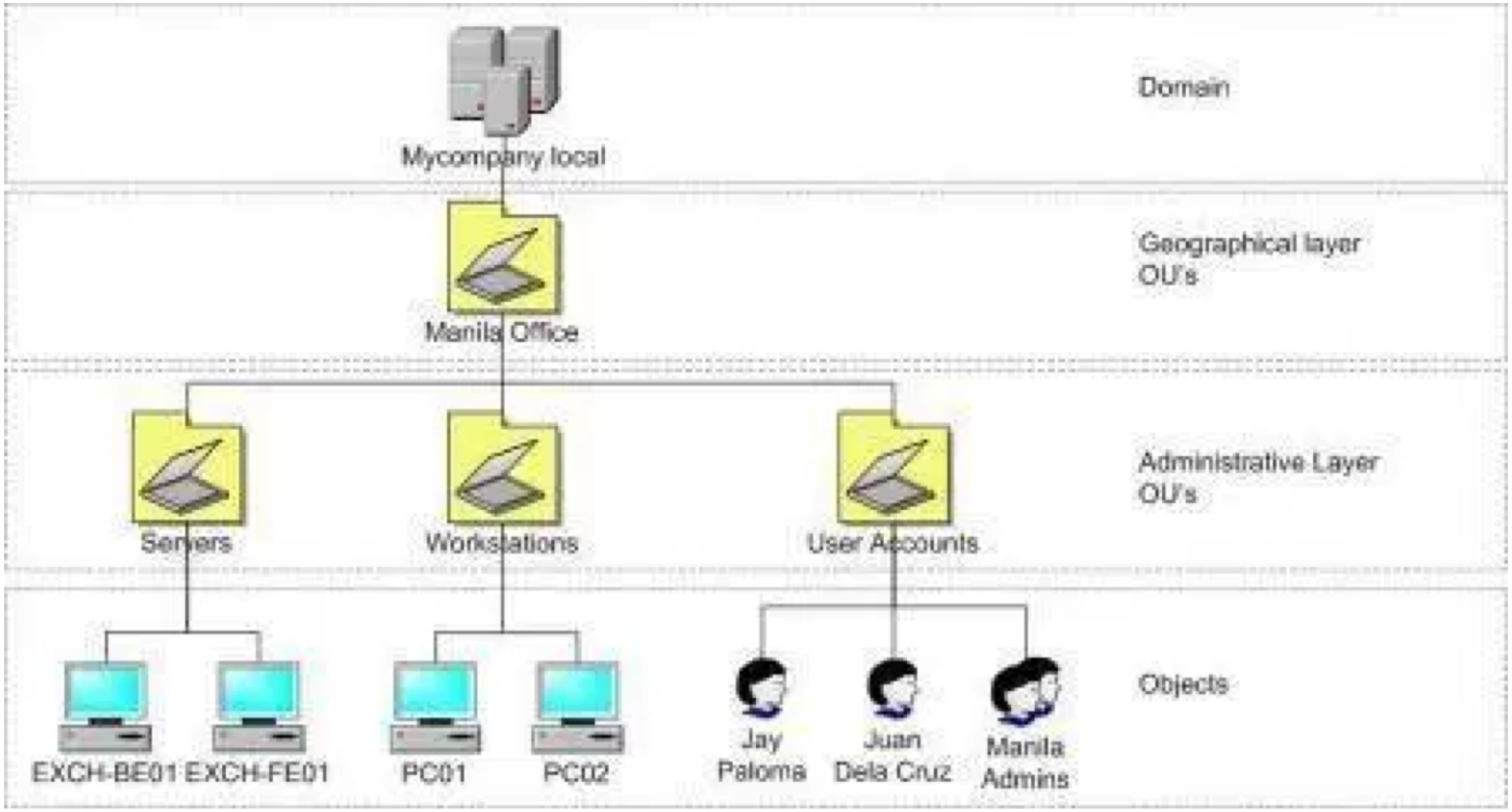
Active Directory Basics

From Containers to LDAP Interfaces

Active Directory

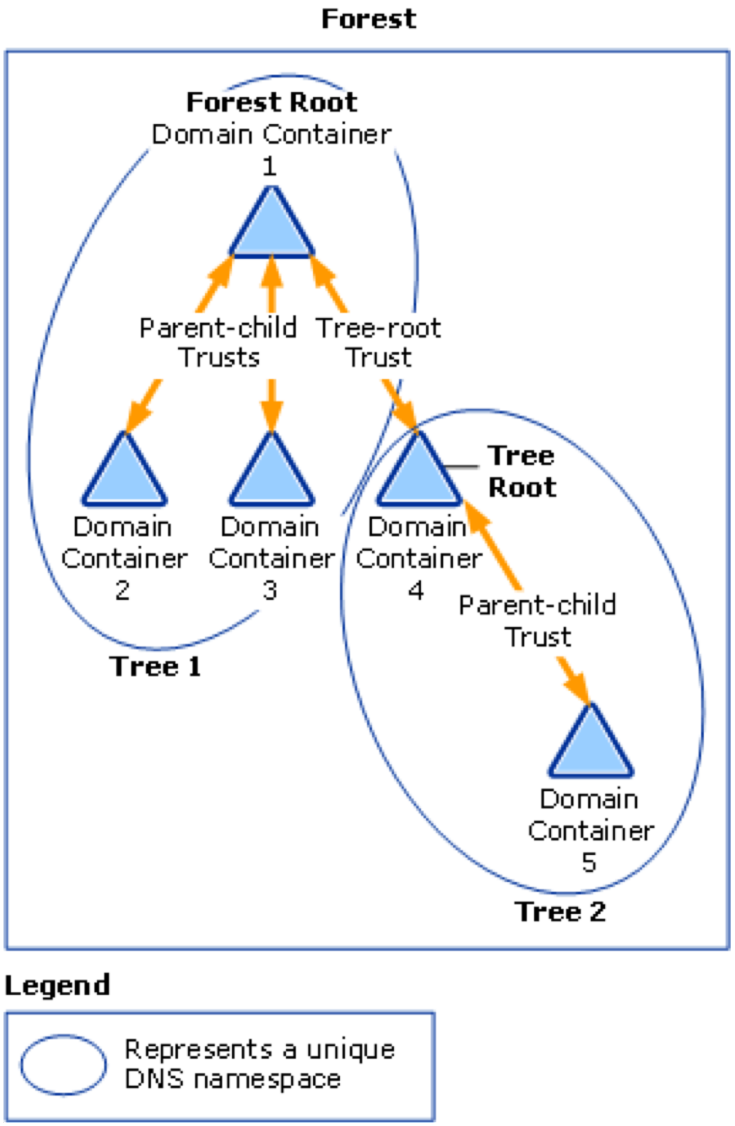
- At its core, Active Directory (AD) is database that
  - Represents the resources (users/computers/shares/etc.) for an organization
  - Contains access rules that govern the control relationships between these resources
  - Provides security policies, centralized management, and other rich features
- Red teams and real bad guys have been abusing AD for years, but not much offensive AD information has existed publicly (until fairly recently)
  - Great reference: https://adsecurity.org/

Active Directory



Active Directory Forests/Domains

- **Domains** are containers within the scope of a forest and define a scope/unit of policy
  - PowerView: **Get-Domain**
  - Can have GPOs linked
- A **forest** is a single instance of Active Directory
  - Essentially a collection of domain containers that trust one another
  - PowerView: **Get-Forest**



Active Directory Containers

- Organizational units (OUs) are logical groupings of users, computers, and other resources
  - Can have GPOs linked
  - PowerView: **Get-DomainOU *name*** [-GPLink GUID]
- Sites and subnets represent the physical network topology
  - A computer automatically joins a subnet based on its dhcp lease
  - Subnets (**Get-DomainSubnet**) are linked to specific sites (**Get-DomainSite**)
  - Sites can have GPOs linked as well [**Get-DomainSite -GPLink GUID**]
- Groups
  - Collections of users/other groups (**Get-DomainGroup**)
  - Can function as a security principal

Active Directory Objects

- The physical entities that make up a network
- Users
  - A security principal that is allowed to authenticate to machines/resources in the domain
  - PowerView: **Get-DomainUser**
- Computers
  - A special type of user account
  - PowerView: **Get-DomainComputer**
- GPOs
  - A collection of policies applied to a domain/site/OU object
  - PowerView: **Get-DomainGPO**

Active Directory Administrators

BUILTIN\Administrators	Local admin access on a domain controller.
Domain Admins	Administrative access to all resources in the associated domain
Enterprise Admins	Exists only in the forest root. Implicitly added to “Domain Admins” of every child domain.
Schema Admins	Can modify the domain/forest schema. Normally not useful from a red team perspective.
Server Operators	Can administer domain servers.
Account Operators	Can manage any user not in a privileged group.

Interfacing With Active Directory

- General approaches are:
  - Built in **net** commands which wrap various Win32 API calls
  - Manual implementation of various Win32 API calls
  - LDAP interfaces (like dsquery/adfind)
  - PowerShell!
- With PowerShell, the main options are:
  - The official RSAT-AD-PowerShell Active Directory cmdlets
  - Interacting with various .NET classes that wrap various RPC interfaces
  - Using the .NET **DirectorySearcher** or **DirectoryEntry** objects to interface with LDAP
- PowerView uses a combination of all of the above

PowerView

- PowerView is a PowerShell version 2.0-compliant network and domain situational-awareness tool
  - Think of it like a recoded version of the official Active Directory cmdlets that works on V2, with some bonus features
  - Rewritten from the ground up in late 2016
- Built to automate large components of our tradecraft used to facilitate red team engagement
- Uses PSReflect for its Win32 function calls (nothing touches disk)
  - Also heavily wraps DirectorySearcher objects under the hood
- All PowerView functions have proper XML-based help
  - Remember **Get-Help**!

-Identity

- Most LDAP (Verb-Domain*) cmdlets also have an **-Identity** parameter instead of -UserName/-GroupName/etc.
- This parameter accepts:
  - samAccountName
  - distinguishedName
  - objectGUID
  - objectSID
  - dnshostname (for computers)
- These can be mixed!
  - **'GUID', 'harmj0y', 'OU=...' | Get-DomainObject**

-Credential

- ALL PowerView functions accept a **-Credential** specification
  - BUT the behavior varies under the hood (WMI vs Win32 API vs LDAP)
- LDAP functions (Verb-Domain*) modules use alternate plaintext creds with DirectoryServices.DirectoryEntry/DirectorySearcher
  - **$SecPassword = ConvertTo-SecureString 'BurgerBurgerBurger!' -AsPlainText -Force**
  - **$Cred = New-Object System.Management.Automation.PSCredential('TEST LAB\dfm.a', $SecPassword)**
  - **Get-DomainUser harmj0y -Credential $Cred**

Other Common Parameters

- **-LDAPFilter '(property=Value)'**
  - Allows you to specify additional optional LDAP filters
- **-Properties property1,property2**
  - Returns *only* the properties specified
  - "Optimizes to the left" in what's returned from the server!
- **-FindOne()**
  - Only return one result (good for object property inspection)
- **-SearchBase "ldap://OU=blah,DC=..."**
  - Searches a particular OU/LDAP bind path
- **-Server computer.domain.com**
  - Specifies a DC to bind to for the query

[DirectoryServices.ActiveDirectory]

- The **[DirectoryServices.ActiveDirectory]** namespace has a number of useful interfaces for various Active Directory taskings

- Ex: to retrieve the current domain object:
  - **[System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain()**

- Ex: to retrieve a foreign domain object:
  - **$Context = New-Object System.DirectoryServices.ActiveDirectory.DirectoryContext('Domain', $Domain)**
  - **[System.DirectoryServices.ActiveDirectory.Domain]::GetDomain($Context)**

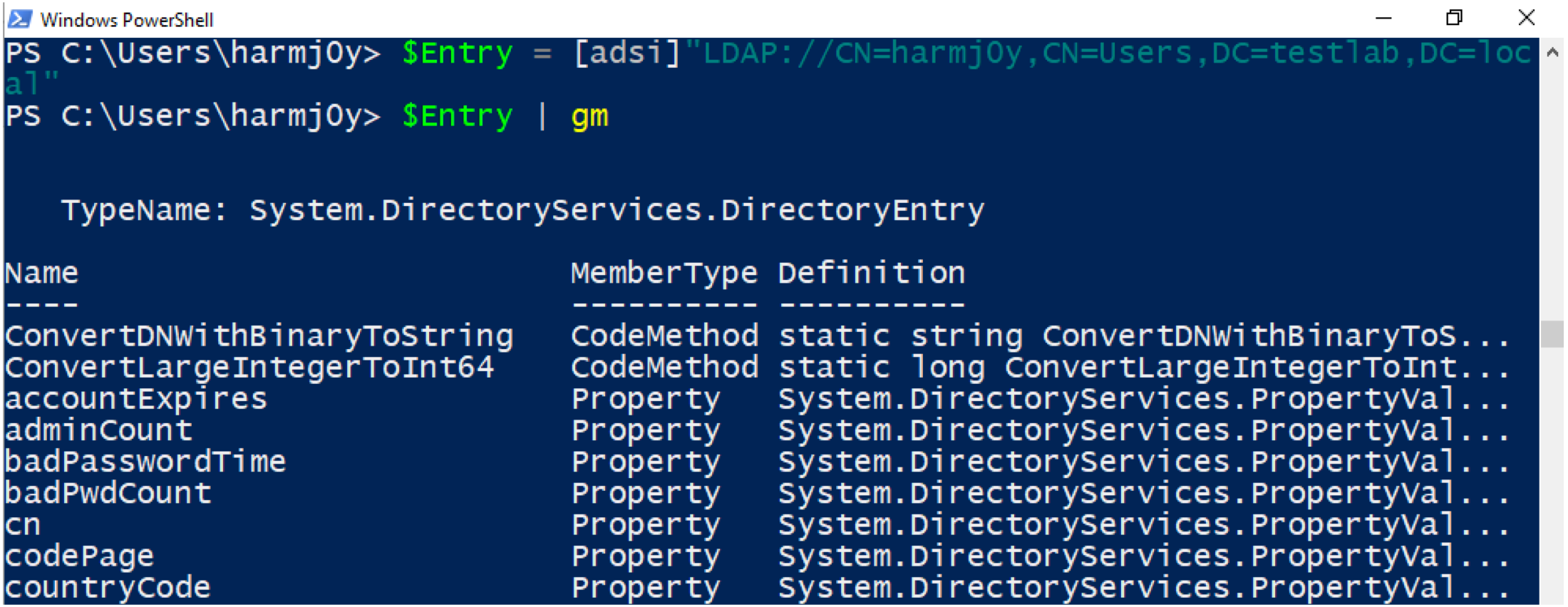
DirectoryEntry

- The **[System.DirectoryServices.DirectoryEntry]** represents a encapsulates a node or object in Active Directory
  - **$Entry = New-Object DirectoryServices.DirectoryEntry('LDAP://CN=harmj0y,CN=Users,DC=testlab,DC=local')**
  - **$Entry.objectclass**

- The **[adsi]** accelerator is an easy **DirectoryEntry** alias:
  - **([adsi]"LDAP://CN=harmj0y,CN=Users,DC=testlab,DC=local").objectclass**

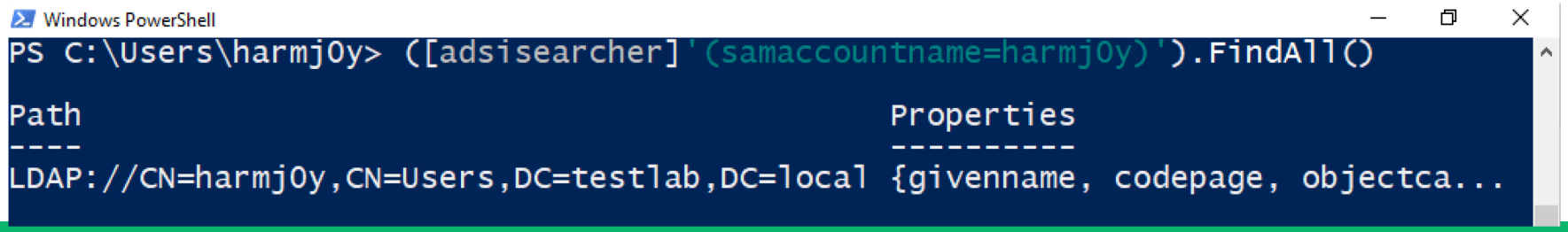
- Note: Be sure to capitalize LDAP:// !

[adsi]


Windows PowerShell
PS C:\Users\harmj0y> $Entry = [adsi]"LDAP://CN=harmj0y,CN=Users,DC=testlab,DC=local"
PS C:\Users\harmj0y> $Entry | gm
TypeName: System.DirectoryServices.DirectoryEntry
Name MemberType Definition
---- ---------- ----------
ConvertDNWithBinaryToString CodeMethod static string ConvertDNWithBinaryToS...
ConvertLargeIntegerToInt64 CodeMethod static long ConvertLargeIntegerToInt...
accountExpires Property System.DirectoryServices.PropertyVal...
adminCount Property System.DirectoryServices.PropertyVal...
badPasswordTime Property System.DirectoryServices.PropertyVal...
badPwdCount Property System.DirectoryServices.PropertyVal...
cn Property System.DirectoryServices.PropertyVal...
codePage Property System.DirectoryServices.PropertyVal...
countryCode Property System.DirectoryServices.PropertyVal...

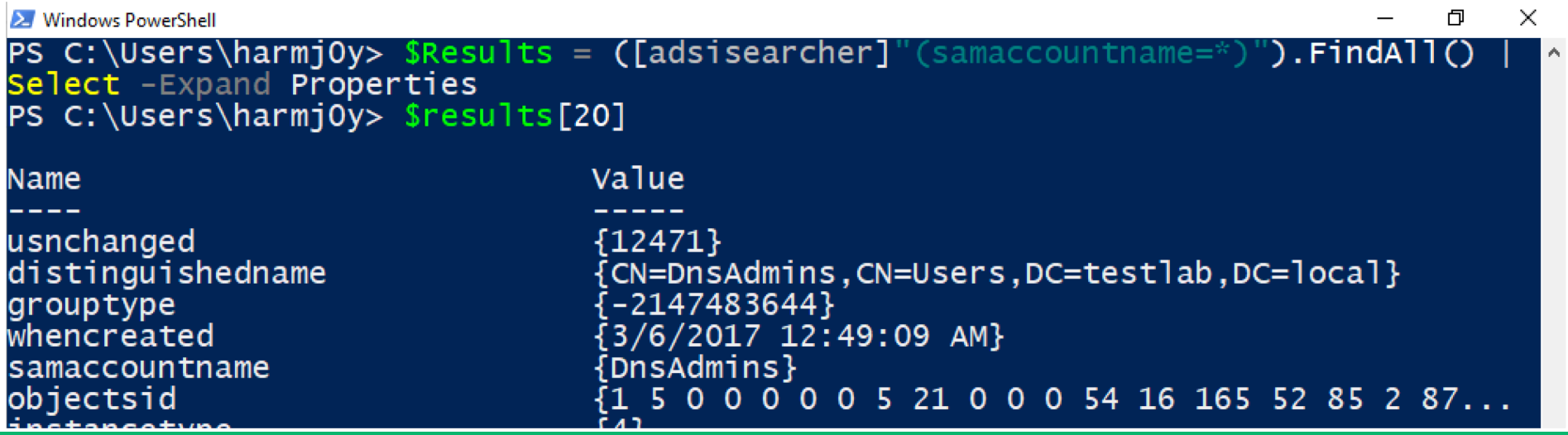
DirectorySearcher

- The **[System.DirectoryServices.DirectorySearcher]** class allows for searching against an Active Directory instance
  - **$Searcher = New-Object DirectoryServices.DirectorySearcher('(samaccountname=harmj0y)')**
  - **$Searcher.FindAll()** : finds ALL results
  - **$Searcher.FindOne()** : finds ONE result
- The **[adsisearcher]** accelerator is an easy **DirectorySearcher** alias:
  - **([adsisearcher]'(samaccountname=harmj0y)').FindAll()**



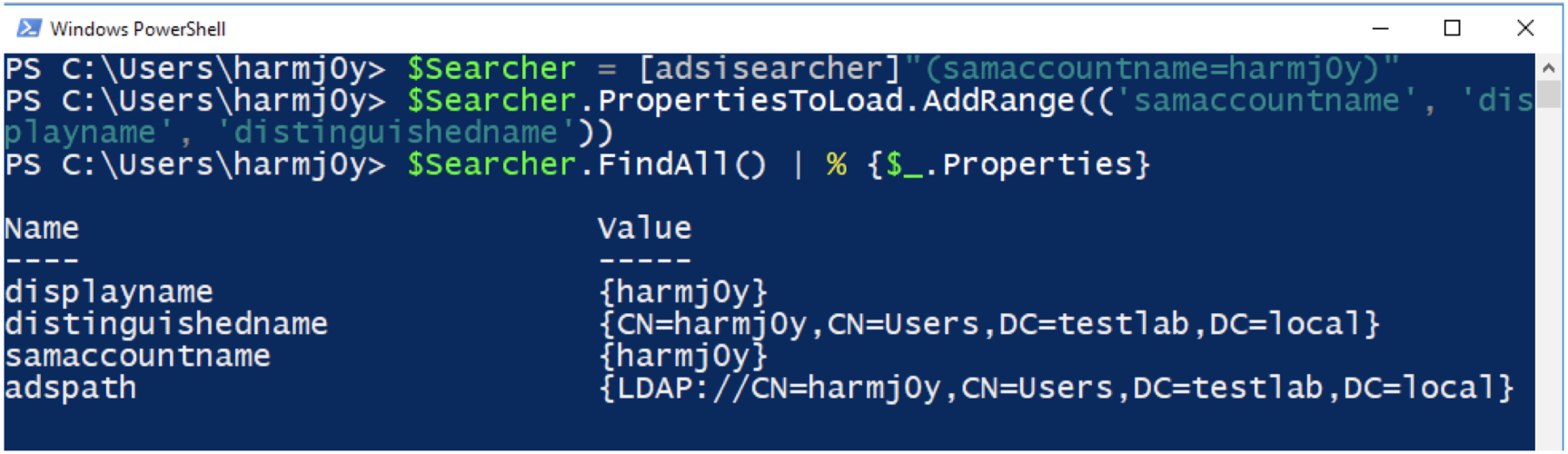
Processing **DirectorySearcher** Results

- The results from **DirectorySearcher** will be one or more objects with **Path** (AdsPath) and **Properties**
  - Some of these properties will be COM objects >_<
  - PowerView unwraps and converts most major object properties for you



Sidenote: Property Optimization

- A **DirectorySearcher** object has a **PropertiesToLoad** property that implements the **.Add()** and .**AddRange()** methods
- This instructs the LDAP server/domain controller to return *only* those specific properties, “optimizing to the left”



LDAP ADsPath

- The Microsoft LDAP provider ADsPath requires the following format:
  - **LDAP://HostName[:PortNumber][/DistinguishedName]**
- This path either points to a specific object to bind to:
  - Ex: **LDAP://CN=harmj0y,CN=Users,DC=testlab,DC=local**
- Or a container to search through (like an OU):
  - Ex: **LDAP://OU=EastUS,DC=testlab,DC=local**
- HostName is used to bind to a specific domain controllers:
  - Ex: **LDAP://primary.testlab.local/CN=harmj0y,CN=Users,DC=testlab,DC=local**

LDAP Filters

- An LDAP filter has to take the form of:
  - **(<AD Attribute><comparison operator><value>)**
- Comparison operators: **=, >=, <=**
  - Wildcards are accepted for non-binary values!
  - Ex: users with “pass” in the description field:
    (&((samAccountType=805306368)(description=*pass*)))
- Logical operators: **!, &, |**
- Combining filters:
  - **(&(|(|(samAccountName=testuser)(name=testuser))))**
- To search for objects with a specific property set:
  - **(property=*)**

Binary LDAP Filters

- To build filters for binary object fields, like **userAccountControl**, you need to use a bitwise filter
  - Format: <attributename:ruleOID:=value>
  - **1.2.840.113556.1.4.803** : true if ALL bits match (AND)
  - **1.2.840.113556.1.4.804** : true if ANY bits match (OR)
- Example: find all users with “Password Never Expires”
  - **(&(samAccountType=805306368)(userAccountControl:1.2.840.113556.1.4.803:=65536))**
- Example: find all groups with a ‘Domain Local’ scope
  - **(groupType:1.2.840.113556.1.4.803:=4)**

objectCategory vs objectClass

objectCategory	objectClass	Result
person	user	user objects
person		user and contact objects
person	contact	contact objects
	user	user and computer objects
computer		computer objects
user		user and contact objects
	contact	contact objects
	computer	computer objects
	person	user, computer, and contact objects
contact		user and contact objects
group		group objects
	group	group objects
person	organizationalPerson	user and contact objects
	organizationalPerson	user, computer, and contact objects
organizationalPerson		user and contact objects

https://social.technet.microsoft.com/wiki/contents/articles/5392.active-directory-ldap-syntax-filters.aspx

The Global Catalog

- The global catalog (GC) is a partial copy of all objects in an Active Directory forest
  - meaning that some object properties (but not all) are contained within it
- This data is replicated among all domain controllers marked as global catalogs for the forest
- To find all global catalogs in the forest:
  - **[System.DirectoryServices.ActiveDirectory.Forest]::GetCurrentForest().FindAllGlobalCatalogs()**
- In practice, you should just be able to use **GC://domainname.com** as the search base, as there has to be at least one GC per domain

The Global Catalog : LDAP Searching

- To use a global catalog with PowerView:
  - **-SearchBase “GC://domain.local”**
- To use a global catalog with manual LDAP searching, you first need to bind to the GC with **[adsi]** and then bind to the result with **[adsisearcher]**:
  - **$Searcher = [ADSISearcher][ADSI]”GC://covertius.local”**
  - **$Searcher.Filter = ‘(samaccountname=harmj0y)’**
  - **$Searcher.FindAll()**
- **Note:** global catalog searches use a different port (3268) than regular LDAP searches (389)

This would be a good time to attempt Lab: LDAP Searching