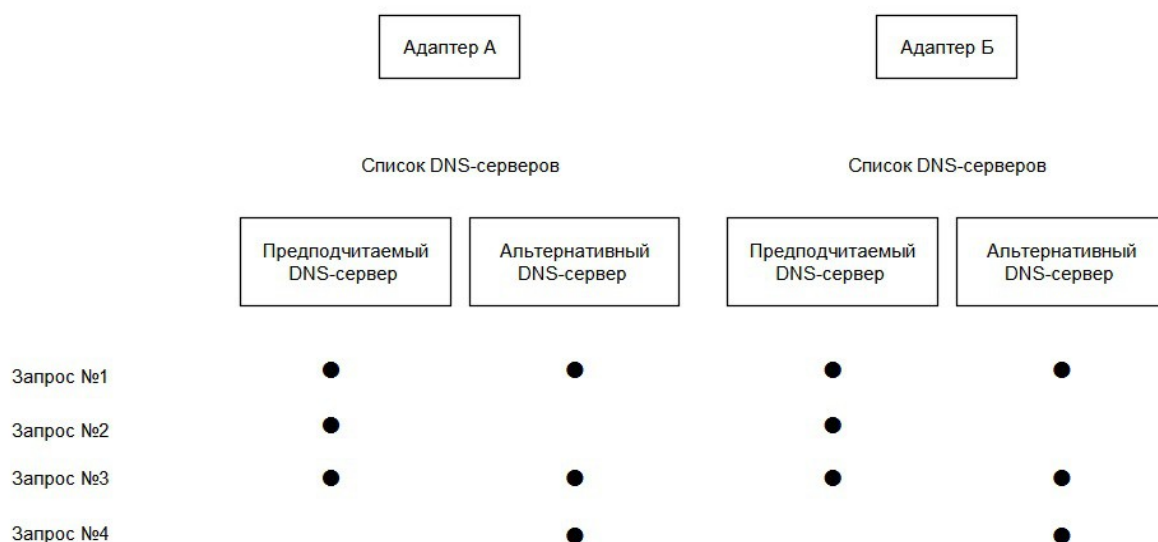


Рисунок 7 - Повторы DNS-запросов

После получения DNS-ответа от DNS-сервера DNS-клиент анализирует DNS-сообщение и производит разбор на DNS-записи. Если DNS-сообщение имеет корректный ID, имя домена, соответствует RFC и содержит DNS-записи, то полученные DNS-записи добавляются в кеш DNS-ответов. Если записей нет, то DNS-ответ корректный, то добавляется запись об отсутствии записей у DNS-серверов в кеш негативных ответов.

После добавления в кеш DNS-клиент отправляет результат обработки прикладной программе по именованному каналу. Далее прикладная программа может использовать DNS-ответ для своих целей.

Для ведения журналов аудита DNS-клиент используется текстовый файл. Название файла имеет следующий формат: DnsResolver_{год}{месяц}{день}.txt

Формат записи в файле следующий:

{год}/{месяц}/{день} {время}|{тип сообщения}: {сообщение}.

В DNS-клиенте используются следующие сообщения:

- «Получено DNS-сообщение, несоответствующее RFC, для запроса {домен} (тип: {тип записи})»;
- «Получено DNS-сообщение, в котором DNS-куки неверные, для запроса {домен} (тип: {тип записи})»;

- «Получено DNS-сообщение, в котором подпись TSIG неверна, для запроса {домен} (тип: {тип записи})»;
- «Получено DNS-сообщение с неверным ID для запроса {домен} (тип: {тип записи})»;
- «Получено DNS-сообщение, в котором секция запрос отличается от секции в DNS-запроса, для запроса {домен} (тип: {тип записи})»;
- «Был превышен лимит (3) неверных DNS-ответов по UDP для запроса {домен} (тип: {тип записи})»;
- «Неверная подпись у файла hosts».

6 Программная реализация функции DNS-запроса для прикладных программ

Для отправки DNS-запроса от прикладной программы DNS-клиенту была написана функция `DnsQuery`.

Функция отправляет DNS-запрос DNS-клиенту и получает результат в виде кода ошибки или DNS-записей. Перед отправкой запроса происходит проверка введенных параметров функции, а именно: доменное имя, тип запроса. При необходимости прикладная программа может сама задать DNS-серверы, которым необходимо отправить запрос. При необходимости прикладная программа может задать ключ TSIG для протокола TSIG.

Если происходит ошибки при отправке или принятии DNS-записей, то функция возвращает ошибку.

Алгоритм функции DNS-запроса для прикладных программ приведен на рисунке 8.

Прототип функции:

DNSAPI

DNS_STATUS

DNSAPI_CALL_TYPE

Dns_Query(

 In PCWSTR pszName,

 In WORD wType,

 In DWORD Options,

 _Inout_opt_ PIP4_ARRAY pDnsServers,

 _In_z_ PCWSTR wszTsigKey,

 _Outptr_result_maybenull_ PDNS_RECORD * ppQueryResults

);

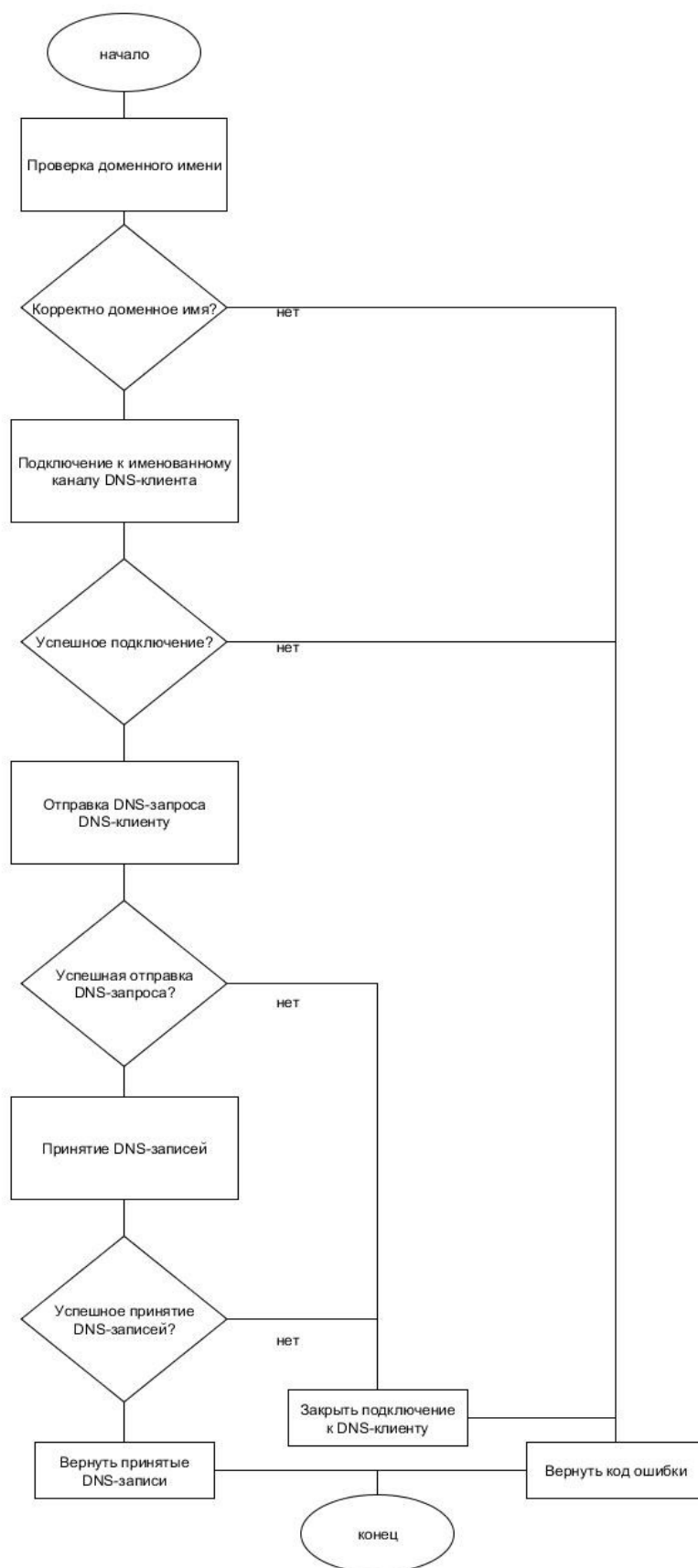


Рисунок 8 - Алгоритм функции DNS-запроса для прикладных программ

Код функции:

DNSAPI

DNS_STATUS

DNSAPI_CALL_TYPE

```
Dns_Query(
    _In_          PCWSTR      pszName,
    _In_          WORD        wType,
    _In_          DWORD        Options,
    _Inout_opt_   PIP4_ARRAY  pDnsServers,
    _In_z_         PCWSTR      wszTsigKey,
    _Outptr_result_maybenull_ PDNS_RECORD * ppQueryResults
)
{
    if ((pszName == NULL) || (ppQueryResults == NULL))
    {
        return ERROR_INVALID_PARAMETER;
    }

    // Проверка типа записи
    if (!IsDnsTypeSupport(wType))
    {
        SetLastError(DNS_ERROR_INVALID_TYPE);
        return DNS_ERROR_RCODE;
    }

    // Проверка опций
    if (!IsValidDnsOption((ULONG64)Options))
    {
```

```

        SetLastError(ERROR_INVALID_PARAMETER);
        return DNS_ERROR_RCODE;
    }
    *ppQueryResults = NULL;
    // DNS статус
    DNS_STATUS DnsStatus;
    // Проверка доменного имени
    DnsStatus = Dns_ValidateName_W(pszName, DnsNameHostnameFull);
    // если некорректное имя
    if ((DnsStatus != ERROR_SUCCESS) && (DnsStatus !=
DNS_ERROR_NON_RFC_NAME))
    {
        SetLastError(DNS_ERROR_RCODE_NAME_ERROR);
        return DnsStatus;
    }
    BYTE DnsCharSet = DnsCharSetUnicode;
    // перевод домена в ASCII
    char* szName = CopyUnicodeToAnsi(pszName);
    if (szName == NULL)
    {
        SetLastError(DNS_ERROR_NO_MEMORY);
        return DNS_ERROR_RCODE;
    }
    // перевод ключ в ASCII
    char* szKey = NULL;
    if (wszTsigKey)
    {
        szKey = CopyUnicodeToAnsi(wszTsigKey);
        if (szKey == NULL)

```

```

        {
            free(szName);
            SetLastError(DNS_ERROR_NO_MEMORY);
            return DNS_ERROR_RCODE;
        }
    }

    // выполнение DnsQuery
    DnsStatus = OsDnsQuery_Private(szName, wType, DnsCharSet,
Options, pDnsServers, szKey, ppQueryResults);
    free(szName);
    if (szKey)
    {
        free(szKey);
    }
    return DnsStatus;
}

```

Код функции OsDnsQuery_Private, в которой и происходит прием и отправка сообщений по именованному каналу:

```

DNS_STATUS
WINAPI
OsDnsQuery_Private(
_In_      char*      pszName,
_In_      WORD       wType,
_In_      BYTE       DnsCharSet,
_In_      DWORD      Options,
_Inout_opt_ PIP4_ARRAY pDnsServers,
_In_z_      char*      szTsigKey,
_Outptr_result_maybenull_ PDNS_RECORD * ppQueryResults)
{

```

```

// Размер буфера для отправки
int SizeBuffer = 0;
// буфер для отправки DNS-запроса службе DNS-клиента
BYTE* pBuffer = NULL;
// формируем буфер для отправки DNS-запроса службе DNS-
клиента
pBuffer = CreateBufferForSendToDnsResolver(pszName, DnsCharSet,
wType, (ULONG64)Options, 0, pDnsServers, 0, szTsigKey, &SizeBuffer);
if (pBuffer == NULL)
{
    return ERROR_INVALID_PARAMETER;
}
// Именованный канал
HANDLE hPipe = INVALID_HANDLE_VALUE;
// код ошибки
DWORD dwErrorCode = 0;
// подключение к именованному каналу DNS-клиента
if (!ConnectToDnsResolver(&hPipe))
{
    free(pBuffer);
    return DNS_ERROR_RCODE;
}
// событие получения ответа от DNS-клиента
HANDLE hDnsEvent = CreateEvent(NULL, TRUE, FALSE, NULL);
if (hDnsEvent == INVALID_HANDLE_VALUE)
{
    free(pBuffer);
    CloseHandle(hPipe);
    return DNS_ERROR_RCODE;
}

```

```

    }
    // Структура OVERLAPPED для операций
    OVERLAPPED sOverlapped;
    // Количество байт принятых или отправленных
    DWORD dwCountIoBytes = 0;
    // отправка запроса службе DNS-клиента
    if (!SendToDnsResolver(hPipe, pBuffer, SizeBuffer, &sOverlapped,
hDnsEvent, &dwCountIoBytes))
    {
        free(pBuffer);
        CloseHandle(hDnsEvent);
        CloseHandle(hPipe);
        return DNS_ERROR_RCODE;
    }
    // очищаем буфер для отправки DNS-клиенту
    free(pBuffer);
    // буфер для принятия размера ответа
    char BufferSize[4];
    // получение размера буфера от службы DNS-клиента
    if (!RecvFromDnsResolver(hPipe, &BufferSize, 4, &sOverlapped,
hDnsEvent, &dwCountIoBytes))
    {
        SetLastError(DNS_ERROR_NO_PACKET);
        CloseHandle(hDnsEvent);
        CloseHandle(hPipe);
        return DNS_ERROR_RCODE;
    }
    // размер буфера для записей
    DWORD* dwSize = (DWORD*)&BufferSize[0];

```

```

if ((*dwSize) > 80000)
{
    SetLastError(DNS_ERROR_NO_MEMORY);
    CloseHandle(hDnsEvent);
    CloseHandle(hPipe);
    return DNS_ERROR_RCODE;
}
// Буфер под DNS-записи
char* DnsResponseBuffer = malloc((*dwSize));
if (DnsResponseBuffer == NULL)
{
    SetLastError(DNS_ERROR_NO_MEMORY);
    CloseHandle(hPipe);
    CloseHandle(hDnsEvent);
    return DNS_ERROR_RCODE;
}
// принятие DNS-записей
if (!RecvFromDnsResolver(hPipe, DnsResponseBuffer, (*dwSize),
&sOverlapped, hDnsEvent, &dwCountIoBytes))
{
    SetLastError(DNS_ERROR_NO_PACKET);
    CloseHandle(hDnsEvent);
    CloseHandle(hPipe);
    return DNS_ERROR_RCODE;
}
CloseHandle(hDnsEvent);
CloseHandle(hPipe);
// Код статуса
DNS_STATUS DnsStatus = 0;

```

```

        // Парсинг ответа от службы и заполнение ppQueryResults
        DnsStatus    =    ParseResponseFromDnsResolver(DnsResponseBuffer,
dwCountIoBytes, ppQueryResults);
        if (DnsStatus != ERROR_SUCCESS)
        {
            free(DnsResponseBuffer);
        }
        return DnsStatus;
    }

```

7 Программная реализация защищенного DNS-клиента

7.1 Реализация главной функции

Был разработан контекст DNS-клиента. Структура контекста:

```
typedef struct _SDnsResolverContext
{
    // порт завершения
    HANDLE hCompletionPort;
    // имя компьютера
    char* szComputerName;
    // имя компьютера FQDN
    char* szComputerNameFQDN;
    // Хендл директории с файлом Hosts
    HANDLE hDirHosts;
    // Информация об интерфейсах: IP-адреса, DNS-серверы
    PSInterfaceInfo pInterfacesInfo;
    // Информация о кеше Hosts файла
    SInfoHostsFile SHostsFileInfo;
    // хеш-таблица для DNS-кеша
    PSHashTable pTableDnsCache;
    // хеш-таблица для негативного DNS-кеша
    PSHashTable pTableDnsNegativeCache;
    // Основной DNS-суффикс
    char* szPrimaryDnsSuffix;
    // Хендл события изменения интерфейсов
    HANDLE hChangesInterfaces;
    // Очередь таймеров
    HANDLE hTimerQueue;
    // Настройки
```

```

BYTE bEdnsEnable;
BYTE bDnsCookie;
BYTE bHostsFileSecurity;
BYTE bTsigEnable;
BYTE bDnsNameRandomCase;
BYTE NameInDomain;
BYTE bDnsLogEnable;
BYTE bDnsServerRandom;
BYTE bRecalcChecksum;
// Ключ для файла Hosts
char* szHostsKey;

} SDnsResolverContext, *PSDnsResolverContext;

```

Для инициализации контекста DNS-клиент была написана функция `DnsResolverContextInit(PSDnsResolverContext pDnsResolverContext)`, для освобождения памяти - `DnsResolverContextFree(PSDnsResolverContext pDnsResolverContext)`.

В главной функции `main` происходит инициализация контекста DNS-клиента, открытие именованного канала, разбор входной строки с настройками DNS-клиента, инициализация контекста первой DNS-операции, получение информации об интерфейсах компьютера и запуск потоков для выполнения функций DNS-клиента. После выполнения выше перечисленных действий главный поток «засыпает» до появления критической ошибки в дочерних потоках.

Часть кода главной функции:

```

int wmain(int argc, wchar_t* argv[])
{
    srand(time(NULL));
    // Контекст DNS-клиента

```

```

        PSDnsResolverContext          pDnsResolverContext          =
malloc(sizeof(SDnsResolverContext));
        if (pDnsResolverContext == NULL)
        {
            return 1;
        }
        // инициализация контекста
        if (!DnsResolverContextInit(pDnsResolverContext))
        {
            return 1;
        }
        // Парсинг настроек
        ParseInputArgs(pDnsResolverContext, argv, argc);
        //пересчет хеша, если нужно
        if          ((pDnsResolverContext->bRecalcCheksum)          &&
(pDnsResolverContext->szHostsKey))
        {
            //пересчет хеша
        }
        // получение информации от адаптеров
        if (!GetInfoAdapters(&pDnsResolverContext->pInterfacesInfo))
        {
            return 1;
        }
        if          ((pDnsResolverContext->hCompletionPort          =
CreateIoCompletionPort(INVALID_HANDLE_VALUE, NULL, 0, 0)) == NULL)
        {
            return 1;
        }
    }

```

```

        // количество ядер
        DWORD dwNumberOfProcessors = GetNumberOfProcessors();
        // хендлы процессов
        HANDLE* hThreads = malloc(sizeof(HANDLE) *
dwNumberOfProcessors);
        if (hThreads == NULL)
        {
            WSACleanup();
            DnsResolverContextFree(pDnsResolverContext);
            return 1;
        }
        int i = 0;
        // Создаются рабочие нити для обслуживания завершившихся
запросов ввода-вывода порта. 1 процессор - 1 нить
        for (i = 0; i < (int)dwNumberOfProcessors; i++)
        {
            DWORD ThreadID;
            hThreads[i] = CreateThread(NULL, 0, IOWorkerThread,
pDnsResolverContext->hCompletionPort, 0, &ThreadID);
            if (hThreads[i] == NULL)
            {
                return 1;
            }
        }
        // чтение файла Hosts
        if (!InitReadFileHosts(pDnsResolverContext))
        {
            pDnsResolverContext->SHostsFileInfo.pRecords = NULL;
            pDnsResolverContext->SHostsFileInfo.pTable = NULL;
        }
    }
}

```

```

    }
    // создание именованного канала
    if (!CreateNewNamedPipe(pDnsResolverContext))
    {
        return 1;
    }
    wprintf(L"DNS-клиент успешно запущен\n");
    DnsLogInit();
    DnsLogPrint(LOG_LEVEL_INFO, "DNS-клиент успешно запущен");
    // ждем, когда хотя бы один рабочий поток завершит работу
    DWORD dwStatus = WaitForMultipleObjects(dwNumberOfProcessors,
hThreads, FALSE, INFINITE);
    if (dwStatus == WAIT_FAILED)
    {
        return 1;
    }
    // один из потоков завершился
    // завершение работы, очистка памяти
    return 0;
}

```

Для получения списка DNS-серверов и IP-адресов компьютера была написана функция `GetInfoAdapters`. Данная функция вызывает функцию Windows `GetAdaptersAddresses` и копирует нужную информацию в структуру для хранения информации об интерфейсе.

Структура с информацией об интерфейсе:

```

typedef struct _SInterfaceInfo
{
    // Следующий адаптер
    struct _SInterfaceInfo* m_pNext;

```

```

// Индекс адаптера IPv4
DWORD m_dwIndexv4;
// Индекс адаптера IPv6
DWORD m_dwIndexv6;
IFTYPE InterfaceType;
TUNNEL_TYPE TunnelType;
// IP-адреса интерфейса
PSAddressInfo m_pIpAddressList;
// Список DNS серверов на адаптере
PSDnsAddressInfo m_pDnsServerList;
// Список DNS серверов IPv6 на адаптере
PSDnsAddressInfo m_pDnsServerList6;
// DNS-суффикс интерфейса
char* m_pszDnsSuffix;
// Флаги
union {
    BYTE Flags;
    struct {
        BYTE Ipv4Enabled : 1;
        BYTE Ipv6Enabled : 1;
    };
};

} SInterfaceInfo, *PSInterfaceInfo;

```

Для чтения DNS-записей из файла hosts была написана функция UpdateHostsRecords. Если включена защитная мера «проверка целостности hosts», то перед копированием записей в кеш производится проверка контрольной суммы. Рядом с файлом hosts лежит файл с именем hosts.hash с контрольной суммой. Вычисляется хеш от самого файла hosts и проверяется с записанным в

файле hosts.hash. Используется хеш HMAC-SHA256 пользователя GitHub aperezdc[13].

Ниже приведен код функции UpdateHostsRecords:

```
UpdateHostsRecords(
LPPER_IO_OPERATION_HOSTS PerIoHostsData)
{
    // если включена ЗМ
    if (PerIoHostsData->DnsResolverContext->szHostsKey)
    {
        DWORD dwLenKey = strlen(PerIoHostsData-
>DnsResolverContext->szHostsKey);

        // вычисление хеша файла
        BYTE* pRealHash = GetHashHostsFile(PerIoHostsData-
>DnsResolverContext->szHostsKey, dwLenKey);

        // чтение хеша из файла hosts.hash
        BYTE* pHashInFile = ReadHashHostsFile();
        // сравнение хешей
        if (!CheckHashHosts(pHashInFile, pRealHash))
        {
            DnsLogPrint(LOG_LEVEL_ERROR, "Неверная подпись
у файла hosts");

            return FALSE;
        }
    }

    // копирование в буфер
}
```

Как уже ранее упоминалась в главе 2, основная работа DNS-клиента происходит в функции IOWorkerThread, в которой работают дочерние потоки. Для каждого типа внутренней операции была написана своя функция.

7.2 Реализация функции подключения к именованному каналу

Была разработана функция `IocpClientConnectedNamedPipe` для начала принятия ответов от прикладных программ. Данная функция вызывается, как только прикладная программа подключилась к именованному каналу.

Код функции:

```
IocpClientConnectedNamedPipe(
    _In_ HANDLE CompletionPort,
    _In_ DWORD dwBytesTransferred,
    _Inout_ LPPER_HANDLE_DATA PerHandleData,
    _Inout_ LPPER_IO_OPERATION_DATA PerIoData)
{
    // создание нового именованного канала
    if (!CreateNewNamedPipe(PerIoData->DnsResolverContext))
    {
        return IOCP_EXIT_THREAD;
    }
    // заполнение информации о следующей операции
    PerIoData->OperationType =
    READ_DNS_QUESTION_IN_NAMEDPIPE;
    // начинаем слушать сообщения от программы
    if (!NamedpipeRecv(PerHandleData->hPipe, PerHandleData->pBuffer,
        PerHandleData->dwBufferLength,      &(PerHandleData->dwCountRecvBytes),
        &(PerIoData->OverlappedPipe)))
    {
        return IOCP_EXIT_THREAD;
    }
    return IOCP_CONTINUE;
}
```

7.3 Реализация функции принятия DNS-запроса от программы

Была разработана функция `IocpReadDnsQuestionInNamedPipe` для принятия и обработки сообщения, которое было получено по именованному каналу.

Прототип функции:

`DNSTESTAPI`

`BYTE`

`DNSTESTAPI_CALL_TYPE`

```
IocpReadDnsQuestionInNamedPipe(  
    _In_ HANDLE CompletionPort,  
    _In_ DWORD dwBytesTransferred,  
    _Inout_ LPPER_HANDLE_DATA PerHandleData,  
    _Inout_ LPPER_IO_OPERATION_DATA PerIoData)
```

Сначала происходит копирование принятых данных в контекст DNS-операции.

Если DNS-запрос относится к локальным именам или адресам, то разрешение имени происходит локально. Если DNS-запрос не был разрешен локально, то выполняется поиск в кеше файла `hosts`. Если ответ на DNS-запрос отсутствует в файле `hosts`, то выполняется поиск в кеше DNS-ответов и в кеше негативных DNS-ответов.

После успешного разрешения имени или происходит отправка найденных DNS-записей прикладной программе, в противном случае начинается подготовка к отправке DNS-запроса по сети DNS-серверам.

Алгоритм локального разрешения доменного имени приведен на рисунке 9.

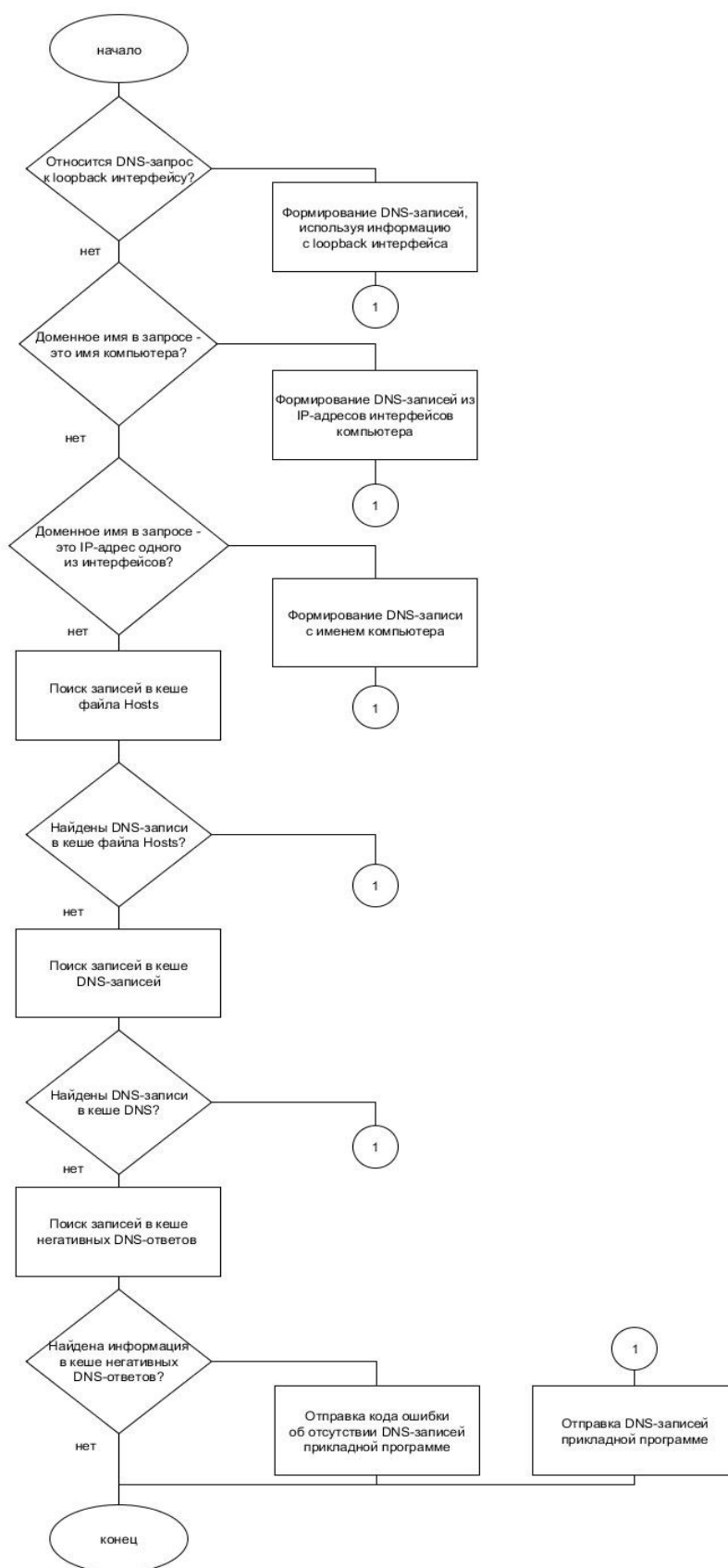


Рисунок 9 - Алгоритм локального разрешения доменного имени

Часть кода, касающаяся разрешения DNS-имени без отправки DNS-серверу, показана ниже:

```
DNSTESTAPI
BYTE
DNSTESTAPI_CALL_TYPE
IoctlReadDnsQuestionInNamedPipe(
    _In_ HANDLE CompletionPort,
    _In_ DWORD dwBytesTransferred,
    _Inout_ LPPER_HANDLE_DATA PerHandleData,
    _Inout_ LPPER_IO_OPERATION_DATA PerIoData)
{
    // принятие DNS-запроса от программы и парсинг
    // принятый буфер от программы
    PSBufferForDnsResolver          pInBuffer          =
(PBufferForDnsResolver)PerHandleData->pBuffer;

    // копирование данных, полученных от прикладной программы, в
контекст
    if (!CopyDnsDataToContext(PerIoData, pInBuffer))
    {
        return IOCP_EXIT_THREAD;
    }
    // Длина домена
    int nLenName = strlen(PerIoData->QueryName);
    // Удаление точки в конце
    RemoveLastDotInName(PerIoData->QueryName, &nLenName);

    // необходимо ли искать DNS-записи в списке локальных имен
    if (IsNeedQueryLocalName(PerIoData->QueryOptions))
    {
```

```

        // проверка на loopback
        if (IsLoopbackQuery(PerIoData->QueryName, PerIoData-
>QueryType))
        {
            // записи с loopback
            PDNS_RECORD pLoopbackRecords =
GetLoopbackRecords(PerIoData->DnsResolverContext->szComputerNameFQDN,
PerHandleData->pBuffer, PerHandleData->dwBufferLength, &PerHandleData-
>dwCountSendBytes, PerIoData->QueryName, PerIoData->QueryType, PerIoData-
>CharSet);

            if (pLoopbackRecords == NULL)
            {
                return IOCP_EXIT_THREAD;
            }
            PDNS_RECORD pRec = PerHandleData->pBuffer;
            if (wcscmp(pRec->pName, L"localhost") != 0)
            {
                return IOCP_CLOSE_OPERATION;
            }
            // отправка записей программе
            BYTE bRetSend = ProcessSendingRecords(PerIoData,
PerHandleData, pLoopbackRecords);
            return bRetSend;
        }
        // проверка на А или АААА-записи с интерфейсов
        else if (IsInterfacesAorAAAAQuery(PerIoData->QueryName,
PerIoData->QueryType, PerIoData->DnsResolverContext-
>szComputerNameFQDN))
        {

```

```

        // Записи с интерфейсов
        PDNS_RECORD                pInterfacesRecords                =
GetInterfacesAandAAAARecords(
        PerIoData->DnsResolverContext->pInterfacesInfo,
        PerHandleData->pBuffer,
        PerHandleData->dwBufferLength,
        &PerHandleData->dwCountSendBytes,
        PerIoData->QueryName,
        PerIoData->QueryType,
        PerIoData->CharSet);
    if (pInterfacesRecords != NULL)
    {
        // отправка записей программе
        BYTE                bRetSend                =
ProcessSendingRecords(PerIoData, PerHandleData, pInterfacesRecords);
        return bRetSend;
    }
}
else if (PerIoData->QueryType == DNS_TYPE_PTR)
{
    // пробуем найти PTR-запись
    PDNS_RECORD                pInterfacePtrRecord                =
GetInterfacePtrRecord(
        PerIoData->DnsResolverContext-
>szComputerNameFQDN,
        PerIoData->DnsResolverContext->pInterfacesInfo,
        PerHandleData->pBuffer,
        PerHandleData->dwBufferLength,
        &PerHandleData->dwCountSendBytes,

```

```

        PerIoData->QueryName,
        nLenName,
        PerIoData->QueryType,
        PerIoData->CharSet);
    if (pInterfacePtrRecord != NULL)
    {
        // отправка записей программе
        BYTE bRetSend =
ProcessSendingRecords(PerIoData, PerHandleData, pInterfacePtrRecord);
        return bRetSend;
    }
}

if (PerIoData->DnsResolverContext->SHostsFileInfo.pRecords !=
NULL)
{
    // необходимо ли искать DNS-записи в файле Hosts
    if (IsNeedQueryHosts(PerIoData->QueryOptions))
    {
        PDNS_RECORD pHostsRecords =
HostsFile_Query(PerIoData, PerHandleData, PerIoData->QueryName, PerIoData-
>QueryType);

        if (pHostsRecords != NULL)
        {
            // отправка записей программе
            BYTE bRetSend =
ProcessSendingRecords(PerIoData, PerHandleData, pHostsRecords);
            return bRetSend;

```

```

        }
    }
}

// смотрим в кеше, если серверы не назначены самим пользователем
if (PerIoData->pDnsServersCustom != NULL)
{
    if (IsNeedUseDnsCache(PerIoData->QueryOptions))
    {
        // поиск записей в DNS-кеше
        PDNS_RECORD pCacheRecords =
DnsCache_Query(PerIoData, PerHandleData, PerIoData->QueryName, PerIoData-
>QueryType);

        if (pCacheRecords != NULL)
        {
            // отправка записей программе
            BYTE bRetSend =
ProcessSendingRecords(PerIoData, PerHandleData, pCacheRecords);
            return bRetSend;
        }
        if (DnsNegativeCache_Query(PerIoData, PerIoData-
>QueryName, PerIoData->QueryType))
        {
            PerIoData->QueryStatus =
DNS_INFO_NO_RECORDS;

            // отправка кода об отсутствии записей
            BYTE bRetSend = ProcessSendingError(PerIoData,
PerHandleData);

            return bRetSend;
        }
    }
}

```

```

    }
}
...
}

```

Далее в функции `IocpReadDnsQuestionInNamedPipe` происходит формирование DNS-запроса для отправки DNS-серверу.

Если включена защитная мера «использование случайного регистра букв в имени домена при DNS-запросе», то доменное изменяется с помощью функции `RandomCaseQName`.

Код функции `RandomCaseQName`:

```

void RandomCaseQName(char* OriginalName, int nLenName)
{
    int i = 0, BYTE SeedXor = nLenName;
    for (i = 0; i < nLenName; i++)
        SeedXor = SeedXor ^ OriginalName[i];
    long lSeed = time(NULL) + nLenName + SeedXor ^ 0x89654;
    srand(lSeed);
    for (i = 0; i < nLenName; i++)
    {
        if (((OriginalName[i] >= 'a') && (OriginalName[i] <= 'z')) ||
            ((OriginalName[i] >= 'A') && (OriginalName[i] <= 'Z'))))
        {
            OriginalName[i] = (OriginalName[i] | 0x20) - (rand() &
0x20);
        }
    }
}

```

Алгоритм процесса формирования и отправки DNS-запроса приведен на рисунке 10.

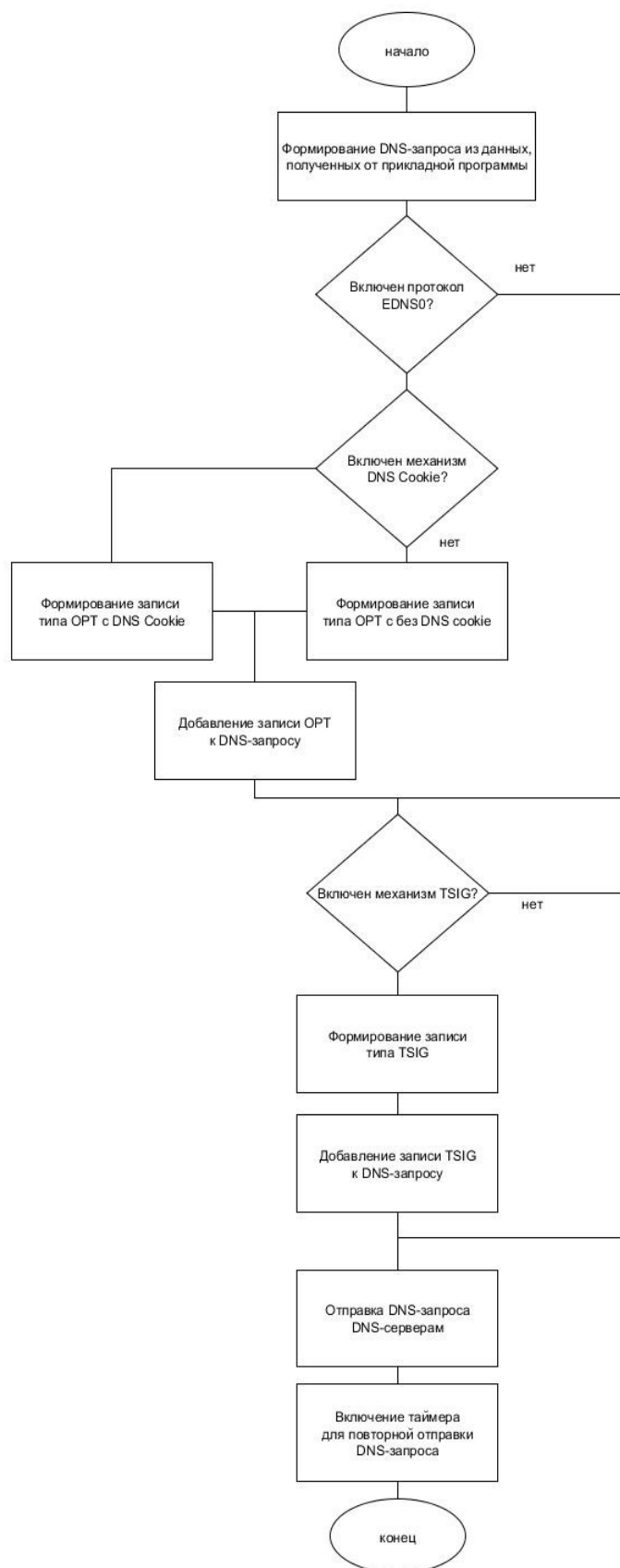


Рисунок 10 - Алгоритм процесса формирования и отправки DNS-запроса

Если включена защитная мера «использование протокола TSIG», то к отправляемому сообщению добавляется DNS-запись типа TSIG, которая содержит контрольную сумму DNS-запроса. Для вычисления контрольной суммы используется HMAC-SHA256.

Код функции добавления TSIG-записи показан ниже:

```
BOOL Dns_AddTsig(
    _Inout_ LPPER_HANDLE_DATA PerHandleData,
    _Inout_ LPPER_IO_OPERATION_DATA PerIoData,
    DWORD dwOffsetBuf)
{
    DWORD dwOffset = dwOffsetBuf;
    // TSIG-запись
    DNS_RECORD Record;
    Record.pName = malloc(255);
    strcpy(Record.pName, "secret-key");
    Record.Data.TSIG.pNameAlgorithm = malloc(100);
    strcpy(Record.Data.TSIG.pNameAlgorithm, "hmac-sha256");
    Record.Data.TSIG.cAlgNameLength = strlen("hmac-sha256");
    Record.Data.TSIG.wFudgeTime = 300;
    Record.Data.TSIG.wOriginalXid = PerIoData->Xid;
    Record.Data.TSIG.i64CreateTime = time(NULL);
    Record.Data.TSIG.wSigLength = 32;
    Record.Data.TSIG.pOtherData = NULL;
    Record.Data.TSIG.wOtherLength = 0;
    Record.Data.TSIG.wError = 0;
    // вычисление контрольной суммы
    Record.Data.TSIG.pSignature = GetMacRecords(PerHandleData-
        >pBuffer, &dwOffset, PerHandleData->dwBufferLength, &Record, PerIoData-
        >pKeyTsig, PerIoData->wLenKey);
}
```

```

        PerIoData->pRequestMac = malloc(32);
        memcpy(PerIoData->pRequestMac, Record.Data.TSIG.pSignature, 32);
        // добавление TSIG-записи к сообщению
        if (!Dns_AddTsigRecordToMessage(PerHandleData->pBuffer,
&dwOffset, PerHandleData->dwBufferLength, &Record))
        {
            if (Record.Data.TSIG.pSignature)
            {
                free(Record.Data.TSIG.pSignature);
            }
            return FALSE;
        }
        if (Record.Data.TSIG.pSignature)
        {
            free(Record.Data.TSIG.pSignature);
        }
        PerHandleData->dwCountSendBytes = dwOffset;
        PerHandleData->DataBuf.len = dwOffset;
        return TRUE;
    }

```

Если в DNS-клиенте включена защитная мера «использование DNS-cookie», то к сообщению добавляется DNS-запись OPT с DNS-cookie.

Код функции показан ниже:

```

BOOL Dns_AddDnsCookieToMessage(
    _Inout_ LPPER_HANDLE_DATA PerHandleData,
    _Inout_ LPPER_IO_OPERATION_DATA PerIoData,
    DWORD dwOffsetBuf)
{
    DWORD dwOffset = dwOffsetBuf;

```

```

PerIoData->pClientCookie = malloc(8);
if (PerIoData->pClientCookie == NULL)
{
    return FALSE;
}
int j = 0;
for (j = 0; j < 8; j++)
{
    PerIoData->pClientCookie[j] = rand() % 256;
}
DNS_RECORD_OPT2 pRecordOpt;
pRecordOpt.Name[0] = 0;
pRecordOpt.wDataLength = 12;
pRecordOpt.wPayloadSize = 512;
pRecordOpt.wType = DNS_TYPE_OPT;
pRecordOpt.Ttl = 0;
pRecordOpt.wOptionCode = 10;
pRecordOpt.wOptionLength = 8;
for (j = 0; j < 8; j++)
{
    pRecordOpt.ClientCookie[j] = PerIoData->pClientCookie[j];
}
// добавление OPT-записи к сообщению
if (!Dns_AddOptRecordToMessage(PerHandleData->pBuffer,
&dwOffset, PerHandleData->dwBufferLength, &pRecordOpt))
{
    return FALSE;
}
PerHandleData->dwCountSendBytes = dwOffset;

```

```

        PerHandleData->DataBuf.len = dwOffset;
        return TRUE;
    }

```

После того, как DNS-запрос готов, вызывается функция для непосредственной отправки DNS-запроса DNS-серверам.

Функция для отправки показан ниже:

```

BOOL
Dns_OpenUdpPortAndSend(
    _In_ HANDLE CompletionPort,
    _Inout_ LPPER_IO_OPERATION_DATA PerIoData,
    _Inout_ LPPER_HANDLE_DATA PerHandleData)
{
    // открытие сокета
    SOCKET SocketTemp = Dns_CreateUdpSocket(CompletionPort,
PerIoData, &PerIoData->Socket4Info, AF_INET);
    if (SocketTemp == INVALID_SOCKET)
    {
        return FALSE;
    }
    // открытие сокета
    SocketTemp = Dns_CreateUdpSocket(CompletionPort, PerIoData,
&PerIoData->Socket6Info, AF_INET6);
    if (SocketTemp == INVALID_SOCKET)
    {
        return FALSE;
    }
    // отправка серверам по UDP
    if (!SendUdpToDnsServers(PerIoData, PerHandleData,
ALL_SERVERS))

```

```

    {
        return FALSE;
    }
    // заполнение информации о следующей операции
    PerIoData->OperationType = RECEIVE_DNS_RESPONSE;
    int nLenSocket = GetSizeSocket(AF_INET);
    if (!Dns_RecvUdp(PerIoData->Socket4Info->Socket,
        (LPWSABUF)&PerIoData->Socket4Info->DataBuf,
        &PerIoData->Socket4Info->dwCountRecvBytes,
        PerIoData->Socket4Info->pSourceAddr,
        nLenSocket,
        (LPWSAOVERLAPPED)&(PerIoData->OverlappedSocket4)))
    {
        return FALSE;
    }
    nLenSocket = GetSizeSocket(AF_INET6);
    if (!Dns_RecvUdp(PerIoData->Socket6Info->Socket,
        (LPWSABUF)&PerIoData->Socket6Info->DataBuf,
        &PerIoData->Socket6Info->dwCountRecvBytes,
        PerIoData->Socket6Info->pSourceAddr,
        nLenSocket,
        (LPWSAOVERLAPPED)&(PerIoData->OverlappedSocket6)))
    {
        return FALSE;
    }
    return TRUE;
}

```

Если включена защитная мера «случайный выбор DNS-сервера», то внутри функции SendUdpToDnsServers выбирается случайный сервер.

Код функции:

```
SendUdpToDnsServers(
    _Inout_ LPPER_IO_OPERATION_DATA PerIoData,
    _Inout_ LPPER_HANDLE_DATA PerHandleData,
    _In_ BYTE bCountServers)
{
    // если включена функция - случайный сервер
    if (PerIoData->DnsResolverContext->bDnsServerRandom)
    {
        bCountServers = rand() % 2 + 1;
    }
    BOOL bRet;
    // Если пользователь назначил DNS-серверы
    if (PerIoData->pDnsServersCustom != NULL)
    {
        bRet = SendToDnsServersCustom(PerIoData, PerHandleData,
bCountServers);
    }
    else
    {
        // отправка DNS-серверам, которые назначены на адаптерах
        bRet = SendToDnsServersAdapters(PerIoData, PerHandleData,
bCountServers);
    }
    return bRet;
}
```

Часть кода функции IoctlReadDnsQuestionInNamedPipe, касающаяся разрешения DNS-имени по сети, показана ниже:

```
// устанавливать бит рекурсии или нет
```

```

BOOL fRecursionDesired = IsRecursiveQuery(PerIoData);
// ID DNS-запроса
PerIoData->Xid = rand();

if (PerIoData->DnsResolverContext->bDnsNameRandomCase == 1)
{
    // изменение регистра букв на случайный
    RandomCaseQName(PerIoData->QueryName, nLenName);
}

// Размер DNS-запроса
int DnsQuestionSize = PerHandleData->dwBufferLength;

// Протокол передачи
PerIoData->bProtocolSocket = IPPROTO_UDP;
// смотрим бит TCP в опциях запроса
if (IsUseOnlyTcpQuery(PerIoData))
{
    PerIoData->bProtocolSocket = IPPROTO_TCP;
}

// формирование DNS-запроса для отправки DNS-серверу
BOOL bCreateBuffer = OsDnsWriteQuestionToBuffer_A(
    (PDNS_MESSAGE_BUFFER)PerHandleData->pBuffer,
    PerIoData->bProtocolSocket,
    &DnsQuestionSize,
    PerIoData->QueryName,
    PerIoData->QueryType,
    PerIoData->Xid,
    fRecursionDesired);

```

```

        if (!bCreateBuffer)
        {
            return IOCP_EXIT_THREAD;
        }

        // если включен TSIG и задан ключ
        if ((PerIoData->DnsResolverContext->bTsigEnable) && (PerIoData-
>pKeyTsig))
        {
            // если включены DNS-куки
            if (PerIoData->DnsResolverContext->bDnsCookie)
            {
                if (!Dns_AddDnsCookieToMessage(PerHandleData,
PerIoData, DnsQuestionSize))
                {
                    return IOCP_CLOSE_OPERATION;
                }
            }
            // добавление подписи к DNS-запросу
            if (!Dns_AddTsig(PerHandleData, PerIoData, PerHandleData-
>DataBuf.len))
            {
                return IOCP_CLOSE_OPERATION;
            }
        }
        // если нужно использовать DNS-куки
        else if ((PerIoData->DnsResolverContext->bDnsCookie) &&
(PerIoData->DnsResolverContext->bEdnsEnable))
        {

```

```

        if (!Dns_AddDnsCookieToMessage(PerHandleData, PerIoData,
DnsQuestionSize))
        {
            return IOCP_CLOSE_OPERATION;
        }
    }
    // если необходимо добавить OPT-запись
    else if ((IsNeedOptRecord(PerIoData)) || (PerIoData-
>DnsResolverContext->bEdnsEnable))
    {
        if (!AddOptRecordWithoutData(PerIoData, PerHandleData,
UDP_BUFSIZE, &DnsQuestionSize))
        {
            return IOCP_CLOSE_OPERATION;
        }
    }
    else
    {
        PerHandleData->dwCountSendBytes = DnsQuestionSize;
        PerHandleData->DataBuf.len = DnsQuestionSize;
    }

    // заполнение информации об отправке DNS-запроса серверу
    PerIoData->OperationType = SEND_DNS_QUESTION_TO_SERVER;

    if (PerIoData->bProtocolSocket == IPPROTO_TCP)
    {
        if (!Dns_OpenTcpConnectionAndSend(CompletionPort,
PerHandleData, PerIoData, NULL))

```

```

    {
        return IOCP_CLOSE_OPERATION;
    }

    // таймер ожидания ответа
    if (!CreateTimerQueueTimer(&PerIoData->hTimer, PerIoData-
>DnsResolverContext->hTimerQueue, DnsTimerTimeoutCallback, PerIoData,
DNS_TIMER_TCP_TIMEOUT, INFINITE, WT_EXECUTEDefault))
    {
        return IOCP_EXIT_THREAD;
    }
}
else if (PerIoData->bProtocolSocket == IPPROTO_UDP)
{
    if (!Dns_OpenUdpPortAndSend(CompletionPort, PerIoData,
PerHandleData))
    {
        return IOCP_CLOSE_OPERATION;
    }

    // таймер для отправки повторных UDP дейтаграмм
    if (!CreateTimerQueueTimer(&PerIoData->hTimer, PerIoData-
>DnsResolverContext->hTimerQueue, DnsTimerTimeoutCallback, PerIoData, 1000,
INFINITE, WT_EXECUTEDefault))
    {
        return IOCP_EXIT_THREAD;
    }
}
return IOCP_CONTINUE;

```

}

7.4 Реализация функции обработки принятого DNS-ответа

Для обработки принятого DNS-ответа была написана функция `IocpReceivedDnsResponse`.

Алгоритм функции обработки принятого DNS-ответа приведен на рисунке 11.

Для начала у принятого DNS-ответа проверяется IP-адрес отправителя. Проверка осуществляется с помощью функции `IsValidSourceDnsServer`. Если IP-адрес отправителя не является IP-адресом одного из DNS-серверов, то DNS-клиент начинает ожидать другой DNS-ответ, а текущий пропускает.

Далее осуществляется разбор полученного ответа от DNS-сервера и формирование ответа программе с помощью функции `CreateResponseForNamedpipeClient`.

Для разбора DNS-ответа была разработана функция `Dns_ParseMessage`. В данной функции DNS-ответ проверяется на соответствие RFC, проверяется подпись TSIG, значение DNS-cookie и формируются DNS-записи.

Если разбор принятого DNS-сообщения прошел не успешно, то DNS-клиент начинает ожидать другой DNS-ответ. Если разбор был выполнен успешно, то полученные DNS-записи сохраняются в кеш, а если принят негативный DNS-ответ, то добавляется информация в кеш негативных ответов.

После этого выполняется отправка данных прикладной программе.

Сначала производится начальная проверка DNS-пакета: проверка ID, заголовка и секции запроса.

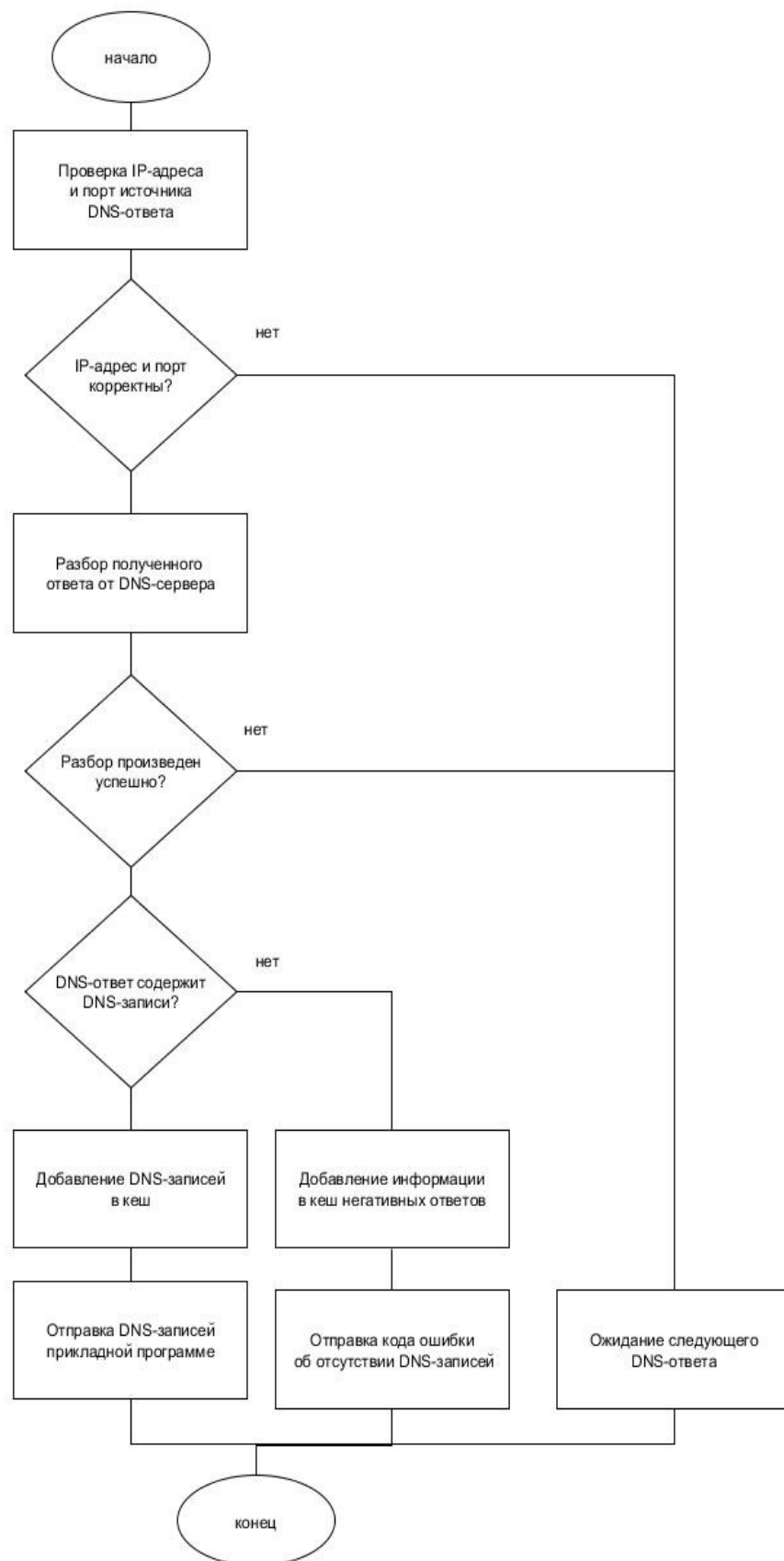


Рисунок 11 - Алгоритм функции обработки принятого DNS-ответа

Код функции проверки DNS-пакета:

```
/**
 * Функция проверки DNS-ответа
 * @param[in] pDnsPacketBuffer Указатель на буфер DNS-сообщения
 * @param[in] iDnsPacketBufferSize Размер DNS-сообщения
 * @param[in] wDnsId Идентификатор DNS-запроса, который был
отправлен
 * @param[in] szName Доменное имя, которое было отправлено
 * @param[in] wTypeRecord тип записи
 * @return TRUE или FALSE
 */
DNSTESTAPI
BOOL
DNSTESTAPI_CALL_TYPE
IsValidDnsResponse(
_In_ BYTE* pDnsPacketBuffer,
_In_ int iDnsPacketBufferSize,
_In_ WORD wDnsId,
_In_ char* szName,
_In_ WORD wTypeRecord,
_Out_ BOOL* bNeedReseiveResponse,
_Inout_ DWORD* pdwCodeError)
{
    *bNeedReseiveResponse = FALSE;
    if ((pDnsPacketBuffer == NULL) || (szName == NULL))
    {
        return FALSE;
    }
    //
```

```

// Проверка DNS-заголовка
//
// проверка на мин.размер пакета (заголовок + тип (2) + класс (2))
if (iDnsPacketBufferSize <= sizeof(DNS_HEADER) + 4)
{
    *pdwCodeError = DNS_ERROR_BAD_PACKET;
    DnsLogPrint(LOG_LEVEL_ERROR, "Получено DNS-
сообщение, несоответствующее RFC, для запроса %s (тип: %x)", szName,
wTypeRecord);
    return FALSE;
}
// DNS-заголовок
DNS_HEADER* pDnsHeader = (DNS_HEADER*)pDnsPacketBuffer;
// проверка ID
if (ntohs(pDnsHeader->Xid) != wDnsId)
{
    *pdwCodeError = DNS_ERROR_UNSECURE_PACKET;
    DnsLogPrint(LOG_LEVEL_ERROR, "Получено DNS-
сообщение с неверным ID для запроса %s (тип: %x)", szName, wTypeRecord);
    return FALSE;
}
// Если количество вопросов не равно одному
if ((ntohs(pDnsHeader->QuestionCount) != 1))
{
    *pdwCodeError = DNS_ERROR_BAD_PACKET;
    DnsLogPrint(LOG_LEVEL_ERROR, "Получено DNS-
сообщение несоответствующее RFC для запроса %s (тип: %x)", szName,
wTypeRecord);
    return FALSE;
}

```

```

    }

    // Проверка имени в секции "запрос"
    // смещение относительно начала датаграммы
    int OffsetDatagram = sizeof(DNS_HEADER);

    // чтение имени
    BYTE*   QName    =   Dns_ReadPacketName_A(pDnsPacketBuffer,
iDnsPacketBufferSize, &OffsetDatagram, NULL, NULL, 0, NULL, pdwCodeError);
    if (QName == NULL)
    {
        DnsLogPrint(LOG_LEVEL_ERROR,      "Получено   DNS-
сообщение, в котором секция запрос отличается от секции в DNS-запроса, для
запроса %s (тип: %x)", szName, wTypeRecord);
        return FALSE;
    }

    // проверка имени, которое было отправлено и то, которое пришло
    if (strcmp((char*)szName, (char*)QName) != 0)
    {
        DnsLogPrint(LOG_LEVEL_ERROR,      "Получено   DNS-
сообщение, в котором секция запрос отличается от секции в DNS-запроса, для
запроса %s (тип: %x)", szName, wTypeRecord);
        free(QName);
        *pdwCodeError = DNS_ERROR_UNSECURE_PACKET;
        return FALSE;
    }

    free(QName);

    // проверка кода ошибки
    if (pDnsHeader->ResponseCode != DNS_RCODE_NO_ERROR)
    {

```

```

        // ошибка в формате сообщения
        if (pDnsHeader->ResponseCode ==
DNS_RCODE_FORMAT_ERROR)
        {
            *pdwCodeError =
DNS_ERROR_RCODE_FORMAT_ERROR;
        }
        // отказ сервера, сервер не может выполнить запрос в
результате внутренних проблем
        else if (pDnsHeader->ResponseCode ==
DNS_RCODE_SERVER_FAILURE)
        {
            *pdwCodeError =
DNS_ERROR_RCODE_SERVER_FAILURE;
        }
        // ошибка в имени
        else if (pDnsHeader->ResponseCode ==
DNS_RCODE_NAME_ERROR)
        {
            *bNeedReseiveResponse = TRUE;
            *pdwCodeError =
DNS_ERROR_RCODE_NAME_ERROR;
        }
        // не реализовано, сервер не поддерживает данный тип
запросов
        else if (pDnsHeader->ResponseCode ==
DNS_RCODE_NOT_IMPLEMENTED)
        {

```

```

        *pdwCodeError =
DNS_ERROR_RCODE_NOT_IMPLEMENTED;
    }
    // отказ сервера, сервер отказался выполнять запрос,
    // например, по соответствию политике безопасности
    else if (pDnsHeader->ResponseCode ==
DNS_RCODE_REFUSED)
    {
        *pdwCodeError = DNS_ERROR_RCODE_REFUSED;
    }
    else if (pDnsHeader->ResponseCode ==
DNS_RCODE_YXDOMAIN)
    {
        *pdwCodeError = DNS_ERROR_RCODE_YXDOMAIN;
    }
    else if (pDnsHeader->ResponseCode ==
DNS_RCODE_YXRRSET)
    {
        *pdwCodeError = DNS_ERROR_RCODE_YXRRSET;
    }
    else if (pDnsHeader->ResponseCode ==
DNS_RCODE_NXRRSET)
    {
        *pdwCodeError = DNS_ERROR_RCODE_NXRRSET;
    }
    else if (pDnsHeader->ResponseCode ==
DNS_RCODE_NOTAUTH)
    {
        *pdwCodeError = DNS_ERROR_RCODE_NOTAUTH;
    }

```

```

        }
        else if (pDnsHeader->ResponseCode ==
DNS_RCODE_NOTZONE)
        {
            *pdwCodeError = DNS_ERROR_RCODE_NOTZONE;
        }
        else
        {
            // неизвестный код ошибки
            *pdwCodeError = DNS_ERROR_BAD_PACKET;
        }
        return FALSE;
    }
    *bNeedReseiveResponse = TRUE;
    // проверка запрос/ответ
    if (pDnsHeader->IsResponse != 1)
    {
        *pdwCodeError = DNS_ERROR_BAD_PACKET;
        DnsLogPrint(LOG_LEVEL_ERROR, "Получено DNS-
сообщение, несоответствующее RFC, для запроса %s (тип: %x)", szName,
wTypeRecord);
        return FALSE;
    }
    // проверка OPCODE
    if (pDnsHeader->Opcode != DNS_OPCODE_QUERY)
    {
        *pdwCodeError = DNS_ERROR_BAD_PACKET;
    }

```

```

        DnsLogPrint(LOG_LEVEL_ERROR, "Получено DNS-
сообщение, несоответствующее RFC, для запроса %s (тип: %x)", szName,
wTypeRecord);

        return FALSE;

    }

    // если выход за буфер
    if ((int)(OffsetDatagram + sizeof(DNS_WIRE_QUESTION)) >
iDnsPacketBufferSize)
    {
        *pdwCodeError = DNS_ERROR_UNSECURE_PACKET;

        DnsLogPrint(LOG_LEVEL_ERROR, "Получено DNS-
сообщение, несоответствующее RFC, для запроса %s (тип: %x)", szName,
wTypeRecord);

        return FALSE;

    }

    // проверка типа записи и класса
    PDNS_WIRE_QUESTION QInfo =
(PDNS_WIRE_QUESTION)&pDnsPacketBuffer[OffsetDatagram];

    if ((QInfo->QuestionClass != DNS_RCLASS_INTERNET) ||
(ntohs(QInfo->QuestionType) != wTypeRecord))
    {
        DnsLogPrint(LOG_LEVEL_ERROR, "Получено DNS-
сообщение, в котором секция запрос отличается от секции в DNS-запроса, для
запроса %s (тип: %x)", szName, wTypeRecord);

        *pdwCodeError = DNS_ERROR_BAD_PACKET;

        return FALSE;

    }

    return TRUE;

}

```

Для разбора конкретной DNS-записи была разработана функция Dns_RecordRead. Функция может безопасно разобрать следующие типы записей:

- узел IPv4 (тип A);
- узел IPv6 (тип AAAA);
- псевдоним (тип CNAME);
- ответственный DNS-сервер (тип NS);
- расположение службы (тип SRV);
- почтовый сервер (тип MX);
- указатель (тип PTR);
- текст (тип TXT).

В процессе разбора DNS-записи осуществляется проверка доменного имени. Если запись находится в секции ответов, то доменное имя должно совпадать с доменным именем из секции запроса. Если запись находится в другой секции, то проводится проверка относится ли доменное имя в записи к доменному имени из секции запроса, в этом случае последние части доменов должны совпадать.

Код функции проверки:

```
/**
 * Функция проверяет относится ли имя в ответе к имени в запросе или
нет (ASCII)
 * @param[in] szQuestionName Доменное имя из секции запроса
 * @param[in] nQuestionNameLen Размер доменного имени из секции
запроса
 * @param[in] szRecordName Размер доменного имени из записи
 * @param[in] nRecordNameLen Размер доменного имени из записи
 * @return TRUE или FALSE
 */
BOOL
```

```

IsRecordInDomain_A(
    _In_ BYTE* szQuestionName,
    _In_ int nQuestionNameLen,
    _In_ BYTE* szRecordName,
    _In_ int nRecordNameLen)
{
    // счетчик по имени в запросе
    int i = 0;

    // счетчик по имени в записи
    int j = 0;

    // индекс, где должна стоять точка
    int EndIndex = 0;
    if (nRecordNameLen > nQuestionNameLen)
    {
        EndIndex = nRecordNameLen - nQuestionNameLen - 1;
    }
    else if (nRecordNameLen < nQuestionNameLen)
    {
        EndIndex = nQuestionNameLen - nRecordNameLen - 1;
    }
    for (i = nQuestionNameLen - 1, j = nRecordNameLen - 1; i >= 0; i--, j--)
    {
        if (j < 0)
        {
            break;
        }

        if (szQuestionName[i] != szRecordName[j])
        {

```

```

        return FALSE;
    }
}
if (nRecordNameLen != nQuestionNameLen)
{
    if (nRecordNameLen > nQuestionNameLen)
    {
        if (szRecordName[EndIndex] != '.')
        {
            return FALSE;
        }
    }
    else
    {
        if (szQuestionName[EndIndex] != '.')
        {
            return FALSE;
        }
    }
}
return TRUE;
}

```

После проверка имени домена проверяется значения поля «длина данных». Проверяется не будет ли выхода за буфер DNS-сообщения при заданной длине данных.

Далее проверяется поле «тип записи» на предмет того, поддерживается ли тип записи или нет. Если тип записи не поддерживается, то DNS-клиент переходит в чтению следующий записи.

После проверяется поле «класс записи».

После вызывается одна из функции разбора поля «данные». Для каждого типа записи была разработана своя функция разбора.

Функция разбора SRV-записи показана ниже:

```
/**
 * Функция парсинга SRV-записи
 * @param[in, out] pRecord Указатель на DNS-запись
 * @param[in] pDnsPacketBuffer Указатель на буфер DNS-сообщения
 * @param[in] iDnsPacketBufferSize Размер DNS-сообщения
 * @param[in, out] pDnsPacketBufferOffset Указатель на смещение в буфере
DNS-сообщения
 * @param[in, out] pBufferForRecords Указатель на буфер для DNS-записей
 * @param[in] iBufferForRecordsSize Размер буфера для DNS-записей
 * @param[in, out] pBufferForRecordsOffset Указатель на смещение в
буфере для DNS-записей
 * @return TRUE или FALSE
 */
DNSTESTAPI
BOOL
DNSTESTAPI_CALL_TYPE
Srv_RecordRead(
    _Inout_          PDNS_RECORD pRecord,
    _In_             BYTE* pDnsPacketBuffer,
    _In_             int iDnsPacketBufferSize,
    _Inout_          int* pDnsPacketBufferOffset,
    _Inout_          BYTE* pBufferForRecords,
    _In_             int iBufferForRecordsSize,
    _Inout_          int* pBufferForRecordsOffset,
    _Inout_          PSHashTable pNameTable,
    _Inout_          DWORD* pdwCodeError)
```

```

{
    // проверка типа записи
    if (pRecord->wType != DNS_TYPE_SRV)
    {
        *pdwCodeError = DNS_ERROR_BAD_PACKET;
        return FALSE;
    }
    // проверка поля длина данных
    if ((pRecord->wDataLength < 9) || (pRecord->wDataLength > 256 + 8))
    {
        *pdwCodeError = DNS_ERROR_BAD_PACKET;
        return FALSE;
    }
    // Размер поля данных
    int Size = 6;
    // если выход за буфер
    if ((*pDnsPacketBufferOffset) + Size >= iDnsPacketBufferSize)
    {
        *pdwCodeError = DNS_ERROR_UNSECURE_PACKET;
        return FALSE;
    }
    // копирование приоритета, относительного веса, номера порта
    memcpy(&pRecord->Data.SRV.wPriority, pDnsPacketBuffer +
(*pDnsPacketBufferOffset), Size);
    *pDnsPacketBufferOffset = (*pDnsPacketBufferOffset) + Size;
    pRecord->Data.SRV.wPort = ntohs(pRecord->Data.SRV.wPort);
    pRecord->Data.SRV.wPriority = ntohs(pRecord->Data.SRV.wPriority);
    pRecord->Data.SRV.wWeight = ntohs(pRecord->Data.SRV.wWeight);
    pRecord->Data.SRV.Pad = 0;
}

```

```

        // длина имени
        BYTE LenName = 0;
        // чтение имени
        pRecord->Data.SRV.pNameTarget =
Dns_ReadPacketName(pDnsPacketBuffer, iDnsPacketBufferSize,
pDnsPacketBufferOffset, &LenName, pRecord->Flags.S.CharSet,
pBufferForRecords, iBufferForRecordsSize, pBufferForRecordsOffset,
pdwCodeError);
        if (pRecord->Data.SRV.pNameTarget == NULL)
        {
            return FALSE;
        }
        ToLowChars(pRecord->Data.SRV.pNameTarget, LenName, pRecord-
>Flags.S);
        // проверяем кеш имен
        pRecord->Data.SRV.pNameTarget = DnsCashNameFind(pRecord-
>Data.SRV.pNameTarget, LenName, pNameTable, pBufferForRecordsOffset,
pRecord->Flags.S);
        // новое значение
        pRecord->wDataLength = sizeof(DNS_SRV_DATA);
        return TRUE;
    }

```

Разбор SRV-записи начинается с проверки соответствия поля «длина данных» документам RFC.

Далее перед тем, начать копировать данные из полученного DNS-сообщения, осуществляется проверка выхода за буфер. После проверки данные необходимых полей копируются в структуру DNS-записи. Далее осуществляется смена порядка байт.

Потом вызывается функция чтения доменных имен из DNS-сообщений. В данной функции происходит формирование доменного имени, после проверяется будущее расположение точек, и далее доменное имя конвертируется из доменного имени с длинами частей имени в доменное имя с точками.

```
/**
 * Функция чтения доменного имени из DNS-пакета (ASCII)
 * @param[in] pDnsPacketBuffer Указатель на буфер DNS-сообщения
 * @param[in] iDnsPacketBufferSize Размер DNS-сообщения
 * @param[in, out] pDnsPacketBufferOffset Указатель на смещение в буфере
DNS-сообщения
 * @param[out] NameLength Указатель на длину прочитанного доменного
имени из DNS-сообщения
 * @param[in, out] pBufferForRecords Указатель на буфер для DNS-записей
(если NULL, то функция возвращает указатель на буфера, полученного malloc)
 * @param[in] iBufferForRecordsSize Размер буфера для DNS-записей
 * @param[in, out] pBufferForRecordsOffset Указатель на смещение в
буфере для DNS-записей
 * @return указатель на прочитанное доменное имя, иначе NULL
 */
BYTE*
Dns_ReadPacketName(
    _In_      BYTE* pDnsPacketBuffer,
    _In_      int iDnsPacketBufferSize,
    _Inout_   int* pDnsPacketBufferOffset,
    _Out_opt_ BYTE* NameLength,
    _Inout_opt_ BYTE* pBufferForRecords,
    _In_      int iBufferForRecordsSize,
    _Inout_opt_ int* pBufferForRecordsOffset,
```

```

_Inout_          DWORD* pdwCodeError)
{
    if ((*pDnsPacketBufferOffset) >= iDnsPacketBufferSize)
    {
        *pdwCodeError = DNS_ERROR_BAD_PACKET;
        return NULL;
    }
    // имя
    BYTE Name[256];
    // счетчик символов
    int j = 0;
    // индекс в буфере
    int i = *pDnsPacketBufferOffset;
    // использовалось ли смещение
    BYTE bUsedOffset = 0;
    // Использовалось ли смещение на прошлой букве
    BYTE bUsedOffsetPrev = 0;
    // счетчик смещений
    int nCountOffset = 0;
    while (pDnsPacketBuffer[i] != 0)
    {
        // если встретили смещение (2 старших бита равны единице),
а не символ
        if (IsSetBitsOffset(pDnsPacketBuffer[i]))
        {
            // если в прошлый раз было смещение и теперь снова.
обнаружение заикливания
            if (bUsedOffsetPrev == 1)
            {

```

```

        *pdwCodeError =
DNS_ERROR_UNSECURE_PACKET;
        return NULL;
    }
    // проверка, не выйдем ли за буфер
    if (i + 1 >= iDnsPacketBufferSize)
    {
        *pdwCodeError =
DNS_ERROR_UNSECURE_PACKET;
        return NULL;
    }
    // вычисление смещения
    i = (pDnsPacketBuffer[i] << 8) + pDnsPacketBuffer[i + 1] -
49152; // 49152 = 11000000 00000000
    // проверка, не вышли ли мы за буфер
    if (i >= iDnsPacketBufferSize)
    {
        *pdwCodeError =
DNS_ERROR_UNSECURE_PACKET;
        return NULL;
    }
    nCountOffset++;
    // если смещение встречалось слишком часто
    if (nCountOffset > 255)
    {
        *pdwCodeError =
DNS_ERROR_UNSECURE_PACKET;
        return NULL;
    }
}

```

```

        // если встретили первое смещение
        if (bUsedOffset == 0)
        {
            *pDnsPacketBufferOffset =
(*pDnsPacketBufferOffset) + 2;
            bUsedOffset = 1;
        }
        // устанавливаем флаг использования смещения
        bUsedOffsetPrev = 1;
    }
    // если встретили символ
    else
    {
        // сброс флага использования смещения
        bUsedOffsetPrev = 0;
        // если имя длинее 255 символов
        if (j > 255)
        {
            *pdwCodeError = DNS_ERROR_BAD_PACKET;
            return NULL;
        }

        Name[j] = pDnsPacketBuffer[i];
        i++;
        j++;

        // проверка, не вышли ли мы за буфер
        if (i >= iDnsPacketBufferSize)
        {

```

```

        *pdwCodeError
DNS_ERROR_UNSECURE_PACKET;
        return NULL;
    }
}
// считаем кол-во байт
if (bUsedOffset == 0)
{
    *pDnsPacketBufferOffset = (*pDnsPacketBufferOffset) + 1;
}
}
Name[j] = 0;
// если смещения не было, то прибавляем 1, так как в конце есть \0
if (bUsedOffset == 0)
{
    *pDnsPacketBufferOffset = (*pDnsPacketBufferOffset) + 1;
}
// Проверка точек в имени
if (!IsValidDnsNameWithNums((char*)&Name, j + 1))
{
    *pdwCodeError = DNS_ERROR_BAD_PACKET;
    return NULL;
}
// смена формата имени с числом байт на формат с точками
BYTE* szNameNewFormat = DnsNameChangeFormat((char*)&Name,
j,  DNSNAME_FORMAT_DOTS,  pBufferForRecords,  iBufferForRecordsSize,
pBufferForRecordsOffset);
if (szNameNewFormat == NULL)
{

```

```

        *pdwCodeError = DNS_ERROR_NO_MEMORY;
        return NULL;
    }
    // возвращаем длину имени
    if (NameLength != NULL)
    {
        if (j != 0)
        {
            *NameLength = j - 1;
        }
        else
        {
            *NameLength = 0;
        }
    }
    return szNameNewFormat;
}

```

Аналогичным образом происходит разбор остальных типов записей.

Полный код функции разбора DNS-записей:

PDNS_RECORD

```

Dns_RecordRead(
    _In_ BYTE* pDnsPacketBuffer,
    _In_ int iDnsPacketBufferSize,
    _Inout_ BYTE* pBufferForRecords,
    _In_ int iBufferForRecordsSize,
    _Inout_ int* pBufferForRecordsOffset,
    _In_ DNS_RECORD_FLAGS sFlags,
    _Inout_ int* pDnsPacketBufferOffset,
    _In_ BYTE* szDnsName,

```

```

_In_ int nDnsNameLen,
_Inout_ PSHashTable pNameTable,
_Inout_ DWORD* pdwCodeError,
_In_ BYTE bNameInDomain)
{
    if ((pDnsPacketBuffer == NULL) || (pDnsPacketBufferOffset == NULL)
|| (szDnsName == NULL))
    {
        return NULL;
    }
    BOOL bNeedAddNameToCach = FALSE;
    // доменное имя DNS-записи
    void* Name = NULL;
    // Длина имени
    BYTE nLenName = 0;
    // чтение имени
    Name = Dns_ReadPacketName(pDnsPacketBuffer,
iDnsPacketBufferSize, pDnsPacketBufferOffset, &nLenName, sFlags.CharSet,
NULL, 0, NULL, pdwCodeError);
    if (Name == NULL)
    {
        return NULL;
    }
    // чтение типа, класса, ttl, длины данных
    PDNS_WIRE_RECORD pWiredRecord =
Dns_WireRecordRead(pDnsPacketBuffer, iDnsPacketBufferSize,
pDnsPacketBufferOffset);
    if (pWiredRecord == NULL)
    {

```

```

        free(Name);

        *pdwCodeError = DNS_ERROR_BAD_PACKET;
        return NULL;
    }

    // имена должны совпадать
    if (sFlags.Section == DNSREC_ANSWER)
    {
        if (!DnsNamesCompare(szDnsName, nDnsNameLen, Name,
nLenName, sFlags))
        {
            free(Name);

            *pdwCodeError = DNS_ERROR_UNSECURE_PACKET;
            return NULL;
        }
    }

    // проверка, относится ли имя в ответе к имени в запросе
    else if (((sFlags.Section == DNSREC_AUTHORITY) || (sFlags.Section
== DNSREC_ADDITIONAL)) && (pWiredRecord->RecordType !=
DNS_TYPE_OPT) && (pWiredRecord->RecordType != DNS_TYPE_TSIG))
    {
        if ((bNameInDomain) && (!IsRecordInDomain(szDnsName,
nDnsNameLen, Name, nLenName, sFlags)))
        {
            free(Name);

            *pdwCodeError = DNS_ERROR_UNSECURE_PACKET;
            return NULL;
        }
    }

    // перевод в нижний регистр

```

```

ToLowChars(Name, nLenName, sFlags);
// доменное имя из кеша
char* NameCash = NULL;
// ищем имя в кеше имен
NameCash = HashTable_Find_Name(pNameTable, Name, sFlags);
// если имени нет, то его нужно добавить
if (NameCash == NULL)
{
    bNeedAddNameToCach = TRUE;
}
// проверка корректности длины данных
if (!IsValidDataLength(iDnsPacketBufferSize, pDnsPacketBufferOffset,
pWiredRecord->DataLength))
{
    free(Name);
    *pdwCodeError = DNS_ERROR_BAD_PACKET;
    return NULL;
}
// поддерживается ли данный тип
if (!IsDnsTypeSupport(pWiredRecord->RecordType))
{
    // пропускаем запись
    Dns_SkipRecord(pDnsPacketBufferOffset, pWiredRecord-
>DataLength);
    free(Name);
    *pdwCodeError = DNS_ERROR_INVALID_TYPE;
    return NULL;
}
// проверка поля Класс

```

```

        if (!IsValidDnsClass(pWiredRecord->RecordClass, pWiredRecord-
>RecordType))
        {
            free(Name);
            *pdwCodeError = DNS_ERROR_BAD_PACKET;
            return NULL;
        }

        // DNS-запись
        PDNS_RECORD pRecord = Dns_RecordBuild(pWiredRecord, sFlags,
pBufferForRecords, iBufferForRecordsSize, pBufferForRecordsOffset);
        if (pRecord == NULL)
        {
            free(Name);
            *pdwCodeError = DNS_ERROR_NO_MEMORY;
            return NULL;
        }
        // чтение данных конкретного типа записи
        switch (pRecord->wType)
        {
            case DNS_TYPE_A:
                if (!A_RecordRead(pRecord, pDnsPacketBuffer,
iDnsPacketBufferSize, pDnsPacketBufferOffset))
                {
                    free(Name);
                    *pdwCodeError = DNS_ERROR_BAD_PACKET;
                    return NULL;
                }
                break;

```

```

        case DNS_TYPE_AAAA:
            if (!Aaaa_RecordRead(pRecord, pDnsPacketBuffer,
iDnsPacketBufferSize, pDnsPacketBufferOffset))
            {
                free(Name);
                *pdwCodeError = DNS_ERROR_BAD_PACKET;
                return NULL;
            }
            break;

        case DNS_TYPE_SOA:
            if (!Soa_RecordRead(pRecord, pDnsPacketBuffer,
iDnsPacketBufferSize, pDnsPacketBufferOffset, pBufferForRecords,
iBufferForRecordsSize, pBufferForRecordsOffset, pNameTable, pdwCodeError))
            {
                free(Name);
                return NULL;
            }
            break;

        case DNS_TYPE_SRV:
            if (!Srv_RecordRead(pRecord, pDnsPacketBuffer,
iDnsPacketBufferSize, pDnsPacketBufferOffset, pBufferForRecords,
iBufferForRecordsSize, pBufferForRecordsOffset, pNameTable, pdwCodeError))
            {
                free(Name);
                return NULL;
            }
            break;

        case DNS_TYPE_TEXT:

```

```

        if (!Text_RecordRead(pRecord, pDnsPacketBuffer,
iDnsPacketBufferSize, pDnsPacketBufferOffset, pBufferForRecords,
iBufferForRecordsSize, pBufferForRecordsOffset, pdwCodeError))
        {
            free(Name);
            return NULL;
        }
        break;
    case DNS_TYPE_TSIG:
        if (!Tsig_RecordRead(pRecord, pDnsPacketBuffer,
iDnsPacketBufferSize, pDnsPacketBufferOffset, pBufferForRecords,
iBufferForRecordsSize, pBufferForRecordsOffset, pNameTable, pdwCodeError))
        {
            free(Name);
            return NULL;
        }
        break;
    case DNS_TYPE_OPT:
        if (!Opt_RecordRead(pRecord, pDnsPacketBuffer,
iDnsPacketBufferSize, pDnsPacketBufferOffset, pBufferForRecords,
iBufferForRecordsSize, pBufferForRecordsOffset, pdwCodeError))
        {
            free(Name);
            return NULL;
        }
        break;
    case DNS_TYPE_PTR:
    case DNS_TYPE_NS:
    case DNS_TYPE_CNAME:

```

```

        if (!Ptr_RecordRead(pRecord, pDnsPacketBuffer,
iDnsPacketBufferSize, pDnsPacketBufferOffset, pBufferForRecords,
iBufferForRecordsSize, pBufferForRecordsOffset, pNameTable, pdwCodeError))
        {
            free(Name);
            return NULL;
        }
        break;
    case DNS_TYPE_MX:
        if (!Mx_RecordRead(pRecord, pDnsPacketBuffer,
iDnsPacketBufferSize, pDnsPacketBufferOffset, pBufferForRecords,
iBufferForRecordsSize, pBufferForRecordsOffset, pNameTable, pdwCodeError))
        {
            free(Name);
            return NULL;
        }
        break;
    default:
        free(Name);
        *pdwCodeError = DNS_ERROR_INVALID_TYPE;
        return NULL;
        break;
    }

    pWiredRecord->DataLength = htons(pWiredRecord->DataLength);
    pWiredRecord->RecordClass = htons(pWiredRecord->RecordClass);
    pWiredRecord->RecordType = htons(pWiredRecord->RecordType);
    pWiredRecord->TimeToLive = htonl(pWiredRecord->TimeToLive);
    // если имя отсутствует в кеше
    if (bNeedAddNameToCach)

```

```

    {
        // добавляем имя в буфер
        pRecord->pName =
Dns_WriteDnsNameToBuffer(pBufferForRecords, iBufferForRecordsSize,
pBufferForRecordsOffset, Name, nLenName, (BYTE)sFlags.CharSet);
        if (pRecord->pName == NULL)
        {
            free(Name);
            *pdwCodeError = DNS_ERROR_NO_MEMORY;
            return NULL;
        }
        // добавляем указатель в кеш
        if (!HashTable_PushBack_Name(pNameTable, (char*)pRecord-
>pName, sFlags))
        {
            free(Name);
            return NULL;
        }
    }
    // если присутствует
    else
    {
        pRecord->pName = NameCash;
    }
    free(Name);
    return pRecord;
}

```

7.5 Реализация функции журналирования DNS-клиента

Для ведения журнала аудита DNS-клиента были разработаны функции: DnsLogInit, DnsLogPrint и DnsLogClose.

Функция DnsLogInit открывает или создает текстовый файл для журнала. Для удобства было решено создавать текстовый файл на каждый день. Функция DnsLogPrint записывает сообщение в журнал аудита. Функция DnsLogClose закрывает журнал аудита.

Код функций:

BOOL

DnsLogInit()

```
{
    wchar_t wszLogfile[MAX_PATH] = { 0 };
    // время сейчас
    time_t tsecs = time(NULL);
    struct tm *tstruct;
    tstruct = localtime((const time_t *)&tsecs);
    _snwprintf(wszLogfile, MAX_PATH - 1, L"%S_%04d%02d%02d.txt",
"DnsResolver", tstruct->tm_year + 1900, tstruct->tm_mon + 1, tstruct->tm_mday);
    g_LogFile = _w fopen(wszLogfile, L"a, ccs=UTF-16LE");
    if (g_LogFile == NULL)
    {
        return FALSE;
    }

    return TRUE;
}

// Функция записи строки в файл с логами
void DnsLogPrint(enum LogLevel LogLevelValue, char* szLogMessage, ...)
```

```

{
    if ((g_LogFile == NULL) || (szLogMessage == NULL))
    {
        return;
    }
    char buffer[4096];
    // время сейчас
    time_t tsecs = time(NULL);
    struct tm *tstruct;
    va_list args;
    tstruct = localtime((const time_t *)&tsecs);
    sprintf(buffer, "%04d/%02d/%02d %02d:%02d:%02d|", tstruct-
>tm_year + 1900, tstruct->tm_mon + 1, tstruct->tm_mday, tstruct->tm_hour, tstruct-
>tm_min, tstruct->tm_sec);
    switch (LogLevelValue)
    {
    case LOG_LEVEL_INFO:
        strcat(buffer, "INFO: ");
        break;
    case LOG_LEVEL_WARNING:
        strcat(buffer, "WARNING: ");
        break;
    case LOG_LEVEL_ERROR:
        strcat(buffer, "ERROR: ");
        break;
        break;
    }
    va_start(args, szLogMessage);

```

```

        vsnprintf((char*)&buffer[strlen((char*)buffer)],      sizeof(buffer)      -
strlen((char*)buffer) - 1, szLogMessage, args);
        va_end(args);

        int nCount = fwprintf(g_LogFile, L"%S\n", buffer);
        fflush(g_LogFile);
    }
    void
    DnsLogClose()
    {
        if (g_LogFile != NULL)
        {
            fclose(g_LogFile);
            g_LogFile = NULL;
        }
    }
}

```

8 Проверка работоспособности DNS-клиента

Для проверки работоспособности DNS-клиента было написано консольное приложение DNSLookup. В приложении вызывается функция Dns_Query библиотеки DnsApiLib.dll, и выводится DNS-ответ на экран.

Программа DNSLookup имеет следующие параметры запуска: -type, -tsigkey, -name, -dnssrv. С помощью данных параметров можно задать тип запроса, ключ для TSIG, доменное имя, IPv4-адрес DNS-сервера.

Для тестирования типов записи был запущен DNS-клиент без задания дополнительных параметров. Успешный запуск показан на рисунке 12.

```
D:\Diplom2\Diplom\DnsResolver2\WinDebug>DnsClient.exe
DNS-клиент успешно запущен
```

Рисунок 12 - Успешный запуск DNS-клиенте

Было проведено тестирование локального разрешения имен. «DESKTOP-UUUDSGE» - это имя компьютера. Результаты тестирования показаны на рисунке 13.

```
D:\Diplom2\Diplom\NsLookup1\WinDebug>DNSLookup.exe -type=A -name=localhost
A localhost IP адрес = 127.0.0.1

D:\Diplom2\Diplom\NsLookup1\WinDebug>DNSLookup.exe -type=AAAA -name=localhost
AAAA localhost IP адрес = ::1

D:\Diplom2\Diplom\NsLookup1\WinDebug>DNSLookup.exe -type=A -name=DESKTOP-UUUDSGE
A DESKTOP-UUUDSGE IP адрес = 192.168.56.1
A DESKTOP-UUUDSGE IP адрес = 192.168.1.101
A DESKTOP-UUUDSGE IP адрес = 169.254.229.171
A DESKTOP-UUUDSGE IP адрес = 192.168.186.1
A DESKTOP-UUUDSGE IP адрес = 192.168.202.1

D:\Diplom2\Diplom\NsLookup1\WinDebug>DNSLookup.exe -type=AAAA -name=DESKTOP-UUUDSGE
AAAA DESKTOP-UUUDSGE IP адрес = fe80::dda1:21aa:cffc:d908
AAAA DESKTOP-UUUDSGE IP адрес = fe80::b0:d4f2:8ce5:3819
AAAA DESKTOP-UUUDSGE IP адрес = fe80::ece7:d432:60d2:e5ab
AAAA DESKTOP-UUUDSGE IP адрес = fe80::708d:5c7d:5786:61b3
AAAA DESKTOP-UUUDSGE IP адрес = fe80::ed53:6c37:f586:d05a
AAAA DESKTOP-UUUDSGE IP адрес = fe80::2887:ebff:afa0:d251
AAAA DESKTOP-UUUDSGE IP адрес = 2001:0:2851:782c:2887:ebff:afa0:d251

D:\Diplom2\Diplom\NsLookup1\WinDebug>DNSLookup.exe -type=PTR -name=1.0.0.127.in-addr.arpa
PTR 1.0.0.127.in-addr.arpa DESKTOP-UUUDSGE
```

Рисунок 13 - Проверка работоспособности разрешения локальных имен

После этого был запущен DNS-сервер BIND на виртуальной машине. DNS-сервер был настроен как первичный и содержал записи зоны test.ru. На DNS-сервере был установлен ключ TSIG: TesT123. IP-адрес сервера: 192.168.202.130. IP-адрес компьютера: 192.168.202.1.

Для анализа трафика была запущена программа Wireshark.

На рисунке 13 показан результат тестирования типов A и AAAA записей, так же возможность задать DNS-сервер. На рисунке 14 показаны DNS-сообщения в программе Wireshark.

```
D:\Diplom2\Diplom\NsLookup1\WinDebug>DNSLookup.exe -type=A -name=yandex.ru -dnssrv=8.8.8.8
A yandex.ru IP адрес = 5.255.255.88
A yandex.ru IP адрес = 77.88.55.80
A yandex.ru IP адрес = 77.88.55.88
A yandex.ru IP адрес = 5.255.255.5

D:\Diplom2\Diplom\NsLookup1\WinDebug>DNSLookup.exe -type=AAAA -name=yandex.ru
AAAA yandex.ru IP адрес = 2a02:6b8:a::a
```

Рисунок 14 - Результат тестирования типов A и AAAA записей

No.	Time	Source	Destination	Protocol	Length	Info
26	4.166249	192.168.1.101	8.8.8.8	DNS	69	Standard query 0x0029 A yandex.ru
27	4.196448	8.8.8.8	192.168.1.101	DNS	133	Standard query response 0x0029 A yandex.ru A 5.255.255.88 A 77.88.55.80 A 77.88.55.88 A 5.255.255.5
28	11.391443	192.168.1.101	192.168.1.1	DNS	69	Standard query 0x0029 AAAA yandex.ru
29	11.416017	192.168.1.1	192.168.1.101	DNS	97	Standard query response 0x0029 AAAA yandex.ru AAAA 2a02:6b8:a::a

Рисунок 15 - DNS-сообщения в Wireshark

Для дальнейшего тестирования был запущен DNS-сервер BIND на виртуальной машине.

Было проведено тестирование получения нескольких DNS-записей и получения типов SRV и NS. Результат показан на рисунке 15. На рисунке 16 показан DNS-ответ в программе Wireshark.

```
D:\Diplom2\Diplom\NsLookup1\WinDebug>DNSLookup.exe -type=SRV -name=_xmpp-server._tcp.test.ru -dnssrv=192.168.202.130
SRV _xmpp-server._tcp.test.ru
sip.xmppptest.ru порт:5269
Приоритет: 0
Вес: 0
NS test.ru DNS-сервер = ns1.test.ru
A ns1.test.ru IP адрес = 192.168.1.1
```

Рисунок 16 - Результат тестирования получения нескольких типов записей

```

▼ Domain Name System (response)
  Transaction ID: 0x0029
  > Flags: 0x8580 Standard query response, No error
  Questions: 1
  Answer RRs: 1
  Authority RRs: 1
  Additional RRs: 1
  > Queries
  ▼ Answers
    > _xmpp-server._tcp.test.ru: type SRV, class IN, priority 0, weight 0, port 5269, target sip.xmpptest.ru
  ▼ Authoritative nameservers
    > test.ru: type NS, class IN, ns ns1.test.ru
  ▼ Additional records
    > ns1.test.ru: type A, class IN, addr 192.168.1.1
  [Request In: 1]
  [Time: 0.000342000 seconds]

```

Рисунок 17 - DNS-ответ в Wireshark

Была проведена проверка работоспособности типа CNAME. Результат представлен на рисунке 17. На рисунке 18 показан DNS-ответ в программе Wireshark.

```

D:\Diplom2\Diplom\NsLookup1\WinDebug>DNSLookup.exe -type=CNAME -name=ibst.pnzgu.ru
CNAME ibst.pnzgu.ru псевдоним = beda.pnzgu.ru

```

Рисунок 18 - Проверка типа CNAME

```

▼ Domain Name System (response)
  Transaction ID: 0x0029
  > Flags: 0x8180 Standard query response, No error
  Questions: 1
  Answer RRs: 1
  Authority RRs: 0
  Additional RRs: 0
  ▼ Queries
    > ibst.pnzgu.ru: type CNAME, class IN
  ▼ Answers
    > ibst.pnzgu.ru: type CNAME, class IN, cname beda.pnzgu.ru
  [Request In: 18]
  [Time: 0.054137000 seconds]

```

Рисунок 19 - DNS-ответ в Wireshark

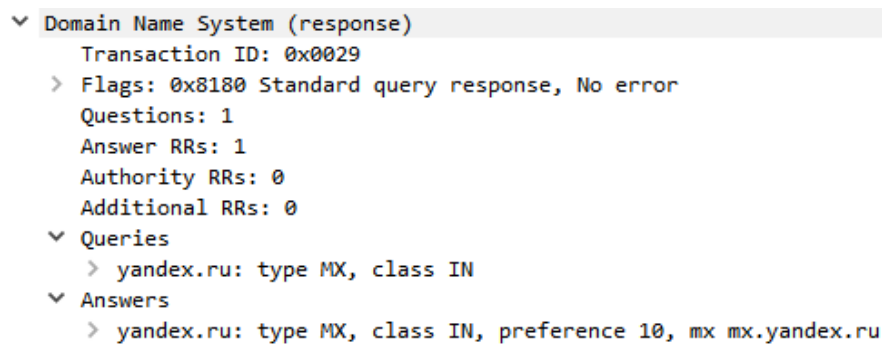
Была проведена проверка работоспособности типа MX. Результат представлен на рисунке 19. На рисунке 20 показан DNS-ответ в программе Wireshark.

```

D:\Diplom2\Diplom\NsLookup1\WinDebug>DNSLookup.exe -type=MX -name=yandex.ru
MX yandex.ru приоритет 10, почтовый сервер mx.yandex.ru

```

Рисунок 20 - Проверка типа MX



```

▼ Domain Name System (response)
  Transaction ID: 0x0029
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  ▼ Queries
    > yandex.ru: type MX, class IN
  ▼ Answers
    > yandex.ru: type MX, class IN, preference 10, mx mx.yandex.ru

```

Рисунок 21 - DNS-ответ в Wireshark

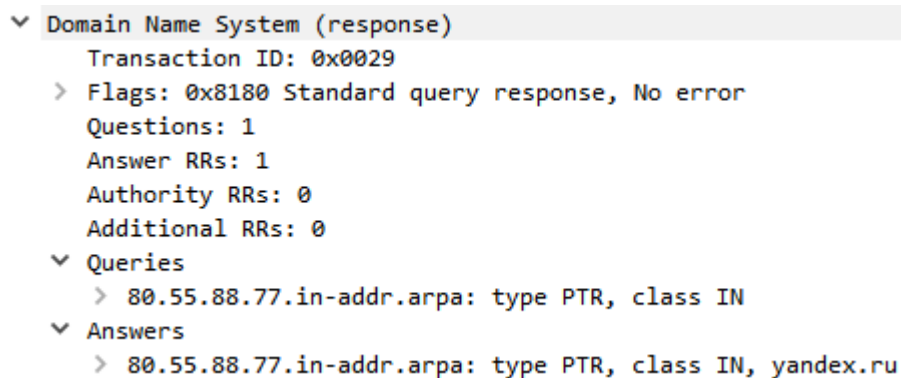
Была проведена проверка работоспособности типа PTR. Результат представлен на рисунке 21. На рисунке 22 показан DNS-ответ в программе Wireshark.

```

D:\Diplom2\Diplom\NsLookup1\WinDebug>DNSLookup.exe -type=PTR -name=80.55.88.77.in-addr.arpa -dnssrv=8.8.8.8
PTR 80.55.88.77.in-addr.arpa yandex.ru

```

Рисунок 22 - Проверка типа PTR



```

▼ Domain Name System (response)
  Transaction ID: 0x0029
  > Flags: 0x8180 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  ▼ Queries
    > 80.55.88.77.in-addr.arpa: type PTR, class IN
  ▼ Answers
    > 80.55.88.77.in-addr.arpa: type PTR, class IN, yandex.ru

```

Рисунок 23 - DNS-ответ в Wireshark

Была проведена проверка работоспособности типа NS. Результат представлен на рисунке 23. На рисунке 24 показан DNS-ответ в программе Wireshark.

```

D:\Diplom2\Diplom\NsLookup1\WinDebug>DNSLookup.exe -type=NS -name=yandex.ru -dnssrv=8.8.8.8
NS yandex.ru DNS-сервер = ns2.yandex.ru
NS yandex.ru DNS-сервер = ns9.z5h64q92x9.net
NS yandex.ru DNS-сервер = ns1.yandex.ru

```

Рисунок 24 - Проверка типа NS

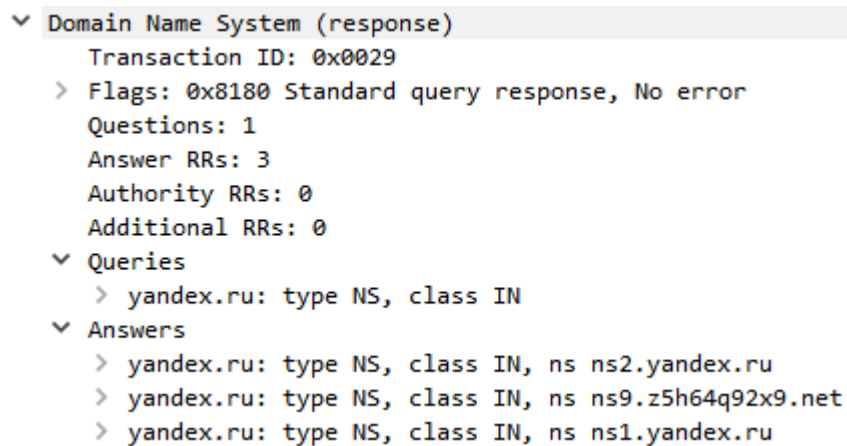


Рисунок 25 - DNS-ответ в Wireshark

Была проведена проверка работоспособности типа TXT. Результат представлен на рисунке 25. На рисунке 26 показан DNS-ответ в программе Wireshark.

```

D:\Diplom2\Diplom\NsLookup1\WinDebug>DNSLookup.exe -type=TXT -name=text.test.ru -dnssrv=192.168.202.130
TEXT text.test.ru
test 1234
test TXT
NS test.ru DNS-сервер = ns1.test.ru
A ns1.test.ru IP адрес = 192.168.1.1

```

Рисунок 26 - Проверка типа TXT

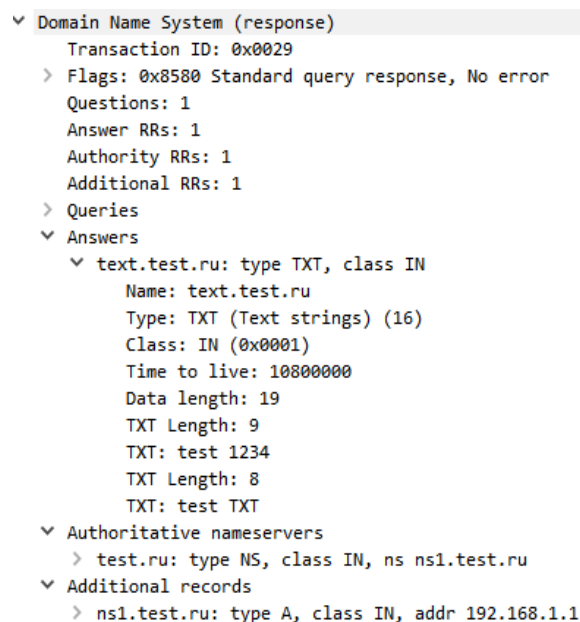


Рисунок 27 - DNS-ответ в Wireshark

У DNS-клиента был включен протокол EDNS0, Dns Cookie, подпись сообщений с помощью протокола TSIG, использование букв случайного

регистра в домене, проверка принадлежности имен в записях к доменному имени запроса. Результат запуска показан на рисунке 3.

```
D:\Diplom2\Diplom\DnsResolver2\WinDebug>DnsClient.exe -edns -dnscookie -tsig -dnsname_randomcase -name_in_domain
DNS-клиент успешно запущен
```

Рисунок 28 - Запуск DNS-клиента

Был выполнен запрос записей типа А с помощью протокола TSIG. Результат работы показан на рисунке 29.

```
D:\Diplom2\Diplom\NsLookup1\WinDebug>DNSLookup.exe -type=A -name=test.ru -dnssrv=192.168.202.130 -tsigkey=Test123
A test.ru IP адрес = 5.5.3.6
A test.ru IP адрес = 6.6.64.0
```

Рисунок 29 - Результат работы DNSLookup

Перехваченные пакеты показаны на рисунке 30.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.202.1	192.168.202.130	DNS	173	Standard query 0x0029 A TeSt.Ru OPT TSIG
2	0.000402	192.168.202.130	192.168.202.1	DNS	228	Standard query response 0x0029 A TeSt.Ru A 5.5.3.6 A 6.6.64.0 OPT TSIG

Рисунок 30 - Перехваченные пакеты

Содержимое DNS-запроса показано на рисунке 31.

```

Domain Name System (query)
  Transaction ID: 0x0029
  > Flags: 0x0380 Standard query
  Questions: 1
  Answer RRs: 0
  Authority RRs: 0
  Additional RRs: 2
  > Queries
  > TeSt.Ru: type A, class IN
  > Additional records
  > <Root>: type OPT
    Name: <Root>
    Type: OPT (41)
    UDP payload size: 512
    Higher bits in extended RCODE: 0x00
    EDNS0 version: 0
    > Z: 0x0000
    Data length: 12
    > Option: COOKIE
      Option Code: COOKIE (10)
      Option Length: 8
      Option Data: 3c366a991e08d016
      Client Cookie: 3c366a991e08d016
      Server Cookie: <MISSING>
  > secret-key: type TSIG, class ANY
    Name: secret-key
    Type: TSIG (Transaction Signature) (250)
    Class: ANY (0x00ff)
    Time to live: 0
    Data length: 61
    Algorithm Name: hmac-sha256
    Time Signed: Jun 6, 2019 01:52:52.000000000 RTZ 2 (зима)
    Fudge: 300
    MAC Size: 32
    > MAC
      Original Id: 41
      Error: No error (0)
      Other Len: 0

```

Рисунок 31 - Содержимое DNS-запроса

Содержимое DNS-ответа показано на рисунке 32.

```

Domain Name System (response)
  Transaction ID: 0x0029
  > Flags: 0x8580 Standard query response, No error
  Questions: 1
  Answer RRs: 2
  Authority RRs: 0
  Additional RRs: 2
  > Queries
  > TeSt.Ru: type A, class IN
  > Answers
  > test.ru: type A, class IN, addr 5.5.3.6
  > test.ru: type A, class IN, addr 6.6.64.0
  > Additional records
  > <Root>: type OPT
    Name: <Root>
    Type: OPT (41)
    UDP payload size: 4096
    Higher bits in extended RCODE: 0x00
    EDNS0 version: 0
  > Z: 0x0000
    Data length: 28
  > Option: COOKIE
    Option Code: COOKIE (10)
    Option Length: 24
    Option Data: 3c366a991e08d0169b19ca605cf847c459b694c61b4c164f
    Client Cookie: 3c366a991e08d016
    Server Cookie: 9b19ca605cf847c459b694c61b4c164f
  > secret-key: type TSIG, class ANY
    Name: secret-key
    Type: TSIG (Transaction Signature) (250)
    Class: ANY (0x00ff)
    Time to live: 0
    Data length: 61
    Algorithm Name: hmac-sha256
    Time Signed: Jun  6, 2019 01:52:52.000000000 RTZ 2 (зима)
    Fudge: 300
    MAC Size: 32

```

Рисунок 32 - Содержимое DNS-ответа

Были успешно получены DNS-записи. Подпись DNS-ответа верна, из этого следует, что записи не были изменены в процессе передачи, и что не было подмены IP-адреса DNS-сервера.

DNS-клиент имеет простой консольный интерфейс. Настройка DNS-клиента производится в момент запуска с помощью ключей запуска. После запуска изменять настройки нельзя.

Для того, чтобы пользователь мог правильно настроить механизмы защиты DNS-клиента, было разработано руководство пользователя DNS-клиента на основе диаграммы вариантов использования DNS-клиента. Руководство пользователя содержит информацию о ключах запуска DNS-клиента.

Руководство представлено в приложении Б.

10 Вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации. Методы и средства физической культуры, снижающие их воздействие

Психоэмоциональное состояние — это особая форма психических состояний человека с преобладанием эмоционального реагирования по типу доминанты.

На психоэмоциональное состояние специалиста по защите информации могут влиять следующие психофизиологические факторы:

- особенность темперамента;
- дефекты органов чувств;
- неудовлетворенность данным видом деятельности;
- профессиональная непригодность;
- неопытность;
- утомление;
- отсутствие мотивации;
- стресс;
- монотонная работа;
- умственные и нервно-эмоциональные перегрузки.

Для снижения выше описанных факторов необходим целый комплекс мер[14], а именно:

- активный образ жизни;
- правильное питание;
- здоровый сон;
- положительные эмоции.

Необходимо вести активный образ жизни, то есть иметь постоянные умеренные физические нагрузки. Они позволяют избавиться от стрессового состояния. Например, можно ходить пешком на работу, в выходные летом кататься на велосипеде, зимой — на лыжах или коньках.

Многочисленные исследования показывают, что правильное питание положительно сказывается на психоэмоциональном состоянии человека.[15]

Сон должен быть регулярным и длиться не менее 8 часов. Нарушение сна может быть причиной снижения работоспособности и депрессии специалиста по защите информации.

Радость и счастье - это надежные защитники психологического здоровья человека. Нужно всегда стараться находить источники радости и удовольствия в жизни.

Заключение

В ходе работы было произведено описание принципа работы системы DNS и DNS-клиента. Были идентифицированы и описаны угрозы информационной безопасности DNS-клиента. Были идентифицированы и описаны уязвимости DNS-клиента. Была произведена оценка рисков информационной безопасности DNS-клиента. Были описаны выбранные защитные меры DNS-клиента.

Была разработана архитектура DNS-клиента. Была разработана диаграмма вариантов использования и алгоритмы работы DNS-клиента.

Была выполнена программная реализация DNS-клиента, библиотека для работы с DNS-клиентом для прикладных программ и утилита DNSLookup для проверки работоспособности DNS-клиента.

С помощью разработанной утилиты DNSLookup была проведена проверка работоспособности DNS-клиента.

Было разработано руководство пользователя DNS-клиента.

Были изучены вредные психофизиологические факторы, влияющие на психоэмоциональное состояние специалиста по защите информации, а так же методы и средств физической культуры, снижающие их воздействие.

Таким образом, все пункты технического задания выполнены в полном объеме.

Список используемых источников

- 1 RFC 1034 — «Domain Names – concepts and facilities» - <https://tools.ietf.org/html/rfc1034> - режим доступа свободный, дата последнего обращения 10.05.2019
- 2 RFC 1035 — «Domain names – implementation and specification» - <https://tools.ietf.org/html/rfc1035> - режим доступа свободный, дата последнего обращения 10.05.2019
- 3 ФСТЭК России «Банк данных угроз безопасности информации» - [электронный ресурс] - <http://bdu.fstec.ru> - режим доступа свободный, дата последнего обращения 10.05.2019
- 4 Удаление вируса Редактируем файл HOSTS - <http://viruson.ru/remove/13-hosts-file.html> - режим доступа свободный, дата последнего обращения 10.05.2019
- 5 Надежность UDP протокола - https://tproger.ru/translations/udp_reliability/ - режим доступа свободный, дата последнего обращения 10.05.2019
- 6 Кибератаки в подробностях: вмешательство в работу DNS - <http://rus-linux.net/MyLDP/sec/cyber-attacks-dns-invasions.html> - режим доступа свободный, дата последнего обращения 10.05.2019
- 7 Уязвимости сетевых протоколов - <http://www.nestor.minsk.by/kg/2008/19/kg81912.html> - режим доступа свободный, дата последнего обращения 10.05.2019
- 8 ГОСТ Р ИСО/МЭК 27005-2010 Информационная технология (ИТ). Методы и средства обеспечения безопасности. Менеджмент риска информационной безопасности
- 9 RFC 5452 «Measures for Making DNS More Resilient against Forged Answers» - <https://tools.ietf.org/html/rfc5452> - режим доступа свободный, дата последнего обращения 10.05.2019

- 10 RFC 2845 «Secret Key Transaction Authentication for DNS» - <https://tools.ietf.org/html/rfc2845> - режим доступа свободный, дата последнего обращения 10.05.2019
- 11 RFC 7873 «Domain Name System (DNS) Cookies» - <https://tools.ietf.org/html/rfc7873> - режим доступа свободный, дата последнего обращения 10.05.2019
- 12 RFC Draft «Use of Bit 0x20 in DNS Labels to Improve Transaction Identity» - <https://tools.ietf.org/html/draft-vixie-dnsext-dns0x20-00> - режим доступа свободный, дата последнего обращения 26.02.2019
- 13 HMAC-SHA256 implemented in C - <https://github.com/aperezdc/hmac-sha256> - режим доступа свободный, дата последнего обращения 26.02.2019
- 14 Взаимосвязь психологического состояния и здоровья человека - <https://psyfactor.org/news/zdorovye4.htm> - режим доступа свободный, дата последнего обращения 10.05.2019
- 15 Открытие: питание человека напрямую влияет на его психическое здоровье - <http://meddaily.ru/article/18Sep2015/troisregim> - режим доступа свободный, дата последнего обращения 10.05.2019

Приложение А
(обязательное)
Диаграмма вариантов использования

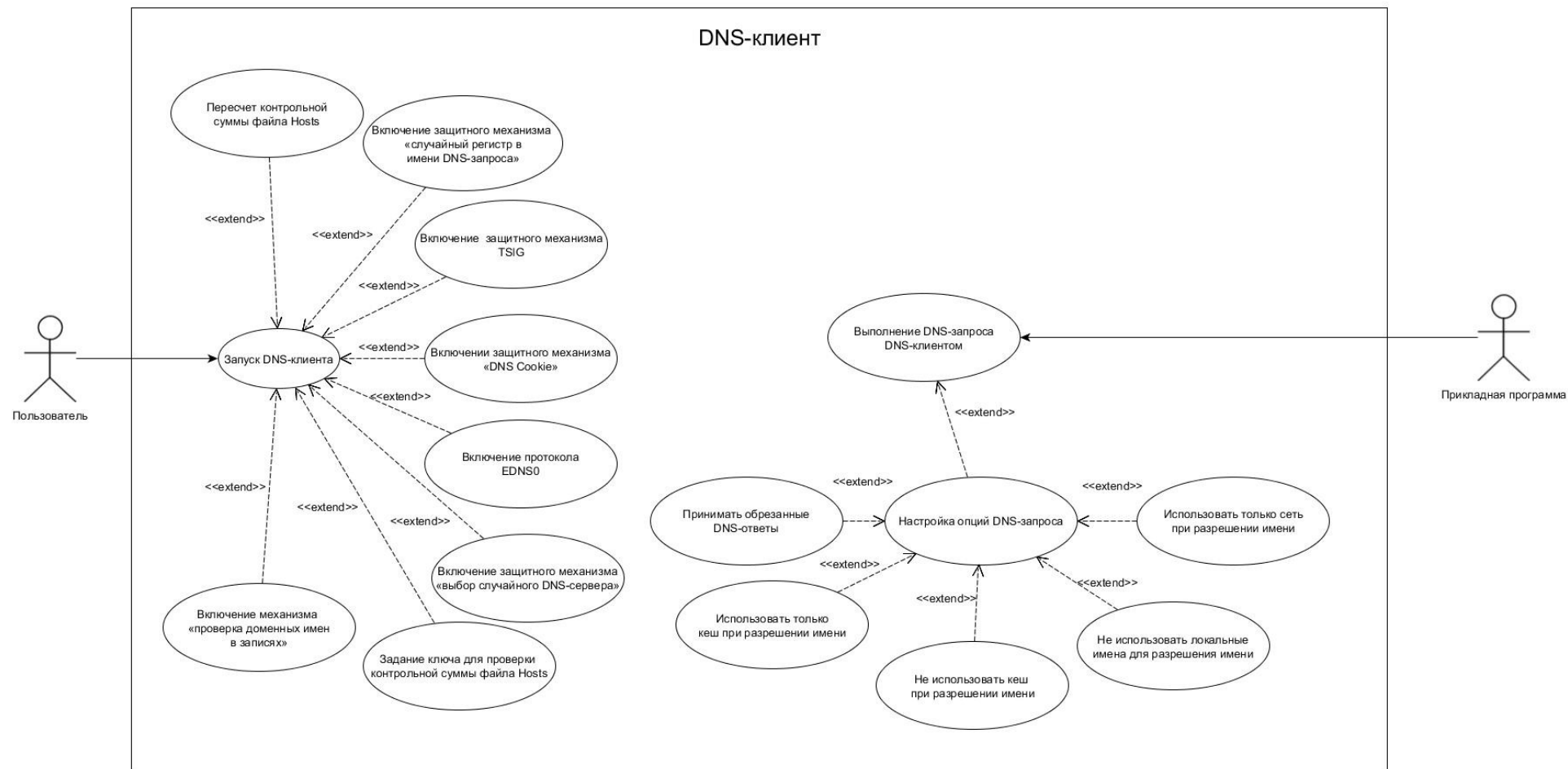


Рисунок А.1 - Диаграмма вариантов использования DNS-клиента

Приложение Б
(обязательное)
Руководство пользователя DNS-клиента

DNS-клиент — Руководство пользователя

Аннотация

В настоящем руководстве содержатся инструкции пользователю для настройки DNS-клиента.

Б.1 Возможности программы

DNS-клиент предназначен для безопасного выполнения DNS-запросов к DNS-серверам и получения DNS-ответов на них.

Возможности программы:

- выполнения DNS-запросов к DNS-серверам и получения DNS-ответов на них;
- получение DNS-записей из файла hosts;
- подпись файла hosts;
- получение локальных DNS-записей с интерфейсов;
- хранение полученных DNS-записей в кеше;
- хранение негативных DNS-ответов;
- поддержка протокола TSIG;
- поддержка DNS Cookies.

Б.2 Выполнение программы

При успешном запуске DNS-клиента появляется надпись: «DNS-клиент успешно запущен». После этого прикладные программы могут делать запросы к DNS-клиенту для разрешения доменных имен.

Б.3 Настройка DNS-клиента

Настройка DNS-клиента производится в консоли при запуске DNS-клиента. Список доступных для пользователя настроек показан в таблице Б.1.

Таблица Б.1 - Список настроек

Ключ	Описание
-edns	Включение протокола EDNS0. Включение данного протокола необходимо при включении защитного механизма «DNS Cookie»
-dnscookie	Включении защитного механизма «DNS Cookie»
-tsig	Включение защитного механизма TSIG. После этого DNS-сообщения будут иметь подпись.
-dnsname_randomcase	Включение механизма «случайный регистр в имени DNS-запроса»
-name_in_domain	Включение механизма «проверка доменных имен в записях»
dnssrv_random	Включение защитного механизма «выбор случайного DNS-сервера»
-hostskey:{ключ}	Задание ключа для проверки контрольной суммы файла Hosts
-recalc_checksum	Пересчет контрольной суммы файла Hosts при запуске DNS-клиента

Ключи необходимо вводить через пробел. По умолчанию все настройки выключены.

Пример настройки DNS-клиента: DnsClient.exe -edns -dnscookie -tsig