ajitdash Microsoft

May 14 2023 0

...

How to Secure Azure OpenAI Keys Using Environment Variables, Azure Vault, and Streamlit Secrets

How to Secure Azure OpenAI Keys Using Environment Variables, Azure Vault, and Streamlit Secrets

Abstract :

In today's fast-paced world, securing sensitive data and credentials is of utmost importance. When working with Azure and OpenAI, it's crucial to ensure that your API keys and other credentials are stored securely. One common way to do this is by using environmental variables. However, environmental variables are not always the safest option, and there are better alternatives available.

In this blog post, we'll explore how to secure Azure OpenAI keys using environment variables, Azure Key Vault, and Streamlit Secrets. We'll also cover best practices for securing your API keys and provide examples to help you implement these techniques in your own projects.

Introduction :

Azure OpenAI is a powerful platform that allows developers to build and deploy AI models at scale. However, as with any platform, it's important to ensure that your data and credentials are kept secure. One way to do this is by using environment variables to store your API keys. However, environment variables can be accessed by anyone with access to the server, making them less secure.

To address this issue, Azure Key Vault can be used to store and manage your API keys securely. In addition, Environment variables allow you to store sensitive data outside of your code, Azure Vault provides secure key storage, and Streamlit Secrets allows you to securely access your keys in a Streamlit app.

Installation :

Before we get started, there are a few things you need to set up:

Create an Azure account and subscribe to Azure OpenAI.
Install the Azure CLI on your local machine.
Install the Streamlit library. **(pip install streamlit)**

After Setup :

Challenges:

One of the biggest challenges in securing keys is ensuring that they are not accidentally exposed. This can happen if they are hard-coded into your code or if they are stored in an unsecured location. By using environment variables, Azure Vault, and Streamlit Secrets, we can ensure that our keys are stored securely and only accessed by authorized parties.

Here are the steps to secure your Azure OpenAI keys using environment variables, Azure Vault, and Streamlit Secrets:

1. **Store your Azure OpenAI key in an environment variable:**

Open your command prompt or terminal and enter the following command:
export OPENAI_KEY=<your_key_here>
This will store your key in an environment variable called OPENAI_KEY.
Best practice: Use a unique and complex name for your environment variable to prevent accidental exposure.

2. **Create an Azure Key Vault:**

Log in to the Azure portal and create a new Key Vault.
Generate a new access key for the Key Vault.
2a> # Create a new Key Vault instance
az keyvault create --name mykeyvault --resource-group myresourcegroup --location westus2

Best practice: Use strong authentication methods such as multi-factor authentication to protect your Azure account.

2b> # Add a secret to the Key Vault
az keyvault secret set --name myapikey --vault-name mykeyvault --value <API_KEY>
Store your Azure OpenAI key in the Azure Key Vault:

Using the Azure CLI, enter the following command:

az keyvault secret set --vault-name <your_vault_name> --name openai-key --value <API_KEY>

This will store your Azure OpenAI key in the Azure Key Vault.

Best practice: Use role-based access control to ensure that only authorized users have access to the Key Vault.

3 .Access your Azure OpenAI key in a Streamlit app using Streamlit Secrets:

In your Streamlit app, import the Streamlit Secrets library.
Retrieve your Azure OpenAI key from the Key Vault using the Azure CLI:
az keyvault secret show --vault-name <your_vault_name> --name openai-key --query value -o tsv
Use Streamlit Secrets to securely access your key:
st.secrets['openai-key']

Example 1 :
Next, let's create a Streamlit application that retrieves our API key from the Key Vault using the Azure SDK:

```
import os
from azure.identity import DefaultAzureCredential
from azure.keyvault.secrets import SecretClient

# Set up the Key Vault client
credential = DefaultAzureCredential()
client = SecretClient(vault_url=os.environ["AZURE_KEYVAULT_URL"], credential=credential)

# Retrieve the API key from the Key Vault
In this example, the Azure OpenAI key is loaded from a Streamlit secret and used to make requests to the Azure OpenAI API. The key is not exposed in the code, making it more secure.
api_key = client.get_secret("myapikey").value
```

Example 2: Streamlit Secrets to secure your Azure OpenAI key:

```
import streamlit as st
import os

# Load the Azure OpenAI key from a Streamlit secret
```

key = st.secrets["azure_openai_key"]

 # Use the key to make requests to the Azure OpenAI API
response = requests.get("https://api.openai.com/v1/engines/davinci-codex/completions", headers={"Authorization": f"Bearer {key}"})
 # Use the API key in your application

Best practice: Limit the scope of your Streamlit app to only those users who need access to the key.

Example 3 : Using - Using Streamlit "secrets"
This is the easiest way using Streamlit secrets:

1.Create a folder within your director where you have the code name as *".**streamlit** "*

2.Create a file name as *" **secrets.toml" under the folder** ".**streamlit** "*

3.Assign the key in the *" **secrets.toml"***
Path = '363e5eaaaaaabbbbbccccc'

Flow will look like this *: **projectfolder\.streamlit***

When you call the key within your code use this : ***openai.api_key = st.secrets['path']***

Make sure to add the other credentials. Will look like this
 # Set up OpenAI API credentials
openai.api_type = "azure"
openai.api_base = 'https://xxxxxx.openai.azure.com/'
openai.api_version = "2023-03-15-preview"
openai.api_key = st.secrets['path']

Example 4: Using environment variables

To illustrate how to securely store your Azure OpenAI key, let's consider an example application that uses the GPT-3 language model to generate text. To start, we'll store the API key as an environment variable, which can be done by adding the following line to your code:
import os
os.environ["OPENAI_API_KEY"] = "your_api_key_here"

While this is a simple solution, it's not the most secure, as anyone with access to your code can easily view your API key. Instead, we can use Azure Vault to store our key securely and access it programmatically. This can be done by creating a secret in Azure Vault and using the Azure Key Vault library to retrieve it in your code:

```
from azure.keyvault.secrets import SecretClient
from azure.identity import DefaultAzureCredential

credential = DefaultAzureCredential()
client = SecretClient(vault_url="https://yourvaultname.vault.azure.net/", credential=credential)

key = client.get_secret("openai-api-key")
os.environ["OPENAI_API_KEY"] = key.value
```

Overall Best practices:

To ensure the security and confidentiality of your Azure OpenAI keys, here are some best practices to follow:

- Never hardcode keys in your code or store them in plain text files.
- Use a secure key management solution like Azure Key Vault or Streamlit secrets to protect your keys.
- Rotate your keys regularly and revoke any unused keys.
- Use Azure AD authentication to control access to your keys.
- Monitor and audit key usage to detect any unauthorized access.

Conclusion:

Securing Azure OpenAI keys is essential to prevent unauthorized access and abuse. By using Azure Key Vault and Streamlit secrets, you can securely store and use the keys in your applications. Remember to follow the best practices for key protection to ensure the security of your applications.

References:

Azure Key Vault documentation: https://docs.microsoft.com/en-us/azure/key-vault/

Share