How Attackers Use Kerberos Silver Tickets to Exploit Systems

Real-World Example of How Active Directory Can Be Compromised (RSA Conference Presentation)

NOV 22 2015

Dump Clear-Text Passwords for All Admins in the Domain Using Mimikatz DCSync

By Sean Metcalf in ActiveDirectorySecurity, Microsoft Security, Technical Reference

The two key goals of any attack is access and persistence. This post covers elements of each.
In a post-exploitation scenario where the attacker has compromised the domain or an account with delegated rights, it's possible to dump the clear-text passwords of admins without being a Domain Admin*. This method requires the Active Directory Domain Functional Level (DFL) to be Windows Server 2008 or higher and a patient attacker (as well as appropriate rights).
* depending on existing AD delegation or attacker configured delegation. Required rights are described at the end of this post

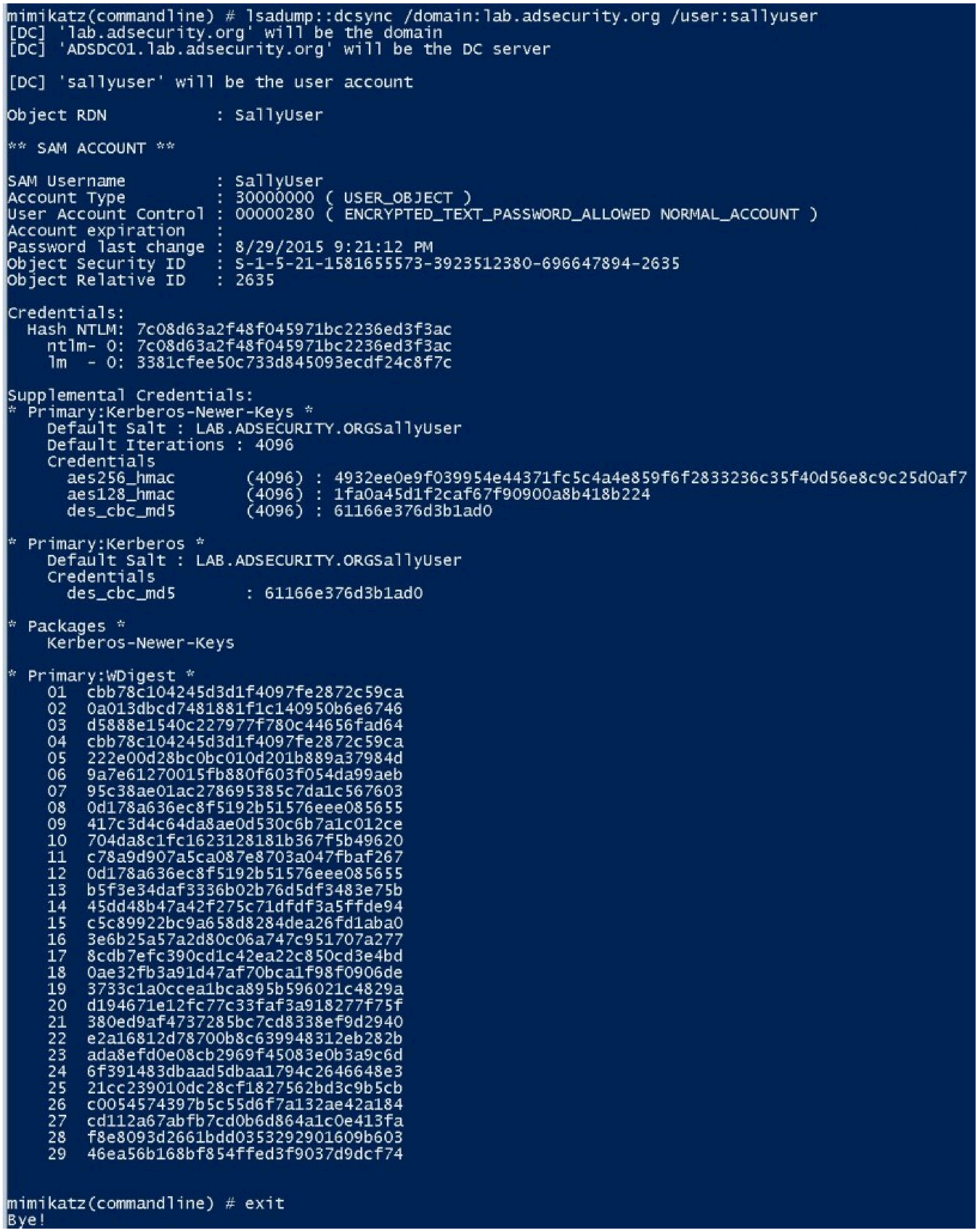
I spoke about some of this information at several security conferences in 2015 (BSides, Shakacon, Black Hat, DEF CON, & DerbyCon).

Mimikatz DCSync Capability:
The Mimikatz DCSync capability is pretty amazing from an offensive perspective since it provides the capability to pull domain account password data remotely from the Domain Controller. The account that runs DCSync needs to have the proper rights since DCSync pulls account data through the standard

Domain Controller replication API. Prior to this Mimikatz capability, added in late August, dumping all or selective account password hashes from Active Directory required code execution on the Domain Controller, pulling the AD database (ntds.dit) and dumping the contents, or running something like Invoke-Mimikatz over PowerShell Remoting.

With DCSync, and the proper rights, the attacker can pull useful password data on the account including (potentially):

- SAM Account name
- Account Type
- User Account Control options
- Account Expiration
- Password last set date
- Security ID (SID)
- RID
- Current and previous (password history) NTLM password hashes which shows the password history, at least in NTLM hash format.
- Current and previous LM password hashes which shows the password history in LM hash format.
- Clear text password (requires reversible encryption)



```
mimikatz(commandline) # lsadump::dcsync /domain:lab.adsecurity.org /user:sallyuser
[DC] 'lab.adsecurity.org' will be the domain
[DC] 'ADSDC01.lab.adsecurity.org' will be the DC server

[DC] 'sallyuser' will be the user account

Object RDN           : SallyUser

** SAM ACCOUNT **

SAM Username         : SallyUser
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000280 ( ENCRYPTED_TEXT_PASSWORD_ALLOWED NORMAL_ACCOUNT )
Account expiration   :
Password last change : 8/29/2015 9:21:12 PM
Object Security ID   : S-1-5-21-1581655573-3923512380-696647894-2635
Object Relative ID   : 2635

Credentials:
  Hash NTLM: 7c08d63a2f48f045971bc2236ed3f3ac
    ntlm- 0: 7c08d63a2f48f045971bc2236ed3f3ac
    lm  - 0: 3381cfee50c733d845093ecdf24c8f7c

Supplemental Credentials:
* Primary:Kerberos-Newer-Keys *
    Default Salt : LAB.ADSECURITY.ORGSallyUser
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 4932ee0e9f039954e44371fc5c4a4e859f6f2833236c35f40d56e8c9c25d0af7
      aes128_hmac       (4096) : 1fa0a45d1f2caf67f90900a8b418b224
      des_cbc_md5       (4096) : 61166e376d3b1ad0

* Primary:Kerberos *
    Default Salt : LAB.ADSECURITY.ORGSallyUser
    Credentials
      des_cbc_md5       : 61166e376d3b1ad0

* Packages *
    Kerberos-Newer-Keys

* Primary:WDigest *
    01  cbb78c104245d3d1f4097fe2872c59ca
    02  0a013dbcd7481881f1c140950b6e6746
    03  d5888e1540c227977f780c44656fad64
    04  cbb78c104245d3d1f4097fe2872c59ca
    05  222e00d28bc0bc010d201b889a37984d
    06  9a7e61270015fb880f603f054da99aeb
    07  95c38ae01ac278695385c7da1c567603
    08  0d178a636ec8f5192b51576eee085655
    09  417c3d4c64da8ae0d530c6b7a1c012ce
    10  704da8c1fc1623128181b367f5b49620
    11  c78a9d907a5ca087e8703a047fbaf267
    12  0d178a636ec8f5192b51576eee085655
    13  b5f3e34daf3336b02b76d5df3483e75b
    14  45dd48b47a42f275c71dfdf3a5ffde94
    15  c5c89922bc9a658d8284dea26fd1aba0
    16  3e6b25a57a2d80c06a747c951707a277
    17  8cdb7efc390cd1c42ea22c850cd3e4bd
    18  0ae32fb3a91d47af70bca1f98f0906de
    19  3733c1a0ccea1bca895b596021c4829a
    20  d194671e12fc77c33faf3a918277f75f
    21  380ed9af4737285bc7cd8338ef9d2940
    22  e2a16812d78700b8c639948312eb282b
    23  ada8efd0e08cb2969f45083e0b3a9c6d
    24  6f391483dbaad5dbaa1794c2646648e3
    25  21cc239010dc28cf1827562bd3c9b5cb
    26  c0054574397b5c55d6f7a132ae42a184
    27  cd112a67abfb7cd0b6d864a1c0e413fa
    28  f8e8093d2661bdd0353292901609b603
    29  46ea56b168bf854ffed3f9037d9dcf74


mimikatz(commandline) # exit
Bye!
```

User Account Password Encrypted, Not Hashed:

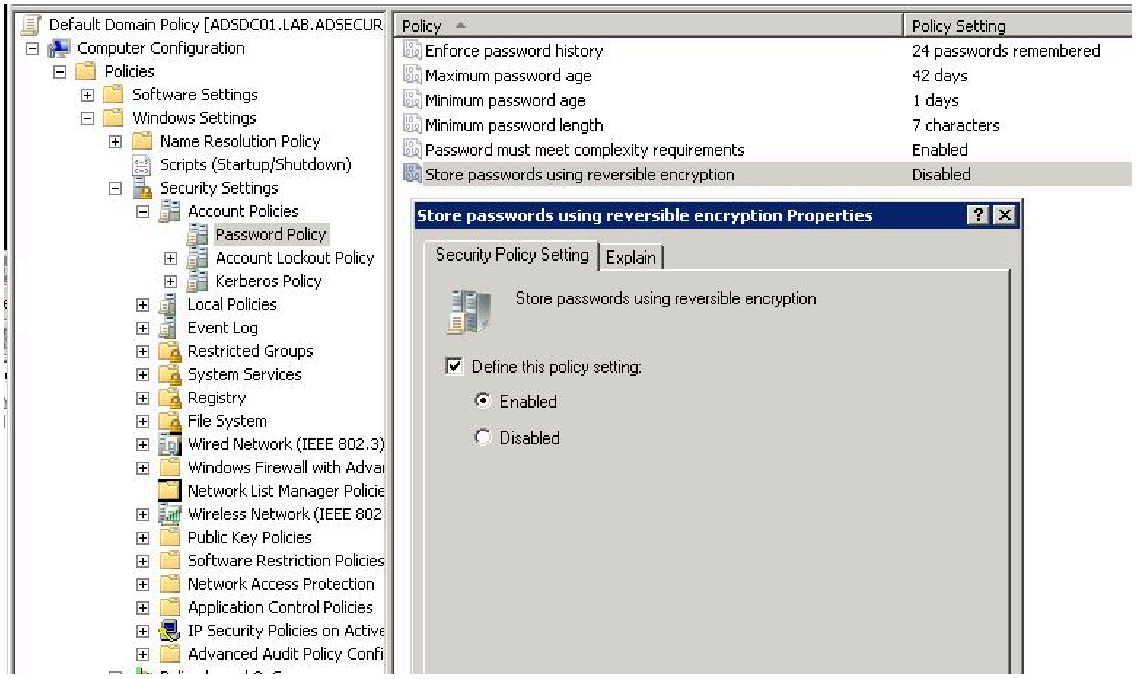
There's a legacy feature for Active Directory accounts called "reversible encryption". Normally a user's password is hashed using the NT one-way function to create the NTLM password hash. The NTLM password hash can't be reversed it would have to be cracked, meaning that a tool would have to be used to create passwords and perform the NT hash function to get the NTLM password hash. If the user's password hash matches the generated one, then the password was successfully guessed (known as brute force password guessing). If reversible encryption is enabled, then the user's password is stored using encryption which means the encrypted data can be reversed back to the user's password. The password stored with reversible encryption is not a hash since a function can be called to get back to the original clear-text password.

The next question is: once the account is enabled for reversible encryption, is the clear-text password for the account immediately available?
No, the Domain Controllers only stores the user's NTLM password hash, so the clear-text password is not available. If reversible encryption is enabled on the account and the user changes the password after this configuration is set, then the clear-text password is saved in the Active Directory database. This occurs when the user sends the clear-text password to the DC when changing the password (occurs over encrypted RPC).

There's a few different ways to set an account to use "reversible encryption" which effectively has a clear-text password in the Active Directory database (ntds.dit) on all the Domain Controllers in the domain.

1. Modify the domain-linked Group Policy with the highest precedence that configures the domain password policy so it enables reversible encryption. Modifying this GPO is much more likely to be noticed. The policy option is Computer Configuration, Policies, Windows Settings, Security Settings, Account Policies, Password Policy: "Store passwords using reversible encryption". Check the box and set to Enabled and all user accounts will have their passwords stored using reversible encryption

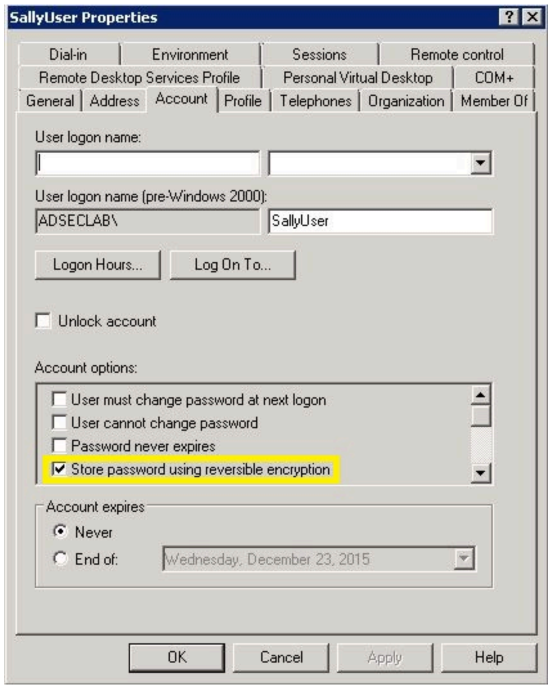
after the next password change.


Default Domain Policy [ADSDC01.LAB.ADSECUR
Computer Configuration
Policies
Software Settings
Windows Settings
Name Resolution Policy
Scripts (Startup/Shutdown)
Security Settings
Account Policies
Password Policy
Account Lockout Policy
Kerberos Policy
Local Policies
Event Log
Restricted Groups
System Services
Registry
File System
Wired Network (IEEE 802.3)
Windows Firewall with Advai
Network List Manager Policie
Wireless Network (IEEE 802
Public Key Policies
Software Restriction Policies
Network Access Protection
Application Control Policies
IP Security Policies on Active
Advanced Audit Policy Confi

Policy	Policy Setting
Enforce password history	24 passwords remembered
Maximum password age	42 days
Minimum password age	1 days
Minimum password length	7 characters
Password must meet complexity requirements	Enabled
Store passwords using reversible encryption	Disabled

Store passwords using reversible encryption Properties
Security Policy Setting | Explain
Store passwords using reversible encryption
☑ Define this policy setting:
[✓ Enabled] Disabled

2. Modify the individual user accounts to enable “AllowReversiblePasswordEncryption” by checking the box next to “Store password using reversible encryption”. This is not very obvious and most organizations probably aren’t looking at this. An example of how to do this is shown in the “Defense” section at the end.


SallyUser Properties
Dial-in | Environment | Sessions | Remote control | Remote Desktop Services Profile | Personal Virtual Desktop | COM+ | General | Address | Account | Profile | Telephones | Organization | Member Of
User logon name:
User logon name (pre-Windows 2000): ADSECLAB\ SallyUser
Logon Hours... Log On To...
☐ Unlock account
Account options:
☐ User must change password at next logon
☐ User cannot change password
☐ Password never expires
☑ Store password using reversible encryption
Account expires
[✓ Never] End of: Wednesday, December 23, 2015
OK Cancel Apply Help

3. If the Domain Functional Level is set to “Windows Server 2008” or higher, a new feature called “Fine-Grained Password Policy” is available to provide a wide-variety of password policies that can be applied to users or groups (not OUs).

I like the third option the best since it's not obvious – no modifying a domain root-linked GPO or an admin account directly. The only way to track this is to monitor group membership of domain-level admin accounts (and associate with existing Fine Grained Password Policies). When group membership is actually monitored, the opposite is true: monitor the members of specific privileged groups, not the groups privileged users are members of. While Microsoft made Fine-Grained Password Policies available starting with Windows Server 2008 (DFL), the Active Directory Administrative Center (ADAC) (which most admins still don't use) wasn't updated to support FGPP administration until Windows Server 2012. This means an attacker could configure new FGPP and they probably wouldn't be noticed, especially if called something benign ("Exchange users", "SharePoint Users", etc). Enabling "Advanced Features" from the "View" menu option in Active Directory Users and Computers and then browsing down to System, Password Settings Container (CN=Password Settings Container,CN=System,DC=DOMAIN,DC=COM) will typically display any domain FGPP objects. Note that if "Advanced Features" is not enabled, the System container is not visible.

The Scenario:

Compromise a Domain Admin account or account with delegated rights to create Fine-Grained Password Policies ("Create msDS-PasswordSettings" objects which exist in CN=Password Settings Container,CN=System,DC=DOMAIN,DC=COM).
Note that the PowerShell cmdlets in this post are from the Active Directory module, although these are not strictly required for these operations, it makes it easier. HarmJ0y's PowerView provides a lot of AD functionality in PowerShell without the Active Directory module.

Create a new group called something like "SharePont users" (misspelled on purpose so it won't appear when admins search for SharePoint groups) and add all domain accounts with the attribute "AdminCount" set to 1. Accounts with the AdminCount attribute set to 1 are members of certain privileged domain groups.

Once the group is created, find all AD domain accounts with AdminCount set to 1 and add them to the new group "SharePontUsers".

Create a Fine-Grained Password Policy (FGPP) called "SharePoint Policy" and apply to "SharePont users".

The settings on this Fine-Grained Password Policy are interesting:

- ComplexityEnabled: False
  This determines whether the password complexity rules are required: 3 out of 4 categories including upper alpha, lower alpha, numeric, and symbols.
  This should be set to TRUE
- LokoutDuration: 00:00:00
  This determines how long the account should be locked out after the bad password attempt threshold is triggered. 00:00:00 means the account is never locked out.
  This should be set to a number if account lockout is desired.
- LockoutObservationWindow: 00:00:00
  This determines the period during which the lockout threshold is monitored. If set to 00:05:00, the account lockout observation window is 5 minutes, so if the lockout threshold is set to 5 and there are 5 bad password attempts in less than 5 minutes, the account is locked out.
- LockoutThreshold: 0
  This combined with the LockoutObservationWindow determines when an account is locked out.
- MaxPasswordAge: 00:00:00
  This determines how long a password may be used by the user. Typically this is set to a value between 90:00:00 and 180:00:00 to ensure that users change their passwords at least once a year..
- MinPasswordAge: 00:00:00
  This determines how long a new password must be kept before changing. Setting this value to 00:00:00 means the user can change a password as frequently as they desire.

- MinPasswordLength: 0
  This determines the minimum length of the password which effectively sets the standard password length for all users in the domain.
- PasswordHistoryCount: 0
  This determines how many passwords the user must have before re-using a previous password. Typically set to 20 – 30.
- Precedence: 1
  This setting is specific to FGPPs since it determines which FGPP takes precedence if there are multiple FGPP applied to an account or accounts.
- ReversibleEncryptionEnabled: True
  This should be set to False. TThis sets all accounts associated with the FGPP (in this case, admin accounts) to store their passwords in clear text on the DCs at next password change.

Note that the first item, "AppliesTo" is blank. Since we haven't associated the new policy with an account or group, this is empty.
On to the next step: associate the new FGPP with the group we created.

After associating the FGPP with the group, we re-run Get-ADFineGrainedPasswordPolicy to ensure it is properly associated.

The new FGPP effectively zeros out all standard domain password security settings usually configured by the Default Domain Policy at the domain root. should be set to False. ReversibleEncryptionEnabled sets all accounts associated with the FGPP (in this case, admin accounts) to store their passwords in clear text on the DCs at next password change. The clear-text password can be pulled using Mimikatz's DCSync. Of course, initiating a password change "this password has expired and must be set" could be fun as well.

Granted, the attacker can get the hashes using Mimikatz's DCSync feature, so why bother with clear-text passwords? Why not? 😳
Clear-text passwords provide insight into password versioning and are useful for RDP access as well as other "normal" user activity while use of password hashes may be flagged as "suspicious".

Note: The fine-grained password policy could be configured directly on the user accounts, but that's probably a little obvious.

After the FGPP is in place and the targeted user has changed his/her password, running DCSync shows the clear-text password.

Rights Required:

If an attacker gains access to an account in the Domain administrators or Domain Admins group, they have the necessary rights for this configuration. Otherwise, here are the required rights:

Action	Rights Required
Create Group	"Create Group object" in OU/domain.
Modify Group Membership	"Read/Write Members" on group or OU/domain.
Modify User Account (enable "AllowReversiblePasswordEncryption")	"Write all properties" and "All Validated Writes" on user objects in OU/domain.
Create Fine-Grained Password Policy (FGPP)	"Create msDS-PasswordSettings objects" in the Password Settings Container (in System).
Modify Fine-Grained Password Policy (FGPP)	"Write all properties" on msDS-PasswordSettings objects in the Password Settings Container (in System).
Use Mimikatz DCSync to pull password data for account.	"Replicating Directory Changes" and "Replicating Directory Changes All" (and sometimes ""Replicating Directory Changes in Filtered Set"

Defense:

- Monitor Fine-Grained Password Policies and the applied settings as well as who they apply to.

- Monitor privileged account group membership.

- Regularly audit user accounts for potential issues: reversible encryption, password never expires, admin accounts that allow delegation, etc.

References:

- Fine-Grained Password Policies
- Mimikatz DCSync Usage, Exploitation, and Detection
- PowerView

(Visited 71,193 times, 5 visits today)

Active Directory Administrative Center, ActiveDirectoryModule, ADAC, Add-ADFineGrainedPasswordPolicy, AdminCount, AllowReversiblePasswordEncryption, ClearTextPassword, DCSync, DCSyncRights, DomainPasswordPolicy, DumpClearTextPassword, FGPP, FineGrainedPasswordPolicies, Get-ADFineGrainedPasswordPolicy, GroupPolicy, mimikatz, msDS-PasswordSettings, ntds.dit, Password Settings Container, PowerShell, PowerView, ReversibleEncryptionEnabled

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

3 comments

Brad on *November 23, 2015 at 1:38 pm* #

I notice in the DCSync output both NTLM and lm hashes are listed. Does this mean your lab domain is storing LM credentials for users in addition to NTLM hashes?

Sean Metcalf on *November 23, 2015 at 2:50 pm* #
Author

That's correct. I have several lab environments with different levels of security configuration. This one is the default lab environment where I test most things first. It shows how keeping the default settings may leave an enterprise more open to compromise. Note that unless KB2871997 is installed on Windows 7/8/008R2/2012, the LM hash for users that logon is generated and stored in memory (LSASS), at least for passwords that are 14 characters or less. Furthermore, unless the registry key HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest\UseLog is set to 0 (set to 1 by default), the "clear-text" password is stored in LSASS as well (https://adsecurity.org/?p=559).

Brad on *December 1, 2015 at 12:36 pm* #

Hmm, but I thought the default in Server 2008/Vista or later environments was to set "Do not store LAN Manager hash value on next password change" to enabled, thus disabling the storage of LM hashes in AD. I tested in my environment by creating a user in ADUC then resetting its password via Powershell. Dcsync returns me LM hashes, however I don't think they are valid. The passwords I set both start with "Password" which should generate two LM hashes that both have an identical first half, but they are completely different. They also don't match what online LM hash generators says they should be. So not sure if this is a dcsync bug or if a junk value gets stored there or what.

Comments have been disabled.

Attack Methods for Gaining Domain Admin Rights in…

Kerberos & KRBTGT: Active Directory's…

Finding Passwords in SYSVOL & Exploiting Group…

Securing Domain Controllers to Improve Active…

Securing Windows Workstations: Developing a Secure Baseline

Detecting Kerberoasting Activity

Mimikatz DCSync Usage, Exploitation, and Detection

Scanning for Active Directory Privileges &…

Microsoft LAPS Security & Active Directory LAPS…

CATEGORIES

ActiveDirectorySecurity

Apple Security

Cloud Security

Continuing Education

Entertainment

Exploit

Hacking

Hardware Security

Hypervisor Security

Linux/Unix Security

Malware

Microsoft Security

Mitigation

Network/System Security

PowerShell

RealWorld

Security

Security Conference Presentation/Video

Security Recommendation

Technical Article

Technical Reading

Technical Reference

TheCloud

Vulnerability

TAGS

ActiveDirectory Active Directory Active Directory Security ActiveDirectorySecurity ADReading AD Security ADSecurity Azure AzureAD DCSync DomainController GoldenTicket GroupPolicy HyperV Invoke-Mimikatz KB3011780 KDC Kerberos KerberosHacking KRBTGT LAPS LSASS MCM MicrosoftEMET MicrosoftWindows mimikatz MS14068 PassTheHash PowerShell PowerShellCode Pow erShellHacking Pow erShellv5 PowerSploit Presentation Security SilverTicket SneakyADPersistence SPN TGS TGT Window s7 Windows10 WindowsServer2008R2 WindowsServer2012 WindowsServer2012R2

Search

RECENT POSTS

BSides Dublin – The Current State of Microsoft Identity Security: Common Security Issues and Misconfigurations – Sean Metcalf

DEFCON 2017: Transcript – Hacking the Cloud

Detecting the Elusive: Active Directory Threat Hunting

Detecting Kerberoasting Activity

Detecting Password Spraying with Security Event Auditing

RECENT COMMENTS

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

Sean Metcalf on Securing Microsoft Active Directory Federation Server (ADFS)

Brad on Securing Microsoft Active Directory Federation Server (ADFS)

Joonas on Gathering AD Data with the Active Directory PowerShell Module

Sean Metcalf on Gathering AD Data with the Active Directory PowerShell Module

ARCHIVES

June 2024

May 2024

May 2020

January 2020

August 2019

March 2019

February 2019

October 2018

CATEGORIES

ActiveDirectorySecurity

Apple Security

Cloud Security

Continuing Education

Entertainment

Exploit

Hacking

Hardware Security

Hypervisor Security

Linux/Unix Security

Malware

Microsoft Security

Mitigation

Network/System Security

PowerShell

RealWorld

Security

Security Conference Presentation/Video

Security Recommendation

Technical Article

Technical Reading

Technical Reference

TheCloud

Vulnerability

META

Log in

Entries feed

Comments feed

WordPress.org

COPYRIGHT