LAB GUIDE

Analyzing Microsoft 365 through PowerShell

Breakout Session Three – Modify a simple script to check for security parameters.

Overview

This breakout session focuses on implementing a sample script to iterate through some key control items that can be detected using the MSOL module. The script below is meant to provide a general starting point to be filled out according to your organizational needs. Use the reference to additional MSOL module commands to look at specific parameters which may interest you.

Objectives

- Run the vanilla script and get a result
- Modify the script by adding in automated logon (previous lab)
- Add additional checks based on the Commands worth reviewing section

Commands worth reviewing

The script provides a sample look at using the MSOL service to lookup users who may have an expired password based on a 90-day policy. For this exercise, identify additional/alternative properties that are interesting and evaluate them through the script. Some recommendations below are provided for each module.

Module	Command	Parameter	Equals
Azure Ad	Get-AzureAdApplication	PasswordCredentials	$False
Msol	Get-MsolUser	StrongAuthenticationRequired	$True
Exchange	Get-AdminAuditLogConfig	AdminAuditLogEnabled	$True
SharePoint	Get-SpoTenant	ExternalUserExpirationRequired	$True
Teams	Get-CsTeamsMeetingPolicy	AllowAnonymousUsersToJoinMeeting	$False

Script

The script takes no input and evaluates all users based on an assumed 90-day password expiration policy. This can be modified in the main variables setting found near the top of the script. At execution, it will prompt you for your scan user credentials. With those permissions, it grabs a list of all domain users and then evaluates each user with the Get-ExpiredPasswordUsers function. This function in turn populates a list of users who failed the 90-day password change. At the close of the script, the list is displayed in the terminal.

```
# -- MSOL Security Automation Tool -- #
#  Author:                           #
#  Version:                          #
#  Date:                             #
# -- -- -- -- -- -- -- -- -- -- -- -- #

# -- Imports
Import-Module MSOnline

# -- Variables
$PasswordPolicyExpireDays = 90
$Script:UsersWithOldPasswords = @()

# -- Functions
Function Get-ExpiredPasswordUsers() {
    Param(
        # Should be a positive integer, but will handle later
        [Parameter(Mandatory=$True)]
        [int]$Days,

        [Parameter(Mandatory=$True)]
        [array]$UserList
    )
```

```
    # Func Variables
    $TotalUsers = $UserList.Count
    $SearchedUsers = 0
    $AbsoluteDays = [Math]::Abs($Days)
    Try {
        # Get Date that is (x) number of days behind today
        $ExpirationDate = (Get-Date).AddDays(-1*$AbsoluteDays)

        # Iterate through user list
        ForEach($User in $UserList) {
            $SearchedUsers++
            Write-Progress `
                -Activity "Old password lookup." `
                -Status "Checking: $($User.DisplayName)" `
                -PercentComplete ($SearchedUsers*100/$TotalUsers)

            If ($User.LastPasswordChangeTimestamp -lt $ExpirationDate) {
                # Append to user list
                $Script:UsersWithOldPasswords += ($User)
            }
        }

        Return $True
    } Catch {
        Write-Host $_
        Return $False
    }
}

# -- Logic
# -- Logic -- Get Signed In
Connect-MsolService

# -- Logic -- Run a command, store in a variable
$Users = Get-MsolUser -All

# -- Logic -- Review Results
If(
    (Get-ExpiredPasswordUsers `
        -Days $PasswordPolicyExpireDays `
        -UserList $Users) ) {

        $UsersWithOldPasswords | Select-Object `
            UserPrincipalName, LastPasswordChangeTimestamp

} Else {
    Write-Host "An error has occurred and users could not be evaluated."
    Write-Host "Check permissions and configuration."
}
```