сырые сокеты на Висле, XP, W2K…

крис касперски ака мыщъх, no-email

сырые сокеты (raw sockets) широко используются как в хакерских, так и в легальных коммерческих программах: exploit'ах, спуферах, сниферах, сканерах, брандмауэрах, NAT'ах, etc. никсы поддерживают сырые сокеты изначально, 9x – лишь формально. с выходом W2K некрософт подарила нам полноценную поддержку сырых сокетов, но, начиная с XP SP2, сурово урезала их функциональность, в результате чего многие системные программы перестали работать и, чтобы вернуть функциональность взад программистам пришлось опуститься на уровень ядра или заюзать библиотеку WinCap. а простым пользователям что делать? как "оживить" старые программы, не имея исходных текстов на руках? без паники! мыщъх сейчас все расскажет, только покурит слегонца.

введение

Сокеты представляют собой индустриальный стандарт унифицированного интерфейса, ориентированный на межпроцессорное взаимодействие, и поддерживаемый практически всеми операционными системами. Причем, с точки зрения сокета, ему все равно (ну, практически, все равно) где находится соседний процесс — на локальной машине или на другом конце света (тьмы). В общем случае, сокет представляет собой комбинацию IP-адреса и порта (например: 192.168.6.9:25), а так же набор функций для установки соединения и обмена данными.

Обычные сокеты — явные приверженцы парадигмы ООП и позволяют воздействовать с заголовками пакетов только путем вызова соответствующих API-функций. Мы можем задавать целевой адрес и порт назначения (а при желании — и локальный порт), устанавливать некоторые флаги типа TTL (Time To Live – "время" жизни) или TOS (Type of Service – тип сервиса). Остальные же поля (например, поле контрольной суммы или флаг фрагментации) операционная система заполняет самостоятельно. Программист лишен возможности вмешиваться в этот "тонкий" процесс. Впрочем, большинству прикладных протоколов (POP3/SMTP/HTTP) обозначенных возможностей вполне достаточно и программистам жаловаться не приходится. А хакерам?!

Сырые сокеты выгодно отличаются тем, что позволяют собирать TCP/IP пакеты вручную, контролируя каждый бит заголовка, отправляя в сеть нестандартные пакеты, к приему которых операционная система ни морально, ни физически не готова. Хакер может завесить целевой компьютер, забросить зловредный shell-код, обойти брандмауэр, незаметно просканировать порты, отправить пакет от чужого имени, да много чего можно придумать!

Во времена рассвета 9x, поддерживающей сырые сокеты лишь на уровне ICMP, хакеры во всю ставили Linux/BSD только для затем, чтобы получить полноценный доступ к сырым сокетам. Linux/BSD превращал хакеров в богов, контролирующих обширные сетевые территории и скрывающихся за поддельными IP-адресами. Типа хрен меня найдешь! И хотя основные ошибки в TCP/IP стеке за последнее десятилетие были исправлены, внедрение новомодного (а, значит, ни хвоста не протестированного) IPv6 вкупе с полностью переписанным сетевым стеком в Висле, вызвало всплеск интереса к атакам старого типа.

Полноценная поддержка сырых сокетов в W2K вызвала настоящий фурор! Программисты перенести многие хакерские программы (типа nmap) в Windows и необходимость ставить никсы отпала. По сети прокатилась волна атак. Сырые сокеты использовали не только хакеры, но и черви (например, червь Stumbler), в результате чего в XP функциональность сырых сокетов была существенно урезана. В XP SP2 наступил сплошной ахтунг, а XP SP2 с заплаткой MS05-019 — это уже не ахтунг, а просто ктулху какой-то с кучей блокировок на уровне ядра, которые с прикладного уровня просто так не обойдешь…

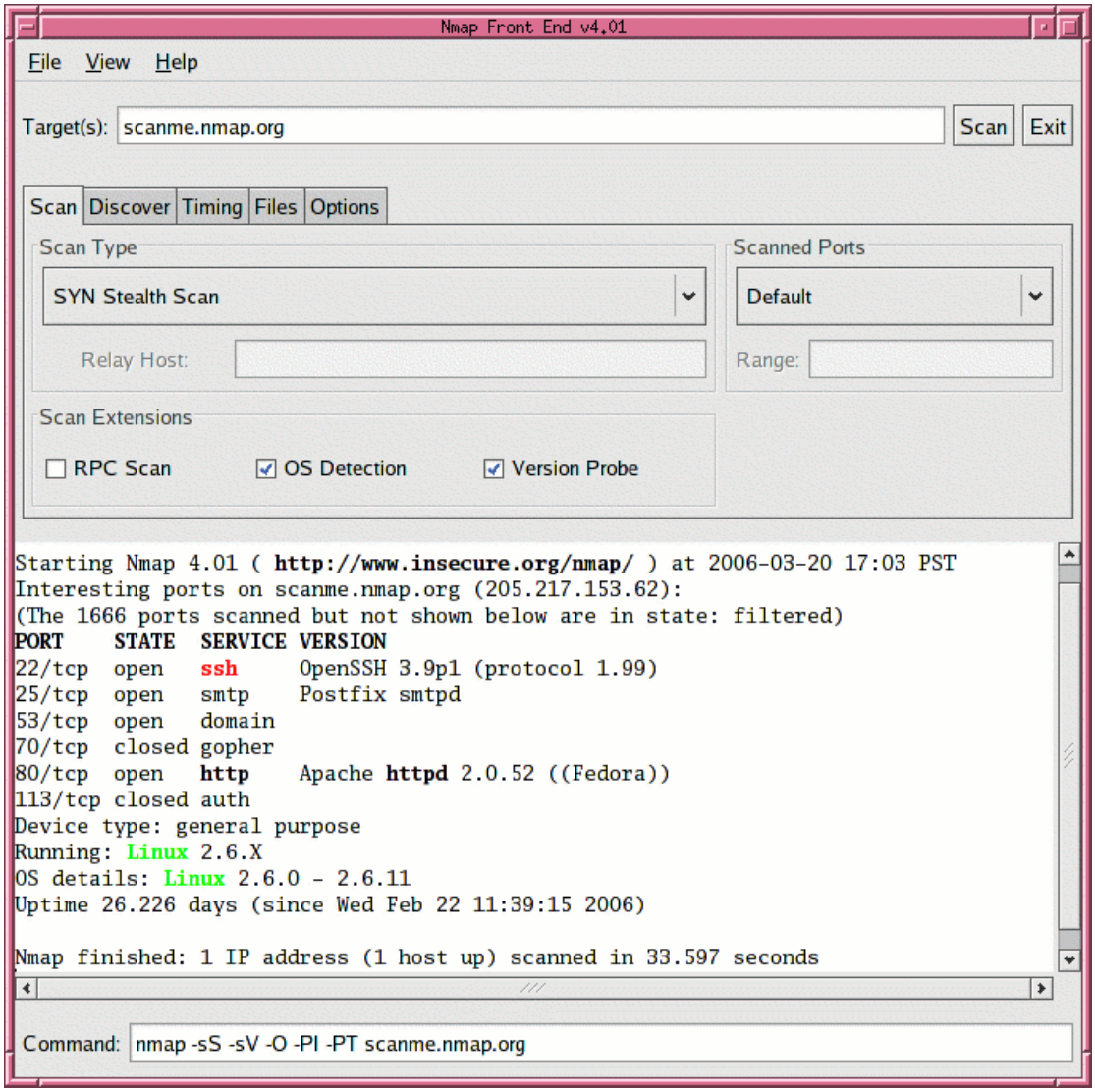

Рисунок 1 nmap – одна из многих программ, нуждающихся в сырых сокетах

Как следствие — программы, нуждающиеся в сырых сокетах, с переходом на XP SP2 перестали работать и их разработчикам пришлось искать обходные пути для возвращения утраченной функциональности (например, патчить сетевые драйвера или работать напрямую с NDIS). К сожалению, далеко не все разработчики удосужились обновить свои программы, особенно, если они распространялись в узком кругу на бесплатной основе.

Отсутствие поддержки сырых сокетов _никак_ не увеличивает защищенность Windows, хоть и не позволяет использовать ее в качестве плацдарма для атак на другие системы (точнее, затрудняет атаку в несколько раз). Но ведь на дворе не 95 год поди! Воздвигнуть Linux/BSD на виртуальной машине сегодня может даже начинающий хакер, да и способы обхода ограничений сырых сокетов тоже имеются. В ассортименте.

Настоящая статья главным образом ориентирована на пользователей, работающих с чужими программами и не имеющих возможности (времени, желания) дорабатывать их исходный код. Как заставить старые программы работать на новых системах?! Вот в чем вопрос! Мы рассмотрим все имеющиеся на данный момент оси (вплоть до Вислы и Server 2008 включительно), демонстрируя различные пути обхода наложенных ограничений.

Рисунок 2 мыщъх хипует

>>> врезка сырые сокеты в разных осях

ось	поддержка сырых сокетов
95, 98, 98SE	не поддерживает сырых сокетов (ограниченно поддерживает сырые сокеты для ICMP-протокола);
NT 4.0	сырые сокеты ограничены 10 входящими соединениями за 10 минут (лечится правкой реестра);
W2K	полностью поддерживает сырые сокеты;
XP SP0	полностью поддерживает сырые сокеты после остановки брандмауэра ("net stop sharedaccess");
XP SP1	_без_ заплатки MS05-019 поддерживает сырые сокеты так же, как и XP SP0;
XP SP1	_с_ заплаткой MS05-019 блокирует исходящие сырые TCP-сокеты, если брандмауэр _запущен_
XP SP2	_без_ MS05-019 полностью поддерживает входящие сырые сокеты и частично исходящие;
XP SP2	_с_ заплаткой MS05-019 полностью поддерживает входящие сырые сокеты и некоторые исходящие;
Висла[1]	не поддерживает сырых сокетов (ограниченно поддерживает сырые сокеты для ICMP-протокола);
Server 2003	полностью поддерживает сырые сокеты после остановки брандмауэра ("net stop alg");
Server 2008[2]	полностью поддерживает сырые сокеты после остановки брандмауэра ("net stop alg");
Mac OS X	полностью поддерживает сырые сокеты;
Linux	полностью поддерживает сырые сокеты;
xBSD	полностью поддерживает сырые сокеты;

Таблица 1 поддержка сырых сокетов в разных осях

>>> врезка вокруг MS05-019

Заплатка MS05-019 представляет собой обновление безопасности, затыкающее критическую дыру в TCP/IP стеке путем замещения драйверов afd.sys, tcpip.sys и tdpi.sys, а так же некоторых динамических библиотек (подробнее об этом можно прочитать на http://www.microsoft.com/technet/security/bulletin/ms05-019.mspx).

Отказ от использования заплатки MS05-019 теоретически возможен, но практически — крайне нежелателен, поскольку, "поймав" определенным образом сконструированный IP-пакет, атакованный компьютер начнет исполнять зловредный shell-код на уровне ядра или (что более вероятно) уйдет в глухой завис полной нирваны голубого экрана.

Впрочем, существует возможность заблокировать незапрошенный IP-трафик на брандмауэре (Windows Firewall умеет делать это), однако, тогда перестанут работать и многие легальные программы. С другой стороны, установка MS05-019 проходит весьма болезненно,

[1] поведение системы может быть изменено в любую минуту

[2] поведение системы может быть изменено в любую минуту

порождая огромное количество проблем, неполный список которых лежать на http://support.microsoft.com/kb/897656 и потому хакерам настоятельно рекомендуется снести ее напрочь (как серпом по яйцам), а трафик рулить брандмауэром. Например, Outpost'ом для которого можно написать специальный plug-in, распознающий зловредные IP-пакеты и дропающие их. Наградой за это станет "реабилитация" сырых сокетов.

Что же касается мыщъх'а, то он сидит под W2K и вообще не имеет никаких проблем с сырыми сокетами.

>>> врезка от юзера до админа

По умолчанию, сырые сокеты доступны только из-под учетной записи администратора, что не есть труъ, однако, любые ограничения можно обойти и чтобы сырые сокеты заработали и на пользовательском уровне, достаточно открыть следующую ветвь системного реестра: HKLM\System\CurrentControlSet\Services\Afd\Parameters, найти там параметр DisableRawSecurity типа DWORD (если такого параметра нет — создать его), присвоить ему значение "1", после чего перезагрузиться и все! С этой минуты сырые сокеты доступны всем!

в хлюпающей грязи зловонных болот M$

Сырые сокеты делятся на две категории: первые знают номер протокола, с которым они работают (например, ICMP), вторые же — принимают пакеты всех протоколов, независимо от номера, прописанного в их заголовке. Достаточно часто встречается утверждение, что *"при попытке открыть сырой сокет вызовом API-функций socket/WSAsocket со вторым параметром SOCK_RAW (type) и с нулевым третьим параметром (protocol) такой сокет тебе откроется, не вопрос. Однако, при попытке сделать sendto или recvfrom, ядро скажет тебе, что ты мудак. Ну конечно же, оно сделает это более политкорректно: мол, твой системный вызов прерван"* и что правильный вариант выглядит так: "`socket(AF_INET, SOCK_RAW, IPPROTO_IP)`".

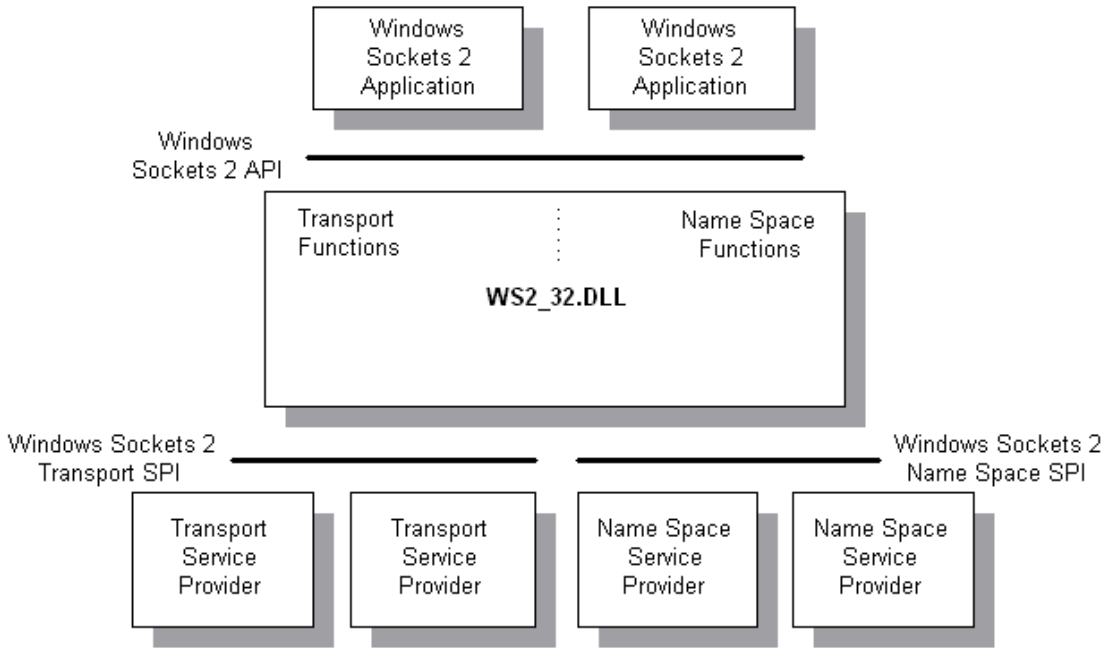

Рисунок 3 архитектура сокетов в W2K и более старших системах

Базару нет, вариант действительно правильный, однако, поскольку макрос IPPROTO_IP равен 0, то "`socket( AF_INET, SOCK_RAW, _0_ )`" будет работать ничуть не хуже. Кстати, AF_INET можно смело заменить на PF_INET — суть дела от этого не меняется. А вот с остальными типами протоколов надо разобраться. Параметр IPPROTO_IP тождественен IPPROTO_RAW — в обоих случаях мы принимаем _все_ IP-пакет _целиком_ вместе с IP-заголовками независимо от того была ли установлена опция или IP_HDRINCL или нет.

Параметр IPPROTO_ICMP распространяется только на ICMP-сообщения, однако, если к сокету была применена операция SIO_RCVALL, протокол нивелируется и ловит все пакеты

без разбора, обеспечивая тот же самый эффект, что и IPPROTO_IP/IPPROTO_RAW (по крайней мере в текущих версиях Windows дело обстоит именно так). Параметр IPPROTO_UDP в этом случае не ловит никаких пактов вообще, но конкретно срывает крышу персональным брандмауэрам, многие из которых отображают большое количество блокируемого UDP-трафика, но на самом деле не блокируют его, а спокойно доставляют пакеты до целевого приложения. То есть, все работает нормально, только брандмауэр ругается. Неплохая шутка для администраторов. Хи-хи!

Параметр IPPROTO_TCP вызывает ошибку при вызове функции bind, так что с TCP выходит полный облом, но если кто здесь и мудак, так это ядро, но не никак мыщъх.

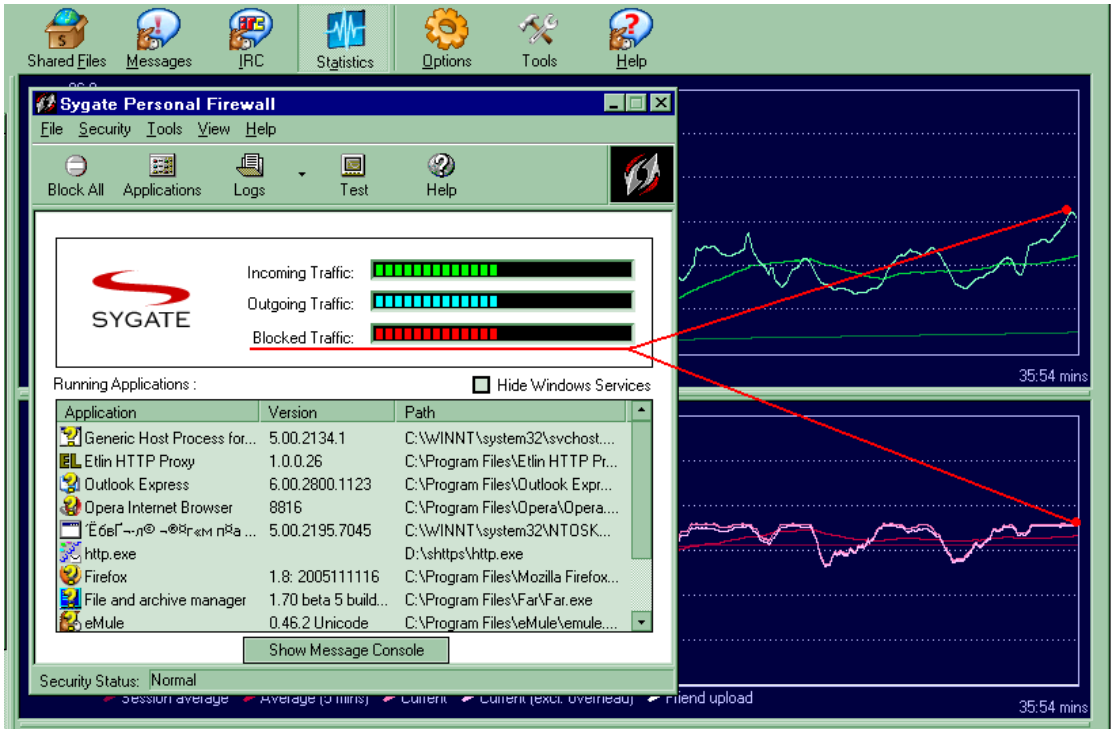

Рисунок 4 Sygate Personal Firewall показывает, что весь UDP трафик заблокирован, хотя Осел шурует нормально

Если для сырого сокета задан локальный IP-адрес, он должен соответствовать целевому IP-адресу входящего пакета, прописанного в IP-заголовке (задать локальный IP-адрес можно API-функцией bind). Если же сырой сокет не сопоставлен ни с каким локальным адресом, пакет _копируется_ в сокет, независимо от того, кому он адресован. "Копируется" в том смысле, что сырой сокет не "пожирает" проходящие через интерфейс пакеты, а снимает с них копии, как бы превращаясь в пассивный сниффер, не нарушающий работу остальных приложений. "Ретранслировать" пакеты не нужно — они и сами дойдут до приложения-получателя.

При этом нужно быть готовым к приему большого количества "левых" пакетов, совершенно нам не предназначенных, что особенно актуально для локальных сетей или DSL-подключения с криво настроенным провайдерским маршрутизатором. Забавно, но при этом часто удается "выудить" довольно интересную информацию, например, незашифрованные пароли к некоторым сайтам или почтовым ящикам, причем, обнаружить факт сниффенга подобного рода совершенно невозможно (мы просто собираем все пакеты, физически проходящие через нашу машину, что же в этом незаконного?!)

Если для сырого сокета задан foreign-адреес (т. е. удаленный IP-адрес), он будет "ловить" только IP-пакеты, приходящие с заданного узла, т. е. пакеты, чей source-адрес равен установленному (задать foreign-адреесс можно с помощью API-функции connect). Особого смысла в этом нет, ну разве что мы хотим ограничить сбор трафика каким-то конкретным узлом. Если же foreign-адреес не установлен, в сырой сокет копируются все пакеты, независимо от адреса-источника.

Как вариант, грабеж проходящего мимо трафика можно осуществить путем установки IOCTL-параметра SIO_RCVALL путем вызова функции WSAIoctl, главным недостатком которой является невозможность работать с протоколами, отличными от IPPROTO_IP и

необходимость привязки сокета на конкретный интерфейс. То есть, если у нас обозначена сетевая карта локальной сети, DSL- и GPRS-модем, в таком случае API-функцию bind придется вызывать трижды, каждый раз делая это в отдельном потоке (на блокируемых сокетах). Попытка привязки к любому адресу (INADDR_ANY) ведет к провалу. К тому же флаг SIO_RCVALL поддерживается только начиная с W2K и в NT не работает. В общем, решайте сами — иметь или не иметь. Между тем, существуют и другие IOCTL-команды полезные для грабежа — SIO_RCVALL_MCAST получает весь multicast IP-трафик (при этом тип протокола должен быть установлен в IPPROTO_UDP), а SIO_RCVALL_IGMPMCAST, соответственно, грабит весь IGMP multicast IP-трафик (при этом тип протокола должен быть установлен в PPROTO_IGMP).

Ниже приведен фрагмент простейшего IP-снифера (полный исходный текст содержится на DVD, прилагаемому к журналу):

```
// создаем сырой сокет
//--------------------------------------------------------------------------
if ((raw_socket =  socket(AF_INET,SOCK_RAW,IPPROTO_IP)) == INVALID_SOCKET)
        return printf("-ERR:socket(,SOCK_RAW,) -> %d\n", WSAGetLastError());

// (не)устанавиваем атрибут IP_HDRINCL (ибо он на фиг не нужен)
//--------------------------------------------------------------------------
//if (setsockopt(raw_socket,IPPROTO_IP,IP_HDRINCL,(char*)&rcv_all_enabled,
// sizeof(optval)) == SOCKET_ERROR)  return printf("-ERR:setsockopt(,,IP_"\
// "HDRINCL) -> %d\n", WSAGetLastError());

// перечисляем все интерфейсы
//--------------------------------------------------------------------------
if (WSAIoctl(raw_socket,SIO_ADDRESS_LIST_QUERY,0,0,addrlist,sizeof(addrlist),
&N,0,0)==SOCKET_ERROR) return printf("-ERR:WSAIoctl(SIO_ADDRESS_LIST_QUERY)"\
"error %d\n", WSAGetLastError());

// найден хоть один интерфейс?
if (!(n_addr=llist->iAddressCount)) return printf("-ERR:list is empty\n");

// распечатываем список имеющихся интерфейсов
//--------------------------------------------------------------------------
for(a = 0; a < n_addr; a++)
        printf("IP - %s\n", inet_ntoa(((struct sockaddr_in*)
                        llist->Address[a].lpSockaddr)->sin_addr));

// биндим последний интерфейс (последний - для простоты понимания)
addr.sin_family = AF_INET;
addr.sin_addr   = ((struct sockaddr_in*)llist->Address[a-1].lpSockaddr)->sin_addr;

if (bind(raw_socket,(struct sockaddr*)&addr,sizeof(addr)) == SOCKET_ERROR)
        return printf("-ERR:bind\n");

#define SIO_RCVALL  0x98000001

// сообщаем системе, что мы хотим получать все IP-пакеты, проходящие мимо
//--------------------------------------------------------------------------
if (WSAIoctl(raw_socket,SIO_RCVALL,&rcv_all_enabled,sizeof(rcv_all_enabled),
0,0,&N,0,0)==SOCKET_ERROR) return printf("-ERR:WSAIoctl(SIO_RCVALL)-->%d\n",
WSAGetLastError());

// получаем все пакеты, приходящие на данный интерфейс
// и распечатываем их содержимое
while(1)
{
        if ((len = recv(raw_socket, buf, sizeof(buf), 0)) < 1)
                return printf("-ERR:recv()->%d,WSAErr=%i\n",len,WSAGetLastError());

        // печатаем содержимое пакета
        …
}
```

Листинг 1 фрагмент простейшего IP-снифера, основанного на SIO_RCVALL

Но это все, что касается сырых сокетов, _принимающих_ пакеты. В полной мере их функциональность впервые была реализована в W2K и с тех пор не претерпела никаких существенных изменений. Судьба сырых сокетов, передающих пакеты, намного более печальна. NT не позволяет создавать больше 10 TCP-соединений за 10 минут (ахнунг сплошной!), правда, при желании это значение можно увеличить, покопавшись в реестре.

В W2K сырые сокеты поддерживаются без каких бы то ни было ограничений, однако, отправляя IP-пакет с чужим IP, мы рискуем нарваться на кучу неприятностей. "Неправильный" пакет может зарезать как наш собственный персональный брандмауэр, так и NAT, встроенный в DSL-модем. Даже если пакет благополучно покинет хакерский компьютер, его наверняка прибьет первый же маршутизатор провайдера. Так что мало научиться создавать сырые сокеты с поддельными IP-адресами, хакеру еще необходимо найти демократично настроенного провайдера (вот потому, у одних поддельные IP работают, а у других нет). В локальной сети (во всяком случае, в переделах одного сегмента) подобных проблем не возникает и все работает на ура. (Впрочем, подделка IP – не главная функция сырых сокетов и обычно хакеры используют их для создания "битых" TCP/IP пакетов с "диким" набором флагов, который целевая операционная система не переваривает и уходит в завис, нирвану или передает управление на shell-код).

В XP SP0 функциональность сырых сокетов в плане отправки данных существенно ограничена и они жестоко фильтруются персональным брандмауэром ака Windows Firewall, который, впрочем, легко остановить командой "net stop sharedaccess", после чего все проблемы исчезают. XP SP1 (с установленной заплаткой безопасности MS05-019) блокирует сырые сокеты, если брандмауэр _не_ запущен! Политика запретов продолжила набирать обороты и в XP SP2, из которой исчезли сырые TCP-сокеты и хакеры оказались вынуждены вручную собирать TCP-пакеты из IP или использовать протоколы ICMP и/или UDP. В Висле от всего этого богатства остался всего лишь один ICMP, что вплотную приблизило ее к 9x (правда, ходят слухи, что поддержку сырых IP и UDP Висле очень скоро вернут).

Короче, в XP SP2 мы имеем следующий перечень ограничений (подробнее см, http://technet.microsoft.com/en-us/library/bb457156.aspx):

- TCP-пакеты не могут быть посланы через сырые сокеты;
- UDP-пакеты с "левым" адресом источника дропаются системой;

Server 2003 и Server 2008 полностью поддерживают сырые сокеты, но… только после остановки встроенного брандмауэра, что осуществляется командой "net stop alg", причем, касательно Server 2008, информация пока неполная, противоречивая и может измениться в любую минуту. В общем, держите лапы на пульсе, в смысле на клавиатуре.

Таким образом, для хакинга идеально подходит W2K или (с некоторой натяжкой) Server 2003/2008. А что делать тем, у кого установлена XP/Висла и кто слезать с нее ни за что не собирается. Даже ради хакерства и крутизны.

Рисунок 5 мыщъх пиратствует

сырые сокеты на XP SP2/Висле

Используя XP SP2/Вислу, будьте готовы к тому, что многие атакующие программы откажутся работать. Ну, с подделкой IP-адреса никаких вопросов не возникает. Достаточно завести себе интерфейс с IP-адресом, который мы хотим подделывать и система благополучно пропустит его наружу. А вот с остальными ограничениями бороться сложнее.

Разработчики коммерческих утилит (сканеров безопасности, например) матерясь и разрывая свой хвост напополам, опускаются с прикладного уровня на уровень ядра, создавая специальный драйвер-отмычку.

Штатный драйвер TCPIP.SYS создает несколько устройств: "TCP", "UDP" и… "RAWIP", причем, "RAWIP" потребовался макрушникам для обслуживания своего же собственного NAT'a, впервые появившегося в W2K. NAT есть NAT и ему позарез необходимо слать пакеты от имени внешних узлов, обеспечивая прозрачную трансляцию адресов. Грубо говоря, NAT это легальный IP-спуфер встроенный в систему и грех не упасть ему на хвост.

Хакерский драйвер должен открыть доступ к "\Device\RawIp", назначить ему атрибут IPHDR_INCL, после чего можно слать все, что угодно и от кого угодно. А чтобы псевдоустройство виднелось с прикладного уровня, достаточно вызывать API-функцию DefineDosDevice и это будет работать на любой системе, причем, поиском правильного интерфейса для отправки пакета займется непосредственно сам драйвер TCPIP.SYS и нам не придется с этим заморачиваться.

Способ универсальный, но без драйвера тут не обойтись. Коммерческие программисты даже не крякнут, но вот авторы разных exploit'ов и других бесплатных атакующих утилит просто забили на XP SP2 и перешли на никсы или вернулись обратно в W2K, что вызвало огромное недовольство "простых смертных". Им-то сносить свою любимую XP/Вислу ох как не хочется! Лучше якорь в задницу!

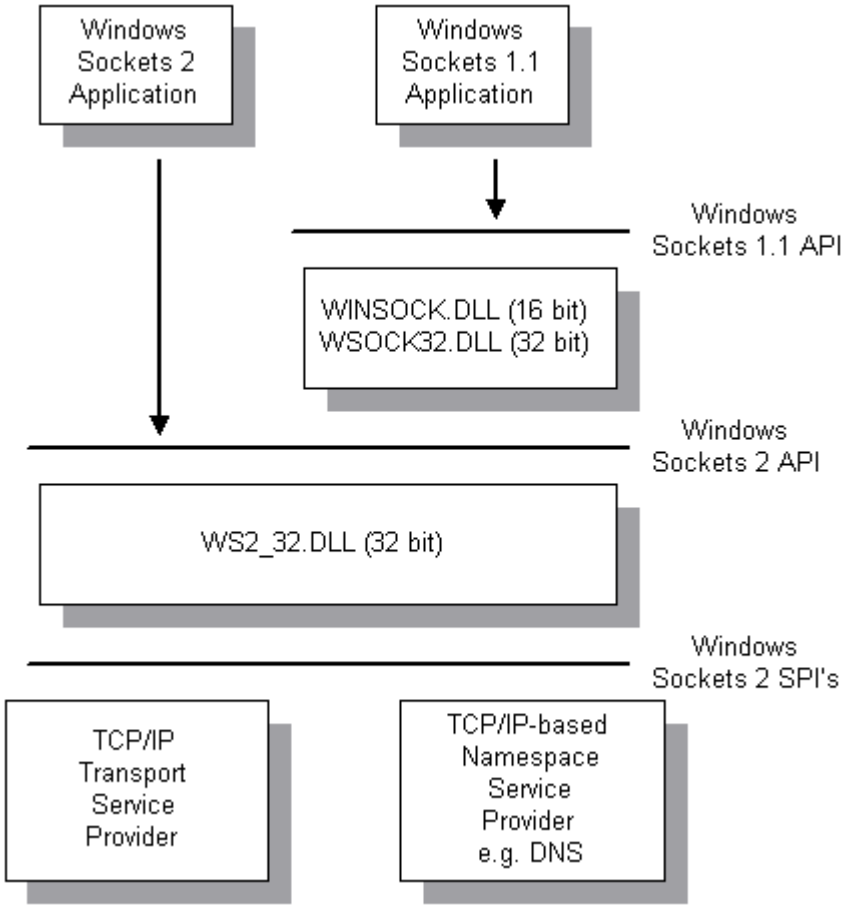

Рисунок 6 схема взаимодействия приложений с ядром через сокеты

Но что мешает установить VM Ware и натянуть поверх XP любую другую систему, например, W2K, Linux или BSD? Однако, без тонкостей и здесь не обходится… Чтобы этот "бутерброд" заработал необходимо обеспечить физический доступ виртуальной машины к сетевой карте (если используется Ethernet-интернет), USB-порту, в который воткнут DSL-модем, или COM-порту с Dial-Up модемом. Все это осуществляется легальными средствами самой VM Ware и не вызывает никаких проблем (см. рис. 7).

Заходим в свойства виртуальной машины ("Edit virtual machine settings"), находим там сетевую карту и говорим "Bridget: connect directly to physical network"), после чего гостевая операционная система подключается прямиком к… виртуальному адаптеру и, чтобы пакеты уходили с базовой машины в сеть, еще надо настроить маршрутизацию пакетов штатной командой route. Это может отпугнуть начинающих и на первых порах лучше использовать DSL-модемы на USB.

По умолчанию, VM Ware видит все USB-устройства, так что никаких проблем тут не возникает, а вот COM-модемы уже требуется сконфигурировать вручную. Возвращаемся к "Edit virtual machine settings", жмем "Add", находим в списке устройств последовательный порт ("serial port") и говорим "use physical serial port on the host", после чего модем увидится гостевой осью как родной, и нам остается только войти в Интернет… Аналогичным образом обстоят дела и с сотовыми телефонами — подключаем их либо через COM/USB-шнуток, либо через ИК/Bluetooth-порт с адаптером, воткнутым в COM/USB. (Тут правда необходимо отметить, что мне не известны сотовые операторы поддерживающие сырые сокеты, в том смысле, что пакеты все равно пересобираются на ближайшем же узле и все наши хитрые манипуляции с заголовками идут лесом).

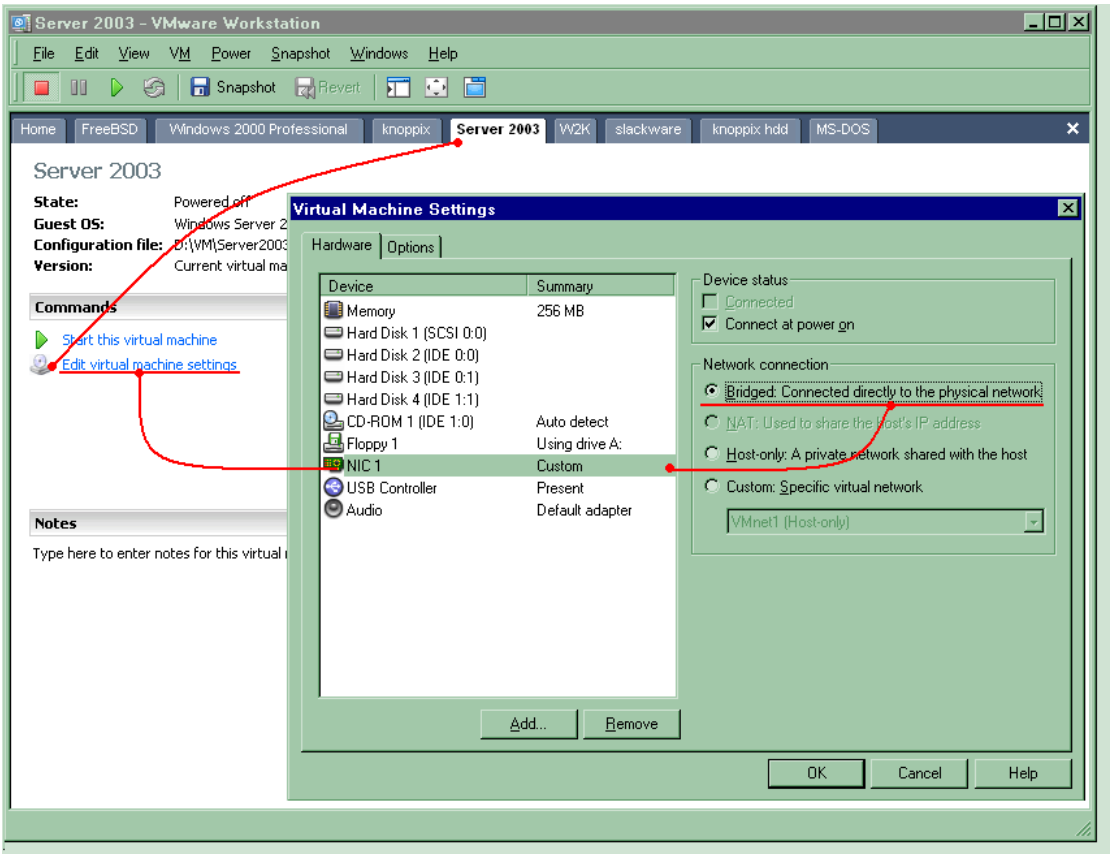

Рисунок 7 настойка VM Ware для работы с сырыми сокетами

Как вариант, можно загрузиться в Live CD (типа KNOPPIX) и юзать сырые сокеты уже через него. В общем, возможных решений — много. И какие бы препятствия нам ни городила некрософт, мы — хакеры — все равно их обойдем! Ведь у нас есть хвост!

заключение

Некрософт (и куча сетевых обозревателей, включая вроде бы неглупого му… мужика Стива Гибсона — http://www.grc.com/dos/intro.htm) расценивает сырые сокеты исключительно

как орудие зла и в стремлении защитить мир от вандалов планомерно щемит функциональность, забывая о том, что сырые сокеты — это еще и превосходное средство обучения. Только конструируя TCP/IP пакеты своим "руками" можно постичь истинное Дао, на котором держится весь Интернет и нам остается только радоваться, что сырые сокеты по прежнему с нами!

Рисунок 8 просто мыщъх

>>> врезка нажми на газ!

Сырые TCP/UDP сокеты работают _намного_ медленнее обычных (что особенно хорошо заметно при открытии большого количества соединений или интенсивном трафике), а потому применять их следует тогда и только тогда, когда стандартными средствами поставленная задача не решается.

Рисунок 9 мыщъх за работой