Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина» (УрФУ)

Институт радиоэлектроники и информационных технологий – РтФ

Школа профессионального и академического образования

Отчет по дисциплине   
«Криптографические методы защиты информации»

Лабораторная работа №5  
«*SSH* и *SFTP*. Настройка и организация работы. Как обеспечить безопасность сервиса. Анализ трафика»

Студенты: Клоченко И.Е.

Преподаватель: Соколов И.П.

Группа: РИ-300024

Екатеринбург

2023

**Оглавление**

[Ход работы: 3](#_Toc134741267)

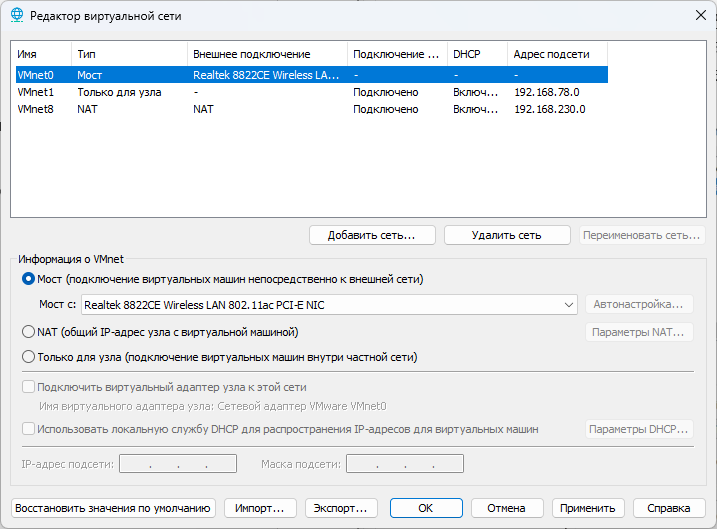
[Вывод 17](#_Toc134741268)

**Цель работы:**

Знакомство с протоколом *SSH* и протоколом передачи файлов *SFTP*.

## Ход работы:

1. Запускаем *VMware*. В настройках виртуального сетевого редактора добавляем новую виртуальную сеть с типом «Мост» и сетевым адаптером, который используется на основной машине.



**Рисунок 1**. Создание новой виртуальной сети в VMware

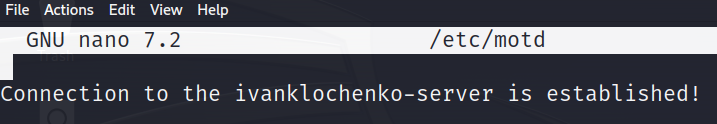
1. В лабораторной работе будет использоваться дистрибутив *Kali* *Linux*. Приглашения оболочки терминала виртуальных машин будут отличаться наличием или отсутствием постфикса -*server*.



**Рисунок 2**. Созданные виртуальные машины

В настройках сетевого адаптера виртуальных машин выставляем в сетевом соединении «Мост».

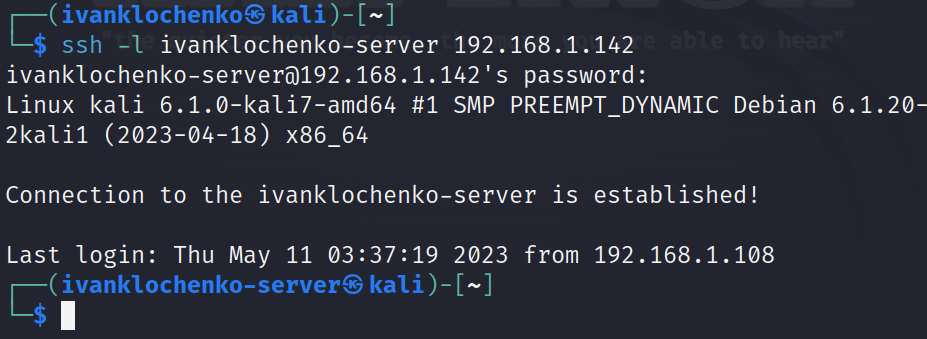
1. Запускаем обе виртуальные машины.
2. Настраиваем сетевое окружение таким образом, чтобы виртуальные машины видели друг друга в сети. Вычисляем *IP* адреса виртуальных машин в локальной сети (*client* – 192.168.1.108, *server* – 192.168.1.142).
3. Создаем приветственное сообщение при успешном соединении по *SSH* (на сервере).



**Рисунок 3**. Создание приветственного сообщения при подключении по SSH

Важно отметить, что за приветственное сообщение об успешном подключении по *SSH* отвечает не /*etc*/*issue*, который был указан в методичке, а другие файлы, например, /*etc*/*motd*.

1. Проверяем возможность подключения по *SSH* (клиент).



**Рисунок 4**. Проверка подключения клиентом к серверу по SSH

Важно отметить, что, если служба *SSH* не была установлена ранее, ее необходимо:

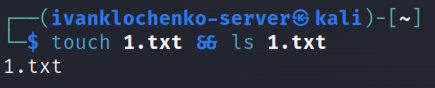
* Установить (*sudo* *apt* *install* *ssh*)
* Запустить (*service* *ssh* *start*)

В противном случае подключение будет уходить в ошибку «*Connection* *refused*».

Также, установить соединение используя только *IP* адрес не получится, потому что *SSH* будет подставлять в форму «*login*@*host*» на место *login* текущее имя пользователя в системе, что не равно имени пользователя, от лица которого мы пытаемся установить соединение *SSH*. Поэтому при подключении нужно либо использовать указанную на скриншоте команду, либо форму «*login*@*host*».

В противном случае будет выдаваться ошибка «*Permission* *denied*».

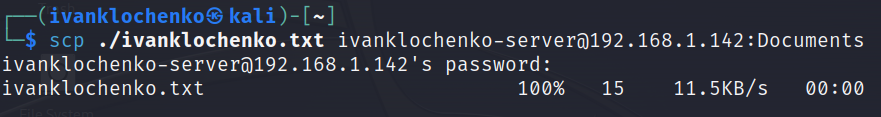
1. Создаем файл 1.*txt* на сервере:



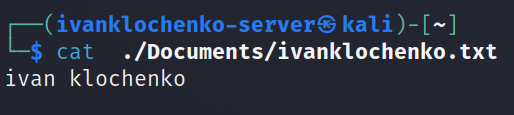
**Рисунок 5**. Создание файла на сервере.

Стоит отметить, что данный метод проверки успешного соединения не самый логичный.

1. Попробуем переместить файл. Создаем файл на клиенте с ФИО в названии и содержании. Далее с помощью команды *scp* перемещаем файл на сервер.



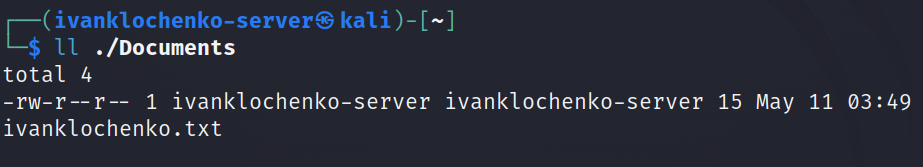
**Рисунок 6**. Перемещение файла на сервер через scp



**Рисунок 7**.Просмотр содержимого файла на сервере

Файл был успешно отправлен.

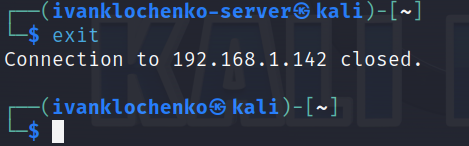
1. Просмотрим права доступа у файла.



**Рисунок 8**. Просмотр прав доступа у перемещенного файла

Как видно, файл имеет стандартные (644) права доступа. Владельцем файла выступает принимающая сторона – *ivanklochenko*-*server*.

1. Разрываем соединение между клиентом и сервером командой *exit*.

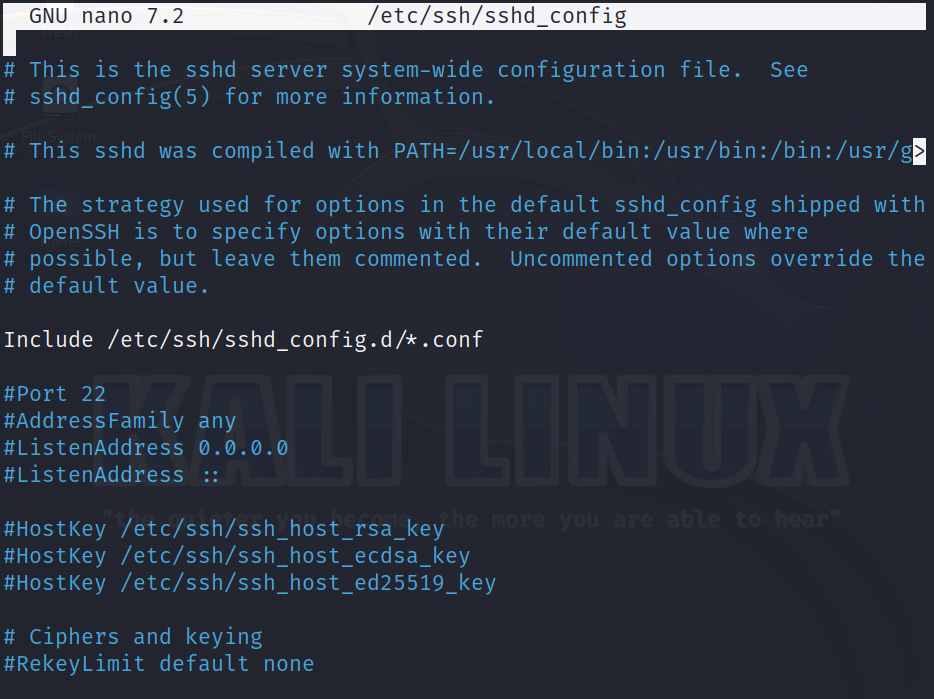


**Рисунок 9**. Разрыв соединения между клиентом и сервером

1. Открываем для редактирования файл (/*etc*/*ssh*/*sshd\_config*), содержащий настройки сервера *SSH*.

Следующие опции в файле конфигурации мне знакомы:

* *Banner* – отвечает за баннер, который выводится пользователю до запроса пароля.
* *usePAM* *yes* – позволяет *sshd* выполнять аутентификацию по паролю без прав суперпользователя.



**Рисунок 10**. Начало файла конфигурации sshd

1. Поменяем порт подключения по умолчанию на 7777 порт.

*Port* 7777

1. Включим использование только 2 версии протокола *SSH*.

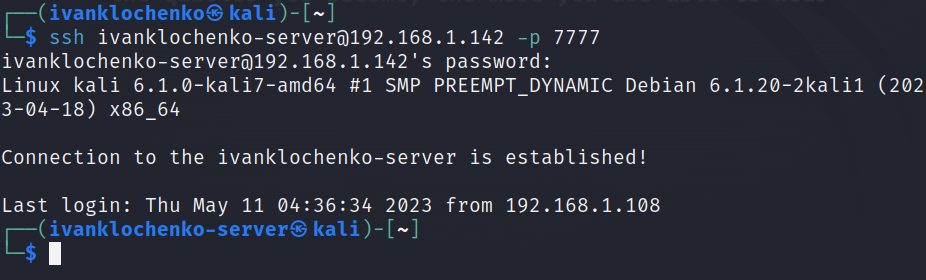
*Protocol* 2

1. Запрещаем *root*-доступ по *SSH*.

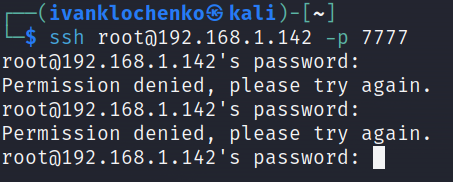
*PermitRootLogin* *no*

1. Перезапускаем сервер.

Все внесенные изменения вступили в силу.

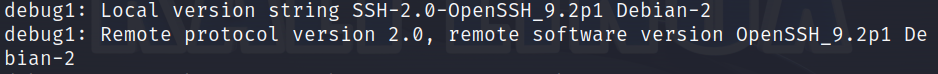


**Рисунок 11**. Изменение порта по умолчанию



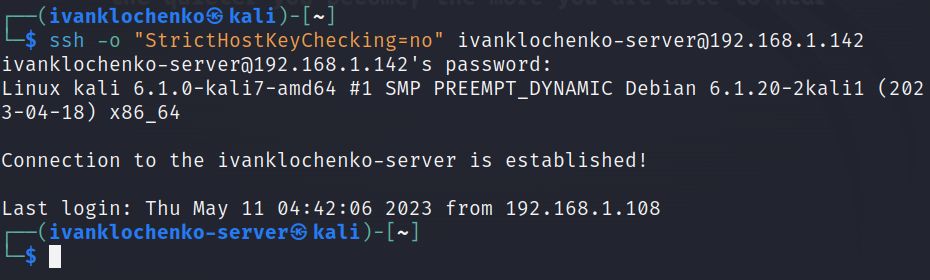
**Рисунок 12**. Запрет root подключения по SSH

Чтобы посмотреть версию используемого протокола *SSH* нужно при подключении указать флаг -*v*, что запустит режим отладки, в котором среди прочего будет указана версия протокола:



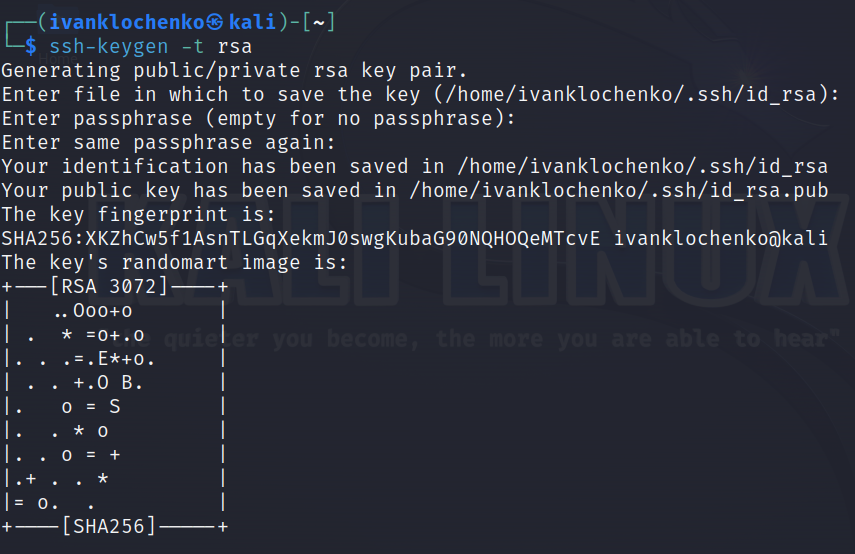
**Рисунок 13**. Изменение версии протокола

1. Восстанавливаем настройки по умолчанию в файле конфигурации *SSH* сервера.
2. Отключаем проверку отпечатка ключа *SSH*.



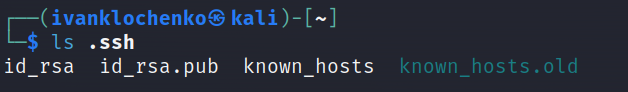
**Рисунок 14**. Отключение отпечатка ключа SSH

1. (Клиент) Сгенерируем пару *RSA* ключей. Устанавливаем парольную фразу для ключа.



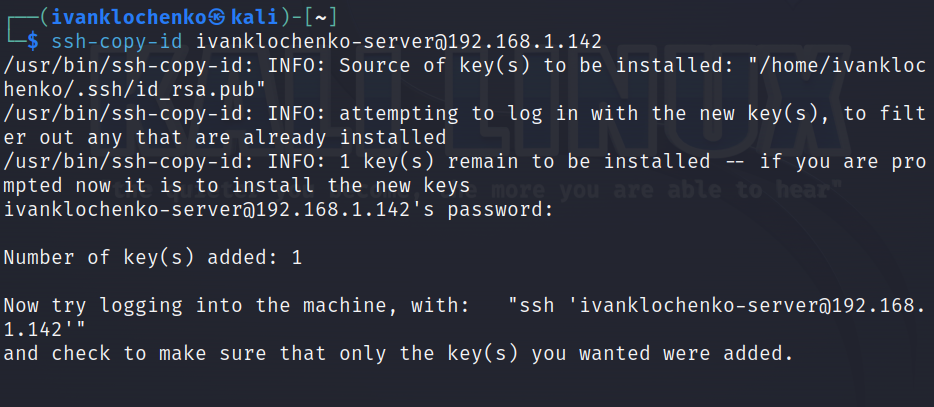
**Рисунок 15**. Генерация пары RSA ключей

1. Проверяем созданные ключи:

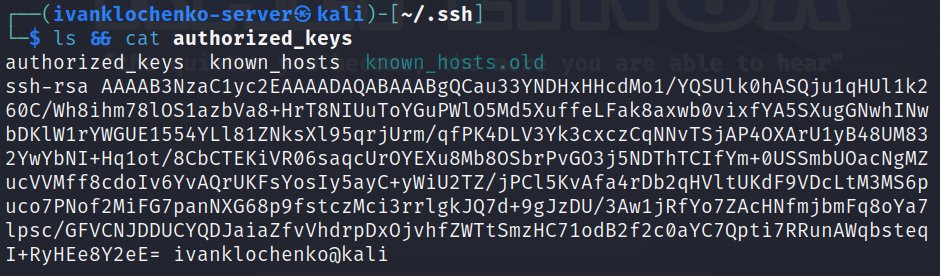


**Рисунок 16**. Созданные RSA ключи

1. Скопируем .*pub* файл на компьютер сервера.

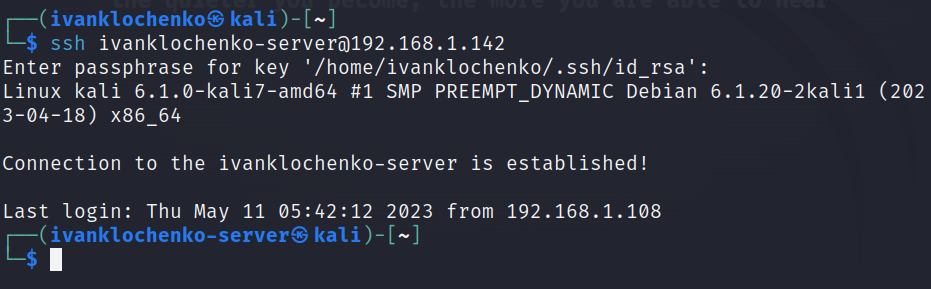


**Рисунок 17**. Копирование публичного ключа на сервер



**Рисунок 18**. Проверка ключа на сервере

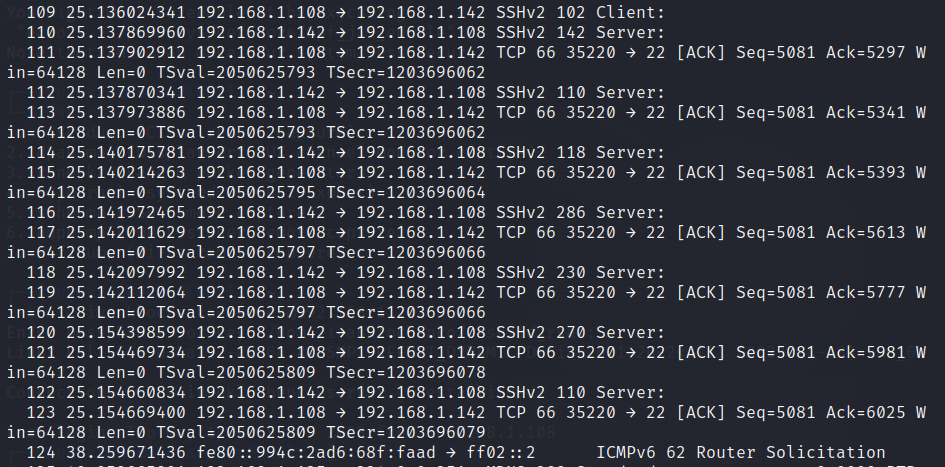
1. Выполним подключение с помощью клиента *SSH*.



**Рисунок 19**. Подключение по SSH с ключом RSA

Пароль от учетной записи не требуется, но требуется парольная фраза, которую мы установили ранее. Пароль не требуется, потому что вместо обычной парольной аутентификации используется криптографическая аутентификация по приватному ключу клиента.

1. (Клиент) Войдем с правами суперпользователя и установим программу *tshark* и обновим ее модули.
2. Просматриваем список доступных интерфейсов. Среди них есть те, с которыми я уже знаком: *udp*-,*ssh*-,*wifi*- *dump*.
3. Устанавливаем *SSH*-соединение между клиентом и сервером.
4. Проанализируем входящий трафик.



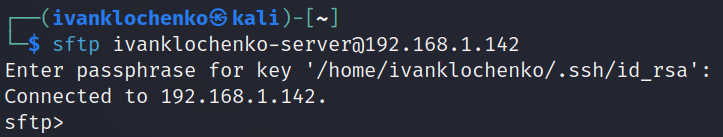
**Рисунок 20**. Входящий трафик

1. Сформируем и проанализируем трафик, отличный от *SSH*.

Сходства: использование *TCP*-соединения, обмен данными между клиентом и сервером, и отправку и получение пакетов.

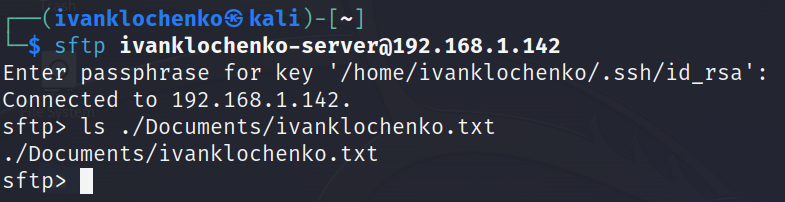
Различия: пакеты *SSH* зашифрованы. Это может и не быть различием, если сравнивать, например, с пакетами *SSL*/*TLS*/*HTTP*2.

1. Подключаемся к серверу по *SFTP*.



**Рисунок 21**. Подключение к серверу по SFTP

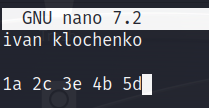
1. Находим файл с ФИО, созданный ранее.



**Рисунок 22**. Файл ivanklochenko.txt, созданный ранее и найденный через подключение sftp

1. Установим соответствие между командами, используемыми в *SFTP*.

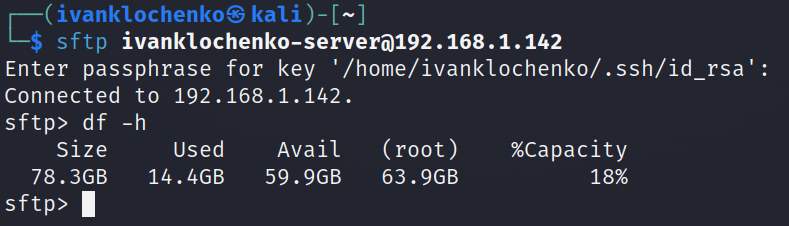
Ответ записываем в файл *ivanklochenko*.*txt* в одну строку.



**Рисунок 23**. Соответствие команд SFTP

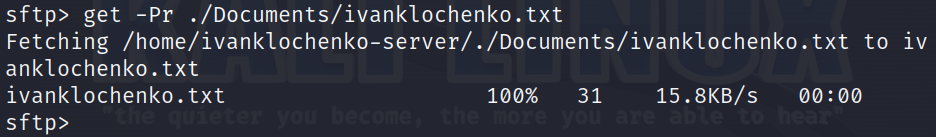
Важно отметить, что запись в файл по SFTP подключению невозможна, поэтому записываем в ВМ *ivanklochenko-server*.

1. Проверяем, достаточно ли места для передачи нужного нам файла.



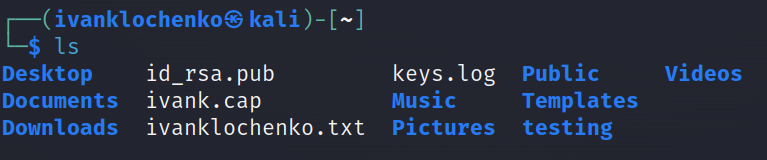
**Рисунок 24**. Проверка достаточности места.

1. Получаем файл с моей фамилией.



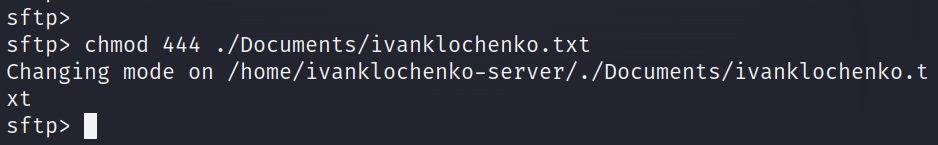
**Рисунок 25**. Получение файла через SFTP

1. Проверяем корректность передачи файла.



**Рисунок 26**. Проверка корректности передачи файла

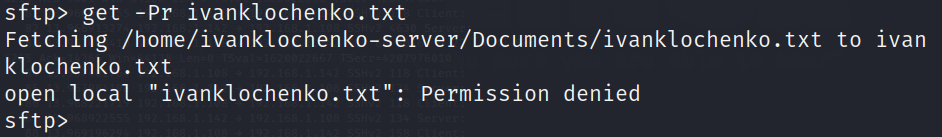
1. Выставим права только на чтение на файл с моей фамилией, который находится на сервере.



**Рисунок 27**. Изменение прав на файл

Пароль при изменении прав не был запрошен, потому что, как было сказано ранее, после копирования файла (команда *scp* на предыдущих шагах) владельцем файла стала принимающая сторона, то есть *ivanklochenko*-*server*, от лица которого выполнялось дальнейшее изменение прав.

1. Скопируем файл с новыми правами на компьютер клиента через *SFTP*.



**Рисунок 28**. Копирование файла с новыми правами

Файл не удалось скопировать. Запрещено.

1. Попробуем записать что-либо в файл, находясь в *SFTP* сессии.

Примечание. Если в задании имелось ввиду записать в файл что-либо именно во время сессии *SFTP*, то это невозможно, потому что функционале команд в *SFTP* нет необходимой команды. Будучи в оригинальной сессии на сервере, это не получится сделать, потому что установлены права только на чтение для владельца и группы-владельца файла.

*SFTP* соединение не безопасно, человек, подключающийся по *SFTP* может изменить права доступа на файлы, находящиеся на удаленном сервере, чем лишит эти данные одного из ключевых свойств защищенной информации – доступности. Разумеется, это возможно при условии, что злоумышленник знает парольную фразу для соединения по SFTP.

1. Проанализируем трафик, создаваемый *SFTP* соединением.

Отличия от трафика *SFTP*:

Порты: В пакетах *SSH* обычно используется порт 22 (или другой, если настройки были изменены), который указывает на установку соединения *SSH*. В пакетах *SFTP* также может присутствовать порт 22 в поле источника или назначения, но в них также будет указан дополнительный порт, который *SFTP* использует для передачи файлов.

Протоколы: В пакетах *SSH* присутствуют заголовки и данные, связанные с *SSH*-протоколом, включая приветственные сообщения, алгоритмы шифрования, ключи и команды, отправленные и полученные во время сеанса *SSH*. В пакетах *SFTP* можно обнаружить команды и ответы, связанные с управлением файлами, например, запросы на чтение, запись, создание и удаление файлов, а также передачу содержимого файлов.

Шифрование: В пакетах *SSH* можно наблюдать обмен данными, зашифрованными с помощью алгоритмов шифрования, определенных в протоколе *SSH*. В пакетах *SFTP* можно обнаружить зашифрованные данные, относящиеся к передаче файлов, чтобы обеспечить конфиденциальность содержимого файлов во время передачи.

Данные файла: В пакетах *SFTP* могут присутствовать данные файлов, передаваемых между клиентом и сервером. В них можно обнаружить содержимое файлов или их фрагменты, которые передаются в соответствующих пакетах *SFTP*.

1. Завершаем *SFTP* сессию.

Преимущества *SFTP* соединения:

Безопасность: *SFTP* использует *SSH* для шифрования и аутентификации, что обеспечивает безопасную передачу файлов по открытой сети. Весь трафик, включая содержимое файлов и учетные данные, шифруется, что защищает их от перехвата информации.

Интеграция с *SSH*: *SFTP* интегрирован с протоколом *SSH*, что означает, что вы можете использовать те же учетные данные и механизмы аутентификации, что и для *SSH*. Это упрощает управление доступом и обеспечивает единый механизм аутентификации для удаленного доступа и передачи файлов.

Поддержка стандартных команд: *SFTP* поддерживает стандартные команды для управления файлами, такие как создание, чтение, запись, переименование и удаление файлов и директорий. Это обеспечивает удобный интерфейс для работы с файлами на удаленном сервере.

Передача больших файлов: *SFTP* позволяет передавать файлы любого размера, включая большие файлы, без ограничений. Это особенно полезно при работе с файлами больших объемов, например, при резервном копировании или обмене медиафайлами.

Интуитивный интерфейс: *SFTP*-клиенты обычно предоставляют графический интерфейс или командную строку с понятными командами и функциями. Это делает работу с удаленными файлами простой и интуитивно понятной.

Поддержка перезапуска передачи: Если передача файла прерывается, *SFTP* позволяет возобновить передачу с места остановки, минимизируя потери времени и ресурсов.

Расширенные функции: *SFTP* поддерживает дополнительные функции, такие как сжатие данных, управление правами доступа к файлам и учетными записями, а также поддержку прогресса передачи и событий.

## Вывод

В ходе выполнения лабораторной работы было проведено знакомство с протоколами *SSH* и *SFTP*. Было настроено две виртуальных машины клиент – сервер для практического использования этих протоколов.

Были рассмотрены настройки файла конфигураций *SSH* с незначительным изменением и последующим анализом изменений при работе протокола.

Была создана пара *RSA* ключей для более безопасного *SSH* соединения. Выяснен механизм аутентификации при соединении по протоколу *SSH* в случае ввода пароля и в случае использования приватного ключа для подключения к сессии.

Освоены навыки копирования файлов с клиента на удаленный сервер и с удаленного сервера на клиент через утилиту scp и протокол SFTP.

Освоен механизм защиты от копирования файлов через *SFTP* путем установления прав доступа на файлы.

Был практически применен в работе инструмент *tshark* для анализа трафика *SSH* и *SFTP* соединений. Были обозначены различия в трафике обоих протоколов.

Были проанализированы преимущества использования *SFTP*.