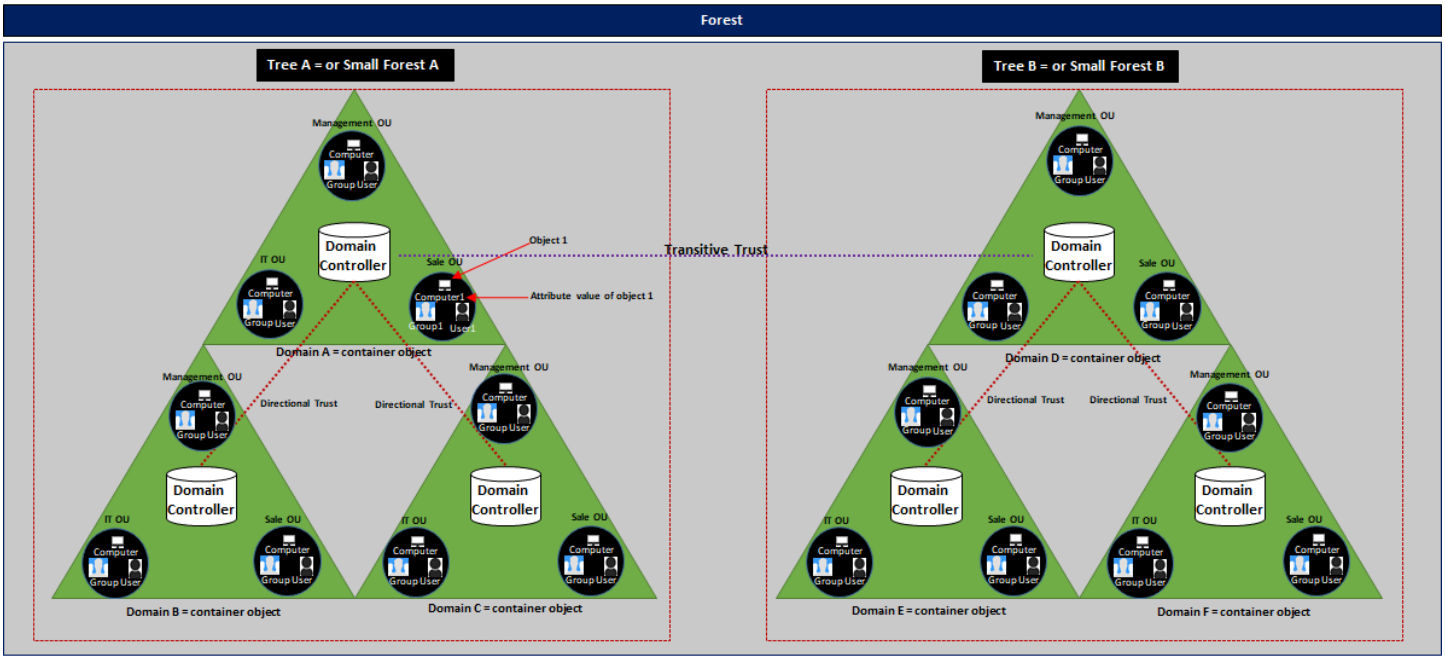

1. Forest

 # get details about the current forest.
Get-NetForest

 # get details about another forest.
Get-NetForest -Forest dampy.com

 # get all the domains in the current forest.
Get-NetForestDomain

 # get all global catalogs for the current forest.
Get-NetForestCatalog

 # determine which domain controller holds the PDC emulator FSMO role in the forest root domain
>> Get-ADForest | Select-Object -ExpandProperty RootDomain | Get-ADDomain | Select-Object -Property PDCEmulator

1.1. Domain

 # Get domain information such as what forest is it in, all of the domain controllers, any child domain and the domain mode, which again tells us what kind of security is available.
>> Get-NetDomain

 # get the same results for another domain, use the above command
Get-NetDomain -domain “Domain Name”

 # get the domain SID (Security IDentifier is a unique ID number that a computer or domain controller uses to identify you).
Get-DomainSID

 # get the policy of the current domain
(Get-DomainPolicy).”system access”

 # Use this command to get information about the current domain controller (DC)
Get-NetDomainController

1.1.1. OU (ACL)

 # get all the OUs (Organization Units) in the current domain.
Get-NetOU

 # Identify administrator credentials in SYSVOL
>> We can use the PowerSploit's get-GPPPassword

 # To understand/identify what delegation has been configured on the OUs in the domain
>> Invoke-ACLScanner –ResolveGUIDs –ADSpath 'OU=X,OU=Y,DC=Z,DC=W' | Where {$_.ActiveDirectoryRights -eq 'GenericAll'}

 # enumerate the ACLs for the users group.
Get-ObjectAcl -SamAccountName "users" -ResolveGUIDs

 # see if there is any user has a modification rights to a GPO.
Get-NetGPO | %{Get-ObjectAcl -ResolveGUIDs -Name $_.Name}

 # check if the user "Sarah" has the permission (Reset Password).
Get-ObjectAcl -SamAccountName labuser -ResolveGUIDs -RightsFilter "ResetPassword"

 # Using PowerView we can also get the ACLs for all OUs where someone is allowed to read the LAPS password attribute, as follows.
>> Get-NetOU -FullData | Get-ObjectAcl -ResolveGUIDs | Where-Object {($_.ObjectType -like 'ms-Mcs-AdmPwd') -and ($_.ActiveDirectoryRights -match 'ReadProperty')} | ForEach-Object {$_ | Add-Member NoteProperty 'IdentitySID'

1.1.1.1. GROUP

 # Use this command to get all the groups in the current domain.
Get-NetGroup

 # get all the groups that contain the word "admin" in the group name.
Get-NetGroup *admin*

 # Use this command to get the group membership of the user "Khalid"
Get-NetGroup –UserName "khalid"

 # Identifying Computers Having Admin Rights
>> Get-NetGroup "*admins*" | Get-NetGroupMember –Recurse |?{$_.MemberName –Like '*$'}

##

 # get the members of the group "Domain Admin"
Get-NetGroupMember -GroupName "Domain Admins"

 # request the members of a particular group
>> Get-NetGroupMember 'Domain Admins' –Recurse

 # identify administrator accounts indirectly
>> Get-NetGroupMember –GroupName "Denied RODC Password Replication Group" –Recurse

##

 # get all the local administrators on a machine. (Note that it needs administrative rights).
Get-NetLocalGroup –ComputerName Client-02

 # Retrieve more information using Get-NetLocalGroup
>> Get-NetLocalGroup -ComputerName computer_name

 # Get local group membership with the NetLocalGroupGetMembers API call.
>> Get-NetLocalGroup -ComputerName computer_name –API

 # The following retrieves the names of the local groups themselves.
>> Get-NetLocalGroup -ComputerName computer_name –ListGroups

 # determine the actual users having RDP rights
>> Get-NetLocalGroup -ComputerName computer_name -GroupName "Remote Desktop Users" -Recurse

##

 # get actively logged users on a computer (Note that it needs administrative rights)
Get-NetLoggedon –ComputerName "Client-02"

 # get the last logged user on a computer (Note that it needs administrative rights)
Get-LastLoggedOn –ComputerName Client-02

##

 # identify groups and users have local administrative access on domain controllers
>> Get-NetDomainController | Get-NetLocalGroup –Recurse

 # find shares on the hosts in the current domain
Invoke-ShareFinder

 # Find groups in a remote domain that include users not in the target domain.
>> Find-ForeignGroup -Domain els.local

 # Retrieve the members of the 'Administrators' local group on a specific remote machine:
>> ([ADSI]'WinNT://computer_name/Administrators').psbase.Invoke('Members') | %{$_.GetType().InvokeMember('Name', 'GetProperty', $null, $_, $null)}

1.1.1.2. POLICY (Group, User, Computers)

 # discover all the group policies inside a domain
>> Get-NetGPO | select displayname,name,whenchanged

 # get a list of the GPO in the computer (Client-02).
Get-NetGPO -ComputerName client-02.fanzy.com

 # find users who have local admin rights over the machine Client-02 through GPO.
Find-GPOComputerAdmin –Computername client-02.fanzy.com

 # find all computers that ”Aziz” has local administrator rights in the current domain through the applied GPO.
Find-GPOLocation -UserName Aziz

 # Identify all computers that the specified user has local RDP access rights to in the domain
>> Find-GPOLocation -UserName username -LocalGroup RDP

 # identify which AD groups have admin rights to which computers
>> Get-NetGPOGroup
>> Get-NetGroupMember -GroupName "Local Admin"

 # Request for all the members of “Domain Admins”
>> Get-NetGroupMember -GroupName 'Domain Admins' -FullData | %{ $a=$_.displayname.split(' ')[0..1] -join ' '; Get-NetUser -Filter "(displayname=*$a*)" } | Select-Object -Property displayname,samaccountname

1.1.1.3. Users

 # list all the users in the current domain with information about each user
Get-NetUser

 # Identify potentially privileged accounts using the AdminCount property only without group enumeration
>> Get-NetUser –AdminCount | select name,whencreated,pwdlastset,lastlogon

 # see the last password set of each user in the current domain.
Get-UserProperty –Properties pwdlastset

 # Search for the word “pass” in the field “description” for each user in the domain.
Find-UserField -SearchField Description –SearchTerm “pass”

 # Get the list of effective users who can access a target system
>> Get-NetLocalGroup -ComputerName computer_name –Recurse

 # find all machines on the current domain where the current user has local admin access.
Find-LocalAdminAccess

 # find all machines on the current domain where the current user has local admin access.
Find-LocalAdminAccess

 # Find local admins on all machines of the domain (needs administrator privs on non-dc machines).
Invoke-EnumerateLocalAdmin

 # find computers where a domain has logged in
Invoke-UserHunter

 # find computers where a specific user has sessions
Invoke-UserHunter -UserName “Aziz"

 # find computers where a domain admin is logged in and current user has access.
Invoke-UserHunter –CheckAccess

1.1.1.4. Computers

 # Use this command to list all the computers in the current domain.
Get-NetComputer

 # list all the operating systems "Windows 7 Ultimate".
Get-NetComputer –OperatingSystem "Windows 7 Ultimate"

 # get all the pingable computers (live hosts) in the current domain.
Get-NetComputer -Ping

 # identify machines inside the domain
>> get-adcomputer –filter * -Properties ipv4address | where {$_.IPV4address} | select name,ipv4addressor
>> get-adcomputer -filter {ipv4address -eq 'IP'} -Properties Lastlogondate,passwordlastset,ipv4address

 # queries the domain for all the computer objects and then for each computer
>> Invoke-UserHunter -Stealth –ShowAll

 # SPN scanning:
>> Get-ADComputer -filter {ServicePrincipalName -Like "*SPN*" } –Properties
OperatingSystem,OperatingSystemVersion,OperatingSystemServicePack,PasswordLastSet,LastLogonDate,ServicePrincipalName,TrustedForDelegation,TrustedtoAuthForDelegation

2. Trust

 # get a list of all domain trusts for the current domain to map the domain trust.
Get-NetDomainTrust

 # map the trusts of a forest.
Get-NetForestTrust

 # Enumerate all current domain trusts.
>> Get-NetUser -Domain associated_domain

 # Find admin groups across a trust.
>> Get-NetGroup *admin* -Domain associated_domain

 # Map all reachable domain trusts.
>> Invoke-MapDomainTrust

 # Map all reachable domain trusts through LDAP queries, reflected through the current primary domain controller.
>> Invoke-MapDomainTrust -LDAP

 # Export domain trust mappings for visualization
>> Invoke-MapDomainTrust | Export-Csv -NoTypeInformation trusts.csv

 # Find users in the current domain that reside in groups across a trust.
>> Find-ForeignUser