Network Working Group
Request for Comments: 4256
Category: Standards Track

F. Cusack
savecore.net
M. Forssen
AppGate Network Security AB

January 2006

Базовая аутентификация обмена сообщениями для SSH

Generic Message Exchange Authentication for the Secure Shell Protocol (SSH)

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

Тезисы

Протокол SSH [6] обеспечивает защищенный вход в удаленные системы (remote login) и другие защищенные услуги в сетях без защиты. В этом документе описан метод проверки подлинности общего назначения для протокола SSH, пригодный для интерактивной проверки подлинности, когда данные для проверки должны вводиться с клавиатуры (или заменяющего ее алфавитно-цифрового устройства ввода). Основная цель этого метода заключается в обеспечении клиентам SSH возможности поддержки целого класса механизмов проверки подлинности без наличия сведений о специфике конкретного механизма.

1. Введение

Протокол аутентификации SSH [SSH-USERAUTH] представляет собой протокол общего назначения для проверки подлинности пользователей. Этот протокол предназначен для работы поверх транспортного протокола SSH [SSH-TRANS]. Протокол аутентификации предполагает, что нижележащие протоколы обеспечивают целостность и конфиденциальность данных.

В этом документе описан метод проверки подлинности общего назначения для протокола аутентификации SSH. Этот метод подходит для интерактивных вариантов проверки подлинности, не требующих какой-либо специальной программной поддержки на стороне клиента. Все аутентификационные данные должны вводиться с клавиатуры. Основная цель этого метода заключается в том, чтобы позволить клиентам SSH не иметь информации об используемых сервером SSH механизмах аутентификации или обходиться минимальным объемом такой информации. Это будет позволять серверам произвольно выбирать или менять механизмы аутентификации без необходимости обновления кода на клиентской стороне.

Для этого метода проверки подлинности используется имя "keyboard-interactive".

Данный документ следует читать после ознакомления с архитектурой SSH [SSH-ARCH] и протоколом аутентификации SSH [SSH-USERAUTH]. В данном документе используется терминология и нотация из упомянутых документов без ссылок и дополнительных разъяснений.

В документе также описаны некоторые процедуры взаимодействия с пользователем на клиентской стороне при получении информации для проверки подлинности пользователя. Хотя это несколько выходит за рамки спецификации протокола, здесь приведено описание такого взаимодействия, поскольку некоторые аспекты протокола разработаны с учетом взаимодействия с пользователем и отказ от рассмотрения этой информации может привести к появлению несовместимых или неудобных в использовании реализаций.

Ключевые слова **необходимо** (MUST), **недопустимо** (MUST NOT), **требуется** (REQUIRED), **нужно** (SHALL), **не следует** (SHALL NOT), **следует** (SHOULD), **не нужно** (SHOULD NOT), **рекомендуется** (RECOMMENDED), **возможно** (MAY), **необязательно** (OPTIONAL) в данном документе интерпретируются в соответствии с [RFC-2119].

2. Обоснование

Определенные к настоящему времени методы аутентификации SSH тесно связаны с лежащими в их основе механизмами проверки подлинности. Это осложняет добавление новых механизмов проверки подлинности, поскольку требуется обновление всех клиентов для поддержки каждого нового механизма. С использованием описанного здесь общего метода для поддержки новых механизмов аутентификации не потребуется менять клиентские программы, а при использовании отдельного уровня проверки подлинности (например, [PAM]) можно обойтись без изменения программного кода даже на серверах.

Это обеспечивает значимые преимущества по сравнению с другими методами типа парольной аутентификации - "password" (определена в [SSH-USERAUTH]), поскольку новые (предположительно, более строгие) методы проверки подлинности могут добавляться «по желанию», обеспечивая прозрачное повышение уровня защищенности системы.

Механизм «запрос-отклик» (Challenge-response) и одноразовые пароли (One Time Password) легко могут поддерживаться при использовании этого метода.

Однако этот метод проверки подлинности ограничивается использованием механизмов, не требующих на клиентах специального кода (драйверов устройств или программ «искажения» паролей.

3. Протокольный обмен данными

Клиент инициирует аутентификацию сообщением SSH_MSG_USERAUTH_REQUEST, на которое сервер отвечает запросом аутентификационной информации от клиента в сообщении SSH_MSG_USERAUTH_INFO_REQUEST. Клиент получает информацию от пользователя и передает серверу сообщение SSM_MSG_USERAUTH_INFO_RESPONSE. Серверу **недопустимо** передавать другой запрос SSH_MSG_USERAUTH_INFO_REQUEST, пока не получен ответ от клиента.

3.1. Начальный обмен

Проверка подлинности начинается с передачи клиентом сообщения, показанного на врезке справа.

byte	SSH_MSG_USERAUTH_REQUEST
string	имя пользователя (ISO-10646 UTF-8, как определено в [RFC-3629])
string	имя службы (US-ASCII)
string	"keyboard-interactive" (US-ASCII)
string	тег языка (как определено в [RFC-3066])
string	субметоды (ISO-10646 UTF-8)

Использование тега языка осуждается и это поле **следует** оставлять пустым. В будущей версии спецификации поле будет удалено. Серверу **следует** выбирать язык на базе тегов, переданных в процессе обмена ключами [SSH-TRANS].

Если задан непустой тег языка, серверу **следует** использовать указанный язык для всех сообщений, передаваемых клиенту в рамках этого протокола. Тег языка **не следует** применять для выбора языка сообщений, выходящих за рамки протокола. Если сервер не поддерживает запрошенный язык, выбор используемого языка будет зависеть от реализации.

Поле субметодов включено для того, чтобы пользователь мог дать рекомендации о методах, которые он желает использовать. Это поле представляет собой список разделенных запятыми субметодов (аппаратных или программных) предпочитаемых пользователем. Если клиенту известны пользовательские предпочтения (предположительно, из конфигурационных параметров), он **может** использовать это поле для передачи имеющейся информации серверу. В остальных случаях поле **должно** оставаться пустым.

Реальные имена субметодов должны быть как-то согласованы между пользователем и сервером.

Интерпретация сервером поля субметодов зависит от реализации.

Одной из возможных стратегий обработки поля субметодов в серверной реализации является простое игнорирование этого поля, если у пользователя нет возможности применять множество разных субметодов. Если пользователь может проверять подлинность с помощью одного из нескольких различающихся субметодов, серверу следует трактовать это поле, как рекомендацию по выбору желательного для пользователя метода.

Отметим, что к моменту передачи сообщения серверу клиент еще не выдает запроса пользователю на ввод пароля и парольная информация **не** включается в начальное сообщение (в отличие от метода "password").

Сервер **должен** ответить на запрос сообщением SSH_MSG_USERAUTH_SUCCESS, SSH_MSG_USERAUTH_FAILURE, или SSH_MSG_USERAUTH_INFO_REQUEST.

Серверу **не следует** передавать ответ SSH_MSG_USERAUTH_FAILURE, если причиной отказа является имя пользователя или службы - вместо этого **следует** передать сообщение(я) SSH_MSG_USERAUTH_INFO_REQUEST, которое(ые) выглядит, как сообщение, передаваемое в случаях, когда следует выполнить аутентификацию, и после этого передать сообщение об отказе (с подобающей задержкой, как описано ниже). Цель такого поведения заключается в предотвращении возможности подбора корректных имен пользователей путем сравнения результатов аутентификации для различных имен.

Сервер **может** ответить сообщением SSH_MSG_USERAUTH_SUCCESS, если для указанного пользователя проверять подлинность не требуется. Однако, в силу описанных выше причин, лучше будет передать в ответ сообщение SSH_MSG_USERAUTH_INFO_REQUEST и игнорировать (без проверки) отклик на него.

3.2. Запросы информации

Запросы информации генерируются со стороны сервера с использованием сообщений SSH_MSG_USERAUTH_INFO_REQUEST message.

Сервер может передать для аутентификации клиента столько запросов информации, сколько сочтет нужным. Клиент **должен** быть готов к обработке множества таких запросов. Однако для сервера **недопустимо** иметь даже одно не обработанное сообщение SSH_MSG_USERAUTH_INFO_REQUEST. Т. е., сервер не может передавать другое сообщение, пока клиент не ответит на предыдущее.

byte	SSH_MSG_USERAUTH_INFO_REQUEST
string	имя (ISO-10646 UTF-8)
string	инструкция (ISO-10646 UTF-8)
string	тег языка (как определено в [RFC-3066])
int	num-prompts
string	prompt[1] (ISO-10646 UTF-8)
boolean	echo[1]
...	
string	prompt[num-prompts] (ISO-10646 UTF-8)
boolean	echo[num-prompts]

Формат сообщения SSH_MSG_USERAUTH_INFO_REQUEST показан на врезке справа.

Использование тега языка осуждается и это поле **следует** оставлять пустым. В будущей версии спецификации поле будет удалено. Серверу **следует** выбирать язык на базе тегов, переданных в процессе обмена ключами [SSH-TRANS].

Если задан непустой тег языка, серверу **следует** использовать указанный язык для всех сообщений, передаваемых клиенту в рамках этого протокола. Тег языка **не следует** применять для выбора языка сообщений, выходящих за рамки протокола. Если сервер не поддерживает запрошенный язык, выбор используемого языка будет зависеть от реализации.

Серверу **следует** принимать в внимание неспособность некоторых клиентов корректно отображать длинные поля имен или приглашений (см. следующий параграф) и по возможности ограничивать размер этих полей. Например, вместо инструкции "Enter Password"[1] и поля приглашения "Password for user23@host.domain: "[2] лучше будет задать инструкцию "Password authentication for user23@host.domain"[3] и поле приглашения "Password: ". Предполагается, что этот метод аутентификации будет в основном использоваться с [PAM] и такого выбора просто не будет возникать.

Поля имени и инструкции **могут** быть пустыми строками и клиент **должен** быть готов к корректной обработке таких полей. В качестве полей приглашения (prompt) **недопустимо** указывать пустые строки.

Поле num-prompts может иметь значение 0, показывающее отсутствие в сообщении полей prompt/echo, но клиенту по-прежнему **следует** отображать поля имени и инструкции (как описано ниже).

3.3. Пользовательский интерфейс

При получении запроса клиенту **следует** выдать приглашение пользователю на ввод данных, как описано ниже.

> Клиентам с интерфейсом CLI[4] **следует** отобразить имя и инструкцию (при наличии), добавляя новые строки. После этого для для каждого из полей prompt[ ] клиенту **следует** вывести на экран приглашение и прочитать введенные пользователем данные.
> Клиенты с графическим интерфейсом (GUI) имеют множество вариантов вывода приглашения для пользователя. Одним из таких вариантов является использование значения поля name (возможно, с именем приложения в качестве префикса) в качестве заголовка диалогового окна, в котором содержится приглашение на ввод информации. В этом диалоговом окне поле инструкции будет служить текстовым сообщением, а поля prompt[ ] - метками для полей ввода информации. Пльзователю **следует** показывать все поля. Например, реализациям **не следует** отбрасывать поле имени по причине отсутствия заголовка у окна - вместо этого **следует** найти другой вариант отображения информации. Если в диалоговом окне выводятся приглашения, клиенту **не следует** представлять каждое из таких приглашений в отдельном окне.

Все клиенты **должны** корректно обрабатывать поле инструкции с символами новой строки. **Следует** также обеспечивать отображение полей имен и приглашений размером не менее 30 символов. Если сервер представляет имена и приглашения размером более 30 символов, клиент **может** укоротить эти поля до поддерживаемого им размера. Если клиент укорачивает поля, он **должен** очевидным образом указать этот факт. Поля инструкции укорачивать **не следует**.

Клиентам **следует** использовать фильтрацию символов управления, как описано в [SSH-ARCH], для предотвращения атак с использованием символов управления в отображаемых полях.

Для каждого приглашения (prompt) соответствующее поле echo указывает, нужно ли отображать введенную пользователем информацию. Клиентам **следует** корректно отображать/маскировать пользовательский ввод для каждого приглашения независимо. Если клиент по тем или иным причинам не способен выполнить требования поля echo, он **должен** маскировать вводимые пользователем символы. GUI-клиенты могут добавлять поле переключения (checkbox) для управления отображением/маскированием. Клиентам **не следует** добавлять какие-либо символы (типа двоеточия) в приглашение, поскольку сервер полностью отвечает за текст, отображаемый пользователю. Клиенты **должны** также воспринимать от пользователя пустые отклики и передавать их в виде пустых строк.

3.4. Информационные отклики

После получения от пользователя запрошенной информации клиент **должен** ответить сообщением SSH_MSG_USERAUTH_INFO_RESPONSE.

Формат SSH_MSG_USERAUTH_INFO_RESPONSE показан на врезке справа.

```
byte      SSH_MSG_USERAUTH_INFO_RESPONSE
int       num-responses
string    response[1] (ISO-10646 UTF-8)
...
string    response[num-responses] (ISO-10646 UTF-8)
```

Отметим, что отклики представляются в кодировке ISO-10646 UTF-8. Интерпретация и проверка откликов определяется сервером. Однако, если клиент считывает отклики пользователя в иной кодировке (например, ISO 8859-1), он **должен** преобразовать их в кодировку ISO-10646 UTF-8 до передачи серверу.

С точки зрения поддержки разных языков желательно, чтобы при обработке пользовательских откликов процесс аутентификации работал независимо от используемой операционной системы и клиентских программ. Это достигается за счет нормализации. Системам, поддерживающим пароли в кодировках, отличных от ASCII, **следует** всегда нормализовать пароли и имена пользователей при добавлении в базу данных и сравнении их (с хэшированием или без него) с имеющимися в базе записями. Реализациям SSH, хранящим и сравнивающим пароли, **следует** использовать нормализацию [SASLPREP].

Если число откликов (num-responses) не соответствует числу запросов (num-prompts), сервер **должен** передать сообщение об отказе.

Если в запросе сервера было 0 полей num-prompts, клиент **должен** передать отклик с нулевым значением num-responses.

---

[1] Введите пароль.
[2] Пароль для пользователя user23@host.domain:
[3] Парольная аутентификация для пользователя user23@host.domain.
[4] Command line interface - командный (текстовый) интерфейс.

Отклики **должны** упорядочиваться в соответствии с порядком приглашений. Т. е., отклик response[n] **должен** быть ответом на приглашение prompt[n].

После получения отклика сервер **должен** передать сообщение об успехе (SSH_MSG_USERAUTH_SUCCESS) или отказе (SSH_MSG_USERAUTH_FAILURE) или дополнительный запрос SSH_MSG_USERAUTH_INFO_REQUEST.

Если сервер отказался аутентифицировать пользователя (на основе применяемого механизма проверки подлинности), ему **не следует** передавать другие запросы с целью получения новой аутентификационной информации. Вместо этого серверу **следует** передать сообщение об отказе. Единственным случаем, когда серверу следует передавать множество сообщений с запросами является потребность в дополнительных аутентификационных данных (т. е., при использовании для проверки подлинности пользователя множества механизмов аутентификации). Если сервер считает нужным передать сообщение об отказе, он **может** задержать его отправку клиенту (время задержки определяется реализацией). Предполагается, что реализации будут позволять настраивать время задержки, предлагаемая по умолчанию задержка составляет 2 секунды.

4. Примеры проверки подлинности

Здесь приведены два примера обмена информацией между клиентом и сервером. В первом случае используется механизм challenge/response с ручным маркером (token). Такая аутентификация невозможна при использовании других методов проверки подлинности.

```
C:   byte      SSH_MSG_USERAUTH_REQUEST
C:   string    "user23"
C:   string    "ssh-userauth"
C:   string    "keyboard-interactive"
C:   string    ""
C:   string    ""

S:   byte      SSH_MSG_USERAUTH_INFO_REQUEST
S:   string    "CRYPTOCard Authentication"
S:   string    "The challenge is '14315716'"[1]
S:   string    "en-US"
S:   int       1
S:   string    "Response: "[2]
S:   boolean   TRUE

[Клиент запрашивает у пользователя пароль]

C:   byte      SSH_MSG_USERAUTH_INFO_RESPONSE
C:   int       1
C:   string    "6d757575"

S:   byte      SSH_MSG_USERAUTH_SUCCESS
```

Во втором случае показана обычная парольная аутентификация с истекшим сроком действия пароля.

```
C:   byte      SSH_MSG_USERAUTH_REQUEST
C:   string    "user23"
C:   string    "ssh-userauth"
C:   string    "keyboard-interactive"
C:   string    "en-US"
C:   string    ""

S:   byte      SSH_MSG_USERAUTH_INFO_REQUEST
S:   string    "Password Authentication"[3]
S:   string    ""
S:   string    "en-US"
S:   int       1
S:   string    "Password: "[4]
S:   boolean   FALSE

[Клиент запрашивает у пользователя пароль]

C:   byte      SSH_MSG_USERAUTH_INFO_RESPONSE
C:   int       1
C:   string    "password"

S:   byte      SSH_MSG_USERAUTH_INFO_REQUEST
S:   string    "Password Expired"[5]
S:   string    "Your password has expired."[6]
S:   string    "en-US"
S:   int       2
```

[1]Запрос «14315716».
[2]Отклик:
[3]Парольная аутентификация.
[4]Пароль:
[5]Срок действия пароля закончился.
[6]Срок действия Вашего пароля закончился.

```
S:   string    "Enter new password: "[1]
S:   boolean   FALSE
S:   string    "Enter it again: "[2]
S:   boolean   FALSE

[Клиент запрашивает у пользователя новый пароль]

C:   byte      SSH_MSG_USERAUTH_INFO_RESPONSE
C:   int       2
C:   string    "newpass"
C:   string    "newpass"

S:   byte      SSH_MSG_USERAUTH_INFO_REQUEST
S:   string    "Password changed"[3]
S:   string    "Password successfully changed for user23."[4]
S:   string    "en-US"
S:   int       0

[Клиент выводит сообщение для пользователя]

C:   byte      SSH_MSG_USERAUTH_INFO_RESPONSE
C:   int       0

S:   byte      SSH_MSG_USERAUTH_SUCCESS
```

5. Согласование с IANA

Для этого метода проверки подлинности используется тип userauth "keyboard-interactive".

В этом методе аутентификации используются две специфичных для метода константы:

```
SSH_MSG_USERAUTH_INFO_REQUEST           60
SSH_MSG_USERAUTH_INFO_RESPONSE          61
```

6. Вопросы безопасности

Протокол и метод аутентификации зависят от защищенности нижележащего транспортного уровня SSH. Без обеспечиваемой этим уровнем конфиденциальности данные аутентификации, передаваемые с использованием этого метода могут быть перехвачены.

Число обменов данными между клиентом и сервером в процессе аутентификации может меняться. Вполне возможно, что наблюдатель сможет получить ценную информацию, просто посчитав число пакетов. Например, наблюдатель может догадаться, что у пользовательского пароля истек срок действия, а дальнейшие наблюдения могут позволить определить время действия паролей, раскрывая частично политику сайта.

7. Литература

7.1. Нормативные документы

[RFC-2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119[5], March 1997.

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

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

[SSH-ARCH] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Protocol Architecture", RFC 4251[5], January 2006.

[SSH-USERAUTH] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Authentication Protocol", RFC 4252[5], January 2006.

[SSH-TRANS] Ylonen, T. and C. Lonvick, Ed., "The Secure Shell (SSH) Transport Layer Protocol", RFC 4253[5], January 2006.

[SASLPREP] Zeilenga, K., "SASLprep: Stringprep Profile for User Names and Passwords", RFC 4013, February 2005.

7.2. Дополнительная литература

[PAM] Samar, V., Schemers, R., "Unified Login With Pluggable Authentication Modules (PAM)", OSF RFC 86.0[5], October 1995.

Адреса авторов

Frank Cusack

savecore.net

[1]Введите новый пароль:
[2]Введите пароль еще раз:
[3]Пароль изменен.
[4]Пароль для пользователя user23 изменен.
[5]Перевод этого документа доступен на сайте www.protocols.ru. *Прим. перев.*

EMail: frank@savecore.net

Martin Forssen

AppGate Network Security AB

Otterhallegatan 2

SE-411 18 Gothenburg

SWEDEN

EMail: maf@appgate.com

Перевод на русский язык

Николай Малых

nmalykh@gmail.com

Полное заявление авторских прав

Интеллектуальная собственность

Подтверждение

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