УДК 621. 39 (075)

СЕТЕВОЕ ОБНАРУЖЕНИЕ ПАССИВНЫХ СКРЫТЫХ КАНАЛОВ ПЕРЕДАЧИ ДАННЫХ В ПРОТОКОЛЕ TCP/IP

Д.Ш.ОСМОНАЛИЕВ, Р.Р.КАМАЕВ,
М.Ч.БОРУБАЕВ, К.Б.ТУРГАНОВ
E.mail. ksucta@elcat.kg

Бул макалада жасалма нейрондук тармактын негизинде ISN моделинин жардамы менен TCP/IP протоколунда маалымат жиберүү жашыруун каналдарын табуунун ыкмалары келтирилген.

В данной статье предлагается способ обнаружения наличия скрытого канала передачи данных в протоколе TCP/IP с помощью модели генерации ISN на основе искусственной нейронной сети.

The given article proposes the method of determining the presence of hidden data transfer channel in TCP/IP protocol with the help of ISN generation model on the basis of artificial neuron net.

В большинстве случаев после захвата управления атакующему необходимо получать информацию с компрометированного хоста (например, файлы паролей, личные документы пользователя, cookies и т.д.) и посылать команды на компрометированный хост. Для обеспечения скрытой передачи данных злонамеренным программным обеспечением используются скрытые каналы. Основная идея скрытых каналов заключается в том, чтобы передавать информацию в неиспользуемых полях сетевых протоколов или изменять некритичную информацию в сетевом протоколе. Разработано достаточно большое количество программ для создания скрытых каналов.

Loki2 /1/ для Linux использует скрытие информации в поле данных ICMP-пакетов и DNS-запросах и ответах. Инструмент Reverse WWW Shell, разработанный Ван Хаузером (van Hauser) /2/, использует протокол HTTP. Сервер Reverse WWW Shell создает обратное соединение с клиентом – периодически «проталкивает» запрос на получение команд с клиента и «вытягивает» команды, после выполнения команды результат «проталкивается»

обратно. Однако притом данные передаются в поле данных пакета и могут быть легко обнаружены. Более высокую незаметность обеспечивает изменение некритичных полей пакетов. Такие методы описаны в работе Крэйга Х. Роулэнда (Craig H. Rowland) “Covert Channels in TCP/IP Protocol Suite” /3/. Разработанный им инструмент Covert_TCP использует замену информации в служебных полях протокола IP и TCP: IP Identification, Sequence Number (SEQ#), Acknowledgement Number (ACK#).

Поле SEQ# представляет собой 32-битовое поле, которое идентифицирует положение пакета в сессии TCP, а также является идентификатором принадлежности пакета к определенной сессии. При установлении соединения TCP SEQ# инициализируется псевдослучайным значением Intial Sequence Number (ISN) и в течение сессии значение SEQ# инкрементируется на определенное значение. Значение ACK# используется для поддержания уникальности сессии другим абонентом соединения и равно SEQ# пакетов другого абонента. Оно также генерируется псевдослучайным значением ISN в процессе установки соединения. При передаче данных ACK# соответствует SEQ# другого абонента и соответственно также инкрементируется на определенное значение /4/. Величины начальных SYN# (ACK#) не являются существенными. Таким образом, значение SEQ# можно произвольно менять только один раз в начале сессии TCP. Атакующий может воспользоваться данным фактом, чтобы тайно передать в значение ISN нужную информацию.

1. NUSHU – экспериментальное средство создания пассивных каналов в TCP/IP

Один из наиболее новых инструментов, реализующих подобную атаку, является NUSHU для Linux, разработанный Джоанной Рутковска (Joanna Rutkowska). NUSHU – экспериментальный (proof-of-concept) инструмент для ОС Linux с ядрами 2.4 и 2.6. NUSHU обеспечивает так называемый пассивный канал передачи, т.е. не создает новый трафик, а изменяет данные уже существующего. Подробное описание NUSHU можно найти в /5/ и исходных кодах NUSHU /6/, здесь необходимо остановиться на некоторых аспектах важных для сетевого обнаружения скрытого канала с использованием NUSHU.

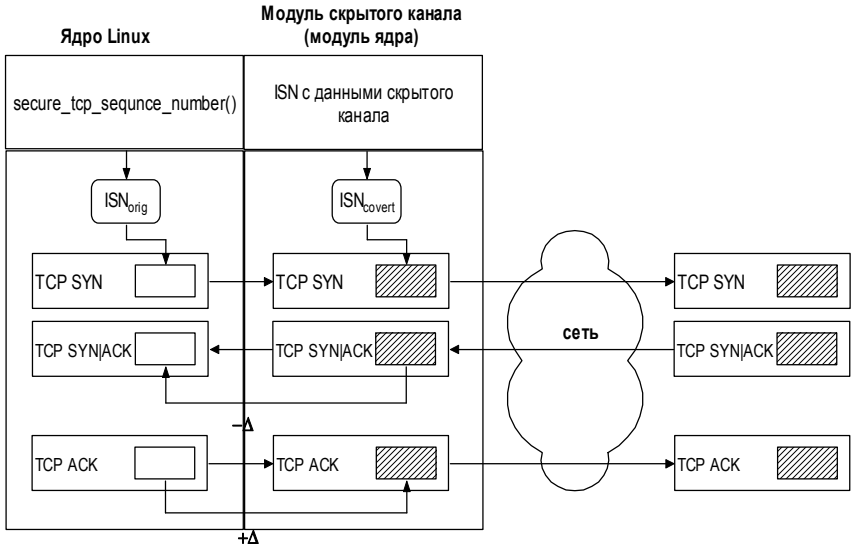

Рис. 1. Принцип работы NUSHU

NUSHU создает скрытый канал следующим образом. При установке соединения ядро генерирует и помещает его в поле SEQ# заголовка TCP отправляемого пакета. NUSHU является модулем ядра и имеет возможность изменить информацию в пакете, включая заголовок. NUSHU помещает в поле SEQ# данные, которые необходимо передать. Нужно отметить, что для того чтобы оригинальный стек поддерживал соединения, измененные NUSHU во входящих пакетах, необходимо заменить ACK#. Для этого NUSHU сохраняет величину. После чего пакет отправляется в сеть. При приходе пакета, содержащего ACK# на отправленные ранее пакеты со скрытым каналом, NUSHU вычисляет и получает оригинальный ACK#. Идентификация соединения в данном случае осуществляется по полям адреса отправителя, адреса получателя, порта отправителя, порта получателя. Принцип работы NUSHU иллюстрируется рис. 1.

Перед помещением в TCP пакет данные шифруются. Для шифрования используется простая блочная версия алгоритма Вернама (Gilbert Vernam) с генерацией одноразового ключа на основе алгоритма DES.

Основная цель шифрования – не обеспечение безопасности, для этого алгоритм слишком слаб, а создание у передаваемых SEQ# статистических характеристик, близких к случайным. Процесс шифрования приведен на рис. 2.

На рис. 2 TCP.sport и TCP.dport – порты отправителя и получателя, соответственно, IP.saddr и IP.daddr – IP адреса отправителя и получателя, соответственно. Ключ – это секретный ключ, разделяемый данной копией NUSHU и ее владельцем – атакующим, который планирует читать передаваемые NUSHU данные. Таким образом, для каждого нового соединения создается новый (сессионный) ключ.

Для получения данных, передаваемых NUSHU, злоумышленник должен иметь возможность перехватить весь (или большую часть) трафика со скрытой информацией. В работе /5/ Джоанна Рутковска предлагает следующие схемы применения NUSHU:

1. Скомпрометированная рабочая станция соединена с Интернет через захваченный злоумышленником шлюз. При этом захваченная станция сама является инициатором большинства соединений (SYN-пакетов). В этом случае NUSHU помещает данные в SYN# первого пакета сессии.

2. Скомпрометированный сервер соединен с Интернет через захваченный злоумышленником шлюз. При этом инициатором соединений являются клиенты данного сервера. В этом случае NUSHU помещает данные в SYN# второго пакета сессии (SYN|ACK - пакета).

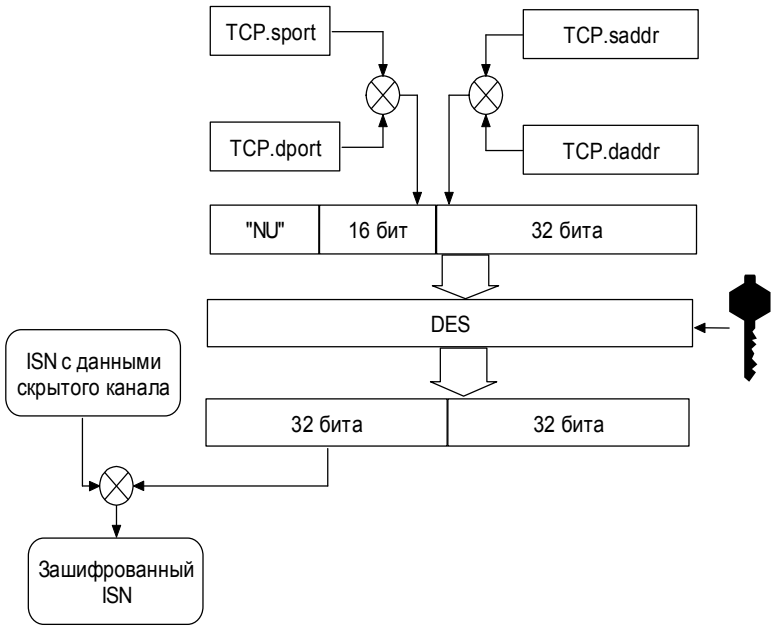

Рис. 2. Шифрование данных при отправке

2. Обнаружение скрытого канала

В своей работе Джоанна Рутковска указывает, что существует большое количество возможностей хостового обнаружения пассивного скрытого канала. Сетевое обнаружение наличия скрытого канала представляется более сложным. На рис. 3 представлена ситуация сетевого обнаружения скрытого канала. Сенсор перехватывает весь трафик, проходящий по сегменту сети, и определяет наличие в трафике скрытого канала. При обнаружении необходимо построить решающее правило, которое позволит разделять ISN, сгенерированные оригинальным стеком, и ISN, сгенерированные NUSHU. Построение разделяющего правила возможно при учете следующих факторов. В большинстве

операционных систем ядро генерирует ISN не случайным образом. ISN является сложной функцией от текущего времени и предыдущего значения ISN. NUSHU генерирует ISN случайным образом, поскольку шифрует данные на случайном сеансовом ключе, сгенерированном на DES.

В данной работе предлагается следующий способ обнаружения наличия скрытого канала. На основе экспериментальных данных ISN оригинального стека строится модель генерации ISN, позволяющая предсказывать следующее значение ISN на основе предыдущих.

Модель создается на основе искусственной нейронной сети. Нейронная сеть позволяет создать модель по экспериментальным данным, без знания алгоритма генерации данных, что в ряде случаев является очень важным. Методы обучения нейронных сетей и их архитектуры подробно описаны в /7/, а сейчас отметим следующий важный момент: нейронные сети обладают свойствами обобщения (т.е. выдают правильный результат не только на обучающих данных, но и других подобных им) и коррекции ошибки (т.е. выдают правильный результат, если часть входных данных искажена или отсутствует). Эти свойства появляются у нейронной сети в процессе обучения и являются его следствием.

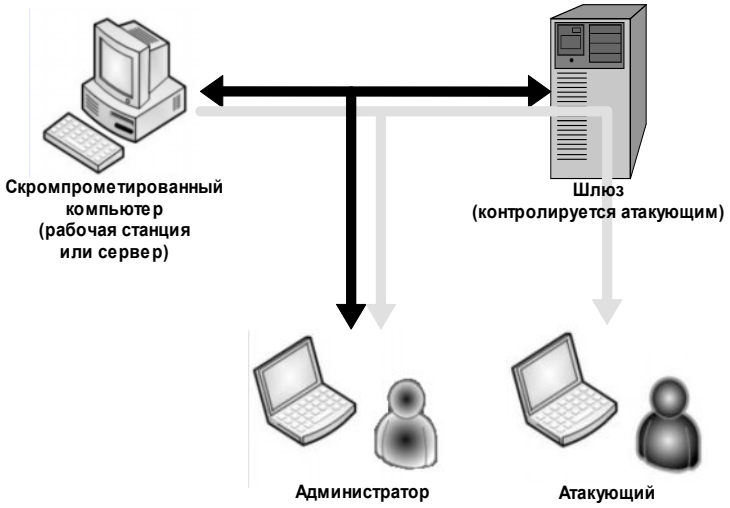

Рис. 3. Схема обнаружения

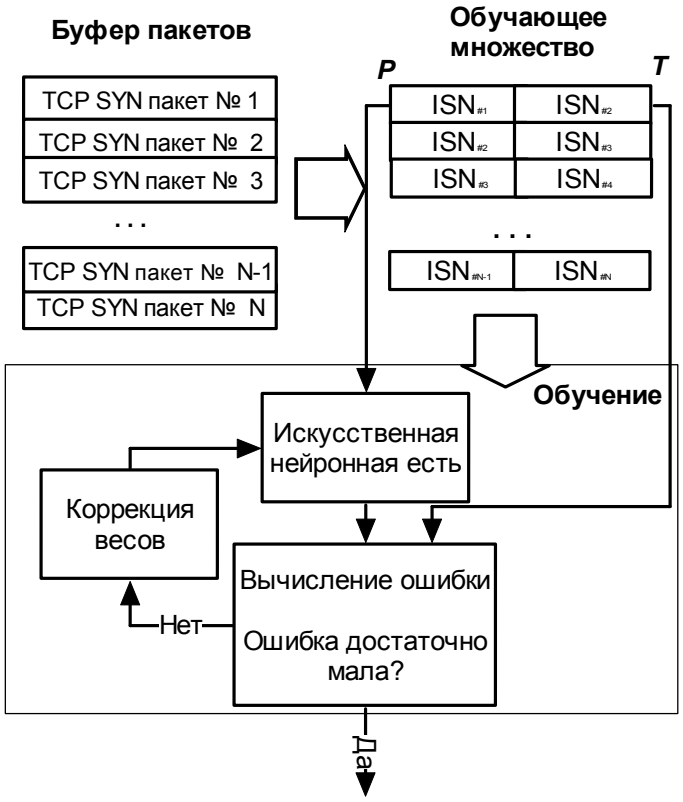

Рис. 4. Обучение нейронной сети

Таким образом, разработанная система обнаружения должна предварительно построить модель генерации ISN, фактически система должна быть предварительно обучена распознавать наличие скрытого канала. Для этого собираются ISN, сгенерированные гарантированно чистым стеком. Собранные ISN рассматриваются как обучающая выборка. По данной выборке строится таблица, $i=N-1$, где N – число собранных чистых ISN. Далее нейронная сеть обучается воспроизводить отношение. Обучение нейронной сети приведено на рис. 6. Кроме того, на этапе обучения рассчитывается порог близости. При превышении данного порога система принимает решение, что тестируемые пакеты не соответствуют модели стека.

После построения модели в фазе детектирования система перехватывает SYN-пакеты (другие пакеты соединения не нужны) в контролируемой сети. По перехваченному значению пакетов система пытается предсказать следующий ISN. Поскольку этот ISN также перехвачен, то имеется возможность сравнить предсказанный и актуальный ISN.

Степень совпадения значений показывает степень совпадения данных в сети построенной модели. При значительном отличии предсказанных ISN от актуальных можно считать, что ISN сгенерирован не оригинальным стеком. В качестве меры отличия выбрано расстояние Хэмминга между двумя двоичными числами. Процесс показан на рис. 5.

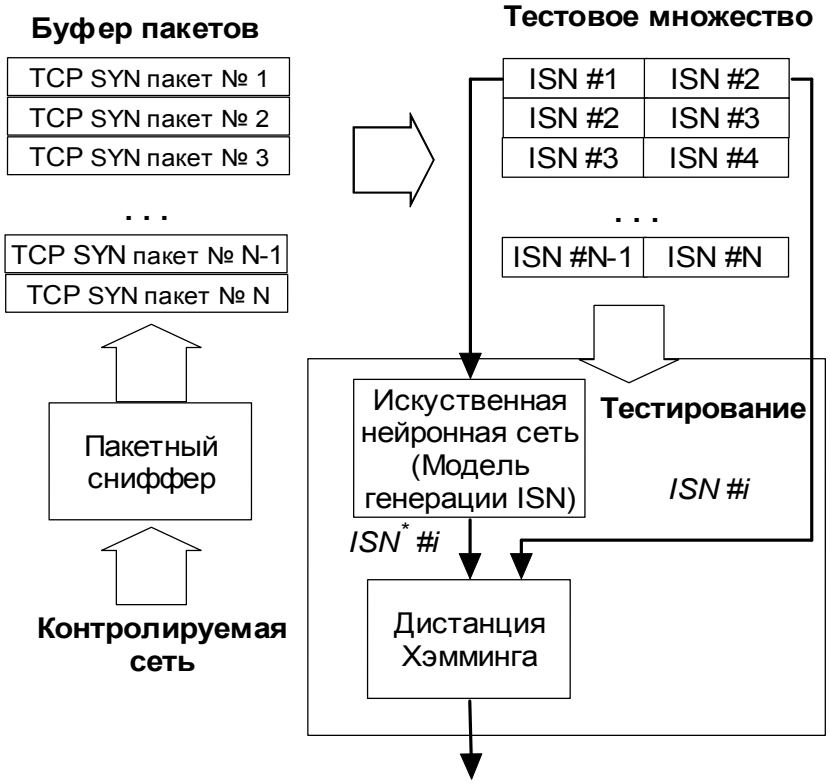

Рис. 5. Определение похожести пакетов

Нам необходимо учитывать, что в сети могут находиться рабочие станции с различными операционными системами. Опираясь на тот факт, что ISN в различных операционных системах генерируются с использованием различных формул с учетом времени или числа соединений, можно утверждать, что модели генерации ISN различных ОС могут отличаться весьма сильно. Для того чтобы учесть такие различия, создаются модели ISN для нескольких (как можно большего числа) операционных систем. Каждая модель представляет собой нейронную сеть, обученную на предсказание пакетов данного стека. В процессе мониторинга для каждого анализируемого пакета проводится тестирование всех имеющихся в распоряжении моделей стеков и находится модель, для которой ошибка предсказания минимальна. Далее минимальная ошибка сравнивается с порогом.

К сожалению, по одному SYN-пакету невозможно определить наличие скрытого канала. Для детектирования необходимо большее число пакетов. Таким образом, процесс, показанный на рис. 6, повторяется несколько раз для нескольких пакетов из буфера. При этом всегда вычисляется мера близости между предыдущим и следующим за ним пакетом. Чем больше пакетов предсказывается, тем большую точность имеет оценка соответствия реальных ISN модели.

3. Экспериментальные результаты

Для проверки разработанного метода использованы данные, предоставленные Джоанной Рутковска /5/, а также полученные на тестовой сети. Данные представляют собой логи TCPDump содержащие SYN-пакеты. В экспериментах использовались SYN-пакеты оригинальной системы Fedora Core 2 (5), системы Fedora Core 3, Windows 2000 SP4, Windows XP SP1, Windows XP SP2, а также системы Fedora Core с NUSHU. Пример записи:

18:11:32.624303 172.16.7.10.1025 > local-172-16-0-1.ttn.16131:
S **130110378**:130110378(0) win 5840 <mss 1460, sackOK, timestamp 5603914 0,nop, wscale 0> (DF)

Для экспериментов необходимо только значение SEQ# (выделенное жирным шрифтом). В логах содержится по 20 тысяч записей каждого типа. Для обучения сети использованы 1500 значений оригинальной системы. 1500 записей каждого типа используется для построения экспериментальной модели. Остальные данные используются для тестирования системы.

На стадии генерации модели считываются 1500 значений для каждой операционной системы и используется скрипт Mathworks Matlab 6.5 R13 для создания и обучения нейронной сети. В работе использована рекуррентная сеть Элмана, поскольку для предсказания следующего ISN необходимо не только значение одного предыдущего, но и всех остальных. Поле обучения – нейронная сеть для каждой модели – сохраняется в файле.

На стадии мониторинга используется скрипт Mathworks Matlab 6.5 R13, который читает пакеты из лога WinDump (или любого заданного файла) один за одним. Это нужно для того, чтобы организовать мониторинг сети в режиме on-line. На рис. 8, а) приведен вывод скрипта мониторинга при мониторинге логов чистой Fedora. На рис. 8, б) приведен вывод скрипта при мониторинге логов чистой Windows 2000 SP4. На рис. 8, в) приведен вывод скрипта при мониторинге логов с NUSHU.

```
********************* IP STATISTICS ********************
IP:172.16.7.10->STACK:FEDORA CORE 2.0 ORIGINAL->9.5|6
```

а

```
********************* IP STATISTICS ********************
IP:192.168.164.105->STACK:WINDOWS 2000 SP4 ORIGINAL->6.8421|19
```

б

```
********************* IP STATISTICS ********************
IP:172.16.100.2->STACK:NUSHU ISN COVERT CHANNEL->15.8333|6
```

в

Рис. 6. Вывод скрипта мониторинга

На рис. 6 первая колонка (IP) IP-адрес источника проверяемых пакетов, вторая (STACK) предполагаемый стек, третья – средняя ошибка предсказания (от 0 до 32), последняя – число проверенных пакетов с данного IP-адреса. В табл. 1 приведена короткая сводка результатов тестирования скрипта создания модели.

Таблица 1

Создание модели

Размер сети Элмана	Число итераций	Время создания модели, минут *
30x32	1000	8 – 10
40x32	1000	15 – 20
50x32	1000	20 – 30
100x32	1000	40 – 60

* - PC AMD Athlon XP 1800+, RAM 512

В табл. 2 приведена сводка тестирования скрипта мониторинга.

Таблица 2

Зависимость вероятности ошибки от размера нейронной сети

Размер сети Элмана	Ложное обнаружение скрытого канала	Ложный пропуск скрытого канала	Ошибка определения ОС **
30x32	<0.1 %	5 %	<1%
40x32	<0.2 %	7 %	<2%
50x32	<0.2 %	8 %	<3%
100x32	<0.5 %	10 %	<4%

** – для 4 ОС: Fedora Core 2, Fedora Core 3, Windows 2000 SP4, Windows XP SP1, Windows XP SP2 не учтена.

В табл. 3 приведены результаты тестирования точности распознавания для оптимального размера сети (30x32) при разном количестве перехваченных ISN.

Таблица 3

Зависимость точности распознавания от числа ISN

Число ISN	<3	<50	<100	>100
Ложное обнаружение скрытого канала	50 %	5 %	0.1 %	<0.1%
Ложный пропуск скрытого канала	50 %	8 %	6 %	5 %
Ложное определение ОС **	50 %	5 %	1 %	<1 %

Как уже упоминалось выше, в случае малого количества ISN не может быть гарантирована точность решения. Поэтому, если число накопленных ISN меньше 3, то система считает, что стек не распознан. Необходимо отметить, что для Windows XP SP2 вероятность ошибки определения NUSHU составляет порядка 80 %. Это является следствием того, что механизм генерации ISN данной операционной системой близок к истинному случайному распределению и таким образом подобен методу генерации NUSHU. Производительность скрипта мониторинга позволяет выполнять мониторинг 10 Мегабитной сети при 50%-ной загрузке в режиме on-line.

Заключение

Эксперименты показывают, что разработанный метод обеспечивает точное определение наличия пассивного скрытого канала. Кроме того, метод позволяет автоматически создавать модели генерации ISN для любых операционных систем (при генерации можно не знать принципа формирования ISN в ОС). Другой важной возможностью данной методики является автоматическое определение операционной системы по значениям ISN.

Основная проблема, которой будут посвящены наши дальнейшие исследования, – это детектирование скрытого канала при условии апериодического установления соединений. Известно, что во многих операционных системах ISN генерируется в зависимости от интервала между предыдущим и текущими соединениями. Пока мы использовали данные, в которых соединения как чистых, так и скомпрометированных машин устанавливаются через фиксированный малый промежуток времени в течение нескольких дней. Мы не знаем, как поведет себя наш метод в случае нерегулярных соединений на длительных временных интервалах. Кроме того, наши дальнейшие исследования будут включать разработку программы обнаружения скрытого канала. Сейчас мы используем WinDump и скрипт, выполняемый в Mathworks Matlab, что неизбежно снижает производительность нашей системы. Разработка программы выполняющей захват и мониторинг позволит исключить такое падение производительности.

Список литературы

1. Daemon9. L O K I 2 Implementation, Phrack Magazine, Volume 7, Issue 51 September 01, 1997, URL, c.35.

2. van Hauser/THC. Placing Backdoors Through Firewalls, URL, c. 47.

3. Rowland C.H. Covert channels in the TCP/IP protocol suite, Now available, c. 23.

4. RFC793: TRANSMISSION CONTROL PROTOCOL, URL, c. 20.

5. Rutkowska J. The Implementation of Passive Covert Channels in the Linux Kernel, URL, c. 31.

6. Rutkowska J. NUSHU sources, URL, c. 43.