# Практическое занятие: Защита данных.

Целью занятия является формирование и закрепление навыков по защите данных при работе с сетью в Linux системах.

Задачи:

1. Изучить возможности защиты передаваемых по открытой сети файлов в Linux.

2. Получить практические навыки по работе с системами шифрования.

**Необходимые теоретические сведения.**

Шифрование с открытым ключом

PGP (Pretty Good Privacy) - компьютерная программа, которая позволяет выполнять операции шифрования/дешифрования и цифровой подписи файлов или сообщений, а так же другой информации, представленной в электронном виде, в том числе шифрование данных на запоминающих устройствах. Алгоритм шифрования с открытым ключом показан на рис.1.

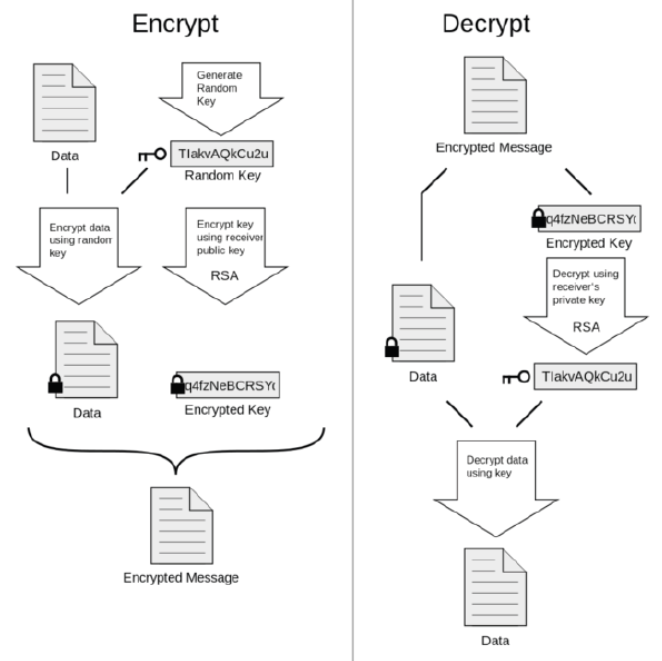


Рис. 1. Алгоритм работы шифрования с открытым ключом

Процесс шифрования в PGP проходит в несколько этапов: хеширование, сжатие данных, шифрование с симметричным ключом, и, наконец, шифрованием с открытым ключом. Причём каждый этап может использовать разные алгоритмы. Так симметричное шифрование производится с использованием одного из семи симметричных алгоритмов (AES, Blowfish, 3DES, CAST5, IDEA,Twofish, Camellia) на сеансовом ключе. Сеансовый ключ в свою очередь генерируется с использованием криптографически стойкого генератора псевдослучайных чисел. Он зашифровывается открытым ключом получателя с использованием алгоритмов RSA или Elgamal (в зависимости от исходного открытого ключа получателя).

Изначально PGP разрабатывалась для защиты электронной почты на стороне клиента, но начиная с 2002 года также включает в себя шифрование жёстких дисков, директорий, файлов, сессий программ мгновенного обмена сообщениям, защиту файлов и директорий в сетевых хранилищах, пакетной передачи файлов, а в новых версиях – шифрование HTTP-запросов и ответов на стороне сервера и клиента.

**Практическая часть**

Для контроля содержимого двоичного файла применяется команда hexdump. Она выводит на экран файл, переданный ей в качестве параметра в виде шестнадцатеричных кодов. Если указать ещё и параметр –C, то также будет выведено и содержание файла в виде ASCII символов.

Команда file определяет тип файла. Единственным параметром команды является имя файла. В ответ команда выводит тип файла (текстовый, графический, архив и т.п.).

Программа gpg позволяет зашифровать файлы, хранящиеся в пределах одной машины. Для этого существует режим шифрования с паролем. Команда gpg -c имя\_файла позволяет зашифровать файл. Будет запрошен пароль, который нужно запомнить и указать при расшифровке. Чтобы другой пользователь мог расшифровать данный файл, ему нужно передать пароль. Зашифрованный файл получает расширение .gpg. Для расшифровки нужно использовать:

gpg -d имя\_файла

Будет запрошен пароль и файл будет выведен на экран. Чтобы получить сразу файл, нужно использовать перенаправление ввода-вывода. Данный способ шифрования требует передачи пароля для обмена файлами. Пароль может быть перехвачен во время передачи, что снижает надёжность данного способа.

Обмен зашифрованными файлами при помощи GnuPG не требует передачи пароля. Вместо него передаётся специальный открытый ключ, который безопасно передавать по незащищённому каналу. Сначала создадим ключи, которые нам потребуются для шифрования и подписи. Для этого в терминале необходимо ввести команду:

gpg --gen-key

GnuPG спросит какой тип ключа вы хотите создать. Выберем пункт (1) RSA and RSA, т.е. алгоритм RSA будет использоваться как для шифрования секретного ключа симметричного алгоритма шифрования, так и для цифровой подписи. В качестве алгоритмов симметричного шифрования используются IDEA, CAST и AES. Далее программа попросит указать размер ключа. Рекомендуется указать максимально большую длину ключа: 4096 - для увеличения надежности шифрования. Далее программа спросит о времени жизни создаваемого ключа. Выберем вариант, стоящий по умолчанию - неограниченное по времени использование ключа. Далее последует вопрос о вашем имени и email. Введите их. После этого программа предложит написать комментарий: писать его не обязательно. Далее программа попросит ввести парольную фразу (т.е. любой надежный пароль, в котором могут присутствовать пробелы). После того, как вы введете все запрашиваемые данные, программа начнет генерировать ключи, что может занять несколько минут. После завершения процесса генерации ключей вы получите сообщение наподобие нижеприведенного:

pub 2048R/2E8F71E6 2013-06-24

Key fingerprint = 343D 6A32 EB8C 4995 4C9A 22EC 0206 EE26 2E8F 71E6

uid test\_name (no comment) <test@mail.ru>

sub 2048R/561FDA97 2013-06-24

Теперь можно посмотреть список ключей при помощи команды:

gpg --list-keys

Чтобы зашифровать данные и передать их другому пользователю, нужно экспортировать ключ в файл \*.asc, который нужно переслать принимающей стороне. Экспорт делается следующей командой:

gpg --armor --output user\_name.asc --export user\_name

где user\_name - ваше имя, которое вы указывали на этапе генерации ключей. В результате выполнения этой команды в текущей директории появится файл user\_name.asc, хранящий ваш открытый ключ. Этот файл вы и должны любым удобным способом передать людям, от которых собираетесь получать шифрованные сообщения.

Человек, который собирается от вас получать шифрованные сообщения, должен проделать точно такие действия и предоставить вам свой открытый ключ. Предположим, что этот ключ он назвал other\_user\_name.asc. Теперь вам необходимо импортировать предоставленный вам ключ в базу ключей GnuPG. Для используется команда:

gpg --import other\_user\_name.asc

После того, как вы импортировали себе ключ other\_user\_name.asc, вы должны проверить достоверность ключа, который Вы добавили себе. В случае успешной проверки требуется ключ подписать. Проверка достоверности ключа производится с помощью команды fpr , введенной после ввода в терминале команды

gpg --edit-key other\_user\_name

После ввода этой команды вы увидите надпись

Command>

после которой нужно ввести команду fpr. Подписать ключ можно с помощью команды sign

Теперь, когда переданный вам открытый ключ подписан и проверен, вы можете шифровать файл (пусть это будет файл test). Сделать это можно следующей командой:

gpg --output test.asc --encrypt --sign --recipient user\_name test

где test.asc - имя зашифрованного файла. Далее вы можете передавать зашифрованный файл test.asc пользователю user\_name. Пользователь user\_name в свою очередь с помощью вашего открытого ключа может зашифровать какой-нибудь файл и послать его вам.

Для того, чтобы вам расшифровать файл other\_test.asc, запустите в терминале следующую команду:

gpg --output other\_test --decrypt other\_test.asc

При выполнении команды, программа потребует у вас парольную фразу, которую нужно будет ввести, после чего в текущей директории появится расшифрованный файл other\_test, а в окне терминала появится сообщение, содержащее некоторые данные о подписи отправителя, а также информация о том, что подпись верна:

gpg: Signature made Mon 24 Jun 2013 10:13:49 PM MSK using RSA key ID AB735B42

gpg: Good signature from "other\_user <other\_user@mail.ru>"

# Задание для закрепления практических навыков:

1. При помощи hexdump вывести на экран несколько файлов размером 0-2 кБ. Использовать команду с и без параметра -C. При помощи команды file проконтролировать несколько текстовых и графических файлов.

2. Зашифровать при помощи gpg текстовый файл. Убедиться при помощи ls, что создан файл с расширением gpg. Каков размер исходного файла и зашифрованного файла (Узнать при помощи ls -l)? При помощи file узнать тип зашифрованного файла. Расшифровать файл, вывести результат в файл на диске или на экран.

3. Обменяться зашифрованными файлами с соседним компьютером. Для этого:

* Сгенерировать ключ RSA-2048 при помощи gpg, экспортировать файл ключа, проверить содержимое файла ключа при помощи текстового редактора.
* Обменяться ключами с соседним компьютером, проверить список ключей, сверить полученный отпечаток ключа, подписать ключ.
* Зашифровать какой-либо файл, предназначенный для соседнего компьютера и передать его.
* Расшифровать полученный файл.