Advanced Scripting   
Access Rights in Active Directory

Last Updated: 4/20/2024 4:01 PM Version 1  
Document Prepared for: CYBER360 Student

# Name Click here to enter name ID Click here to enter id

# Instructions

Answer all questions directly in this document. You will save and upload this completed document as your homework submission.

# Overview

Refer to the textbook (chapter 6) for an introduction to Active Directory from a security point of view, and how attacks work in a corporate environment.

Access to resources in an Active Directory domain is restricted according to its security mechanisms: *security identifiers* (SIDs), tokens, and *access control entries* (ACEs) contained in *access control lists* (ACLs). ACLs are also known as *security descriptors*.

# Requirements

Internet connection, and an SSH client.

# Setup

Use SSH again to connect to your account at **cit361-lab.cit.byui.edu**.

# Task 1—SIDs

As you have seen in previous exercises, SIDs start with the letter **S**, followed by a sequence of numeric values joined by hyphens.

## Steps

1. Make sure you are logged in to the lab VM. (The **hostname** command should return **horace**.)
2. Get the SID of your account:   
   **Get-ADUser -Identity** *<your-username>* # replace *<your-username>* with your account name   
   What’s the SID of your account? Click or tap here to enter text.
   1. You may recall from your reading of chapter 6 that user account SIDs follow this pattern:   
      **S-1-5-21-**<three longer numerals that uniquely represent the domain>**-**<user number>,   
      where the last number is 500 for the domain administrator account, or 1000 or greater for other user accounts.   
      What are the three numbers that represent the **cit361.com** domain? Click or tap here to enter text.
      1. What is the user number of your domain account? Click or tap here to enter text.
   2. *Administrator* is the default account name (and default login name) for the administrator of a Windows workstation, server, or AD domain. A commonly recommended security practice is to change these names, so that they aren’t obvious default account names that attackers can target. Write a PowerShell command that will find the (altered) names of the **cit361.com** administrator account. (*Hints*: you can use a dollar character $ at the end of a regular expression to match the end of a string; the regex '-500$' will match a string that ends with the four characters -500. An ADUser object’s sAMAccountName property contains the user’s login-name, and the object’s Name property contains the user’s formal name.)
      1. Your command: Click or tap here to enter text.
      2. What are the UserPrincipalName values revealed by your command? Click or tap here to enter text.

# Task 2 —Access Control Lists and Access Control Entries

Whenever a Windows application tries to access a resource, Windows’ Security subsystem consults the ACEs found in ACLs to determine whether to deny access, to grant access, or to let some other ACE make that determination.

## Steps

1. Look at the metadata of \ (the root directory) on horace:   
   **Get-Item C:\ | fl -Property \***   
   Find the **Attributes** property. What are the file attributes applied to the root folder? Click or tap here to enter text.
2. Those file attributes are there for backward compatibility with earlier Microsoft operating systems, all the way back to DOS. They’re limited to just a few settings, such as ‘Readonly,’ ‘Hidden,’ ‘System,’ or ‘Archive.’ But ever since Windows NT, Windows uses a more robust access control regime.
3. Look at the Access Control List of the root directory:   
   **Get-Acl \**
   1. An ACL is also generically known as a *security descriptor*. Save the result to a variable:   
      **$sd = Get-Acl \**
   2. What is the full name of the type of this object?   
      **$sd.GetType().FullName**   
      Your output: Click or tap here to enter text.
   3. What are the members (methods and properties) of this object? Enter the command line you used to find out (*not* your command’s output): Click or tap here to enter text.
   4. What is the name of the service account or group (probably the root folder’s original creator) that owns **C:\** ?   
      **$sd.Owner**   
      Your output: Click or tap here to enter text.
   5. Look at its **Access** property. (You may have noticed that its MemberType is *CodeProperty*. That just means it’s a member that works like a method, but is accessed like a property.)   
      **$sd.Access**
      1. This looks like the actual access control list. What’s the full name of its type?   
         **$sd.Access.GetType().FullName**   
         Your output: Click or tap here to enter text.
      2. Again, the ACL is a collection (an array) of access control entries. Enter:   
         **$sd.Access | Foreach\_Object { $\_.GetType().FullName }**   
         What’s the full name for the type of each ACE? Click or tap here to enter text.
      3. Let’s list all of the IndentyReferences found among the ACEs in the ACL:   
         **$sd.Access.IdentityReference**   
         Your output (one line, comma-separated): Click or tap here to enter text.
      4. List all of the FileSystemRights found among the ACEs in the ACL. What command (not its output) did you use to list the FileSystemRights? Click or tap here to enter text.
      5. Based on this ACL, Which are the users or groups (IdentityReferences) that are granted “Full Control” of the root folder? Click or tap here to enter text.
      6. Based on this ACL, what file system rights are granted to users or groups who are members of the “BUILTIN/Users” group? Click or tap here to enter text.
4. Create a new file in your Documents subdirectory.  
   (For example, you could name it **ex6.4.txt**, like this:   
    **New-Item $HOME\Documents\ex6.4.txt** )
   1. Get the ACL of your new file:   
      **$sd = Get-ACL ~\Documents\ex6.4.txt** # (or whatever you named your new file)
   2. Use the Access property to view its access control entries:   
      **$sd.Access**
      1. What’s the value of **IdentityReference** in the first ACE?   
         **$sd.Access[0].IdentityReference**   
         Your output: Click or tap here to enter text.
      2. What’s the value of **IdentityReference** in the second ACE?   
         **$sd.Access[1].IdentityReference**   
         Your output: Click or tap here to enter text.
      3. What’s the value of **IdentityReference** in the third ACE?   
         **$sd.Access[2].IdentityReference**   
         Your output: Click or tap here to enter text.
      4. Were there additional ACEs in your ACL? If so, what were their **IdentityReference** values? Click or tap here to enter text.
   3. Lets try to find the SIDs of those identities. Here’s a short script to convert each into an SID:   
         
      **$sids = foreach ($i in $sd.Access.IdentityReference.Value) {  
       $domain,$acct=$i -split '\\'  
       $nta=[System.Security.Principal.NTAccount]::new($domain,$acct)  
       $nta.Translate([System.Security.Principal.SecurityIdentifier])  
      }  
      $sids.Value**   
      1. What SID corresponds to **$sd.Access[0].IdentityReference** ? Click or tap here to enter text.
      2. What SID corresponds to **$sd.Access[1].IdentityReference** ? Click or tap here to enter text.
      3. What SID corresponds to **$sd.Access[2].IdentityReference** ? Click or tap here to enter text.
5. Files aren’t the only items with ACLs. Lets look at the ACL attached to your Active Directory user account:   
   **$MyUserName = '***<your-account-name>***'**   
   **$MyADUser = Get-ADUser $MyUserName**   
   **$MyACL=Get-ACL "AD:$($MyADUser.distinguishedName)"**   
   **$MyACL.Access**
   1. What is the full name of the object type of **$MyACL** ? Click or tap here to enter text.
   2. What is the full name of the type of each ACE in **$MyACL** ? Click or tap here to enter text.
   3. Write a command line that uses the **Get-Member** cmdlet to list all the properties of one of these ACEs. Your command line: Click or tap here to enter text.
   4. Write a command or expression that counts the number of ACEs in **$MyACL**. Your command line: Click or tap here to enter text.
      1. Your command line’s output: Click or tap here to enter text.
6. Group Policy Objects have their permission lists, which the textbook author calls GPO ACLs.
   1. Get the permissions of the GPO named **Student demo**:   
      **Get-GPPermission -Name 'Student demo' -All**
   2. Save the result to a variable:   
      **$GPOPerms = Get-GPPermission -Name 'Student demo' -All**
   3. What is the full name of the type of *each* ACE in **$GPOPerms** ? Click or tap here to enter text. (Hint: if you answered **System.Object[]** *or* **System.Object**, you’re doing it wrong!)
   4. What are the property names of each ACE in $GPOPerms (one line, comma-separated): Click or tap here to enter text.
   5. What GPO permission is associated with “trustees” who are members of the built-in “Authenticated Users” group? Click or tap here to enter text.

# Deliverable

Upload this document with completed answers to I-Learn Canvas.