Securing Active Directory: Performing an Active Directory Security Review

hub.trimarcsecurity.com/post/securing-active-directory-performing-an-active-directory-security-review

Sean Metcalf June 23, 2020

During the Trimarc Webcast on June 17, 2020, Sean Metcalf covered a number of Active Directory (AD) components and areas that should be reviewed for potential security issues. The presentation included PowerShell code in the presentation and that code is incorporated in the PowerShell script Trimarc released for free that can be used to perform an AD security scan. This script is called Invoke-TrimarcADChecks.ps1 and only needs AD user rights to be run.

Download the Invoke-TrimarcADChecks,ps1 by visiting the Trimarc Github Page.
Last updated: 02-13-2024

The Invoke-TrimarcADChecks.ps1 PowerShell script is designed to gather data from a single domain AD forest based on our similar checks performed during Trimarc's Active Directory Security Assessment (ADSA) engagement. It can be run against each domain in a multi-domain environment, but there is no guarantee that it captures the type of cross-domain (or cross-forest) data elements that may be involved. This script is meant to enable smaller organizations to more quickly identify potential issues. For larger, more complex AD environments, please contact Trimarc to talk about your concerns.

This script is designed for a single AD forest and is not designed to capture all data for a multiple domain forest. If this script is used for a single domain in a multi-domain AD forest, not all elements may be captured.

The Invoke-TrimarcADChecks.ps1 PowerShell script requires the following:

- PowerShell 5.0 (minimum)
- Windows 10 or Windows Server 2016 (or newer)
- Active Directory PowerShell Module
- Group Policy PowerShell Module

If the above requirements are not met, results will be inconsistent.

This script is licensed under BSD 3-Clause License and is provided as-is, without support.

The PowerShell script gathers domain data using the Active Directory PowerShell & Group Policy modules and displays some results on the screen. The data shown on the screen is also saved to a transcript log file and all captured data is also saved to csv/text files in c:\temp\Trimarc-ADReports (report folder is created upon script execution and this folder location can be changed).

This article covers the key AD security areas in which the Invoke-TrimarcADChecks.ps1 PowerShell script gathers data.

The Invoke-TrimarcADChecks PowerShell script gathers data for key AD security items in a domain:

- User Account Issues
- Domain Password Policy
- Tombstone Lifetime & AD Backup Dates
- Trusts
- Duplicate SPNs
- Group Policy Preference Passwords
- AD Administration & Privileged Accounts
- KRBTGT Account
- Kerberos Delegation
- Group Policy Object Owners

User Account Issues

There are several potential issues that Active Directory domain user accounts may have. Trimarc scans for these (and others) during our Active Directory Security Assessment (ADSA) engagement.

- Inactive accounts – these are accounts that have not changed their password or logged on in a while (we typically focus on accounts with a combination of Password Age & User Logon Age over 90 to 180 days old)

- Reversible Encryption – accounts configured with reversible encryption are stored in the AD database in a reversible format (which effectively means a clear-text version of the password is available).
- Password Not Required – accounts created programmatically may have this set which means that technically a password is not required for the account to authenticate.
- Password Never Expires – accounts with a password that is not subject to password expiration requirements. These accounts typically have very old passwords.
- DES Kerberos Encryption Enabled – accounts enabled for Kerberos DES encryption.
- Do not require Kerberos Pre-Authentication – accounts that don't require Kerberos pre-authentication. There are known attacks against accounts with this configuration.
- SID History – accounts with SID History configured

Trimarc Recommendation:

- Disable and eventually delete inactive accounts
- Remove the following from accounts:
    - Reversible Encryption
    - Password Not Required
    - Password Never Expires
    - DES Kerberos Encryption Enabled
    - Do not require Kerberos Pre-Authentication

        Review accounts configured with SID History and clean up SIDs in SID History for accounts from domains that no longer have trusts configured with them.

Sample PowerShell Code & Results:

```powershell
## Get Domain User Information
$LastLoggedOnDate = $(Get-Date) - $(New-TimeSpan -days $UserLogonAge)
$PasswordStaleDate = $(Get-Date) - $(New-TimeSpan -days $UserPasswordAge)
$ADLimitedProperties = @("Name","Enabled","SAMAccountname","DisplayName","Enabled","LastLogonDate","PasswordLastSet","P

[array]$DomainUsers = Get-ADUser -Filter * -Property $ADLimitedProperties -Server $DomainDC
[array]$DomainEnabledUsers = $DomainUsers | Where {$_.Enabled -eq $True }
[array]$DomainEnabledInactiveUsers = $DomainEnabledUsers | Where { ($_.LastLogonDate -le $LastLoggedOnDate) -AND `
  ($_.PasswordLastSet -le $PasswordStaleDate) }

[array]$DomainUsersWithReversibleEncryptionPasswordArray = $DomainUsers | Where { $_.UserAccountControl -band 0x0080 }
[array]$DomainUserPasswordNotRequiredArray = $DomainUsers | Where {$_.PasswordNotRequired -eq $True}
[array]$DomainUserPasswordNeverExpiresArray = $DomainUsers | Where {$_.PasswordNeverExpires -eq $True}
[array]$DomainKerberosDESUsersArray = $DomainUsers | Where { $_.UserAccountControl -band 0x200000 }
[array]$DomainUserDoesNotRequirePreAuthArray = $DomainUsers | Where {$_.DoesNotRequirePreAuth -eq $True}
[array]$DomainUsersWithSIDHistoryArray = $DomainUsers | Where {$_.SIDHistory -like "*"}

Write-Output " "
$DomainUserReport =
@"
$Domain Domain User Report:

Total Users: $($DomainUsers.Count)
Enabled Users: $($DomainEnabledUsers.Count)

Enabled Users Identified as Inactive: $($DomainEnabledInactiveUsers.Count)
Enabled Users With Reversible Encryption Password: $($DomainUsersWithReversibleEncryptionPasswordArray.Count)
Enabled Users With Password Not Required: $($DomainUserPasswordNotRequiredArray.Count)
Enabled Users With Password Never Expires: $($DomainUserPasswordNeverExpiresArray.Count)
Enabled Users With Kerberos DES: $($DomainKerberosDESUsersArray.Count)
Enabled Users That Do Not Require Kerberos Pre-Authentication: $($DomainUserDoesNotRequirePreAuthArray.Count)
Enabled Users With SID History: $($DomainUsersWithSIDHistoryArray.Count)

Review & clean up as appropriate

"@
$DomainUserReport
```

```
lab.trimarcresearch.com Domain User Report:

Total Users: 284
Enabled Users: 85

Enabled Users Identified as Inactive: 75
Enabled Users With Reversible Encryption Password: 3
Enabled Users With Password Not Required: 2
Enabled Users With Password Never Expires: 33
Enabled Users With Kerberos DES: 4
Enabled Users That Do Not Require Kerberos Pre-Authentication: 0
Enabled Users With SID History: 0

Review & clean up as appropriate
```

Domain Password Policy

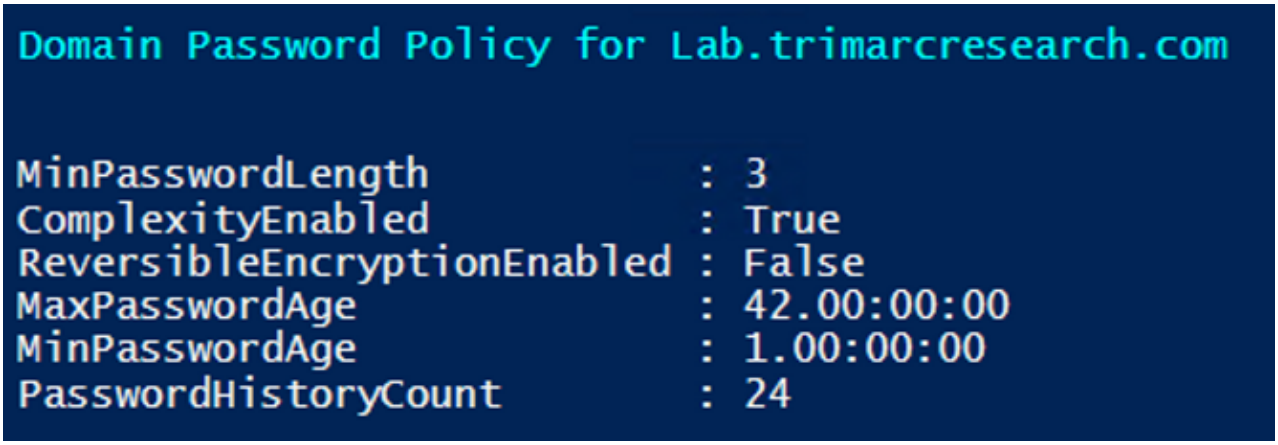
The Domain Password policy determines how passwords are created and how often they need to be changed, etc. The AD default password minimum is 7 characters. Most AD environments we see have this set to between 8 and 10. Some are set to 0 or 3 or 5.

Password Spray attacks are effective against Active Directory due to bad passwords (often with short minimum requirements).Increasing password length can limit Password Spray effectiveness. Fine-Grained Password Policies (FGPP) provide additional flexibility, especially for admin and service accounts.

Trimarc Recommendation: Trimarc recommends that the Domain Password Policy be set to 12 characters (preferably 14 to 16 characters).

Sample PowerShell Code & Results:

The Active Directory module cmdlet “Get-ADPasswordPolicy” can easily capture the domain password policy.



Tombstone Lifetime

The AD Tombstone Lifetime determines how long deleted items exist in AD before they are purged. The default value was originally 60 days, but this was increased to 180 days starting with new AD forests created with Windows 2003 SP1. While the tombstone lifetime directly affects deleted items, it also has an impact on Domain Controllers. If a DC hasn’t replicated within the tombstone lifetime with another DC, it is effectively orphaned from the domain. Additionally, DC backups are only useful for restoring AD data within this tombstone lifetime – a backup that is 181 days old is no longer useful when the tombstone lifetime is 180 days.

Trimarc Recommendation: Trimarc recommends configuring the Tombstone Lifetime to 180 days (or more if needed).

Active Directory Backups

Microsoft supported backups update a partition attribute to identify the last backup date for that partition. Not all systems that backup Active Directory set this attribute since they are likely not using a Microsoft supported method. In most scenarios, System State Backups on a Domain Controller are required to properly backup AD. We can easily check last Microsoft supported AD backup date/time by querying an attribute on each partition. Trimarc also identified that events like “Install From Media” creation on a DC can update AD backup date/time attribute.

Trimarc Recommendation:

Ensure you run a System State Backup on at least the FSMO role holders in the AD forest at least once a month and keep for at least 6 months.

Sample PowerShell Code & Results:

```
## Tombstone Lifetime & AD Backups
$ADRootDSE = get-adrootdse
$ADConfigurationNamingContext = $ADRootDSE.configurationNamingContext
$ForestRootDN = $ADRootDSE.rootDomainNamingContext
$ForestNCs = $ADRootDSE.NamingContexts
$DomainControllerSiteNameDN = "CN=Sites,$ADConfigurationNamingContext"
$TombstoneObjectInfo = Get-ADObject -Identity "CN=Directory Service,CN=Windows NT,CN=Services,$ADConfigurationNamingContext" `
-Partition "$ADConfigurationNamingContext" -Properties *
[int]$TombstoneLifetime = $TombstoneObjectInfo.tombstoneLifetime
IF ($TombstoneLifetime -eq 0) { $TombstoneLifetime = 60 }
Write-Host "The AD Forest Tombstone lifetime is set to $TombstoneLifetime days." -Fore Cyan
```

```
## AD Backups
# From https://devblogs.microsoft.com/scripting/use-a-powershell-script-to-show-active-directory-backup-status-info/
[string[]]$Partitions = (Get-ADRootDSE).namingContexts
$contextType = [System.DirectoryServices.ActiveDirectory.DirectoryContextType]::Domain
$context = new-object System.DirectoryServices.ActiveDirectory.DirectoryContext($contextType,$ADDomainName)
$domainController = [System.DirectoryServices.ActiveDirectory.DomainController]::findOne($context)
Write-Host "Determining last supported backup of AD partitions... `n" -ForegroundColor Cyan
ForEach($partition in $partitions)
{
    $domainControllerMetadata = $domainController.GetReplicationMetadata($partition)
    $dsaSignature = $domainControllerMetadata.Item("dsaSignature")
    Write-Host "$partition was backed up $($dsaSignature.LastOriginatingChangeTime.DateTime)"
}
```

```
The AD Forest Tombstone lifetime is set to 30 days.

Determining last supported backup of AD partitions...
CN=Configuration,DC=trimarcresearch,DC=com was backed up Wednesday, May 6, 2020 8:44:21 PM
CN=Schema,CN=Configuration,DC=trimarcresearch,DC=com was backed up Wednesday, May 6, 2020 8:44:21 PM
DC=ForestDnsZones,DC=trimarcresearch,DC=com was backed up Wednesday, May 6, 2020 8:44:21 PM
DC=Lab,DC=trimarcresearch,DC=com was backed up Wednesday, May 6, 2020 8:44:21 PM
DC=DomainDnsZones,DC=Lab,DC=trimarcresearch,DC=com was backed up Wednesday, May 6, 2020 8:44:21 PM
```

Trusts

An Active Directory forest is the security boundary for AD. When a trust is configured, that extends the security boundary in include the system included in the trust since the trust extends the authentication boundary.

Trimarc Recommendation:

- Review trust configuration & ensure that trusts are appropriate. Trusts should be reviewed at least annually to ensure they are still required
- Bidirectional trusts should be scrutinized to ensure they need to be two-way trusts and still required.
- Trusts with DMZ environments should be removed (if possible).
- If a trust is required, look to shift to Selective Authentication which changes from Allow All to only those explicitly allowed to connect across the trust.

Selective Authentication changes the default behavior from allow any user across the trust to see everything to users require explicit approval to access resources otherwise they can't view or access anything across the trust

Sample PowerShell Code & Results:

```
## Trusts
$ADTrusts = Get-ADTrust -Filter * -Server $DomainDC
$ADTrustFile = $ReportDir + "\TrimarcADChecks-DomainTrustReport-$Domain-$TimeVal.csv"
$ADTrusts | Export-CSV $ADTrustFile -NoTypeInformation
Write-Host "$Domain Active Directory Trusts: `n" -Fore Cyan
$ADTrusts | Select Source,Target,Direction,IntraForest,SelectiveAuth,SIDFilteringForestAware,SIDFilteringQuarantined `
| Format-Table -AutoSize
Write-Host "Active Directory Trusts ($Domain) saved to the file $ADTrustFile `n"
```

```
lab.trimarcresearch.com Active Directory Trusts:


Source                       : DC=Lab,DC=trimarcresearch,DC=com
Target                       : trimarcresearch.com
Direction                    : BiDirectional
IntraForest                  : True
SelectiveAuth                : {}
SIDFilteringForestAware      : False
SIDFilteringQuarantined      : False
```

Active Directory Duplicate Service Principal Names (SPNs)

The Kerberos Service Principal Name (SPN) is effectively a signpost that connects a service on a server supporting Kerberos authentication with the service account. When there are duplicate SPNs, Kerberos authentication breaks since a Domain Controller can't identify a single account associated with the SPN. Authentication will need to fallback to NTLM authentication. Duplication of the Krbtgt config is normal in a multiple domains in an AD forest.

Trimarc Recommendation:

Ensure there aren't any duplicate SPNs (other than krbtgt).

Sample Code & Results:

Using SetSPN.exe we can discover any duplicate SPNs in the forest using the following command:

```
SetSPN -X -F
```

Any identified duplicate SPNs should be resolved since duplicate SPNs break Kerberos authentication which forces NTLM for authentication.

```
PS C:\> SetSPN -X -F | where {$_ -notlike "Processing entry*"}
Checking forest DC=trimarcresearch,DC=com
Operation will be performed forestwide, it might take a while.

MSSQLSvc/EchoDB11:1433 is registered on these accounts:
    CN=Alpha35_svc,OU=Service Accounts,DC=Prod,DC=trimarcresearch,DC=com
    CN=Alpha42_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com

kadmin/changepw is registered on these accounts:
    CN=krbtgt,CN=Users,DC=Dev,DC=trimarcresearch,DC=com
    CN=krbtgt,CN=Users,DC=Prod,DC=trimarcresearch,DC=com
    CN=krbtgt,CN=Users,DC=trimarcresearch,DC=com
    CN=krbtgt,CN=Users,DC=Lab,DC=trimarcresearch,DC=com

found 2 groups of duplicate SPNs.
```

Group Policy Preference Passwords

Group Policy Preferences was released in the 2008 time-frame and included capability to provide and update credentials. These credentials were encrypted using AES256 which sounds good until you realize that a static key is used to encrypt them. The cpassword value in the GPP xml files is the encrypted password. Using a PowerShell function from PowerSploit, we can reverse this encryption and get the plain-text value. Since authenticated users have read access to Group Policies in the SYSVOL share on all DCs, anyone can view this information and get the passwords stored in GPP xml files, even across trusts. This is less of an issue now that Microsoft released a patch preventing the use of Group Policy Preference passwords. Old GPOs with these credentials remain though and Trimarc has identified a few in the past year during AD security assessments that would lead to either full AD compromise or compromising sensitive server data.

GPP capability with credentials:

- Map drives (Drives.xml)
- Create Local Users
- Data Sources (DataSources.xml)
- Printer configuration (Printers.xml)
- Create/Update Services (Services.xml)
- Scheduled Tasks (ScheduledTasks.xml)
- Change local Administrator passwords

Trimarc Recommendation:

Ensure there are no Group Policy Preference passwords in SYSVOL.

Sample Code & Results:

The fast way to identify passwords in Group Policy Preferences is by using the findstr.exe command:

```
Findstr /S /I cpassword [SYSVOL_PATH]
```

```
PS C:\temp> (findstr /S /I cpassword $DomainSYSVOLShareScan ) -Split ':'
\\lab.trimarcresearch.com\SYSVOL\lab.trimarcresearch.com\Policies\{BC5325A39-9710-4B1F-93A5-AF754E97AC1A}\groups.xml
cpassword="3462erwehgwryhwrtgq9eihurt89`34ht89qhergiudhfgqhsdufgbqsoduiqhret89hqr89hqrht"
```

A more comprehensive method is the Microsoft scanner provided here.

References: https://adsecurity.org/?p=2288

Active Directory Admin Account Checks

During Trimarc's standard Active Directory Security Assessment, we focus on identifying "AD Admins" which includes members of the domain Administrators group, Domain Admins, Enterprise Admins, etc. These accounts have full AD rights and require careful protection.

The fastest way to identify AD admins is to run the following command which leverages the AD PowerShell module:

```
Get-ADGroupMember 'Administrators' -Recursive
```

Trimarc Recommendation:

- Ensure passwords change regularly (every year)
- Disable inactive account
- Remove disabled accounts
- Ensure no SPNs on accounts associated with people
- Remove any computer accounts
- Scrutinize Service Accounts
    - What do they do?
    - Where do they run?
    - What computers do they authenticate to?
    - What rights are actually required?

PowerShell Sample Code & Results:

First, we get the recursive membership of the domain Administrators group (which includes all the members of member groups).

Then, loop through them and get more detail from each member.

```
## Identify AD Admins
$ADAdminArray = @()
$ADAdminMembers = Get-ADGroupMember Administrators -Recursive
ForEach ($ADAdminMemberItem in $ADAdminMembers)
 {
  Switch ($ADAdminMemberItem.objectClass)
   {
    'User' { [array]$ADAdminArray += Get-ADUser $ADAdminMemberItem -Properties LastLogonDate,PasswordLastSet,ServicePrincipalName }
    'Computer' { [array]$ADAdminArray += Get-ADComputer $ADAdminMemberItem -Properties LastLogonDate,PasswordLastSet }
    'msDS-GroupManagedServiceAccount' { [array]$ADAdminArray += Get-ADServiceAccount $ADAdminMemberItem -Properties LastLogonDate,PasswordLastSet }
   }
 }
Write-Host " "
Write-Host "$ADDomainName AD Admins: " -Fore Cyan
$ADAdminArray | sort PasswordLastSet | select name,DistinguishedName,PasswordLastSet,LastLogonDate,ObjectClass | Format-Table -AutoSize
```

```
Lab.trimarcresearch.com AD Admins:

name          DistinguishedName                                                    PasswordLastSet       LastLogonDate        ObjectClass
----          -----------------                                                    ---------------       -------------        -----------
SVC-LAB-GMSA1 CN=SVC-LAB-GMSA1,CN=Managed Service Accounts,DC=Lab,DC=trimarcresearch,DC=com     6/10/2020 8:15:07 AM  6/2/2020 3:06:11 AM  msDS-GroupManagedServiceAccount
Administrator CN=Administrator,CN=Users,DC=Lab,DC=trimarcresearch,DC=com            5/19/2020 4:32:44 PM  5/11/2020 1:16:56 PM user
VMWareAdmin   CN=VMWareAdmin,OU=Service Accounts,DC=trimarcresearch,DC=com         11/10/2019 11:57:14 PM                     user
Administrator CN=Administrator,CN=Users,DC=trimarcresearch,DC=com                  2/11/2020 2:08:55 PM  6/9/2020 8:12:05 PM   user
SharepointSVC CN=SharepointSVC,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com 11/13/2019 9:18:33 AM                      user
admEGray      CN=admEGray,OU=Admin Accounts,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com   11/10/2019 11:27:06 PM                     user
admMBailey    CN=admMBailey,OU=Admin Accounts,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com 11/10/2019 11:26:46 PM                     user
```

AD Admin Accounts Not Member of Protected Users

Protected Users is a new group created when the domain PDC Emulator is running Windows Server 2012 R2. Full Domain protection is only available when the domain functional level is 2012 R2.

Protected Users group provides additional protections:

- Kerberos AES authentication only (No Kerberos DSE/RC4 or NTLM)
- No Kerberos delegation (constrained or unconstrained)
- Kerberos TGT set to 4 hours
- Credential delegation (CredSSP) will not cache the user's plain text credentials
- NTLM will not cache the user's plain text credentials or NT one-way function
- Offline sign-in is not supported

Trimarc Recommendation:

All AD Admin accounts should be added to the “Protected Users” group

PowerShell Sample Code & Results:

Compare AD Admins group membership output with the results of the following command:

```
Get-ADGroupMember 'Protected Users'
```

```
Lab.trimarcresearch.com AD Admins:

name          DistinguishedName                                                                                  LastLogonDate        PasswordLastSet      ObjectClass
----          -----------------                                                                                  -------------        ---------------      -----------
SVC-LAB-GMSA1 CN=SVC-LAB-GMSA1,CN=Managed Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                      6/2/2020 3:06:11 AM  6/10/2020 8:15:07 AM  msDS-GroupManagedServiceAccount
Administrator CN=Administrator,CN=Users,DC=Lab,DC=trimarcresearch,DC=com                                          5/11/2020 1:16:56 PM 5/19/2020 4:32:44 PM  user
VMWareAdmin   CN=VMWareAdmin,OU=Service Accounts,DC=trimarcresearch,DC=com                                                             11/10/2019 11:57:14 PM user
Administrator CN=Administrator,CN=Users,DC=trimarcresearch,DC=com                                                6/9/2020 8:12:05 PM  2/11/2020 2:08:55 PM  user
SharepointSVC CN=SharepointSVC,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                                                    11/13/2019 9:18:33 AM  user
admEGray      CN=admEGray,OU=Admin Accounts,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com                                          11/10/2019 11:27:06 PM user
admMBailey    CN=admMBailey,OU=Admin Accounts,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com                                        11/10/2019 11:26:46 PM user
```

```
Lab.trimarcresearch.com Domain Protected Users Group Membership:

name        DistinguishedName                                                                          objectClass
----        -----------------                                                                          -----------
admAGreen   CN=admAGreen,OU=Admin Accounts,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com           user
admBCollins CN=admBCollins,OU=Admin Accounts,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com         user
admALong    CN=admALong,OU=Admin Accounts,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com            user
admAButler  CN=admAButler,OU=Admin Accounts,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com          user
admGMoore   CN=admGMoore,OU=Admin Accounts,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com           user
admNEvans   CN=admNEvans,OU=Admin Accounts,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com           user
admDSimmons CN=admDSimmons,OU=Admin Accounts,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com         user
```

AD Admins with Old Passwords

AD Admin accounts with old passwords, especially those older than 5 years, are vulnerable to password spraying (and password guessing).

Trimarc Recommendation:

- Ensure privileged account passwords change annually.
- Older passwords are typically poor and easier to guess.
- Password Spraying & Kerberoasting are popular attack methods for compromising accounts lacking strong passwords.

PowerShell Sample Code & Results: *Note that this check leverages the AD Admin array data collected earlier.*

```
Lab.trimarcresearch.com AD Admins:

name          DistinguishedName                                                                          PasswordLastSet
----          -----------------                                                                          ---------------
admMBailey    CN=admMBailey,OU=Admin Accounts,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com          11/10/2019 11:26:46 PM
admEGray      CN=admEGray,OU=Admin Accounts,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com            11/10/2019 11:27:06 PM
VMWareAdmin   CN=VMWareAdmin,OU=Service Accounts,DC=trimarcresearch,DC=com                              11/10/2019 11:57:14 PM
SharepointSVC CN=SharepointSVC,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                      11/13/2019 9:18:33 AM
Administrator CN=Administrator,CN=Users,DC=trimarcresearch,DC=com                                        2/11/2020 2:08:55 PM
Administrator CN=Administrator,CN=Users,DC=Lab,DC=trimarcresearch,DC=com                                 5/19/2020 4:32:44 PM
SVC-LAB-GMSA1 CN=SVC-LAB-GMSA1,CN=Managed Service Accounts,DC=Lab,DC=trimarcresearch,DC=com              6/10/2020 8:15:07 AM
```

Using the information from PowerShell, we can build a table in Excel.

This is an adjusted version similar to what we find in customer environments during AD security assessments

SamAccountName	Enabled	PasswordLastSet	Password Age (years)
Administrator	**Yes**	**1/9/2005 10:58:24 AM**	**15**
AGPMService	**Yes**	**5/3/2009 3:17:32 PM**	**10.8**
SCCMsvc	**Yes**	**11/14/2011 5:23:12 PM**	**8.6**
VMWareAdmin	**Yes**	**8/28/2012 10:23:41 AM**	**7.5**
admAEdwards	**Yes**	**1/12/2013 2:20:06 PM**	**7.0**
VulnerabilityScanner	**Yes**	**9/19/2015 4:43:19 PM**	**4.5**
admBWalker	No	6/11/2017 10:14:08 AM	2.8
admCGriffin	Yes	3/1/2019 12:41:18 PM	1.0

AD Admins with Kerberos Service Principal Names (SPNs)

Kerberos Service Principal Name or SPN is effectively the signpost that points to the service account for a service on a server that supports Kerberos authentication. When the client needs to connect to a service, it must request a Kerberos service ticket from a DC and in order to do this it needs to provide a SPN for that service. The DC looks up the account in the AD forest that has that SPN, identifies the account, and uses the account's password data to encrypt the ticket. Once the service ticket is delivered to the service, it attempts to open the service ticket and if it can, it can assume the DC provided it, so it validates access for the user.

Kerberoasting, developed by Tim Medin in 2014, requests service tickets (usually RC4 encrypted) for target service accounts and saves them to attempt offline brute password forcing – once the attacker can open a service ticket, they have successfully guessed the account password. This is a big deal and I have been talking about the importance of ensuring AD admin accounts don't have SPNs and service accounts needing long passwords since Tim released his Kerberoast tool at DerbyCon 2014.

Attackers can "Kerberoast" accounts with Kerberos Service Principal Names (SPNs). This shifted attacking service accounts with old passwords from hypothetical to practical and many attackers use this method now. Trimarc published Kerberoast detection with 100% accuracy, no false positives as while back on the Trimarc website.

Kerberoast in a nutshell:

1. Request Kerberos service ticket for SPN (RC4 encrypted which uses the NT password hash for the ticket encryption key).
2. Move the service ticket to a password cracking system and perform hybrid password guessing

3. Guess the password correctly and the service ticket opens.

Trimarc Recommendation:

- Ensure that no AD Admin accounts associated with people have SPNs.
- Limit service account membership in privileged Active Directory groups and ensure these service account passwords are longer than 25 characters (to mitigate Kerberoasting).

PowerShell Sample Code & Results:

Here we simply filter the AD Admins list by only displaying the accounts with SPNs. *Note that this check leverages the AD Admin array data collected earlier.*

```
Lab.trimarcresearch.com AD Admin Accounts with SPNs:

name          DistinguishedName                                                ServicePrincipalName
----          -----------------                                                --------------------
Administrator CN=Administrator,CN=Users,DC=Lab,DC=trimarcresearch,DC=com       {MSSQLSvc/GammaDB23:1434, MSSQLSvc/GammaDB14:1434, MSSQLSvc/GammaDB007:1433, MSSQLSvc/GammaDB001:1..
SharepointSVC CN=SharepointSVC,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com {http/TRDSharePoint02, http/TRDSharePoint01}
admEGray      CN=admEGray,OU=Admin Accounts,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com {gateway/gatewayAPP47}
```

Check Default Domain Administrator Account for Issues

The default domain administrator account should only be used when the other AD admin accounts are inaccessible. It should not be logged on to so set loud alerting for when it does. This account should never have a SPN since that means it is being used as a service account for a service on a server on the network. This means that compromise of that server would compromise Active Directory. Or Kerberoast the domain Administrator account and get that password to compromise AD.

If this account is disabled, GREAT! Are you monitoring for it being re-enabled?

Trimarc Recommendation:

- The account password should change at least annually (and when an AD Admin leaves the organization).
- Ensure the account has no SPNs.
- Account should be reserved as an emergency account (aka “break glass).
- The account can be enabled or disabled.

PowerShell Sample Code & Results:

```
## Default Domain Administrator Account
$DomainAdminAccountSID = "$($ADDomainInfo.DomainSID)-500"
$DomainDefaultAdminAccount = Get-ADUser $DomainAdminAccountSID -Server $DomainDC -Properties Name,Enabled,Created,PasswordLastSet,LastLogonDate,ServicePrincipalName,SID
Write-Output "$Domain Default Domain Administrator Account:" # -Fore Cyan
$DomainDefaultAdminAccount | Select Name,Enabled,Created,PasswordLastSet,LastLogonDate,ServicePrincipalName | Format-Table -AutoSize
$ForestDomainDefaultAdminAccountFile = $ReportDir + "\TrimarcADChecks-DomainDefaultAdminAccount-$Domain-$TimeVal.csv"
$DomainDefaultAdminAccount | Export-CSV $ForestDomainDefaultAdminAccountFile -NoTypeInformation
```

```
lab.trimarcresearch.com Default Domain Administrator Account:

Name          Enabled Created              PasswordLastSet      LastLogonDate        ServicePrincipalName
----          ------- -------              ---------------      -------------        --------------------
Administrator    True 11/10/2019 3:36:51 PM 5/19/2020 4:32:44 PM 5/11/2020 1:16:56 PM {MSSQLSvc/GammaDB23:1434, MSSQLSvc/GammaDB14:1434, MSSQLSvc/GammaDB007:1433, MSSQLSvc/GammaDB001:1433}
```

Review AD Default/Built-In Group Membership

Reviewing the default privileged groups in Active Directory is important to identify accounts with high-level privileges.

Trimarc Recommendation:

Ensure that only accounts that require full AD rights are members of these groups.

PowerShell Sample Code & Results:

We can create an array of the default/builtin common groups that have privileged in AD and loop through these to get the group membership

```
## Privileged AD Group Array
$ADPrivGroupArray = @(
 'Administrators',
 'Domain Admins',
 'Enterprise Admins',
 'Schema Admins',
 'Account Operators',
 'Server Operators',
 'Group Policy Creator Owners',
 'DNSAdmins',
 'Enterprise Key Admins',
 # Exchange Privileged Groups
 'Exchange Domain Servers',
 'Exchange Enterprise Servers',
 'Exchange Admins',
 'Organization Management',
 'Exchange Windows Permissions'
)
```

Here's the looping code which gets the group membership and counts members. Note this membership enumeration is NOT recursive, so any groups listed in these groups need to be enumerated separately (add to the Array above to include them).

```powershell
## Discover Default privileged group membership
ForEach ($ADPrivGroupItem in $ADPrivGroupArray)
{
    $ADPrivGroupItemGroupMembership = @()
    TRY
     {
        $ADPrivGroupItemGroupMembership = Get-ADGroupMember $ADPrivGroupItem -Server $DomainDC
        IF ($ADPrivGroupItemGroupMembership.count -ge 1)
         {
            Write-Host "$ADDomainName Domain $ADPrivGroupItem Group:" -Fore Cyan
            $ADPrivGroupItemGroupMembership | Select name,DistinguishedName,objectClass | Format-Table

            $ADPrivGroupItemGroupMembershipFile = $ReportDir + "\TrimarcADChecks-PrivGroups-$Domain-$ADPrivGroupItem-$TimeVal.csv"
            $ADPrivGroupItemGroupMembership | Export-CSV $ADPrivGroupItemGroupMembershipFile -NoTypeInformation
         }
        ELSE
         { Write-Host "$ADDomainName Domain $ADPrivGroupItem Group:  No members" -Fore Cyan }
     }
    CATCH
     { Write-Warning "An error occured when attempting to enumerate group membership for the group $ADPrivGroupItem `
     in the domain $Domain using the DC $DomainDC" }

    Write-Host ""
}
```

The following screenshots show output. The PowerShell script will show an error if it can't enumerate the group which is usually because it doesn't exist (or is inaccessible).

Default AD Groups: Administrators

Default Rights:

- Active Directory admin rights (for the domain)
- Domain Controller admin rights (for the domain)

PowerShell Sample Code & Results:

Get-ADGroupMember 'Administrators'

```
Lab.trimarcresearch.com Domain Administrators Group:

name                DistinguishedName                                                                  objectClass
----                -----------------                                                                  -----------
SVC-LAB-GMSA1       CN=SVC-LAB-GMSA1,CN=Managed Service Accounts,DC=Lab,DC=trimarcresearch,DC=com      msDS-GroupManagedServiceAccount
Server Admins - DCs CN=Server Admins - DCs,OU=Admin Groups,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com group
Enterprise Admins   CN=Enterprise Admins,CN=Users,DC=trimarcresearch,DC=com                            group
Domain Admins       CN=Domain Admins,CN=Users,DC=Lab,DC=trimarcresearch,DC=com                         group
Administrator       CN=Administrator,CN=Users,DC=Lab,DC=trimarcresearch,DC=com                         user
```

Reference:

https://docs.microsoft.com/en-us/windows/security/identity-protection/access-control/active-directory-security-groups#bkmk-admins

Default AD Groups: Domain Admins

Default Rights:

- Membership in the domain Administrators group which provides most rights.
- Active Directory admin rights (for the domain)
- Domain Controller admin rights (for the domain)
- Default rights on all domain Group Policies
- Default local Administrator on all domain-joined computers

PowerShell Sample Code & Results:

Get-ADGroupMember 'Domain Admins'

```
Lab.trimarcresearch.com Domain Domain Admins Group:

name          DistinguishedName                                                    objectClass
----          -----------------                                                    -----------
Administrator CN=Administrator,CN=Users,DC=Lab,DC=trimarcresearch,DC=com           user
admMBailey    CN=admMBailey,OU=Admin Accounts,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com user
admEGray      CN=admEGray,OU=Admin Accounts,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com   user
SharepointSVC CN=SharepointSVC,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com  user
admCMorgan    CN=admCMorgan,OU=Admin Accounts,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com user
```

Reference:

https://docs.microsoft.com/en-us/windows/security/identity-protection/access-control/active-directory-security-groups#bkmk-domainadmins

Default AD Groups: Enterprise Admins

Default Rights:

- Membership in every domain Administrators group which provides most rights.
- Active Directory admin rights (in every forest domain)
- Domain Controller admin rights (in every forest domain)
- Default rights on all domain Group Policies
- This group should remain empty in a single domain forest and membership very limited in a multi-domain forest.

PowerShell Sample Code & Results:

Get-ADGroupMember 'Enterprise Admins'

```
trimarcresearch.com Domain Enterprise Admins Group:

name          DistinguishedName                                          objectClass
----          -----------------                                          -----------
VMWareAdmin   CN=VMWareAdmin,OU=Service Accounts,DC=trimarcresearch,DC=com user
Administrator CN=Administrator,CN=Users,DC=trimarcresearch,DC=com         user
```

Reference:

https://docs.microsoft.com/en-us/windows/security/identity-protection/access-control/active-directory-security-groups#bkmk-entadmins

Default AD Groups: Server Operators

This group is effectively "Domain Controller Admins" and members of this group should be scrutinized at a similar level to Domain Admins. This group has no default members.
Default Rights:

Default rights on Domain Controllers:

- Log on locally
- create and delete shared resources
- start and stop some services
- backup and restore files
- format the hard disk
- shut down the computer

PowerShell Sample Code & Results:

Get-ADGroupMember 'Server Operators'

```
Lab.trimarcresearch.com Domain Server Operators Group:

name              DistinguishedName                                                                  objectClass
----              -----------------                                                                  -----------
Tyrone Johnson    CN=Tyrone Johnson,OU=Users,OU=WashingtonDC,OU=Branch Offices,DC=Lab,DC=trimarcresearch,DC=com   user
Keshia Washington CN=Keshia Washington,OU=Users,OU=NewYorkCity,OU=Branch Offices,DC=Lab,DC=trimarcresearch,DC=com user
SVC-LAB-GMSA1     CN=SVC-LAB-GMSA1,CN=Managed Service Accounts,DC=Lab,DC=trimarcresearch,DC=com               msDS-GroupManagedServiceAccount
admCJohnson       CN=admCJohnson,OU=Admin Accounts,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com           user
```

Reference:

https://docs.microsoft.com/en-us/windows/security/identity-protection/access-control/active-directory-security-groups#bkmk-serveroperators

Default AD Groups: Account Operators

This group has no default members. This group should remain empty.

Default Rights:

Has rights to most objects in the domain (users, groups, computers, etc).

Microsoft recommends this group remain empty.

> ⓘ Note
>
> By default, this built-in group has no members, and it can create and manage users and groups in the domain, including its own membership and that of the Server Operators group. This group is considered a service administrator group because it can modify Server Operators, which in turn can modify domain controller settings. As a best practice, leave the membership of this group empty, and do not use it for any delegated administration. This group cannot be renamed, deleted, or moved.

PowerShell Sample Code & Results:

Get-ADGroupMember 'Account Operators'

Reference https://docs.microsoft.com/en-us/windows/security/identity-protection/access-control/active-directory-security-groups#bkmk-accountoperators

Privileged Group: VMWare Admins

Trimarc sees "VMWare Admins" in most environments. The VMWare Admins AD group is granted full VMWare Admin rights. This means that if an attacker can compromise an account that is a member of the VMWare Admins group, they can compromise VMWare. In an environment with virtual Domain Controllers hosting in VMWare, this would result in AD compromise.

Trimarc Recommendation:

- Ensure that "admin" groups only contain admin accounts.
- Ensure that VMWare Admins follow privileged account best practices:
  - Use separate admin accounts
  - Use admin workstations
  - Passwords change about once a year

PowerShell Sample Code & Results:

Get-ADGroupMember 'VMWare Admins'

```
Lab.trimarcresearch.com Domain CN=VMWare Admins,OU=Admin Groups,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com Group:

name            DistinguishedName                                                                                  objectClass
----            -----------------                                                                                  -----------
admEGray        CN=admEGray,OU=Admin Accounts,OU=AD Management,DC=Lab,DC=trimarcresearch,DC=com                     user
Isabel Gonzalez CN=Isabel Gonzalez,OU=Users,OU=Bogota,OU=Branch Offices,DC=Lab,DC=trimarcresearch,DC=com           user
Jose Rodriquez  CN=Jose Rodriquez,OU=Users,OU=BuenosAires,OU=Branch Offices,DC=Lab,DC=trimarcresearch,DC=com       user
Crystal Kennedy CN=Crystal Kennedy,OU=Users,OU=WashingtonDC,OU=Branch Offices,DC=Lab,DC=trimarcresearch,DC=com    user
Ebony Smith     CN=Ebony Smith,OU=Users,OU=Dallas,OU=Branch Offices,DC=Lab,DC=trimarcresearch,DC=com               user
```

Krbtgt Password Not Changed Recently

The krbtgt account is the domain service account. This account is disabled but used for Kerberos Tickets. The password set when created & practically never changes. DC/AD backups contains the KRBTGT account password. If an attacker gains knowledge of this password, they can create Golden Tickets!

Trimarc Recommendation:

- Change this password 2x every year (DoD STIG requirement)
- Change after AD admins leave

PowerShell Sample Code & Results:

```
## KRBTGT Account Password
$DomainKRBTGTAccount = Get-ADUser 'krbtgt' -Server $DomainDC -Properties 'msds-keyversionnumber',Created,PasswordLastSet
Write-Host "$Domain Domain Kerberos Service Account (KRBTGT): `n" -Fore Cyan
$DomainKRBTGTAccount | Select DistinguishedName,Created,PasswordLastSet,'msds-keyversionnumber' | Format-Table -AutoSize
$ForestDomainKRBTGTAccountFile = $ReportDir + "\TrimarcADChecks-DomainKRBTGTAccount-$Domain-$TimeVal.csv"
$DomainKRBTGTAccount | Export-CSV $ForestDomainKRBTGTAccountFile -NoTypeInformation
```

```
lab.trimarcresearch.com Domain Kerberos Service Account (KRBTGT):

DistinguishedName                                Created              PasswordLastSet      msds-keyversionnumber
-----------------                                -------              ---------------      ---------------------
CN=krbtgt,CN=Users,DC=Lab,DC=trimarcresearch,DC=com 11/10/2019 3:37:59 PM 11/10/2019 3:37:59 PM                    2
```

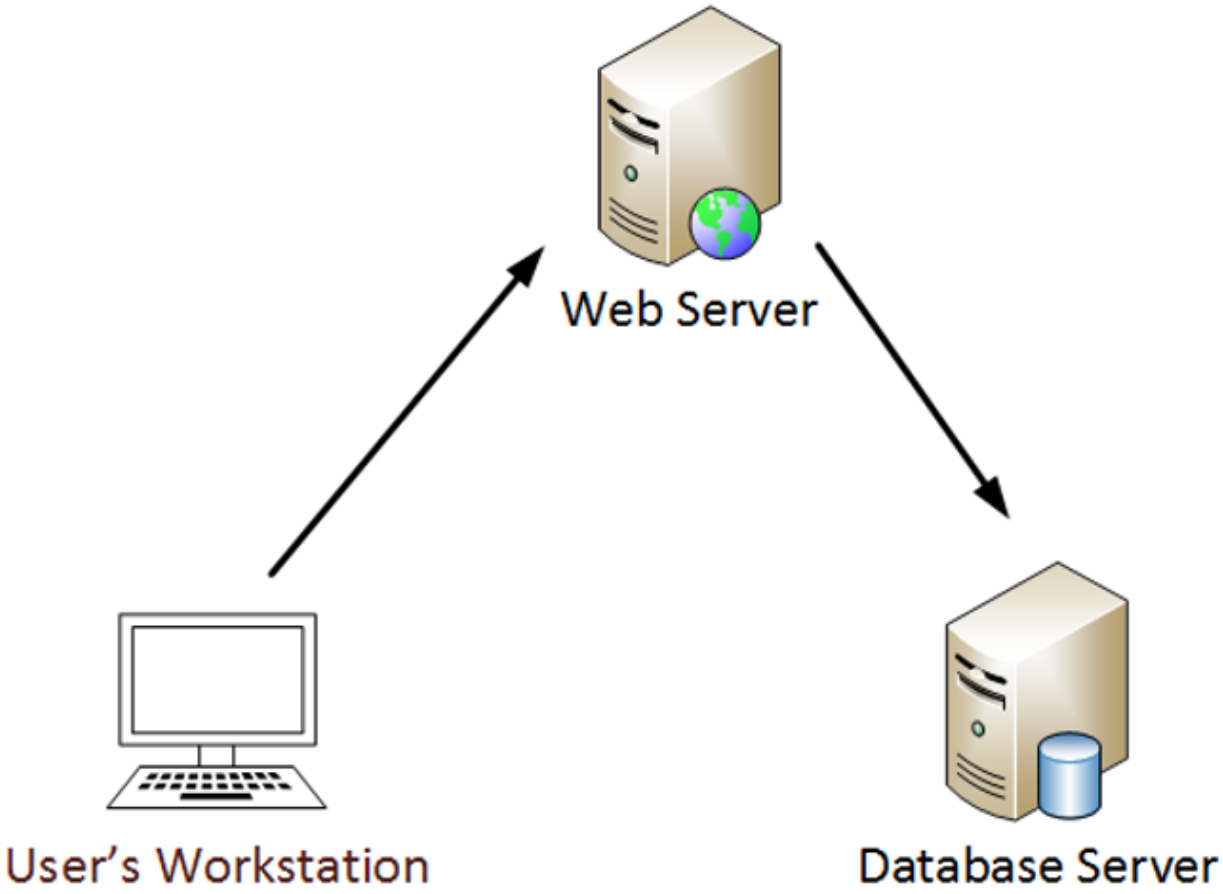
Reference:

- Krbtgt Account https://adsecurity.org/?p=483
- Golden Ticket Attack https://adsecurity.org/?tag=goldenticket

Kerberos Delegation

Kerberos “Double Hop” Issue

The double hop issue occurs when you have a user connecting to a web server which then connects to a back-end database server. The web server updates records on the back-end database on behalf of the user. The problem is that the TGS Service ticket the user provides to the web server is only good for the web server. The web server can’t authenticate to the database server as the user to make the updates *as* the user. Enter delegation. Kerberos Delegation enables the web server to impersonate the user to the database server in order to perform the updates the user requested.



Kerberos Delegation Types:

- **Unconstrained:** Impersonate users connecting to service to ANY Kerberos service.
- **Constrained:** Impersonate authenticated users connecting to service to SPECIFIC Kerberos services on servers.
- **Constrained with Protocol Transition:** Impersonate any user to SPECIFIC Kerberos services on servers. (aka "Kerberos Magic")
- **Resource-based Constrained Delegation:** Enables delegation configured on the resource instead of the account.

Trimarc Recommendation:

- Set all AD Admin accounts to: "Account is sensitive and cannot be delegated"
- Add all AD Admin accounts to the "Protected Users" group (Windows 2012 R2 DCs).
- Ensure service accounts with Kerberos delegation have long, complex passwords (preferably group Managed Service Accounts).
- Remove delegation from accounts that don't require it.
- Don't use Domain Controller SPNs when delegating.

- Work to shift accounts with unconstrained delegation to constrained.
- Restrict & monitor who has the ability to configure Kerberos delegation.

PowerShell Sample Code & Results:

```
## Identify Accounts with Kerberos Delegation
$KerberosDelegationArray = @()
[array]$KerberosDelegationObjects =  Get-ADObject -filter { ( (UserAccountControl -BAND 0x0080000) -OR (UserAccountControl -BAND 0x1000000) -OR `
   (msDS-AllowedToDelegateTo -like '*') -OR (msDS-AllowedToActOnBehalfOfOtherIdentity -like '*')) -AND (PrimaryGroupID -ne '516') -AND `
    (PrimaryGroupID -ne '521') } -Server $DomainDC -prop Name,ObjectClass,PrimaryGroupID,UserAccountControl,ServicePrincipalName, `
     msDS-AllowedToDelegateTo,msDS-AllowedToActOnBehalfOfOtherIdentity -SearchBase $DOmainDN

ForEach ($KerberosDelegationObjectItem in $KerberosDelegationObjects)
 {
    IF ($KerberosDelegationObjectItem.UserAccountControl -BAND 0x0080000)
     { $KerberosDelegationServices = 'All Services' ; $KerberosType = 'Unconstrained' }
    ELSE
     { $KerberosDelegationServices = 'Specific Services' ; $KerberosType = 'Constrained' }

    IF ($KerberosDelegationObjectItem.UserAccountControl -BAND 0x1000000)
     { $KerberosDelegationAllowedProtocols = 'Any (Protocol Transition)' ; $KerberosType = 'Constrained with Protocol Transition' }
    ELSE
     { $KerberosDelegationAllowedProtocols = 'Kerberos' }

    IF ($KerberosDelegationObjectItem.'msDS-AllowedToActOnBehalfOfOtherIdentity')
     { $KerberosType = 'Resource-Based Constrained Delegation'  }

    $KerberosDelegationObjectItem | Add-Member -MemberType NoteProperty -Name Domain -Value $Domain -Force
    $KerberosDelegationObjectItem | Add-Member -MemberType NoteProperty -Name KerberosDelegationServices -Value $KerberosDelegationServices -Force
    $KerberosDelegationObjectItem | Add-Member -MemberType NoteProperty -Name DelegationType -Value $KerberosType -Force
    $KerberosDelegationObjectItem | Add-Member -MemberType NoteProperty -Name KerberosDelegationAllowedProtocols -Value $KerberosDelegationAllowedProtocols -Force

    [array]$KerberosDelegationArray += $KerberosDelegationObjectItem
 }

Write-Host ""
Write-Host "$Domain Domain Accounts with Kerberos Delegation:" -Fore Cyan
$KerberosDelegationArray | Sort DelegationType | Select DistinguishedName,DelegationType,Name,ServicePrincipalName | Format-Table -AutoSize
Write-Host ""
$KerberosDelegationReportFile = $ReportDir + "\TrimarcADChecks-KerberosDelegationReport-$Domain-$TimeVal.csv"
$KerberosDelegationArray | Sort DelegationType | Export-CSV $KerberosDelegationReportFile -NoTypeInformation
```

```
lab.trimarcresearch.com Domain Accounts with Kerberos Delegation:

DistinguishedName                                                                 DelegationType                       Name          ServicePrincipalName
-----------------                                                                 --------------                       ----          --------------------
CN=Delta38_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com               Constrained                          Delta38_svc   {MSSQLSvc/DeltaDB4:1433, MSSQLSvc/DeltaDB4
CN=Echo36_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                Constrained                          Echo36_svc    {MSSQLSvc/EchoDB35:5555}
CN=Alpha22_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com               Constrained                          Alpha22_svc   {MSSQLSvc/AlphaDB11:1433, MSSQLSvc/AlphaDB
CN=Gamma16_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com               Constrained with Protocol Transition Gamma16_svc   {MSSQLSvc/GammaDB37:Gamma}
CN=Gamma40_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com               Constrained with Protocol Transition Gamma40_svc   {MSSQLSvc/GammaDB23:Gamma}
CN=Delta35_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com               Constrained with Protocol Transition Delta35_svc   {MSSQLSvc/DeltaDB35:Delta}
CN=Beta29_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                Constrained with Protocol Transition Beta29_svc    {MSSQLSvc/BetaDB45:Beta}
CN=Delta26_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com               Constrained with Protocol Transition Delta26_svc   {MSSQLSvc/DeltaDB13:Delta}
CN=Echo20_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                Constrained with Protocol Transition Echo20_svc    {MSSQLSvc/EchoDB27:Echo}
CN=Echo28_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                Constrained with Protocol Transition Echo28_svc    {}
CN=Alpha41_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com               Constrained with Protocol Transition Alpha41_svc   {MSSQLSvc/AlphaDB32:5555, MSSQLSvc/AlphaDB
CN=Echo37_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                Constrained with Protocol Transition Echo37_svc    {MSSQLSvc/EchoDB30:Echo}
CN=Beta39_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                Constrained with Protocol Transition Beta39_svc    {MSSQLSvc/BetaDB7:3341}
CN=Beta48_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                Constrained with Protocol Transition Beta48_svc    {MSSQLSvc/BetaDB6:3341}
CN=Beta20_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                Constrained with Protocol Transition Beta20_svc    {MSSQLSvc/BetaDB44:3341}
CN=Beta23_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                Constrained with Protocol Transition Beta23_svc    {MSSQLSvc/BetaDB33:Beta}
CN=Echo27_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                Constrained with Protocol Transition Echo27_svc    {MSSQLSvc/EchoDB11:Echo}
CN=Delta11_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com               Constrained with Protocol Transition Delta11_svc   {MSSQLSvc/DeltaDB20:Delta}
CN=Beta37_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                Constrained with Protocol Transition Beta37_svc    {}
CN=Alpha35_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com               Constrained with Protocol Transition Alpha35_svc   {MSSQLSvc/AlphaDB39:Alpha}
CN=Gamma44_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com               Constrained with Protocol Transition Gamma44_svc   {MSSQLSvc/GammaDB8:Gamma}
CN=Beta1_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                 Constrained with Protocol Transition Beta1_svc     {MSSQLSvc/BetaDB26:3341}
CN=Gamma29_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com               Constrained with Protocol Transition Gamma29_svc   {MSSQLSvc/GammaDB40:5555}
CN=Echo15_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                Constrained with Protocol Transition Echo15_svc    {MSSQLSvc/EchoDB35:1433}
CN=Beta12_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                Constrained with Protocol Transition Beta12_svc    {MSSQLSvc/BetaDB39:1433}
CN=SVC-LAB-GMSA1,CN=Managed Service Accounts,DC=Lab,DC=trimarcresearch,DC=com     Resource-Based Constrained Delegation SVC-LAB-GMSA1 {MSSQLSvc/LCNSQL01.lab.trimarcresearch.com
CN=GammaDB8,OU=Servers,DC=Lab,DC=trimarcresearch,DC=com                           Resource-Based Constrained Delegation GammaDB8    {}
CN=Echo2_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                 Unconstrained                        Echo2_svc     {MSSQLSvc/EchoDB13:Echo}
CN=Beta33_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                Unconstrained                        Beta33_svc    {MSSQLSvc/BetaDB15:Beta}
CN=Delta29_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com               Unconstrained                        Delta29_svc   {MSSQLSvc/DeltaDB40:Delta}
CN=Beta6_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                 Unconstrained                        Beta6_svc     {}
CN=Beta17_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                Unconstrained                        Beta17_svc    {MSSQLSvc/BetaDB46:Beta}
CN=Alpha5_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                Unconstrained                        Alpha5_svc    {MSSQLSvc/AlphaDB30:1433}
CN=Delta39_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com               Unconstrained                        Delta39_svc   {MSSQLSvc/DeltaDB1:3341}
CN=Echo48_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                Unconstrained                        Echo48_svc    {MSSQLSvc/EchoDB23:Echo}
CN=Alpha31_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com               Unconstrained                        Alpha31_svc   {MSSQLSvc/AlphaDB17:1433}
CN=Alpha26_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com               Unconstrained                        Alpha26_svc   {MSSQLSvc/AlphaDB41:3341}
CN=Delta12_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com               Unconstrained                        Delta12_svc   {MSSQLSvc/DeltaDB1:1433}
CN=Gamma6_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com                Unconstrained                        Gamma6_svc    {MSSQLSvc/GammaDB21:3341}
CN=Alpha15_svc,OU=Service Accounts,DC=Lab,DC=trimarcresearch,DC=com               Unconstrained                        Alpha15_svc   {MSSQLSvc/AlphaDB21:Alpha}
```

GPO Permissions: Review Owners

Group Policy Objects (GPO) have owners which are able to change permisisons. Typically when an account creates a GPO, the account that created it is delegate modify rights and I confiugred as the owner.

Trimarc Recommendation:

> Ensure all GPO owners are set to Domain Admins or Enterprise Admins, especially GPOs linked to the domain root and Domain Controllers OU.

PowerShell Sample Code & Results:

```
## Get GPO Owners
[Array]$DomainGPOs = Get-GPO -All -Domain $Domain
$DomainGPOs | Select DisplayName,Owner | Format-Table -AutoSize
$DomainGPODataReportFile = $ReportDir + "\TrimarcADChecks-DomainGPOData-$Domain-$TimeVal.csv"
$DomainGPOs | Out-File $DomainGPODataReportFile
```

```
DisplayName                    Owner
-----------                    -----
Atlanta Site GPO               TrimarcRD\Domain Admins
Default Domain Policy          TrimarcRD\Domain Admins
Default Domain Controllers Policy TrimarcRD\Domain Admins
DC Audit Policy                TrimarcRD\Domain Admins
```

Review Domain Permissions

Domain permissions should be reviewed to ensure that configuration is appropriate. Security of the domain often depends on proper domain permissions.

Trimarc Recommendation:

- Review domain root permissions with special attention paid to the following:Any non-default admin groups (Domain Admins, domain Administrators, Enterprise Admins, etc) with GenericAll (Full Control), WriteDACL (change permissions), write property (modify), and ExtendedRights.
- Ensure that the domain root owner is configured to Domain Admins or Enterprise Admins.

PowerShell Sample Code & Results:

```
## Get Domain Permissions
Write-Output "Gathering Domain Permissions for $Domain"
$ForestDomainObjectData = Get-ADObject $ADDomainInfo.DistinguishedName -Properties *
$ForestDomainObjectSecurityData = $ForestDomainObjectData.nTSecurityDescriptor.Access
$ForestDomainObjectPermissions = @()
ForEach ($ForestDomainObjectSecurityDataItem in $ForestDomainObjectSecurityData)
{
    $ObjectTypeName = Get-NameForGUID $ForestDomainObjectSecurityDataItem.ObjectType
    $InheritedObjectTypeName = Get-NameForGUID $ForestDomainObjectSecurityDataItem.InheritedObjectType

    $ForestDomainObjectSecurityDataItem | Add-Member -MemberType NoteProperty -Name Domain -Value $Domain -Force
    $ForestDomainObjectSecurityDataItem | Add-Member -MemberType NoteProperty -Name ObjectTypeName -Value $ObjectTypeName -Force
    $ForestDomainObjectSecurityDataItem | Add-Member -MemberType NoteProperty -Name InheritedObjectTypeName -Value $InheritedObjectTypeName -Force

    [array]$ForestDomainObjectPermissions += $ForestDomainObjectSecurityDataItem
}
$ForestDomainObjectPermissionFile = $ReportDir + "\TrimarcADChecks-DomainRootPermissionReport-$Domain-$TimeVal.csv"
$ForestDomainObjectPermissions | Sort IdentityReference | Select IdentityReference,ActiveDirectoryRights,InheritedObjectTypeName,ObjectTypeName,`
InheritanceType,ObjectFlags,AccessControlType,IsInherited,InheritanceFlags,PropagationFlags,ObjectType,InheritedObjectType | `
Export-CSV $ForestDomainObjectPermissionFile -NoTypeInformation
```

The PowerShell script outputs to a CSV file which Excel can open. Use the Data, Filter feature to filter known grouprs from the IdentityReference column (Administrators, Domain Admins, Enterprise Admins, etc). This screenshot shows a sample environment with some IdentityReference filtering.

IdentityReference	ActiveDirectoryRights	InheritedObjectTypeNam	ObjectTypeName	AccessControlTyp	InheritanceType	ObjectFlags	IsInherited	InheritanceFlags	PropagationFlags
S-1-5-21-1362129023-2533104153-2294267348-527	ReadProperty, WriteProperty	All	msDS-KeyCredentialLink	Allow	All	ObjectAceTypePresent	FALSE	ContainerInherit	None
TRDLab\AADC_Svc	ExtendedRight	All	DS-Replication-Get-Changes	Allow	All	ObjectAceTypePresent	FALSE	ContainerInherit	None
TRDLab\AADC_Svc	ExtendedRight	All	DS-Replication-Get-Changes-All	Allow	All	ObjectAceTypePresent	FALSE	ContainerInherit	None
TRDLab\AADC_Svc	ReadProperty, WriteProperty, GenericExecute	All	All	Allow	All	None	FALSE	ContainerInherit	None
TRDLab\Cloneable Domain Controllers	ExtendedRight	All	DS-Clone-Domain-Controller	Allow	None	ObjectAceTypePresent	FALSE	None	None
TRDLab\Domain Computers	GenericAll	All	All	Allow	None	None	FALSE	None	None
TRDLab\Key Admins	ReadProperty, WriteProperty	All	msDS-KeyCredentialLink	Allow	All	ObjectAceTypePresent	FALSE	ContainerInherit	None
TRDLab\MSOL_trd326848506	ReadProperty, WriteProperty, GenericExecute	All	All	Allow	All	None	FALSE	ContainerInherit	None
TRDLab\MSOL_trd326848506	ExtendedRight	All	DS-Replication-Get-Changes	Allow	All	ObjectAceTypePresent	FALSE	ContainerInherit	None
TRDLab\MSOL_trd326848506	ExtendedRight	All	DS-Replication-Get-Changes-All	Allow	All	ObjectAceTypePresent	FALSE	ContainerInherit	None
TRDLab\SharepointSVC	ReadProperty, WriteProperty, GenericExecute	All	All	Allow	All	None	FALSE	ContainerInherit	None
TRDLab\SharepointSVC	ExtendedRight	All	DS-Replication-Get-Changes	Allow	All	ObjectAceTypePresent	FALSE	ContainerInherit	None
TrimarcRD\Enterprise Admins	GenericAll	All	All	Allow	All	None	FALSE	ContainerInherit	None
TrimarcRD\Enterprise Read-only Domain Controllers	ExtendedRight	All	DS-Replication-Get-Changes	Allow	None	ObjectAceTypePresent	FALSE	None	None
TrimarcRD\Incoming Forest Trust Builders	ExtendedRight	All	Create-Inbound-Forest-Trust	Allow	None	ObjectAceTypePresent	FALSE	None	None

The following screenshot identify some potentially concerning permissions, including the DS-Replication-Get-Changes & DS-Replication-Get-Changes ExtendedRights.

IdentityReference	ActiveDirectoryRights	InheritedObjectTypeNam	ObjectTypeName
S-1-5-21-1362129023-2533104153-2294267348-527	ReadProperty, WriteProperty	All	msDS-KeyCredentialLink
TRDLab\AADC_Svc	ExtendedRight	All	DS-Replication-Get-Changes
TRDLab\AADC_Svc	ExtendedRight	All	DS-Replication-Get-Changes-All
TRDLab\AADC_Svc	ReadProperty, WriteProperty, GenericExecute	All	All
TRDLab\Cloneable Domain Controllers	ExtendedRight	All	DS-Clone-Domain-Controller
TRDLab\Domain Computers	GenericAll	All	All
TRDLab\Key Admins	ReadProperty, WriteProperty	All	msDS-KeyCredentialLink
TRDLab\MSOL_trd326848506	ReadProperty, WriteProperty, GenericExecute	All	All
TRDLab\MSOL_trd326848506	ExtendedRight	All	DS-Replication-Get-Changes
TRDLab\MSOL_trd326848506	ExtendedRight	All	DS-Replication-Get-Changes-All

The following screenshot identifies that the Domain Computers group has GenericAll (Full Control) rights on All objects.

IdentityReference	ActiveDirectoryRights	InheritedObjectTypeNam	ObjectTypeName	AccessControlTyp
S-1-5-21-1362129023-2533104153-2294267348-527	ReadProperty, WriteProperty	All	msDS-KeyCredentialLink	Allow
TRDLab\AADC_Svc	ExtendedRight	All	DS-Replication-Get-Changes	Allow
TRDLab\AADC_Svc	ExtendedRight	All	DS-Replication-Get-Changes-All	Allow
TRDLab\AADC_Svc	ReadProperty, WriteProperty, GenericExecute	All	All	Allow
TRDLab\Cloneable Domain Controllers	ExtendedRight	All	DS-Clone-Domain-Controller	Allow
TRDLab\Domain Computers	GenericAll	All	All	Allow

Domain Controllers Running Old Versions

Domain Controllers must be running Microsoft supported Windows versions. You can run all Windows Server 2012 R2 DCs even with older Windows Server versions in the domain (note that some testing to ensure Windows 2003/XP works with the new DCs, especially for SMB shares).

All DCs should be running a minimum of Windows Server 2012 R2, preferably 2016/2019. If all DCs are not running 2012 R2 with 2012R2 DFL, Protected Users group doesn't have full domain protection.

Trimarc Recommendation:

> All DCs should be running a minimum of Windows Server 2012 R2, preferably 2016/2019.

PowerShell Sample Code & Results:

```
## Get Domain Controllers
$DomainDCs = Get-ADDomainController -filter *
Write-Host "$ADDomainName AD Forest Domain Controllers and OS Version:  `n" -Fore Cyan
$DomainDCs | Select HostName,OperatingSystem | Format-Table -AutoSize
```

```
Lab.trimarcresearch.com AD Forest Domain Controllers and OS Version:

HostName                          OperatingSystem
--------                          ---------------
TRDC11.Lab.trimarcresearch.com Windows Server 2019 Datacenter
```

AD Forest Functional Level / Domain Functional Level Older than Domain Controller Operating System

The Domain Functional Level (DFL) provides additional AD security features up to Windows Server 2016. Forest & Domain Functional Levels have less emphasis in recent versions. All AD Forests should match the DC version (after all DCs in the AD forest are running the same version). This can be configured after the DCs are updated to a recent OS and have had time to "bake in" over time.

Trimarc Recommendation:

> Ensure all Domain Controllers are updated to Windwos Server 2012 R2 (or newer) and set DFL/FFL to 2012 R2 (or newer).

PowerShell Sample Code & Results:

```
## Identify AD FFL/DFL
## Get AD Forest & Domain Info
$ADFFL = (Get-ADForest).ForestMode
$ADDFL = (Get-ADDomain $Domain).DomainMode
Write-Host "The AD Forest Functional Level is $ADFFL `n" -Fore Cyan
Write-Host "The AD Domain Functional Level ($Domain) is $ADFFL `n" -Fore Cyan
```

```
The AD Forest Functional Level is Windows2008Forest

The AD Domain Functional Level (lab.trimarcresearch.com) is Windows2008Forest
```

Tools for Reviewing AD Note that Trimarc provides these as references and does not necessarily endorse any tool listed

Ping Castle

Get Active Directory Security at 80% in 20% of the time

Active directory is quickly becoming a critical failure point in any big sized company, as it is both complex and costly to secure..

https://www.pingcastle.com/

ADCollector https://hakin9.org/adcollector-a-lightweight-tool-to-quickly-extract-valuable-information-from-the-active-directory-environment-for-both-attacking-and-defending/

ADRecon: Active Directory Recon

ADRecon is a tool which extracts and combines various artefacts (as highlighted below) out of an AD environment. The information can be presented in a specially formatted Microsoft Excel report that includes summary views with metrics to facilitate analysis and provide a holistic picture of the current state of the target AD environment.

The tool is useful to various classes of security professionals like auditors, DFIR, students, administrators, etc. It can also be an invaluable post-exploitation tool for a penetration tester.

It can be run from any workstation that is connected to the environment, even hosts that are not domain members. Furthermore, the tool can be executed in the context of a non-privileged (i.e. standard domain user) account. Fine Grained Password Policy, LAPS and BitLocker may require Privileged user accounts. The tool will use Microsoft Remote Server Administration Tools (RSAT) if available, otherwise it will communicate with the Domain Controller using LDAP.

The following information is gathered by the tool:

- Forest;
- Domain;
- Trusts;
- Sites;
- Subnets;
- Default and Fine Grained Password Policy (if implemented);
- Domain Controllers, SMB versions, whether SMB Signing is supported and FSMO roles;
- Users and their attributes;
- Service Principal Names (SPNs);
- Groups and memberships;
- Organizational Units (OUs);
- GroupPolicy objects and gPLink details;
- DNS Zones and Records;
- Printers;
- Computers and their attributes;
- PasswordAttributes (Experimental);
- LAPS passwords (if implemented);
- BitLocker Recovery Keys (if implemented);
- ACLs (DACLs and SACLs) for the Domain, OUs, Root Containers, GPO, Users, Computers and Groups objects;
- GPOReport (requires RSAT);
- Kerberoast (not included in the default collection method); and
- Domain accounts used for service accounts (requires privileged account and not included in the default collection method).

ADRecon https://github.com/sense-of-security/ADRecon

ADRecon: Active Directory Recon

Follow @ad_recon 245

ADRecon is a tool which extracts and combines various artefacts (as highlighted below) out of an AD environment. The information can be presented in a specially formatted Microsoft Excel report that includes summary views with metrics to facilitate analysis and provide a holistic picture of the current state of the target AD environment.

The tool is useful to various classes of security professionals like auditors, DFIR, students, administrators, etc. It can also be an invaluable post-exploitation tool for a penetration tester.

It can be run from any workstation that is connected to the environment, even hosts that are not domain members. Furthermore, the tool can be executed in the context of a non-privileged (i.e. standard domain user) account. Fine Grained Password Policy, LAPS and BitLocker may require Privileged user accounts. The tool will use Microsoft Remote Server Administration Tools (RSAT) if available, otherwise it will communicate with the Domain Controller using LDAP.

The following information is gathered by the tool:

- Forest;
- Domain;
- Trusts;
- Sites;
- Subnets;
- Default and Fine Grained Password Policy (if implemented);
- Domain Controllers, SMB versions, whether SMB Signing is supported and FSMO roles;
- Users and their attributes;
- Service Principal Names (SPNs);
- Groups and memberships;
- Organizational Units (OUs);
- GroupPolicy objects and gPLink details;
- DNS Zones and Records;
- Printers;
- Computers and their attributes;
- PasswordAttributes (Experimental);
- LAPS passwords (if implemented);
- BitLocker Recovery Keys (if implemented);
- ACLs (DACLs and SACLs) for the Domain, OUs, Root Containers, GPO, Users, Computers and Groups objects;
- GPOReport (requires RSAT);
- Kerberoast (not included in the default collection method); and
- Domain accounts used for service accounts (requires privileged account and not included in the default collection method).

By Sean Metcalf

#Powershell #TrimarcTools #ADSecurityReview #Scripts