Note

This documentation applies to Legacy BloodHound and is no longer maintained.

See up-to-date documentation for BloodHound CE here: AzureHound Community Edition

AzureHound

AzureHound is a Go binary that collects data from AzureAD and AzureRM via the MS Graph and Azure REST APIs. It does not use any external dependencies and will run on any operating system.

Building AzureHound From Source

You can build AzureHound from source by cloning this repository:

Then, cd into the directory you just cloned and type:

```
go build .
```

This will build AzureHound and you will have a new binary called azurehound in this directory.

Collecting Data with AzureHound

AzureHound supports several authentication flows for collecting information from Azure. You can supply a username/password combo, a JWT, a refresh token, a service principal secret, or service principal certificate. You can combine these various authentication methods with several collection scoping options.

For example, to authenticate with a username/password and list all groups in a tenant:

```
./azurehound -u "MattNelson@contoso.onmicrosoft.com" -p "MyVeryStrongPassword" list groups
```

AzureHound will authenticate as that user and print all groups in the “Contoso” tenant.

Or, you may want to supply a JWT and collect all users from the tenant instead. You do not need to supply a username or password when supplying a JWT:

```
./azurehound -j "ey..." list users --tenant "contoso.onmicrosoft.com"
```

When collecting data for import into BloodHound, you must use the -o switch to instruct AzureHound to output to a file. For example, to list all available data in both AzureAD and AzureRM, you can do this:

```
./azurehound -u "MattNelson@contoso.onmicrosoft.com" -p "MyVeryStrongPassword" list groups
```

Dealing with Multi-Factor Auth and Conditional Access Policies

latest

If a user has MFA or CAP restrictions applied to them, you will not be able to authenticate with just a username and password with AzureHound. In this situation, you can acquire a refresh token for the user and supply the refresh token to AzureHound.

The most straight-forward way to accomplish this is to use the device code flow. In this example I will show you how to perform this flow using PowerShell, but this example can be very easiliy ported to any language, as we are simply making calls to Azure APIs.

Open a PowerShell window on any system and paste the following:

```
$body = @{
    "client_id" =     "1950a258-227b-4e31-a9cf-717495945fc2"
    "resource" =      "https://graph.microsoft.com"
}
$UserAgent = "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36 (KHTML, li
$Headers=@{}
$Headers["User-Agent"] = $UserAgent
$authResponse = Invoke-RestMethod `
    -UseBasicParsing `
    -Method Post `
    -Uri "https://login.microsoftonline.com/common/oauth2/devicecode?api-version=1.0" `
    -Headers $Headers `
    -Body $body
$authResponse
```

The output will contain a user_code and device_code. Now, open a browser where your AzureAD user either already logged on or can log on to Azure. In this browser, navigate to https://microsoft.com/devicelogin

Enter the code you generated from the above PowerShell script. Follow the steps in the browser to authenticate as the AzureAD user and approve the device code flow request. When done, the browser page should display a message similar to "You have signed in to the Microsoft Azure PowerShell application on your device. You may now close this window."

Now go back to your original PowerShell window and paste this:

```
$body=@{
    "client_id" =  "1950a258-227b-4e31-a9cf-717495945fc2"
    "grant_type" = "urn:ietf:params:oauth:grant-type:device_code"
    "code" =       $authResponse.device_code
}
$Tokens = Invoke-RestMethod `
    -UseBasicParsing `
    -Method Post `
    -Uri "https://login.microsoftonline.com/Common/oauth2/token?api-version=1.0" `
    -Headers $Headers `
    -Body $body
$Tokens
```

The output will include several tokens including a refresh_token. It will start with characters similar to "0.ARwA6Wg...". Now you are ready to run AzureHound! Take the refresh token and supply it to AzureHound using the -r switch:

```
./azurehound -r "0.ARwA6Wg..." list --tenant "contoso.onmicrosoft.com" -o output.json
```

This will attempt to list all possible data from that particular tenant, but you can ALSO use that same refresh token to target any other tenant your user has access to!

Previous   Next

latest