

Лабораторная работа №2

Тема: Анализ работы протокола TCP.

Студента группы МФ-21
Чистякова Артема

1.

Определить хосты (IP-адреса и протоколы, порты), взаимодействующие с рабочим местом.

Решение

Запустим **Wireshark**, настроим на отслеживание трафика сетевой карты: на моем компьютере – интерфейс **wlp1s0**.

Увидим следующую таблицу:

No.	Time	Source	Destination	Protocol	Length	Info
10.000000000	173.194.221.189	10.0.1.44	10.0.1.44	UDP	82	443 - 47480 Len=40
20.011457446	10.0.1.44	173.194.221.189	10.0.1.44	UDP	71	47480 - 443 Len=29
31.761939602	fe80::55a3:b1ca:d97:f02::fb	fe80::55a3:b1ca:d97:f02::fb	fe80::55a3:b1ca:d97:f02::fb	MDNS	103	Standard query 0x0000 A Annas-MacBook-Pro.local, "QM" question
41.764584441	10.0.1.44	224.0.0.251	224.0.0.251	MDNS	83	Standard query 0x0000 A Annas-MacBook-Pro.local, "QM" question
52.763490591	fe80::55a3:b1ca:d97:f02::fb	fe80::55a3:b1ca:d97:f02::fb	fe80::55a3:b1ca:d97:f02::fb	MDNS	103	Standard query 0x0000 A Annas-MacBook-Pro.local, "QM" question
62.763647884	10.0.1.44	224.0.0.251	224.0.0.251	MDNS	83	Standard query 0x0000 A Annas-MacBook-Pro.local, "QM" question
73.799829502	10.0.1.44	10.0.1.1	10.0.1.1	DNS	89	Standard query 0x51ce A clientservices.googleapis.com
83.812626602	10.0.1.1	10.0.1.44	10.0.1.44	DNS	105	Standard query response 0x51ce A clientservices.googleapis.com A 172.217.16.35
93.816294704	10.0.1.44	172.217.16.35	172.217.16.35	UDP	1392	60499 - 443 Len=1350
103.816722318	10.0.1.44	172.217.16.35	172.217.16.35	UDP	341	60499 - 443 Len=299
113.833894952	10.0.1.44	239.255.255.250	239.255.255.250	SSDP	213	M-SEARCH * HTTP/1.1
123.857324716	172.217.16.35	10.0.1.44	10.0.1.44	UDP	1392	443 - 60499 Len=1350
133.858202987	10.0.1.44	172.217.16.35	172.217.16.35	UDP	70	60499 - 443 Len=28
143.876137312	172.217.16.35	10.0.1.44	10.0.1.44	UDP	304	443 - 60499 Len=262
153.876450886	172.217.16.35	10.0.1.44	10.0.1.44	UDP	58	443 - 60499 Len=16
163.876575402	10.0.1.44	172.217.16.35	172.217.16.35	UDP	70	60499 - 443 Len=28
174.765538395	fe80::55a3:b1ca:d97:f02::fb	fe80::55a3:b1ca:d97:f02::fb	fe80::55a3:b1ca:d97:f02::fb	MDNS	103	Standard query 0x0000 A Annas-MacBook-Pro.local, "QM" question
184.768130672	10.0.1.44	224.0.0.251	224.0.0.251	MDNS	83	Standard query 0x0000 A Annas-MacBook-Pro.local, "QM" question
194.834505172	10.0.1.44	239.255.255.250	239.255.255.250	SSDP	213	M-SEARCH * HTTP/1.1
205.195357794	10.0.1.44	151.101.114.49	151.101.114.49	TLSv1.2	140	Application Data
215.199959366	10.0.1.44	151.101.114.49	151.101.114.49	TLSv1.2	112	Application Data
225.237600880	151.101.114.49	10.0.1.44	10.0.1.44	TLSv1.2	282	Application Data
235.237609638	10.0.1.44	151.101.114.49	151.101.114.49	TCP	66	53986 - 443 [ACK] Seq=121 Ack=217 Win=18471 Len=0 TSval=527386283 TSecr=983036690
245.238580516	151.101.114.49	10.0.1.44	10.0.1.44	TLSv1.2	112	Application Data
255.238622991	10.0.1.44	151.101.114.49	151.101.114.49	TCP	66	53986 - 443 [ACK] Seq=121 Ack=263 Win=18471 Len=0 TSval=527386284 TSecr=983036900
265.835312096	10.0.1.44	239.255.255.250	239.255.255.250	SSDP	213	M-SEARCH * HTTP/1.1
276.836110399	10.0.1.44	239.255.255.250	239.255.255.250	SSDP	213	M-SEARCH * HTTP/1.1
287.591235877	10.0.1.44	151.101.114.49	151.101.114.49	TLSv1.2	136	Application Data
297.543066396	151.101.114.49	10.0.1.44	10.0.1.44	TCP	1514	443 - 53986 [ACK] Seq=263 Ack=191 Win=79 Len=1448 TSval=983036666 TSecr=527388546 [TCP segment of a reassembled PDU]
307.543128518	10.0.1.44	151.101.114.49	151.101.114.49	TCP	66	53986 - 443 [ACK] Seq=191 Ack=1711 Win=18471 Len=0 TSval=527388588 TSecr=983036666
317.547301375	151.101.114.49	10.0.1.44	10.0.1.44	TLSv1.2	1406	Application Data
327.547344054	10.0.1.44	151.101.114.49	151.101.114.49	TCP	66	53986 - 443 [ACK] Seq=191 Ack=3051 Win=18471 Len=0 TSval=527388593 TSecr=983036666
337.547390680	151.101.114.49	10.0.1.44	10.0.1.44	TCP	1514	443 - 53986 [ACK] Seq=3051 Ack=191 Win=79 Len=1448 TSval=983036666 TSecr=527388546 [TCP segment of a reassembled PDU]
347.547404095	10.0.1.44	151.101.114.49	151.101.114.49	TCP	66	53986 - 443 [ACK] Seq=191 Ack=4499 Win=18461 Len=0 TSval=527388593 TSecr=983036666
357.547411148	151.101.114.49	10.0.1.44	10.0.1.44	TLSv1.2	1514	Application Data [TCP segment of a reassembled PDU]
367.547417260	10.0.1.44	151.101.114.49	151.101.114.49	TCP	66	53986 - 443 [ACK] Seq=191 Ack=5947 Win=18452 Len=0 TSval=527388593 TSecr=983036666
377.547425115	151.101.114.49	10.0.1.44	10.0.1.44	TCP	1406	443 - 53986 [PSH, ACK] Seq=5947 Ack=191 Win=79 Len=1340 TSval=983036666 TSecr=527388546 [TCP segment of a reassembled PDU]
387.547433329	10.0.1.44	151.101.114.49	151.101.114.49	TCP	66	53986 - 443 [ACK] Seq=191 Ack=7287 Win=18443 Len=0 TSval=527388593 TSecr=983036666
397.547440089	151.101.114.49	10.0.1.44	10.0.1.44	TLSv1.2	1406	Application Data
407.547447368	10.0.1.44	151.101.114.49	151.101.114.49	TCP	66	53986 - 443 [ACK] Seq=191 Ack=8627 Win=18434 Len=0 TSval=527388593 TSecr=983036666
417.547455778	151.101.114.49	10.0.1.44	10.0.1.44	TCP	1514	443 - 53986 [ACK] Seq=8627 Ack=191 Win=79 Len=1448 TSval=983036666 TSecr=527388546 [TCP segment of a reassembled PDU]
427.547460802	10.0.1.44	151.101.114.49	151.101.114.49	TCP	66	53986 - 443 [ACK] Seq=191 Ack=10975 Win=18424 Len=0 TSval=527388593 TSecr=983036666
437.547466440	151.101.114.49	10.0.1.44	10.0.1.44	TLSv1.2	1406	Application Data
447.547473554	10.0.1.44	151.101.114.49	151.101.114.49	TCP	66	53986 - 443 [ACK] Seq=191 Ack=11415 Win=18415 Len=0 TSval=527388593 TSecr=983036666
* Frame 220: 1392 bytes on wire (11136 bits), 1392 bytes captured (11136 bits) on interface 0						
* Ethernet II, Src: RivetNet, If: 04:5f:9c:b6:d0:1f:04:5f, Dst: Apple, cs:03:2a (b8:c7:5d:c5:e3:2a)						
* Internet Protocol Version 4, Src: 10.0.1.44, Dst: 172.217.16.10						
* User Datagram Protocol, Src Port: 51643, Dst Port: 443						
* Data (1350 bytes)						
0000	b8 c7 5d c5 e3 2a 9c b6 d0 1f 04 5f 00 00 45 00	[...]				
0010	05 02 f6 d6 40 00 71 a5 0a 00 01 2c ac d9	[...]				
0020	10 0a c9 bb 81 bb 05 4e cd 6e 40 49 63 25 9d 7f	[...]				
0030	c8 9b 7d 03 5a 2b 03 84 69 19 ac 97 49 ac a0 19	[...]				
0040	ed a5 1b 31 42 ad 03 2d 56 41 45 db c1 e9 f9 7a	[...]				
0050	44 57 73 d7 e3 a3 a9 c9 99 08 4f ea 54 67 91 fd	[...]				
0060	d3 a0 56 28 c3 eb bc 7f 52 8d dd ee 09 77 65 8d	[...]				

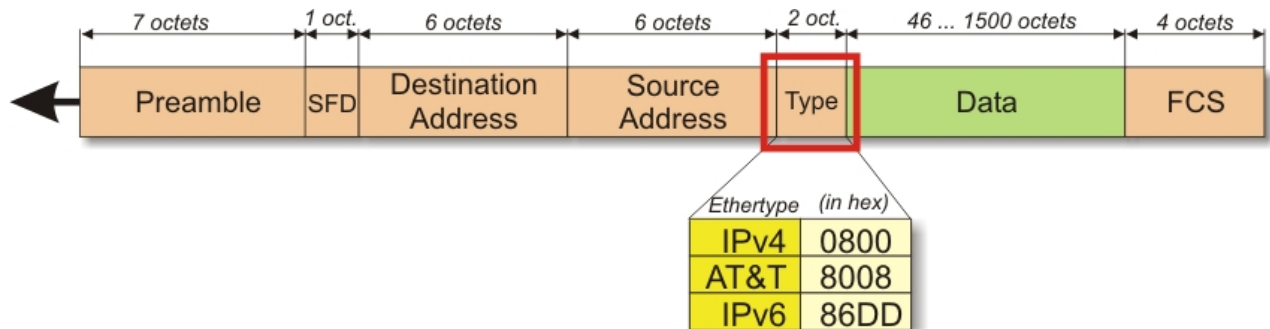
На картинке можно заметить как входящие, так и исходящие пакеты: IP компьютера – **10.0.1.44**.

2.

Определить формат Ethernet кадра.

Решение

Рассмотрим следующую иллюстрацию:



- **Поле преамбулы** состоит из семи байтов синхронизирующих данных. Каждый байт содержит одну и ту же последовательность битов - 10101010.
- **Начальный ограничитель кадра** состоит из одного байта с набором битов 10101011. Появление этой комбинации является указанием на предстоящий прием кадра.
- **Адрес получателя** – может быть длиной 2 или 6 байтов (MAC-адрес получателя). Первый бит адреса получателя - это признак того, является адрес индивидуальным или групповым: если 0, то адрес указывает на определенную станцию, если 1, то это групповой адрес нескольких (возможно всех) станций сети. При широковещательной адресации все биты поля адреса устанавливаются в 1. Общепринятым является использование 6-байтовых адресов.
- **Адрес отправителя** – 2-х или 6-ти байтовое поле, содержащее адрес станции отправителя. Первый бит - всегда имеет значение 0.
- **Поле типа протокола.** Это поле предназначено для указания типа протокола верхнего уровня, вложившего свой пакет в поле данных этого кадра.

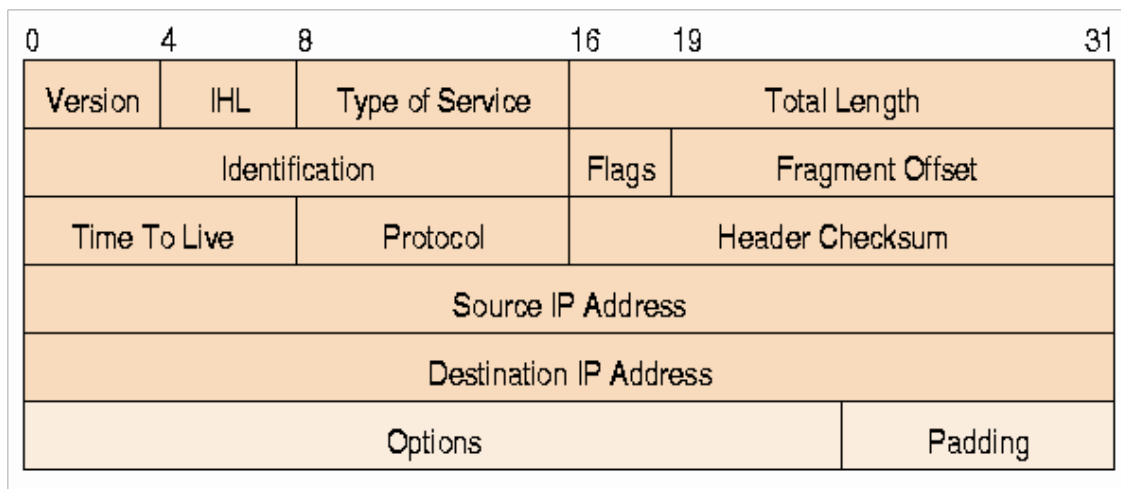
- **Поле данных** может содержать от 0 до 1500 байт. Но если длина поля меньше 46 байт, то используется следующее поле - поле заполнения, чтобы дополнить кадр до минимально допустимой длины.
- **Поле контрольной суммы** – 4 байта, содержащие значение, которое вычисляется по определенному алгоритму (полиному CRC-32). После получения кадра рабочая станция выполняет собственное вычисление контрольной суммы для этого кадра, сравнивает полученное значение со значением поля контрольной суммы и, таким образом, определяет, не искажен ли полученный кадр.

3.

Определить формат IP пакета.

Решение

Рассмотрим следующую иллюстрацию:



Протокол IP является надстройкой над протоколом Ethernet – его header вложен в поле данных протокола Ethernet.

- **Поле номера версии** занимает 4 бита и идентифицирует версию протокола IP. Сейчас повсеместно используется версия 4 (IPv4), хотя все чаще встречается и новая версия (IPv6).
- **Значение длины заголовка** IP-пакета занимает 4 бита и измеряется в 32-битных словах. Обычно заголовок имеет длину в 20 байт (пять 32-битных слов), но при добавлении некоторой служебной информации это значение может быть увеличено. Наибольшая длина заголовка составляет 60 байт.
- **Поле типа сервиса** (Type of Service, ToS) служит одной цели – хранению признаков, которые отражают требования к качеству обслуживания пакета. Первые три бита содержат значение приоритета пакета: от самого низкого – 0 до самого высокого – 7. Следующие три бита поля ToS определяют критерий выбора маршрута. Если бит D установлен в 1, то маршрут должен выбираться для минимизации задержки доставки данного пакета, установленный бит T – для максимизации пропускной способности, а бит R – для максимизации надежности доставки. Оставшиеся два бита имеют нулевое значение.
- **Поле общей длины** занимает 2 байта и характеризует общую длину пакета с учетом заголовка и поля данных. Если это кадры Ethernet, то выбираются пакеты с максимальной длиной 1500 байт, уместяющиеся в поле данных кадра Ethernet. В стандартах TCP/IP предусматривается, что все хосты должны быть готовы принимать пакеты длиной вплоть до 576 байт.
- **Идентификатор пакета** занимает 2 байта и используется для распознавания пакетов, образовавшихся путем деления на части исходного пакета. Все части одного пакета должны иметь одинаковое значение этого поля.
- **Флаги** занимают 3 бита и содержат признаки, связанные с фрагментацией. Установленный в 1 бит DF запрещает маршрутизатору фрагментировать данный пакет, а установленный в 1 бит MF говорит о том, что данный пакет является промежуточным (не последним) фрагментом. Оставшийся бит зарезервирован.
- **Поле смещения фрагмента** занимает 13 бит и задает смещение в байтах поля данных этого фрагмента относительно начала поля данных исходного

пакета. Используется при сборке/разборке фрагментов пакетов. Смещение должно быть кратно 8 байт.

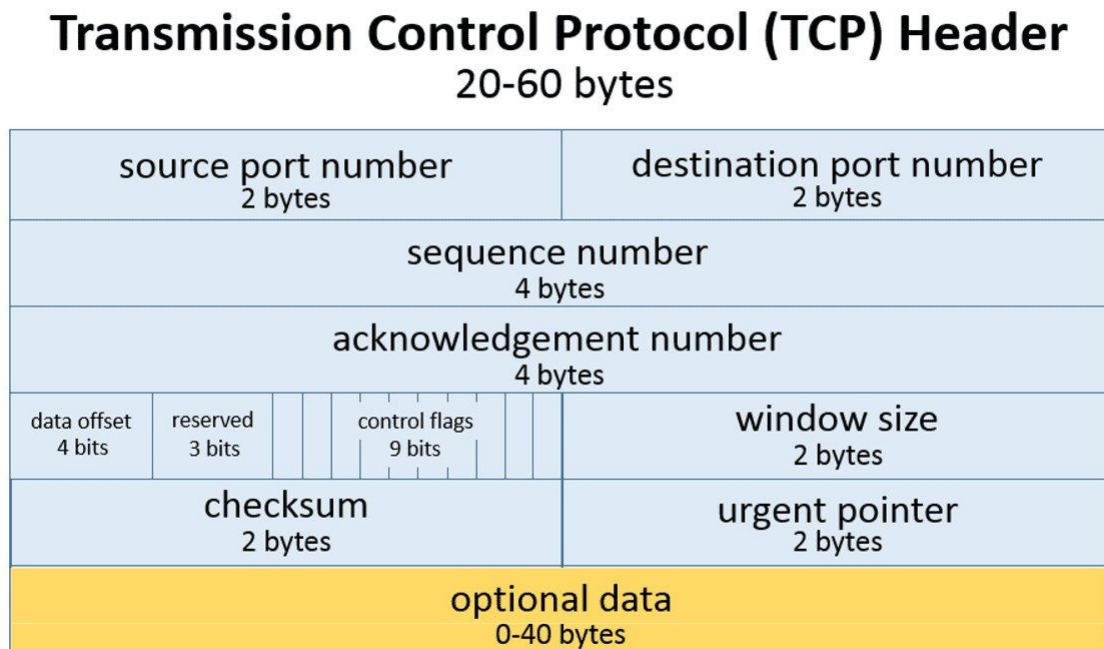
- **Поле времени жизни** (Time To Live, TTL) занимает один байт и используется для задания предельного срока, в течение которого пакет может перемещаться по сети. Время жизни можно интерпретировать как максимальное число транзитных узлов, которые разрешено пройти пакету.
- **Поле протокола верхнего уровня** занимает один байт и содержит идентификатор, указывающий, какому протоколу верхнего уровня принадлежит информация, размещенная в поле данных пакета. Например, 6 означает, что в пакете находится сообщение протокола TCP, 17 – протокола UDP.
- **Контрольная сумма** заголовка занимает 2 байта (16 бит) и рассчитывается только по заголовку. Контрольная сумма проверяется и повторно рассчитывается на каждом маршрутизаторе и конечном узле. Если контрольная сумма неверна, то пакет отбрасывается.
- **Поля IP-адресов** источника и приемника имеют одинаковую длину — 32 бита.
- **Поле параметров** является необязательным и используется обычно только при отладке сети. В этом поле можно указывать точный маршрут, регистрировать проходимые пакетом маршрутизаторы, помещать данные системы безопасности или временные отметки. В конце заголовка должно быть добавлено несколько нулевых байтов для выравнивания заголовка пакета по 32-битной границе.

4.

Определить формат TCP пакета.

Решение

Рассмотрим следующую иллюстрацию:



Протокол TCP является надстройкой над протоколами IP, его header располагается сразу после header'a IP, в том же поле данных протокола Ethernet. Он гарантирует порядок и доставку отправленных данных.

- **Порт источника и порт приемника** – 16-битовые поля, содержащие номера портов, соответственно, источника и адресата TCP-пакета.
- **Номер в последовательности** – 32-битовое поле, содержимое которого определяет «номер TCP пакета данных» внутри исходящего потока данных, существующего в рамках текущего логического соединения.

- **Номер подтверждения** – 32-битовое поле, содержимое которого определяет количество принятых TCP пакетов данных из входящего потока.
- **Смещение данных** – четырехбитовое поле, содержащее длину заголовка TCP-пакета в 32-битовых словах и используемое для определения начала расположения данных в TCP-пакете.
- **Флаг URG** – бит, установленное в 1 значение которого означает, что TCP-пакет содержит важные (urgent) данные.
- **Флаг ACK** – бит, значение которого означает, что TCP-пакет содержит в поле "номер подтверждения" верные данные.
- **Флаг PSH** – бит, значение которого означает, что данные содержащиеся в TCP-пакете должны быть немедленно переданы прикладной программе, для которой они адресованы. Подтверждение для TCP-пакета, содержащего единичное значение во флаге PSH, означает, что и все предыдущие TCP-пакеты достигли адресата.
- **Флаг RST** – бит в TCP-пакете, отправляемом в ответ на получение неверного TCP-пакета. Также может означать запрос на переустройство логического соединения.
- **Флаг SYN** – бит, означающий, что TCP-пакет представляет собой запрос на установление логического соединения. Получение пакета с установленным флагом SYN должно быть подтверждено принимающей стороной.
- **Флаг FIN** – бит, означающий, что TCP-пакет представляет собой запрос на закрытие логического соединения и является признаком конца потока данных, передаваемых в этом направлении. Получение пакета с установленным флагом FIN должно быть подтверждено принимающей стороной.
- **Размер окна** – 16-битовое поле, содержащее количество байт информации, которое может принять в свои внутренние буфера TCP-модуль. Определение оптимального размера окна - одна из наиболее сложных задач реализации протокола TCP.
- **Контрольная сумма** – 16-битовое поле, содержащее Internet-контрольную сумму, подсчитанную для TCP-заголовка, данных пакета и псевдозаголовка.

- **Псевдозаголовок** состоит из IP-адреса отправителя, IP-адреса получателя, типа протокола и длины TCP-пакета. Он предназначен для «страховки» неправильной маршрутизации TCP-пакета.

5.

Определить формат UDP пакета.

Решение

Рассмотрим следующую иллюстрацию:



Протокол UDP является надстройкой над протоколами IP, его header располагается сразу после header`а IP, в том же поле данных протокола Ethernet. В отличие от протокола TCP, он не гарантирует доставку и порядок отправляемых данных.

- **Порт отправителя** – поле может содержать номер порта, с которого был отправлен пакет, когда это имеет значение (например отправитель ожидает ответа). Если это поле не используется, оно заполняется нулями.
- **Порт назначения** – это порт компьютера, на который пакет будет доставлен.

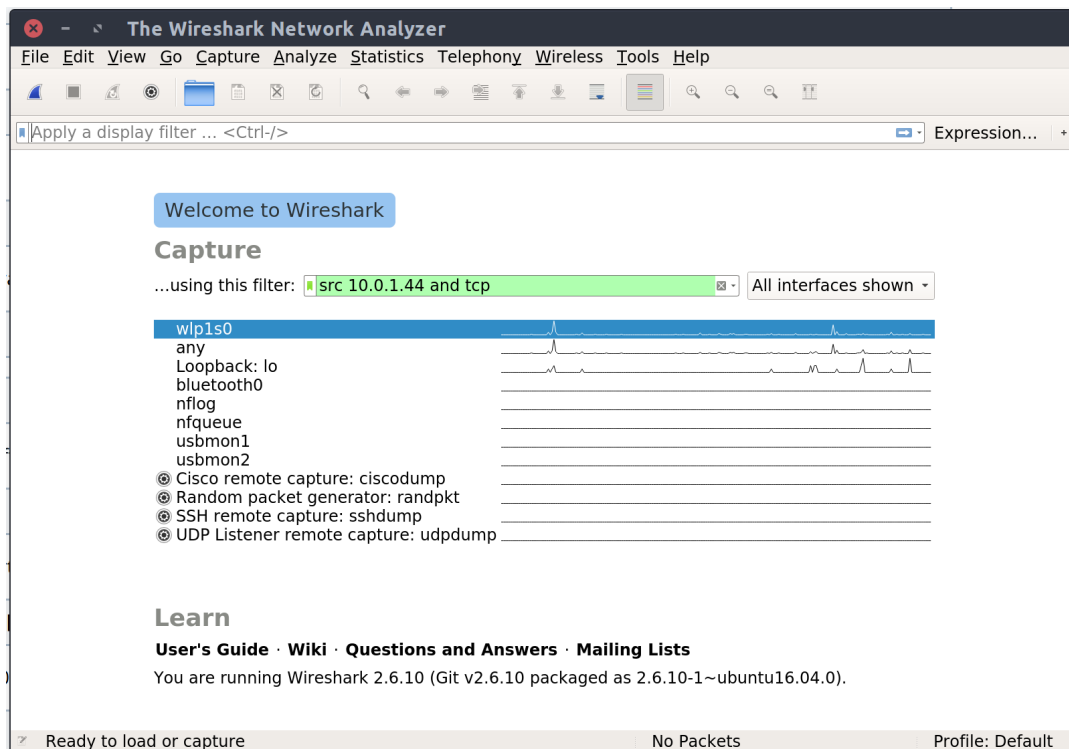
- **Поле длины** – длина (в байтах) этой дейтаграммы, включая заголовок и данные. (Минимальное значение этого поля равно 8).
- **Поле контрольной суммы** – контрольная сумма UDP-пакета, представляющая собой побитное дополнение 16-битной суммы 16-битных слов (аналогично TCP). В вычислении участвуют: данные пакета, заголовок UDP-пакета, псевдозаголовок, поля выравнивания по 16-битной границе.

6.

Настроить анализатор протоколов на перехват только исходящего TCP трафика на рабочем месте.

Решение

Настроим фильтр пакетов следующим образом:



Запустим sniffer.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.1.44	149.154.167.51	TCP	66	41834 → 443 [ACK] Seq=1 Ack=...
2	0.089758895	10.0.1.44	40.115.22.134	TCP	66	50456 → 443 [ACK] Seq=1 Ack=...
3	0.857735541	10.0.1.44	140.82.114.26	TCP	66	49932 → 443 [ACK] Seq=1 Ack=...
4	1.529696117	10.0.1.44	149.154.167.51	SSL	219	Continuation Data
5	1.571949711	10.0.1.44	149.154.167.51	TCP	66	41834 → 443 [ACK] Seq=154 Ac...
6	9.215672925	10.0.1.44	140.82.114.26	TLSv1.2	94	[TCP Previous segment not ca...
7	11.058495120	10.0.1.44	149.154.167.51	TCP	66	41834 → 443 [ACK] Seq=154 Ac...
8	11.058977261	10.0.1.44	149.154.167.51	TCP	155	41834 → 443 [PSH, ACK] Seq=1...
9	12.021709646	10.0.1.44	149.154.167.51	TCP	66	41834 → 443 [ACK] Seq=243 Ac...
10	12.107586813	10.0.1.44	149.154.167.51	TCP	66	41834 → 443 [ACK] Seq=243 Ac...
11	12.594464319	10.0.1.44	149.154.167.51	TCP	66	41834 → 443 [ACK] Seq=243 Ac...
12	13.872951689	10.0.1.44	149.154.167.51	TCP	66	41834 → 443 [ACK] Seq=243 Ac...
13	16.895353912	10.0.1.44	13.69.75.239	TCP	66	59742 → 443 [ACK] Seq=1 Ack=...
14	17.816925896	10.0.1.44	149.154.167.51	TCP	66	41834 → 443 [ACK] Seq=243 Ac...
15	18.521773937	10.0.1.44	173.194.222.188	TCP	66	49500 → 5228 [ACK] Seq=1 Ack=...
16	19.967366266	10.0.1.44	149.154.167.51	TCP	66	41834 → 443 [ACK] Seq=243 Ac...
17	20.967472972	10.0.1.44	151.101.114.49	TLSv1.2	136	Application Data
18	20.972254283	10.0.1.44	151.101.114.49	TLSv1.2	112	Application Data
19	21.049721174	10.0.1.44	151.101.114.49	TCP	66	46466 → 443 [ACK] Seq=117 Ac...
20	21.049985891	10.0.1.44	151.101.114.49	TCP	66	46466 → 443 [ACK] Seq=117 Ac...
21	21.050028882	10.0.1.44	151.101.114.49	TCP	66	46466 → 443 [ACK] Seq=117 Ac...
22	21.050647600	10.0.1.44	151.101.114.49	TCP	66	46466 → 443 [ACK] Seq=117 Ac...
23	21.050676200	10.0.1.44	151.101.114.49	TCP	66	46466 → 443 [ACK] Seq=117 Ac...
24	21.050696519	10.0.1.44	151.101.114.49	TCP	66	46466 → 443 [ACK] Seq=117 Ac...
25	21.050709217	10.0.1.44	151.101.114.49	TCP	66	46466 → 443 [ACK] Seq=117 Ac...

Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface 0
Ethernet II, Src: RivetNet_1f:04:5f (9c:b6:d0:1f:04:5f), Dst: Apple_c5:e3:2a (b8:c7:5d:c5:e3:2a)
Internet Protocol Version 4, Src: 10.0.1.44, Dst: 149.154.167.51
Transmission Control Protocol, Src Port: 41834, Dst Port: 443, Seq: 1, Ack: 1, Len: 0

wlp1s0: <live capture in progress> Packets: 55 · Displayed: 55 (100.0%) Profile: Default

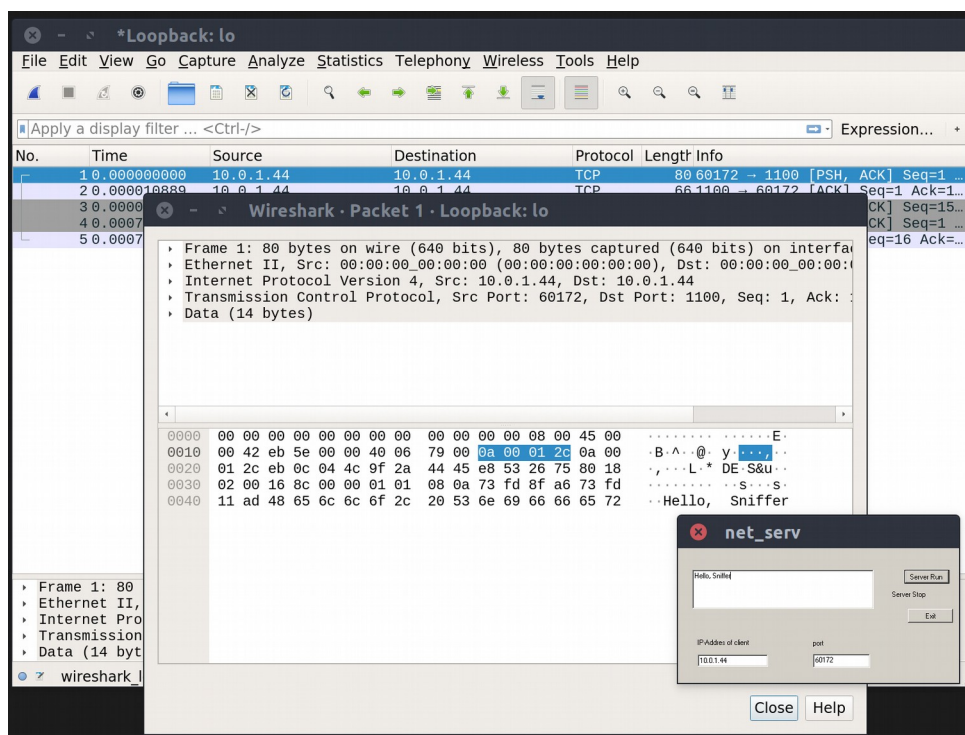
7.

Настроить анализатор протоколов на перехват всего входящего трафика на сервер.

Решение

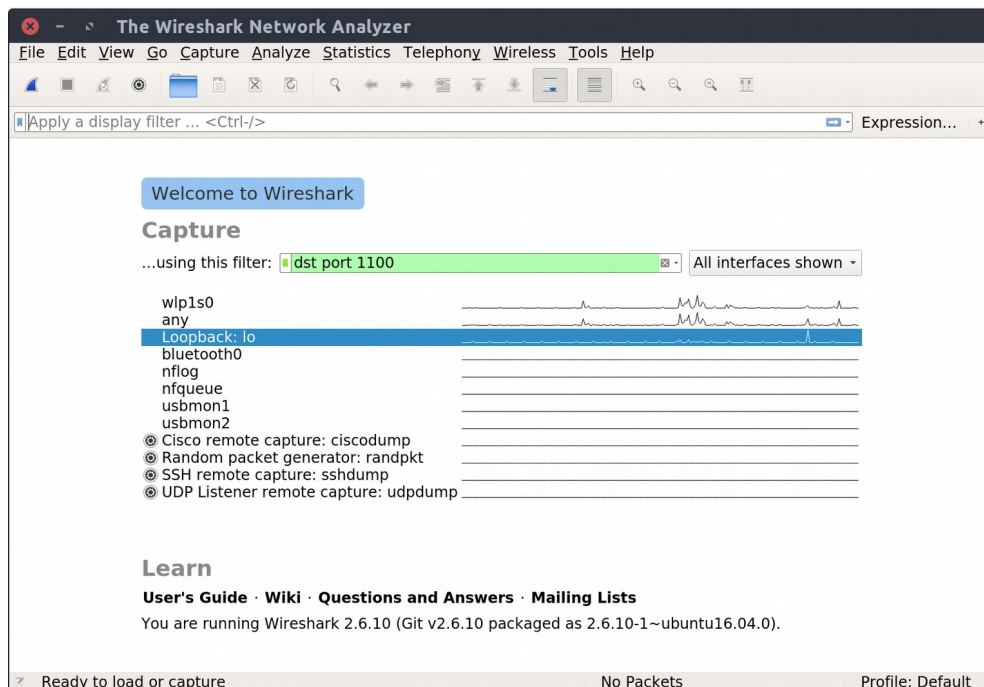
Для решения данной задачи нужно запустить sniffer для отслеживания всех входящих пакетов на рабочую машину, найти в них пакет, который отправил клиент и принял сервер и извлечь из него номер порта, на котором запущен сервер.

Запустим sniffer и найдем соответствующий пакет:



Убедимся, что данные соответствуют передаваемому сообщению. Найдем поле **Dst port = 1100**. Этот порт соответствует порту, на котором запущен сервер.

Настроим Wireshark на прослушивание порта **1100**:



Результат прослушивания порта **1100** при отправке клиентом сообщения:

The image shows a Wireshark packet capture window titled "Capturing from Loopback: lo (dst port 1100)". The packet list contains five entries, with the third packet (No. 3) selected. The packet details pane shows the structure of the selected packet: Ethernet II, Internet Protocol Version 4, and Transmission Control Protocol (TCP). The status bar at the bottom indicates "Loopback: lo: <live capture in progress>" and "Packets: 5 · Displayed: 5 (100.0%) Profile: Default".

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.1.44	10.0.1.44	TCP	74	60508 → 1100 [SYN] Seq=0 Win=6...
2	0.000057069	10.0.1.44	10.0.1.44	TCP	66	60508 → 1100 [ACK] Seq=1 Ack=1...
3	7.221445021	10.0.1.44	10.0.1.44	TCP	77	60508 → 1100 [PSH, ACK] Seq=1 ...
4	7.221590482	10.0.1.44	10.0.1.44	TCP	66	60508 → 1100 [FIN, ACK] Seq=12...
5	7.223887319	10.0.1.44	10.0.1.44	TCP	66	60508 → 1100 [ACK] Seq=13 Ack=...

Frame 3: 77 bytes on wire (616 bits), 77 bytes captured (616 bits) on interface 0
Ethernet II, Src: 00:00:00_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00_00:00:00 (00:00:00:00:00:00)
Internet Protocol Version 4, Src: 10.0.1.44, Dst: 10.0.1.44
Transmission Control Protocol, Src Port: 60508, Dst Port: 1100, Seq: 1, Ack: 1, Len: 11 Source Port: 60508

Loopback: lo: <live capture in progress> Packets: 5 · Displayed: 5 (100.0%) Profile: Default

8.

Алгоритм установления соединения.

Решение

Когда мы хотим установить соединение, мы отправляем удаленной системе пакет следующей структуры:

Client --- SYN (856779) --- Host

Где Client- это мы, а Host - это удаленная система. Мы посылаем пакет лишь с указанием SYN - это значит, что этот пакет первый.

SYN образуется от первоначального номера очереди (ISN) - это 32-битный номер от 1 до 4294967295 (2 в 32-ой степени). ISN при перезагрузке системы равен 1, затем каждую секунду он увеличивается на 128000 (строго говоря изменение происходит каждые 4 микросекунды) + при каждом установленном соединении он увеличивается на 64000. Получается, что цикл уникальности ISN, при условии того, что никакие соединения не устанавливались, составляет примерно 4,55 часа. Поскольку ни один пакет так долго по сети не путешествует, мы можем полагать, что SYN будет абсолютно уникальным.

Получив наш пакет, удаленная система отвечает, что получила и готова установить соединение. Данные пакета выглядят так:

Host --- SYN (758684758) и ACK (856780) --- Client

Удаленная система посылает нам ACK с номером "наш SYN+1". В добавок к этому свой SYN (мы же тоже будем отвечать). А ответ наш будет такой:

Client --- SYN (856780) и ACK (758684759) --- Host

Пакет означает следующее: ваш пакет с SYN (758684758) получен, соединение установлено, наш SYN равен 856780.

Эту процедуру называют "трехкратным рукопожатием". Первые два этапа необходимы для синхронизации SYN систем, а третий - подтверждение того, что синхронизация произошла.

9.

Алгоритм передачи данных.

Решение

Модуль ТСР обеспечивает защиту от повреждения, потери, дублирования и нарушения очередности получения данных.

Для выполнения этих задач все октеты в потоке данных сквозным образом пронумерованы в возрастающем порядке. Заголовок каждого сегмента содержит число октетов данных в сегменте и порядковый номер первого октета той части потока данных, которая пересылается в данном сегменте. Например, если в сегменте пересылаются октеты с номерами от 2001 до 3000, то номер первого октета в данном сегменте равен 2001, а число октетов равно 1000.

Также для каждого сегмента вычисляется контрольная сумма, позволяющая обнаружить повреждение данных.

При удачном приеме октета данных принимающий модуль посылает отправителю подтверждение о приеме - номер удачно принятого октета. Если в течение некоторого времени отправитель не получит подтверждения, считается, что октет не дошел или был поврежден, и он посылается снова. В действительности подтверждение посылается не для одного октета, а для некоторого числа последовательных октетов.

Нумерация октетов используется также для упорядочения данных в порядке очередности и обнаружения дубликатов (которые могут быть посланы из-за большой задержки при передаче подтверждения или потери подтверждения).

10.

Алгоритм завершения соединения.

Решение

Алгоритм завершение соединения аналогичен алгоритму установки соединения, но вместо флага SYN устанавливается флаг FIN, разрыв соединения можно рассмотреть в три этапа:

- Посылка серверу от клиента флага FIN на завершение соединения.
- Сервер посылает клиенту флаги ответа ACK , FIN, что соединение закрыто.
- После получения этих флагов клиент закрывает соединение и в подтверждение отправляет серверу ACK , что соединение закрыто.