Red Teaming Microsoft: Part 1 - Active Directory Leaks via Azure - Black Hills Information Security, Inc.

By BHIS

Published: 2018-08-31 · Archived: 2026-04-02 10:40:57 UTC

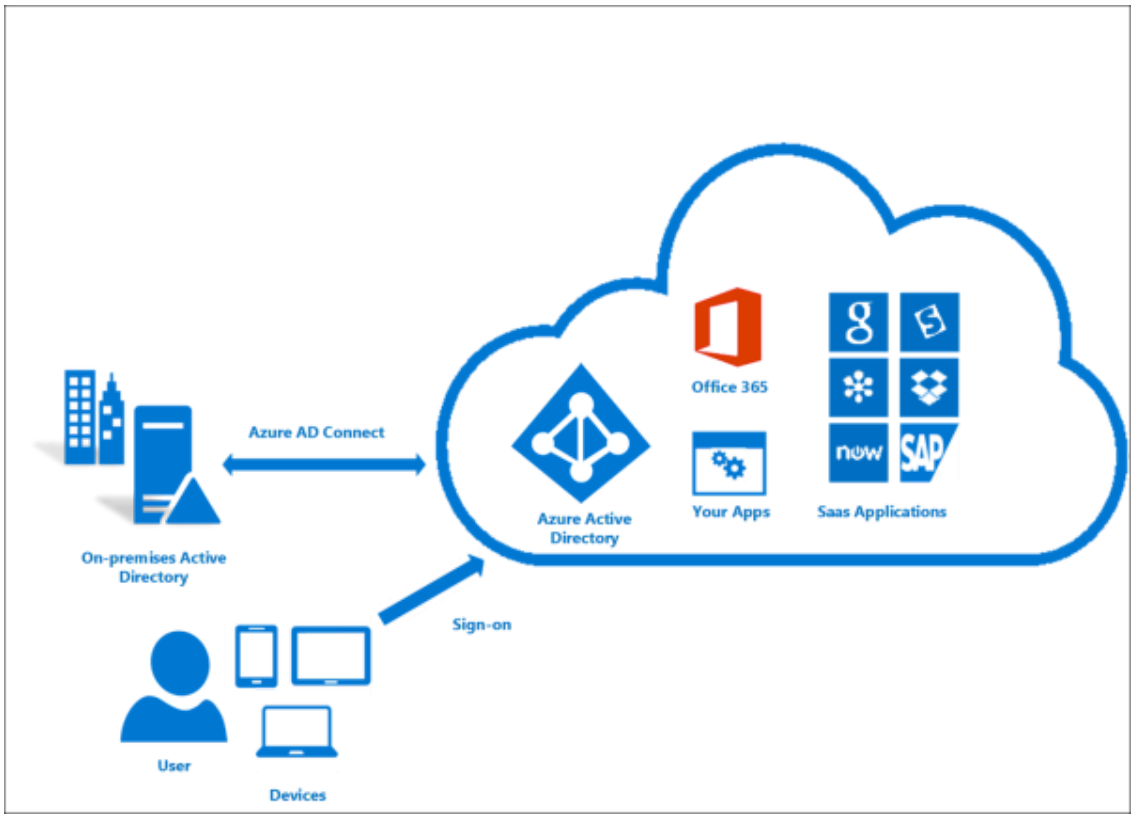
Mike Felch //

With so many Microsoft technologies, services, integrations, applications, and configurations it can create a great deal of difficulty just to manage everything. Now imagine trying to secure an environment that goes well beyond the perimeter. While moving everything to a cloud provider can provide amazing returns in scalability, functionality, and even savings, it can also create major blind-spots. Over the past year, I have been looking into ways to target organizations that utilize Microsoft as their cloud provider. I hope to release a number of different techniques that have been extremely beneficial in uncovering these blind-spots, much like the research Beau Bullock (@dafthack) and I did when we focused our scope on Google.

I won't begin to mislead you, I am no Microsoft expert. In fact, the more I read about the products and services the more I felt lost. While over the past year I've been able to maneuver through and bend these technologies in order to target the organizations better from a red team perspective, I struggled to try to understand many different concepts. What is the default configuration for this? Is this provided by default? Is this syncing with everything? If I make changes here, do they propagate back? Why not? The list goes on and on. When I've shared some of these techniques privately it was inevitable that a question would immediately follow. While I feel bringing a problem without a solution is irresponsible, there may be times like now where solutions aren't black and white. My advice is to know your environment, know your technologies, and if you aren't sure then reach out to your service provider so you can be sure.

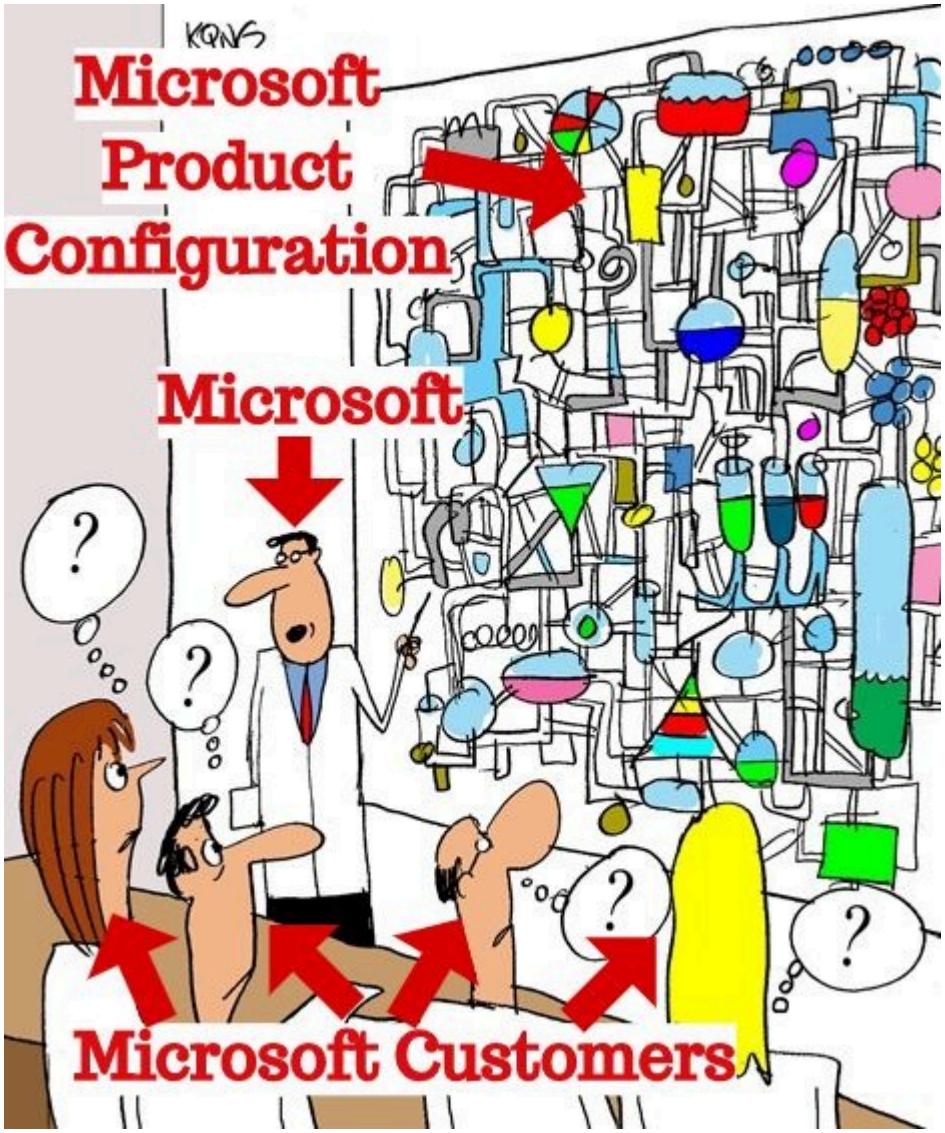
The Microsoft Landscape

So you've been running Microsoft Active Directory and Exchange on-prem for years but want to quickly deploy Microsoft Office to your employees while also providing them with access to a webmail portal, Sharepoint, and SSO for some internal applications. Somewhere along the way, you decided to migrate to Office 365 and everything works well! All your users can authenticate with their network credentials and their email works great! Would you consider yourself an on-prem organization still or are you in the infamous cloud now? Maybe you took a hybrid approach and did both. Microsoft provides an amazing amount of integrations that they support but how do you know if you are managing everything correctly?



A Hypothetical Complex Situation

For managing users on-prem there's the traditional Microsoft AD. For managing users in cloud services, you could leverage Azure AD. For mail, there's Exchange on-prem but you could always move email to Exchange online. If you want the full suite of Microsoft Office there's Office 365 but I think that routes through Exchange online in a Microsoft multi-tenant environment anyhow, so you could technically be using both but paying for one. Since you paid for Office 365 Business, you were also provided a number of services like Skype and OneDrive despite using GDrive or Box for corporate file sharing. You enroll in a multi-factor solution with SMS tokens being the default delivery mechanism but for some reason, your users can still authenticate with Outlook without needing MFA… weird.. (Major thanks to Microsoft EWS) Overall, everything just works and for that we have to thank Azure AD Connect.. or is it Azure AD Synchronization Services.. or are we still running old school DirSync with Forefront Identity Manager? Whatever it is, it's working and that's all that matters!



So.. What's the Big Deal?

A number of problems are created in the situation just illustrated and there is very little a blue team can do to defend or respond to a number of different attacks ranging from dumping active directory remotely to bypassing and even hijacking multi-factor authentication for users.

Understanding who is who within an organizational department is typically done in the reconnaissance phase of an engagement through third-party services like LinkedIn or other OSINT techniques. If you are on the internal network then revisiting this step is crucial because you need to understand deeper details of the organization like what groups are configured and who are the members of those groups. This is vital in being able to successfully pivot to relevant machines and targeting users based on their access so that escalation can be accomplished. But what if you aren't on the internal network but still need to determine who to target? Even better, what if the target gems of the organization are hosted in the cloud and you never actually have to hit the internal network?

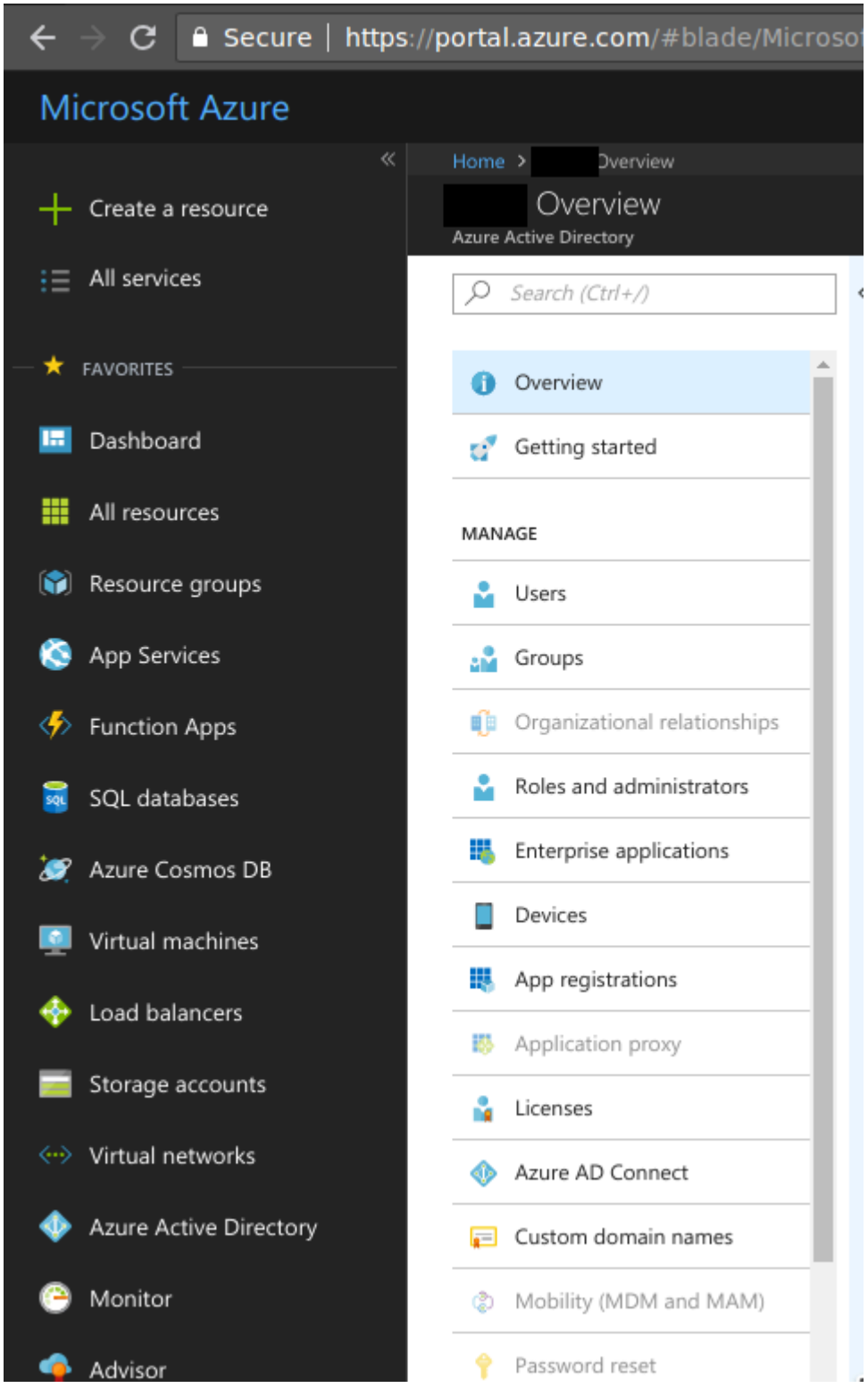
With Microsoft, if you are using any cloud services (Office 365, Exchange Online, etc) with Active Directory (on-prem or in Azure) then an attacker is one credential away from being able to leak your entire Active Directory structure thanks to Azure AD.

Step 1) Authenticate to your webmail portal (i.e. https://webmail.domain.com/)

Step 2) Change your browser URL to: https://azure.microsoft.com/

Step 3) Pick the account from the active sessions

Step 4) Select Azure Active Directory and enjoy!



This creates a number of bad situations. For instance, if we were able to export all the users and groups we would have a very nice list of employees and the groups they are a part of. We can also learn what group we need to land in for VPN, domain administration, database access, cloud servers, or financial data. What's also nice about Azure AD is that it holds the device information for each user so we can see if they are using a Mac, Windows machine, or iPhone along with the version information (i.e. Windows 10.0.16299.0). As if all this wasn't great already, we can also learn about all the business applications with their endpoints, service principal names, other domain names, and even the virtual resources (i.e. virtual machines, networks, databases) that a user might have access to.

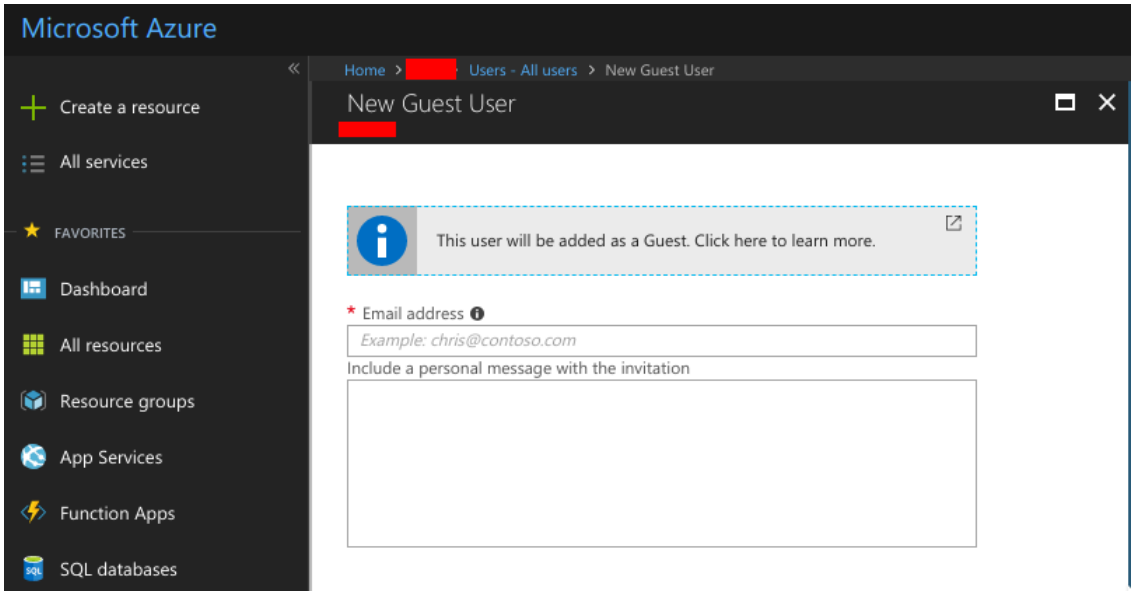
But Wait, There's More!

An added benefit to authenticating to the Azure portal as a regular user is that you can create a backdoor… err… I mean a "Guest" account. How super convenient!

Step 1) Click "Azure Active Directory"

Step 2) Click "Users" under the Manage section

Step 3) Click "New Guest User" and invite yourself



Depending on their configuration, it may or may not sync back to the internal network. In fact, while creating guest accounts is on by default — I've only verified one customer where Azure AD Connect was a bi-directional sync allowing guest accounts to authenticate, enroll a multi-factor device and VPN internally. This is an important configuration component for you to understand since it can create a bad day.

Azure for Red Teams

Accessing the Azure portal through the web browser is great and has many awesome advantages but I have yet to find a way to export the information directly. I started to write a tool that would authenticate and do it in an automated fashion but it felt cumbersome and I knew with all of these awesome technologies tied together that Microsoft has solved this problem for me. There were a number of solutions I came across, some of them are:

Azure CLI (AZ CLI)

Being a Linux user, I naturally gravitated towards AZ CLI. Partially because I pipe as much data into one-liners as possible and partially because I over-engineer tools in .NET. Using AZ CLI is a quick and easy way to authenticate against the OAUTH for Azure while also quickly exporting the raw data. In this post, we will focus on this solution.

Azure Powershell

With a rise in awesome Powershell tools like Powershell Empire and MailSniper, I'm amazed that Azure Powershell hasn't made its way into one of these tools. There are a massive number of Active Directory Cmdlets to interact with. To get started, simply install Azure RM Powershell then run: Connect-AzureRmAccount

Azure .NET

I am one of those weird nerds who grew up on Linux but wrote C# for a significant portion of my career. Because of this, having an Azure .NET library to interact with Active Directory is encouraging. I didn't dig too much into these libraries but from a high-level, it seems they are some sort of wrapper for the Active Directory Graph API.

Let's Dig In!

As I previously mentioned, we will focus on interacting with Azure using AZ CLI. In order to get started, we have to first establish an active session with Azure. On red teams where the engagement involves an organization using Microsoft or Google services, I rarely try to go straight to a shell on the internal network. I will normally use a tool I wrote called CredSniper to phish credentials and multi-factor tokens than just authenticate as that user in pursuit of sensitive emails, files, access, information, and VPN.

Will that presupposition, we will assume valid credentials were already obtained somehow.

Install AZ CLI

You will need to add the Microsoft source to apt (assuming Linux), install the Microsoft signing key, and then install Azure CLI:

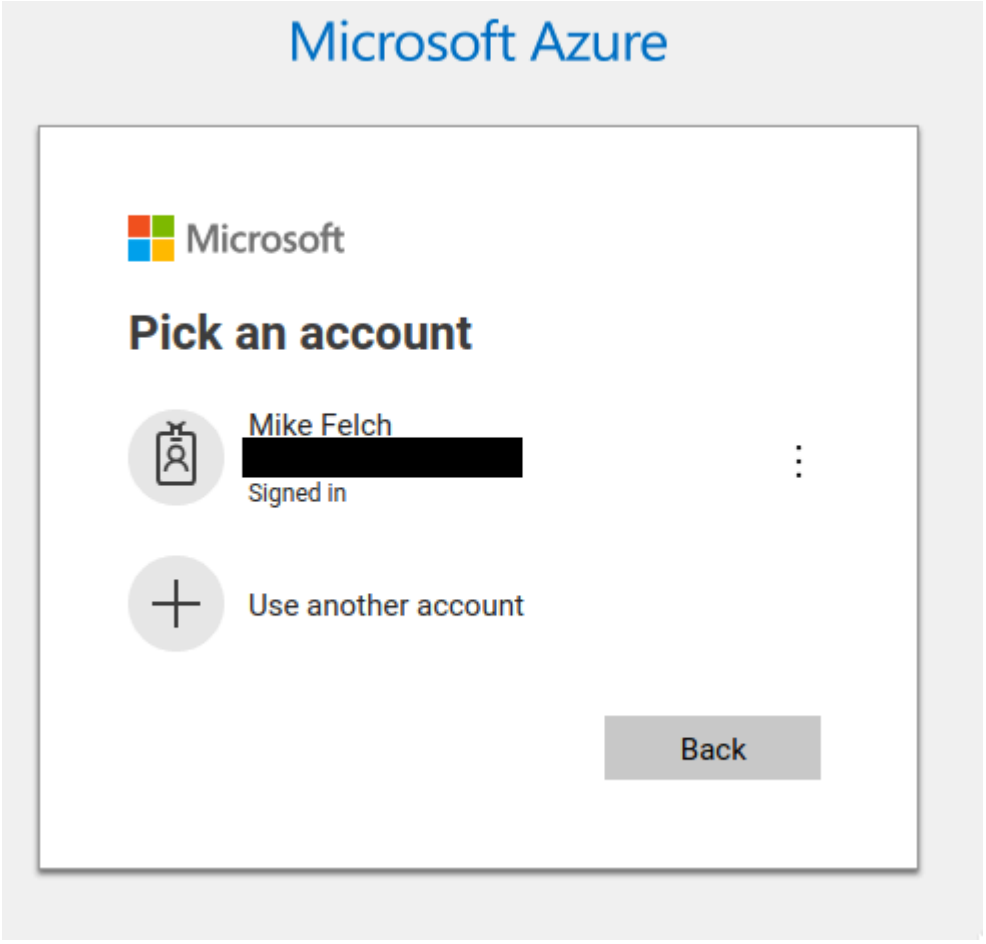
```
AZ_REPO=$(lsb_release -cs) echo "deb [arch=amd64] https://packages.microsoft.com/repos/azure-cli/ $A

curl -L https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key add -

sudo apt-get install apt-transport-https

sudo apt-get update && sudo apt-get install azure-cli
```

Authentication via Web Session

After everything is installed correctly, you will need to create a session to Azure using the credentials you already obtained. The easiest way to do that is by authenticating using ADFS or OWA in a normal browser then:

```
az login
```

This will generate the OAUTH tokens locally, open a browser tab to the authentication page and let you select an account based on the ones you are already authenticated with. Once you select the account, the local OAUTH tokens will be validated by the server and you won't have to do that again unless they expire or get destroyed. You can also pass the –use-device-code flag which will generate a token you provide to https://microsoft.com/devicelogin.



Dumping Users

Now on to my favorite part! There have been numerous techniques for extracting the GAL previously researched, such as using the FindPeople and GetPeopleFilter web service methods in OWA. These techniques have been an excellent resource for red teamers but they definitely have their limitations on what data is available, how long it takes to enumerate users, how loud it is due to the number of web requests required, and how it occasionally breaks. With AZ CLI, it's super easy to extract all the directory information for each user. In the examples below, I apply a JMESPath filter to extract the data I care about. I can also export as a table, JSON, or in TSV format!

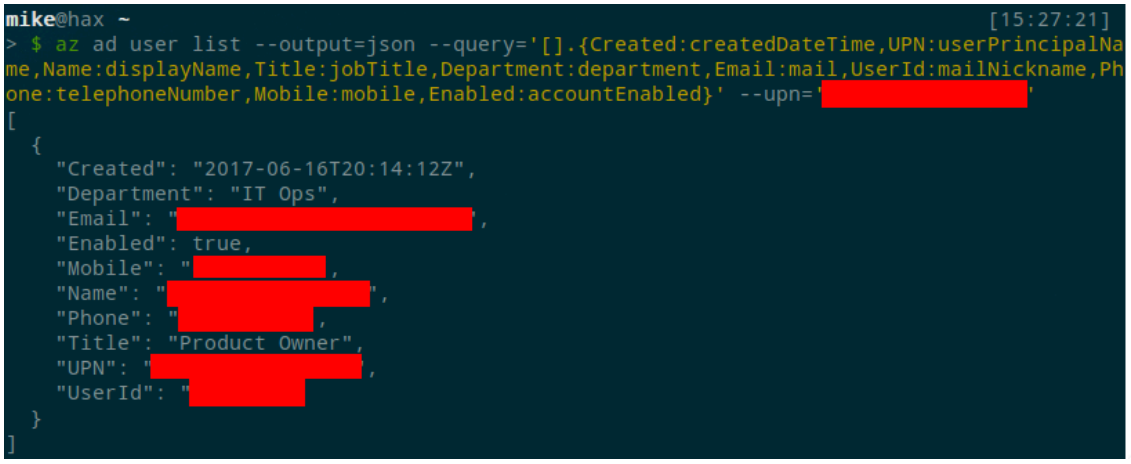
All Users

```
az ad user list --output=table --query='[].{Created:createdDateTime,UPN:userPrincipalName,Name:displ
```

Specific User

If you know the UPN of the target account, you can retrieve specific accounts by passing in the —upn flag. This is convenient if you are wanting to dig into the Active Directory information for a particular account. In the example below, you will notice I supplied the JSON format instead of the table output.

```
az ad user list --output=json --query='[].{Created:createdDateTime,UPN:userPrincipalName,Name:displa
```



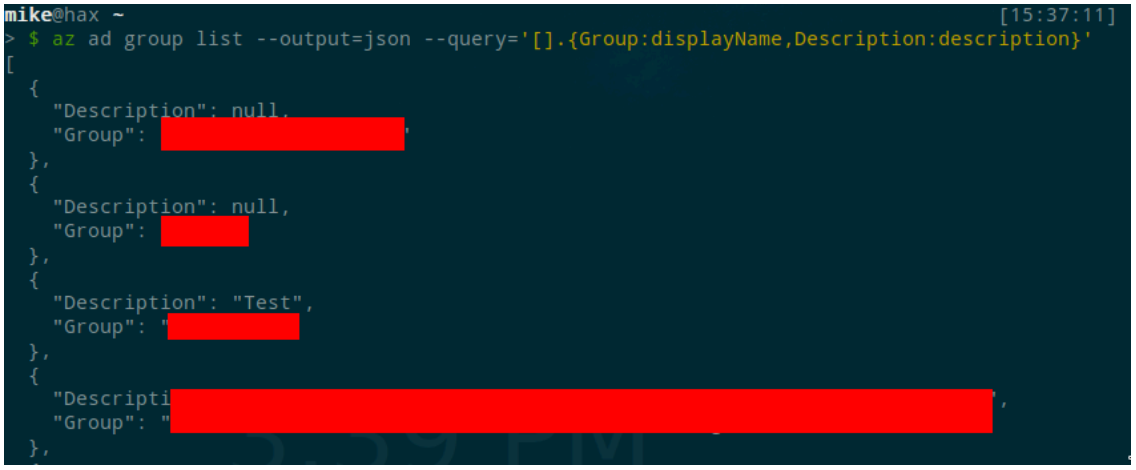
Dumping Groups

My next favorite function is the ability to dump groups. Understanding how groups are used within an organization can provide specific insight into the areas of the business, the users, and who the admins are. AZ CLI provides a few useful commands that can assist here.

All Groups

The first thing I usually do is just export all the groups. Then I can grep around for certain keywords: Admin, VPN, Finance, Amazon, Azure, Oracle, VDI, Developer, etc. While there is other group metadata available, I tend to just grab the name and description.

```
az ad group list --output=json --query='[].{Group:displayName,Description:description}'
```



Specific Group Members

Once you have reviewed the groups and cherry-picked the interesting ones, next it's useful to dump the group members. This will give you an excellent list of targets that are a part of the interesting groups — prime targets for spear phishing! Against popular opinion, I have personally found that the technical ability and title do not lower the likelihood an intended target is more likely to avoid handing over their credentials (and even MFA token). In other words, everyone is susceptible so I usually target back-end engineers and devops teams because they tend to have the most access plus I can usually remain external to the network yet still access private GitHub/GitLab code repositories for creds, Jenkins build servers for shells, OneDrive/GDrive file shares for sensitive data, Slack teams for sensitive files and a range of other third-party services. Once again, why go internal if you don't have to.

```
az ad group member list --output=json --query='[].{Created:createdDateTime,UPN:userPrincipalName,Nam
```

Dumping Applications & Service Principals

Another nice feature Microsoft provides is the ability to register applications that use SSO/ADFS or integrate with other technologies. A lot of companies utilize this for internal applications. The reason this is nice for red teamers is that the metadata associated with the applications can provide deeper insight into attack surfaces that may have not been discovered during reconnaissance, like URLs.

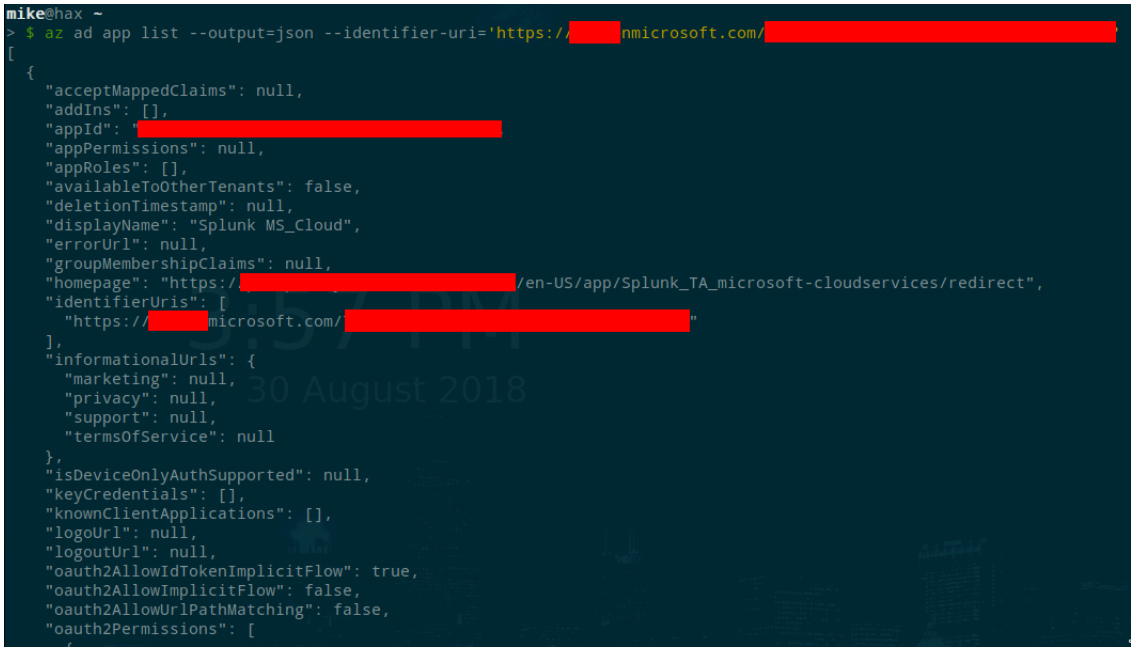
All Applications

```
az ad app list --output=table --query='[].{Name:displayName,URL:homepage}'
```

Specific Application

In the below screenshot, you see we obtained the URL for the Splunk instance by examining the metadata associated with the registered application in Azure.

```
az ad app list --output=json --identifier-uri='<uri>'
```



All Service Principals

```
az ad sp list --output=table --query='[].{Name:displayName,Enabled:accountEnabled,URL:homepage,Publi
```

Specific Service Principal

```
az ad sp list --output=table --display-name='<display name>'
```

Advanced Filtering with JMESPath

You might have noticed in the above examples that I try to limit the amount of data that is returned. This is mainly because I try to snag what I need instead of everything. The way AZ CLI handles this is by using the –query flag with a JMESPath query. This is a standard query language for interacting with JSON. I did notice a few bugs with AZ CLI when combining the query flag with the ‘show’ built-in functions. The other thing to note is that the default response format is JSON which means if you plan on using a query filter you need to specify the correct case-sensitive naming conventions. There was a bit of inconsistency between the names for the different formats. If you used the table format, it might capitalize when JSON had lowercase.

Disable Access to Azure Portal

I spent a bit of time trying to make sense of what to disable, how to prevent access, how to limit, what to monitor, and even reached out to people on Twitter (thanks Josh Rickard!). I appreciate all the people who reached out to help make sense of this madness. I suppose I should learn the Microsoft ecosystem more, in hopes of offering better suggestions. Until then, I offer you a way to disable the Azure Portal access to users. I haven’t tested this and can’t be sure if this includes AZ CLI, Azure RM Powershell, and the Microsoft Graph API but it’s definitely a start.

Step 1) Log in to Azure using a Global Administrator account https://portal.azure.com

Step 2) On the left panel, choose ‘Azure Active Directory’

Step 3) Select ‘Users Settings’

Step 4) Select ‘Restrict access to Azure AD administration portal’

An alternative is to look into Conditional Access Policies: https://docs.microsoft.com/en-us/azure/active-directory/conditional-access/overview

Coming Soon!

There are a number of different tools out there for testing AWS environments and even new tools that have come out recently for capturing cloud credentials like SharpCloud. Cloud environments seem to be a commonly overlooked attack surface.

I will be releasing a (currently private) red team framework for interacting with cloud environments, called CloudBurst. It’s a plugginable framework that gives users the ability to interact with different cloud providers to capture, compromise, and exfil data.
