Криптография

Предисловие

Я хочу добиться высокой степени безопасности своей системы. Как и какими средствами мне этого достичь?

Достижение любой цели не обходится без определённых жертв (переосмыслив которые в будущем, Вы можете даже принять их за благо), и важным шагом на таком пути будет отказ от MS Windows. Однако, это не означает что переход произойдёт моментально и без трудностей, но нужно понемногу работать в этом направлении и изучать Unix-системы, Linux или *BSD.

Зацикленность людей на Windows и поиск решений для укрепления этой ОС напоминает визит к врачу тяжело больного пациента, который просит быстродействующего и быстропомогающего лекарства. Врач понимает, что такое лекарство может ему временно помочь, однако оно не вылечит болезнь, не увеличит срок жизни, и больной всё равно продолжит идти к могиле. Вместо этого ему предлагаются кардинальные методы (*nix-система), которые могут помочь, но они не действуют моментально, требуют терпения и выдержки, требуют сменить привычный образ жизни, требуют регулировать своё питание, и ещё много в чём себя контролировать/исправлять. Но, как показывает практика, больные в большинстве тем не менее выбирают морфий Windows. [источник]

Основные понятия

Стандарной программой шифрования в линуксе является PGP - Pretty Good Privacy (бесплатна для некоммерческого использования) или открытый аналог этой GnuPG - Gnu Privacy Guard.

Симметричные шифры

- Единый ключ для зашифровывания и расшифровывания данных.
- Передача ключа по защищённому каналу.
- Алгоритмы: 3DES, Blowfish, IDEA, ГОСТ.
- Характерный размер ключа ~128 бит.

Шифры с открытым ключом

- Открытый ключ (public key) - передаётся по открытому каналу, хранится, как правило, в системе хранения ключей. Служит для шифрования сообщения к владельцу парного секретного ключа (private key).
- Секретный ключ (private key) - хранится локально и приватно (для сохранности в нескольких местах). Служит для расшифровки сообщений, зашифрованных парным открытым ключом.
- Алгоритмы: RSA, DSA.
- Характерный размер ключа ~2048 бит.

Смешанные шифры

Рас/шифровывание с использованием пары ключей проходит на два-три порядка медленнее, в сравнении с симметричным алгоритмом, требует бóльших вычислительных ресурсов, поэтому на практике используются смешанные шифры:

- Сообщение для получателя кодируется симметричным шифром, с использованием сеансового ключа, который в свою очередь кодируется открытым ключом получателя.
- Получатель расшифровывает свом секретным ключом сеансовый ключ, который используется для расшифровки основного сообщения.

Цифровая подпись

Для реализации ЭЦП необходимо, чтобы шифр обладал тем свойством, что и открытый ключ, и секретный ключ может быть использован алгоритмом шифрования в качестве открытого ключа т.е., могут быть использованы для шифрования данных. RSA - является таким алгоритмом.

- Вычисляется хеш-функция сообщения.
- Полученное значение шифруется с помощью секретного ключа. Документ отправляется получателю.
- Получатель, используя открытый ключ отправителя (соответствующий секретному ключу, которым было зашифровано сообщение), расшифровывает хеш-функцию и сравнивает полученное значение с хешем полученного сообщения.

Источник

Работа с GnuPG

Управление ключами

- **Создание ключа**

```
gpg --gen-key
```

- **Генерация сертификата отзыва**

```
gpg --list-keys
/home/user/.gnupg/pubring.gpg
---------------------------------
pub  1024D/95447C14 2014-02-08 Name Family <user@nowhere.someplace.com>
sub  2048g/96D6CDAD 2014-02-08
```

```
gpg --output revoke.asc --gen-revoke 95447C14
```

После создания ключа и сертификата отзыва необходимо сделать резервную копию полученных файлов (по-умолчанию располагаются в ~/.gpg/) в защищённом хранилище.

- **Редактирование ключа** происходит в полуинтерактивном режиме:

```
gpg --edit-key 4600813F
```

или по данным пользователя

```
gpg --edit-key user@nowhere.someplace.com
```

- **Просмотр списка известных ключей**

```
# публичные ключи
gpg --list-keys
# приватные ключи
gpg --list-secret-keys
```

- **Распространение публичного ключа**

```
# запрос публичного ключа
gpg --keyserver subkeys.pgp.net --recv-keys 5B20C7D2
# отправка своего публичного ключа на сервер
gpg --keyserver subkeys.pgp.net --send-keys 5B20C7D2
# обновление ключей в локальном хранилище ключей
gpg --refresh-keys --keyserver subkeys.pgp.net
# экспорт публичного ключа в файл
gpg --armor -output 4600813F.pub --export 4600813F
# импортировать открытый ключ из файла
gpg --fetch-keys http://example.com/4600813F.pub
```

(Рас)шифрование файлов

- **Шифрование** file.txt открытым ключом, принадлежащим name@dei.uc.pt

```
gpg --output file.txt.gpg --encrypt --recipient name@dei.uc.pt file.txt
```

- **Создать подпись документа в отдельном файле *file.txt.sig***

```
gpg --recipient name@dei.uc.pt --detach-sign file.txt
```

- **Добавить подпись в конец документа**

```
gpg --sign --recipient name@dei.uc.pt file.txt
```

- **Зашифровать и подписать документ**

```
gpg --sign --encrypt --recipient name@dei.uc.pt file.txt
```

- **Расшифровать и проверить подпись документа, созданного личным открытым ключом**

```
gpg --output file.txt --decrypt file.txt.gpg
```

- **Проверка подписи**

```
gpg --verify file.txt.sig file.txt
```

Симметричное шифрование

- **Зашифровать файл**

```
gpg --output file.txt.gpg --symmetric file.txt
```

- **Расшифровать файл**

```
gpg --output file.txt --decrypt file.txt.gpg
```

Источники: раз, два

Хранение паролей: Vim + GnuPG

Общие положения

Дана машина, доступная из сети по ssh, на машине установлен криптографический пакет gpg, установлен vim с плагином vim-gnupg, а также дополнительно доступен git.
Задача обеспечить приватный доступ и хранение паролей на данной машине. **Источник**: тыц.

1. Генерируем пару ключей на целевой машине, тыц
   Не забываем сделать копии секретного ключа и создать сертификат отзыва.
2. Тестируем работоспособность шифрования на файле test.txt:
   Если всё сделано правильно, то будет создан шифрованный файл test.txt.gpg

   ```
   gpg --encrypt --recipient master@jurik-phys.net test.txt
   ```

3. Устанавливаем Vim-GnuPG в систему:

   ```
   sudo apt-get install vim-scripts
   ```

4. Подключаем плагин к vim'у:

   ```
   # создаём локальный каталог плагина
   mkdir ~/.vim/plugin
   # создаём символическую ссылку на плагин
   ln -s /usr/share/vim-scripts/plugin/gnupg.vim ~/.vim/plugin/gnupg.vim
   ```

5. Открываем vim'ом ранее зашифрованный файл

   ```
   vim test.txt.gpg
   ```

   После ввода парольной фразы к закрытому ключу, открывается файл в расшифрованном виде, который может быть прозрачно отредактирован и сохранён.

6. При создании новго зашифрованного файла, vim попросит указать получателя

   ```
   vim secret.gpg
   ```

Настройки в vimrc

Для большего удобства предлагается установить следующие настройки в .vimrc. Подробности используемых опций см. в первоисточнике, тыц.

```
" Tell the GnuPG plugin to armor new files.
let g:GPGPreferArmor=1

" Tell the GnuPG plugin to sign new files.
let g:GPGPreferSign=1

augroup GnuPGExtra
    " Set extra file options.
    autocmd BufReadCmd,FileReadCmd *.\(gpg\|asc\|pgp\) call SetGPGOptions()
    " Automatically close unmodified files after inactivity.
    autocmd CursorHold *.\(gpg\|asc\|pgp\) quit
augroup END

function SetGPGOptions()
    " Set updatetime to 1 minute.
    set updatetime=60000
    " Fold at markers.
    set foldmethod=marker
    " Remove comment symbols.
    set commentstring=%s
    " Automatically close all folds.
    set foldclose=all
    " Only open folds with insert commands.
    set foldopen=insert
endfunction
```

Генерация паролей

Для генерации можно использовать утилиту pwgen прямо из vim'a

```
:r!pwgen -sy 12 1
```

Сертификат для сайта.

Получение сертификата

По материалам habrahabr.ru, дополнительная литература.

1. Регистрация на сайте https://login.wosign.com/reg.html и подтверждение электронной почты.

2. Заказ бесплатного сертификата по адресу https://buy.wosign.com/free/. Рядом с ценой (обозначено красным цветом) нажать на синий квадрат (получение бесплатного сертификата), ниже зелёная кнопка - продолжить оформление сертификата.

3. Ввод нужных доменных имен, подтверждение владения доменом. При вводе домена в виде www.example.com, example.com выписывается сервисом автоматически.

4. Генерация на сервере приватного ключа и CSR (Certificate Signing Request) - запроса на получение сертификата, который представляет собой текстовый файл, содержащий в закодированном виде информацию об администраторе домена и открытый ключ:

```
openssl req -out mydomain.com.csr -new -sha256 -newkey rsa:2048 -nodes -keyout
mydomain.com.key
```

При генерации CSR необходимо ввести, как минимум:

- имя сервера (Common Name) — полностью определенное доменное имя;
- название страны (Country Name) — двухбуквенный код страны, для РФ — «RU».

Указанные в CSR данные, должны соответствовать данным, указанным в Whois-сервисе по домену.

5. Вставка содержимого файла mydomain.com.csr в форму сайта. Завершение формальностей с вводом пароля, принятием лицензионного соглашения и т.д.

6. Скачивание zip-архива с сертификатом сайта с сайта или по ссылке в письме. При подписывании сертификата возможна задержка на несколько часов.

7. После установки сертификата и сгенерированного приватного ключа на сайт, желательно проверить уровень безопасности сайта через SSL-тестере.

8. Настройка OCSP stapling на сервере для полученного сертификата.

Настройка OCSP stapling

По материалам 8host.com и , habrahabr.ru

Получение корневого и промежуточного сертификата CA.

```
wget -O - https://www.startssl.com/certs/ca.pem | tee -a ca-certs.pem > /dev/null
wget -O - https://www.startssl.com/certs/sub.class1.server.ca.pem | tee -a ca-
certs.pem > /dev/null
```

Сертификаты в формате DER необходимо сконвертировать в PEM.

```
wget -O - http://aia.startssl.com/certs/ca.crt | openssl x509 -inform DER -outform
PEM | tee -a ca-certs.pem > /dev/null
wget -O - http://aia1.wosign.com/ca1g2-server1-free.cer | openssl x509 -inform DER
-outform PEM | tee -a ca-certs.pem > /dev/null
wget -O - http://aia6.wosign.com/ca6.server1.free.cer | openssl x509 -inform DER -
outform PEM | tee -a ca-certs.pem > /dev/null
```

Редактирование файла виртуальных хостов SSL. В директиву <VirtualHost></VirtualHost> добавить:

```
SSLCACertificateFile /etc/ssl/ca-certs.pem
SSLUseStapling on
```

Вне директивы <VirtualHost></VirtualHost> указать кэш

```
SSLStaplingCache shmcb:/tmp/stapling_cache(128000)
```

Протестировать изменённый конфиг и перезагрузить настройки:

```
apachectl -t
```

```
service apache2 reload
```

Проверка результата на SSL-тестере.

```
Protocol Details
----------------
OCSP stapling   Yes
```

OCSP stapling итоги

На следующий день после удачной настройки сайт перестал открываться в ошибкой

```
sec_error_ocsp_try_server_later
```

В логах apach'a ошибка Bad gateway 502. Вероятно, что проявилась обозначенная на хабре проблема.

Forward Secrecy по-умолчанию

Способ 1. Рекомендованный: (digicert.com).
Использование Forward Secrecy определяется набором используемых шифров.

```
SSLProtocol all -SSLv2 -SSLv3
SSLHonorCipherOrder On
SSLCipherSuite "EECDH+ECDSA+AESGCM EECDH+aRSA+AESGCM EECDH+ECDSA+SHA384
EECDH+ECDSA+SHA256 EECDH+aRSA+SHA384 EECDH+aRSA+SHA256 EECDH+aRSA+RC4 EECDH
EDH+aRSA RC4 !aNULL !eNULL !LOW !3DES !MD5 !EXP !PSK !SRP !DSS !RC4"
```

Способ 2. Простой, но увеличивающий нагрузку на сервер: (raymii.org).

Увеличение стойкости DH-шифров.

```
#Generate the parameters
cd /etc/ssl/certs
openssl dhparam -out dhparam.pem 4096

# Add the following to your Apache config.
SSLOpenSSLConfCmd DHParameters "/etc/ssl/certs/dhparam.pem"
```

Turn On HSTS

HSTS - HTTP Strict Transport Security. Включение, согласно, itigloo.com.

1. Enable the Apache Headers Module.

   ```
   a2enmod headers
   ```

2. Add the additional header to the HTTPS VirtualHost directive. Max-age is measured in seconds.

   ```
   <VirtualHost *:443>
   ```

```
    # Guarantee HTTPS for 1 Year including Sub Domains
    Header always set Strict-Transport-Security "max-age=31536000;
includeSubDomains"
```