Network Working Group
Request for Comments: 4251
Category: Standards Track

T. Ylonen
SSH Communications Security Corp
C. Lonvick, Ed.
Cisco Systems, Inc.
January 2006

Архитектура протокола SSH

The Secure Shell (SSH) Protocol Architecture

Статус документа

В этом документе содержится спецификация протокола, предложенного сообществу Internet. Документ служит приглашением к дискуссии в целях развития и совершенствования протокола. Текущее состояние стандартизации протокола вы можете узнать из документа Internet Official Protocol Standards (STD 1). Документ может распространяться без ограничений.

Авторские права

Аннотация

Протокол SSH[1] используется для организации безопасного входа в удалённую систему (login) и организации иных безопасных служб через сети, не обеспечивающие защиты. В данном документе описана архитектура протокола SSH, а также нотация и терминология, используемые в документах, посвящённых протоколу SSH. Описывается также алгоритм именования SSH, поддерживающий локализованные расширения. Протокол SSH состоит из трёх основных компонент. Протокол транспортного уровня (Transport Layer Protocol) обеспечивает аутентификацию серверов, конфиденциальность и целостность с высоким уровнем защищенности. Протокол аутентификации пользователей (User Authentication Protocol) используется на серверах для проверки полномочий клиентов. Протокол соединений (Connection Protocol) обеспечивает мультиплексирование шифрованного туннеля в несколько логических каналов. Детальные описания каждого из этих протоколов содержатся в отдельных документах.

Оглавление

[1] Secure Shell - защищённая командная оболочка.

1. Введение

Протокол SSH используется для организации безопасного входа в удалённую систему (login) и организации иных безопасных служб через сети, не обеспечивающие защиты. Протокол включает три основных компоненты.

- Протокол транспортного уровня [SSH-TRANS] обеспечивает аутентификацию серверов, конфиденциальность и целостность. Этот протокол может также обеспечивать сжатие информации. Транспортный уровень работает в основном с использованием соединений TCP/IP, но может быть реализован и на базе иных потоков данных с гарантированной доставкой.

- Протокол аутентификации пользователей [SSH-USERAUTH] используется на серверах для проверки полномочий клиентов. Этот протокол работает на основе протокола транспортного уровня.

- Протокол соединений [SSH-CONNECT] обеспечивает мультиплексирование шифрованного туннеля в несколько логических каналов и работает поверх протокола аутентификации пользователей.

Клиент передаёт один запрос на обслуживание в процессе организации защищённого соединения на транспортном уровне. Другой запрос на обслуживание передаётся после успешной проверки полномочий клиента. Такое решение обеспечивает возможность создания новых протоколов и их совместного использования с перечисленными выше протоколами.

Протокол соединений обеспечивает каналы, которые могут использоваться для решения целого ряда задач. Обеспечиваются стандартные методы для организации защищённых shell-сессий и перенаправления (туннелирования) произвольных портов TCP/IP и соединений X11.

2. Разработчики

Основными разработчиками этого комплекта документов являются: Tatu Ylonen, Tero Kivinen, Timo J. Rinne, Sami Lehtinen (все из SSH Communications Security Corp) и Markku-Juhani O. Saarinen (университет Jyvaskyla). Darren Moffat был редактором этого комплекта документов и внёс важный вклад в работу.

За годы подготовки этого документа множество людей внесло свой вклад. В их число входят: Mats Andersson, Ben Harris, Bill Sommerfeld, Brent McClure, Niels Moller, Damien Miller, Derek Fawcus, Frank Cusack, Heikki Nousiainen, Jakob Schlyter, Jeff Van Dyke, Jeffrey Altman, Jeffrey Hutzelman, Jon Bright, Joseph Galbraith, Ken Hornstein, Markus Friedl, Martin Forssen, Nicolas Williams, Niels Provos, Perry Metzger, Peter Gutmann, Simon Josefsson, Simon Tatham, Wei Dai, Denis Bider, der Mouse и Tadayoshi Kohno. Указанные в списке люди могли не участвовать в написании данного документа, но они внесли свой вклад в его подготовку.

3. Используемые в документе соглашения

Во всех документах, связанных с протоколом SSH, следует использовать ключевые слова: **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не нужно** (SHALL NOT), **следует** (SHOULD), **не следует** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) для описания уровня требования. Интерпретация этих слов описана в [RFC2119].

Ключевые слова **приватное использование** (PRIVATE USE), **иерархическое выделение** (HIERARCHICAL ALLOCATION), **выделение в соответствии с порядком запросов** (FIRST COME FIRST SERVED), **экспертное рассмотрение** (EXPERT REVIEW), **требуется спецификация** (SPECIFICATION REQUIRED), **одобрение IESG** (IESG APPROVAL), **согласование с IETF** (IETF CONSENSUS), **стандартизация** (STANDARDS ACTION) в данном документе при их использовании в контексте распределения пространства имён интерпретируются в соответствии с [RFC2434].

В данном наборе документов определяются поля протокола и возможные значения этих полей. Поля будут определяться вместе с протокольными сообщениями. Например, поле SSH_MSG_CHANNEL_DATA определяется следующим образом:

```
byte      SSH_MSG_CHANNEL_DATA
uint32    recipient channel (канал получателя)
string    data (данные)
```

В данном документе поля протокола будут указываться в одинарных кавычках, а значения полей - в двойных. В приведённом выше примере поле 'data' может содержать значения "foo" и "bar".

4. Архитектура

4.1. Ключи хостов

Каждому серверному хосту **следует** иметь ключ (host key). Хосты **могут** иметь множество ключей, созданных с использованием различных алгоритмов. **Возможно** использование одного ключа множеством хостов. Если хост имеет ключи, он **должен** обеспечивать по крайней мере один ключ, который использует каждый из **требуемых** алгоритмов открытых ключей (DSS [FIPS-186-2]).

Ключ сервера используется в процессе обмена ключами для подтверждения того, что клиент реально связывается с нужным сервером. Для того, чтобы такая проверка стала возможной, клиент должен заранее знать открытый ключ сервера.

Могут использоваться две модели поддержки ключей:

- Клиент поддерживает локальную базу данных, в которой содержатся имена всех хостов (указанные пользователем) и соответствующие открытые ключи. Этот метод не требует создания инфраструктуры централизованного управления или сторонней координации. Недостатком метода является трудоёмкость поддержки ассоциаций между ключами и именами хостов.

- Ассоциации между ключами и хостами сертифицируются CA[1]. Клиент знает только ключ корневого CA и может проверить все ключи хостов, сертифицированные приемлемыми CA.

Второй вариант упрощает задачу поддержки, поскольку клиенту достаточно хранить единственный ключ CA. С другой стороны, в этом варианте требуется централизованная сертификация каждого хоста прежде, чем станет возможной процедура проверки полномочий. Кроме того требуется обеспечить высокий уровень доверия для централизованной инфраструктуры сертификации.

Протокол позволяет отключить проверку ассоциации «сервер - ключ» при первом подключении к хосту. Это даёт возможность организовать соединение с хостом до получения от него ключа или сертификации хоста. При таком соединении обеспечивается защита от пассивного прослушивания, но существует уязвимость для активного перехвата с участием человека[2]. Реализациям в общем случае **не следует** разрешать такие соединения по умолчанию, поскольку они могут снижать уровень безопасности. Однако по причине отсутствия в сети Internet широко доступной структуры обмена ключами на момент написания документа эта опция может оказаться весьма полезной в переходный период, пока такая инфраструктура не будет создана, и обеспечит значительно более высокий уровень безопасности, нежели более старые решения (например, telnet [RFC0854] и rlogin [RFC1282]).

Реализациям протокола **следует** предпринимать разумные меры по проверке ключей хостов. Примером возможной стратегии может служить принятие ключа без проверки только при первом соединении с хостом, сохранение полученного ключа в локальной базе данных и его использование при каждом последующем подключении к данному хосту.

Реализации **могут** обеспечивать дополнительные возможности по проверке корректности ключей хостов (например, использование шестнадцатеричных «отпечатков», полученных хэшированием открытого ключа с помощью алгоритма SHA-1 [FIPS-180-2]). Такие отпечатки можно легко проверить по телефону или с использованием другого коммуникационного канала.

Всем реализациям **следует** обеспечивать опцию для запрета принятия непроверенных ключей хостов.

Члены рабочей группы полагают, что простота использования играет важную роль при выборе безопасных решений конечными пользователями и уровень безопасности не станет выше, если не будут применяться новые решения. Таким образом, можно предположить, что обеспечение опции, позволяющей отказаться от проверки ключа, будет повышать общий уровень безопасности в сети Internet, несмотря на локальное снижение уровня безопасности протокола в тех случаях, когда проверка отключена.

4.2. Возможности расширения

Мы полагаем, что протокол будет использоваться достаточно долго и некоторые организации захотят использовать свои методы шифрования, аутентификации и обмена ключами. Централизованная регистрация всех расширений весьма обременительна, особенно для экспериментальных или засекреченных вариантов. С другой стороны, отсутствие централизованной регистрации будет приводить к конфликтам с идентификацией методов, осложняющим взаимодействие реализаций протокола.

Мы выбрали вариант идентификации алгоритмов, методов, форматов и расширений протокола с помощью текстовых имён, использующих определённый формат. Имена DNS используются для создания локальных пространств имён, где могут определяться экспериментальные или засекреченные расширения без возникновения конфликтов с другими реализациями.

Одной из задач является обеспечение максимальной простоты базового протокола и использование минимального числа алгоритмов. Однако все реализации **должны** поддерживать минимальный набор алгоритмов для обеспечения взаимодействия (это не означает, что локальная политика на каждом хосте должна обеспечивать поддержку этих алгоритмов). Обязательные алгоритмы указываются в соответствующих протокольных документах.

Дополнительные алгоритмы, методы, форматы и протоколы расширения могут определяться в отдельных документах. Дополнительная информация по этому вопросу содержится в разделе 6 (Алгоритм и метод именования).

4.3. Политика

Протокол допускает полное согласование алгоритмов и форматов шифрования, обеспечения целостности, обмена ключами, сжатия данных и открытых ключей. Алгоритмы шифрования, обеспечения целостности, сжатия и открытых ключей могут отличаться для каждого из направлений.

Приведённые ниже правила **следует** использовать в механизмах настройки конфигурационных параметров каждой реализации:

- Алгоритмы шифрования, обеспечения целостности и сжатия данных задаются раздельно для каждого направления. Политика **должна** задавать предпочтительный алгоритм (например, первый алгоритм из списка для каждой категории).

- Алгоритм открытых ключей и метод обмена ключами, используемые для аутентификации хоста. Существование доверенных ключей хостов для различных алгоритмов открытых ключей также должно приниматься во внимание.

- Методы аутентификации, требуемые сервером от каждого пользователя. Политика сервера **может** требовать несколько методов аутентификации для некоторых или всех пользователей. Набор обязательных алгоритмов **может** зависеть от местонахождения пользователя, пытающегося получить доступ.

- Операции, которые пользователю дозволено выполнять с применением протокола соединений. Некоторые аспекты политики могут быть связаны с безопасностью - например, **не следует** позволять серверу инициировать сессии или выполнять команды на клиентской машине и **недопустимо** разрешать соединения с агентом аутентификации, если не запрошено перенаправление таких соединений. Иные аспекты (такие, как возможность и назначение перенаправления портов TCP/IP) определяются локальной политикой. Многие из таких аспектов могут быть связаны

[1]Certification authority - удостоверяющий центр.
[2]Active man-in-the-middle attack.

с прохождением или обходом межсетевых экранов и, в той или иной степени, связаны с локальной политикой безопасности.

4.4. Параметры безопасности

Основной задачей протокола SSH является повышение уровня безопасности в Internet. Протокол пытается сделать это с обеспечением максимальной простоты пусть даже в ущерб защищенности.

- все алгоритмы шифрования, обеспечения целостности и открытых ключей относятся к числу известных и проверенных;
- все алгоритмы используются с криптографически обоснованным размером ключей, который позволяет надеяться на обеспечение защиты от самых мощных криптоаналитических атак в течение десятилетий;
- все алгоритмы согласуются и в тех случаях, когда тот или иной алгоритм не поддерживается, обеспечивается простой переход к использованию другого алгоритма без изменения базового протокола.

Будут предприниматься специальные меры для того, чтобы упростить широкое и быстрое развёртывание протокола. Частным случаем таких мер является возможность работы протокола без проверки ключа хоста при первом подключении, однако использовать эту меру **не рекомендуется**. Предполагается, что принимаемые меры помогут значительно расширить использование протокола в короткие сроки, пока не будет широко развёрнута инфраструктура обмена открытыми ключами в сети Internet.

4.5. Локализация и поддержка различных наборов символов

В большинстве случаев протоколы SSH не будут непосредственно передавать текст, отображаемый пользователю. Однако в некоторых случаях такие данные могут передаваться. В тех случаях, когда это применимо, набор символов для данных **должен** указываться явно. В большинстве случаев используется кодировка ISO-10646 UTF-8 [RFC3629]. В тех случаях, когда это применимо, обеспечивается также поле для тега языка [RFC3066].

Достаточно сложным является вопрос выбора набора символов для интерактивных сессий. Здесь нет очевидного решения, поскольку разные приложения могут отображать данные в различных форматах. У клиентов также могут быть установлены разнотипные эмуляторы терминалов и используемый набор символов будет определяться этим эмулятором. В таких случаях нет возможности прямого задания набора символов или кодировки данных для терминальных сессий. Однако тип эмулятора терминала (например, vt100) передаётся удалённому сайту и неявно задает набор символов и кодировку. Приложения обычно используют тип терминала для определения применяемого набора символов с помощью тех или иных дополнительных средств. Эмулятор терминала может также поддерживать выбор используемого по умолчанию набора символов. В любом случае выбор набора символов для терминальной сессии рассматривается, прежде всего, как локальная задача клиента.

Внутренние имена, используемые для идентификации алгоритмов и протоколов, обычно не отображаются пользователю и должны содержать символы набора US-ASCII.

Требования к именам пользователей для сервера и клиентов определяются возможностями сервера. Однако в некоторых случаях эти имена могут отображаться в журнальных файлах, отчётах и т. п. В именах **должен** использоваться набор символов ISO 10646 UTF-8, но в некоторых случаях может потребоваться иная кодировка. Сервер сам определяет способ отображения имён пользователей в поддерживаемые сервером имена. **Рекомендуется** использовать прямое сравнение битов.

В целях локализации для протокола предприняты попытки минимизации числа передаваемых текстовых сообщений. Сохранённые для протокола сообщения этого типа обычно связаны с ошибками, отладочной информацией и некоторыми задаваемыми извне данными. Для отображаемой информации **следует** обеспечить возможность выбора локализованных сообщений взамен сообщений, передаваемых протоколом с использованием цифровых кодов. Остальные сообщения **следует** делать настраиваемыми.

5. Представление типов данных, используемых в протоколах SSH

byte

Байт представляет собой произвольное 8-битовое значение (октет). Данные фиксированного размера в некоторых случаях представляются как массивы байтов byte[n], где n показывает число байтов в массиве.

boolean

Логические значения сохраняются в одном бите. Значение 0 представляет логическое значение FALSE (ложь), а 1 - логическое значение TRUE (истина). Любые, отличные от 0 значения **должны** интерпретироваться как TRUE, однако в приложениях **недопустимо** использование для логических переменных каких-либо значений, кроме 0 и 1.

uint32

32-битовое целое число без знака. Переменные этого типа сохраняются в четырёх байтах с уменьшением значимости (сетевой порядок). Например, значение 699921578 (0x29b7f4aa) будет иметь байтовое представление 29 b7 f4 aa.

uint64

64-битовое целое число без знака. Переменные этого типа сохраняются в восьми байтах с уменьшением значимости (сетевой порядок).

string

Двоичная строка произвольной длины. Строки могут содержать любые бинарные данные, включая null-символ и 8-битовые символы. Строки хранятся в виде переменной типа uint32, указывающей размер (число байтов) и последовательности (возможно нулевой длины) байтов, содержащих отдельные символы строки. Завершающих 0-символов для строк не используется.

Строки также служат для хранения текста. В этом случае для внутренних имён используется набор символов US-ASCII, а для текста, отображаемого пользователю, - набор символов ISO-10646 UTF-8. Завершающий null-символ в общем случае **не следует** включать в строку. Например, строка US-ASCII «testing» будет представлена, как 00 00 00 07 t e s t i n g. Отображение UTF-8 не изменяет представления символов набора US-ASCII.

mpint

Представляет целые числа в формате дополнения до 2, сохраняемые как строки байтов (старший байт сначала). Отрицательные значения задаются единицей в старшем бите первого байта строки данных. Если старший бит имеет значение 0 (положительное число), все остальные биты первого октета также должны иметь значение 0 (собственно число начинается со второго байта). **Недопустимо** использование значений с незначимыми старшими байтами, установленным в 0 или 255. Нулевое значение **должно** сохраняться как строка нулевой длины.

По соглашению число, используемое в расчётах по модулю n (Z_n), **следует** представлять значением из диапазона 0 <= x < n.

Пример:

```
значение (hex)      представление (hex)
--------------      -------------------
0                   00 00 00 00
9a378f9b2e332a7     00 00 00 08 09 a3 78 f9 b2 e3 32 a7
80                  00 00 00 02 00 80
-1234               00 00 00 02 ed cc
-deadbeef           00 00 00 05 ff 21 52 41 11
```

name-list

Строка, содержащая список разделённых запятыми имён. Такие списки представляются значением типа uint32, показывающим длину списка (число байтов), за которым следует список (возможно пустой) разделённых запятыми (,) имён. Имена в списке **должны** иметь отличную от 0 длину, **недопустимо** включать в имена символ запятой (,). Поскольку этот тип является списком имён, все элементы списка **должны** быть представлены набором символов US-ASCII. В зависимости от контекста к именам в списке могут предъявляться дополнительные требования. Например, имена в списке могут быть корректными идентификаторами алгоритмов (см. раздел 6) или тегами языков [RFC3066]. Порядок имён в списке может иметь значение, но это не обязательно и зависит от контекста использования списка. **Недопустимо** использование завершающих null-символов ни в отдельных именах, ни в списке в целом.

Пример:

```
значение                  представление (hex)
--------                  -------------------
(), пустой список имён    00 00 00 00
("zlib")                  00 00 00 04 7a 6c 69 62
("zlib,none")             00 00 00 09 7a 6c 69 62 2c 6e 6f 6e 65
```

6. Имена алгоритмов и методов

Протоколы SSH идентифицируют конкретные алгоритмы и методы хэширования, шифрования, обеспечения целостности, сжатия и обмена ключами по именам. Существуют некоторые стандартные алгоритмы и методы, которые **должны** поддерживаться всеми реализациями. Существуют также алгоритмы и методы, которые включены в спецификацию протокола, но являются **необязательными**. Кроме того следует ожидать, что некоторые организации пожелают использовать свои алгоритмы и методы.

В этом протоколе все идентификаторы алгоритмов и методов **должны** быть непустыми строками печатаемых символов из набора US-ASCII, не превышающими по размеру 64 символов. Регистр символов в именах **должен** приниматься во внимание.

Существует два формата представления имён алгоритмов и методов:

- Имена, не содержащие символа @, зарезервированы и выделяются **по согласованию с IETF**. Примерами таких имён могут служить 3des-cbc, sha-1, hmac-sha1, zlib. Имена в таком формате корректны только после их регистрации в IANA. **Недопустимо** включение в регистрируемые имена символов @, запятых (,), пробелов, управляющих символов (коды ASCII до 32 включительно) и символов ASCII с кодом 127 (DEL). Регистр символов в именах принимается во внимание, **недопустимы** имена размером более 64 символов.

- Каждый может определять дополнительные имена алгоритмов и методов в формате name@domainname (например, ourcipher-cbc@example.com). Формат левой части имени (до символа @) не задаётся спецификацией, однако эта часть **должна** быть строкой печатаемых символов US-ASCII и **недопустимо** включение в имена запятых (,), пробелов, символов управления (коды ASCII до 32 включительно) и символов с кодом ASCII 127 (DEL). Имя **должно** включать единственный символ @. Правая часть имени (после символа @) **должна** быть корректным полным доменным именем [RFC1034], контролируемым персоной или организацией, определяющей имя с этим доменом. Регистр символов в именах принимается во внимание, **недопустимы** имена размером более 64 символов. Каждый домен сам определяет способы управления своим пространством имён. Следует отметить, что эти имена похожи на определённые в STD 11 [RFC0822] адреса электронной почты. Однако это сходство является только внешним и имена никак не связаны с STD 11 [RFC0822].

7. Номера сообщений

Пакеты SSH используют номера сообщений от 1 до 255. Эти номера распределены между различными компонентами.

Протокол транспортного уровня

1 - 19	базовые сообщения транспортного уровня (например, disconnect, ignore, debug и т. п.);
20 - 29	согласование алгоритма;
30 - 49	сообщения, связанные с обменом ключами (допускается совпадение номеров для разных методов аутентификации).

Протокол аутентификации пользователей

50 - 59	базовые сообщения протокола аутентификации;
60 - 79	сообщения, связанные с методом аутентификации (допускается совпадение номеров для различных методов).

Протокол соединений

80 - 89 базовые сообщения протокола;
90 - 127 сообщения, связанные с каналом.

Зарезервировано для клиентских протоколов

128 - 191 резерв

Локальные расширения

192 - 255 локальные расширения.

8. Взаимодействие с IANA

Этот документ является частью набора документов, задающих спецификацию протокола. Инструкции по взаимодействию с IANA относительно протокола SSH, определённые в данном документе, [SSH-USERAUTH], [SSH-TRANS] и [SSH-CONNECT], детализированы в [SSH-NUMBERS]. Далее для удобства приведена краткая информация, но реальные инструкции по взаимодействию с IANA, содержащиеся в [SSH-NUMBERS], могут впоследствии изменить или отменить написанное здесь.

Выделение перечисленных ниже типов имён для протокола SSH осуществляется по согласованию с IETF:

- Имена служб:
  - методы аутентификации;
  - имена каналов протокола соединений;
  - имена глобальных запросов протокола соединений;
  - имена запросов канала протокола соединений.
- Названия методов обмена ключами.
- Имена алгоритмов:
  - алгоритмы шифрования;
  - алгоритмы MAC[1];
  - алгоритмы открытых ключей;
  - алгоритмы сжатия данных.

Имена **должны** указываться с использованием набора символов US-ASCII, **недопустимо** включение в имена символов @, запятых, пробелов, символов управления (коды ASCII до 32 включительно) и символов ASCII с кодом 127 (DEL). Регистр символов в именах принимается во внимание; **недопустимы** имена размером более 64 символов.

Имена с символом @ являются локальными расширениями и не контролируются IANA.

Каждая из перечисленных выше категорий использует независимое пространство имён. Однако использования совпадающих имён в различных категориях **следует** избегать, чтобы не возникало путаницы.

Номера сообщений (см. раздел 7) в диапазоне от 0 до 191 выделяются **по согласованию с IETF**, как описано в [RFC2434]. Номера сообщений от 192 до 255 (локальные расширения) зарезервированы **для приватного использования**, как описано в [RFC2434].

9. Вопросы безопасности

В целях улучшения восприятия этот раздел содержит информацию по вопросам безопасности из документов, посвящённых транспорту, аутентификации и соединениям.

Транспортный протокол [SSH-TRANS] обеспечивает поддержку конфиденциальных каналов через сети, не обеспечивающие защиты. Этот протокол выполняет операции по аутентификации серверных хостов, обмену ключами, шифрованию и обеспечению целостности. Протокол также обеспечивает уникальный идентификатор сессии, который может использоваться протоколами вышележащих уровней.

Протокол аутентификации [SSH-USERAUTH] обеспечивает набор механизмов, который позволяет проверить полномочия пользователя на сервере. Отдельные механизмы протокола аутентификации используют идентификатор сессии, предоставленный транспортным протоколом, и/или зависят от гарантий безопасности и целостности, предоставляемых транспортным протоколом.

Протокол соединений [SSH-CONNECT] задает механизм мультиплексирования множества потоков (каналов) данных через одно конфиденциальное доверенное транспортное соединение. Этот протокол также поддерживает каналы для доступа к интерактивным сеансам (shell), доставки данных различных внешних протоколов (включая любые протоколы TCP/IP) с использованием защищённого транспорта и доступа к защищённым системам серверного хоста.

9.1. Генерация псевдослучайных чисел

Этот протокол связывает с каждым сеансом свой ключ (session key), используя случайные данные, связанные с сеансом в хэше, служащем для создания сеансовых ключей. Следует принимать специальные меры по обеспечению высокого качества случайных чисел. Если случайные данные (например, параметры Diffie-Hellman) являются псевдослучайными, следует использовать криптографически защищённый генератор случайных чисел (т. е., следующий результат не должен быть предсказуемым на основе предыдущих результатов даже в тех случаях, когда известен их полный набор) с подходящим уровнем энтропии. [RFC4086] содержит рекомендации по выбору источников случайных чисел и энтропии. Разработчикам следует принимать во внимание важность энтропии и не забывать о сложностях реализации функций генерации псевдослучайных чисел.

[1] Message Authentication Code - код аутентификации сообщения. *Прим. перев.*

Уровень энтропии, доступный для данного клиента или сервера, может в некоторых случаях оказаться недостаточным. В таких случаях следует воспользоваться генератором псевдослучайных значений, несмотря на недостаток энтропии, или отвергнуть использование протокола. Второй вариант является более предпочтительным.

9.2. Фильтрация управляющих символов

При выводе текста для пользователя (сообщения об ошибках или отладочная информация) клиентским программам **следует** заменять управляющие символы (кроме табулятора, перевода строки и возврата каретки) безопасными последовательностями для предотвращения атак, организуемых путём передачи управляющих символов.

9.3. Транспорт

9.3.1. Конфиденциальность

В число задач этого документа и рабочей группы Secure Shell не входит анализ или рекомендации по выбору шифров, отличающихся от тех, которые уже разработаны и приняты в отрасли. На момент создания документа к таким шифрам относятся 3DES, ARCFOUR, twofish, serpent и blowfish. Алгоритм AES был опубликован The US Federal Information Processing Standards, как [FIPS-197] и принят в качестве одного из стандартных методов шифрования. Как всегда разработчикам и пользователям следует обратиться к современной литературе для того, чтобы убедиться в отсутствии уязвимостей в алгоритме или его реализациях. Разработчикам следует также посмотреть, какие алгоритмы считаются более сильными по сравнению с другими и рекомендовать пользователям применение более сильных алгоритмов. Хорошим тоном для разработчиков является вежливое и ненавязчивое уведомление пользователей о существовании более сильных алгоритмов шифрования при выборе пользователем более слабого алгоритма.

Режим шифрования none поддерживается для отладки, поэтому этим режимом **не следует** пользоваться для иных целей. Криптографические свойства этого режима достаточно подробно описаны в [RFC2410] и из этого документа можно сделать вывод о неприменимости данного режима для протокола SSH.

Сравнение алгоритмов шифрования также можно найти в современной литературе. В качестве примеров таких публикаций укажем работы [SCHNEIER] и [KAUFMAN]. Оба документа описывают CBC-режим[1] использования некоторых шифров и недостатки этой схемы. Важно отметить, что этот режим CBC теоретически уязвим для атак chosen cipher-text по причине достаточно высокой предсказуемости стартовых последовательностей пакетов. Однако такие атаки достаточно сложны и не могут считаться применимыми на практике особенно в случаях использования блоков большого размера.

Атаки против режима CBC можно также предотвратить путём вставки пакетов, содержащих сообщение SSH_MSG_IGNORE. Без использования этого метода некоторые атаки могут приводить к успеху. Для того, чтобы данный тип атак (обычно их называют Rogaway-атаками [ROGAWAY], [DAI], [BELLARE]) работал, атакующему нужно знать вектор инициализации IV[2] блока, который будет шифроваться следующим. В режиме CBC это результат шифрования предыдущего блока. Если атакующий не имеет возможности вовремя увидеть пакет (например, пакет находится в буфере модуля SSH или даже в ядре), атака не достигнет успеха. Если пакет будет передан в сеть, к которой атакующий имеет доступ, атака может оказаться успешной.

В оптимальном варианте от разработчика требуется добавление лишь одного пакета, если пакет уже передан в сеть и нет других пакетов, ожидающих передачи. Разработчики могут проверить наличие пакетов ожидающих передачи, но получение такой информации от ядра или буферов может оказаться достаточно сложной задачей. Если ожидающие передачи пакеты отсутствуют, **следует** передавать пакет, содержащий SSH_MSG_IGNORE. Если новый пакет будет всякий раз добавляться в поток, атакующий, даже зная IV (который по его предположению будет использоваться для шифрования следующего пакета), не сможет корректно предсказать IV и атака не достигнет успеха.

Рассмотрим пример:

```
Клиент                                                  Сервер
------                                                  ------
TCP(seq=x, len=500)              ---->
 содержит запись Record 1

                    [прошло 500 ms, пакета ACK нет]

TCP(seq=x, len=1000)             ---->
 содержит записи Records 1, 2

                                                          ACK
```

1. Алгоритм Nagle вкупе с алгоритмом повтора передачи TCP подразумевают возможность включения двух записей в один сегмент TCP.
2. Запись Record 2 не находится в начале сегмента и никогда не будет там, поскольку она уже подтверждена с помощью ACK.
3. Возможность атаки сохраняется, поскольку запись Record уже передана в сеть.

Этот пример показывает опасность опасность использования имеющихся в буфере TCP неотправленных данных в качестве признака необходимости передачи пустого пакета, поскольку в этом случае на момент повторного вызова write() в буфере содержится неподтвержденная запись Record 1.

Показанная ниже ситуация является совершенно безопасной

```
Клиент                                                  Сервер
------                                                  ------
TCP(seq=x, len=500)              ---->
   содержит SSH_MSG_IGNORE
```

[1]Cipher-block chaining - сцепка шифрованных блоков. Прим. перев.
[2]Initialization Vector.

```
      TCP(seq=y, len=500)              ---->
         содержит данные (Data)
```

В предположении, что IV для второй записи SSH Record фиксировано после получения данных для пакета Data, нужно выполнить следующие операции:

чтение пользовательских данных;

шифрование пустого пакета;

шифрование пакета данных.

9.3.2. Целостность данных

Данный протокол позволяет отключить механизм проверки целостности данных (Data Integrity). Разработчикам **следует** с осторожностью относится к возможности использования этой функции для каких-либо целей за исключением отладки. Пользователей и администраторов **следует** явно предупреждать при выборе значения none для MAC.

Если не используется опция none для MAC, данный протокол обеспечивает целостность данных.

Поскольку MAC использует 32-битовый порядковый номер, возможна утечка информации после передачи 2^{32} пакетов. Однако выполнение рекомендаций по регенерации ключей должно предотвратить возможность таких атак. Спецификация транспортного протокола [SSH-TRANS] рекомендует менять ключ (rekeying) после передачи 1 Гигабайта данных, а минимальный размер пакета составляет 16 байтов. Следовательно, смена ключей при таких условиях должна произойти до того, как будет передано 2^{28} пакетов.

9.3.3. Использование перехваченных данных (Replay)

Использование для MAC опций, отличных от none, обеспечивает целостность данных и аутентификацию. В дополнение к этому транспортный протокол обеспечивает уникальный идентификатор сессии (связанный с псевдослучайными данными, которые являются частью алгоритма и процесса обмена ключами), который может использоваться протоколами вышележащего уровня для привязки данных к текущей сессии и предотвращения случаев воспроизведения данных, перехваченных из других сессий. Например, протокол аутентификации [SSH-USERAUTH] использует идентификатор сессии для предотвращения повторного использования сигнатур предыдущих сеансов. Поскольку обмен открытыми ключами криптографически связан с текущей сессией (т. е, начальным обменом ключами), перехваченные сигнатуры невозможно повторно использовать в других сессиях. Отметим, что идентификатор сессии можно делать открытым без снижения уровня защищенности протокола.

Если две сессии имеют одинаковые идентификаторы (хэш обмена ключами), пакеты из одной сессии могут использоваться в другой. Следует подчеркнуть, что вероятность такого совпадения при использовании современных методов криптографии является минимальной. Дополнительное снижение этой вероятности может обеспечиваться увеличением размерности результата хэш-функции и параметров DH.

Детектирование попыток использования перехваченных данных с использованием монотонного роста порядковых номеров на входе MAC (или HMAC в некоторых случаях) рассматривается в документах [RFC2085], [RFC2246], [RFC2743], [RFC1964], [RFC2025], [RFC4120]. Основа этих методов описана в документе [RFC2104]. Существенно, что различие порядковых номеров в каждом пакете гарантирует, что, по крайней мере один, аргумент (input) MAC-функции будет уникальным и обеспечит не повторяющийся результат MAC, который не может быть предсказан атакующим. Если сессия длится достаточно долго, порядковые номера могут начать повторяться. Этот повтор может дать атакующему возможность повторного использования ранее записанных пакетов с идентичными порядковыми номерами, но такая ситуация возможна только в тех случаях, когда партнёры не используют повторной генерации ключей до того, как начнётся повтор порядковых номеров. Если партнёры выполнили регенерацию ключей, повторное использование пакетов будет обнаружено, поскольку проверка MAC даст отрицательный результат. По этой причине необходимо сделать упор на то, что партнёры **должны** выполнить регенерацию ключей до того, как начнётся повтор порядковых номеров. Естественно, если атакующий будет пытаться повторно использовать собранные пакеты до того, как партнёры проведут регенерацию ключей, получатель дубликата не получит корректное значение MAC и пакет будет отброшен. Причина некорректности результата MAC в таких случаях обусловлена тем, что получатель будет создавать MAC на основе принятого пакета, разделяемого ключа (shared secret) и ожидаемого порядкового номера. Поскольку используемый повторно пакет не имеет ожидаемого порядкового номера (пакет с таким номером уже был принят получателем), рассчитанное значение MAC не будет соответствовать значению из принятого пакета.

9.3.4. Перехват с участием человека (Man-in-the-middle)

Протокол не включает предположений о распространении открытых ключей хостов, требований к инфраструктуре обмена ключами или методам их распространения. Предполагается, что протокол в некоторых случаях будет использоваться без предварительной проверки достоверности связи между ключом и именем хоста. Такое использование уязвимо для атак с участием человек (man-in-the-middle attack). В этом параграфе рассматривается данная проблема и даются рекомендации для администраторов и пользователей по вопросам важности проверки связи между ключом и именем хоста до начала первого сеанса.

Существует три варианта атак на протокол с использованием перехвата с участием человека. В первом варианте атакующий размещает устройство перехвата на пути между клиентом и сервером до начала сеанса. В этом случае устройство пытается прикинуться легитимным сервером и будет предлагать клиенту свой ключ в начале сеанса. Если устройство станет предлагать реальный ключ сервера, оно не сможет расшифровать или подписать данные, передаваемые между клиентом и легитимным сервером, пока не получит доступа к закрытому ключу (private key) хоста. Используемое для атаки устройство одновременно будет инициировать сессию с легитимным сервером, маскируя себя под клиента. Если открытый ключ сервера был до начала сессии получен клиентом с помощью безопасного канала, предложенный используемым для атаки устройством ключ не будет соответствовать ключу сервера, хранящемуся у клиента. В таким случае пользователю **следует** выдавать предупреждение о несоответствии ключей. Как указано в параграфе 4.1, пользователь волен принять предложенный ключ и продолжить сессию. **Рекомендуется** включать в предупреждение о несоответствии ключей информацию, которая позволит пользователю клиентского хоста принять обдуманное решение. Если пользователь выберет продолжение сессии с хранящимся у него открытым ключом сервера (вместо ключа, предложенного в начале сеанса), связанные с этим сеансом данные,

передаваемые между атакующим и сервером будут отличаться от данных, передаваемых между атакующим и клиентом в силу описанного выше использования случайных чисел. В результате атакующий не сможет добиться своей цели, поскольку он не имеет возможности корректно подписывать пакеты, содержащие связанные с сессией данные от сервера, так как ему не известен закрытый ключ сервера.

Второй вариант похож на первый и также основан на перехвате на этапе организации соединения, но этот случай подчёркивает важность безопасного распространения открытых ключей. Если для открытого ключа сервера не обеспечивается безопасное распространение, клиент не может удостовериться в том, что он соединился с нужным сервером. Атакующий может воспользоваться мошенническими методами (social engineering) для передачи ключей ничего не подозревающим пользователям и после этого поместить устройство перехвата на пути между клиентом и легитимным сервером. В этом случае клиент просто соединится с устройством атакующего, а атакующий организует соединение с легитимным сервером и сможет организовать мониторинг данных от клиента и сервера, а также манипулировать этими данными. Администраторам серверов рекомендуется делать «отпечатки» ключей доступными для проверки с помощью тех или иных средств, безопасность которых не влияет на целостность реальных ключей. Возможные механизмы обсуждались в параграфе 4.1 и могут также включать защищённые Web-страницы, передачу на бумажных носителях и т. п. Разработчикам **следует** обеспечивать рекомендации по эффективному использованию таких мер с их реализациями. Поскольку протокол является расширяемым, в будущих версиях могут обеспечиваться более эффективные методы распространения ключей серверов до организации первого соединения. Например, могут создаваться отпечатки ключей доступные через защищённые серверы DNS или использоваться протокол Kerberos ([RFC4120]) через GSS-API ([RFC1964]) в процессе обмена ключами для аутентификации сервера.

В третьем варианте атакующий может попытаться манипулировать пакетами в процессе их передачи между партнёрами после организации сеанса. Как описано в параграфе 9.3.3, вероятность успеха при таких атаках весьма мала. Как и для описанного в параграфе 9.3.3 случая причина низкой вероятности успеха таких атак обусловлена тем, что алгоритм MAC является достаточно безопасным и нереально подобрать входные данные для него, чтобы получить на выходе желаемый результат. Более детальное обсуждение этого вопроса приводится в разделе 6 [RFC2104]. Если алгоритм MAC имеет уязвимости или достаточно слаб, атакующий может оказаться способен задать те или иные данные на входе для получения известного значения MAC. Используя это, он сможет изменить передаваемые через сеть пакеты. Кроме того, атакующий может воспользоваться уязвимостью алгоритма или его слабостью для определения разделяемого ключа (shared secret) путём просмотра MAC в захваченных пакетах. В любом из этих случаев атакующий сможет создавать пакеты и помещать их в поток SSH. Для предотвращения этого разработчикам следует использовать общепринятые проверенные алгоритмы MAC, а администраторам следует читать современную литературу и дискуссии для того, чтобы вовремя узнавать о найденных в используемых алгоритмах MAC уязвимостях или недостатках.

В заключение отметим, что использование этого протокола без надлежащей проверки связи между хостами и их ключами, не обеспечивает должного уровня безопасности и **не рекомендуется**. Однако такое использование может оказаться необходимым для сред, где безопасность не играет критически важной роли и обеспечивается защита от пассивных атак. Разработчикам протоколов и использующих их приложений следует принимать этот факт во внимание.

9.3.5. DoS-атаки

Этот протокол предназначен для работы на основе надёжного транспорта. При возникновении ошибок передачи или манипуляции сообщениями соединение закрывается. В таких случаях **следует** предпринять попытку восстановить соединение. Атак этого типа (обрыв провода) почти невозможно избежать.

В дополнение к сказанному данный протокол уязвим для атак на службы (denial of service attack), при которых атакующий может вывести сервер из строя за счёт потребления ресурсов процессора и памяти путём организации множества соединений и обмена ключами без аутентификации. Разработчикам **следует** принимать меры по осложнению таких атак (например, разрешая соединения лишь для ограниченного числа известных клиентов).

9.3.6. Скрытые каналы

Протокол не включает мер предотвращения скрытых каналов. Например, поля заполнения (padding), сообщения SSH_MSG_IGNORE и некоторые другие места могут использоваться для передачи скрытой информации, для которой не существует надёжного способа обнаружения на принимающей стороне.

9.3.7. Сохранность тайн

Следует отметить, что алгоритм обмена ключами Diffie-Hellman (DH) может обеспечивать PFS[1]. PFS определяется как криптографическое свойство протокола организации ключей, при котором компрометация сеансового ключа или долгосрочного закрытого ключа после завершения сессии не ведёт к компрометации более ранних сессий [ANSI-T1.523-2001]. Сессии SSH, основанные на обмене ключами с использованием методов DH, описанные в разделе «Обмен ключами по методу Diffie-Hellman» документа [SSH-TRANS] (включая diffie-hellman-group1-sha1 и diffie-hellman-group14-sha1), обеспечивают защиту даже при последующем раскрытии ключей или данных аутентификации. Следовательно, протокол SSH обеспечивает PFS. Однако это свойство не передаётся каким-либо приложениям или протоколам, использующим SSH в качестве транспорта. Транспортный уровень SSH обеспечивает конфиденциальность аутентификации и других методов, основанных на секретных данных.

Естественно при раскрытии параметров DH для клиента или сервера раскрываются сеансовые ключи, но они не представляют ценности после завершения обмена ключами. Это лишний раз подчёркивает, что такие элементы не должны сбрасываться в область подкачки и их следует удалять из памяти после завершения обмена ключами.

9.3.8. Упорядочивание методов обмена ключами

Как сказано в разделе «Согласование алгоритма» документа [SSH-TRANS], каждое устройство будет передавать список предпочтительных методов обмена ключами. Наиболее предпочтительный метод указывается в списке первым. **Рекомендуется** сортировать алгоритмы по силе криптографических методов, размещая первым наиболее сильный. Некоторые рекомендации по этому вопросу содержатся в документе [RFC3766].

[1]Perfect forward secrecy - идеальная секретность.

9.3.9. Анализ трафика

Пассивный мониторинг любого протокола может дать атакующему некоторую информацию о сессии, пользователе и протоколе, которую не удалось бы получить иным путём. Например, показано, что анализ трафика SSH-сессии позволяет получить информацию о размере пароля - [Openwall] и [USENIX]. Разработчикам **следует** использовать пакеты SSH_MSG_IGNORE вкупе с заполнением случайной длины для осложнения попыток анализа трафика. Для этих же целей могут использоваться и другие методы.

9.4. Протокол аутентификации

Задачей этого протокола является аутентификация пользователей. Предполагается, что этот протокол работает на основе защищённого протокола транспортного уровня, который уже обеспечил аутентификацию сервера, организовал шифрованный коммуникационный канал и создал уникальный идентификатор для данной сессии.

Допускается использование нескольких методов аутентификации с различными параметрами защиты. Выбор методов аутентификации (или их комбинации), дозволенных каждому пользователю, определяется локальной политикой сервера. Уровень аутентификации не превышает уровень наиболее слабой из допустимых комбинаций методов.

Сервер может «засыпать» после нескольких неудачных попыток аутентификации для того, чтобы осложнить атакующему поиск ключа. Следует принимать меры, чтобы в таких случаях не возникала возможность организации DoS-атак.

9.4.1. Проблема незащищённого транспорта

Если транспортный уровень не обеспечивает конфиденциальности, методы аутентификации, основанные на секретных данных, **следует** отключить. Если не обеспечивается достаточной защиты целостности, **следует** отключить запросы на изменение параметров аутентификации (например, смена паролей) для предотвращения возможности незаметной подмены атакующим шифрованного текста или блокирования им возможности использования новых параметров аутентификации (DoS-атака).

Приведённое выше допущение о том, что протокол аутентификации использует только защищённый транспорт, имеет очень важное значение. При развёртывании систем SSH не следует забывать о пагубных последствиях MITM-атак в тех случаях, когда у клиента нет способа достаточно надёжно проверить связь ключа с сервером до начала сессии. В частности, применительно к протоколу аутентификации, клиент может организовать соединение с устройством перехвата, используемым атакующим, и выдать последнему используемые для аутентификации сведения (в частности, имя пользователя и пароль). Даже в тех случаях, когда используемые для аутентификации данные не будут раскрыты, атакующий может получить информацию, перехватывая данные о нажатии клавиш, подобно тому, как это делается в приманках (honeypot).

9.4.2. Отладочные сообщения

Следует с осторожностью подходить к созданию отладочных сообщений. Эти сообщения могут раскрывать информацию о хосте, если к их содержанию не отнестись должным образом. Отладочные сообщения могут быть отключены (на этапе аутентификации пользователя), если требуется обеспечить высокий уровень защиты. Администраторам хостов следует принять все возможные меры по изоляции всех уведомлений о событиях и их защите от просмотра нежелательными лицами. Разработчикам следует внимательно отнестись к возможности раскрытия информации через некоторые вполне обыденные события или отладочные сообщения. Неплохо также подготовить рекомендации для администраторов по методам предотвращения доступа к такой информации нежелательным лицам. Разработчикам следует предусмотреть возможность минимизировать с помощью локальной политики объем критичной к раскрытию информации, передаваемой пользователю на этапе аутентификации. Для решения этой задачи **рекомендуется** по умолчанию отключать выдачу отладочных сообщений и разрешать их включение только в результате целенаправленных действий администратора. **Рекомендуется** также обеспечивать выдачу администратору соответствующего предупреждения в тех случаях, когда он разрешает выдачу отладочных сообщений.

9.4.3. Локальная политика безопасности

Разработчики **должны** обеспечить проверку пользователя на основании представленных им данных, а также возможность применения локальной политики сервера для этого пользователя. По причине гибкости протокола SSH может отсутствовать возможность определения локальной политики безопасности, которую следует применять во время аутентификации, поскольку в этот момент невозможно установить тип запрашиваемого пользователем сервиса. Например, локальная политика может разрешать пользователю доступ к файлам на сервере, но не разрешать доступ к командному процессору (interactive shell). Однако при работе протокола аутентификации неизвестно, что собирается делать пользователь - работать с файлами, использовать командный процессор или выполнять оба типа операций. В любом случае при существовании на сервере локальной политики она должна применяться корректно.

Разработчикам рекомендуется обеспечивать принятую по умолчанию локальную политику и делать её параметры известными администраторам и пользователям. По усмотрению разработчиков эта локальная политика может разрешать пользователям выполнение любых операций или вносить ограничения в действия пользователей - администраторы могут изменить установленную по умолчанию политику в соответствии с реальными задачами. Возможен также вариант, когда принятая по умолчанию политика основана на практическом опыте и может помочь администраторам, не имеющим достаточного опыта работы с SSH. В любом случае политика должна применяться и реализоваться в соответствии с описанными выше требованиями.

9.4.4 Аутентификация с использованием открытых ключей

Использование для аутентификации открытых ключей предполагает, что клиентский хост не компрометирован. Предполагается также отсутствие компрометации для закрытого ключа сервера.

В целях ослабления риска можно можно использовать для закрытых ключей дополнительные пароли (passphrase), однако это не реализуется на основе политики. Предлагается использовать смарт-карты или иные технологии для согласования парольных фраз с политикой.

Сервер может требовать одновременно пароль и открытый ключ, однако это требование ведёт к тому, что пользовательский пароль становится известным серверу (см. следующий параграф).

9.4.5. Парольная аутентификация

Парольный механизм, как задано протоколом аутентификации, предполагает, что сервер не компрометирован. При компрометации сервера использование парольной аутентификации будет раскрывать атакующему комбинации имён пользователей и паролей, что может вести к дальнейшему возрастанию риска.

Эту уязвимость можно преодолеть за счёт использования другой формы аутентификации. Например, аутентификация на основе открытых ключей не делает допущений об уровне безопасности сервера.

9.4.6. Аутентификация по хостам

Аутентификация по хостам предполагает, что клиентские хосты не компрометированы. Для этого варианта не существует стратегий снижения риска, кроме использования аутентификации по хостам в комбинации с другими методами аутентификации.

9.5. Протокол соединений

9.5.1. Защищенность конечных точек

Защищенность конечных точек предполагается для протокола соединений. При компрометации сервера все терминальные сессии, перенаправление портов или доступ в систему также подвергаются компрометации. Способов предотвращения этого риска нет.

При компрометации клиента, если серверу не удаётся остановить атаку на уровне протокола аутентификации, все видимые службы (подсистемы и перенаправление) становятся уязвимыми для атаки. Разработчикам **следует** обеспечивать для администраторов механизмы управления доступными службами для снижения уровня уязвимости. Такие механизмы могут включать выбор хостов и портов, которые могут использоваться в качестве цели перенаправления, хостов, к которым пользователи получают shell-доступ или на которых разрешается использование видимых подсистем.

9.5.2. Перенаправление

Протокол SSH поддерживает перенаправление (proxy forwarding) для других протоколов типа SMTP, POP3 и HTTP. Это может представлять интерес для сетевых администраторов, которые заинтересованы в контроле доступа удалённых пользователей к некоторым приложениям. Важно отметить, что перенаправление протоколов может приводить к нарушению политики безопасности для сайта, поскольку позволяет организовать незаметные туннели через межсетевые экраны. Разработчикам **следует** обеспечивать для администраторов механизмы контроля перенаправления, позволяющие обеспечить выполнение политики безопасности для сайтов.

Кроме того, возможно обратное перенаправление, которое также позволяет обходить контроль в межсетевых экранах.

Как было сказано выше, перенаправление предполагает защищенность конечных точек. Компрометация конечной точки будет приводить к компрометации всех данных, передаваемых с использованием перенаправления.

9.5.3. Перенаправление трафика X11

Другой формой перенаправления, обеспечиваемого протоколом соединений SSH, является перенаправление для протокола X11. При нарушении системы защиты конечной точки перенаправление X11 может открыть возможность для атаки на сервер X11. Пользователям и администраторам следует применять соответствующие механизмы обеспечения защиты X11 для предотвращения несанкционированного доступа к серверам X11. Разработчикам, администраторам и пользователям, желающим повысить уровень защиты X11, рекомендуется прочесть документ [SCHEIFLER] и проанализировать известные проблемы для случаев использования SSH-пересылки с X11 в бюллетенях CERT VU#363181 и VU#118892 [CERT].

X11-дисплей, использующий пересылку SSH, сам по себе не может решить проблем, связанных с защитой X11 [VENEMA]. Однако пересылка X11 с использованием SSH (или иного защищённого протокола) в комбинации с псевдотерминалами, принимающими соединения только через локальные механизмы IPC, использующими авторизацию или списки контроля доступа (ACL[1]), могут решить большинство проблем с безопасностью X11, если для MAC не выбран режим none. Разработчикам дисплеев X11 **рекомендуется** по умолчанию разрешать отображение только через локальные IPC. Разработчикам серверов SSH, поддерживающих перенаправление X11, **рекомендуется** по умолчанию разрешать соединения только через локальные IPC. На однопользовательских системах может оказаться разумным открывать по умолчанию также соединения через TCP/IP.

Разработчикам протоколов перенаправления X11 **следует** реализовать для контроля доступа механизм magic cookie, описанный в [SSH-CONNECT], в качестве дополнительного средства предотвращения несанкционированного использования прокси-сервера.

10. Литература

10.1. Нормативные документы

[SSH-TRANS] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253, January 2006.

[SSH-USERAUTH] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252, January 2006.

[SSH-CONNECT] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Connection Protocol", RFC 4254, January 2006.

[1]Access Control List.

[SSH-NUMBERS] Lehtinen, S. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Assigned Numbers", RFC 4250, January 2006.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2434] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[RFC3066] Alvestrand, H., "Tags for the Identification of Languages", BCP 47, RFC 3066, January 2001.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

10.2. Дополнительная литература

[RFC0822] Crocker, D., "Standard for the format of ARPA Internet text messages", STD 11, RFC 822, August 1982.

[RFC0854] Postel, J. and J. Reynolds, "Telnet Protocol Specification", STD 8, RFC 854, May 1983.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC1282] Kantor, B., "BSD Rlogin", RFC 1282, December 1991.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.

[RFC1964] Linn, J., "The Kerberos Version 5 GSS-API Mechanism", RFC 1964, June 1996.

[RFC2025] Adams, C., "The Simple Public-Key GSS-API Mechanism (SPKM)", RFC 2025, October 1996.

[RFC2085] Oehler, M. and R. Glenn, "HMAC-MD5 IP Authentication with Replay Prevention", RFC 2085, February 1997.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[RFC2410] Glenn, R. and S. Kent, "The NULL Encryption Algorithm and Its Use With IPsec", RFC 2410, November 1998.

[RFC2743] Linn, J., "Generic Security Service Application Program Interface Version 2, Update 1", RFC 2743, January 2000.

[RFC3766] Orman, H. and P. Hoffman, "Determining Strengths For Public Keys Used For Exchanging Symmetric Keys", BCP 86, RFC 3766, April 2004.

[RFC4086] Eastlake, D., 3rd, Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[FIPS-180-2] US National Institute of Standards and Technology, "Secure Hash Standard (SHS)", Federal Information Processing Standards Publication 180-2, August 2002.

[FIPS-186-2] US National Institute of Standards and Technology, "Digital Signature Standard (DSS)", Federal Information Processing Standards Publication 186-2, January 2000.

[FIPS-197] US National Institute of Standards and Technology, "Advanced Encryption Standard (AES)", Federal Information Processing Standards Publication 197, November 2001.

[ANSI-T1.523-2001] American National Standards Institute, Inc., "Telecom Glossary 2000", ANSI T1.523-2001, February 2001.

[SCHNEIER] Schneier, B., "Applied Cryptography Second Edition: protocols algorithms and source in code in C", John Wiley and Sons, New York, NY, 1996.

[SCHEIFLER] Scheifler, R., "X Window System : The Complete Reference to Xlib, X Protocol, Icccm, Xlfd, 3rd edition.", Digital Press, ISBN 1555580882, February 1992.

[KAUFMAN] Kaufman, C., Perlman, R., and M. Speciner, "Network Security: PRIVATE Communication in a PUBLIC World", Prentice Hall Publisher, 1995.

[CERT] CERT Coordination Center, The., "http://www.cert.org/nav/index_red.html".

[VENEMA] Venema, W., "Murphy's Law and Computer Security", Proceedings of 6th USENIX Security Symposium, San Jose CA http://www.usenix.org/publications/library/proceedings/sec96/venema.html, July 1996.

[ROGAWAY] Rogaway, P., "Problems with Proposed IP Cryptography", Unpublished paper http://www.cs.ucdavis.edu/~rogaway/ papers/draft-rogaway-ipsec-comments-00.txt, 1996.

[DAI] Dai, W., "An attack against SSH2 protocol", Email to the SECSH Working Group ietf-ssh@netbsd.org ftp://ftp.ietf.org/ietf-mail-archive/secsh/2002-02.mail, Feb 2002.

[BELLARE] Bellaire, M., Kohno, T., and C. Namprempre, "Authenticated Encryption in SSH: Fixing the SSH Binary Packet Protocol", Proceedings of the 9th ACM Conference on Computer and Communications Security, Sept 2002.

[Openwall] Solar Designer and D. Song, "SSH Traffic Analysis Attacks", Presentation given at HAL2001 and NordU2002 Conferences, Sept 2001.

[USENIX] Song, X.D., Wagner, D., and X. Tian, "Timing Analysis of Keystrokes and SSH Timing Attacks", Paper given at 10th USENIX Security Symposium, 2001.

Адреса авторов

Tatu Ylonen

SSH Communications Security Corp

Valimotie 17

00380 Helsinki

Finland

EMail: ylo@ssh.com

Chris Lonvick (редактор)

Cisco Systems, Inc.

12515 Research Blvd.

Austin 78759

USA

EMail: clonvick@cisco.com

Перевод на русский язык

Николай Малых

nmalykh@protokols.ru

Торговые марки

ssh - торговый знак, зарегистрированный в США и/или других странах.

Подтверждение

Финансирование функций RFC Editor обеспечено IETF Administrative Support Activity (IASA).