прячем трафик от админа

крис касперски ака мыщъх, a.k.a nezumi, a.k.a souriz, a.k.a. elraton, no-email

до сих пор рассматриваемые нами способы маскировки трафика сводились к сокрытию сетевых соединений, но на физическом уровне весь "левый" трафик элементарно обнаруживался снифферами и прочими защитными средствами, вот хакеры и напряглись, решив эту проблему путем создания секретных пассивных каналов, передающих информацию без генерации какого-либо трафика вообще. исходные тексты "движков" выложены в сеть и все, что нам нужно — это разобраться как прикрутить их к нашему кей-логгеру или удаленному шеллу, чем мы сейчас и займемся, но сначала покурим

введение

Забросить shell-код на удаленную машину и застолбить там back-door — это только полдела. Так, развлечение для малолеток. А вот что дальше делать мы подумали? Необходимо заныкать свой IP-адрес и обойти все брандмауэры, не оставляя никаких следов в логах, анализируемых как вручную, так и автоматизированными системами определения вторжения, коих в последнее время развелось просто до фига.

Существует множество утилит, прячущих "левые" сетевые соединения от глаз администраторов, однако, на физическом уровне весь "хакерский" трафик элементарно обнаруживается и пресекается практически любым брандмауэром, чего атакующему допускать ни в коем случае нельзя. Это и обломно, да и для здоровья вредно.

В идеале, необходимо пробить тоннель, открыв секретный канал связи, не создающий _никаких_ дополнительных соединений и не генерирующий _никакого_ избыточного трафика, чтобы даже самый строгий разбор дампов, награбленных tcpdump'ом, не выявил ничего подозрительного.

Над решением этой проблемы бились лучшие хакерские умы. Сначала идея получила чисто теоретическое обоснование (Andrew Hintz, Craig Rowland) с чисто лабораторной реализацией, не пригодной для практического использования. Затем к делу подключилась Жанна Рутковская, разработавшая специальный протокол с кодовым названием **NUSHU** (см. одноименную врезку) и выложила исходные тексты вполне жизнеспособных модулей, ориентированных на работу в Linux Kernel 2.4 с обещанием портировать их под другие ядра и операционные системы семейства NT и xBSD. Как говориться, обещанного три года ждут, но вот уже прошло пять лет, а воз и ныне там.

Но не будем унывать. В конечном счете, заняться переносом каждый страждущий может и сам (да и ядро 2.4 достаточно часто встречается). Главное, что Жанна вручила нам мощное оружие для управления удаленными shell'ами, против которого практически невозможно разработать адекватное противоядие.

Осталось только разобраться как им (оружием) пользоваться. А то тут некоторые жаловались — давим на все кнопки, а оно даже не включается.

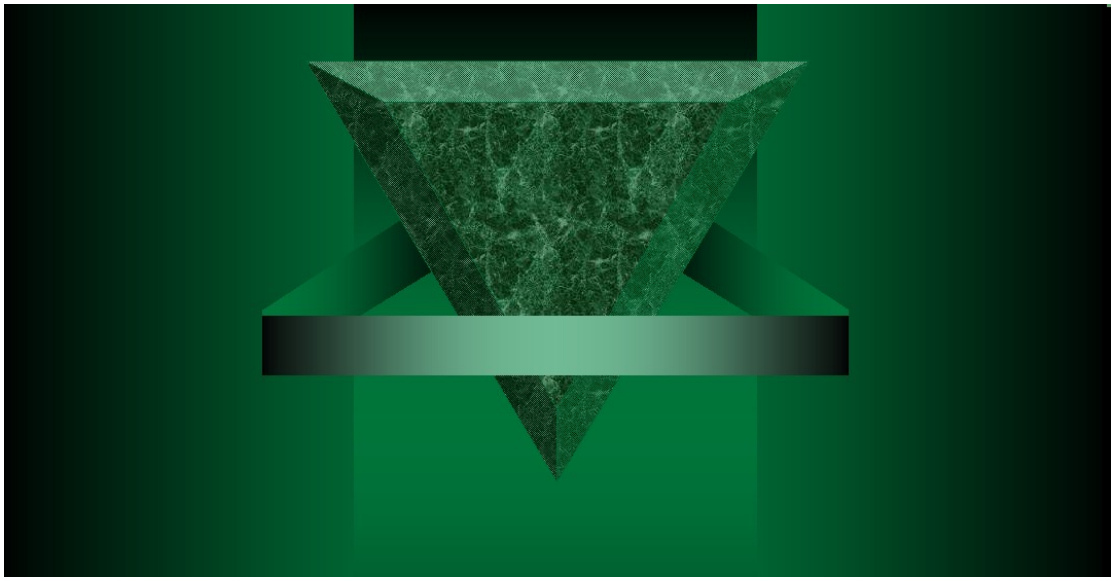

Рисунок 1 невидимая точка пересечения двух прямых

Скрытые Пассивные Каналы: основные концепции

С недавних пор в хакерском лексиконе появилось понятие ***Скрытых Пассивных Каналов*** (***P****assive* ***C****overt* ***C****hannels* — или, сокращенно, **PCC**), представляющих собой разновидность обычных Скрытых Каналов (Covert Channels — или, сокращенно CC), но в отличии от последних, не только не устанавливающие "своих" соединений и вообще не генерирующие _никакого_ собственного трафика! Передача информации осуществляется исключительно путем модификации пакетов, пролетающих мимо атакованного узла.

Соль в том, что эти пакеты направляются _не_ к хакеру, а шуруют своими путями на различные узлы Интернета, например, www.google.com, за счет чего достигается высочайшая степень анонимности. Естественно, возникает резонный вопрос, — как хакер сможет добраться до содержимого пакетов, идущих мимо него? Для этого необходимо подломать один из промежуточных маршрутизаторов (как правило, принадлежащих провайдеру, обслуживающему атакуемую организацию) и установить на них специальный модуль, анализирующий заголовки TCP/IP пакетов на предмет наличия скрытой информации или внедрения оной.

Таким образом, хакер организует двухсторонний секретный пассивный канал связи с узлом-жертвой, засекретив не только факт передачи "левой" информации, но еще и надежно замаскировав свой IP, который может определить только администратор взломанного марштутизатора, но никак не владелец узла-жертвы!

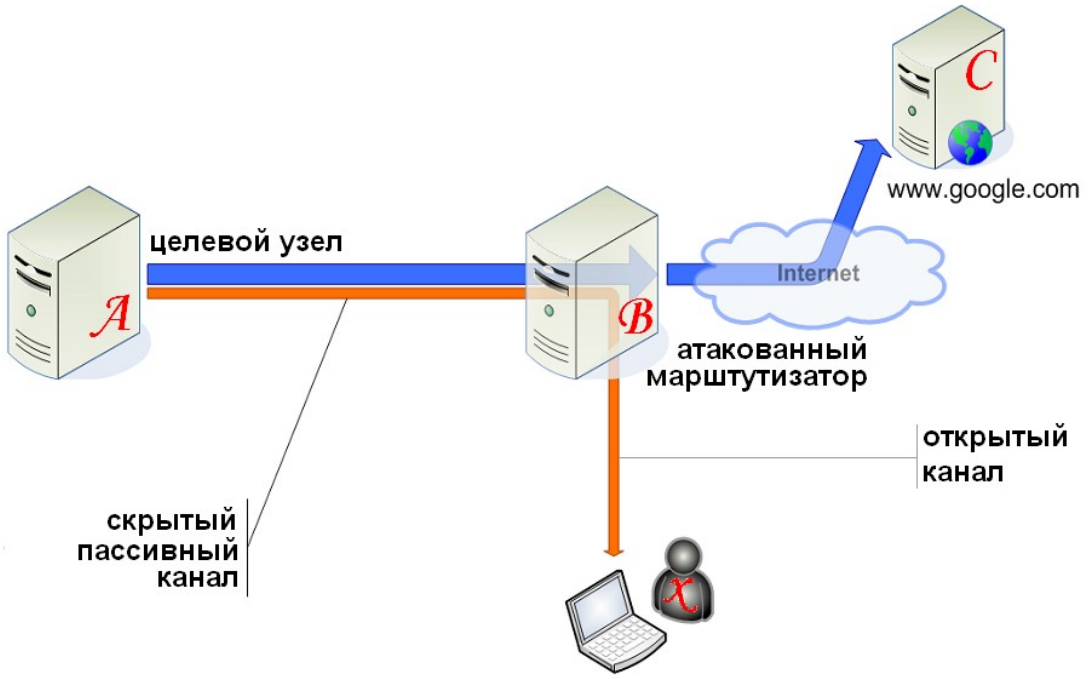

Рисунок 2 схема взаимодействия с целевым узлом (жертвой) по скрытому пассивному каналу

Рассмотрим рис. 2. Хакер (обозначенный буквой "X") каким-то совершенно не относящийся к обсуждаемой теме образом забрасывает на целевой узел (обозначенный буквой "A") shell-код, захватывающий управление и устанавливающий back-door, вместе со специальным модулем, обеспечивающим функционирование РСС-канала. Теперь все TCP/IP пакеты, отправляемые жертвой во внешний мир, содержат незначительные изменения, кодирующие, например, пароли или другую конфиденциальную информацию.

Часть этих пакетов проходит через внешний маршрутизатор "B", заблаговременно взломанный хакером, внедрившим в него РСС-модуль, анализирующий заголовки всех TCP/IP пакетов на предмет поиска "скрытого содержимого", после чего декодирует его и отправляет хакеру по открытому каналу.

Передача данных от хакера к жертве работает по аналогичной схеме. РСС-модуль, установленный на маршрутизаторе, выявляет пакеты, направленные на целевой IP-адрес, и модифицирует их заголовки в соответствии с выбранным принципом кодирования информации.

Таким образом, мы получаем защищенный канал A-B и открытый B-X, однако, хакеру ничего не стоит общаться с узлом "B" через анонимный Proxy-сервер или даже выстроить цепочку из нескольких защищенных каналов. К тому же, выбор маршрутизатора "B" – произволен. Главное, чтобы он располагался между целевым узлом "A" и одним из узлов, с которыми "сношается" жертва.

наш извозчик — протокол IP

Возьмем протокол IP и попробуем создать на его основе скрытый пассивный канал. Среди множества полезных и бесполезных полей заголовка, наше внимание привлекает 16-битовое поле "Identification" (см. рис. 3), генерируемое операционной системой случайным образом и используемое для идентификации дейтаграммы в случае ее фрагментации. Узел-получатель группирует фрагменты с одинаковыми IP-адресами источника/назначения, типом протокола и, разумеется, идентификатором.

Строгих правил, определяющих политику генерации идентификатора, в RFC нет. Одни операционные системы используют для этого таймер, другие вычисляют идентификатор на основе TCP-пакетов, чтобы при повторной передаче TCP-сегмента, IP-пакет использовал тот же самый идентификатор, однако, если даже идентификатор окажется иным ничего ужасного не произойдет. Ну, подумаешь, чуть-чуть упадет скорость.

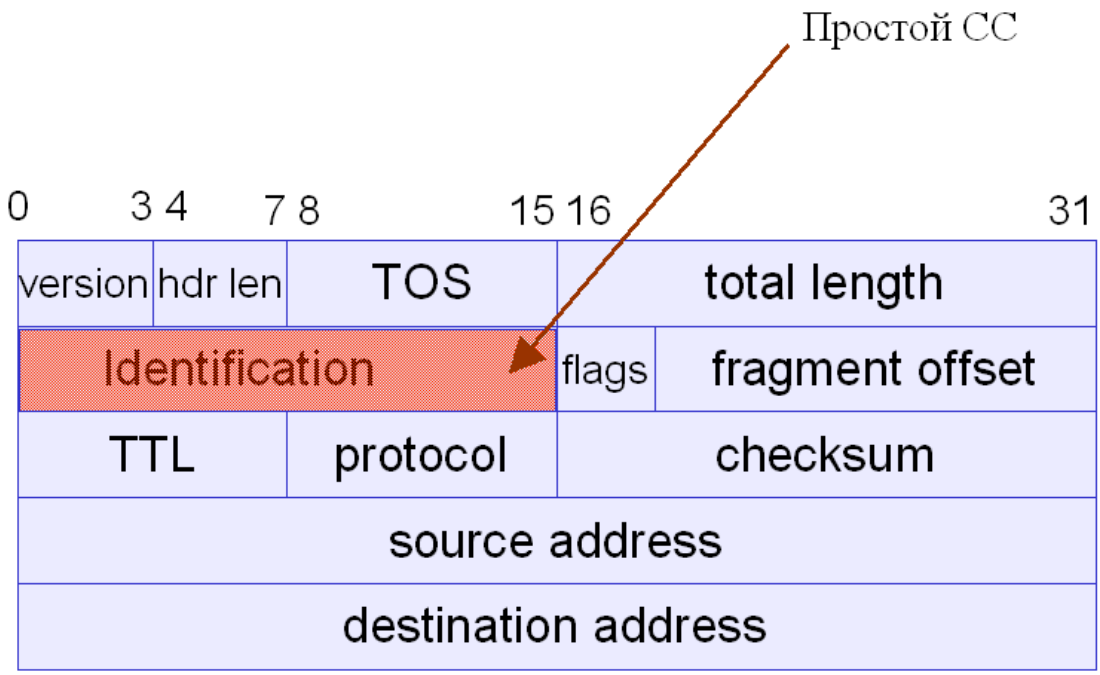

Рисунок 3 формат заголовка IP-пакета

Суть в том, что РСС-модуль может беспрепятственно модифицировать поле идентификатора по своему вкусу, передавая с _каждым_ IP-пакетов 16-бит полезных данных. Это — в теории. На практике же нам потребуется выделить несколько бит для маркировки "своих" пакетов, иначе РСС-приемник ни за что не сможет отличить их от остальных. Пусть в 12ти младших битах передаются полезные данные, а в 4х старших — их контрольная сумма. Тогда, РСС-приемнику останется всего лишь взять 12 бит, рассчитать их CRC и сравнить с оставшимися 4 битами. Если они совпадают, значит, это наш пакет, если же нет — пускай идет себе лесом.

Так же следует позаботиться о нумерации пакетов, поскольку порядок следования IP-пакетов в общем случае не совпадает с порядком их отправки. А для этого так же требуются биты, в результате чего реальная информационная емкость IP-заголовка стремится к одному байту, что, в общем-то не так уж и плохо. Для передачи небольших объемов данных (типа паролей) вполне себе сойдет. Главное — не забывать о том, что идентификатор должен быть: а) уникальным; б) выглядеть случайным. Поэтому, необходимо прибегнуть к скремблированию, то есть наложению на передаваемый текст некоторой псевдослучайной последовательности данных (известной как РСС-отправителю, так и РСС-получателю) через оператор XOR.

Кроме идентификатора можно (с некоторой осторожностью) менять поля TTL (Time To Live – максимальное время жизни пакета), типа сервиса (TOS) и протокола (protocol). Однако, это слишком заметно и легко обнаруживается просмотров дампов, полученных tcpdump или любым другим сниффером.

наш извозчик — протокол TCP

При установке TCP-соединение, передающая сторона (узел "A") устанавливает флаг SYN и выбирает произвольный 32-битный номер последовательности (Sequence Number, или, сокращенно, SEQ). Если приемная сторона (узел "B") согласна принять узел "A" в свои объятия, она отправляет ему пакет с установленным флагом ACK и номером подтверждения (Acknowledgment Number) равным SEQ+1, а так же сгенерирует свой собственный номер последовательности, выбираемый случайными образом. Узел "A", получив подтверждение, поступает аналогичным образом, что наглядно демонстрирует следующая схема:

```
узел "A" ------ SYN(ISN) -----------> узел "B"
узел "A" <----- SYN(ISN+1)/ACK ------ узел "B"
узел "A" ------ ACK ----------------> узел "B"
```

Рисунок 4 порядок установки TCP-соединения

ISN – это начальный номер последовательности (Initial Sequence Number), уникальный для каждого TCP/IP соединения. С момента установки соединения, номера последовательности планомерно увеличиваются на количество принятых/отправленных байт. Впрочем, не будем углубляться в теорию.

Остановимся на том факте, что 32-битное поле ISN можно изменять псевдослучайным образом, "промодулированным" секретными данными и… никто ничего не заметит! Конечно, пропускная способность упадет до 4х байт на каждое TCP-соединение, устанавливаемое узлом-жертвой, а TCP-соединений устанавливается не так уж и много (особенно, если мы имеем дело не с сервером, а с рабочей станцией).

Тем не менее, для "перекачки" паролей и удаленного управления через командную строку, даже такой скромной пропускной способности вполне достаточно.

```
0           4              8           16     19    24                 32
-----------------------------------------------------------------------
|              Source Port             |         Destination Port       |
-----------------------------------------------------------------------
|                              Sequence Number                          |
-----------------------------------------------------------------------
|                           Acknowledgment Number                       |
-----------------------------------------------------------------------
| HLEN  |  Reserved |   Code Bits     |         Window                 |
-----------------------------------------------------------------------
|              Checksum               |         Urgent Pointer         |
-----------------------------------------------------------------------
|                                 Options    |        Padding          |
-----------------------------------------------------------------------
|                                  Data                                 |
-----------------------------------------------------------------------
```

Рисунок 5 формат TCP-заголовка

Жанна Рутковская решив не ограничивать себя лабораторными опытами, разработала протокол NUSHU, создающий скрытые пассивные каналы посредством модификации ISN с последующим шифрованием последнего алгоритмом DES (см. рис. 6) на основе идентификатора IP-пакета (IP.id), порта-источника (TCP.sport) и IP-адреса назначения (IP.daddr).

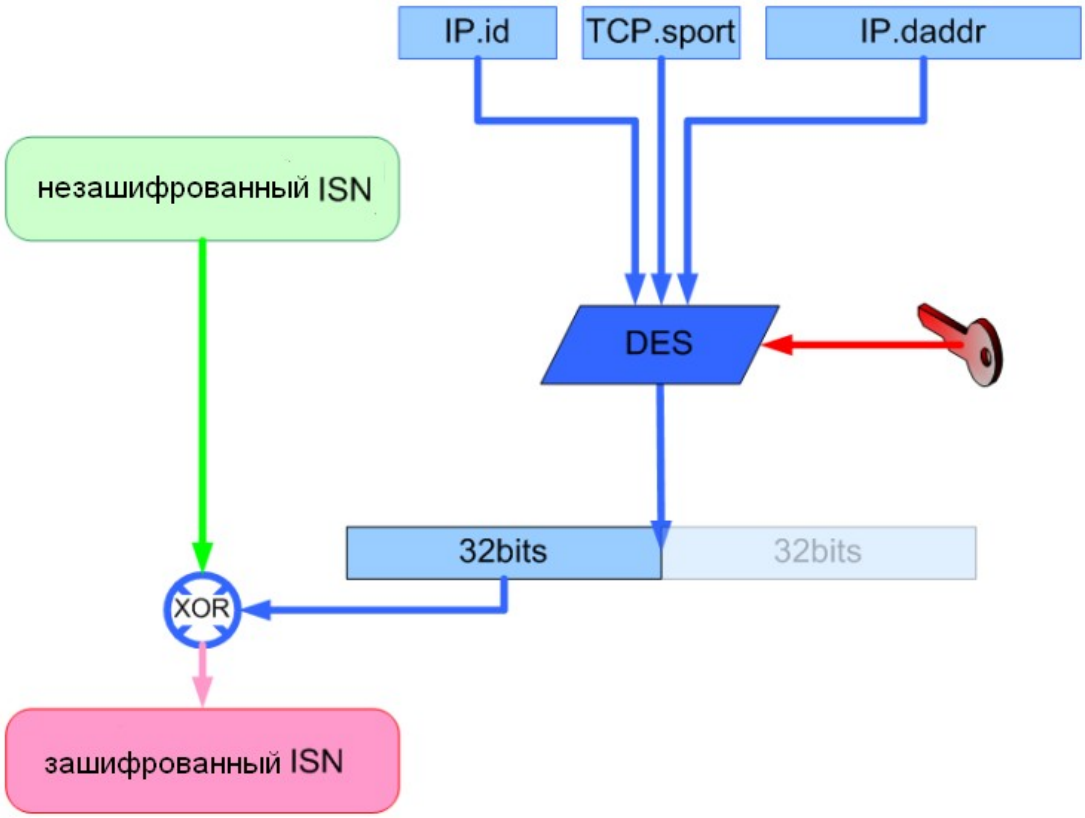

Рисунок 6 механизм шифрования ISN в протоколе NUSHU

сеанс практической магии

Идем на сайт Жанный Рутковской http://invisiblethings.org, видим там раздел с инструментами "tools", находим в нем "**NUSHU — passive covert channel engine for Linux 2.4 kernels**" и качаем архив исходных текстов http://invisiblethings.org/tools/nushu/nushu.tar.gz (всего 18 Кб) вместе с файлом readme — http://invisiblethings.org/tools/nushu/readme.txt.

Распаковываем, компилируем. Компиляция осуществляется стандартно. Просто запускаем утилиту make и получаем три модуля ядра: **nushu_receiver.o** (приемник), **nushu_sender.o** (передатчик), и **nushu_hider.o** (если компиляция прерывается сообщением об ошибке, значит, это косяк и прежде чем писать Рутковской гневные письма следует сначала почитать толковый faq по сборке модулей или раскурить статью "хачим ядро xBSD", ранее опубликованную в рубрике "юниксоида").

Приемник устанавливается на поломанный маршрутизатор, передатчик — на целевой узел жертвы. Для организации двухсторонней связи приемник/передатчик устанавливаются на оба узла сразу. Модуль nushu_hider.o в огранизации скрытого канала не участвует и предназначен для обмана локальных снифферов типа tcpdump, не позволяя им обнаружить факт изменения ISN.

Из readme следует, что модуль-передатчик обрабатывает следующие параметры командной строки:

- **dev**=<*device*>, где "device" – сетевое устройство, с которым предполагается работать (например, eth0, ppp0, etc);
- **cipher**=[*0|1*], где "0" означает передачу без шифрования, а "1" предписывает использовать DES;
- **key**="string", где "string" произвольная строка-маркер, идентичная строке-маркеру, установленной на модуле-приемнике (используется только в том случае если шифрование выключено, в противном случае игнорируется);
- **src_ip**=<ip_where_nushu_sender_is_placed> — IP-адрес узла, на котором установлен передатчик;

Модуль-приемник обрабатывает следующие ключи:

- **dev**=<*device*>, где "device" – сетевое устройство, с которым предполагается работать (например, eth0, ppp0, etc);
- **cipher**=[*0|1*], где "0" означает передачу без шифрования, а "1" предписывает использовать DES;
- **key**="string", где "string" произвольная строка-маркер, идентичная строке-маркеру, установленной на модуле-передатчике (используется только в том случае если шифрование выключено, в противном случае игнорируется);
- exclude_ip=<172.16.*.*> — задает список "неприкосновенных" IP-адресов, при отправке пакетов на которых протокол NUSHU задействован не будет, т. е. ISN останется неизменным (данный параметр является опциональным);

Модуль nushu_hider.o загружается без каких либо параметров (да и то, только в том случае, если в этом возникает необходимость).

Хорошо! Все модули успешно загружены, ядро функционирует нормально и в панику судя по всему впадать не собирается. Что делать дальше? А ничего!!! Ведь это только "движок" ("engine"), обеспечивающий функционирование PCC-каналов. К нему можно прикрутить key-logger или удаленный shell, но это уже придется делать самостоятельно. А как?! Ни readme, ни сопроводительные презентации не дают ответа на этот вопрос, поэтому приходится зарываться в исходные тексты и разбирать их на отельные байты.

Начнем с передатчика, реализованного в файле sender.c. В процедуре "init_module()", отвечающей за инициализацию модуля, сразу же бросаются в глаза следующие строки:

```
struct proc_dir_entry *proc_de = proc_mkdir ("nushu", NULL);
create_proc_read_entry ("info", 0, proc_de, cc_read_proc_info, NULL);
struct proc_dir_entry * wpde = create_proc_entry ("message_to_send", 0, proc_de);
```

Листинг 1 ключевой фрагмент процедуры init_module() файла sender.c

Все ясно! Модуль использует псевдофайловую систему /proc, создавая директорию "nushu", а в ней — два файла: "info" и "message_to_send", с которыми можно работать с прикладного уровня как с обычными устройствами (то есть, псевдоустройствами, но это уже детали).

Аналогичным образом обстоят дела и с приемником, реализованном в файле receiver.c, ключевой фрагмент из которого приведен ниже:

```
struct proc_dir_entry *proc_de = proc_mkdir ("nushu", NULL);
create_proc_read_entry ("message_received", 0, proc_de, cc_read_proc_message, NULL);
create_proc_read_entry ("info", 0, proc_de, cc_read_proc_info, NULL);
```

Листинг 2 ключевой фрагмент процедуры init_module() файла receiver.c

Как видно, вместо устройства "message_to_send" на этот раз создается "message_received" из которого можно читать получаемые сообщения через стандартные функции ввода/вывода.

В общем, имея исходные тексты на руках, со всеми этими причиндалами совсем несложно разобраться, тем более, что их суммарный объем составляет всего 69 Килобайт. Как говориться, настоящему хакеру — всего на одну затяжку.

Там же, в разделе инструментов, рядом с архивом NUSHU, лежит пара дампов, демонстрирующих технику его работы: invisiblethings.org/tools/nushu/nushu_dumps_10k.tar.bz2 и invisiblethings.org/tools/nushu/nushu_dumps_100k.tar.bz2.

заключение

Помимо описанных существуют и другие транспортные средства, пригодные для передачи скрытого трафика, например, опция штампа времени в TCP-заголовке. HTTP-протокол дает еще большие возможности, поскольку включает в себе множество факультативных полей, которые можно безболезненно модифицировать в весьма широких пределах. Однако, все это слишком заметно и наиболее стойким к обнаружению на сегодняшний день остается протокол NUSHU, работающий с ISN.

Может ли атакованный администратор обнаружить скрытые пассивные каналы хотя бы теоретически? Скрупулезный анализ сетевого трафика, вообще-то позволяет выявить некоторую ненормальность распределения ISN, но для этого потребуется обработать сотни тысяч "хакнутых" пакетов, сравнивая их с эталонными оригиналами, а потому намного проще выявить посторонний ядерный модуль, отвечающий за создание и поддержку PCC-каналов, используя общие методики верификации целостности системы, однако, это уже совсем другой разговор, к которому мы еще вернемся.

>>> врезка сравнительные характеристики различных транспортеров

транспортер	пропускная способность	легкость обнаружения	помехоустойчивость	легкость реализации
IP Identification	16-бит на пакет	практически невозможно обнаружить	очень высокая	легко реализуетсяр
TPC Initial Sequence Number, ISN	32-бита на каждое создаваемое TCP/IP соединение	практически невозможно обнаружить	очень высокая	умеренно сложная
TCP-Timestamp	1 бит на TCP сегмент	практически невозможно обнаружить на медленных каналах, но легко на быстрых	низкая	достаточно сложная
зарезервированные биты в TCP заголовке	6 бит на пакет	легко обнаруживаемая	низкая	предельно легкая
тип сервиса в IP	1 байт на пакет	легко обнаруживаемая	низкая	предельно легкая

Таблица 1 сравнительные характеристики основных транспортеров PCC-каналов

>>> врезка сакраментальное значение Nü Shu

Nu Shu – китайское слово, в буквальном смысле означающее "women's writing" (женское письмо). Это специальный слоговый алфавит фонетического типа, в котором каждый символ (из 600-700 возможных) обозначает часть слова или все слово целиком.

Рисунок 7 Nu Shu, начертанное на китайском языке

Nu Shu возник приблизительно в 15 веке нашей эры, зародившись в местечке Джангйонг (Jiangyong County), входящему в провинцию Хунань (Hunan, произносится как "пиньинь"), расположенной на юго-востоке Китая, женщины которой изобрели свой тайный язык, стилизованный под декоративные завитушки, наносимые, на одежду или непосредственно на само тело. Тайна, передавая от матерей к дочерям, сохранялась на протяжении веков и только в 1983 году о ней узнала мужская часть! Просто восхитительный пример конспирации!!! Женщины общаются, а мужики-то и не знают!

Рисунок 8 письмена NUSHU, замаскированные под декоративную роспись

Напрашивается очевидная параллель между "женскими письменами" и протоколом NUSHU, разработанном Жанной Рутковской и позволяющим передавать скрытую информацию в заголовков TCP-пакетов, неподвластную для выявления администраторам и всяким прочим мужчинам.

Подробнее о языке Nu Shu можно прочитать по ссылкам, приведенных в соответствующей врезке.

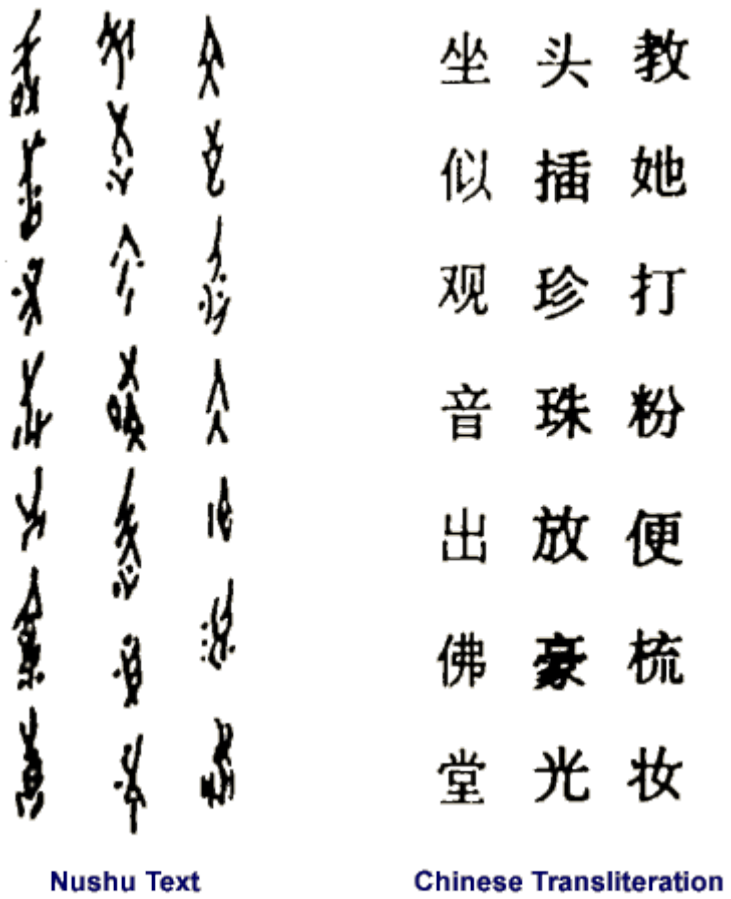

Рисунок 9 NUSHU и традиционные китайские письмена

>>> *врезка полезные ссылки*

- **DEVCC**:
  - законченная реализация модуля PCC-каналов, использующая в качестве транспортного средства опции штампа времени в TCP-заголовках (TCP time stamps), спроектированная для работы с ядрами Linux'a версии 2.4.9 или выше. распространяется в исходных текстах на бесплатной основе: http://www.mit.edu/~gif/covert-channel/src;
- *Craig H. Rowland* **"Covert Channels in the TCP/IP Protocol Suite"**:
  - статья, посвященная вопросом проектирования, реализации и выявления скрытых пассивных каналов с кучей демонстрационных примеров на си: http://www.firstmonday.dk/issues/issue2_5/rowland;
- *Andrew Hintz* **"Covert Channels in TCP and IP Headers"**:
  - основные тезисы презентации, рассматривающей доступные способы организации PCC-каналов и методы их обнаружения, а так же затрагивающая вопросы пропускной способности и устойчивости PCC-каналов к различных proxy-серверам, маршрутизаторам и брандмауэрам (на английском языке): http://guh.nu/projects/cc/covertchan.ppt;
- *Joanna Rutkowska* "**Linux Kernel Backdoors And Their Detection**":
  - презентация Жанны Рутковской, рассказывающая о методах сокрытия и обнаружения rootkit'ов, использующих PCC-каналы (на английском языке): http://invisiblethings.org/papers/ITUnderground2004_Linux_kernel_backdoors.ppt;
- *Joanna Rutkowska* "**Passive Covert Channels Implementation in Linux Kernel**":
  - еще одна шикарная презентация Жанны Рутковской, буквально нашпигованная техническими подробностями по организации сетевого стека Linux'a и технике перехвата чужих пакетов с модификацией их заголовков (на английском языке) http://invisiblethings.org/papers/joanna-passive_covert_channels-CCC04.ppt;

- *Joanna Rutkowska* **"The Implementation of PassiveCovertChannels in Linux Kernel":**
    - доклад, зачитанный Жанной на конференции Chaos Communication Congress December 2004, детально описывающим механизмы создания PCC-каналов во всех подробностях, ориентированный на программистов (на английском языке) http://invisiblethings.org/papers/passive-covert-channels-linux.pdf;
- NUSHU — proof-of-concept code (TGZ)
    - исходные тексты ядерных модулей Linxu'a, обеспечивающими создание и поддержку PCC-каналов по протоколу NUSHU (на языке си): http://invisiblethings.org/tools/nushu/nushu.tar.gz;
- Nü Shu:
    - познавательная статья о языке Nu Shu на википедии (на английском языке): http://en.wikipedia.org/wiki/N%C3%BC_Shu;
- Nushu:
    - еще одна статья, посвященная секретному женскому языку Nu Shu: http://www.ancientscripts.com/nushu.html;
- World of Nushu:
    - сайт для тех, кто хочет выучить язык NuShu с кучей методических материалов, настоятельно рекомендованных к всестороннему изучению, так же имеется много текстов на Nu Shu и образцов искусства, на которые стоит взглянуть: http://www2.ttcn.ne.jp/~orie/home.htm;