



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

*Загружаемый модуль ядра для контроля сетевого
трафика*

Студент ИУ7-72Б
(Группа)

(Подпись, дата) А. А. Зайцева
(И.О.Фамилия)

Руководитель курсового проекта

(Подпись, дата) Н.Ю. Рязанова
(И.О.Фамилия)

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ИУ7
(Индекс)
И.В. Рудаков
(И.О.Фамилия)
« » 20 г.

ЗАДАНИЕ
на выполнение курсового проекта

по дисциплине Операционные системы

Студент группы ИУ7-72Б

Зайцева Алена Андреевна
(Фамилия, имя, отчество)

Тема курсового проекта Загружаемый модуль ядра для контроля сетевого трафика.

Направленность КП (учебный, исследовательский, практический, производственный, др.)
учебный

Источник тематики (кафедра, предприятие, НИР) кафедра

График выполнения проекта: 25% к 4 нед., 50% к 7 нед., 75% к 11 нед., 100% к 14 нед.

Задание Разработать загружаемый модуль ядра для ОС Linux, осуществляющий контроль проходящего через него сетевого трафика и позволяющий блокировать пакеты по заданному списку правил. Предоставить пользователю возможность добавлять и удалять правила фильтрации пакетов. Предусмотреть защиту от подмены ip-адресов. Выводить информацию о DNS пакете, если обращение осуществляется по доменному имени.

Оформление курсового проекта:

Расчетно-пояснительная записка на 25-30 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

На защиту проекта должна быть представлена презентация. На слайдах должны быть: постановка задачи, использованные методы и алгоритмы, расчётные соотношения, структура комплекса программ, диаграмма классов, интерфейс, характеристики ПО, результаты исследования.

Дата выдачи задания « » 20 г.

Руководитель курсового проекта

Н.Ю. Рязанова
(Подпись, дата) (И.О.Фамилия)

Студент

А. А. Зайцева
(Подпись, дата) (И.О.Фамилия)

Оглавление

ВВЕДЕНИЕ	5
1 Аналитический раздел	6
1.1 Постановка задачи	6
1.2 Принципы работы сетевой подсистемы Linux	6
1.3 Библиотека Netfilter	8
1.4 Функции перехвата	9
1.5 DNS	12
1.5.1 Поле заголовка	13
1.5.2 Поле запроса	15
1.5.3 Поля ответа, полномочий и дополнительной информа- ции	15
1.6 Правила фильтрации	16
1.7 Выводы	17
2 Конструкторский раздел	18
2.1 IDEF0	18
2.2 Структура программного обеспечения	19
2.3 Алгоритм инициализации модуля	20
2.4 Алгоритм фильтрации сетевых пакетов	21
2.5 Алгоритм проверки совпадения правил	23
2.6 Алгоритм вывода информации о DNS-пакете	24
3 Технологический раздел	25
3.1 Выбор средств разработки	25
3.2 Инициализация модуля	25
3.3 Фильтрация сетевых пакетов	26
3.4 Вывод информации о DNS-пакете	28
4 Исследовательский раздел	30
4.1 Доступные команды и параметры	30
4.2 Тестирование фильтрации по ip-адресу	31
4.3 Тестирование фильтрации по протоколу	33
4.4 Вывод информации о DNS-пакетах	35

4.5	Тестирование запрета приема ответов от неавторитетных DNS-серверов	37
	Заключение	39
	Литература	40

ВВЕДЕНИЕ

Обеспечение сохранности данных в сети является одной из актуальных задач. Для обнаружения и предупреждения сетевых атак используются специальные программные или программно-аппаратные комплексы, ключевым компонентом которых является межсетевой экран.

Межсетевой экран, как правило, представляет собой часть сетевой подсистемы хоста в распределенной системе и реализует функции перехвата, анализа и модификации сетевых пакетов на основе заданной системы правил [1]. Разработке межсетевого и будет посвящена данная работа.

1 Аналитический раздел

1.1 Постановка задачи

В соответствии с заданием на курсовую работу необходимо разработать загружаемый модуль ядра для ОС Linux, осуществляющий контроль проходящего через него сетевого трафика и позволяющий блокировать пакеты по заданному списку правил. Также необходимо предоставить пользователю возможность добавлять и удалять правила фильтрации пакетов, предусмотреть защиту от подмены ip-адресов, выводить информацию о DNS пакете, если обращение осуществляется по доменному имени.

Для достижения поставленной цели необходимо решить следующие задачи:

- изучить принципы работы сетевой подсистемы Linux;
- проанализировать способы перехвата сетевых пакетов;
- ознакомиться со структурами и функциями ядра для работы с сетевыми пакетами;
- определить параметры, по которым будет осуществляться фильтрация пакетов;
- реализовать межсетевой экран в виде загружаемого модуля ядра;
- протестировать работоспособность разработанного межсетевого экрана.

1.2 Принципы работы сетевой подсистемы Linux

Сетевая подсистема Linux построена на стеке BSD. В этой подсистеме прием и передача данных на транспортном и сетевом уровнях модели OSI¹

¹OSI (Open Systems Interconnection) — концептуальная модель взаимодействия открытых систем, которая состоит из 7 взаимозависимых уровней, каждый из которых выполняет определенные функции [2]

происходят с помощью интерфейса сокетов.

Когда в сетевую подсистему приходит пакет, возникает прерывание. Обработчик прерывания от сетевого адаптера копирует пакет в ядро, где тот ставится в так называемую буферную очередь и ожидает обработки соответствующим потоком ядра, и вызывает соответствующий softirq, который будет запущен демоном ksoftirqd и закончит обработку данного сетевого пакета (рисунок 1.1) [3].

Для обработки сетевых пакетов, поступающих с высокоскоростных интерфейсов, в Linux был добавлен NAPI (New API). В этом API механизм прерываний сочетается с механизмом опроса, и его основная цель – сократить количество прерываний, генерируемых при получении пакета. В NAPI-совместимых драйверах при поступлении пакета прерывания отключаются, и обработчик только вызывает планировщик rx_scheduler, который гарантирует, что в дальнейшем обработка пакета будет выполнена [4].

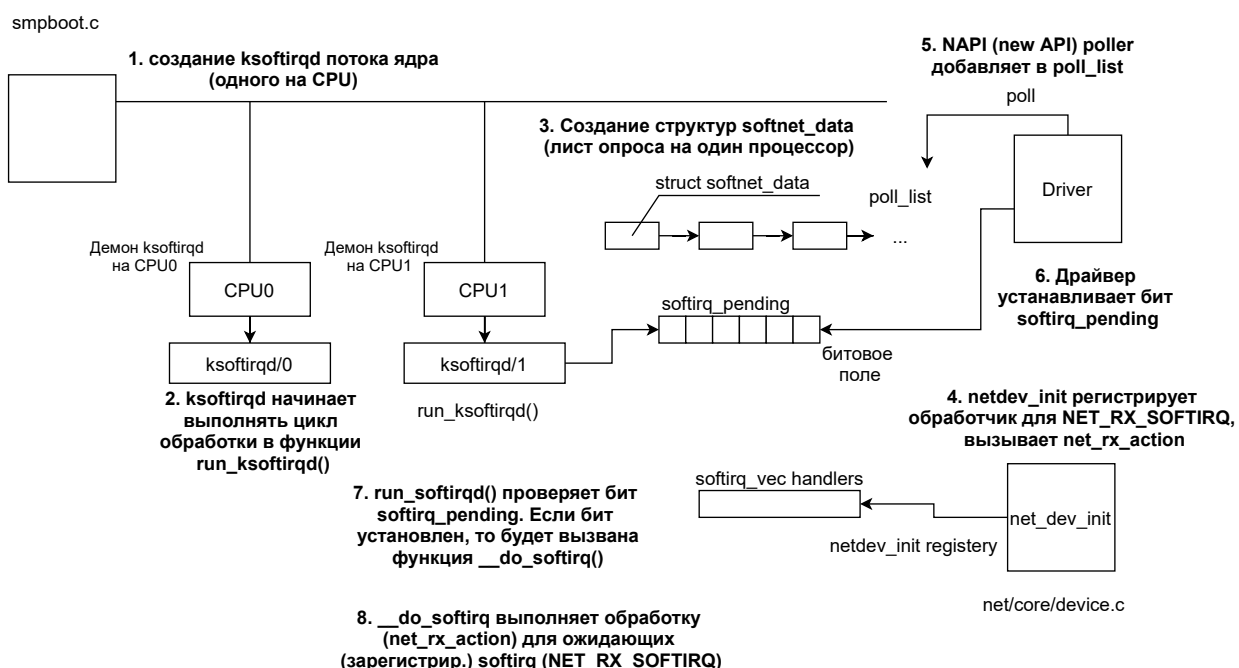


Рисунок 1.1 – Порядок выполняемых действий при приеме сетевого пакета

На рассматриваемых уровнях модели OSI и происходит фильтрация пакетов с учетом информации о протоколе и об IP-адресах и номерах портов источника и назначения. Широко распространённым средством фильтрации сетевых пакетов является библиотека Netfilter.

1.3 Библиотека Netfilter

Netfilter - это библиотека ядра Linux, предоставляющая функционал для фильтрации и перенаправления пакетов [6]. Netfilter представляет собой набор перехватчиков внутри ядра, позволяющий модулям ядра регистрировать функции обратного вызова в сетевом стеке. Эти функции вызываются для каждого пакета, который удовлетворяет соответствующему правилу перехвата [5].

Netfilter позволяет перехватить пакет в любой из пяти стандартных точек, через которые он проходит (рисунок 1.2):

1. `NF_INET_PRE_ROUTING` – все входящие пакеты;
2. `NF_INET_LOCAL_IN` – входящие пакеты, предназначенные для локального процесса;
3. `NF_INET_FORWARD` – транзитные пакеты (предназначенные для другого интерфейса);
4. `NF_INET_LOCAL_OUT` – исходящие пакеты, сформированные локальными процессами;
5. `NF_INET_POST_ROUTING` – все исходящие пакеты (поступившие из точки `NF_INET_FORWARD` или сформированные локальными процессами).

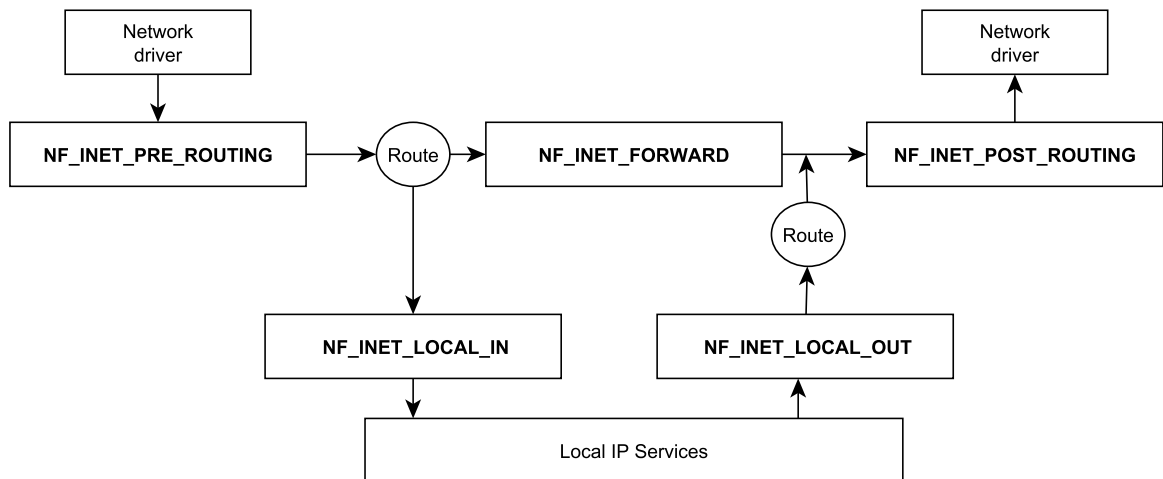


Рисунок 1.2 – Точки, через которые проходит сетевой пакет

1.4 Функции перехвата

Чтобы перехватить пакет в одной из перечисленных ранее точек, к ней необходимо подключить функцию перехвата (hook, ловушку). Для этого сначала необходимо заполнить структуру `nf_hook_ops`, основные поля которой приведены в листинге 1.1.

Листинг 1.1 – Структура `struct nf_hook_ops`

```

1 struct nf_hook_ops {
2     nf_hookfn    *hook;
3     ...
4     u8          pf;
5     ...
6     unsigned int hooknum;
7     int         priority;
8 };

```

Основные поля структуры:

- `hook` – функция, которая будет вызвана для обработки пакета и принятия решения: отбросить или принять пакет;
- `pf` – семейство протоколов (`PF_INET` для IPv4);

- `hooknum` – точка подключения функции;
- `priority` – приоритет (вводится, чтобы установить порядок вызова ловушек, подключенных к одной точке).

После заполнения структуры `nf_hook_ops` функцию перехвата необходимо зарегистрировать, вызвав функцию `nf_register_net_hook`. Для удаления ловушки необходимо вызвать функцию `nf_unregister_net_hook`. Прототипы этих функций приведены в листинге 1.2.

Листинг 1.2 – Функции для регистрации и удаления функций перехвата

```

1  int nf_register_net_hook(struct net *net, const struct
    nf_hook_ops *ops);
2
3  void nf_unregister_net_hook(struct net *net, const struct
    nf_hook_ops *ops);

```

Прототип функции перехвата приведен в листинге 1.3.

Листинг 1.3 – Прототип функции-ловушки

```

1  typedef unsigned int nf_hookfn(void *priv, struct sk_buff *skb,
    const struct nf_hook_state *state);

```

Аргументы функции:

- `priv` – код одной из пяти точек подключения ловушки;
- `skb` – указатель на структуру `sk_buff`, содержащую информацию о сетевом пакете;
- `state` – указатель на структуру `nf_hook_state`, содержащую информацию, связанную с перехватом пакета (интерфейс ввода/вывода, приоритет и т. д.).

Все сетевые пакеты в ядре представляются структурой `struct sk_buff`, поля которой приведены в листинге 1.4.

Листинг 1.4 – Структура `struct sk_buff`

```
1  struct sk_buff {
2      /* временная метка */
3      ktime_t      tstamp;
4
5      /* указатель на сокет, который отправил или получил этот пакет */
6      struct sock      *sk;
7
8      /* указатель на устройство, с которого получен или которому будет
   отправлен пакет */
9      struct net_device      *dev;
10
11     /* L3 протокол */
12     __be16      protocol;
13
14     /* смещения заголовков относительно head */
15     __u16      transport_header;
16     __u16      network_header;
17     __u16      mac_header;
18
19     /* указатели на конец и начало данных */
20     sk_buff_data_t      tail;
21     sk_buff_data_t      end;
22
23     /*
24     head — указатель на начало буфера выделенного под данные
25     data — указатель на начало данных.
26     */
27     unsigned char      *head, *data;
28
29     // счетчик ссылок
30     atomic_t      users;
31 };
```

1.5 DNS

Служба доменных имен (DNS, domain name system) – это стандартный протокол, выполняющий две основные функции:

- позволяет клиентским компьютерам запрашивать у DNS-сервера данные об IP-адресе или имени хоста в сети;
- позволяет производить обмен информацией между базами данных серверов DNS[9].

В этом протоколе используется стандартный формат типа «запрос-ответ», где клиент посылает пакет запроса, и сервер отвечает либо пакетом с информацией, полученной из базы данных, либо сообщением об ошибке, в котором указывается причина отказа в обработке запроса. Протокол DNS использует порт 53 и протоколы TCP или UDP.

Пакет DNS состоит из пяти полей: заголовка, запроса, ответа, полномочий и поля дополнительной информации. На рисунке 1.3 приведена общая структура DNS-пакета.

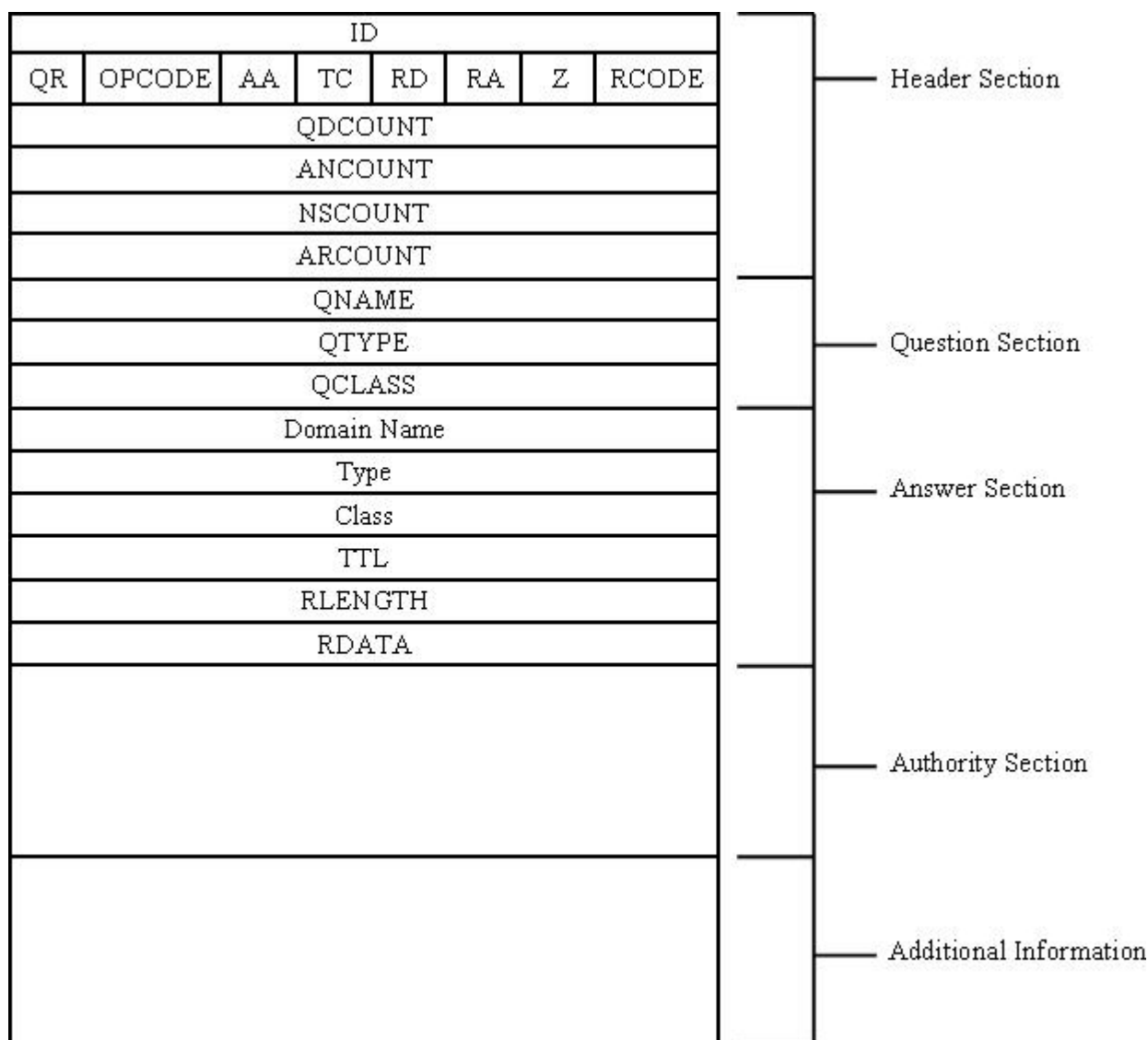


Рисунок 1.3 – Общая структура DNS-пакета.

1.5.1 Поле заголовка

В поле заголовка содержится информация о пакете, его назначении и количестве данных, содержащихся в каждом поле данных.

- Биты 0-15: ID – уникальный 16-битовый идентификационный номер пакета запроса. Пакет ответа, формируемый сервером, также использует этот идентификационный номер, чтобы клиент мог сопоставить ответ сервера со своим запросом.
- Бит 16: QR – тип пакета (0 – пакет запроса, 1 – пакет ответа).

- Биты 17-20: OPCODE – тип запроса: стандартный (0), обратный (1) или запрос о статусе сервера (2) (номера 3-15 зарезервированы на будущее).
- Бит 21: AA – устанавливается, когда ответ является авторитетным (то есть данные поступают напрямую от DNS-сервера, ответственного за зону). Неавторитетные ответы могут поступать от серверов DNS, в кэше которых сохранилась информация об исходных записях от предыдущих запросов.
- Бит 22: TC – устанавливается, если сервер не смог поместить всю необходимую информацию в пакет из-за существующих ограничений.
- Бит 23: RD – устанавливается в запросе и копируется в ответ, если клиент просит сервер не сообщать ему промежуточных ответов, а вернуть только IP-адрес.
- Бит 24: RA – устанавливается, чтобы уведомить клиента о возможности рекурсивного запроса на данный сервер.
- Биты 25-27: Z – в настоящее время не используются и зарезервированы на будущее.
- Биты 28-31: RCODE – используются только в пакетах ответов и отображают состояние ответа:
 - 0 – без ошибок;
 - 1 – ошибки в пакете запроса;
 - 2 – внутренние ошибки не дали возможности серверу обработать запрос;
 - 3 – имя, указанное в запросе, не существует;
 - 4 – данный тип запроса не поддерживается сервером;
 - 5 – сервер не может удовлетворить запрос клиента в силу административных ограничений безопасности.

Остальные четыре параметра заголовка представляют собой 16-битные числа и используются для учета количества исходных записей, возвращаемых в пакете:

- Биты 28-31: QDCOUNT – количество записей в поле запросов;
- Биты 28-31: ANCOUNT – количество записей в поле ответов;
- Биты 28-31: NSCOUNT – количество записей в поле об авторитетных серверах имен;
- Биты 28-31: ARCOUNT – количество записей в поле дополнительной информации.

1.5.2 Поле запроса

Поле запроса содержит запросы, ответы на которые клиент желает получить от DNS-сервера, и состоит из трех частей.

1. QNAME – список имен, для которых клиент желает получить IP-адреса. Перед каждым именем ставится однобайтное значение, которое определяет длину имени, а конец списка обозначается именем с нулевой длиной.
2. QTYPE – в каком виде клиент желает принимать информацию об имеющихся доменах (MX, A, TXT и т.д.).
3. QCLASS – класс запроса (для сети Internet – IN).

1.5.3 Поля ответа, полномочий и дополнительной информации

В поле ответа содержатся исходные записи базы DNS, доступные на сервере на момент запроса клиента. Если бит заголовка AA не установлен, то в поле полномочий будут указаны DNS-серверы, к которым клиент может обращаться за авторитетными ответами. В поле дополнительной информации передаются все исходные записи, которые воспринимаются DNS-сервером как имеющие отношение к данному запросу.

Поля ответа, полномочий и дополнительной информации имеют одинаковый формат:

- NAME – доменное имя, относящееся к исходной записи (тот же формат, что и QNAME в секции запроса);
- TYPE – тип исходной записи;
- CLASS – класс исходной записи (IN для Internet);
- TTL – допустимое время хранения данной ресурсной записи в кэше неавторитетного DNS-сервера;
- RDLLENGTH – длина поля данных в записи;
- RDATA – поле данных (формат зависит от типа записи).

1.6 Правила фильтрации

Для фильтрации сетевых пакетов принято учитывать следующие параметры:

- протокол передачи;
- ip-адрес источника;
- ip-адрес назначения;
- порт источника;
- порт назначения.

Особенности работы протоколов передачи пакетов в данной работе не рассматриваются.

Перечисленных выше параметров зачастую оказывается недостаточно. Например, когда на хост осуществляется спуфинг-атака (от англ. spoofing – подмена).

Спуфинг – это кибер-атака, в рамках которой мошенник выдает себя за некоторый надежный источник, чтобы получить доступ к важным данным или информации[7]. Различают несколько видов спуфинг-атак, в данной работе будут рассмотрены два из них.

1. IP-спуфинг. Данный вид спуфинга заключается в замене реальных ip-адресов на поддельные. Это дает возможность скрывать местоположение мошенника в интернете, перенаправлять данные пользователя к ненадежным устройствам и т.д.. Защитой от простейших попыток таких атак является использование двух правил [8].
 - (а) Отклонять любой входящий пакет с ip-адресом сети защищаемого хоста в качестве ip-адреса источника для любых номеров портов и протоколов. Логика этого правила в том, что на межсетевой экран не может прийти пакет в сеть защищаемого хоста, и при этом в нем указано, что послан он из этой же сети.
 - (b) Не выпускать пакет из внутренней сети, если в нем указан ip-адрес источника, отличный от внутренних адресов сети (логика данного правила аналогична).
2. DNS-спуфинг. В данном виде спуфинга DNS-кэш хоста или некоторого промежуточного DNS-сервера заполняется поддельными данными, в результате чего пользователь получает некорректный ip-адрес и перенаправляется на вредоносный сайт. Один из способов защиты от DNS-спуфинг – запрет на получение ответов от неавторитетных серверов, так как их кэш может быть «отравлен». Для этого необходимо проверять бит AA поля заголовка DNS-пакета.

1.7 Выводы

В результате проведенного анализа был определен способ перехвата входящих и исходящих пакетов – путем регистрации функций перехвата с использованием библиотеки Netfilter. В качестве точек перехвата предлагается использовать точку, которую проходят все входящие пакеты (NF_INET_PRE_ROUTING), и точку, которую проходят все исходящие пакеты (NF_INET_POST_ROUTING).

Были рассмотрены структуры и функции ядра, предоставляющие информацию о сетевых пакетах, а также структура DNS-пакетов. Определены параметры правил фильтрации и способы защиты от спуфинг-атак.

2 Конструкторский раздел

2.1 IDEF0

На рисунках 2.1-2.2 представлена IDEF0-диаграмма, описывающая работу межсетевого экрана.

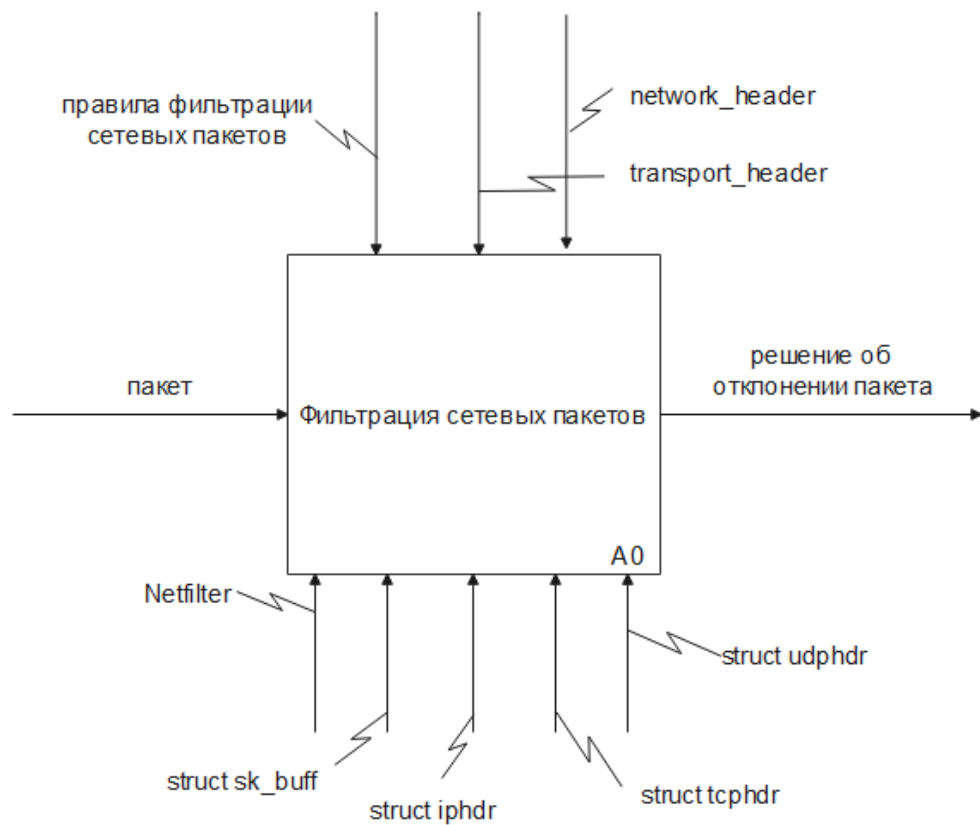


Рисунок 2.1 – Фильтрация сетевых пакетов, верхний уровень

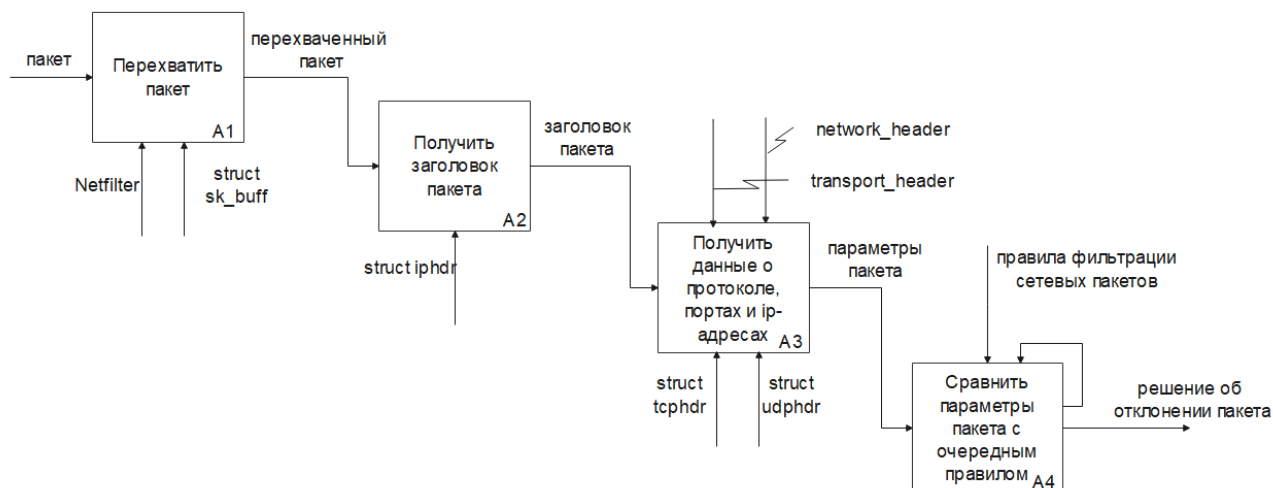


Рисунок 2.2 – Фильтрация сетевых пакетов, нижний уровень

2.2 Структура программного обеспечения

Межсетевой экран должен быть реализован в виде загружаемого модуля ядра и работать в режиме ядра. Однако пользователь должен иметь возможность управлять работой межсетевого экрана и редактировать правила фильтрации, для чего необходимо разработать отдельное приложение, которое будет выполняться в режиме пользователя.

Структура программного обеспечения приведена на рисунке 2.3.

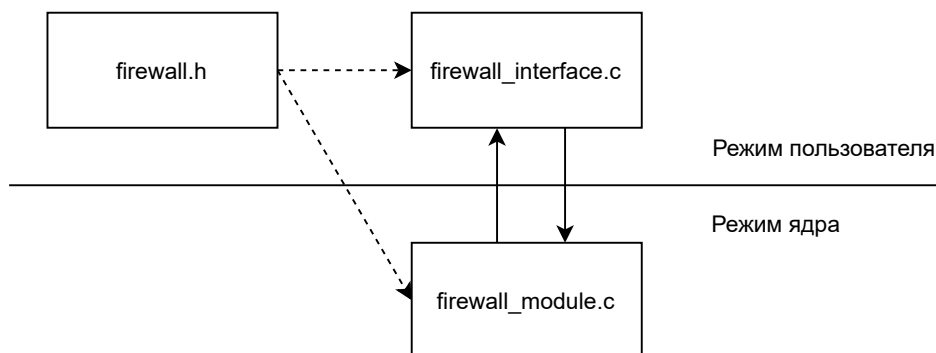


Рисунок 2.3 – Структура программного обеспечения

2.3 Алгоритм инициализации модуля

На рисунке 2.4 приведена схема алгоритма инициализации модуля.

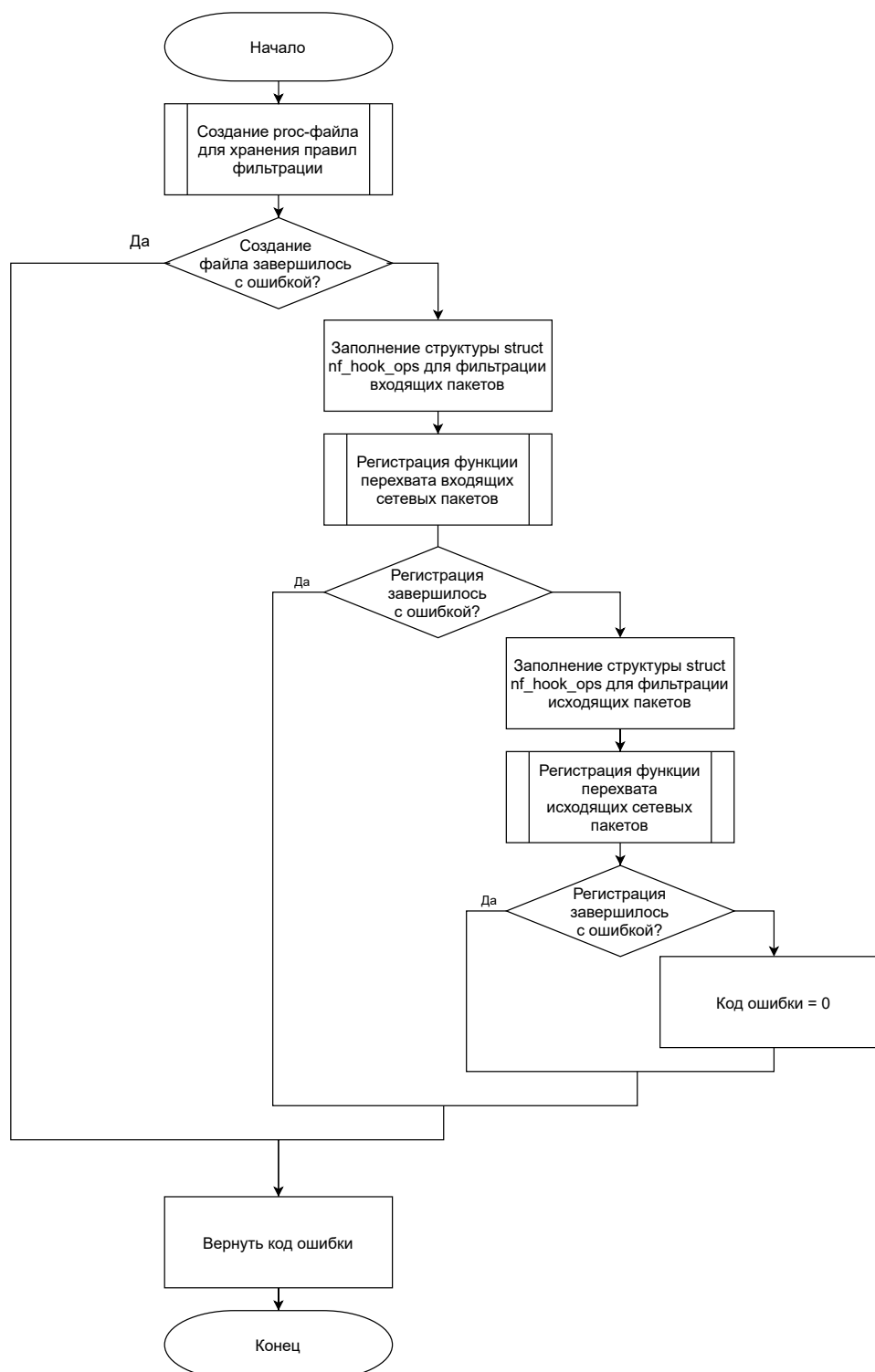


Рисунок 2.4 – Алгоритм инициализации модуля

2.4 Алгоритм фильтрации сетевых пакетов

На рисунках 2.5-2.6 приведена схема алгоритма фильтрации на примере входящих сетевых пакетов.

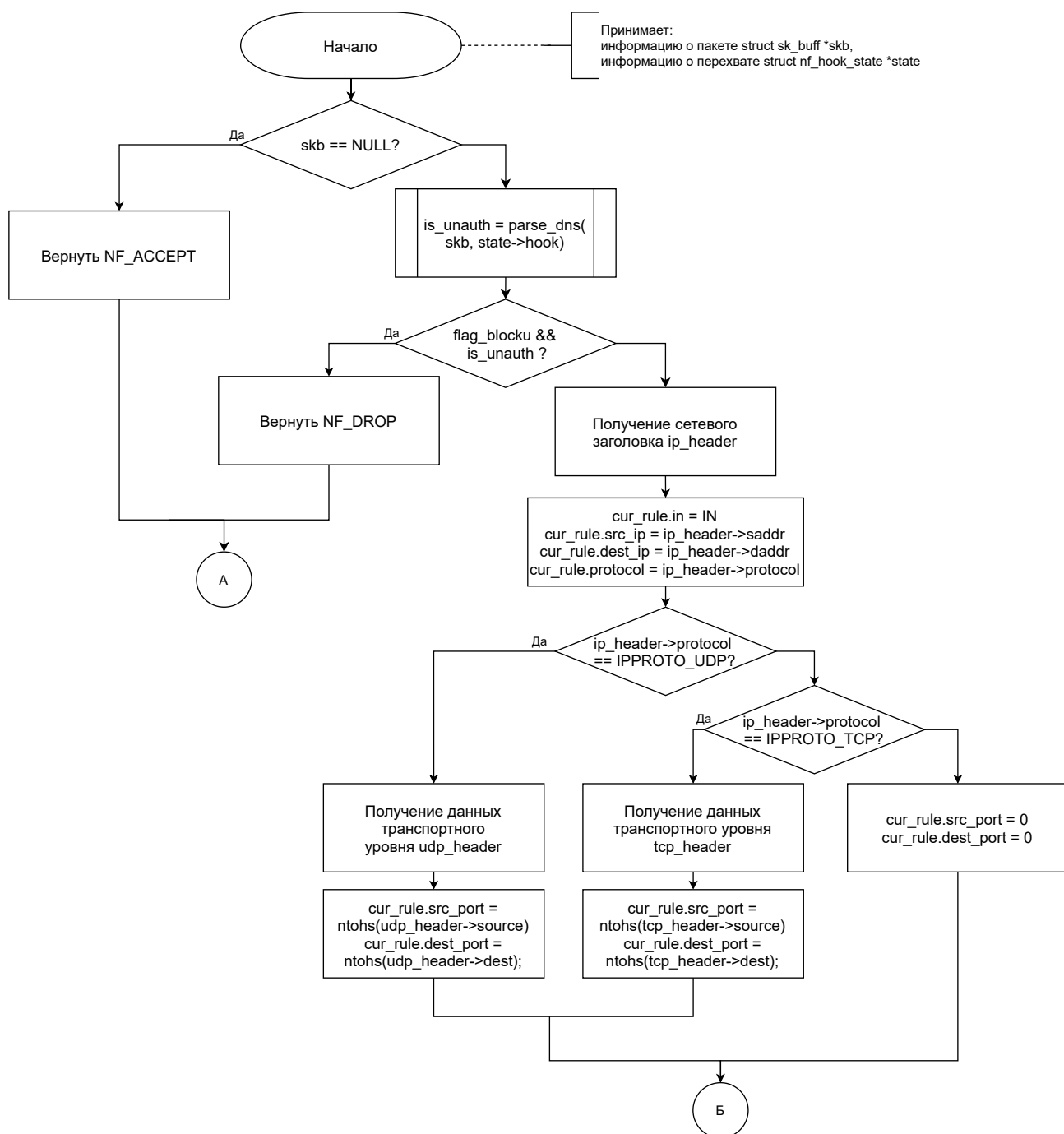


Рисунок 2.5 – Алгоритм фильтрации сетевых пакетов, начало.

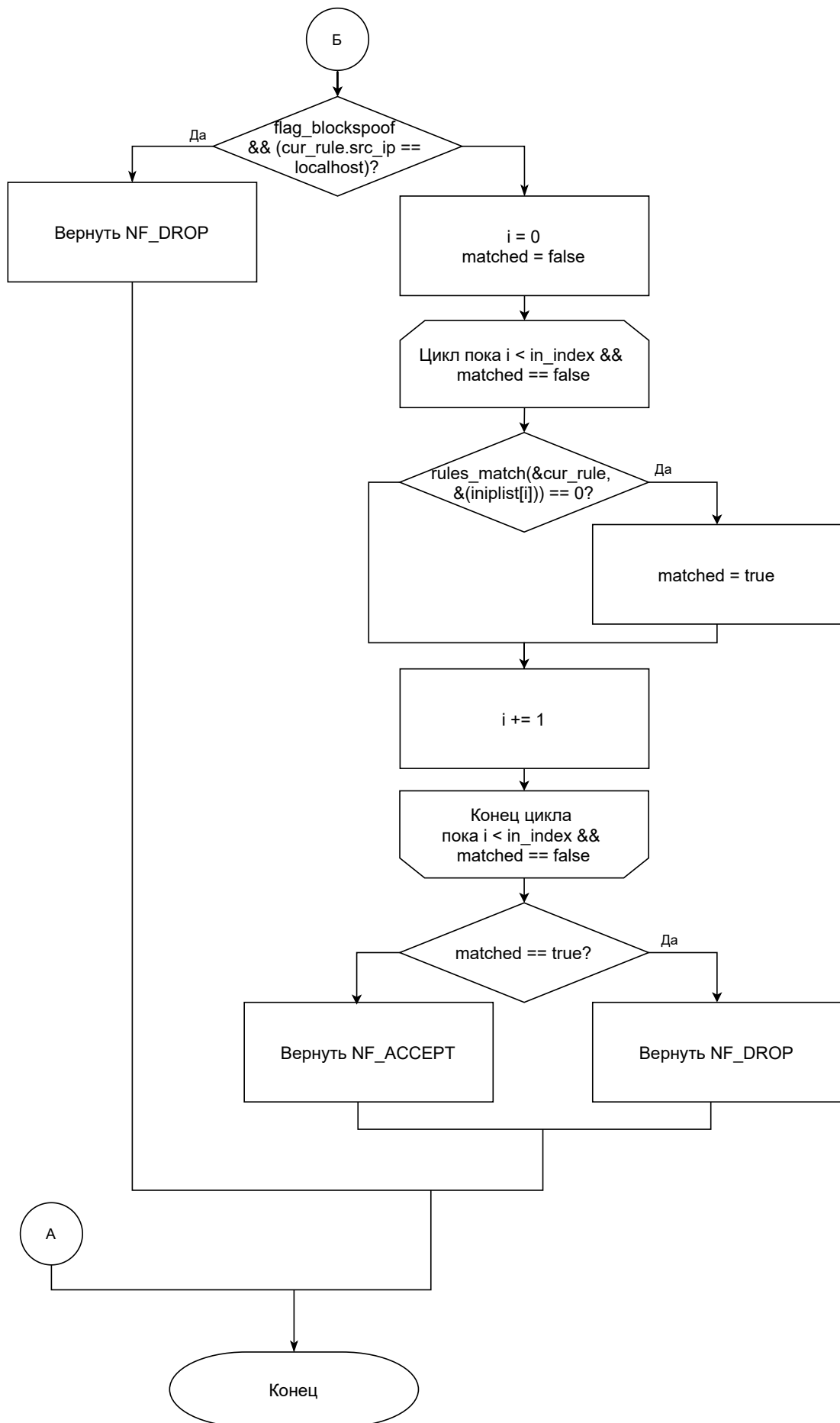


Рисунок 2.6 – Алгоритм фильтрации сетевых пакетов, конец.

2.5 Алгоритм проверки совпадения правил

На рисунке 2.7 приведена схема алгоритма проверки совпадения правил, который применяется для определения того, удовлетворяет ли рассматриваемый пакет очередному правилу.

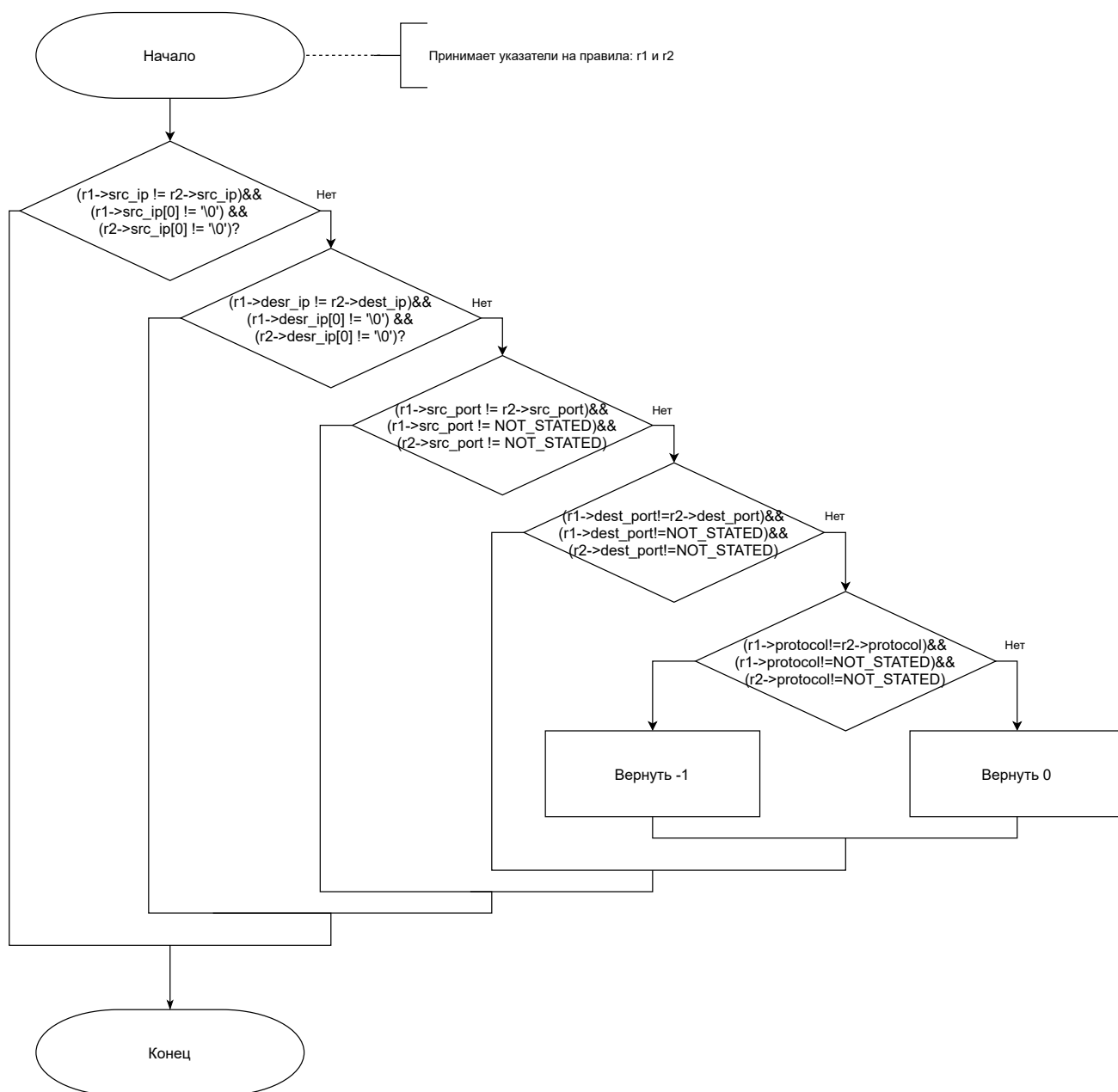


Рисунок 2.7 – Алгоритм проверки совпадения правил

2.6 Алгоритм вывода информации о DNS-пакете

На рисунке 2.8 приведена схема алгоритма вывода информации о DNS-пакете.



Рисунок 2.8 – Алгоритм вывода информации о DNS-пакете

3 Технологический раздел

3.1 Выбор средств разработки

В качестве языка программирования для реализации поставленной задачи был выбран язык Си. Для сборки модуля использовалась утилита make. В качестве среды разработки был выбран Qt Creator[11], так как он кроссплатформенный, бесплатный и использовался в курсе программирования ранее.

3.2 Инициализация модуля

В листинге 3.1 приведена реализация функции инициализации модуля.

Листинг 3.1 – Функция инициализации модуля

```
1 int netfilter_init(void)
2 {
3     DMSG("call init");
4
5     /* procфайл— для записи правил */
6     proc_file = proc_create(PROC_FILE_NAME, S_IRUGO | S_IWUGO,
7         NULL, &proc_fops);
8     if (!proc_file)
9     {
10         DMSG("call proc_create_data() fail");
11         return -ENOMEM;
12     }
13     DMSG("proc file created");
14
15     /* структура для фильтрации входящих пакетов */
16     nfho_in.hook = hook_func_in; // функция, которая будет
17     обрабатывать пакеты
18     nfho_in.hooknum = NF_INET_LOCAL_IN; // точка, в которой должна
19     срабатывать функция
20     nfho_in.pf = PF_INET; //семейство протоколов
21     nfho_in.priority = NF_IP_PRI_FIRST; // приоритет функции самый(
22     высокий)
```

```

20  /* регистрация структуры */
21  if (nf_register_net_hook(&init_net, &nfho_in) < 0)
22  {
23      DMSG("call nf_register_net_hook(NF_INET_LOCAL_IN) fail");
24      return -ENOMEM;
25  }
26
27  /* структура для фильтрации исходящих пакетов*/
28  nfho_out.hook = hook_func_out;
29  nfho_out.hooknum = NF_INET_LOCAL_OUT;
30  nfho_out.pf = PF_INET;
31  nfho_out.priority = NF_IP_PRI_FIRST;
32
33  /* регистрация структуры */
34  if (nf_register_net_hook(&init_net, &nfho_out) < 0)
35  {
36      DMSG("call nf_register_net_hook(NF_INET_LOCAL_OUT) fail");
37      return -ENOMEM;
38  }
39
40  DMSG("Firewall module loaded successfully");
41  return 0;
42 }

```

3.3 Фильтрация сетевых пакетов

В листинге 3.2 приведена реализация функции фильтрации входящих сетевых пакетов.

Листинг 3.2 – Функция фильтрации входящих сетевых пакетов

```

1  unsigned int hook_func_in(void *info, struct sk_buff *skb, const
    struct nf_hook_state *state)
2  {
3      int is_unauthorotuve = parse_dns(skb, state->hook);
4      if (bu && is_unauthorotuve)
5      {
6          DMSG("!!! Drop unauthorotive DNS answer");
7          return NF_DROP;

```

```

8     }
9     struct iphdr *ip_header;
10    sock_buff = skb;
11    if (!sock_buff)
12    {
13        return NF_ACCEPT;
14    }
15
16
17    ip_header = ip_hdr(sock_buff);
18
19    struct firewall_rule cur_rule;
20    cur_rule.in = IN;
21    snprintf(cur_rule.src_ip, BUFFER_SIZE, "%pI4", &ip_header->
        saddr);
22    snprintf(cur_rule.dest_ip, BUFFER_SIZE, "%pI4", &ip_header->
        daddr);
23    cur_rule.protocol = ip_header->protocol;
24
25    if (ip_header->protocol == IPPROTO_UDP)
26    {
27        struct udphdr *udp_header = (struct udphdr *)
            skb_transport_header(skb);
28        cur_rule.src_port = (unsigned int)ntohs(udp_header->source);
29        cur_rule.dest_port = (unsigned int)ntohs(udp_header->dest);
30    }
31    else if (ip_header->protocol == IPPROTO_TCP)
32    {
33        struct tcphdr *tcp_header = (struct tcphdr *)
            skb_transport_header(skb);
34        cur_rule.src_port = (unsigned int)ntohs(tcp_header->source);
35        cur_rule.dest_port = (unsigned int)ntohs(tcp_header->dest);
36    }
37    else
38    {
39        cur_rule.src_port = (unsigned int) 0;
40        cur_rule.dest_port = (unsigned int) 0;
41    }
42
43    if ((bs == 1) && (!strcmp(cur_rule.src_ip, localhost)))
44    {

```

```

45     DMSG("!!!!!! Drop incoming packet as suspicious for spoof,
src_ip: %s", cur_rule.src_ip);
46     return NF_DROP;
47 }
48
49 int i;
50 for (i = 0; i < in_index; i++)
51 {
52     if (rules_match(&cur_rule, &(iniplist[i])) == 0)
53     {
54         DMSG("Drop incoming packet: %s", str_rule(&cur_rule));
55         return NF_DROP;
56     }
57 }
58
59 return NF_ACCEPT;
60 }

```

3.4 Вывод информации о DNS-пакете

В листинге 3.3 приведена реализация функции вывода информации о DNS-пакете и проверки того, от какого сервера получен ответ – авторитетного или нет.

Листинг 3.3 – Функция вывода информации о DNS-пакете

```

1 static int show_info_about_dns(struct dnshdr *dns_hdr)
2 {
3     DMSG("DNS %s", dns_hdr->qr == 0 ? "query" : "response");
4     DMSG("AA bit: %s", dns_hdr->aa == 1 ? "set" : "unset");
5     int is_anauthorotive = dns_hdr->aa == 1 ? 0 : 1;
6
7     // количество записей в секциях запроса и ответа
8     size_t q_count = ntohs(dns_hdr->qdcount);
9     size_t a_count = ntohs(dns_hdr->ancount);
10    DMSG("question count = %zd, answer count = %zd", q_count,
        a_count);
11

```

```

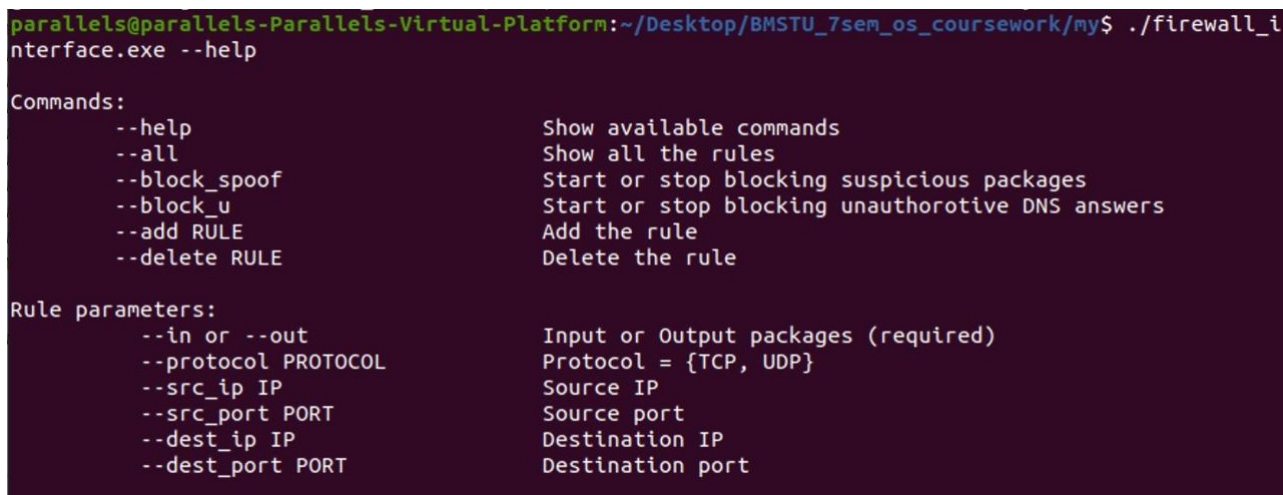
12  __u8 *section_start = ((__u8 *) dns_hdr) + sizeof(struct
    dnshdr);
13  size_t s_index, s_count = 0;
14
15  // секция запросов
16  for(s_index = 0; s_index < q_count; s_index++)
17  {
18      DMSG("question section %zd", s_index + 1);
19      s_count = parse_question_section(dns_hdr, section_start);
20      section_start += s_count;
21  }
22
23  // секция ответов
24  for(s_index = 0; s_index < a_count; s_index++)
25  {
26      DMSG("answer section %zd", s_index + 1);
27      s_count = parse_answer_section(dns_hdr, section_start);
28      section_start += s_count;
29  }
30
31  return is_anauthorotive;
32  }

```

4 Исследовательский раздел

4.1 Доступные команды и параметры

Для просмотра всех доступных команд и параметров следует вызвать команду **help**.



```
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/BMSTU_7sem_os_coursework/my$ ./firewall_interface.exe --help

Commands:
  --help          Show available commands
  --all           Show all the rules
  --block_spoof   Start or stop blocking suspicious packages
  --block_u       Start or stop blocking unauthorotive DNS answers
  --add RULE      Add the rule
  --delete RULE   Delete the rule

Rule parameters:
  --in or --out   Input or Output packages (required)
  --protocol PROTOCOL  Protocol = {TCP, UDP}
  --src_ip IP     Source IP
  --src_port PORT Source port
  --dest_ip IP    Destination IP
  --dest_port PORT Destination port
```

Рисунок 4.1 – Вывод команды help

Команды **help**, **all**, **block_spoof** и **block_u** вызываются без параметров, а для команд **add** и **delete** необходимо указать параметры добавляемого или удаляемого правила: обязательный параметр – направление (in или out), а также произвольное количество необязательных параметров:

- протокол (TCP или UDP);
- ip-адрес источника;
- порт источника;
- ip-адрес назначения;
- порт назначения.

4.2 Тестирование фильтрации по ip-адресу

После загрузки модуля список правил пуст, в чем можно убедиться вызовом команды **all** (рисунок 4.2).

```
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/BMSTU_7sem_os_coursework/my$ ./firewall_interface.exe
--all
direction      protocol      source IP      source port      destination IP      destination port
-----
```

Рисунок 4.2 – Вывод команды all

При пустом списке правил была вызвана утилита **ping** для проверки соединения с устройством с ip-адресом 127.0.0.8. Результат вызова приведен на рисунке 4.3: пакеты были успешно отправлены.

```
parallels@parallels-Parallels-Virtual-Platform:~$ ping 127.0.0.8
PING 127.0.0.8 (127.0.0.8) 56(84) bytes of data.
64 bytes from 127.0.0.8: icmp_seq=1 ttl=64 time=0.031 ms
64 bytes from 127.0.0.8: icmp_seq=2 ttl=64 time=0.037 ms
64 bytes from 127.0.0.8: icmp_seq=3 ttl=64 time=0.032 ms
^Z
[2]+  Stopped                  ping 127.0.0.8
parallels@parallels-Parallels-Virtual-Platform:~$ s
```

Рисунок 4.3 – Вывод утилиты ping до добавления правила

Было добавлено правило фильтрации входящих пакетов по ip-адресу назначения (рисунок 4.4).

```
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/BMSTU_7sem_os_coursework/my$ ./firewall_interface.exe
--all
direction      protocol      source IP      source port      destination IP      destination port
-----
IN              ANY           ANY            ANY              127.0.0.8           ANY
```

Рисунок 4.4 – Добавленное правило фильтрации

При вызове **ping** для проверки соединения с тем же устройством после добавления правила пакеты не были доставлены (рисунок 4.5).

```
parallels@parallels-Parallels-Virtual-Platform:~$ ping 127.0.0.8
PING 127.0.0.8 (127.0.0.8) 56(84) bytes of data.
^Z
[3]+  Stopped                  ping 127.0.0.8
```

Рисунок 4.5 – Вывод утилиты ping после добавления правила

Сообщения межсетевого экрана об отброшенных пакетах приведены на рисунке 4.6.

```
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/BMSTU_7sem_os_coursework/my$ sudo dmesg | grep FW | tail -n 3
[48549.757050] FW: firewall_module.c(0739): Drop incoming packet: IN      src_ip: 127.0.0.1      src_port: 0
dest_ip: 127.0.0.8      dest_port: 0
[48550.780291] FW: firewall_module.c(0739): Drop incoming packet: IN      src_ip: 127.0.0.1      src_port: 0
dest_ip: 127.0.0.8      dest_port: 0
[48551.804513] FW: firewall_module.c(0739): Drop incoming packet: IN      src_ip: 127.0.0.1      src_port: 0
dest_ip: 127.0.0.8      dest_port: 0
```

Рисунок 4.6 – Сообщения межсетевого экрана об отброшенных пакетах

После удаления правила (рисунок 4.7) пакеты вновь успешно доставляются по указанному адресу (рисунок 4.8).

```
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/BMSTU_7sem_os_coursework/my$ ./firewall_interface.exe --in --del --dest_ip=127.0.0.8
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/BMSTU_7sem_os_coursework/my$ ./firewall_interface.exe --all
direction      protocol      source IP      source port      destination IP      destination port
```

Рисунок 4.7 – Удаление правила

```
parallels@parallels-Parallels-Virtual-Platform:~$ ping 127.0.0.8
PING 127.0.0.8 (127.0.0.8) 56(84) bytes of data.
64 bytes from 127.0.0.8: icmp_seq=1 ttl=64 time=0.023 ms
64 bytes from 127.0.0.8: icmp_seq=2 ttl=64 time=0.063 ms
64 bytes from 127.0.0.8: icmp_seq=3 ttl=64 time=0.052 ms
64 bytes from 127.0.0.8: icmp_seq=4 ttl=64 time=0.033 ms
^Z
[4]+  Stopped                  ping 127.0.0.8
```

Рисунок 4.8 – Вывод утилиты ping после удаления правила

4.3 Тестирование фильтрации по протоколу

Для тестирования фильтрации по протоколу была использована программа **Wireshark**. До добавления правила фильтрации регулярно происходил обмен пакетами по протоколу TCP (рисунок 4.9).

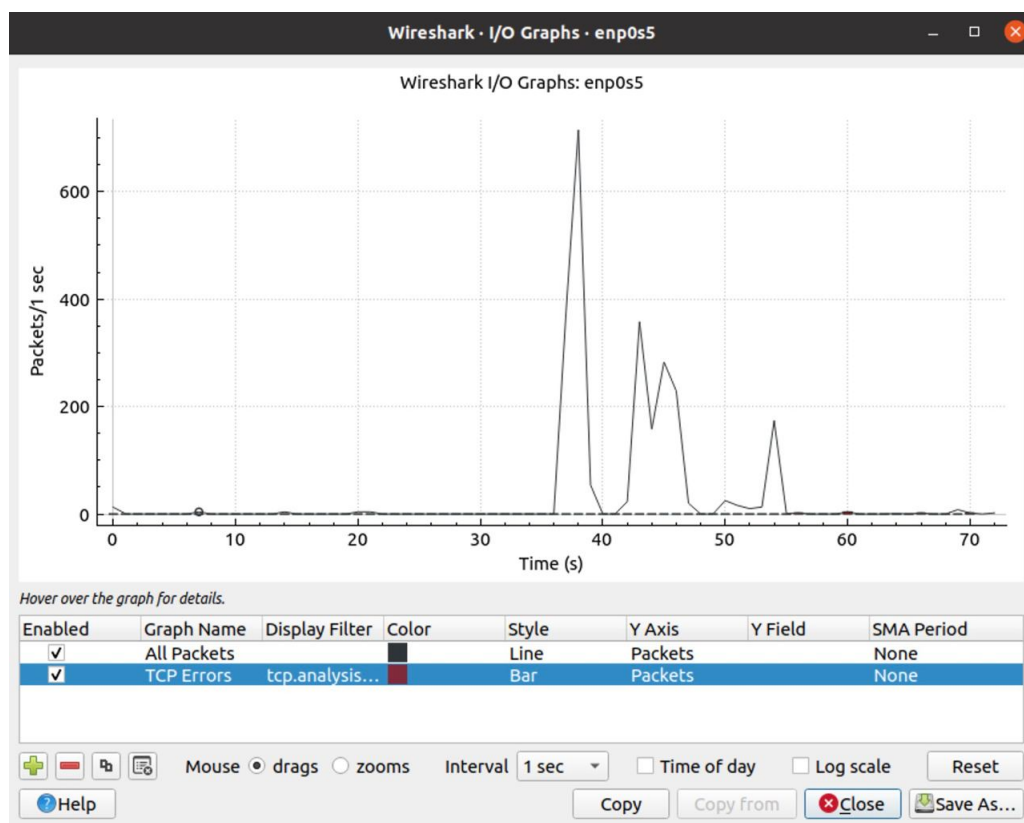


Рисунок 4.9 – Обмен пакетами по протоколу TCP до добавления правила

После добавления правил фильтрации, согласно которым необходимо отбрасывать все передаваемые по протоколу TCP пакеты (рисунок 4.10), обмен такими пакетами практически прекратился, что продемонстрировано на рисунке 4.11.

```
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/BMSTU_7sen_os_coursework/my$ ./firewall_interface.exe --out
--add --protocol=TCP
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/BMSTU_7sen_os_coursework/my$ ./firewall_interface.exe --in --
add --protocol=TCP
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/BMSTU_7sen_os_coursework/my$ ./firewall_interface.exe --all
direction      protocol      source IP      source port      destination IP      destination port
-----
IN              TCP           ANY            ANY              ANY                 ANY
OUT            TCP           ANY            ANY              ANY                 ANY
```

Рисунок 4.10 – Добавление правил фильтрации по протоколу TCP

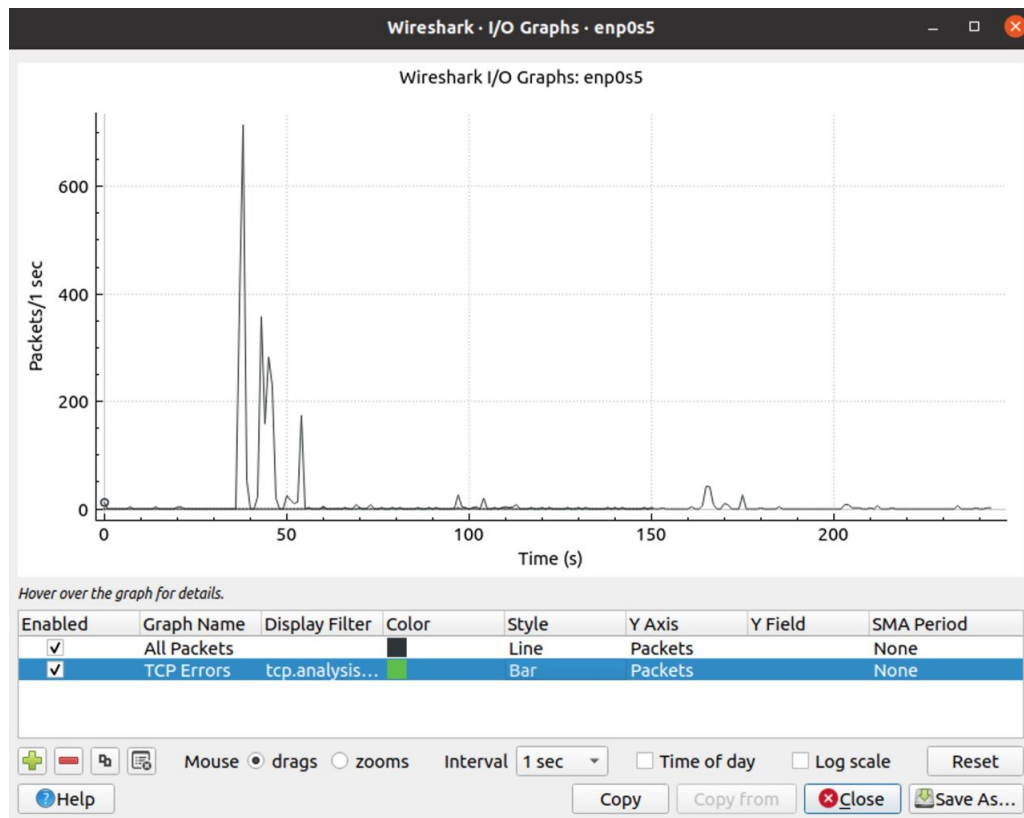


Рисунок 4.11 – Обмен пакетами по протоколу TCP после добавления правил

Соответствующие сообщения об отброшенных пакетах от межсетевого экрана приведены на рисунке 4.12.

```
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/BMSTU_7sem_os_coursework/my$ sudo dmesg | grep FW | tail -n 3
[sudo] password for parallels:
[ 1937.176381] FW: firewall_module.c(0806): Drop outgoing packet: OUT      src_ip: 10.211.55.3      src_port: 64649
dest_ip: 93.184.220.29      dest_port: 20480      protocol: TCP
[ 1938.200254] FW: firewall_module.c(0806): Drop outgoing packet: OUT      src_ip: 10.211.55.3      src_port: 64649
dest_ip: 93.184.220.29      dest_port: 20480      protocol: TCP
[ 1939.224452] FW: firewall_module.c(0806): Drop outgoing packet: OUT      src_ip: 10.211.55.3      src_port: 64649
dest_ip: 93.184.220.29      dest_port: 20480      protocol: TCP
```

Рисунок 4.12 – Сообщения об отброшенных пакетах от межсетевого экрана

После удаления правил (рисунок 4.13) обмен пакетами возобновился (рисунок 4.14).

```
parallels@parallels-Virtual-Platform:~/Desktop/BMSTU_7sem_os_coursework/my$ ./firewall_interface.exe --out -
-del --protocol=TCP
parallels@parallels-Virtual-Platform:~/Desktop/BMSTU_7sem_os_coursework/my$ ./firewall_interface.exe --in --
del --protocol=TCP
parallels@parallels-Virtual-Platform:~/Desktop/BMSTU_7sem_os_coursework/my$ ./firewall_interface.exe --all
direction      protocol      source IP      source port      destination IP      destination port
-----
```

Рисунок 4.13 – Удаление правил фильтрации по протоколу TCP

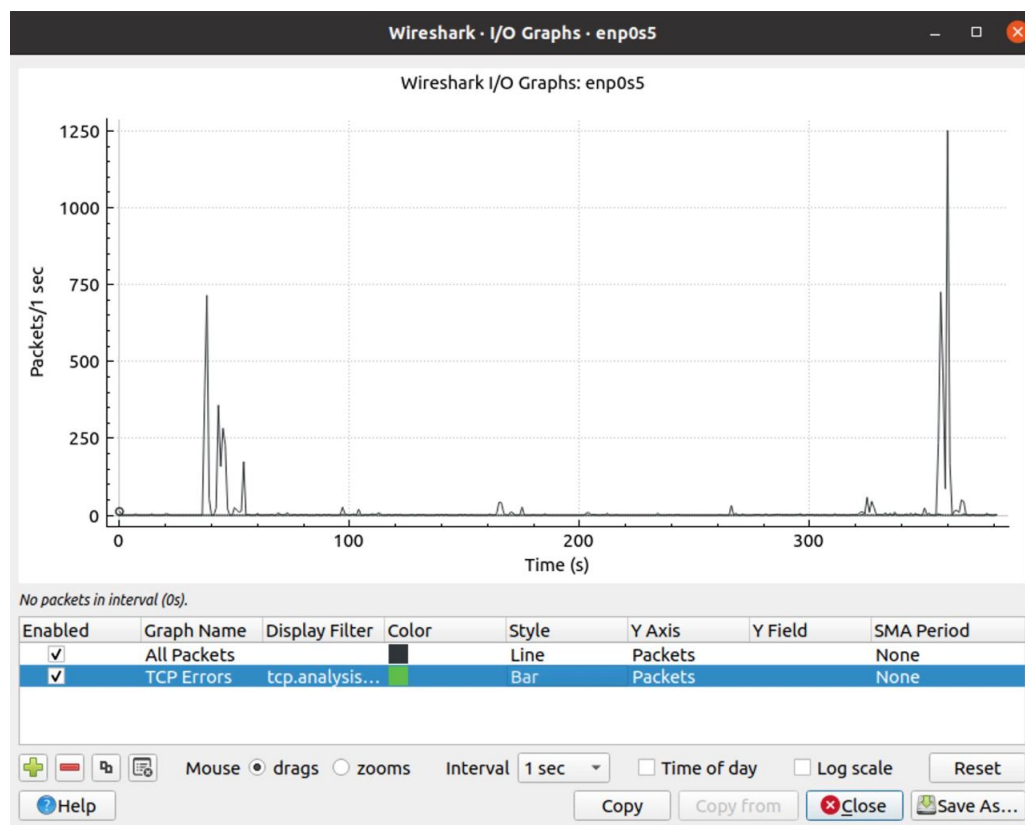


Рисунок 4.14 – Обмен пакетами по протоколу TCP после удаления правил

4.4 Вывод информации о DNS-пакетах

Для тестирования вывода информации о DNS-пакетах было произведено обращение по доменному имени bmstu.ru (рисунок 4.15).

```

parallels@parallels-Parallels-Virtual-Platform:~$ ping bmstu.ru
PING bmstu.ru (195.19.50.250) 56(84) bytes of data.
64 bytes from h250.net50.bmstu.ru (195.19.50.250): icmp_seq=1 ttl=128 time=6.80 ms
64 bytes from h250.net50.bmstu.ru (195.19.50.250): icmp_seq=2 ttl=128 time=4.98 ms
64 bytes from h250.net50.bmstu.ru (195.19.50.250): icmp_seq=3 ttl=128 time=12.4 ms
^Z
[4]+  Stopped                  ping bmstu.ru

```

Рисунок 4.15 – Обращение по доменному имени bmstu.ru

На рисунках 4.16 и 4.17 приведен частичный вывод информации о пакете-запросе и пакете-ответе, соответственно.

```

[ 2423.897920] FW: firewall_module.c(0270): DNS query
[ 2423.897922] FW: firewall_module.c(0271): AA bit: unset
[ 2423.897923] FW: firewall_module.c(0279): question count = 1, answer count = 0
[ 2423.897925] FW: firewall_module.c(0289): question section 1
[ 2423.897927] FW: firewall_module.c(0183): qname = bmstu.ru
[ 2423.897928] FW: firewall_module.c(0193): qtype = 0x0000, qclass = 0x0100
[ 2423.897930] FW: firewall_module.c(0194):
[ 2423.897965] FW: firewall_module.c(0269):
[ 2423.897966] FW: firewall_module.c(0270): DNS query
[ 2423.897968] FW: firewall_module.c(0271): AA bit: unset
[ 2423.897969] FW: firewall_module.c(0279): question count = 1, answer count = 0
[ 2423.897970] FW: firewall_module.c(0289): question section 1
[ 2423.897971] FW: firewall_module.c(0183): qname = bmstu.ru
[ 2423.897972] FW: firewall_module.c(0193): qtype = 0x0000, qclass = 0x1C00
[ 2423.897974] FW: firewall_module.c(0194):
[ 2423.898168] FW: firewall_module.c(0269):
[ 2423.898193] FW: firewall_module.c(0270): DNS query
[ 2423.898194] FW: firewall_module.c(0271): AA bit: unset
[ 2423.898196] FW: firewall_module.c(0279): question count = 1, answer count = 0
[ 2423.898198] FW: firewall_module.c(0289): question section 1
[ 2423.898199] FW: firewall_module.c(0183): qname = bmstu.ru
[ 2423.898200] FW: firewall_module.c(0193): qtype = 0x0000, qclass = 0x0100

```

Рисунок 4.16 – Информации о пакете-запросе

```

[ 2423.904124] FW: firewall_module.c(0270): DNS response
[ 2423.904126] FW: firewall_module.c(0271): AA bit: unset
[ 2423.904127] FW: firewall_module.c(0279): question count = 1, answer count = 1
[ 2423.904130] FW: firewall_module.c(0289): question section 1
[ 2423.904131] FW: firewall_module.c(0183): qname = bmstu.ru
[ 2423.904133] FW: firewall_module.c(0193): qtype = 0x0000, qclass = 0x0100
[ 2423.904134] FW: firewall_module.c(0194):
[ 2423.904136] FW: firewall_module.c(0297): answer section 1
[ 2423.904137] FW: firewall_module.c(0217): name = bmstu.ru
[ 2423.904138] FW: firewall_module.c(0226): type = 0x0001, class = 0x0001
[ 2423.904140] FW: firewall_module.c(0236): ttl = 325, rdlength = 4
[ 2423.904141] FW: firewall_module.c(0247): rdata (IPv4) = 195.019.050.250
[ 2423.904144] FW: firewall_module.c(0261):
[ 2423.904172] FW: firewall_module.c(0269):
[ 2423.904173] FW: firewall_module.c(0270): DNS response
[ 2423.904175] FW: firewall_module.c(0271): AA bit: unset
[ 2423.904176] FW: firewall_module.c(0279): question count = 1, answer count = 0
[ 2423.904177] FW: firewall_module.c(0289): question section 1
[ 2423.904178] FW: firewall_module.c(0183): qname = bmstu.ru
[ 2423.904179] FW: firewall_module.c(0193): qtype = 0x0000, qclass = 0x1C00

```

Рисунок 4.17 – Информации о пакете-ответе

4.5 Тестирование запрета приема ответов от неавторитетных DNS-серверов

До включения запрета приема ответов от неавторитетных DNS-серверов было произведено обращение по доменному имени google.com, и соединение было успешно установлено (рисунок 4.18).

```
parallels@parallels-Parallels-Virtual-Platform:~$ ping google.com
PING google.com (64.233.162.139) 56(84) bytes of data.
64 bytes from li-in-f139.1e100.net (64.233.162.139): icmp_seq=1 ttl=128 time=19.7 ms
64 bytes from li-in-f139.1e100.net (64.233.162.139): icmp_seq=2 ttl=128 time=19.8 ms
64 bytes from li-in-f139.1e100.net (64.233.162.139): icmp_seq=3 ttl=128 time=19.7 ms
^Z
[1]+  Stopped                  ping google.com
```

Рисунок 4.18 – Обращение по доменному имени google.com до включения запрета

Затем был включен запрет приема ответов от неавторитетных DNS-серверов (рисунок 4.19).

```
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/BMSTU_7sem_os_coursework/my$ ./firewall_interface.exe --block_u
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/BMSTU_7sem_os_coursework/my$ sudo dmesg | grep FW | tail -n 1
[ 2139.999379] FW: firewall_module.c(0501): Start blocking unauthoritive DNS answers
```

Рисунок 4.19 – Включение запрета приема ответов от неавторитетных DNS-серверов

После включения запрета при обращении по тому же доменному имени был получен ответ от неавторитетного DNS-сервера, в результате чего пакет был отброшен (рисунок 4.20), а соединение не было установлено (рисунок 4.21)

```
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/BMSTU_7sem_os_coursework/my$ sudo dmesg | grep FW | tail -n 30
[ 2251.971387] FW: firewall_module.c(0269):
[ 2251.971396] FW: firewall_module.c(0270): DNS response
[ 2251.971399] FW: firewall_module.c(0271): AA bit: unset
[ 2251.971402] FW: firewall_module.c(0279): question count = 1, answer count = 0
[ 2251.971405] FW: firewall_module.c(0289): question section 1
[ 2251.971408] FW: firewall_module.c(0183): qname = mozilla.cloudflare-dns.com
[ 2251.971410] FW: firewall_module.c(0193): qtype = 0x0000, qclass = 0x1C00
[ 2251.971413] FW: firewall_module.c(0194):
[ 2251.971415] FW: firewall_module.c(0680): !!! Drop unauthorotive DNS answer
```

Рисунок 4.20 – Сообщение от межсетевого экрана об отброшенном пакете от неавторитетного DNS-сервера

```
parallels@parallels-Parallels-Virtual-Platform:~$ ping google.com
ping: google.com: Temporary failure in name resolution
```

Рисунок 4.21 – Обращение по доменному имени google.com после включения запрета

При снятии запрета (рисунок 4.22) соединение вновь устанавливается успешно (рисунок 4.23).

```
[ 2307.505389] FW: firewall_module.c(0506): Stop blocking unauthoritive DNS answers
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/BMSTU_7sem_os_coursework/my$ ./firewall_interface.exe --block_u
parallels@parallels-Parallels-Virtual-Platform:~/Desktop/BMSTU_7sem_os_coursework/my$ sudo dmesg | grep FW | tail -n 1
```

Рисунок 4.22 – Снятие запрета на прием пакетов от неавторитетных DNS-серверов

```
parallels@parallels-Parallels-Virtual-Platform:~$ ping google.com
PING google.com (64.233.161.113) 56(84) bytes of data:
64 bytes from lh-in-f113.1e100.net (64.233.161.113): icmp_seq=1 ttl=128 time=18.6 ms
64 bytes from lh-in-f113.1e100.net (64.233.161.113): icmp_seq=2 ttl=128 time=26.0 ms
64 bytes from lh-in-f113.1e100.net (64.233.161.113): icmp_seq=3 ttl=128 time=20.0 ms
64 bytes from lh-in-f113.1e100.net (64.233.161.113): icmp_seq=4 ttl=128 time=18.6 ms
^Z
[2]+  Stopped                  ping google.com
```

Рисунок 4.23 – Обращение по доменному имени google.com после снятия запрета

Заключение

В ходе выполнения курсовой работы был определен способ перехвата входящих и исходящих пакетов – путем регистрации функций перехвата с использованием библиотеки Netfilter. В качестве точек перехвата было решено использовать точку, которую проходят все входящие пакеты (NF_INET_PRE_ROUTING), и точку, которую проходят все исходящие пакеты (NF_INET_POST_ROUTING).

В качестве параметров правил фильтрации пакетов были выбраны протокол передачи и ip-адреса и порты источника и назначения.

Были рассмотрены различные виды спуфинг-атак и разработаны меры защиты от них. Для защиты от IP-спуфинга было решено отбрасывать все входящие пакеты с ip-адресом сети защищаемого хоста в качестве ip-адреса источника и все исходящие пакеты с ip-адресом источника, отличным от внутренних адресов сети, независимо от протокола передачи и портов. Для защиты от DNS-спуфинга было решено проверять бит AA поля заголовка DNS-пакета и отбрасывать все ответы от неавторитетных DNS-серверов.

Литература

- [1] Будько М.Б. Будько М.Ю. Гирик А.В. Использование межсетевого экрана Netfilter для обеспечения сетевой безопасности в ОС Linux: Учебное пособие. СПб: Университет ИТМО, 2020. с. 56.
- [2] Уровни модели OSI. [Электронный ресурс]. Режим доступа: https://www.securitylab.ru/analytics/533599.php?clear_cache=Y (дата обращения: 12.10.2022).
- [3] Рязанова Н. Ю., Курс лекций по дисциплине «Операционные системы» [Текст].
- [4] Документация NAPI. [Электронный ресурс]. Режим доступа: <https://wiki.linuxfoundation.org/networking/napi> (дата обращения: 12.10.2022).
- [5] The netfilter.org project. [Электронный ресурс]. Режим доступа: <https://www.netfilter.org/> (дата обращения: 12.10.2022).
- [6] В. Мешков, Netfilter, журнал «Системный администратор». [Электронный ресурс]. Режим доступа: <http://samag.ru/archive/article/169> (дата обращения: 12.10.2022).
- [7] Что такое спуфинг и как предотвратить спуфинг-атаку. [Электронный ресурс]. Режим доступа: <https://www.cloudav.ru/mediacenter/tips/what-is-spoofing/> (дата обращения: 13.10.2022).
- [8] Создание и тестирование Firewall в Linux. [Электронный ресурс]. Режим доступа: <https://habr.com/ru/post/316756/> (дата обращения: 13.10.2022).
- [9] DNS и доменные имена. [Электронный ресурс]. Режим доступа: <https://intuit.ru/studies/courses/116/116/lecture/3361?page=5> (дата обращения: 13.10.2022).
- [10] Linux (ядро). [Электронный ресурс]. Режим доступа: [https://ru.bmstu.wiki/Linux_\(\T2A\cyrya\T2A\cyrd\T2A\cyrr\T2A\cyro\)](https://ru.bmstu.wiki/Linux_(\T2A\cyrya\T2A\cyrd\T2A\cyrr\T2A\cyro)) (дата обращения: 13.10.2022).

- [11] Qt Creator – A Cross-platform IDE for software development. [Электронный ресурс]. Режим доступа: <https://www.qt.io/product/development-tools> (дата обращения: 13.10.2022).