Практическая работа №10
Анализ трафика компьютерной сети с помощью снифферов

Цель работы – приобретение практических знаний и навыков в перехватеи анализе трафика сегмента компьютерной сети.

Теоретическая часть. Снифферы (дословный перевод - ‘вынюхиватели’) являются специализированным ПО, предназначенным для анализа потока сообщений (трафика) компьютерной сети передачи информации [4]. Известными системами подобного рода (но глобального уровня) являются ЭШЕЛОН(североамериканский проект, назначением которого является анализ содержимого линий связей Европы) и СОРМ (тотальное протоколирование трафика русскоязычной Сети). Большинство программ и сервисов (ICQ, TelNet,FTP, HTTP, POP3 и т.д.) пересылают пароль и логин пользователя открытым текстом (без всякой кодировки и шифровки), и работающий сниффер без труда позволит перехватывать такие сессии.

К простым ПО подобного класса относится, например комплект SpyNet (simik.lgg.ru/ spynet312.exe); в штатную поставку Windows’NT Server и др.входит утилита Network Monitor (устанавливается добавлением сервиса Network Monitor Tools & Agent).

Обычно сетевая карта, работающая в сегменте некоммутируемой Ethernet в принципе ‘прослушивает’ весь трафик своего сегмента; однако в нормальном (без PROMISCUOUS MODE) режиме анализируются лишь первые 48 бит заголовка пакета и, если он не соответствует собственному MAC (*Media Ac-cess Control*)– адресу карты, последняя перестает читать пакет, считающийся при этом ‘чужим’. Функциональность сниффера достигается переводом сетевой карты в режим PROMISCUOUS MODE, обеспечивающий перехват всех сообщений, циркулирующих в данном сегменте сети без относительно MAC-адресов (достигается программной установкой соответствующего битауправляющего регистра карты). В случае коммутируемого Ethernet переводкарты в PROMISCUOUS MODE не позволяет прослушивать ‘чужие’ сообщения, в этом случае используется технология ‘ARP-спуфинга’ (путем соответствующей подделки ARP-сообщений данная сетевая карта ‘притворяется’маршрутизатором с MAC-адресом, однако, данной карты), при этом трафик всех составляющих сегмент сети насильственно направится в сторону карты-обманщика.

Ниже приведен максимально упрощенный исходный текст небольшой программы - сниффера для Unix/Linux.

```
/* сниффер пишет на stdout все, что захватывает */
    #include <sys/socket.h>
     #include<netinet/in.h>
    #include <net/if.h>
    #include <unistd.h>
    #include <signal.h>
    #include <stdio.h>
    static volatile int done;
    void
    handler(int signum)
    {
      done = 1;
    } /* конец HANDLER */
    Int
    main(int argc, char **argv)
    {
            char buff[0x10000];
            struct ifreq ifr;
            int s, n;
            if (argc < 2)
            {
                    fprintf(stderr, "Usage: %s <interface>\n", argv[0]);
                    return 1;
            }
            s = socket(PF_INET, SOCK_PACKET, htons(0x0003));
            if (s == -1)
            {
                    perror("socket");
                    return 1;
            }
            strcpy(ifr.ifr_name, argv[1]);
            if (ioctl(s, SIOCGIFFLAGS, &ifr) < 0)
            {
                    perror("ioctl(SIOCGIFFLAGS)");
                    return 1;
            }
/*************************************************************************************
************/
    ifr.ifr_flags |= IFF_PROMISC; /* установка режима перехвата */
                                  /* ВСЕХ пакетов, поступающих */
                                  /* на сетевую карту */
    if (ioctl(s, SIOCSIFFLAGS, &ifr) < 0)
    {
            perror("ioctl(SIOCGIFFLAGS)");
            return 1;
```

```
} signal(SIGINT, handler)

puts("starting capturing:\n");
fflush(stdout);
for (done = 0; !done; )
{
        n = read(s, buff, sizeof(buff)); /* считываем перехваченный трафик в буфер buff */
        if ( n!=-1 )
                write(STDOUT_FILENO, buff, n);
}
ifr.ifr_flags &= ~IFF_PROMISC; /* снятие режима перехвата всех пакетов */
if (ioctl(s, SIOCSIFFLAGS, &ifr) < 0)
{
        perror("ioctl(SIOCGIFFLAGS)");
        return 1;
}
        close(s);
        printf("Finished\n");
        return 0;
} /* конец MAIN */
```

Один из примеров выдачи (определенным образом скомпонованной) информации сниффером приведен ниже; распечатка содержимого перехваченного пакета (датаграммы) состоит из разделенных двоеточием трех колонок: формат пакета-носителя, имя поля, содержимое поля в десятичном и восьмеричном представлении. Этот пакет содержит 14-байтовый заголовок EtherNet, 20-байтовый IP-заголовок, 20-байтовый TCP-заголовок, заголовок HTTP, оканчивающийся двумя подряд CRLF (0D 0A 0D 0A) и далее собственно данные прикладного уровня (WEB-трафик по протоколу HTTP версии1.1, [4]):

```
ETHER: Destination address: 0000BA5EBA11
ETHER: Source address: 00A0C9B05EBD
ETHER: Frame Length: 1514 (0x05EA
ETHER: Ethernet Type: 0x0800 (IP)
        IP: Version = 4 (0x4)
        IP: Header Length = 20 (0x14)
        IP: Service Type = 0 (0x0)
        IP: Precedence = Routine
        IP:...0.... = Normal Delay
        IP:....0... = Normal Throughput
        IP:.....0.. = Normal Reliability
        IP: Total Length = 1500 (0x5DC)
        IP: Identification = 7652 (0x1DE4)
        IP: Flags Summary = 2 (0x2)
        IP:.......0 = Last fragment in datagram
        IP:......1. = Cannot fragment datagram
```

IP: Fragment Offset = 0 (0x0) bytes
IP: Time to Live = 127 (0x7F)
IP: Protocol = TCP — Transmission Control
IP: Checksum = 0xC26D
IP: Source Address = 10.0.0.2
IP: Destination Address = 10.0.1.201
TCP: Source Port = Hypertext Transfer Protocol
TCP: Destination Port = 0x0775
TCP: Sequence Number = 97517760 (0x5D000C0)
TCP: Acknowledgement Number = 78544373 (0x4AE7DF5)
TCP: Data Offset = 20 (0x14)
TCP: Reserved = 0 (0x0000)
TCP: Flags = 0x10:.A....
TCP:..0..... = No urgent data
TCP:...1.... = Acknowledgement field significant
TCP:....0... = No Push function
TCP:.....0.. = No Reset
TCP:......0. = No Synchronize
TCP:.......0 = No Fin
TCP: Window = 28793 (0x7079)
TCP: Checksum = 0x8F27
TCP: Urgent Pointer = 0 (0x0)
HTTP: Response (to client using port 1909)
HTTP: Protocol Version = HTTP/1.1
HTTP: Status Code = OK
HTTP: Reason = OK

...и так далее...

Сниффер может быть установлен на маршрутизаторе (шлюзе, при этом он перехватывает проходящий через интерфейсы этого шлюза трафик) и на конечном узле сети (перехватывается трафик данного сегмента сети).

Обычно сниффер можно настроить на ‘прослушку’ пакетов по заранее определенным протоколам, портам, диапазону IP, направлению передачи и др.; допустимо указывать ‘горячие’ сочетания символов (слова и словосочетания, наличие которых является признаком подготовки определенного действия – например, атаки на конкретный хост с целью его разрушения или взятия под контроль, договоренности о теракте и т.п.).

Области применения снифферов можно разделить на легальные - отладка ПО сетевого класса, обучение, оптимизация сети (обнаружение проблем и ‘узких мест’), выявление несанкционированных атак на сервера Сети и нелегальные - перехват важной информации (в первую очередь паролей и login’ов пользователей). Стандартным нелегальным приемом использования сниффера является запуск его на целевом сервере (после удачного root-входа) в скрытом

режиме (например, маскируясь под named-канал) и несанкционированном анализе перехваченной информации. Для предотвращения подобных сценариев используется средства, осуществляющие шифровку трафика (например, протокол SSH).

Анализ вышеприведенной перехваченной информации в реальном случае (анализ многих тысяч пакетов) чрезвычайно затруднен, поэтому разработаны специальные системы для фильтрации и компоновки перехваченного трафика (например, визуализация имеющегося WEB-трафика броузером).

Одной из несложных систем подобного рода является комплект SpyNet, включающий два модуля - CaptureNet (собственно перехват трафика, рис.1) и PeepNet (фильтрация пакетов и визуальное представление содержимого, рис.3). CaptureNet определяет MAC- и IP-адреса сетевой карты и по нажатию Start capture переводит ее в выбранный режим (задается набором флажков Hardware Filter); при этом в верхнем правом фрейме окна выводятся параметры каждого захваченного пакета (показываются время, MAC- и IP-источника и адресата, используемый протокол, порт и др.), в правом нижнем фрейме можно просмотреть содержимое выбранного пакета (восьмеричное), при ра- боте CaptureNet записывает отфильтрованные пакеты в файл (по умолчанию CAPTURE.CAP). Заметим, что фактический сброс информации в этот файл происходит в момент превышения объема буфера (задается выбором вариан- тов меню Capture|Settings) или при остановке захвата пакетов нажатием Stop capture).

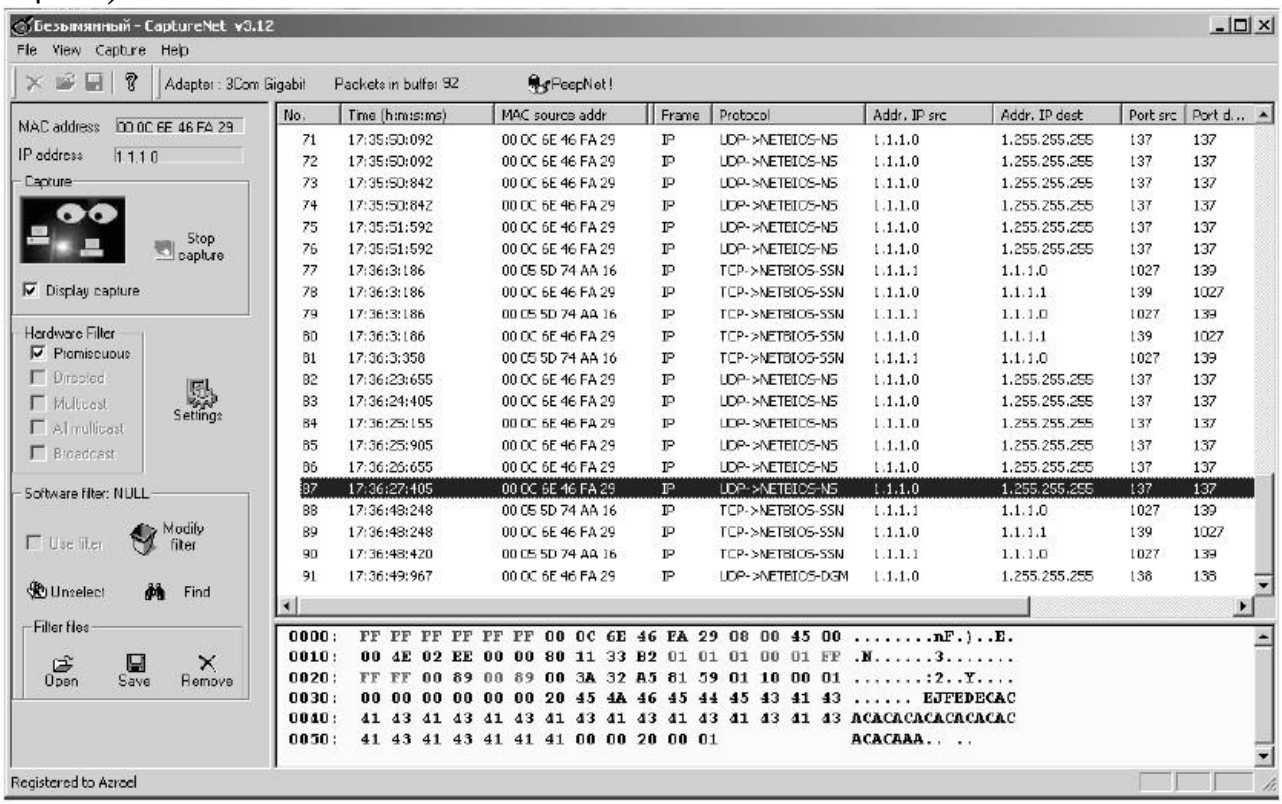

Рис. 1. Главное окно утилиты CaptureNet v3.12

В окне Software filter (рис.2, инициализируется кнопкой Modify filter главного окна) задаются параметры фильтрации пакетов: на вкладке Layer 2,3 протокол (например, для перехвата IP/TCP-пакетов целесообразно задать Frame IP, Layer 3+ TCP), на вкладке Pattern matching – ключевые слова для поиска в па- кетах, на вкладке IP addreses – диапазон сканируемых адресов и направление трафика, на вкладке Ports - номера контролируемых портов.

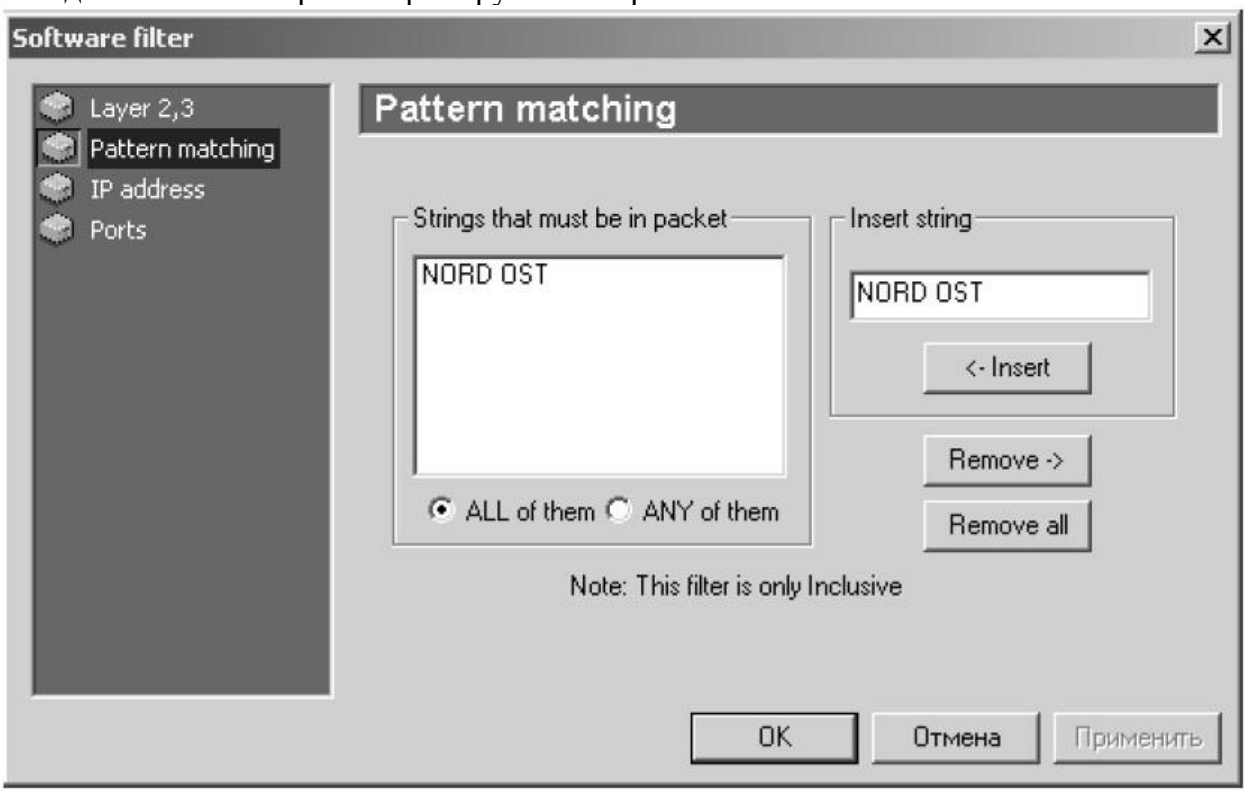

Рис. 2. Окно Software Filter ут**и**литы CaptureNet v3.12

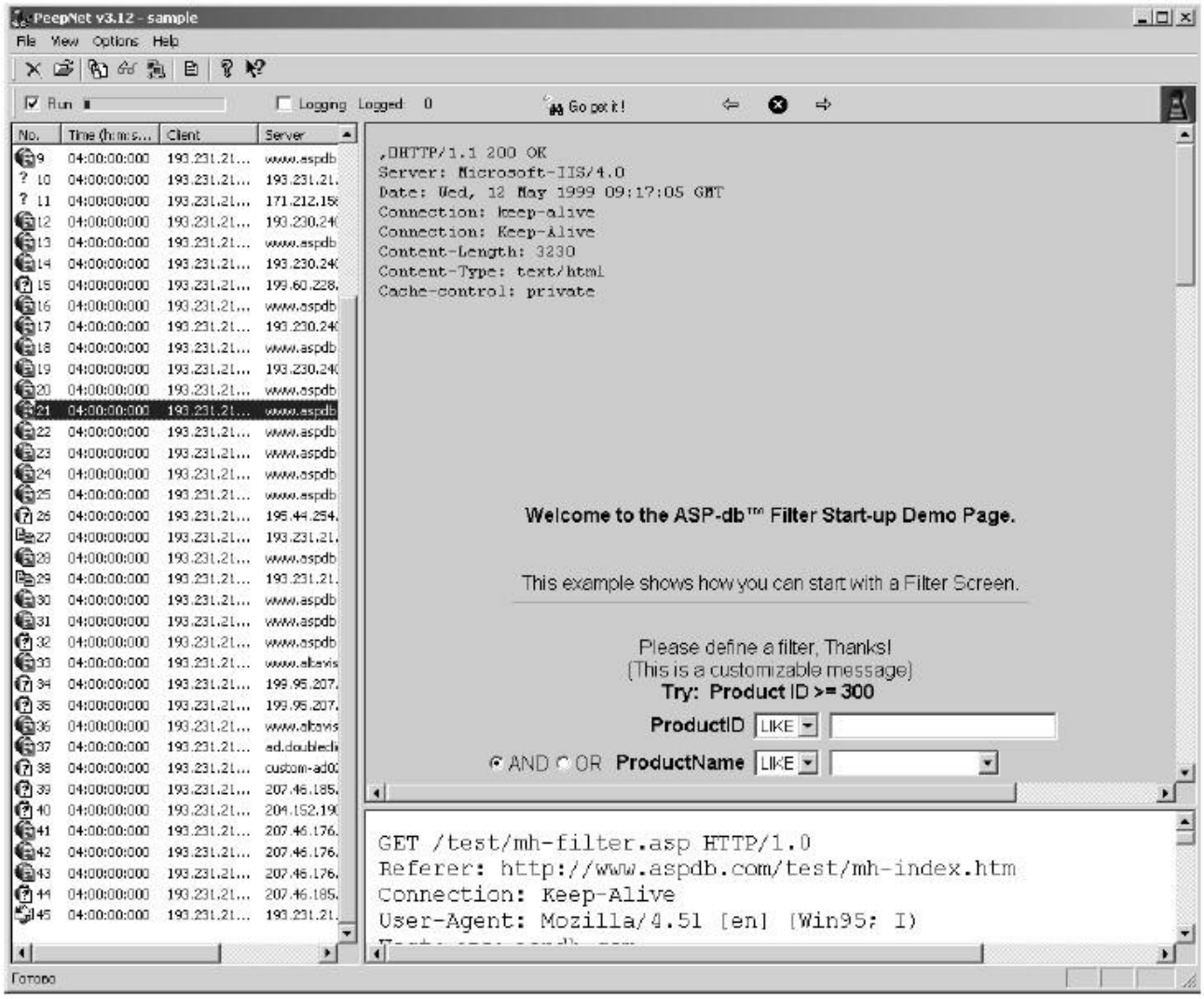

Рис. 3. Окно визуализации перехваченного сообщения утилитой PeepNet v3.12

Утилита PeepNet может быть вызвана из CaptureNet или отдельно, при этом может быть проанализирован любой из захваченных CAP-файлов (File|Open), при этом также возможно установить фильтрацию пакетов Options|Settings). В правом верхнем фрейме выдается графическое представление информационной части пакета в стиле WEB-страницы, в правом нижнем – текст соответствующего запроса клиентской части (рис.3).

Необходимое оборудование – IBM PC-совместимая ЭВМ, предустановлен- ная ОС Windows, пакеты CaptureNet и PeepNet версии 3.12.

Порядок проведения работы. Студент получает задание на анализ трафика локальной сети с заданным условием фильтрации пакетов. Условиями пере- хвата могут быть, например, только TCP/IP-пакеты, диапазон просматриваемых IP-адресов, просмотр направленных к конкретному порту пакетов (HTTP – обычно порт 80, FTP – 21, SMTP – 25, SSL – 443, ICQ – 4000 и др.), заданные сочетания символов (USER, PASS, LOG и др.); при этом в сети должны действительно

циркулировать указанные пакеты (пакеты ARP- и EtherNet 802.3 обычно присутствуют всегда, осуществляя поддержку сети).

Отфильтрованные пакеты накапливаются в CAP-файле и в дальнейшем анализируются утилитой PeepNet, корректность перехвата и анализа проверяется преподавателем.

Оформление отчета по работе. В отчете указываются заданные преподавателем условия фильтрации пакетов и информация о содержании перехваченных датаграмм. Студент должен сделать выводы о направленности (адресации) пакетов, имеющих широковещательные адреса. *Вопросы для самопроверки*.

1. Что представляет из себя ПО класса снифферов и с какой целью применяется?
2. Каковы ограничения методов перехвата информации снифферами?
3. Каким образом сетевая плата конкретной ПЭВМ в локальной сети распо- знает назначение пакетов по принципу ‘свой-чужой’?
4. Какие методы применяют с целью исключения возможности перехвата со- общений снифферами?