Документ содержит описание внутреннего устройства сервиса под ОС Windows

Мастер-пароль

Мастер пароль не хранится на жёстком диске, это сделано в целях безопасности. На диске хранится только лишь SHA-256 хеш от пароля. Хеш используется для валидации введёного пользователем пароля. Пароль хранится в памяти в зашифрованном виде, в открытом виде предоставляется по требованию. Находясь в памяти, пароль шифруется симметричным алгоритмом, расшифровывается по требованию, затем после использования открытая версия пароля безопасно затирается.

Шифрование для thrift-сервисов

Обмен данными между сервисом и клиентом должен происходить через HttpTransport, потому что он позволяет добавлять в заголовок запроса дополнительную информацию.

Запросы могут быть шифрованными и открытыми. Если запрос зашифрован, значит в запросе должен присутствовать заголовок Dgi\_transport\_encrypted, который включает в себя следующие заголовки:

* Dgi\_content\_hash – контрольная сумма оригинального сообщения (sha-256)
* Dgi\_dsign – цифровая подпись оригинального сообщения
* Dgi\_valid\_key – публичный ключ для проверки цифровой подписи (rsa-pssr-sha256)
* Dgi\_

-

-

File encryption

Шифрование содержимого файла осуществляется с использованием Grader, Aes, Aes256 алгоритмов. Содержимое файла подвергается обработке, после чего в конец файла записывается мета-информация, которая содержит следующие пооля:

struct EncodingMetaInfo

{

unsigned char headerSignature[16]; // FE\_Signature container for

::logic::common::Sfci sfc;

char dateTime[50 /\* including last zero symbol \*/];

bool usedMasterPassword;

common::CryptAlgorithm encodingAlgorithm;

*std*::*size\_t* originalFileSize;

char originalChecksum[64 + 1];

char keyChecksum[64 + 1];

unsigned char endSignature[16]; // FE\_SignatureEnd container for

};

Где headerSignature и endSignature идентификационная сигнатура для определения наличия мета информации в теле файла. Данную структуру можно поместить в любое место файла, главное знать, что первоначальные данные, на место которых записывается информация, записываются в конец файла. Что требует последующего перемещения данных на их первоначальное место. Второй вариант, более простой – структуру можно просто дописывать в overlay файла, в самый конец, что избавляет от необходимости восстановления первоначальных данных и поиск самой мета информации в теле файла. Ава

dateTime – время и адата когда содержимое файла было закодировано. Пример - "11:11:11 12.09.2019"

usedMasterPassword – true если в качестве ключа использовался мастер-пароль.

encodingAlgorithm - информация по используемому для шифрования алгоритму.

originalFileSize – размер файла до шифрования в байтах.

originalChecksum – sha256 хеш в строковом представлении для незашифрованного файла.

keyChecksum - sha256 хеш в строковом представлении для используемого ключа шифрования.

Данная структура позволяет проверить наличие валидного ключа, алгоритма шифрования ещё до процесса расшифровки, чтобы убедиться в верности входных параметров (алгоритм, ключ). В случае нарушения содержимого файла можно будет выявить факт изменения уже зашифрованных данных используя ранее полученный хеш для исходных данных.

(\*) Методы сервиса по дешифрации содержимого файла, могут возвращать следующая информацию:

1. Файл декодирован успешно;

2. Файл декодирован, но его содержимое не соответствует исходному – т.е. будучи зашифрованным он подвергся изменению.

3. Файл не может быть декодирован, потому что предлагаемый ключ шифрования не является исходным.

-

-

DgiEngine

Тут описание

SyncEventsHolder – хранитель событий синхронизации.

-

-

-