Introduction

iLAPS implementation based off of PowerShell – Intune Local Administrator Password Solution (iLAPS)

Powershell Implementation:

- Installation script to install and create windows task to run monthly on the first
- Powershell reset script to reset all local admin passwords
- Powershell script to check if password reset is required hourly

Admin UI Web Application Features:

- Search for machines using a configurable machine prefix
- Automatically reseting machine password after configurable amount of hours if not an SuperUser
- Ability for SuperUsers to not trigger an automatic password reset
- Logging of All User Actions
- `User` and `SuperUser` roles in AD for Admin Users

Objectives

1. Deploy Automatically
2. Create Admin UI with Ability to Limit access via Azure AD
3. Automatically Reset passwords which have been viewed by an admin to verify passwords are always random

Getting Started Overview

1. Installation process
   1. Create Customer Specific Secrets File (Steps below)
   2. Build Software dependencies (Steps below)
   3. Deploy Software
2. Software dependencies
   1. GIT Commandline
   2. Dotnet Core 3.1 SDK
   3. Powershell 3+

Local Configuration

1. Clone solution into `c:\dev`
2. Navigate to `c:\dev\iLAPs`
3. Copy `settings.template.json` with name `settings.production.local.json`
   1. You can add as many `settings.environment-name.local.json` as you want if you want to be able to build for many environments. To use the new environments modify step `13.2` to look like `.\Build.ps1 -BuildEnvironment 'environment-name'`
   2. If `BuildEnvironment` is not specified it assumes `Production`

Portal Configuration

1. Create Resouce Group Named `iLaps-RG`
2. Create General Purpose V2 Storage Account (Example name `ilapscustomername`)
    1. Save storage account name into `settings.production.local.json` field named

    ```
        "Storage-Account-Name": "PasteValueHere"
    ```

3. Create Shared Access Signature for `Admin UI`
    1. Allowed Services: Table
    2. Allowed Resource Types: Object
    3. Allowed Permissions: Read, Write, List, Add, Create, Update
    4. Set Start and End Expiration dates
    5. Allowed Protocols: Https only
    6. Generate SAS and Connection String
    7. Save into `settings.production.local.json` field shown below

    ```
        "Admin-UI-Table-Object-Read-Write-List-Add-Create-Update-SAS-Token": "PasteValueHere"
    ```

4. Scroll down to `Tables` on side navigation bar:
    1. Create Table called `AdminPasswords`
    2. Create Table called `ResetPasswords`
    3. Create Table called `Logs`
    4. Create Table called `DEMPasswords`
    5. Save into `settings.production.local.json` field shown below

    ```
        "Admin-Table-Name": "AdminPasswords",
        "Reset-Table-Name": "ResetPasswords",
        "DEM-Table-Name": "DEMPasswords",
        "Log-Table-Name": "Logs"
    ```

5. Click `Tables` click the elipsis on `AdminPassword` table
    1. Select Access Policy
    2. Click Add Policy

   3. Name it `Add-Create`
   4. Permissions `Add` and `Update`
   5. Set start and end time based on your organzations security policy
   6. Click Ok
   7. Create another Policy named `Read` with `Read` Permissions
   8. Click Ok

6. Click `Tables` click the elipsis on `ResetPasswords` table

   1. Select Access Policy
   2. Click Add Policy
   3. Name it `Read-Update`
   4. Permissions `Read` and `Update`
   5. Set start and end time based on your organzations security policy

7. Create Shared Access Signature for `Installation Script`

   1. Allowed Services: Blob

   2. Allowed Resource Types: Object

   3. Allowed Permissions: Read

   4. Set Start and End Expiration dates

   5. Allowed Protocols: Https only

   6. Generate SAS and Connection String

   7. Save into `settings.production.local.json` field shown below

      ```
      "Blob-Object-Read-Installer-SAS-Token": "PasteValueHere"
      ```

8. Scroll down to `Containers` on side navigation bar:

   1. Create Container named `installation`

   2. Save into `settings.production.local.json` field shown below

      ```
      "Installer-Container-Name": "installation"
      ```

9. Navigate back to `iLaps-RG` Resource Group

   1. Click `Add` then search for WebApp
   2. Name WebApp `iLaps-customername` where customername is your customer's name
   3. Runtime Stack: `.Net Core 3.1 (LTS)`
   4. App Service Plan:
      1. Create New and name it

        2. Change Size to `S1`
    5. Click `Review + Create`
    6. Click `Create`
    7. Once created naviagte to `TLS/SSL` in new resource
        1. Turn on the `HTTPS Only` setting

10. Navigate to `Azure Active Directory`

    1. Click `App Registrations`

    2. Click `New Registration`

    3. Name Application `ILAPS`

    4. Click `Authentication` and click `Add a Platform`

        1. Click `Web`
        2. Specify the url as
            1. US Gov Cloud
                1. `https://ilaps-customername.azurewebsites.us/signin-oidc`
            2. US Commercial Cloud
                1. `https://ilaps-customername.azurewebsites.com/signin-oidc`
        3. Add another url for development purposes `https://localhost:5001/signin-oidc`
        4. Click `Configure`

    5. Navigate to Manifest and replace line 8 which contains `appRoles` with the following

```
	"appRoles": [
		{
			"allowedMemberTypes": [
				"User"
			],
			"description": "SuperUser user can view passwords but they DO NOT reset automatically",
			"displayName": "SuperUser",
			"id": "b4a94c3f-bdd5-4c86-8749-d7f110195a56",
			"isEnabled": true,
			"lang": null,
			"origin": "Application",
			"value": "SuperUser"
		},
		{
			"allowedMemberTypes": [
				"User"
			],
			"description": "Helpdesk user can view passwords but they reset automatically",
			"displayName": "User",
			"id": "574cd779-fece-4f33-aa31-d1374e8ea5ca",
			"isEnabled": true,
			"lang": null,
			"origin": "Application",
      "value": "User"
   },
   {
      "allowedMemberTypes": [
         "User"
      ],
      "description": "User can only see DEM User Tab",
      "displayName": "DEM",
      "id": "574cd779-fece-4f33-aa31-d1374e8ea5c2",
      "isEnabled": true,
      "lang": null,
      "origin": "Application",
      "value": "DEM"
   }
],
```

6. Navigate to Manifest and find `publisherDomain` and remember the value for step 8

7. Navigate to `Certificates & secrets` and generate a new secret then save it in step 8

8. Navigate to Overview tab to save the fields below into `settings.production.local.json`

```
"Admin-UI-Domain": "Type publisherDomain here"
"Admin-UI-TenantId": "Tenant-GUID",
"Admin-UI-ClientId": "App-Registration-Client-GUID",
"Admin-UI-ClientSecret":"App-Registration-Client-Secret",
```

9. Click `API Permissions`

   1. Click `Add a Permission`
   2. Click `Microsoft Graph`
      1. Search and add `Group.Read.All` and `User.Read`
      2. Click `Add Permission`
      3. Get admin permissions granted for `Group.Read.All` (You need this or application could fail to work properly)

10. Click `Overview` and in the top header click the link next to `Managed application in local directory`

    1. Click `Properties`
    2. Toggle `User Assignment required` to `Yes`
    3. Click `Users and Groups` and add users who should have access to this application.
       1. Add Role based on if the User is a `User`, `DEM` or a `Super User`. `DEM` has ability to ONLY SEE the DEM tab. `Super Users` have the ability to view DEM Tab, passwords without forcing a reset automatically and view access logs

11. Open `settings.production.local.json` and change set the following settings based on if you are targeting `US Gov Cloud` or `US Commercial Cloud` and your `Customers Name`

1. US Gov Cloud

   ```
       "Company-Name": "My Gov Customer Name",
       "Storage-Account-Suffix":"core.usgovcloudapi.net",
       "Admin-UI-Instance":"https://login.microsoftonline.us/",
       "Admin-UI-GraphApiUrl" : "https://graph.microsoft.us/beta"
   ```

2. US Commercial Cloud

   ```
       "Company-Name": "My Commercial Customer Name",
       "Storage-Account-Suffix":"core.windows.net",
       "Admin-UI-Instance":"https://login.microsoftonline.com/",
       "Admin-UI-GraphApiUrl" : "https://graph.microsoft.com/beta"
   ```

12. If you will be using the DEM Management feature, Please Configure your DEMAdminGroups and DEMSuperAdminGroup. Below is an example how to add multiple to each group. DEMAdminGroups can see DEMAccounts Associate to them and DEMSuperAdmin can see all DEM Accounts. **IF NOT USING DEM ACCOUNT FEATURE PLEASE CLEAR THE FIELDS LIKE SHOWN IN EXAMPLE 2**

    1. Configure DEM Admins **(if configured please read last step to finish configuration)**

       ```
           "Admin-UI-DEMAdminGroups" : "
       ['DEM_ADMIN_GROUP_NAME','DEM_ADMIN_GROUP_NAME_TWO']",
           "Admin-UI-DEMSuperAdminGroups" : "
       ['DEM_SUPERADMIN_GROUP_NAME','DEM_SUPERADMIN_GROUP_NAME_TWO']",
       ```

    2. Disable the Feature

       ```
           "Admin-UI-DEMAdminGroups" : "",
           "Admin-UI-DEMSuperAdminGroups" : "",
       ```

13. Open `Azure Storage Explorer`

    1. Login to Azure and find the storage account we just created

    2. Open the `Tables` section

    3. Right Click `AdminPasswords` table

       1. Click `Get Shared Access Signature...`

       2. Click `Access Policy` and select `Add-Create`

       3. Click Create

        4. Copy the `Query String`

            1. Save into `settings.production.local.json` field named

                ```
                "Table-Object-Add-Create-SAS-Token": "PasteValueHere"
                ```

        5. Click `Back` and change the `Access Policy` to `Read`

        6. Click Next

        7. Copy the `Query String`

            1. Save into `settings.production.local.json` field named

                ```
                "Table-Object-Read-List-SAS-Token": "PasteValueHere"
                ```

    4. Right Click `Reset Passwords` table

        1. Click `Get Shared Access Signature...`
        2. Click `Access Policy` and select `Read-Update`
        3. Click Create
        4. Copy the `Query String`

            1. Save into `settings.production.local.json` field named

                ```
                "Table-Object-Read-Update-SAS-Token": "PasteValueHere"
                ```

14. Ensure you have .NET Core 3.1 SDK installed

    1. Open Powershell window and navigate to `c:\dev\iLAPs`
    2. run `.\Build.ps1`

15. Open `c:\dev\iLAPs\Output\app-service-advanced-editor-script.json`

    1. Select all text and copy

16. Navigate to `iLaps-RG` in the portal

    1. Click `ilaps` App Service
    2. Click `Configuration`
        1. Click `Advanced Edit`
        2. Paste the copied value right before the last `]`
        3. Click `OK`
    3. Scroll down to `Advanced Tools` in the side navigation
        1. Click `Go`
        2. Hover over `Tools` then click `Zip Push Deploy`

        3. Open `c:\dev\iLAPs\Output\` in File Explorer
        4. Drag `AdminUI.zip` to Zip Deploy Interface (You will see it turn blue)

17. Navigate back to `iLaps-RG` Resource Group

    1. Click `ilapscustomername` storage account
    2. Click `File Shares`
    3. Click `installation`
    4. Click `Upload`
        1. Navigate to `c:\dev\iLAPs\Output`
        2. Click both `Reset-LocalAdministratorPassword_v2.0.ps1` and `Check-Reset-LocalAdministratorPassword_v2.0.ps1`
        3. Click `Upload`

18. Navigate to Use PowerShell scripts on Windows 10 devices in Intune

    1. Deploy the code found in `c:\dev\iLAPs\Output\Install-iLaps_v2.0.ps1` using the guide linked above

19. If using DEM feature complete this step. Otherwise ENJOY!

    1. Install and connect to storage account using storage explorer
    2. Fill out the `DEMPasswords - Import Template.csv` script and hash the password using the hashing script in the Output directory.
    3. The hashing script takes a string input and returns a hashed output. You can automate the script further on your own but this is provided as a starting point. Example Usage : `.\Output\SaltDEMPasswords.ps1 -PW "MyCrazySuperSecretPassword123!@#"`
    4. Navigate to `DEMPasswords` table and import the `DEMPasswords - Import Template.csv`
    5. Manually copy the `DEM-Password-Reset-Script.ps1` from the `Outputs` folder to `C:\Windows\System32` folder on a On-Prem server with AD Powershell Tools installed
    6. Open an Admin Powershell and navigate to `C:\Windows\System32`
    7. Run the command `.\DEM-Password-Reset-Script.ps1`
    8. The script will self install a task which run's hourly and checks the DEM table if it needs to update the Password or not.
    9. Enjoy