Advanced Scripting   
Enumeration for Active Directory

Last Updated: 3/2/2024 2:28 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.

The directory service in Active Directory is the open standard called *Lightweight Directory Access Protocol* (LDAP).

# Requirements

Internet connection, and an SSH client.

# Setup

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

# Task 1—Query Active Directory.with a DirectorySearcher object

Let’s enumerate the user accounts of the the CIT361 lab’s Active Directory domain.

## Steps

1. Make sure you are logged in to the lab VM. (The **hostname** command should return **horace**.)
2. Create a DirectorySearcher object:   
   **$searcher = New-Object System.DirectoryServices.DirectorySearcher**   
   Here’s a slightly shorter PowerShell expression that does the exact same thing:   
   **$searcher = [System.DirectoryServices.DirectorySearcher]::new()**
   1. Set the Filter attribute of your DirectorySearcher:   
      $searcher.Filter = '(sAMAccountType=805306368)'   
      With this LDAP filter, the searcher is restricted to just finding Active Directory user accounts.
   2. Now use its **FindAll()** method to enumerate all the user accounts.   
      $searcher.FindAll()
   3. What class of objects are the items returned by FindAll()?   
      ($searcher.FindAll())[0].GetType().Name   
      Your output: Click or tap here to enter text.
      1. What are its methods?   
         ($searcher.FindAll())[0] | Get-Member -MemberType Method   
         Your output (one line, separated by commas): Click or tap here to enter text.
      2. What are its properties?   
         ($searcher.FindAll())[0] | Get-Member -MemberType Property   
         Your output, (one line, separated by commas): Click or tap here to enter text.
   4. Let’s use a Where-Object filter to just return the directory entry of your user account.   
      $MyUserName = 'mer23001' # (replace the highlighted username with yours)   
      **$searcher.FindAll() | Where-Object {$\_.Path -match $MyUserName}**
   5. Look at the Path property of your directory entry:   
      **$searcher.FindAll() | Where-Object {$\_.Path -match $MyUserName}).Path**   
      It should be the LDAP *Uniform Resource Indicator* (URI) that represents the *Distinguished Name* (DN) that uniquely identifies your account in the domain. Example:   
       **LDAP://CN=mer23001,OU=Executives,DC=byui,DC=edu**   
      Copy your DN (the full LDAP URI) here: Click or tap here to enter text.
      1. (Use an Internet search engine or generative AI chatbot to help answer the following.) In the context of LDAP, what does the two-letter code CN mean? Click or tap here to enter text.
      2. What does the two-letter code OU mean? Click or tap here to enter text.
      3. What does the two-letter code DC mean? Click or tap here to enter text.
   6. Look at the Properties property of your directory entry:   
      **$searcher.FindAll() | Where-Object {$\_.Path -match $MyUserName}).Properties**
      1. What is the useraccountcontrol attribute in your directory entry? Click or tap here to enter text.
   7. Instead of filtering using Where-Object, let’s put your username directly in the LDAP filter:   
      **$searcher.Filter =** '**(&**(sAMAccountType=805306368)**(**sAMAccountName=$MyUserName)**)**'   
      **$searcher.FindAll().Properties**
      1. (Note: the ampersand **&** in this LDAP filter syntax is an example of *prefix notation*. In this case, it evaluates a “logical and” operation of the subsequent Boolean conditionals   
          sAMAccountType equals 805306368   
          and  
          sAMAccountName equals <your\_login\_account\_name>)
      2. When would it be advantageous to use PowerShell cmdlets such as **Where-Object** to filter the search results? Your opinion: Click or tap here to enter text.
      3. When would it be advantageous to use LDAP’s filter syntax to filter the searcher’s results? Your opinion: Click or tap here to enter text.
   8. Let’s use an LDAP filter to see if there are any accounts configured with the “Password never expires” option:   
      **$searcher.Filter =** '**(&**(samaccounttype=805306368)**(**useraccountcontrol=66048)**)**'   
      **$searcher.FindAll()**
      1. Did you find any such accounts on the domain? If so, list their CN attributes (on one line, separated by commas): Click or tap here to enter text.
3. Use your Distinguished Name and a DirectorySearcher object to directly access your entry.   
   **$dn =** '<your LDAP URI that you recorded in step 2.5>'   
   **$me = [System.DirectoryServices.DirectorySearcher]$dn**
   1. What are the methods of the object in the variable **$me**? Click or tap here to enter text.
   2. What are the properties of the object in **$me**? Click or tap here to enter text.
   3. What is your account’s **displayName** attribute? Click or tap here to enter text.
   4. When was your account created? Click or tap here to enter text.
4. Use an LDAP URI and a DirectorySearcher object to enumerate the domain’s student accounts.   
   **$dn =** 'LDAP://OU=Students,DC=cit361,DC=com'   
   **$ou = [System.DirectoryServices.DirectorySearcher]$dn**   
   **$ou.Children**

# Task 2 —Query Active Directory.Using Accelerators

.NET on Windows contains convenient *Active Directory Services Interface* (ADSI) “accelerator” classes.

## Steps

1. Convert an LDAP URI into an ADSI object:   
   **$dn =** 'LDAP://OU=Students,DC=cit361,DC=com'   
   **$ou = [adsi]$dn**
   1. Use it to enumerate the student accounts:   
      **$ou.Children**
2. Convert an LDAP “Password never expires” filter into an ADSISearcher object:   
   **$searcher = [adsisearcher]**'**(&**(samaccounttype=805306368)**(**useraccountcontrol=66048)**)**'
   1. Use it:   
      **$searcher.FindAll()**
3. Find the Group Policy Objects (GPOs) linked to the top level of the Active Directory domain:   
   **$dn =** 'LDAP://DC=cit361,DC=com'   
   **$domain = [adsi]$dn**   
   **$domain.gplink**   
   Your output: Click or tap here to enter text.
4. Use an LDAP filter to enumerate GPOs:   
   **$dn =** 'LDAP://DC=cit361,DC=com'   
   **$gpoFilter =** '(objectCategory=groupPolicyContainer)'  **$searcher = [adsisearcher]$gpoFilter**   
   **$searcher.SearchRoot = [adsi]$dn**   
   **$searcher.FindAll() | Foreach-Object {$\_.Properties.displayName}**   
   Your output (one line, comma separated): Click or tap here to enter text.
5. Use an LDAP filter to enumerate the groups in the domain.   
   **$filt = '(&(objectClass=group)(cn=\*))'**   
   **([adsisearcher]$filt).FindAll().Properties.name**   
   Your output (one line, comma separated): Click or tap here to enter text.

# Task 3—Query Active Directory.with PowerShell Modules and Cmdlets

The PowerShell modules ActiveDirectory and GroupPolicy are not installed by default, but come with the Remote Server Administration Tools (RSAT) Windows Feature. These modules provide powerful cmdlets for working with Active Directory.

## Steps

1. List the commands provided by the **ActiveDirectory** PowerShell module.   
   **Get-Command -Module ActiveDirectory**   
   How many of its commands start with the verb **Get**? Click or tap here to enter text.
2. Use the Get-ADUser cmdlet to enumerate domain users:   
   **Get-ADUser -Filter \* | Format-Table**
   1. Write a PowerShell command line to output the Name property of every *disabled* account. Your command: Click or tap here to enter text. Your command’s output: Click or tap here to enter text.
3. List the commands provided by the **GroupPolicy** PowerShell module. How many of its commands start with the verb Get? Click or tap here to enter text.
   1. List the DisplayName property of all the GPOs in the domain:   
      **(Get-GPO -All).DisplayName**   
      Your result (on one line, separated by commas): Click or tap here to enter text.
4. Enumerate the “built-in” groups with elevated privileges in the domain:   
   $pgs = Get-ADGroup -SearchBase 'CN=BuiltIn,DC=cit361,DC=com' -Filter \*   
   **$pgs.Name**   
   Your result (one line, comma-separated): Click or tap here to enter text.
   1. Show a table of the built-in groups’ names, scopes, categories, and security identifiers:   
      **$pgs | ft Name, GroupScope, GroupCategory, SID**

# Deliverable

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