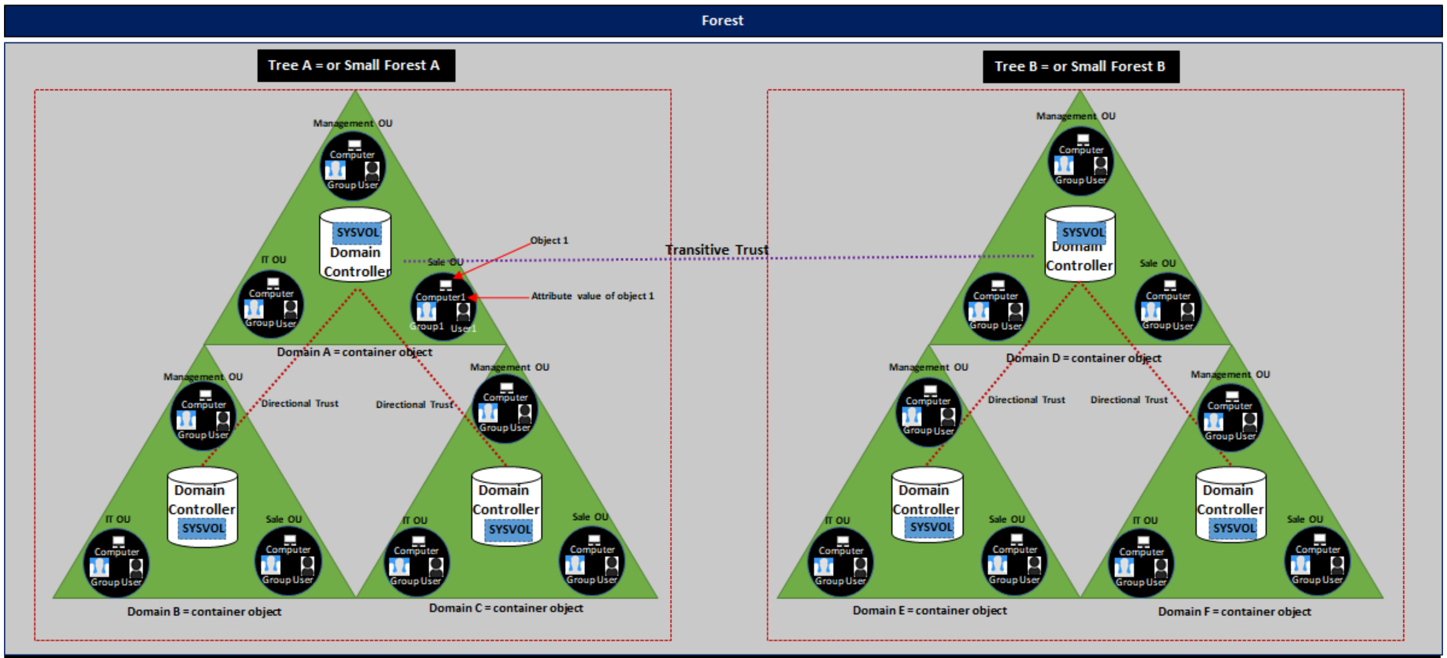

Forest
Tree A = or Small Forest A
Tree B = or Small Forest B
Management OU
Computer
Group User
SYSVOL
Domain Controller
IT OU
Sale OU
Object 1
Computer1
Group1
User1
Attribute value of object 1
Transitive Trust
Domain A = container object
Directional Trust
Domain B = container object
Domain C = container object
Domain D = container object
Domain E = container object
Domain F = container object

The Ntds. dit file is **a database that stores Active Directory data**,
including information about user objects, groups, and group membership.
It includes the password hashes for all users in the domain. ... The extraction and cracking of these passwords can be performed offline, so they will be undetectable.

The term SYSVOL refers to **a set of files and folders that reside on the local hard disk of each domain controller in a domain** and that are replicated by the File Replication service (FRS).

Network clients access the contents of the SYSVOL tree by using the following shared folders: NETLOGON.

The **Security Account Manager** (SAM) is a database file in Windows XP, Windows Vista, Windows 7, 8.1 and 10 that stores users' passwords. It can be used to authenticate local and remote users. ... SAM uses cryptographic measures to prevent unauthenticated users accessing the system.

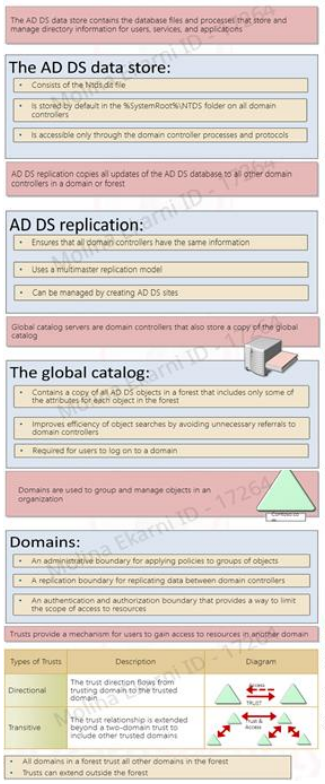
LSASS is the Local Security Authority Subsystem. It's ultimately responsible for making the access granted / access denied decision when you attempt to access resources in a Windows NT-derived operating system.

Each time you try to access any resource, a bit of code deep down in LSASS actually says "Yeah, go ahead" or "Woah! No way!"

Lsass.exe is an executable Windows file and stands for **Local Security Authority Subsystem Service**. As you can see the name of this process contains two words, "**Security Authority**," this process controls the tasks of Windows 10 concerned with the security policy. For example, user's verification in the server, user's authentication during login, password changes, etc.

When you enter the wrong password during login into your account on Windows PC, it is the Lsass.exe process that displays the message "**Password does not match**." If the lsass.exe process fails, the user immediately loses access to all his accounts on the Windows machine.

The **Local Security Authority (LSA)** is responsible for managing interactive logons to the system. ...
The SAM compares the user's credentials with the account information in the SAM database to determine whether the user is authorized to access the system.



The AD DS data store contains the database files and processes that store and manage directory information for users, services, and applications

The AD DS data store:

- Consists of the Ntds.dit file
- Is stored by default in the %SystemRoot%\NTDS folder on all domain controllers
- Is accessible only through the domain controller processes and protocols

AD DS replication copies all updates of the AD DS database to all other domain controllers in a domain or forest

AD DS replication:

- Ensures that all domain controllers have the same information
- Uses a multimaster replication model
- Can be managed by creating AD DS sites

Global catalog servers are domain controllers that also store a copy of the global catalog

The global catalog:

- Contains a copy of all AD DS objects in a forest that includes only some of the attributes for each object in the forest
- Improves efficiency of object searches by avoiding unnecessary referrals to domain controllers
- Required for users to log on to a domain

Domains are used to group and manage objects in an organization

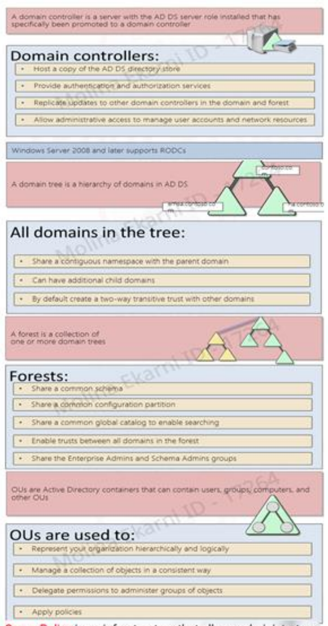
Domains:

- An administrative boundary for applying policies to groups of objects
- A replication boundary for replicating data between domain controllers
- An authentication and authorization boundary that provides a way to limit the scope of access to resources

Trusts provide a mechanism for users to gain access to resources in another domain

Types of Trusts	Description	Diagram
Directional	The trust direction flows from trusting domain to the trusted domain	
Transitive	The trust relationship is extended beyond a two-domain trust to include other trusted domains	

- All domains in a forest trust all other domains in the forest
- Trusts can extend outside the forest



A domain controller is a server with the AD DS server role installed that has specifically been promoted to a domain controller

Domain controllers:

- Host a copy of the AD DS directory store
- Provide authentication and authorization services
- Replicate updates to other domain controllers in the domain and forest
- Allow administrative access to manage user accounts and network resources

Windows Server 2008 and later supports RODCs

A domain tree is a hierarchy of domains in AD DS

All domains in the tree:

- Share a contiguous namespace with the parent domain
- Can have additional child domains
- By default create a two-way transitive trust with other domains

A forest is a collection of one or more domain trees

Forests:

- Share a common schema
- Share a common configuration partition
- Share a common global catalog to enable searching
- Enable trusts between all domains in the forest
- Share the Enterprise Admins and Schema Admins groups

OUs are Active Directory containers that can contain users, groups, computers, and other OUs

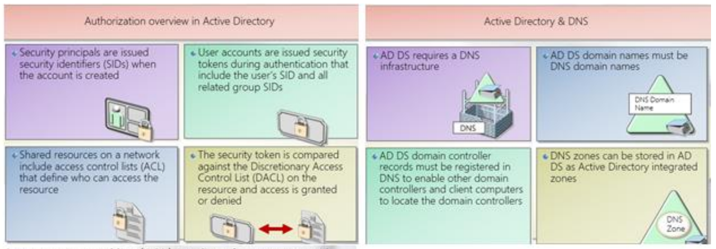
OUs are used to:

- Represent your organization hierarchically and logically
- Manage a collection of objects in a consistent way
- Delegate permissions to administer groups of objects
- Apply policies

Group Policy is an infrastructure that allows administrators to implement specific configurations for users and computers.

Group Policy settings are contained in Group Policy objects (GPOs), which are linked to the following Active Directory service containers:

- Sites,
- Domains, or
- organizational units (OUs).



Authorization overview in Active Directory

- Security principals are issued security identifiers (SIDs) when the account is created
- User accounts are issued security tokens during authentication that include the user's SID and all related group SIDs
- Shared resources on a network include access control lists (ACL) that define who can access the resource
- The security token is compared against the Discretionary Access Control List (DACL) on the resource and access is granted or denied

Active Directory & DNS

- AD DS requires a DNS infrastructure
- AD DS domain names must be DNS domain names
- AD DS domain controller records must be registered in DNS to enable other domain controllers and client computers to locate the domain controllers
- DNS zones can be stored in AD DS as Active Directory integrated zones

An access control list (ACL) is a list of access control entries (ACE). Each ACE in an ACL identifies a security principal and the access rights allowed, denied or audited for that principal.

The security descriptor for a securable object can contain two types of ACLs: a DACL and a SACL.
A discretionary access control list (DACL) identifies the security principals that are allowed or denied access to an object.

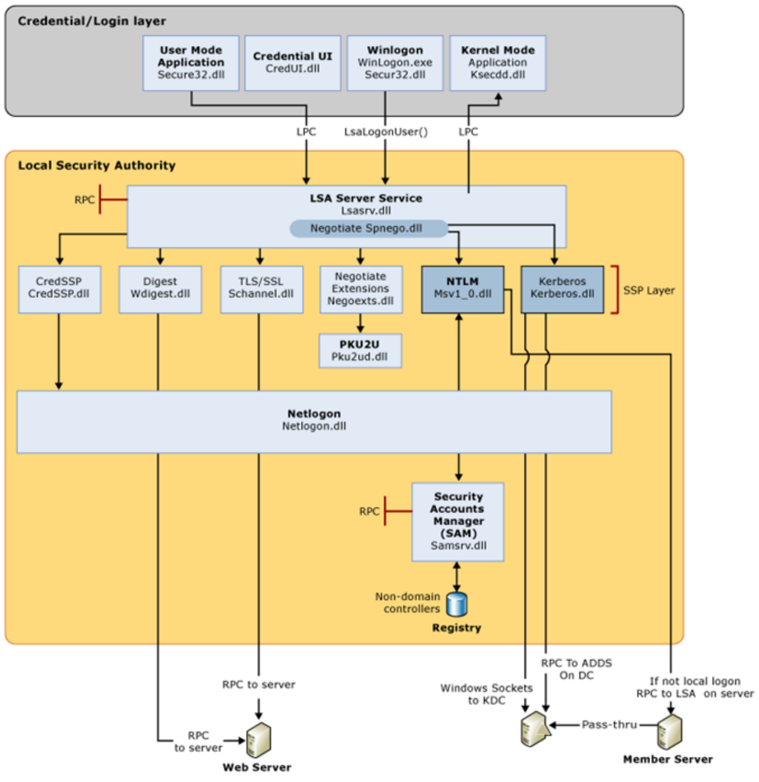
When a person or process tries to access an object, the system checks the ACEs in the object's DACL to determine whether to grant access to it.
A system access control list (SACL) enables administrators to log attempts to access a secured object.

Each ACE specifies the types of access attempts by a specified principal that cause the system to generate a record in the security event log.

An ACE in a SACL can generate audit records when an access attempt fails, when it succeeds, or both.

The following diagram shows the components that are required and the paths that credentials take through the system to authenticate the user or process for a successful logon

At the backend, then, LSA must determine "who to ask" to determine if the credentials are indeed valid. That's where the "negotiate" block comes in - which will do one of two things:
•If the system is configured as a workstation, use the NTLM scheme as configured on the box and talk to the SAM database.
•If the system is configured as part of a domain, talk to the server to see what it supports, and give it what it needs.



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
```