Regularly clean up MEID guest user with automation

> Requirements: Create an App Registration with AuditLog.Read.All and User.ReadWrite.All. For logging you need to have an Azure Function to create an API to centralize logs in an Azure Log Analytics Workspace.

This automation gets all Microsoft Entra ID Guest User via the Microsoft Graph API and checks everyone for their latest sign in. If the login was more than 6 months ago, the user account will be deleted from the Microsoft Entra ID. Afterwards, the key figures for the deleted devices and the errors are passed to the Azure Function API, where they are then stored in the corresponding Log Analytics Workspace.

Use case

This automation concept is perfectly suited for large enterprises which don't have an overview of all of their Microsoft Entra ID Guests. Authentication in the direction of Microsoft Entra ID is user-independent with an app registration. This makes it perfect for automated and regular execution of this code. In combination with an Azure Runbook that runs weekly, the Microsoft Entra ID environment can be simplified significantly.

PowerShell script

This script has no dependency on any modules and is based on all standard powershell modules. After you organized the prerequisites (App Registration with AuditLog.Read.All and User.ReadWrite.All permission and an Azure Function API for centralizing the logs) you can fill the four variables to the PowerShell code and run the script.

```
$tenantId= "<yourtenantid>"
$ClientId= "<yourclientidofappregistration>"
$ClientSecret = "<yourclientsecretofappregistration>"
$url="<yoururlofcentralizedlogcollectionfunction>"
```

```
$expirationThresholdDays = 180
$NotificateOnDays = @(1,5,10,30)
$SenderUPN = "<yoursenderupn>"
$Subject = "<yourinformationsuccess>"

Function Send-Logs(){

    param (
        [String]$LogType,
        [Hashtable]$LogBodyList
    )

    $LogBodyJSON = @"
    {
        "logtype": "$LogType",
        "logbody": {}
    }
"@

    $LogBodyObject = ConvertFrom-JSON $LogBodyJSON
    Foreach($Log in $LogBodyList.GetEnumerator()){
        $LogBodyObject.logbody | Add-Member -MemberType NoteProperty -Name $log.key -Value $log.value
    }
    $Body = ConvertTo-JSON $LogBodyObject

    $Response = Invoke-Restmethod -uri $url -Body $Body -Method POST -ContentType "application/json"
    return $Response
}

function Send-Mail {

    param (
        [String]$SenderUPN,
        [Array]$Recipients,
        [Array]$CCRecipients,
        [String]$Subject,
        [String]$Content,
        [String]$DisplayName,
        [Int]$deletionDay,
```

```
        [String]$UPN
    )

    $Content = @"
    Hello $DisplayName
    <yourmailbody>
"@

    $MailBodyJSON = @"
  {
    "message": {
      "subject": "$Subject",
      "body": {
        "contentType": "Text",
        "content": ""
      },
      "toRecipients": [],
      "ccRecipients": []
    },
    "saveToSentItems": "true"
  }
"@

  $MailBodyObject = ConvertFrom-Json $MailBodyJSON
  $MailBodyObject.message.body.content = $Content

  Foreach($Recipient in $Recipients){
        $RecipientBodyJson = @"
  {
    "emailAddress": {
        "address": "$Recipient"
    }
  }
"@
        $RecipientBodyObject = ConvertFrom-JSON $RecipientBodyJson
        $MailbodyObject.message.toRecipients += $RecipientBodyObject
    }

    Foreach($CCRecipient in $CCRecipients){
        $CCRecipientBodyJson = @"
```

```
  {
    "emailAddress": {
       "address": "$CCRecipient"
    }
  }
"@
       $CCRecipientBodyObject = ConvertFrom-JSON $CCRecipientBodyJson
       $MailbodyObject.message.ccRecipients += $CCRecipientBodyObject
    }

    $MailoutputbodyJson = ConvertTo-JSON $MailbodyObject -Depth 10

    Write-Host "Sending Mail to $Recipient and $CCRecipient (CC)."
    Invoke-RestMethod -Method Post -Uri "https://graph.microsoft.com/v1.0/users/$SenderUPN/sendMail" -Headers
$Header -ContentType "application/json" -Body ([System.Text.Encoding]::UTF8.GetBytes($MailoutputbodyJson))
}


$Body = @{
"tenant" = $TenantId
"client_id" = $ClientId
"scope" = "https://graph.microsoft.com/.default"
"client_secret" = $ClientSecret
"grant_type" = "client_credentials"
}
$Params = @{
"Uri" = "https://login.microsoftonline.com/$TenantId/oauth2/v2.0/token"
"Method" = "Post"
"Body" = $Body
"ContentType" = "application/x-www-form-urlencoded"
}
$AuthResponse = Invoke-RestMethod @Params

$Header = @{
    "Authorization" = "Bearer $($AuthResponse.access_token)"
}

$Result = Invoke-RestMethod -Method GET -Uri "https://graph.microsoft.com/beta/users?`$filter=userType eq
'Guest'&`$select=userPrincipalName,signInActivity,displayName" -Header $Header
$GuestUsers = $Result.value
```

```
while ($Result.'@odata.nextLink') {
    $Result = Invoke-RestMethod -Uri $Result.'@odata.nextLink' -Headers $Header
    $GuestUsers += $Result.value
}
$failedDateGuestUsers = @()
$failedDeletedGuestUsers = @()
$failedNotifiedGuestUsers = @()
$successfullynotifiedGuestUsers = @()
$successfullydeletedGuestUsers = @()
foreach($GuestUser in $GuestUsers){
    try{
        $NonInteractiveDayspan = [math]::Round($(New-TimeSpan -Start $(Get-Date
$GuestUser.signInActivity.lastNonInteractiveSignInDateTime) -End $(Get-Date)).TotalDays)
        $InteractiveDayspan = [math]::Round($(New-TimeSpan -Start $(Get-Date
$GuestUser.signInActivity.lastSignInDateTime) -End $(Get-Date)).TotalDays)
    } catch {
        $failedDateGuestUsers += $GuestUser
    }

    if ($NonInteractiveDayspan -lt $InteractiveDayspan) {
        $deletionDay = $expirationThresholdDays - $NonInteractiveDayspan
    } else {
        $deletionDay = $expirationThresholdDays - $InteractiveDayspan
    }

    if($deletionDay -in $NotificateOnDays -or $deletionDay -lt 0){ # DELETE -or $deletionDay -lt 0
        if($deletionDay -le 0){
            $deletionDay = "paar"
        }
        try{
            $GuestUserMail = (Invoke-RestMethod -Method GET -Uri
"https://graph.microsoft.com/v1.0/users/$($GuestUser.id)?`$select=mail" -Header $Header).mail
            Send-Mail -Recipients @("$GuestUserMail") -SenderUPN $SenderUPN -DisplayName
$GuestUser.displayName -UPN $GuestUser.userPrincipalName -DeletionDay $deletionDay -Subject $Subject
            $successfullynotifiedGuestUsers += $GuestUser
        }
        catch{
            $failedNotifiedGuestUsers += $GuestUser
        }
    }
```

```
    if(($deletionDay -lt -5)){
        try{
            $Result = Invoke-RestMethod -Method DELETE -Uri
"https://graph.microsoft.com/v1.0/users/$($GuestUser.id)" -Header $Header
            $successfullydeletedGuestUsers += $GuestUser
        }catch{
            $failedDeletedGuestUsers += $GuestUser
        }
    }
}

Write-Warning "Date Errors: $($failedDateUsers.count)"
Write-Warning "Delete Errors: $($failedDeletedGuestUsers.count)"
Write-Warning "Notification Errors: $($failedNotifiedGuestUsers.count)"
Write-Output "Notified GuestUsers: $($successfullynotifiedGuestUsers.count)"
Write-Output "Deleted GuestUsers: $($successfullydeletedGuestUsers.count)"

$Logs = @{
    "deletesuccess"="$($successfullydeletedGuestUsers.count)"
    "notificationsuccess"="$($successfullynotifiedGuestUsers.count)"
    "notificationerrors"="$($failedNotifiedGuestUsers.count)"
    "deleteerrors"="$($failedDeletedGuestUsers.count)"
    "dateerrors"="$($failedDateUsers.count)"
}

Send-Logs -LogType "CleanUpMEIDGuestUserExecutions" -LogBodyList $Logs
```

Revision #8
Created 14 December 2022 19:10:14
Updated 21 July 2024 15:11:16