2.4 Шифрование конфиденциальных данных

Шифрование конфиденциальных данных

Конфиденциальные данные, такие как токены, ключи, номера кредитных карт и другие "секреты", должны храниться в зашифрованном виде для предотвращения утечки данных.

Flet включает в себя вспомогательные методы для шифрования и дешифрования чувствительных текстовых данных с помощью симметричного алгоритма (где один и тот же ключ используется для шифрования и дешифрования). Они используют реализацию Fernet из пакета cryptography, которая является AES 128 с некоторым дополнительным усилием, плюс PBKDF2 для получения ключа шифрования из пароля пользователя.

# Секретный ключ

Секретный ключ шифрования (также известный как пароль или ключевая фраза) - это произвольная строка вроде пароля, настроенная пользователем и используемая для шифрования и дешифрования данных. Криптоалгоритм использует секретный ключ для получения ключа шифрования (32 байта).

*ВНИМАНИЕ!*

*Не встраивайте никакие секреты в исходный код, чтобы избежать случайного раскрытия информации!*

Вы можете предоставить секрет вашему приложению через переменную окружения:

import os

secret\_key = os.getenv("MY\_APP\_SECRET\_KEY")

Перед запуском приложения установите секрет в командной строке:

$ export MY\_APP\_SECRET\_KEY="<секрет>"

*ПРИМЕЧАНИЕ*

*Передача секретов через переменные окружения - распространенная практика среди разработчиков и провайдеров услуг, но она не полностью предотвращает утечку секретов в некоторых средах. Для внедрения секретов в ваше приложение могут быть использованы другие механизмы, такие как монтирование секретных файлов или использование сервисов хранилищ секретов.*

# Шифрование данных

Используйте метод для шифрования строки:

encrypt()

import os

from flet.security import encrypt, decrypt

secret\_key = os.getenv("MY\_APP\_SECRET\_KEY")

plain\_text = "Это секретное сообщение!"

encrypted\_data = encrypt(plain\_text, secret\_key)

- это строка, закодированная в URL-безопасном формате base64.

Метод принимает только строки, поэтому все объекты должны быть

encrypted\_data

encrypt

сериализованы в JSON, XML или другой текстовый формат перед шифрованием.

# Дешифрование данных

Используйте метод для дешифрования данных:

decrypt()

import os

from flet.security import encrypt, decrypt

secret\_key = os.getenv("MY\_APP\_SECRET\_KEY")

encrypted\_data = "601llp2zpPp4QjBWe2cOwGdBQUFBQUJqTTFJbmgyWU5jblVp..."

plain\_text = decrypt(encrypted\_data, secret\_key)

print(plain\_text)

В этом фрагменте кода мы используем тот же секретный ключ для дешифрования данных, которые были ранее зашифрованы. Результатом является оригинальное сообщение в текстовом формате. Если ключ или зашифрованные данные

неправильные, обернут в блок

try/except

decrypt()

выбросит исключение, поэтому его вызов должен быть для обработки исключений.

Важно помнить, что секретный ключ, используемый для шифрования, должен быть абсолютно идентичным секретному ключу, используемому для дешифрования. Если вы потеряете секретный ключ, вы не сможете восстановить зашифрованные данные