



Министерство образования и науки 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)

Методические материалы
для выполнения лабораторных работ
по дисциплине
«Сетевые технологии».

г. Москва

2023

Оглавление

Лабораторная работа 1. Изучение принципов работы утилит для исследования и мониторинга состояния сети.....	3
1.1. Цель работы	3
1.2. Задачи	3
1.3. Теоретический материал	3
1.3.1. Утилита <i>ifconfig</i>	3
1.3.2. Утилита <i>ping</i>	4
1.3.3. Утилита <i>traceroute</i>	7
1.3.4. Утилита <i>mtr</i>	9
1.3.5. Утилита <i>tracertap</i>	10
1.4. Порядок выполнения лабораторной работы	11
1.5. Содержание отчета.....	13
1.6. Контрольные вопросы	14
Лабораторная работа 2. Работа с программным анализатором протоколов <i>tcpdump</i>.	15
2.1. Цель работы	15
2.2. Задачи	15
2.3. Теоретический материал	15
2.4. Порядок выполнения лабораторной работы	20
2.5. Содержание отчета.....	22
2.6. Контрольные вопросы	22
Лабораторная работа 3. Ознакомление с системой и протоколом DNS	23
3.1. Цель работы	23
3.2. Задачи	23
3.3. Теоретический материал	23
3.4. Порядок выполнения лабораторной работы.....	28
3.4.1. Разрешение адресов в системе DNS с использованием различных утилит системы Linux	28
3.4.2. Получение ресурсных записей различных типов с использованием утилиты <i>nslookup</i>	28
3.4.3. Проведение обратного запроса DNS с использованием утилиты <i>host</i>	29
3.4.4. Получение всех ресурсных записей для определенного доменного имени с использованием утилиты <i>host</i>	30
3.5. Содержание отчета.....	30
3.6. Контрольные вопросы	30

Лабораторная работа 1.

Изучение принципов работы утилит для исследования и мониторинга состояния сети

1.1. Цель работы

Получение базовых навыков по использованию основных сетевых утилит, применяемых для исследования и мониторинга состояния сети. Изучение принципов их работы.

1.2. Задачи

Проверить доступность заданных узлов в сети Интернет и различными способами проверить маршрут прохождения пакетов до этих узлов.

1.3. Теоретический материал

Существует огромный спектр различных утилит, предназначенных для работы с сетью и сетевыми интерфейсами. Одни из них используются при настройке сетевых интерфейсов компьютера. Другие – для анализа текущего состояния локальной и глобальной компьютерных сетей. Третьи – для мониторинга состояния компьютерных сетей. Сложность этих утилит также различается. Используются как узкоспециализированные утилиты, так и программы-«комбайны», решающие сразу набор задач. Выбор используемого программного обеспечения зависит от стоящих задач. В данной работе предлагается рассмотреть работу ряда широко используемых узкоспециализированных сетевых утилит.

1.3.1. Утилита *ifconfig*

Утилита *ifconfig* предназначена для настройки сетевых интерфейсов ПК и для проверки их состояния. Утилита считается устаревшей, но тем не менее до сих пор широко используется. Ей на замену пришла утилита *ip*, входящая в набор программ *iproute2*.

Для настройки сетевого интерфейса утилитой *ifconfig* необходимо иметь права суперпользователя. Просмотр настроек и состояния сетевых интерфейсов возможен от лица обычного пользователя.

От лица обычного пользователя утилита *ifconfig* запускается следующим образом:

```
user@user-pc:~$ ifconfig
```

В результате на экран будут выведены параметры работающих в данный момент сетевых интерфейсов ПК. Для того, чтобы просмотреть настройки всех интерфейсов, необходимо запустить программу с флагом *-a*:

```
user@user-pc:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.0.108 netmask 255.255.255.0 broadcast 192.168.0.255
    inet6 fe80::a5e:108:b80d:d8b5 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:27:a8:30 txqueuelen 1000 (Ethernet)
    RX packets 248 bytes 289572 (289.5 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 206 bytes 20986 (20.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Более подробную информацию об использовании утилиты можно узнать из официальной *man*-страницы. Команда:

```
user@user-pc:~$ man ifconfig
```

1.3.2. Утилита *ping*

Утилита *ping* предназначена для тестирования сетей, управления сетями и измерения производительности. Из-за нагрузок, которые она создает в сети, не всегда разумно использовать *ping* в рабочее время или в автоматических сценариях.

Она отправляет запросы (ICMP Echo-Request) протокола ICMP указанному узлу сети и фиксирует поступающие ответы (ICMP Echo-Reply). Время между отправкой запроса и получением ответа (RTT - Round Trip Time) позволяет определять двусторонние задержки (RTT) по маршруту и частоту потери пакетов, т. е. косвенно определять загруженность на каналах передачи данных и промежуточных устройствах.

Полное отсутствие ICMP-ответов может также означать, что удаленный узел (или какой-либо из промежуточных маршрутизаторов) блокирует ICMP Echo-Reply или игнорирует ICMP Echo-Request.

Название происходит от английского названия звука импульса, издаваемого сонаром при отражении импульса от объекта.

Также есть несколько альтернативных толкований:

- PING - акроним «Packet InterNet Grouper (Grouper)»;
- Ping – часть названия игры пинг-понг. Это толкование подразумевает, что компьютеры обмениваются сигналами аналогично тому, как игроки в пинг-понг отбивают друг другу мяч.

Утилита ping запускается в терминале. Формат запуска команды:

```
user@user-pc:~$ ping address
```

Пример

```
user@user-pc:~$ ping 8.8.8.8 -c 4
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=58 time=24.8 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=58 time=20.5 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=58 time=21.6 ms
64 bytes from 8.8.8.8: icmp_seq=4 ttl=58 time=21.0 ms

--- 8.8.8.8 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3005ms
rtt min/avg/max/mdev = 20.513/21.990/24.828/1.682 ms
```

Опции/флаг	Описание
-b	Разрешить использование широковещательного адреса в качестве целевого
-c количество	Остановить работу после передачи заданного количества пакетов ECHOREQUEST. Если задано <i>ограничениенавремя работы (-w)</i> , программа будет ждать указанное количествоответных пакетов ECHOREPLY в указанный временной интервал
-i интервал	Интервал в секундах между отправкой пакетов. По умолчанию между отправкой пакетов делается пауза в 1

	с, либо, в случае лавинообразного режима, отправка производится без пауз. Задавать значения меньше 0,2 может только суперпользователь
-L	Подавлять циклические петли для широковещательных пакетов. Этот ключ применяется только если в качестве целевого адреса указан широковещательный
-n	Только цифровой вывод. Не расшифровывать имена (символьный вид) адресов
-q	Выводить только начальные и итоговые данные (не выводить информацию об отдельных запросах)
-s размер- пакета	Размер пакетов для пересылки. По умолчанию - 56, что соответствует размеру 64 байта после добавления 8 байтов заголовка ICMP
-t TTL	Время актуальности пакета IP (TTL - Time to Live)
- во ограничение на время работы	Время, по истечении которого <i>ping</i> завершит свою работу независимо от количества посланных и принятых пакетов. При указании этого параметра время ожидания для одного пакета игнорируется, и работа может быть завершена ранее указанного срока только в случае получения информации об ошибке (т. е. уведомления о том, что ответных пакетов точно не будет)
-W время ожидание ответа	Время ожидания (в секундах) ответного пакета. Принимается во внимание только если не было принято ни одного ответа. В противном случае программа ожидает получения Время актуальности пакета IP (TTL - Time to Live) двух ответов

1.3.3. Утилита *traceroute*

Программа *traceroute* предназначена для определения маршрутов следования данных в сетях TCP/IP. Она основана на использовании протоколов ICMP и UDP, а также механизма TTL (Time to Live).

Traceroute выполняет отправку данных указанному узлу сети, при этом отображая сведения о всех промежуточных маршрутизаторах, через которые прошли данные на пути к целевому узлу. В случае проблем при доставке данных до какого-либо узла программа позволяет определить, на каком именно участке сети возникли неполадки. Программа работает только в направлении от источника пакетов и является весьма грубым инструментом для выявления неполадок в сети. В силу особенностей работы протоколов маршрутизации в сети Интернет, обратные маршруты часто не совпадают с прямыми, причем это справедливо для всех промежуточных узлов. Поэтому, ICMP ответ от каждого промежуточного узла может идти своим собственным маршрутом, затеряться или прийти с большой задержкой, хотя в реальности с пакетами, которые адресованы конечному узлу, этого не происходит. Кроме того, на промежуточных маршрутизаторах часто стоит ограничение числа ответов ICMP в единицу времени, что приводит к появлению ложных потерь.

Утилита *traceroute* входит в поставку большинства современных сетевых операционных систем. В системах Microsoft Windows эта программа носит название *tracert*, а в системах GNU/Linux, Cisco IOS и Mac OS – *traceroute*.

Для определения промежуточных маршрутизаторов *traceroute* отправляет серию (обычно три) UDP датаграмм целевому узлу, при этом каждый раз увеличивая на 1 значение поля TTL («время жизни») в заголовке IP-пакета. Это поле указывает максимальное количество маршрутизаторов, которое может быть пройдено пакетом. Первая серия пакетов отправляется с TTL, равным 1, и поэтому первый же маршрутизатор возвращает обратно сообщение ICMP «время жизни истекло» (тип 11), указывающее на невозможность доставки данных. *Traceroute* фиксирует адрес

маршрутизатора, а также время между отправкой пакета и получением ответа (эти сведения выводятся на монитор компьютера). Затем *traceroute* повторяет отправку серии пакетов, но уже с TTL, равным 2, что позволяет первому маршрутизатору пропустить их дальше.

Процесс повторяется до тех пор, пока при определенном значении TTL пакет не достигнет целевого узла. При получении ответа от этого узла процесс трассировки считается завершенным.

На конечном хосте IP-пакет с TTL = 1 не отбрасывается и должен быть передан приложению. Достижение пункта назначения определяется следующим образом: отсылаемые *traceroute* пакеты содержат датаграмму UDP с таким номером порта адресата (превышающим 30 000), что он заведомо не используется на адресуемом хосте. В пункте назначения модуль UDP, получая подобные дейтаграммы, возвращает сообщения ICMP «порт недоступен» (тип 3, код 3). Таким образом, чтобы узнать о завершении работы, программе *traceroute* достаточно обнаружить, что поступило сообщение ICMP об ошибке этого типа.

Формат запуска команды:

```
user@user-pc:~$ traceroute 8.8.8.8
traceroute to 8.8.8.8 (8.8.8.8), 30 hops max, 60 byte packets
 1  _gateway (192.168.0.1)  1.435 ms  1.391 ms  1.370 ms
 2  10.32.0.1 (10.32.0.1)  4.393 ms  4.374 ms  4.354 ms
 3  h153.net32.bmstu.ru (195.19.32.153)  4.334 ms  4.312 ms  4.288 ms
 4  194.190.254.105 (194.190.254.105)  11.970 ms  11.945 ms  16.286 ms
 5  403.et011.m9-5-gw.msk.niks.su (194.85.42.66)  5.886 ms  5.870 ms  5.856 ms
 6  72.14.210.102 (72.14.210.102)  5.841 ms  5.454 ms  5.400 ms
 7  * * *
 8  108.170.250.129 (108.170.250.129)  5.036 ms  108.170.227.88 (108.170.227.88)  4.656 ms  72.14.233.90 (72.14.233.90)  5.004 ms
 9  108.170.250.51 (108.170.250.51)  4.989 ms  108.170.250.99 (108.170.250.99)  5.082 ms  4.838 ms
10  142.251.238.82 (142.251.238.82)  22.548 ms  22.445 ms  142.250.239.64 (142.250.239.64)  18.086 ms
11  142.251.237.146 (142.251.237.146)  22.413 ms  27.380 ms  142.250.235.62 (142.250.235.62)  29.551 ms
12  216.239.49.107 (216.239.49.107)  27.316 ms  142.250.56.221 (142.250.56.221)  31.505 ms  216.239.58.67 (216.239.58.67)  23.565 ms
13  * * *
14  * * *
15  * * *
16  * * *
17  * * *
18  * * *
19  * * *
20  * * *
21  * * *
22  dns.google (8.8.8.8)  22.407 ms  22.348 ms  22.326 ms
```

Можно видеть время достижения каждого из промежуточных узлов для каждого из трех отправляемых пакетов.

Более подробную информацию об использовании утилиты можно получить из официальной man-страницы.

1.3.4. Утилита *mtr*

Mtr (MyTraceroute) -- это служебная компьютерная программа, предназначенная для определения маршрутов следования данных в сетях TCP/IP. Она постоянно показывает сведения о маршруте, потерях, минимальной и максимальной задержке. При помощи *mtr* можно узнавать, где в сети происходят потери, задержки и обрывается связь.

При запуске *mtr* начинается исследование сетевого соединения между локальной машиной и хостом, заданным пользователем. После того, как она определит адрес каждого транзитного узла на пути следования пакетов, она отправляет каждому из этих узлов последовательности запросов ICMP ECHO, пытаясь определить качество канала связи с каждым из них. По окончании исследования выдается статистика по каждому исследованному узлу.

Формат запуска команды:

```
user@user-pc:~$ mtr 8.8.8.8
```

```
Host                                     Packets                               Plings
Loss%  Snt  Last  Avg  Best  Wrst  StDev
1. _gateway                               0.0%   4    1.3   2.7   1.3   5.1   1.8
2. 10.32.0.1                              0.0%   4    3.8   4.6   3.0   8.2   2.4
3. h153.net32.bmstu.ru                    66.7%   4    7.2   7.2   7.2   7.2   0.0
4. 194.190.254.105                        33.3%   4    2.7   6.0   2.7   9.3   4.7
5. 403.et011.m9-5-gw.msk.niks.su         33.3%   4    3.8  29.1   3.8  54.5  35.9
6. 72.14.210.102                          33.3%   4    4.2   8.3   4.2  12.3   5.7
7. 108.170.250.129                        33.3%   4    9.9   7.4   4.9   9.9   3.6
8. 108.170.250.130                        33.3%   4    3.7   4.4   3.7   5.2   1.0
9. 142.250.238.214                        33.3%   4   21.0  22.8  21.0  24.6   2.5
10. 142.250.235.62                        33.3%   4   23.3  26.5  23.3  29.6   4.4
11. 72.14.236.73                          33.3%   4   20.9  21.9  20.9  22.9   1.4
12. (waiting for reply)
13. (waiting for reply)
14. (waiting for reply)
15. (waiting for reply)
16. (waiting for reply)
17. (waiting for reply)
18. (waiting for reply)
19. (waiting for reply)
20. (waiting for reply)
21. dns.google                             0.0%   3   18.2  20.2  18.2  21.6   1.8
```

Опция/флаг	Описание
-r	Переводит <i>mtr</i> в режим <i>report</i> . В этом режиме <i>mtr</i> отработывает количество циклов, заданное флагом -c, затем выводит статистику и завершает работу
-c число	Задаёт количество циклов опроса маршрута. Каждый цикл длится 1 с
-i число	Задаёт время в секундах на один цикл опроса. По умолчанию - 1

1.3.5. Утилита *tracemap*

Tracemap – программа, позволяющая выполнить трассировку пути на несколько хостов сразу и представить полученные данные в виде графической карты. Она написана Игорем Шубиным на языке Perl. В лабораторной работе используется измененная версия программы, не требующая прав суперпользователя.

Программу *tracemap* необходимо скачать. Для этого можно воспользоваться командой:

```
+++++
```

Также необходимо установить *grafviz*, *libnet-ip-perl*

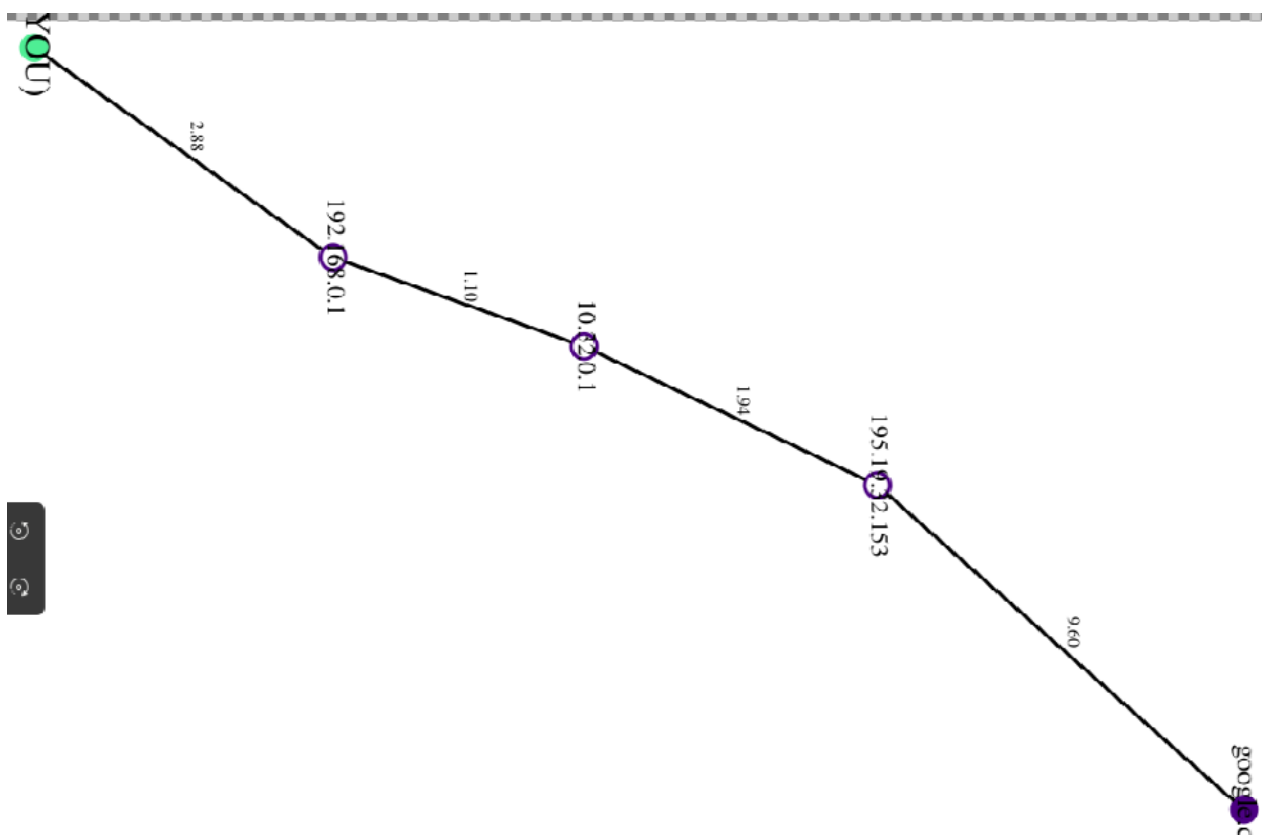
Программа будет сохранена в текущем каталоге.

Формат запуска команды:

```
user@user-pc:~$ echo google.com | perl tracemap.pl  
readline() on closed filehandle PREFIXES at tracemap.pl line 44.  
Tracing path to google.com....Done [last 30.023, total 75.919]
```

В результате выполнения программы в текущем каталоге появятся файлы *tracemap.svg* и *tracemap.png* с графической картой путей.

Пример: запуск утилиты для построения карты маршрутов до хостов



1.4. Порядок выполнения лабораторной работы

1. Просмотреть параметры сетевого интерфейса лабораторного ПК, воспользовавшись утилитой *ifconfig*. Выведенную на экран информацию сохранить для отчёта.

Для того чтобы сразу записать вывод команды в файл, можно воспользоваться утилитой *tee*

```
+++++
```

2.С помощью утилиты *ping* проверить состояние связи с узлами, заданными в табл. 1.3. Результаты выполнения сохранить для отчета.

Количество отправляемых до каждого узла «эхо-запросов» следует ограничить 4-5 пакетами.

По результатам проверки состояния связи заполнить таблицу, приведенную ниже.

Доменное имя	IP-адрес	Страна	Число потерянных запросов	Среднее время прохождения запроса	TTL
...

3. При помощи утилиты *traceroute* произвести трассировку узлов, заданных в табл. 1.3. Результаты протоколировать в файл.

По результатам трассировки составить графики времени прохождения маршрутизаторов для каждого узла (для 3 пакетов), указать наиболее узкие места в сети.

№ варианта	Исследуемые узлы	№ варианта	Исследуемые узлы
0	www.yandex.ru www.gismeteo.ru www.uefa.com	5	bmstu.ru www.iana.org www.jp-australia.com
1	vk.com www.pl.md adobe.com	6	www.rambler.ru www.cisco.com opera.com
2	ubuntu.ru www.vlc.ru www.japantoday.com	7	www.kontest.ru www.ewm1.pl www.nic.ad.jp
3	www.industry.su www.tdpoland.pl www.oracle.com	8	linux.org.ru www.skype.com www.runtu.org
4	mob.ru www.unfoundation.org www.sut.ru	9	www.gaw.ru www.drweb.com www.02.pl

4. Получить маршрут прохождения пакетов до одного из заданных в варианте узлов при помощи утилиты *ping*. Результаты протоколировать в файл.

Для выполнения этого задания необходимо последовательно посылать «эхо-запросы» на искомый узел, последовательно увеличивая параметр TTL на 1. Начиная с TTL = 1 и заканчивая TTL, на котором будет достигнут искомый узел. Количество отправляемых на каждом шаге пакетов следует ограничить 2-3.

5. Определить маршрут прохождения пакетов до узла, выбранного в предыдущем пункте при помощи утилиты *mtr*. Результаты протоколировать в файл.

Провести сравнение результатов определения маршрута, полученных спомощью утилиты *traceroute*, *ping* и *mtr*.

Важно: Утилита *mtr* должна быть запущена в режиме *report* с ограничением количества циклов опроса маршрута.

6. Построить графическую карту трассировки одновременно ко всем заданным в табл. 1.3 узлам при помощи программы *tracemap*.

1.5. Содержание отчета

- а) листинг параметров сетевого интерфейса лабораторного ПК;
- б) листинг результатов, полученных при работе с утилитой *ping*, плюс таблица с результатами исследований при использовании утилиты *ping*;
- в) листинг результатов, полученных при работе с утилитой *traceroute*, плюс графики времени прохождения шлюзов (для 3 пакетов) с указанием наиболее узких мест в сети;
- г) листинг результатов, полученных при определении маршрута прохождения пакетов утилитой *ping*;
- д) листинг результатов, полученных при работе с утилитой *mtr*, и привести сравнение результатов определения маршрута, полученных с помощью, утилита *traceroute*, *ping* и *mtr*.

Важно: перед каждым листингом результатов работы должна быть написана соответствующая команда.

е) Карта трассировки, полученная при помощи утилиты *tracemap*.

1.6. Контрольные вопросы

1. Утилита *ping*. Назначение и принцип работы.
2. Утилита *traceroute*. Назначение и принцип работы
3. Утилита *mtr*. Назначение и принцип работы.
4. Механизм TTL. Назначение и принцип работы.
5. Протокол ICMP.
6. Формат пакета ICMP.
7. Виды пакетов ICMP.

Лабораторная работа 2.

Работа с программным анализатором протоколов *tcpdump.*

2.1. Цель работы

Получение базовых навыков по работе с анализатором протоколов *tcpdump*. Изучение принципов фильтрации пакетов.

2.2. Задачи

Захватить при помощи анализатора протоколов *tcpdump* заданные сетевые пакеты.

2.3. Теоретический материал

Tcpdump (от TCP и англ. dump - свалка, сбрасывать) -- утилита UNIX, позволяющая перехватывать и анализировать сетевой трафик, проходящий через компьютер, на котором запущена данная программа.

Программа *tcpdump* была написана Van Jacobson, Craig Leres и Steven McCanne, во время их работы в лаборатории Lawrence Berkeley, Калифорнийского университета, Беркли.

Программа *tcpdump* разработана таким образом, чтобы переводить сетевую плату в смешанный режим (*promiscuous mode*), при этом, каждый пакет, проходящий по кабелю, фиксируется. Обычно сетевые платы для средпередачи, таких как Ethernet, захватывают только фреймы канального уровня, адресованные конкретному интерфейсу или отправленные на ширококвещательный адрес.

Операционная система должна позволить поместить интерфейс в смешанный режим и позволить пользовательскому процессу захватывать фреймы. Поэтому для выполнения программы требуется наличие прав суперпользователя и прямой доступ к устройству.

Основные назначения *tcpdump*:

- отладка сетевых приложений;

- отладка сети и сетевой конфигурации в целом.

Утилита `tcpdump` запускается в терминале от имени суперпользователя. Формат запуска команды:

```
user@user-pc:~$ sudo tcpdump [options] [filter]
```

Фильтр необходим для того, чтобы отображать на экране (или сохранять в файл) только определенные пакеты (т. е. те, которые соответствуют фильтру). Фильтр представляет из себя один или несколько примитивов фильтрации, объединенных логическими операторами. Предпочтительно (но не обязательно) записывать строку фильтрации в апострофах.

Таблица 2.1 Основные опции (флаги) утилиты `tcpdump`.

Опция/флаг	Описание
-i	Указывает на то, какой сетевой интерфейс будет использоваться для захвата пакетов. По умолчанию - eth0
-w имя файла	Сохраняет данные <code>tcpdump</code> в двоичном формате. Преимуществами использования данного способа по сравнению с обычным перенаправлением в файл являются высокая скорость записи и возможность чтения подобных данных другими программами, например, <code>snort</code> или <code>Wireshark</code> , но этот файл нельзя прочитать человеку
-r имя файла	Этот параметр позволяет <code>tcpdump</code> прочесть трафик из файла, если он был предварительно сохранен параметром <code>-w</code>
-x	Делает распечатку пакета в шестнадцатеричной системе, полезно для более детального анализа пакета. Количество отображаемых данных зависит от опции <code>-s</code>
-xx	То же, что и предыдущий параметр, но включает в себя заголовок начального уровня
-X	Выводит пакет в ASCII- и hex-формате. Полезно в случае анализа инцидента, связанного со взломом, так как позволяет просмотреть, какая текстовая информация передавалась во время соединения
-XX	То же, что и предыдущий параметр, но включает заголовок канального уровня
-s число	Количество байтов пакета, которые будут обрабатывать <code>tcpdump</code> . При установке большого числа отображаемых байтов информация может не уместиться на экране, и ее

	будет трудно изучать. В зависимости от того, какие цели вы преследуете, и следует выбирать значение этого параметра. По умолчанию tcpdump сохраняет первые 68 байт, однако если вы хотите увидеть содержимое всего пакета, используйте значение в 1500 байт (максимально допустимый размер пакета в сети Ethernet)
-c число	tcpdump завершит работу после получения указанного числа пакетов
-v	Вывод подробной информации (TTL; ID; общая длина заголовка, а также его параметры; производит проверку контрольных сумм IP- и ICMP-заголовков)
-vv	Вывод еще более полной информации, в основном касается NFS и SMB
-vvv	Вывод максимально подробной информации
-l	Использование стандартного потокового вывода tcpdump (stdout), например, для записи в файл: tcpdump -l tee out.log отобразит работу tcpdump и сохранит результат в файле out.log

Таблица 2.2 Примитивы фильтрации пакетов утилиты tcpdump.

Примитивы	Описание
dst host хост	Будет отбирать пакеты, в которых поле адреса получателя IPv4/v6 содержит адрес хоста, заданного в примитиве
src host хост	Будет выбирать все пакеты, в которых поле отправителя содержит адрес указанного хоста
host хост	Будет отбирать все пакеты, для которых адрес хоста указан в поле получателя или отправителя
ether dst MAC	Будет выбирать все кадры, в которых поле MAC-адреса получателя содержит значение MAC
ether src MAC	Будет выбирать все кадры, в которых поле MAC-адреса отправителя содержит значение MAC
ether host MAC	Будет отбирать все пакеты с адресом, указанным

	значением MAC в поле отправителя или получателя
gateway хост	Будет отбирать все пакеты, использующие указанный именем хост в качестве шлюза
dst net сеть	Отбирает все пакеты IPv4/v6, направленные в указанную сеть
src net сеть	Выбирает все пакеты IPv4/v6, отправленные из указанной сети
net сеть	Выбирает все пакеты IPv4/v6, содержащие адреса из указанной сети в поле отправителя или получателя
dst port порт	Отбирает все пакеты IP/TCP, IP/UDP, IPv6/TCP и IPv6/UDP, направленные в указанный порт
src port порт	Отбирает все пакеты, отправленные из указанного порта
port порт	Отбирает все пакеты, содержащие указанный номер порта в поле отправителя или получателя. Любое из трех перечисленных правил для портов может включать в качестве префикса идентификатор протокола TCP или UDP (например, tcp src port порт, будет отбирать пакеты TCP, отправленные из указанного порта)
less размер	Будет собирать пакеты, размер которых не превышает указанного значения
greater размер	Будет собирать пакеты, размер которых не меньше указанного значения
ip proto протокол	Отбирает все пакеты IP, содержащие заданный идентификатор типа в поле типа протокола (icmp, icmpv6, igmp, igmp, pim,ah, esp, vrrp, udp, tcp). Поскольку tcp, udp и icmp используются также в качестве ключевых слов, перед этими

	идентификаторами следует помещать символ «\»). Этот примитив не проверяет цепочки протокольных заголовков
ether proto протокол	Отбирает кадры Ethernet с заданным типом протокола (ip, ipb,arp, rarp, atalk, aarp, decnet, sca, lat, mopr, moprc, iso, stp, ipx, netbeui)

Примитивы фильтрации могут группироваться с помощью логических операторов:

- скобки;
- отрицание (! или not);
- логическое пересечение (&& или and);
- логическое объединение (|| или or).

Оператор отрицания имеет высший уровень приоритета, операции объединения и пересечения имеют одинаковый приоритет и выполняются слева направо в порядке следования. Для операции логического пересечения недостаточно просто указать операнды рядом, а требуется явно задать операцию (&& или and).

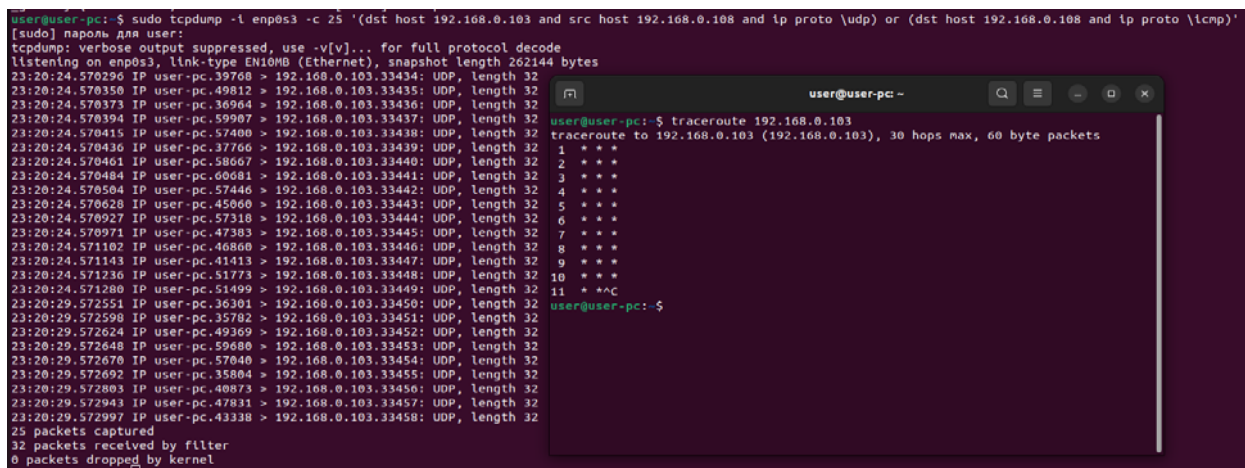
В качестве примера рассмотрим использование *tcpdump* для перехвата трафика, создаваемого утилитой *traceroute*.

Traceroute использует пакеты протокола UDP в качестве «запросов» и получает пакеты протокола ICMP в качестве «ответов». Таким образом, фильтр для отсеивания пакетов будет состоять из двух частей: фильтр перехвата «запросов» и фильтр перехвата «ответов». Ограничим количество захватываемых пакетов двадцатью пятью. Команда будет иметь вид:

```
sudo tcpdump -i enp0s3 -c 25 '(dst host 192.168.0.103 and src host 192.168.0.108 and ip proto \udp) or (dst host 192.168.0.108 and ip proto \icmp)'
```

192.168.0.103 – ip отправителя, 192.168.0.108 – ip получателя.

ВАЖНО!Пример работы с `tcpdump` в рамках ЛР.В одном окне запускаем `tcpdump` с указанием опций и фильтров, а в другом окне запустим `traceroute` до ip-получателя, чтобы сгенерировать поток пакетов.



```
user@user-pc:~$ sudo tcpdump -l enp0s3 -c 25 '(dst host 192.168.0.103 and src host 192.168.0.108 and ip proto \udp) or (dst host 192.168.0.108 and ip proto \icmp)'
[sudo] пароль для user:
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on enp0s3, link-type EN10MB (Ethernet), snapshot length 262144 bytes
23:20:24.570296 IP user-pc.39760 > 192.168.0.103.33434: UDP, length 32
23:20:24.570350 IP user-pc.49812 > 192.168.0.103.33435: UDP, length 32
23:20:24.570373 IP user-pc.36964 > 192.168.0.103.33436: UDP, length 32
23:20:24.570394 IP user-pc.59987 > 192.168.0.103.33437: UDP, length 32
23:20:24.570415 IP user-pc.57400 > 192.168.0.103.33438: UDP, length 32
23:20:24.570436 IP user-pc.37766 > 192.168.0.103.33439: UDP, length 32
23:20:24.570461 IP user-pc.58667 > 192.168.0.103.33440: UDP, length 32
23:20:24.570484 IP user-pc.60681 > 192.168.0.103.33441: UDP, length 32
23:20:24.570504 IP user-pc.57446 > 192.168.0.103.33442: UDP, length 32
23:20:24.570628 IP user-pc.45060 > 192.168.0.103.33443: UDP, length 32
23:20:24.570927 IP user-pc.57318 > 192.168.0.103.33444: UDP, length 32
23:20:24.570971 IP user-pc.47383 > 192.168.0.103.33445: UDP, length 32
23:20:24.571102 IP user-pc.46800 > 192.168.0.103.33446: UDP, length 32
23:20:24.571143 IP user-pc.41413 > 192.168.0.103.33447: UDP, length 32
23:20:24.571236 IP user-pc.51773 > 192.168.0.103.33448: UDP, length 32
23:20:24.571280 IP user-pc.51499 > 192.168.0.103.33449: UDP, length 32
23:20:29.572551 IP user-pc.36301 > 192.168.0.103.33450: UDP, length 32
23:20:29.572598 IP user-pc.35782 > 192.168.0.103.33451: UDP, length 32
23:20:29.572624 IP user-pc.49369 > 192.168.0.103.33452: UDP, length 32
23:20:29.572648 IP user-pc.59680 > 192.168.0.103.33453: UDP, length 32
23:20:29.572670 IP user-pc.57040 > 192.168.0.103.33454: UDP, length 32
23:20:29.572692 IP user-pc.35804 > 192.168.0.103.33455: UDP, length 32
23:20:29.572883 IP user-pc.40873 > 192.168.0.103.33456: UDP, length 32
23:20:29.572943 IP user-pc.47831 > 192.168.0.103.33457: UDP, length 32
23:20:29.572997 IP user-pc.43338 > 192.168.0.103.33458: UDP, length 32
25 packets captured
32 packets received by filter
0 packets dropped by kernel

user@user-pc:~$ traceroute 192.168.0.103
traceroute to 192.168.0.103 (192.168.0.103), 30 hops max, 60 byte packets
 1 * * *
 2 * * *
 3 * * *
 4 * * *
 5 * * *
 6 * * *
 7 * * *
 8 * * *
 9 * * *
10 * * *
11 * +AC
user@user-pc:~$
```

2.4. Порядок выполнения лабораторной работы

Важно: Команды, запускаемые для выполнения заданий, и результаты их работы необходимо сохранять для отчета.

1. Запустить `tcpdump` в режиме захвата всех пакетов, проходящих по сети (без фильтра). Количество захватываемых пакетов ограничить семью.

2. Запустить `tcpdump` в режиме перехвата широковещательного трафика. Фильтровать трафик и по широковещательному аппаратному MAC-адресу (FF:FF:FF:FF:FF:FF), и по широковещательному IP-адресу (можно посмотреть с помощью утилиты `ifconfig`). Фильтры должны быть связаны логическим объединением. Количество захватываемых пакетов ограничить пятью. Включить распечатку пакета в шестнадцатеричной системе (включая заголовок канального уровня).

3. Запустить `tcpdump` так, чтобы он перехватывал только пакеты протокола ICMP, отправленные на IP-адрес одного из лабораторных компьютеров. При этом включить распечатку пакета в шестнадцатеричной системе и ASCII-формате (включая заголовок канального уровня). Количество захватываемых пакетов ограничить восемью. Для генерирования пакетов воспользоваться утилитой `ping`.

4. По образцу рассмотренного в теории примера перехватить трафик утилиты *traceroute* при определении маршрута до какого-либо узла в сети Интернет. IP-адрес узла можно узнать с помощью сетевой утилиты *nslookup*:

```
nslookup domain - name
```

5. Используя утилиту *tcpdump*, отобразить дейтаграммы, принадлежащие соединению TCP между локальным лабораторным ПК и **кафедральным сервером**, и содержащие флаг SYN в заголовке транспортного уровня. Количество дейтаграмм ограничить двумя.

Следует напомнить, что для обработки полей флагов сегмента TCP необходимо использовать выражение `tcp[tcpflags]` с указанием конкретных значений заданных флагов. Например, для выполнения части данного задания можно воспользоваться конструкцией:

```
sudo tcpdump -l vnnSXX 'tcp [tcpflags]& tcp-syn !=0'
```

где опция *S* указывает утилите *tcpdump* отображать реальные номера последовательностей сегментов TCP (по умолчанию указываются номера относительно первого перехваченного сегмента), а аргумент регулярного выражения (в скобках) `tcp-syn !=0` указывает отбирать только те сегменты TCP, в поле флагов которых бит SYN не равен нулю.

6. Отобразить дейтаграммы UDP, пересылаемые между локальным лабораторным ПК и **сервером**, отправленные с номера порта UDP службы DNS, на диапазон портов назначения 10000–65535. Количество дейтаграмм ограничить десятью.

7. Отобразить дейтаграммы, принадлежащие соединениям TCP между локальным лабораторным ПК и **сервером**, установленные между номерами исходящих портов TCP со значением меньше 1024. Количество дейтаграмм ограничить двумя.

8. Отобразить дейтаграммы, пересылаемые между локальным лабораторным ПК и **сервером**, и использующие номера портов назначения (UDP или TCP) со значениями большими 1024. Количество дейтаграмм ограничить двумя.

9.Отобразить дейтаграммы, UDP пересылаемые между локальным лабораторным ПК и **сервером**, размер которых больше 50 байт, но не превышает 100 байт. Количество дейтаграмм ограничить десятью.

Следует напомнить, что для отбора дейтаграмм в соответствии с размером необходимо использовать выражение lessX или greater X, отображающее дейтаграммы размером (в байтах) меньше или больше X, соответственно.

10.Отобразить дейтаграммы IP, пересылаемые между локальным лабораторным ПК и **сервером**, принадлежащие соединению TCP, отправленные с порта источника менее 1024 на порт назначения более 10000, размер которых не превышает 100 байт.

2.5. Содержание отчета

3.Результаты работы с утилитой tcpdump.

Важно: перед каждым листингом необходимо указать задание и написать соответствующую команду.

2.6. Контрольные вопросы

- 1.Назначение и принцип работы анализаторов протоколов.
- 2.Структура заголовка кадра Ethernet.
- 3.МАС-адрес.
- 4.Протокол IP.
- 5.Структура заголовка IP-пакета.
- 6.IP-адрес.
- 7.Протоколы транспортного уровня TCP и UDP.
- 8.Структура заголовка TCP.
- 9.Структура заголовка UDP.
- 10.Понятие порта в протоколах транспортного уровня.
- 11.Виды и назначение флагов в заголовках протоколов транспортного уровня.

Лабораторная работа 3.

Ознакомление с системой и протоколом DNS

3.1. Цель работы

Лабораторная работа ставит цели закрепления теоретического материала по протоколам и программному обеспечению системы доменных имен.

3.2. Задачи

Ознакомиться с работой утилит host, nslookup, dig. Научиться делать обратный DNSзапрос,

3.3. Теоретический материал

DNS (англ. Domain Name System «система доменных имён») — компьютерная распределённая система для получения информации о доменах. Чаще всего используется для получения IP-адреса по имени хоста (компьютера или устройства), получения информации о маршрутизации почты и/или обслуживающих узлах для протоколов в домене (SRV-запись).

Распределённая база данных DNS поддерживается с помощью иерархии DNS-серверов, взаимодействующих по определённому протоколу.

Основой DNS является представление об иерархической структуре имени и зонах. Каждый сервер, отвечающий за имя, может передать ответственность за дальнейшую часть домена другому серверу (с административной точки зрения — другой организации или человеку), что позволяет возложить ответственность за актуальность информации на серверы различных организаций (людей), отвечающих только за «свою» часть доменного имени.

Начиная с 2010 года в систему DNS внедряются средства проверки целостности передаваемых данных, называемые DNS Security Extensions (DNSSEC). Передаваемые данные не шифруются, но их достоверность проверяется криптографическими способами. Внедряемый стандарт DANE обеспечивает передачу средствами DNS достоверной криптографической

информации (сертификатов), используемых для установления безопасных и защищённых соединений транспортного и прикладного уровней.

DNS обладает следующими характеристиками:

- Распределённость администрирования. Ответственность за разные части иерархической структуры несут разные люди или организации.
- Распределённость хранения информации. Каждый узел сети в обязательном порядке должен хранить только те данные, которые входят в его зону ответственности, и (возможно) адреса корневых DNS-серверов.
- Кэширование информации. Узел может хранить некоторое количество данных не из своей зоны ответственности для уменьшения нагрузки на сеть.
- Иерархическая структура, в которой все узлы объединены в дерево, и каждый узел может или самостоятельно определять работу нижестоящих узлов, или делегировать (передавать) их другим узлам.
- Резервирование. За хранение и обслуживание своих узлов (зон) отвечают (обычно) несколько серверов, разделённые как физически, так и логически, что обеспечивает сохранность данных и продолжение работы даже в случае сбоя одного из узлов.

Ключевыми понятиями DNS являются:

- Домен (англ. domain «область») – узел в дереве имён, вместе со всеми подчинёнными ему узлами. Структура доменного имени отражает порядок следования узлов в иерархии; доменное имя читается слева направо от младших доменов к доменам высшего уровня (в порядке повышения значимости): вверху находится корневой домен, ниже идут домены первого уровня (доменные зоны), затем — домены второго уровня, третьего и т. д. (например, для адреса `ru.wikipedia.org`. домен первого уровня – `org`, второго – `wikipedia`, третьего – `ru`).
- Поддомен (англ. subdomain) – подчинённый домен (например, `wikipedia.org` – поддомен домена `org`, а `ru.wikipedia.org` – домена `wikipedia.org`). Теоретически такое деление может достигать глубины 127

уровней, а каждая метка может содержать до 63 символов, пока общая длина вместе с точками не достигнет 254 символов. Но на практике регистраторы доменных имён используют более строгие ограничения.

- Ресурсная запись – единица хранения и передачи информации в DNS. Каждая ресурсная запись имеет имя (то есть привязана к определённому доменному имени, узлу в дереве имён), тип и поле данных, формат и содержание которого зависит от типа.

- Зона – часть дерева доменных имён (включая ресурсные записи), размещаемая как единое целое на некотором сервере доменных имён, а чаще – одновременно на нескольких серверах. Целью выделения части дерева в отдельную зону является передача ответственности за соответствующий домен другому лицу или организации. Это называется делегированием. Как связанная часть дерева, зона внутри тоже представляет собой дерево. Если рассматривать пространство имён DNS как структуру из зон, а не отдельных узлов/имён, тоже получается дерево; оправданно говорить о родительских и дочерних зонах, о старших и подчинённых. На практике большинство зон 0-го и 1-го уровня (ru, com, ...) состоят из единственного узла, которому непосредственно подчиняются дочерние зоны. В больших корпоративных доменах (2-го и более уровней) иногда встречается образование дополнительных подчинённых уровней без выделения их в дочерние зоны.

- Делегирование – операция передачи ответственности за часть дерева доменных имён другому лицу или организации. За счёт делегирования в DNS обеспечивается распределённость администрирования и хранения. Технически делегирование выражается в выделении этой части дерева в отдельную зону, и размещении этой зоны на DNS-сервере, управляемом этим лицом или организацией.

- DNS-сервер – специализированное ПО для обслуживания DNS, а также компьютер, на котором это ПО выполняется. DNS-сервер может быть ответственным за некоторые зоны и/или может перенаправлять запросы вышестоящим серверам.

- DNS-клиент – специализированная библиотека (или программа) для работы с DNS. В ряде случаев DNS-сервер выступает в роли DNS-клиента.

- Авторитетность (англ. authoritative) – признак размещения зоны на DNS-сервере. Ответы DNS-сервера могут быть двух типов: авторитетные (когда сервер заявляет, что сам отвечает за зону) и неавторитетные (англ. Non-authoritative), когда сервер обрабатывает запрос, и возвращает ответ других серверов. В некоторых случаях вместо передачи запроса дальше DNS-сервер может вернуть уже известное ему (по запросам ранее) значение (режим кеширования).

- DNS-запрос (англ. DNS query) – запрос от клиента (или сервера) серверу. Запрос может быть рекурсивным или нерекурсивным.

Система DNS содержит иерархию DNS-серверов, соответствующую иерархии зон. Каждая зона поддерживается как минимум одним авторитетным сервером DNS, на котором расположена информация о домене.

Имя и IP-адрес не тождественны – один IP-адрес может иметь множество имён, что позволяет поддерживать на одном компьютере множество веб-сайтов (это называется виртуальный хостинг). Обратное тоже справедливо – одному имени может быть сопоставлено множество IP-адресов: это позволяет создавать балансировку нагрузки.

Для повышения устойчивости системы используется множество серверов, содержащих идентичную информацию, а в протоколе есть средства, позволяющие поддерживать синхронность информации, расположенной на разных серверах.

Протокол DNS использует для работы TCP или UDP-порт 53 для ответов на запросы. Традиционно запросы и ответы отправляются в виде одной UDP-датаграммы. TCP используется, когда размер данных ответа превышает 512 байт.

Термином рекурсия в DNS обозначают алгоритм поведения DNS-сервера: «выполнить от имени клиента полный поиск нужной информации во всей системе DNS, при необходимости обращаясь к другим DNS-серверам».

DNS-запрос может быть рекурсивным –требующим полного поиска, –и нерекурсивным (или итеративным) –не требующим полного поиска.

Аналогично –DNS-сервер может быть рекурсивным (умеющим выполнять полный поиск) и нерекурсивным (не умеющим выполнять полный поиск).

В случае рекурсивного запроса DNS-сервер опрашивает серверы (в порядке убывания уровня зон в имени), пока не найдёт ответ или не обнаружит, что домен не существует (на практике поиск начинается с наиболее близких к искомому DNS-серверов, если информация о них есть в кэше и не устарела, сервер может не запрашивать другие DNS-серверы).

Рассмотрим на примере работу всей системы.

Предположим, мы набрали в браузере адрес `ru.wikipedia.org`. Браузер ищет соответствие этого адреса IP-адресу в файле `hosts`. Если файл не содержит соответствия, то далее браузер спрашивает у сервера DNS: «какой IP-адрес у `ru.wikipedia.org`»? Однако сервер DNS может ничего не знать не только о запрошенном имени, но и даже обо всём домене `wikipedia.org`. В этом случае сервер обращается к корневому серверу –например, `198.41.0.4`. Этот сервер сообщает – «У меня нет информации о данном адресе, но я знаю, что `204.74.112.1` является ответственным за зону `org`.» Тогда сервер DNS направляет свой запрос к `204.74.112.1`, но тот отвечает «У меня нет информации о данном сервере, но я знаю, что `207.142.131.234` является ответственным за зону `wikipedia.org`.» Наконец, тот же запрос отправляется к третьему DNS-серверу и получает ответ –IP-адрес, который и передаётся клиенту –браузеру.

DNS используется в первую очередь для преобразования символьных имён в IP-адреса, но он также может выполнять обратный процесс. Для этого

используются уже имеющиеся средства DNS. Дело в том, что с записью DNS могут быть сопоставлены различные данные, в том числе и какое-либо символьное имя. Существует специальный домен `in-addr.arpa`, записи в котором используются для преобразования IP-адресов в символьные имена. Например, для получения DNS-имени для адреса 11.22.33.44 можно запросить у DNS-сервера запись `44.33.22.11.in-addr.arpa`, и тот вернёт соответствующее символьное имя. Обратный порядок записи частей IP-адреса объясняется тем, что в IP-адресах старшие биты расположены в начале, а в символьных DNS-именах старшие (находящиеся ближе к корню) части расположены в конце.

3.4. Порядок выполнения лабораторной работы

3.4.1.Разрешение адресов в системе DNS с использованием различных утилит системы Linux

1.Запустить анализатор протоколов Wireshark и указать фильтр для перехвата трафика DNS.

2.В терминале последовательно выполнить разрешение доменных имен согласно варианту из ЛР1 с использованием трех утилит:

а) `host`

б) `nslookup`

в) `dig`

3.Остановить захват пакетов в анализаторе протоколов Wireshark.

4.Проанализировать вывод команд и перехваченные пакеты.

5.В отчете привести последовательно вывод каждой команды, сопроводив его соответствующими перехваченными пакетами и выводами по работе программ.

3.4.2.Получение ресурсных записей различных типов с использованием утилиты *nslookup*

1.Запустить анализатор протоколов Wireshark и указать фильтр для перехвата трафика DNS.

2.Последовательно получить от сервера DNS следующие ресурсные записи для доменного имени согласно варианту:

- а) адреса IPv4
- б) адреса IPv6
- в) почтовые серверы
- г) серверы DNS
- д) авторитетный сервер для доменного имени

3.Остановить захват пакетов в анализаторе протоколов Wireshark.

4.Проанализировать вывод команд и перехваченные пакеты.

В отчете привести последовательно вывод каждой команды, сопроводив его соответствующими перехваченными пакетами и выводами по работе программ.

3.4.3.Проведение обратного запроса DNS с использованием утилиты *host*

1.Запустить анализатор протоколов Wireshark и указать фильтр для перехвата трафика DNS.

2.Провести обратный запрос DNS для IPv4-адреса из предыдущего пункта.

3.Провести обратный запрос DNS для IPv6-адреса из предыдущего пункта. При формировании команды запроса руководствоваться примером ресурсной записи:

```
host -t PTR a.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.a.0.0.0.8.b.6.0.2.0.a.2.ip6.arpa
```

4.Остановить захват пакетов в анализаторе протоколов Wireshark.

5.Проанализировать вывод команд и перехваченные пакеты.

6.В отчете привести последовательно вывод каждой команды, сопроводив его соответствующими перехваченными пакетами и выводами по работе программ.

3.4.4. Получение всех ресурсных записей для определенного доменного имени с использованием утилиты *host*

1. Запустить анализатор протоколов Wireshark и указать фильтр для перехвата трафика: IP-адреса ПК и сервера DNS, протоколы (TCP или UDP).

2. Выполнить запрос «ресурсной записи» ANY для доменного имени согласно варианту.

3. Остановить захват пакетов в анализаторе протоколов Wireshark.

4. Проанализировать вывод команд и перехваченные пакеты.

5. В отчете привести вывод команды, сопроводив его соответствующими перехваченными пакетами и выводами по процедуре получения записи.

3.5. Содержание отчета

1. Заголовок согласно приложению.

2. Цель работы.

3. Результаты работы по каждому из пунктов работы с соответствующими выводами.

3.6. Контрольные вопросы

1. Что такое система доменных имён?

2. Для чего используется файл hosts?

3. Каковы ключевые характеристики DNS?

4. Что такое домен и поддомен?

5. Что такое корневой домен?

6. Что такое рекурсия в DNS?

7. Как выполняется DNS-запрос?

8. Что такое обратный DNS-запрос?

