Simple Active Attack Against TCP
Простая атака на протокол TCP

Laurent Joncheray
Merit Network, Inc.
4251 Plymouth Road, Suite C
Ann Arbor, MI 48105, USA
Phone: +1 (313) 936 2065
Fax: +1 (313) 747 3185
E-mail: lpj@merit.edu

Аннотация
В этой статье[1] описывается способ организации активной атаки на протокол TCP, который позволяет атакующему перенаправить поток данных TCP на свой компьютер и обойти защиту, обеспечиваемую такими средствами как одноразовые пароли [skey] или квитирование [kerberos]. Соединения TCP уязвимы для каждого, кто имеет анализатор протоколов[2] и генератор пакетов на пути, используемом для передачи пакетов данного соединения. Представлены некоторые схемы обнаружения таких атак и методы их предотвращения, а также некоторые интересные детали поведения протокола TCP.

1. Введение
Пассивные атаки с использованием анализаторов становятся все более частым явлением в сети Internet. Атакующий перехватывает имя пользователя и пароль, что позволяет ему зарегистрироваться в системе как легитимному пользователю. Для предотвращения таких атак используются более сложные схемы аутентификации с однократными паролями [skey] или квитированием [kerberos]. Такие решения предотвращают перехват паролей в незащищенных сетях, но пока потоки данных передаются без шифрования[3] и цифровых подписей, сохраняется возможность активных атак. Многие заблуждаются, думая, что активные атаки слишком сложны и риск по этой причине невелик.

В статье описывается очень простой способ организации активной атаки, который можно использовать для несанкционированного доступа к хостам Unix. Для реализации такой атаки не требуется ничего, сверх инструментов, используемых для пассивного прослушивания. Проверка этого способа осуществлялась в реальных условиях с предупреждением атакуемых пользователей. В статье также описаны некоторые особенности поведения протокола TCP и представлены некоторые примеры и статистика воздействия атаки на работу сети. В дополнение к этому рассматриваются некоторые схемы обнаружения и предотвращения таких атак. Статья начинается с краткого описания протокола TCP, чтобы обеспечить понимание тем, кто недостаточно хорошо знаком с этим протоколом.

Читателя может также заинтересовать вариант атаки, предложенных Моррисом в статье [morris85]. Описываемая здесь атака основана на варианте Морриса, однако она применима к любым соединениям TCP. В параграфе 7 приводится сравнение этих атак.

Описание атаки включает три части: “Состояние Established” (состояние открытой сессии с реальной передачей данных), организация сессий и некоторые примеры.

2. Состояние Established
2.1 Протокол TCP
В этом параграфе приводится краткое описание протокола TCP. Подробные сведения об этом протоколе можно найти в [rfc793]. Протокол TCP поддерживает полнодуплексные соединения между парами точек с гарантией доставки пакетов. Уникальная идентификация соединений обеспечивается квартетом из IP-адресов и номеров портов TCP для отправителя и получателя. Каждый байт, переданный хостом, имеет порядковый номер (32-битовое целое число) и прием подтверждается получателем с использованием порядкового номера, указанного отправителем. Порядковый номер для первого байта, передаваемого через соединение определяется на этапе организации соединения. Для каждого нового соединения стартовый порядковый номер изменяется в соответствии с правилами, позволяющими избежать использования совпадающих номеров для двух существующих одновременно соединений TCP.

Далее предполагается, что один из участников соединения является сервером (например, сервером telnet), а другой выступает в качестве клиента этого сервера. Определим также используемые далее термины:

Термин	*Определение*
SVR_SEQ	Порядковый номер следующего байта, который будет передан сервером.
SVR_ACK	Порядковый номер следующего байта, ожидаемого сервером (порядковый номер последнего принятого байта + 1).
SVR_WIND	Размер приемного окна сервера.
CLT_SEQ	Порядковый номер следующего байта, который будет передан клиентом.

[1] Оригинал статьи вы сможете найти на сайте http://www.cotse.com/texts/iphijack.txt. *Прим. перев.*
[2] В оригинале - sniffing tool. *Прим. перев.*
[3] Протокол Kerberos обеспечивает возможность шифрования потока данных.

Термин	*Определение*
`CLT_ACK`	Порядковый номер следующего байта, ожидаемого клиентом (порядковый номер последнего принятого байта + 1).
`CLT_WIND`	Размер приемного окна клиента.

На начальном этапе, когда еще не было реального обмена данными, выполняются условия SVR_SEQ = CLT_ACK и CLT_SEQ = SVR_ACK. Эти равенства будут верны и в тех случаях, когда соединение бездействует (стороны не передают данных, но соединение сохраняется). В процессе обмена данными эти равенства перестают выполняться. Более общие уравнения имеют вид:

```
CLT_ACK <= SVR_SEQ <= CLT_ACK + CLT_WIND
SVR_ACK <= CLT_SEQ <= SVR_ACK + SVR_WIND
```

Поля заголовков TCP называются обычно:

Поле	*Описание*	
`Source Port`	Номер порта на стороне отправителя пакета.	
`Destination Port`	Номер порта на стороне получателя пакета.	
`Sequence number`	Порядковый номер первого байта в данном пакете.	
`Acknowledgment Number`	Порядковый номер первого байта, который получатель ожидает в следующем пакете.	
`Data Offset`	Смещение данных в пакете[1]	
`Control Bits`	**Флаг**	**Значение**
	`URG`	Флаг наличия в пакете срочных данных.
	`ACK`	Подтверждение.
	`PSH`	Необходимость передачи данных без буферизации.
	`RST`	Сброс соединения.
	`SYN`	Синхронизация порядковых номеров для соединения.
	`FIN`	У отправителя больше нет данных для передачи через это соединение.
`Window`	Размер окна отправителя.	
`Checksum`	Контрольная сумма заголовка TCP и данных.	
`Urgent Pointer`	Указатель на срочные данные в пакете.	
`Options`	Опции TCP.	

- **SEG_SEQ** будет использоваться для обозначения порядкового номера (из заголовка пакета.
- **SEG_ACK** обозначает порядковый номер подтверждения.
- **SEG_FLAG** обозначает биты управления (флаги).

В типичном пакете[2], передаваемом клиентом устанавливаются значения SEG_SEQ = CLT_SEQ и SEG_ACK = CLT_ACK.

Протокол TCP использует 3-этапное согласование[3] порядковых номеров при организации нового соединения. Если предположить, что клиент инициировал соединение с сервером и обмена данными еще не произошло, нормальная последовательность пакетов будет иметь вид, показанный на рисунке 1.

- Клиентская сторона соединения находится в состоянии CLOSED, а сервер – в состоянии LISTEN.
- Клиент сначала передает свой стартовый порядковый номер в пакете с флагом SYN

```
SEG_SEQ  =  CLT_SEQ_0,
SEG_FLAG =  SYN
```

  после этого клиентская сторона переходит в состояние SYN-SENT.

- Получив от клиента запрос на соединение, сервер подтверждает порядковый номер клиента и передает свой стартовый номер в пакете с флагом SYN

```
SEG_SEQ  =  SVR_SEQ_0,
SEQ_ACK  =  CLT_SEQ_0+1,
SEG_FLAG =  SYN
```

  после чего устанавливает

```
SVR_ACK=CLT_SEQ_0+1
```

  и переходит в состояние SYN-RECEIVED

- Получив от сервера отклик, клиент подтверждает порядковый номер сервера

```
SEG_SEQ  =  CLT_SEQ_0+1,
SEQ_ACK  =  SVR_SEQ_0+1
```

  и устанавливает

[1] Относительно сегмента данных TCP. *Прим. перев.*
[2] Не повторном.
[3] Английский термин - three-way handshake. *Прим. перев.*

```
CLT_ACK=SVR_SEQ_0+1
```

после чего переходит в состояние ESTABLISHED.

- После получения от клиента подтверждения порядкового номера сервер переходит в состояние ESTABLISHED. В результате мы имеем

```
CLT_SEQ  =  CLT_SEQ_0+1
CLT_ACK  =  SVR_SEQ_0+1
SVR_SEQ  =  SVR_SEQ_0+1
SVR_ACK  =  CLT_SEQ_0+1
```

Сервер	*Пакеты*	*Клиент*
`LISTEN`	`<-  SYN, CLT_SEQ_0`	`CLOSED`
`LISTEN`	`SYN, SVR_SEQ_0, CLT_SEQ_0+1 ->`	`SYN-SENT`

`SYN-RECEIVED`	`<-  ACK, CLT_SEQ_0 + 1 SVR_SEQ_0+1`	`ESTABLISHED`
		`SVR_SEQ = CLT_SEQ_0 + 1`
		`CLT_ACK = SVR_SEQ_0 + 1`

`ESTABLISHED`
`SVR_SEQ = SVR_SEQ_0 + 1`
`SVR_ACK = CLT_SEQ_0 + 1`		

Рисунок 1: Процедура организации соединения

Для закрытия соединения используются пакеты с флагом FIN или RST. При получении пакета RST хост переводит соединение в состояние CLOSED и освобождает все ресурсы, связанные с этим экземпляром соединения. Пакет сброса соединения не подтверждается. Любые входящие пакеты для этого соединения будут отбрасываться хостом.

При получении пакета FIN хост переводит соединение в состояние CLOSE-WAIT и начинает процесс закрытия соединения. Детали этого процесса выходят за пределы темы данной статьи и не рассматриваются здесь. Интересующиеся читатели могут найти описание этого процесса в [rfc793].

При рассмотрении процесса организации соединения были сознательно опущены детали, связанные с возникающими на этапе организации соединения сложностями (повтор передачи, потеря пакетов и т. п.). Эти детали можно не принимать во внимание при изучении атаки.

В состоянии ESTABLISHED хост воспринимает пакет, если порядковый номер в заголовке находится в пределах ожидаемого сегмента

```
[SVR_ACK, SVR_ACK + SVR_WIND]
```

(для сервера) или

```
[CLT_ACK, CLT_ACK + CLT_WIND]
```

(для клиента). Если порядковый номер выходит за указанные пределы, пакет отбрасывается и передается подтверждение с ожидаемым порядковым номером, как показано ниже.

```
SEG_SEQ  =  200,
SVR_ACK  =  100,
SVR_WIND =  50
```

SEG_SEQ > SVR_ACK + SVR_WIND. Сервер формирует пакет ACK с

```
SEG_SEQ  =  SVR_SEQ
SEG_ACK  =  SVR_ACK
```

который показывает, что сервер ожидает получить в пакете.

2.2 Состояние десинхронизации

Термин “состояние десинхронизации[1]” используется для обозначения соединений, обе стороны которых находятся в состоянии ESTABLISHED, данные через соединение не передаются (стабильное состояние) и

```
SVR_SEQ  !=  CLT_ACK
CLT_SEQ  !=  SVR_ACK
```

Такое состояние будет стабильным, пока данные через соединение не передаются. При передаче данных возможны два случая:

- Если CLT_SEQ < SVR_ACK + SVR_WIND и CLT_SEQ > SVR_ACK, пакет принимается и данные сохраняются в буфере для последующего использования[2], но не передаются пользователю, поскольку часть данных (с номером SVR_ACK) отсутствует.
- Если CLT_SEQ > SVR_ACK + SVR_WIND или CLT_SEQ < SVR_ACK, пакет отбрасывается и данные теряются.

В обоих случаях обмен данными невозможен, несмотря на существование соединения в состоянии ESTABLISHED.

2.3 Атака

Предлагаемый вариант атаки состоит в создании состояния десинхронизации на обеих сторонах соединения TCP, чтобы прервать процесс обмена данными между хостами. Атакующий хост в этом случае используется для создания пакетов, воспринимаемых обеими сторонами как часть реального соединения.

Предположим, что сессия TCP находится в состоянии десинхронизации и клиент передает пакет с

```
SEG_SEQ  =  CLT_SEQ
```

[1] В английском языке - desynchronized state. *Прим. перев.*
[2] В зависимости от реализации протокола.

```
SEG_ACK  =  CLT_ACK
```

Поскольку CLT_SEQ != SVR_ACK, данные не будут приняты сервером и пакет будет отброшен. Атакующий хост передает такой же пакет, но изменяет в нем SEG_SEQ и SEG_ACK[2] на значения

```
SEG_SEQ  =  SVR_ACK,
SEG_ACK  =  SVR_SEQ
```

которые будут приняты сервером. В результате сервер будет обрабатывать полученные от атакующего хоста данные.

Если CLT_TO_SVR_OFFSET указывает на SVR_ACK – CLT_SEQ, а SVR_TO_CLT_OFFSET – на CLT_ACK – SVR_SEQ, атакующий хост переписывает значения в пакете TCP, направленном сервером клиенту на:

```
SEG_SEQ <- SEG_SEQ + CLT_TO_SVR_OFFSET
SEG_ACK <- SEG_ACK - SVR_TO_CLT_OFFSET
```

Предположим, что атакующий может просматривать все пакеты, передаваемые между участниками соединения, и способен подменять любые пакеты IP (маскируясь под сервер и клиента одновременно). Тогда атакующий может пропускать все пакеты данных через свою машину, которая может добавлять информацию в поток или удалять из потока данные. Например, если соединение используется для удаленного доступа к серверу с использованием telnet, атакующий может включить в поток любые команды (например, echo merit.edu lpj > ~/.rhosts) и фильтровать любой нежелательный вывод, чтобы легитимный пользователь не заметил подмены. Естественно, что в этом случае потребуется замена значений CLT_TO_SVR_OFFSET и SVR_TO_CLT_OFFSET. Оставим это в качестве упражнения для читателя. Атакующий может просто отключить вывод в сеансе telnet, чтобы избавиться от необходимости фильтрации[3].

2.4 Буря TCP ACK

Недостатком атаки является генерация большого числа пакетов TCP ACK. При получении неприемлемого пакета хост подтверждает его, передавая ожидаемый порядковый номер и указывая свой порядковый номер. Этот пакет также является неприемлемым и будет порождать пакет подтверждения, который, в свою очередь, также породит пакет подтверждения и т. д. В результате возникает цикл обмена пакетами, который может длиться бесконечно.

Поскольку такие пакеты не содержат данных, при их потере повторной передачи не происходит. Это означает, что отбрасывание единственного пакета будет разрывать образовавшийся цикл. К сожалению (или к счастью?) протокол TCP использует на сетевом уровне не обеспечивающий гарантированной доставки протокол IP для которого характерна потеря некоторой части пакетов, что приводит в конце концов к завершению цикла. Чем большее число пакетов отбрасывается, тем короче будет ACK-буря (цикл). Отметим также, что такие циклы обладают некоторой саморегуляцией – чем больше будет циклов, тем больше уровень трафика, связанное с ним насыщение и потеря пакетов.

Цикл возникает всякий раз, когда клиент или сервер передает данные. При отсутствии передачи данных такой цикл просто не возникает. При отсутствии атакующего, который подтвердит получение переданного пакета данных, возникает повторная передача, которая порождает бурю и, в конце концов, соединение будет разорвано ввиду отсутствия пакета ACK для переданных данных. Если атакующий подтверждает данные, возникает только одна буря[4].

Для атаки используется второй тип пакетов, описанных в параграфе 2.2. Первый случай, когда данные сохраняются получателем для последующего использования, просто не проверялся. В этом случае не возникает бури пакетов ACK, однако существует опасность при реальной обработке полученных данных. Кроме того, этот вариант сложнее реализовать для соединений с малым размером окна.

3. Организация сеанса (атаки)

В статье представлены два метода десинхронизации соединений TCP. Возможны и другие способы, но они здесь не рассматриваются. Предполагается, что атакующий может прослушивать все пакеты, передаваемые между участниками соединения.

3.1 Упреждающая десинхронизация

Этот метод основан на прерывании соединения на этапе его организации и создание нового соединения с другими порядковыми номерами. Выполняемые операции перечислены ниже и проиллюстрированы на рисунке 2.

- Атакующий прослушивает пакеты SYN/ACK направленные сервером клиенту (второй этап организации соединения).
- При обнаружении такого пакета атакующий передает серверу пакет RST и вслед за ним пакет SYN с такими же параметрами (порт TCP), но другим порядковым номером (далее этот номер обозначается как ATK_ACK_0).
- При получении пакета RST сервер закроет первое соединение и откроет соединение заново, получив новый пакет SYN для того же порта с другим порядковым номером (SVR_SEQ_0'). В ответ сервер передаст пакет SYN/ACK.
- При обнаружении этого пакета атакующий передаст серверу пакет ACK и сервер переключится в состояние ESTABLISHED.
- Клиент переключится в состояние ESTABLISHED при получении от сервера первого пакета SYN/ACK.

[2] А также контрольную сумму.

[3] Проведенная проверка показала, наличие ошибки в реализации telnet (или в самом протоколе telnet). Если пакет TCP содержит обе команды IAC DONT ECHO и IAC DO ECHO, процессор telnet будет отвечать сообщениями IAC WONT ECHO и IAC WILL ECHO. Другая сторона будет подтверждать получение IAC DONT ECHO и IAC DO ECHO, что приведет к возникновению бесконечного цикла.

[4] На практике атакующий часто теряет пакет данных вследствие загрузки сети и подтверждает первый повтор.

Сервер	*Пакеты*	*Клиент*
LISTEN	<- SYN, CLT_SEQ_0	CLOSED
LISTEN	SYN, SVR_SEQ_0, CLT_SEQ_0+1 ->	SYN-SENT
SYN-RECEIVED	<= RST, CLT_SEQ_0 + 1	ESTABLISHED SVR_SEQ = CLT_SEQ_0 + 1 CLT_ACK = SVR_SEQ_0 + 1
CLOSED	<= SYN, ATK_SEQ_0 SYN, SVR_SEQ_0', ATK_SEQ_0 + 1->	
SYN-RECEIVED	<= SYN, ATK_SEQ_0 + 1, SVR_SEQ_0' + 1	
ESTABLISHED SVR_SEQ = SVR_SEQ_0' + 1 SVR_ACK = ATK_SEQ_0 + 1		

Рисунок 2: Схема атаки (пакеты атакующего хоста отмечены знаком =>)

На рисунке 2 не показан обмен неприемлемыми пакетами ACK. В результате выполнения описанных операций обе стороны соединения будут находиться в десинхронизированном состоянии ESTABLISHED.

SVR_TO_CLT_OFFSET = SVR_SEQ_0 - SVR_SEQ_0'
задан сервером.

CLT_TO_SVR_OFFSET = ATK_SEQ_0 - CLT_SEQ_0
задан атакующим.

Успех атаки связан с выбором значения CLT_TO_SVR_OFFSET. При неудачном значении клиент может воспринять пакеты от сервера с непредсказуемыми последствиями.

3.2 Десинхронизация бессмысленными данными

Этот способ десинхронизации основан на передаче атакующим большого количества данных клиенту и серверу. Эти данные не должны быть видны клиенту или серверу, но должны перевести сеанс TCP в состояние десинхронизации.

Приведенная ниже схема может использоваться для сеансов telnet:

- Атакующий прослушивает обмен данными между клиентом и сервером без вмешательства.
- В нужный момент атакующий передает серверу большое количество “пустой информации” (данные, которые не оказывают прямого влияния на работу сервера но приводят к изменению номеров подтверждений. Например, для сеансов telnet атакующий передает ATK_SVR_OFFSET байтов, содержащих последовательность IAC NOP IAC NOP... Каждая пара байтов IAC NOP[1] будет заставлять демон telnet считывать эту команду из потока данных без выполнения каких-либо дополнительных действий. В результате сервер установит
  SVR_ACK = CLT_SEQ + ATK_SVR_OFFSET
  что, естественно, приведет к десинхронизации.
- Такие же операции атакующий выполняет по отношению к клиенту.

Этот метод полезен только в тех случаях, когда сеанс позволяет передавать пустую информацию, которая не оказывает непосредственного влияния на принявший пакет хост. Время, когда следует передавать пустые данные, определить достаточно сложно, поэтому такой способ десинхронизации может давать побочные эффекты.

4. Примеры

Ниже показаны пакеты, собранные с помощью программы tcpdump [tcpdump] на интерфейсе Ethernet клиентского хоста. Строки комментариев начинаются символами ##.

Первый пример показывает нормальную организацию и завершение сессии telnet между клиентом (35.42.1.56) и сервером (198.108.3.13).

```
## клиент передает пакет SYN со стартовым номером 1496960000
11:07:14.934093 35.42.1.56.1374 > 198.108.3.13.23: S 1496960000:1496960000(0) win 4096
## сервер передает отклик со своим порядковым номером и флагом SYN
11:07:14.936345 198.108.3.13.23 > 35.42.1.56.1374: S 1402880000:1402880000(0) ack 1496960001 win 4096
## клиент подтверждает серверу пакет SYN
11:07:14.937068 35.42.1.56.1374 > 198.108.3.13.23: . 1496960001:1496960001(0) ack 1402880001 win 4096
## обе стороны находятся в состоянии ESTABLISHED
## клиент передает 6 байтов данных
11:07:15.021817 35.42.1.56.1374 > 198.108.3.13.23: P 1496960001:1496960007(6) ack 1402880001 win 4096
255 253 /C 255 251 /X

...

## корректное завершение соединения
11:07:18.111596 198.108.3.13.23 > 35.42.1.56.1374: F 1402880059:1402880059(0) ack 1496960025 win 4096
11:07:18.112304 35.42.1.56.1374 > 198.108.3.13.23: . 1496960025:1496960025(0) ack 1402880060 win 4096
11:07:18.130610 35.42.1.56.1374 > 198.108.3.13.23: F 1496960025:1496960025(0) ack 1402880060 win 4096
11:07:18.132935 198.108.3.13.23 > 35.42.1.56.1374: . 1402880060:1402880060(0) ack 1496960026 win 4095
```

В следующем примере рассматривается такая же сессия, но с участием атакующей стороны. Десинхронизация осуществляется на этапе организации соединения (параграф 3.1). Атакующий добавляет в поток данных команду ls;. Для идентификации пользователя на сервере применяется ключ skey. С точки зрения пользователя сессия имеет вид:

```
<lpj@homefries: 1> telnet 198.108.3.13
Trying 198.108.3.13 ...
```

[1] В протоколе telnet команда NOP означает отсутствие операции (не требует что-либо делать) и просто игнорируется сервером.

```
Connected to 198.108.3.13.
Escape character is '^'.
SunOS UNIX (_host)
login: lpj
s/key 70 cn33287
(s/key required)
Password:
Last login: Wed Nov 30 11:28:21 from homefries.merit.edu
SunOS Release 4.1.3_U1 (GENERIC) #2: Thu Jan 20 15:58:03 PST 1994
(lpj@_host: 1) pwd
Mail/           mbox            src/
elm*            resize*         traceroute*
/usr/users/lpj
(lpj@_host: 2) history
     1  13:18   ls ; pwd
     2  13:18   history
(lpj@_host: 3) logoutConnection closed by foreign host.
<lpj@homefries: 2>
```

Пользователь вводит единственную команду pwd и после этого просматривает стек команд, в котором присутствует команда ls. Вывод этой команды не фильтровался атакующим. Ниже показан протокол обмена пакетами TCP между клиентом и сервером. К сожалению, некоторые пакеты отсутствуют в протоколе, поскольку они были отброшены драйвером интерфейса Ethernet программы сбора пакетов. Показанный протокол не представляет собой список всех пакетов сессии, а состоит из нескольких наиболее важных фрагментов. Размер окна на атакующем хосте имеет нетипичные значения (400, 500, 1000) чтобы усложнить трассировку пакетов. Атакующий хост находится в сети 35.42.1 (три интервала от сервера) на пути между клиентом и сервером. Из соображений безопасности имена и адреса хостов вымышлены.

```
## клиент передает пакет SYN со стартовым номером 896896000
11:25:38.946119 35.42.1.146.1098 > 198.108.3.13.23: S 896896000:896896000(0) win 4096
## сервер передает отклик со своим порядковым номером (1544576000) и флагом SYN
11:25:38.948408 198.108.3.13.23 > 35.42.1.146.1098: S 1544576000:1544576000(0) ack 896896001 win 4096
## клиент подтверждает пакет SYN; оба хоста находятся в состоянии ESTABLISHED
11:25:38.948705 35.42.1.146.1098 > 198.108.3.13.23: . 896896001:896896001(0) ack 1544576001 win 4096
## клиент передает серверу данные
11:25:38.962069 35.42.1.146.1098 > 198.108.3.13.23: P 896896001:896896007(6) ack 1544576001 win 4096
255 253 /C 255 251 /X
## атакующий сбрасывает соединение на серверной стороне
11:25:39.015717 35.42.1.146.1098 > 198.108.3.13.23: R 896896101:896896101(0) win 0
## атакующий создает соединение заново со своим номером 601928704
11:25:39.019402 35.42.1.146.1098 > 198.108.3.13.23: S 601928704:601928704(0) win 500
## сервер передает отклик с новым порядковым номером (1544640000) и флагом SYN
11:25:39.022078 198.108.3.13.23 > 35.42.1.146.1098: S 1544640000:1544640000(0) ack 601928705 win 4096
## поскольку такой пакет неприемлем для клиента, он передает подтверждение с ожидаемым
## порядковым номером (1544576001)
11:25:39.022313 35.42.1.146.1098 > 198.108.3.13.23: . 896896007:896896007(0) ack 1544576001 win 4096
## повторная передача пакета SYN вызванная неприемлемым последним пакетом
11:25:39.023780 198.108.3.13.23 > 35.42.1.146.1098: S 1544640000:1544640000(0) ack 601928705 win 4096
## Цикл с бурей пакетов ACK
11:25:39.024009 35.42.1.146.1098 > 198.108.3.13.23: . 896896007:896896007(0) ack 1544576001 win 4096
11:25:39.025713 198.108.3.13.23 > 35.42.1.146.1098: S 1544640000:1544640000(0) ack 601928705 win 4096
11:25:39.026022 35.42.1.146.1098 > 198.108.3.13.23: . 896896007:896896007(0) ack 1544576001 win 4096

...

11:25:39.118789 198.108.3.13.23 > 35.42.1.146.1098: S 1544640000:1544640000(0) ack 601928705 win 4096
11:25:39.119102 35.42.1.146.1098 > 198.108.3.13.23: . 896896007:896896007(0) ack 1544576001 win 4096
11:25:39.120812 198.108.3.13.23 > 35.42.1.146.1098: S 1544640000:1544640000(0) ack 601928705 win 4096
11:25:39.121056 35.42.1.146.1098 > 198.108.3.13.23: . 896896007:896896007(0) ack 1544576001 win 4096
## В конце концов атакующий подтверждает серверу пакет SYN со своим новым порядковым номером
## (601928705). Данные в этом пакете уже были ранее переданы клиентом, но они не были получены сервером.
11:25:39.122371 35.42.1.146.1098 > 198.108.3.13.23: . 601928705:601928711(6) ack 1544640001 win 400
255 253 /C 255 251 /X
## Некий всплеск пакетов ACK
11:25:39.124254 198.108.3.13.23 > 35.42.1.146.1098: . 1544640001:1544640001(0) ack 601928711 win 4090
11:25:39.124631 35.42.1.146.1098 > 198.108.3.13.23: . 896896007:896896007(0) ack 1544576001 win 4096
11:25:39.126217 198.108.3.13.23 > 35.42.1.146.1098: . 1544640001:1544640001(0) ack 601928711 win 4090
11:25:39.126632 35.42.1.146.1098 > 198.108.3.13.23: . 896896007:896896007(0) ack 1544576001 win 4096

...

11:25:41.261885 35.42.1.146.1098 > 198.108.3.13.23: . 601928728:601928728(0) ack 1544640056 win 1000
## Повторная передача от клиента
11:25:41.422727 35.42.1.146.1098 > 198.108.3.13.23: P 896896018:896896024(6) ack 1544576056 win 4096
255 253 /A 255 252 /A
11:25:41.424108 198.108.3.13.23 > 35.42.1.146.1098: . 1544640059:1544640059(0) ack 601928728 win 4096

...

11:25:42.323262 35.42.1.146.1098 > 198.108.3.13.23: . 896896025:896896025(0) ack 1544576059 win 4096
11:25:42.324609 198.108.3.13.23 > 35.42.1.146.1098: . 1544640059:1544640059(0) ack 601928728 win 4096
## Второй символ идентификатора пользователя
11:25:42.325019 35.42.1.146.1098 > 198.108.3.13.23: P 896896025:896896026(1) ack 1544576059 win 4096
p
11:25:42.326313 198.108.3.13.23 > 35.42.1.146.1098: . 1544640059:1544640059(0) ack 601928728 win 4096

...

11:25:43.241191 35.42.1.146.1098 > 198.108.3.13.23: . 601928731:601928731(0) ack 1544640060 win 1000
## Повторная передача
11:25:43.261287 198.108.3.13.23 > 35.42.1.146.1098: P 1544640059:1544640061(2) ack 601928730 win 4096
l p
11:25:43.261598 35.42.1.146.1098 > 198.108.3.13.23: . 896896027:896896027(0) ack 1544576061 win 4096

...

11:25:43.294192 198.108.3.13.23 > 35.42.1.146.1098: . 1544640061:1544640061(0) ack 601928730 win 4096
11:25:43.922438 35.42.1.146.1098 > 198.108.3.13.23: P 896896026:896896029(3)  ack 1544576061 win 4096
j /M /@
11:25:43.923964 198.108.3.13.23 > 35.42.1.146.1098: . 1544640061:1544640061(0) ack 601928730 win 4096

...
```

```
11:25:43.957528 198.108.3.13.23 > 35.42.1.146.1098: . 1544640061:1544640061(0) ack 601928730 win 4096
## Атакующий переписал переданный сервером пакет с запросом
11:25:44.495629 198.108.3.13.23 > 35.42.1.146.1098: P 1544576064:1544576082(18) ack 896896029 win 1000
s / k e y   7 0   c n 3 3 2 8 7 /M /J
11:25:44.502533 198.108.3.13.23 > 35.42.1.146.1098: P 1544576082:1544576109(27) ack 896896029 win 1000
( s / k e y   r e q u i r e d ) /M /J P a s s w o r d :
11:25:44.522500 35.42.1.146.1098 > 198.108.3.13.23: . 896896029:896896029(0) ack 1544576109 win 4096

...

11:25:44.558320 198.108.3.13.23 > 35.42.1.146.1098: . 1544640109:1544640109(0) ack 601928733 win 4096
## Начало пароля skey, переданного пользователем (клиент)
11:25:57.356323 35.42.1.146.1098 > 198.108.3.13.23: P 896896029:896896030(1) ack 1544576109 win 4096
T
11:25:57.358220 198.108.3.13.23 > 35.42.1.146.1098: . 1544640109:1544640109(0) ack 601928733 win 4096

...

11:25:57.412103 198.108.3.13.23 > 35.42.1.146.1098: . 1544640109:1544640109(0) ack 601928733 win 4096
## Эхо начала пароля skey, переданное сервером
11:25:57.412456 35.42.1.146.1098 > 198.108.3.13.23: P 601928733:601928734(1) ack 1544640109 win 1000
T
11:25:57.412681 35.42.1.146.1098 > 198.108.3.13.23: . 896896030:896896030(0) ack 1544576109 win 4096

...

11:25:57.800953 198.108.3.13.23 > 35.42.1.146.1098: . 1544640109:1544640109(0) ack 601928734 win 4096
## Атакующий переписывает пакет с паролем skey
11:25:57.801254 35.42.1.146.1098 > 198.108.3.13.23: P 601928734:601928762(28) ack 1544640109 win 1000
A U T   S H I M   L O F T   V A S E   M O O   R I D /M /@
11:25:57.801486 35.42.1.146.1098 > 198.108.3.13.23: . 896896058:896896058(0) ack 1544576109 win 4096

...

11:25:58.358275 35.42.1.146.1098 > 198.108.3.13.23: . 896896058:896896058(0) ack 1544576109 win 4096
11:25:58.360109 198.108.3.13.23 > 35.42.1.146.1098: P 1544640263:1544640278(15) ack 601928762 win 4096
( l p j @ _ r a d b :   1 )
11:25:58.360418 35.42.1.146.1098 > 198.108.3.13.23: . 896896058:896896058(0) ack 1544576109 win 4096

...

11:26:00.919976 35.42.1.146.1098 > 198.108.3.13.23: . 896896058:896896058(0) ack 1544576278 win 4096
## Символ p введенной пользователем команды pwd
11:26:01.637187 35.42.1.146.1098 > 198.108.3.13.23: P 896896058:896896059(1) ack 1544576278 win 4096
p
11:26:01.638832 198.108.3.13.23 > 35.42.1.146.1098: . 1544640278:1544640278(0) ack 601928762 win 4096

...

11:26:03.183200 35.42.1.146.1098 > 198.108.3.13.23: . 896896063:896896063(0) ack 1544576280 win 4096
11:26:03.921272 35.42.1.146.1098 > 198.108.3.13.23: P 896896060:896896063(3) ack 1544576280 win 4096
d /M /@
11:26:03.922886 198.108.3.13.23 > 35.42.1.146.1098: . 1544640283:1544640283(0) ack 601928767 win 4096

...

11:26:04.339186 35.42.1.146.1098 > 198.108.3.13.23: . 896896063:896896063(0) ack 1544576280 win 4096
11:26:04.340635 198.108.3.13.23 > 35.42.1.146.1098: P 1544640288:1544640307(19) ack 601928770 win 4096
M a i l / /I /I m b o x /I /I s r c / /M /J
11:26:04.342872 198.108.3.13.23 > 35.42.1.146.1098: P 1544640307:1544640335(28) ack 601928770 win 4096
e l m * /I /I r e s i z e * /I /I t r a c e r o u t e * /M /J
11:26:04.345480 35.42.1.146.1098 > 198.108.3.13.23: . 896896063:896896063(0) ack 1544576280 win 4096
11:26:04.346791 198.108.3.13.23 > 35.42.1.146.1098: P 1544640335:1544640351(16) ack 601928770 win 4096
/ u s r / u s e r s / l p j /M /J
11:26:04.347094 35.42.1.146.1098 > 198.108.3.13.23: . 896896063:896896063(0) ack 1544576280 win 4096
11:26:04.348402 198.108.3.13.23 > 35.42.1.146.1098: P 1544640351:1544640366(15) ack 601928770 win 4096
( l p j @ _ r a d b :   2 )
11:26:04.378571 35.42.1.146.1098 > 198.108.3.13.23: . 896896063:896896063(0) ack 1544576280 win 4096

...

11:26:09.791045 35.42.1.146.1098 > 198.108.3.13.23: P 601928773:601928775(2) ack 1544640369 win 1000
t o
11:26:09.794653 198.108.3.13.23 > 35.42.1.146.1098: P 1544640369:1544640371(2) ack 601928775 win 4096
t o
11:26:09.794885 35.42.1.146.1098 > 198.108.3.13.23: . 896896068:896896068(0) ack 1544576366 win 4096

...

11:26:12.420397 35.42.1.146.1098 > 198.108.3.13.23: P 896896068:896896072(4) ack 1544576368 win 4096
r y /M /@
11:26:12.422242 198.108.3.13.23 > 35.42.1.146.1098: . 1544640371:1544640371(0) ack 601928775 win 4096

...

11:26:12.440765 35.42.1.146.1098 > 198.108.3.13.23: . 896896072:896896072(0) ack 1544576368 win 4096
## Символы ry введенной пользователем команды history
11:26:16.420287 35.42.1.146.1098 > 198.108.3.13.23: P 896896068:896896072(4) ack 1544576368 win 4096
r y /M /@
11:26:16.421801 198.108.3.13.23 > 35.42.1.146.1098: . 1544640371:1544640371(0) ack 601928775 win 4096

...

11:26:16.483943 35.42.1.146.1098 > 198.108.3.13.23: . 896896072:896896072(0) ack 1544576368 win 4096
## Тот же пакет, переписанный атакующим
11:26:16.505773 35.42.1.146.1098 > 198.108.3.13.23: P 601928775:601928779(4) ack 1544640371 win 1000
r y /M /@
## Отклик сервера на команду history; отметим, что перед pwd присутствует команда ls ;
11:26:16.514225 198.108.3.13.23 > 35.42.1.146.1098: P 1544640371:1544640437(66) ack 601928779 win 4096
r y /M /@ /M /J           1 /I 1 1 : 2 8 /I l s ;   p w
        d /M /J           2 /I 1 1 : 2 8 /I /@ /@ /@ L /@ /@ /@ T . 220 167 168 /@ /G
       /@ /@ /@ /X /@ /H 137 148 /@ /@
11:26:16.514465 35.42.1.146.1098 > 198.108.3.13.23: . 896896072:896896072(0) ack 1544576368 win 4096

...
```

```
11:26:16.575344 35.42.1.146.1098 > 198.108.3.13.23: . 896896072:896896072(0) ack 1544576368 win 4096
## Тот же пакет, переписанный атакующим
11:26:16.577183 198.108.3.13.23 > 35.42.1.146.1098: P 1544576368:1544576434(66) ack 896896072 win 1000
r y /M /@ /M /J           1 /I 1 1 : 2 8 /I l s ;   p w
        d /M /J           2 /I 1 1 : 2 8 /I /@ /@ /@ L /@ /@ /@ T . 220 167 168 /@ /H / @ /@ /@
        /X /@ /H 137 148 /@ /@
11:26:16.577490 198.108.3.13.23 > 35.42.1.146.1098: . 1544640437:1544640437(0) ack 601928779 win 4096

...

## пользователь отключается
11:26:20.236907 35.42.1.146.1098 > 198.108.3.13.23: P 601928781:601928782(1) ack 1544640437 win 1000
g
11:26:20.247288 198.108.3.13.23 > 35.42.1.146.1098: . 1544576438:1544576438(0) ack 896896074 win 1000
11:26:20.253500 198.108.3.13.23 > 35.42.1.146.1098: P 1544576435:1544576436(1) ack 896896074 win 1000 o
11:26:20.287513 198.108.3.13.23 > 35.42.1.146.1098: P 1544640439:1544640440(1) ack 601928782 win 4096 g
11:26:20.287942 35.42.1.146.1098 > 198.108.3.13.23: P 896896075:896896076(1) ack 1544576436 win 4096 o
11:26:20.289312 198.108.3.13.23 > 35.42.1.146.1098: . 1544640440:1544640440(0) ack 601928782 win 4096
11:26:20.289620 35.42.1.146.1098 > 198.108.3.13.23: . 896896076:896896076(0) ack 1544576436 win 4096
```

Большая часть пакетов с флагом ACK, но без данных является подтверждениями неприемлемых пакетов. Значительная часть повторов связана с загрузкой сети, вызванной бурей пакетов ACK, которую создал атакующий. Полный протокол, включающий все пакеты ACK, содержал около 3000 строк. Большинство этих пакетов не показано здесь. Собранные во время проверки данные показывают, что на один пакет с реальными данными приходится от 10 до 300 пустых пакетов ACK.

5. Детектирование атак и побочные эффекты

Некоторые недостатки описанной атаки можно использовать для детектирования. Здесь кратко описаны три способа обнаружения атаки, но вы можете найти и другие способы.

Обнаружение десинхронизации.

Сравнив порядковые номера на обеих сторонах соединения, можно определить состояние десинхронизации. Этот метод пригоден для тех случаев, когда можно надеяться, что данные передаются через поток TCP, который не изменяет атакующий.

Таблица 1: Доля (%) пакетов ACK при нормальной работе (без атаки).

	Локальный сегмент Ethernet	Транзитный сегмент Ethernet
Всего TCP	80-100 (60-80)	1400 (87)
Всего ACK	25-75 (25-45)	500 (35)
Всего Telnet	10-20 (10-25)	140 (10)
Всего Telnet ACK	5-10 (45-55)	45 (33)

Детектирование ACK-бури.

Статистика трафика TCP, собранная в локальном сегменте при нормальной работе, показывает, что среднее число пакетов ACK без данных для сеансов telnet составляет около 45%. На более загруженном транзитном сегменте Ethernet среднее число таких пакетов составило 33% (см. таблицу 1)

Общее число пакетов TCP, как и число пакетов ACK или telnet достаточно сильно меняются и в таблице показаны диапазоны изменения. Доля пакетов ACK для протокола telnet весьма стабильна и составляет около 45%. это можно объяснить тем, что сессии telnet являются интерактивными и для каждого введенного пользователем символа генерируется эхо-символ и подтверждение. Общий объем передаваемых данных весьма невелик и каждый пакет обычно содержит один символ или одну строку текста.

Транзитный сегмент сети весьма загружен и некоторые пакеты по этой причине могли быть отброшены собиравшим данные хостом.

Во время атаки картина несколько изменилась. В таблице 2 показаны результаты для двух сессий. Сбор данных осуществлялся только в локальном сегменте Ethernet.

Локальным соединением назван случай, когда сервер находится от клиента на расстоянии нескольких хопов[1] и время кругового обхода (RTD) составляет приблизительно 3 мсек. Для удаленного соединения время RTD составляло около 40 мсек, а число хопов - 9. В первом случае атака была отчетливо видна. Даже при достаточно больших флуктуациях доля пакетов ACK приближалась к 100%, т .е. Почти весь трафик состоял из подтверждений.

Во втором случае атака проявлялась не столь отчетливо. При сравнении данных для удаленного соединения с колонкой локального сегмента в таблице 1 можно видеть, что доля пакетов ACK возросла незначительно. Это можно объяснить большим значением RTD, которое снижает скорость передачи пакетов ACK. В сети терялось от 5% до 10% пакетов, что помогало разорвать цикл генерации пакетов ACK.

Таблица 2. Доля (%) пакетов ACK во время атаки.

	Локальное соединение	Удаленное соединение
Всего Telnet	80-400 (60-85)	30-40 (30-35)
Всего Telnet ACK	75-400 (90-99)	20-25 (60-65)

Возрастает уровень потери пакетов и повторов передачи для отдельной сессии. Этот рост не всегда просто заметить, но протоколирование пакетов сессии обычно показывает увеличение числа теряемых пакетов и повторных передач. Следовательно, должно увеличиваться время отклика сервера. Рост числа теряемых пакетов можно объяснить несколькими причинами:

- дополнительная загрузка сети во время ACK-бури;

[1] Реально было 4 интервала.

- отбрасывание части пакетов используемой атакующим программой захвата.

Неожиданный сброс соединений.

Описанные ниже факты не исследованы полностью, поскольку программа атаки была разработана в основном для концептуальной проверки метода и задачи обеспечить скрытность атаки просто не ставилось. Очевидно, что для реальных атак будут разработаны более изощренные программы. Если протокол атаки выполняется программой некорректно, пользователь может столкнуться со сбросом соединения на начальном этапе его организации. Потеря пакетов RST или SYN от атакующего хоста может привести к возникновению неопределенного состояния на сервере (обычно CLOSED или SYN-RECEIVED) и сделать приемлемы пакеты от клиента. Около 10% процентов атак не привели к успеху, закончившись разрывом соединения (слишком заметно) или неудачей при попытке десинхронизации (атакующий не может перенаправить поток).

Отметим также некоторые побочные эффекты и особенности протокола TCP.

- Реализация TCP.
  Описанный процесс десинхронизации TCP не удается выполнить для некоторых реализаций протокола. В соответствии с [rfc793] пакеты RST не подтверждаются и должны приводить к удалению структуры TCB. Некоторые реализации TCP в зависимости от состояния могут в ответ на пакет RST также передать пакет RST. Когда атакующий передает пакет RST серверу, переданный последним ответный пакет RST в адрес клиента будет закрывать соединение и завершать сессию. В таких случаях для атаки требуются иные механизмы десинхронизации, которые не будут приводить к разрыву соединения.
- При проверке клиент и атакующий хост всегда находились в одном сегменте Ethernet. Это осложняет атаку по причине увеличения уровня загрузки сегмента, которая увеличивает частоту коллизий и буфер используемого для организации атаки анализатора может просто переполняться.
- Можно подумать, что для организации атаки достаточно прослушивания соединения и передачи серверу неких данных без создания десинхронизованного состояния и пересылки пакетов TCP. Такая атака может привести к успеху, но атакующий хост будет очень быстро обнаружен.

6. Предотвращение атак

Единственным известным автору способом предотвращения описанных здесь атак на сессии telnet является использование схемы шифрования Kerberos (уровень приложений) или шифрование на уровне TCP [TCPcrypt]. Шифрование потока данных предотвратит изменение потока информации атакующей стороной. Для обеспечения безопасности электронной почты могут также использоваться цифровые подписи [pgp].

7. Сравнение с атакой Морриса

Описанная в статье Морриса (Morris) [morris85] атака предполагает способность атакующего предсказать стартовый порядковый номер который сервер будет использовать для следующего соединения (SVR_SEQ_0 в данной статье) и использование доверительных отношений, при которых отдельные хосты могут выполнять на сервере те или иные команды без какой-либо дополнительной идентификации.

Атакующий открывает сессию, передавая серверу пакет SYN с адресом доверенного хоста в поле отправителя. Сервер подтверждает запрос SYN пакетом SYN/ACK с SEG_SEQ = SVR_SEQ_0. Атакующий подтверждает этот пакет, используя предсказанный порядковый номер SVR_SEQ_0. Атакующему не нужно собирать и анализировать пакеты, поскольку он может предсказать порядковый номер SVR_SEQ_0. Этому тип атак присущи две особенности:

- Клиент, под которого маскируется атакующий хост, будет получать от сервера пакет SYN/ACK и генерировать в результате пакет RST. Моррис предположил, что генерацию пакета RST можно предотвратить с помощью атаки на клиентский хост, при которой на последнем возникнет переполнение очереди TCP и пакет SYN/ACK будет отброшен.
- Атакующий не получает пакетов от сервера. Однако атакующий может передавать серверу данные, что во многих случаях вполне достаточно для успешной атаки.

Описанная в данной статье атака имеет 4 принципиальных отличия от атаки Морриса:

- Для атаки Морриса требуются доверительные отношения между хостами.
- Описанная в данной статье атака обеспечивает организацию полнодуплексного соединения TCP, через которое атакующий может передавать и принимать данные.
- Описанная здесь атака использует средство сбора кадров Ethernet для предсказания или определения SVR_SEQ_0.
- Данная атака применима к хостам любого типа (не только Unix).

Атаку Морриса можно легко усовершенствовать, добавив возможности описанной выше атаки.

- Использовать программу захвата пакетов для получения стартового порядкового номера. После этого атака на сервер становится возможной без ожидания попытки соединения клиента с сервером.
- В предположении, что клиент не будет передавать серверу пакет RST (например, в силу неработоспособности этого клиента) атакующий может организовать с сервером полнодуплексное соединение TCP, передавая данные от имени клиента. Естественно, атакующему придется преодолеть систему идентификации, но если эта система использует доверительные отношения (как NFS или rlogin), атакующий может получить полный доступ к службе сервера.

Стивен Беллоуин (Steven M. Bellovin) в статье [bellovin89] показал возможность использования пакетов ICMP для блокирования одной из сторон соединения. В этом случае атакующий получает полный контроль над сессией (это называют TCP session hijacking – перехват сессии TCP), но атаку очень легко обнаружить.

8. Заключение

Несмотря на простоту обнаружения описанной атаки в локальной сети, она может быть весьма эффективна при использовании через распределенную сеть с невысокой скоростью и большими задержками (WAN). Для организации такой атаки достаточно средств, используемых обычно при пассивных атаках в сети Internet.

Опасность описанной здесь атаки увеличивается в связи с тем, что она не заметна для пользователя. Взлом хостов Internet становится все более распространенным явлением, и скрытность атаки существенно затрудняет ее обнаружение.

Хотя всеобщее внимание занимает новый протокол IPv6, идущий на замену IPv4, рост числа атак и необходимость обеспечения безопасности систем требует разработки и использования безопасного протокола транспортного уровня для сети Internet. Должны быть обеспечены опции, позволяющие передавать подписанные и зашифрованные данные в целях обеспечения конфиденциальности. Поскольку цифровые подписи играют важную роль в обеспечении надежности доставки, такие сигнатуры можно использовать взамен принятых сегодня контрольных сумм TCP.

В этой статье не предпринималось попыток проанализировать все возможные варианты активных атак с использованием анализаторов. Важнее было предупредить пользователей ключей s/key или системы Kerberos об опасности прослушивания трафика Ethernet. В статье содержатся некоторые идеи, которые могут послужить отправной точкой для дальнейших исследований. Представленный здесь метод был успешно использован для организации атаки минимальными средствами.

Литература

[Bellovin89] "Security Problems in the TCP/IP Protocol Suite", Bellovin, S., Computer Communications Review, April 1989.

[Kerberos] "Kerberos: An Authentication Service for Open Network Systems", Steiner, J., Neuman, C., Schiller, J., USENIX Conference Proceeding, Dallas, Texas, February 1989.

[Morris85] "A Weakness in the 4.2BSD UNIX TCP/IP Software", Morris, R., Computing Science Technical Report No 117, ATT Bell Laboratories, Murray Hill, New Jersey, 1985.

[PGP] Pretty Good Privacy Version 2.6.1, Philip Zimmermann, August 1994.

[RFC 793] RFC 793, “Transmission Control Protocol”, September 1981, J. Postel.

[RFC 854] Request For Comment 854, “Telnet Protocol Specification”, May 1983, J. Postel, J. Reynolds

[SKEY] "The S/Key One-time Password System", Haller, N., Proceeding of the Symposium on Network Distributed Systems, Security, Internet Society, San Diego, CA, February 1994.

[TCPcrypt] "Public Key Encryption Support for TCP", Joncheray, L., Work in progress, May 1995.

[TCPDUMP] tcpdump(8) Version 2.2.1, Van Jacobson, Craig Leres, Steven Berkeley, University of California, Berkeley, CA.

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru