Pour superviser les certificats SSL avec Python, vous pouvez utiliser la bibliothèque ssl pour établir une connexion sécurisée et vérifier les détails du certificat. Vous pouvez également utiliser des bibliothèques tierces telles que cryptography pour des fonctionnalités plus avancées. Voici un exemple simple en utilisant la bibliothèque standard ssl :

import ssl

import socket

from datetime import datetime

def check\_ssl\_certificate(hostname, port=443):

context = ssl.create\_default\_context()

with context.wrap\_socket(socket.socket(socket.AF\_INET), server\_hostname=hostname) as s:

s.connect((hostname, port))

cert = s.getpeercert()

# Date d'expiration du certificat

expiration\_date = datetime.strptime(cert['notAfter'], "%b %d %H:%M:%S %Y %Z")

print(f"Date d'expiration du certificat : {expiration\_date}")

# Sujet du certificat (nom du domaine)

subject = dict(x[0] for x in cert['subject'])

print(f"Sujet du certificat (nom du domaine) : {subject['commonName']}")

# Émetteur du certificat

issuer = dict(x[0] for x in cert['issuer'])

print(f"Émetteur du certificat : {issuer['commonName']}")

if \_\_name\_\_ == "\_\_main\_\_":

hostname\_to\_check = "example.com"

check\_ssl\_certificate(hostname\_to\_check)

Assurez-vous que votre environnement Python dispose de la bibliothèque ssl. Vous pouvez installer cette bibliothèque en exécutant pip install cryptography.

Ce script établit une connexion SSL avec le serveur spécifié, récupère les détails du certificat et les affiche. Vous pouvez personnaliser ce script en fonction de vos besoins, par exemple en ajoutant des vérifications supplémentaires, en automatisant le processus ou en l'intégrant dans un système de surveillance plus large.

Haut du formulaire