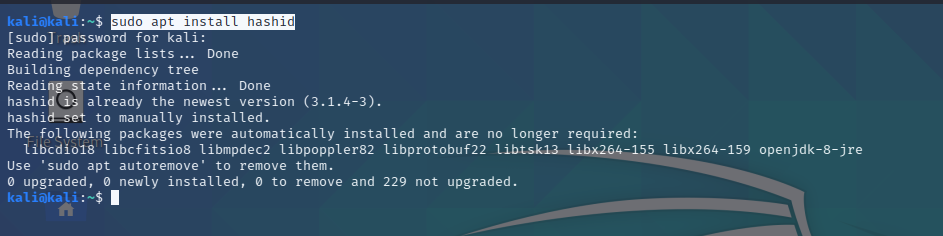
Святной Александр БАСО-02-20

**Электронный отчет по практической работе 5**

# Определение типа hash

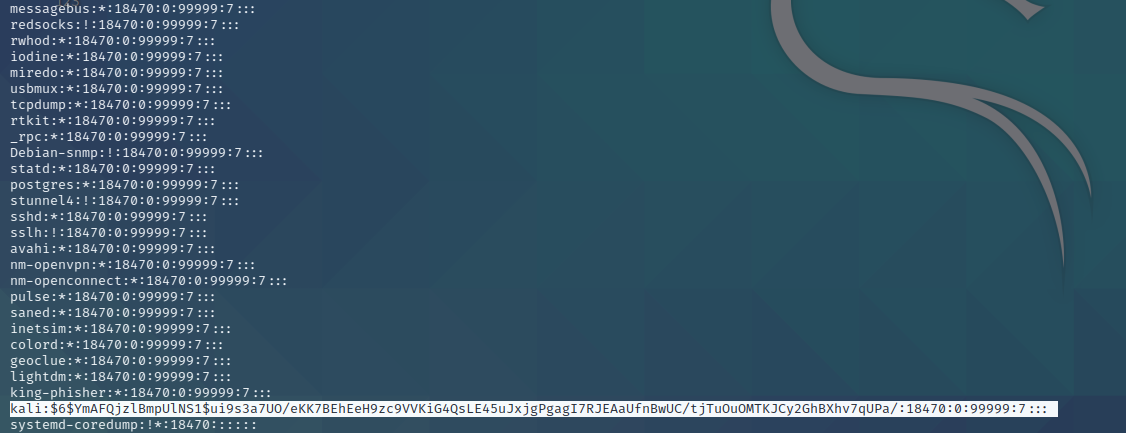
Для начала установим утилиту *hashid*:

$ sudo apt install hashid



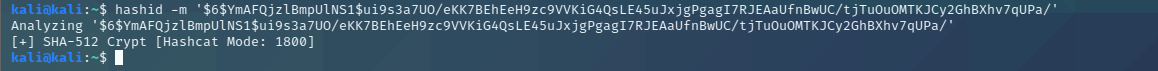
Далее достанем хэш из файла */etc/shadow*:

$ sudo cat /etc/shadow



Узнаем тип хэша:

$ hashid -m 'свой хэш'



Получается, что тип хэша – SHA-512 (ID - $6, Длина хэша пароля – 86 символов)

# Создание hash в Linux

## Создание MD5 Хэш пароля

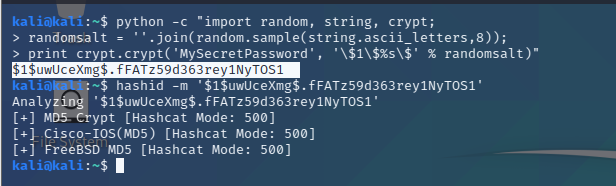
Вводим в терминале:

$ python -c "import random,string,crypt;

> randomsalt = ''.join(random.sample(string.ascii\_letters,8));

> print crypt.crypt('MySecretPassword', '\$1\$%s\$' % randomsalt)"

Полученный хэш сразу же проверяем в *hashid*:

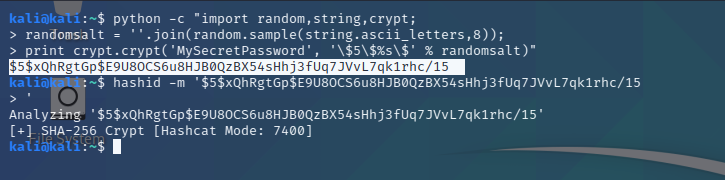


## Создание SHA-256 Хэш пароля

$ python -c "import random,string,crypt;

> randomsalt = ''.join(random.sample(string.ascii\_letters,8));

> print crypt.crypt('MySecretPassword', '\$5\$%s\$' % randomsalt)"

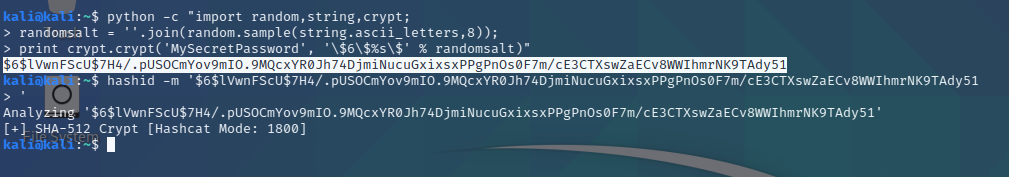


## Создание SHA-512 Хэш пароля

$ python -c "import random,string,crypt;

> randomsalt = ''.join(random.sample(string.ascii\_letters,8));

> print crypt.crypt('MySecretPassword', '\$6\$%s\$' % randomsalt)"

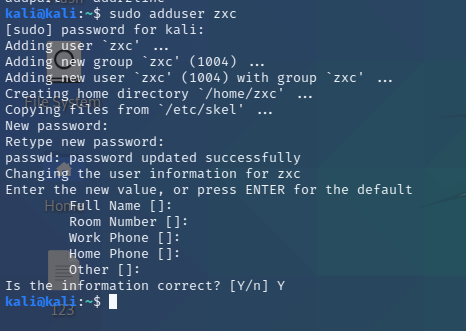


# mkpasswd

Добавим нового пользователя в систему:

$ sudo adduser zxc

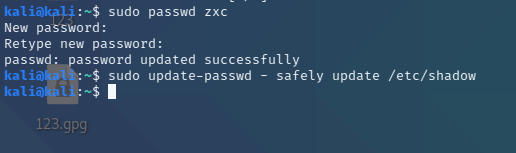
Вводим пароль от учетной записи, информацию оставляем по дефолту и подтверждаем ее правильность:



Обновляем пароль:

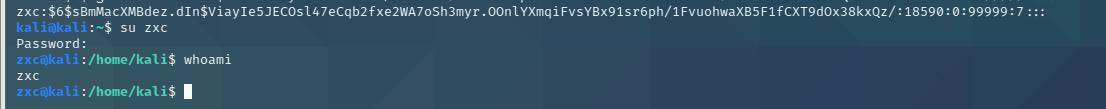
$ sudo passwd zxc

$ sudo update-passwd - safely update /etc/shadow



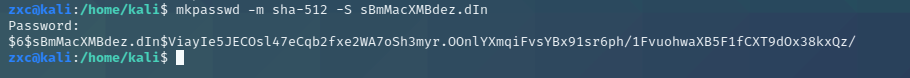
Логинимся под новым пользователем:

$ su zxc



Рассчитываем хэш, зная соль и пароль:

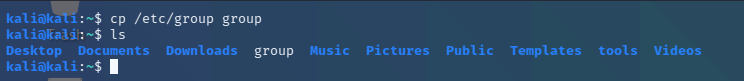
$ mkpasswd -m sha-512 -S sBmMacXMBdez.dIn



# Проверка контрольных сумм

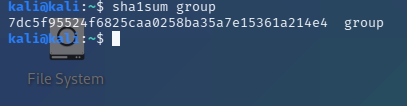
Скопируем файл /etc/group в домашнюю папку:

$ cp /etc/group group



Подсчитаем контрольную сумму:

$ sha1sum group



Сохраняем контрольную сумму в файл:

$ sha1sum group> group.sha1

И проверяем:

$ sha1sum -c group.sha1



Изменяем файл и выполняем проверку:



Контрольные суммы не совпали – файл был подвергнут изменениям.

# GPG

## Шифрование с помощью пароля

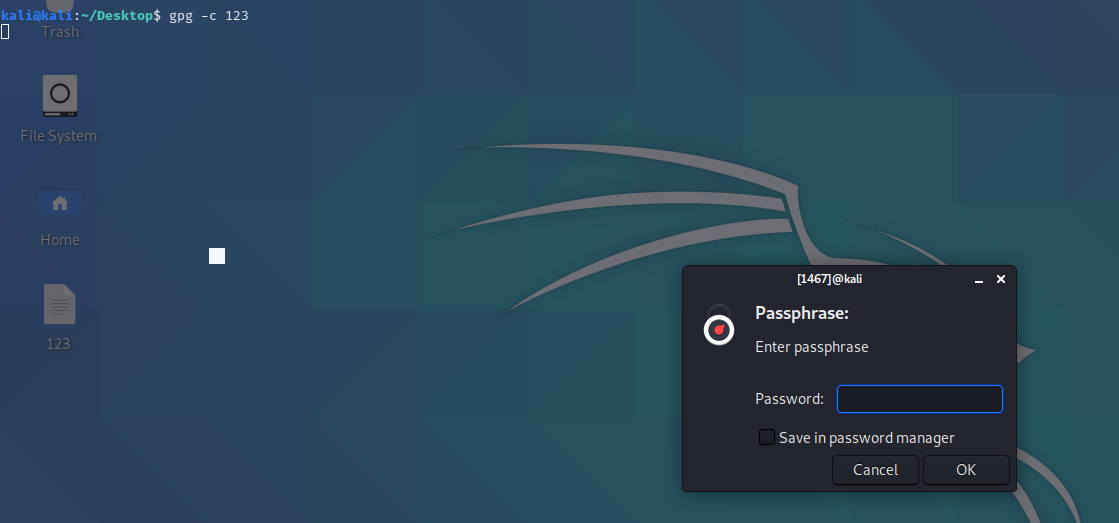
Создадим текстовый документ:

$ touch 123

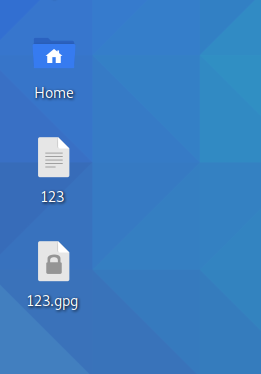
Шифруем:

$ gpg -c 123

Где -c – зашифровать данные с помощью пароля.



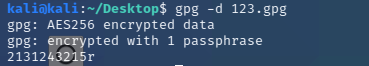
Устанавливаем пароль и видим, что создался файл с расширением gpg.



Расшифровываем:

$ gpg -d 123.gpg

-d – расшифровать данные, зашифрованные с помощью ключа или пароля.

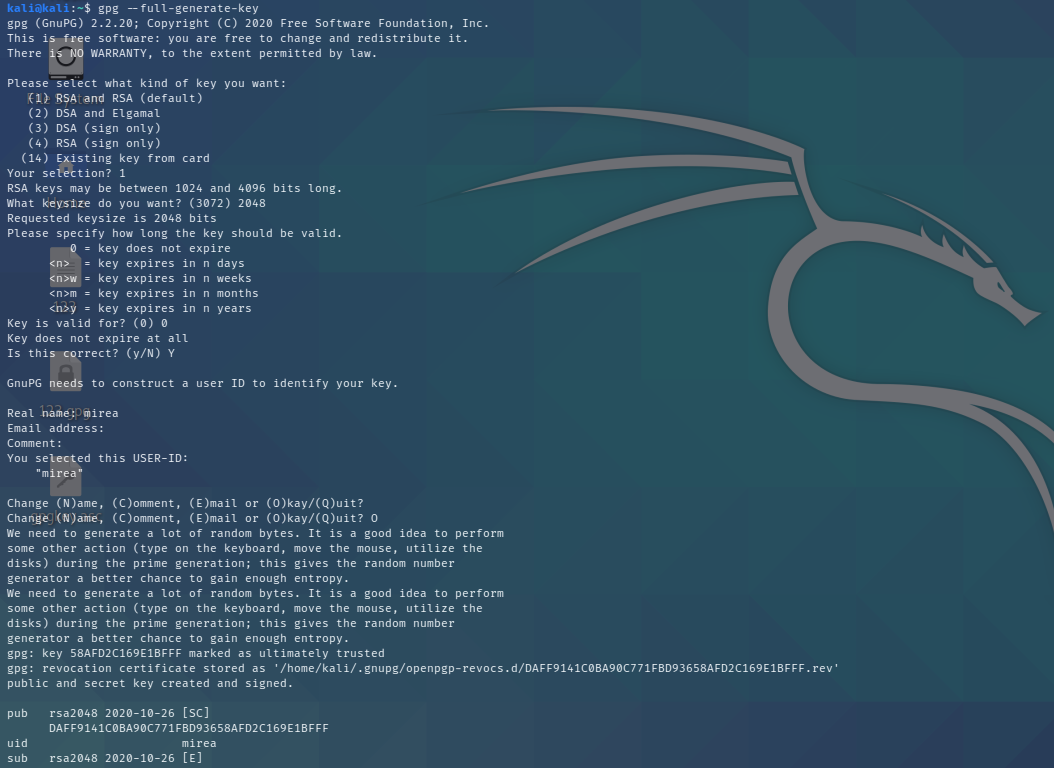


В последней строчке вывелось содержимое текстового документа.

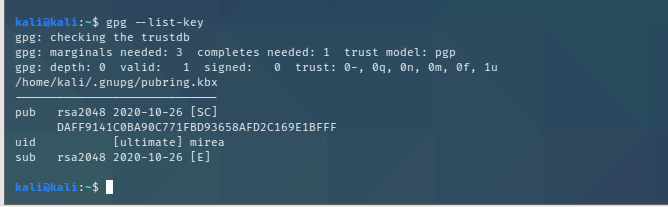
## Шифрование с использованием ключей

Создаем ключ:

$ gpg --full-generate-key



Видим наш ключ в списке доступных:



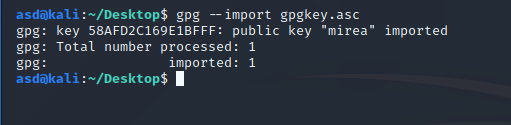
Кроме того, ключ можно экспортировать:

$ gpg -a -o gpgkey.asc --export mirea



Отправив ключ на другое устройство, его необходимо импортировать:

$ gpg --import gpgkey.asc

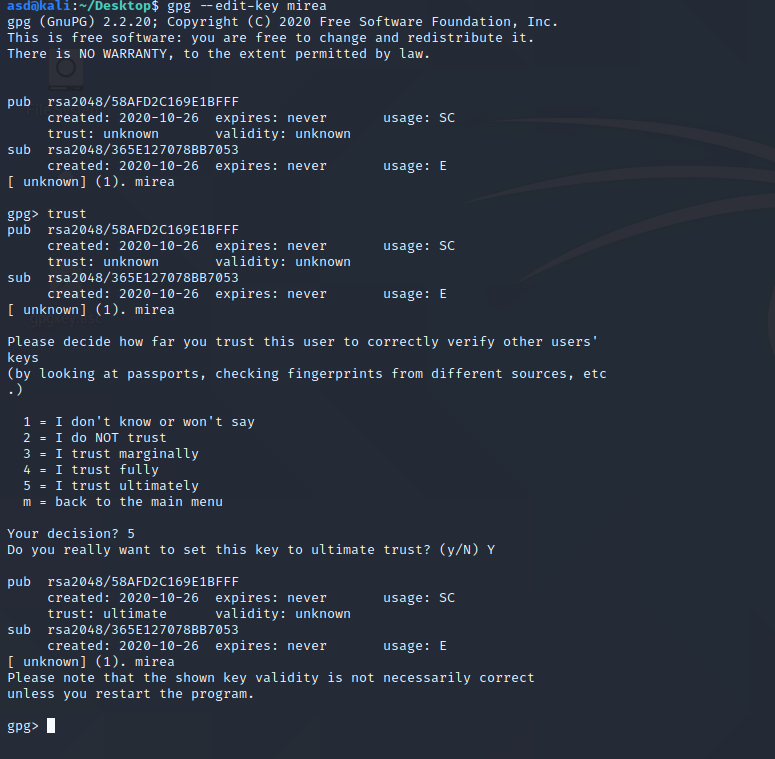


Теперь необходимо изменить уровень доверия к ключу, потому что при каждом шифровании будет всплывать запрос о доверии пользователя к этому ключу. Воспользуемся командой

$ gpg --edit-key mirea

Для выбора уровня доверия требуется ввод команды trust:

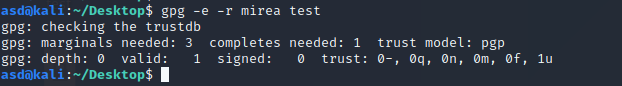
gpg> trust



Выставляем максимальный уровень доверия (5).

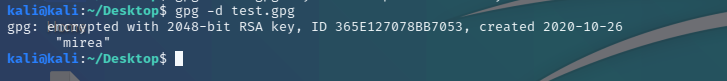
Создаем и шифруем пустой текстовый файл:

$ gpg -e -r mirea test



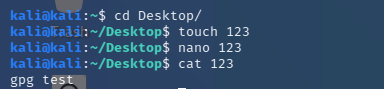
Переносим его на другую учетную запись и расшифровываем:

$ gpg -d test.gpg



## Подписи и шифрование

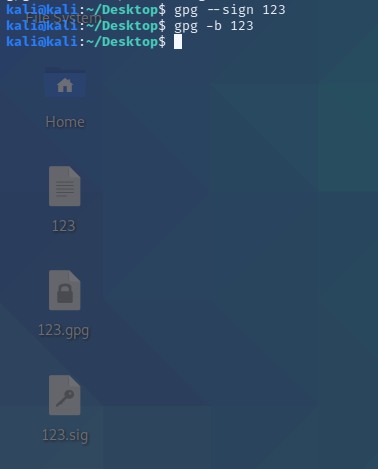
Создадим файл со следующим содержимым:



Создаем подпись и экспортируем ее в отдельный файл:

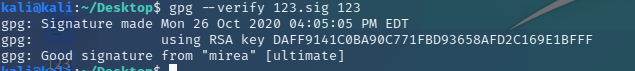
$ gpg –sign 123

$ gpg -b 123

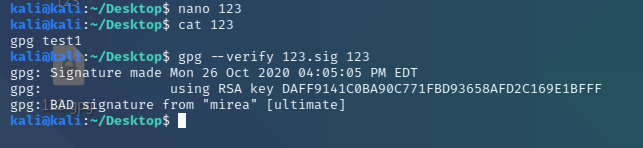


Проверим подпись:

$ gpg --verify 123.sig 123



Далее изменим содержимое файла и проверим подпись:



Получаем сведения о том, что подпись плохая, то есть файл был подвергнут изменениям.

