Implementing Privileged Access Workstation – part 5

michaelfirsov.wordpress.com/implementing-privileged-access-workstation-part-5

July 15, 2021

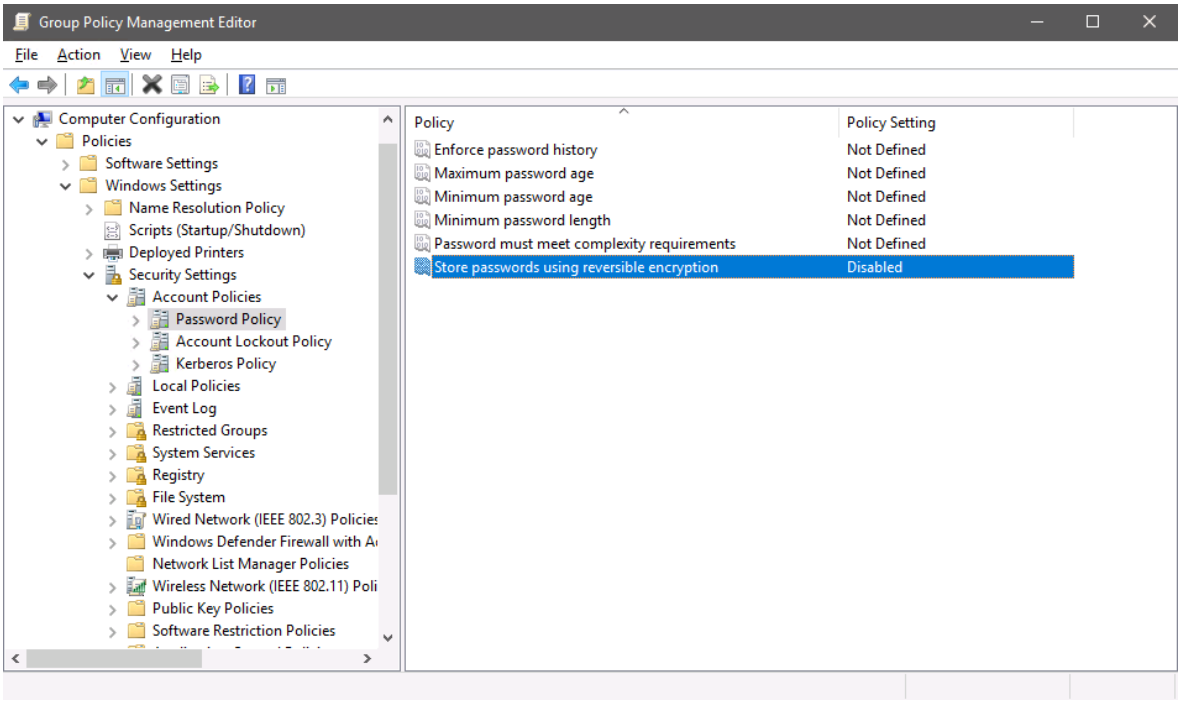
Mitigating Pass-The-Hash attacks

The previous parts of this blog post series were also devoted to mitigating the threat of credential theft (especially part 3 and part 4) – this post will just add several additional well-known techniques that can be used along with the ones described in part 3/part4.

I'm going to create one more GPO named **AntiPtH** with the following settings:

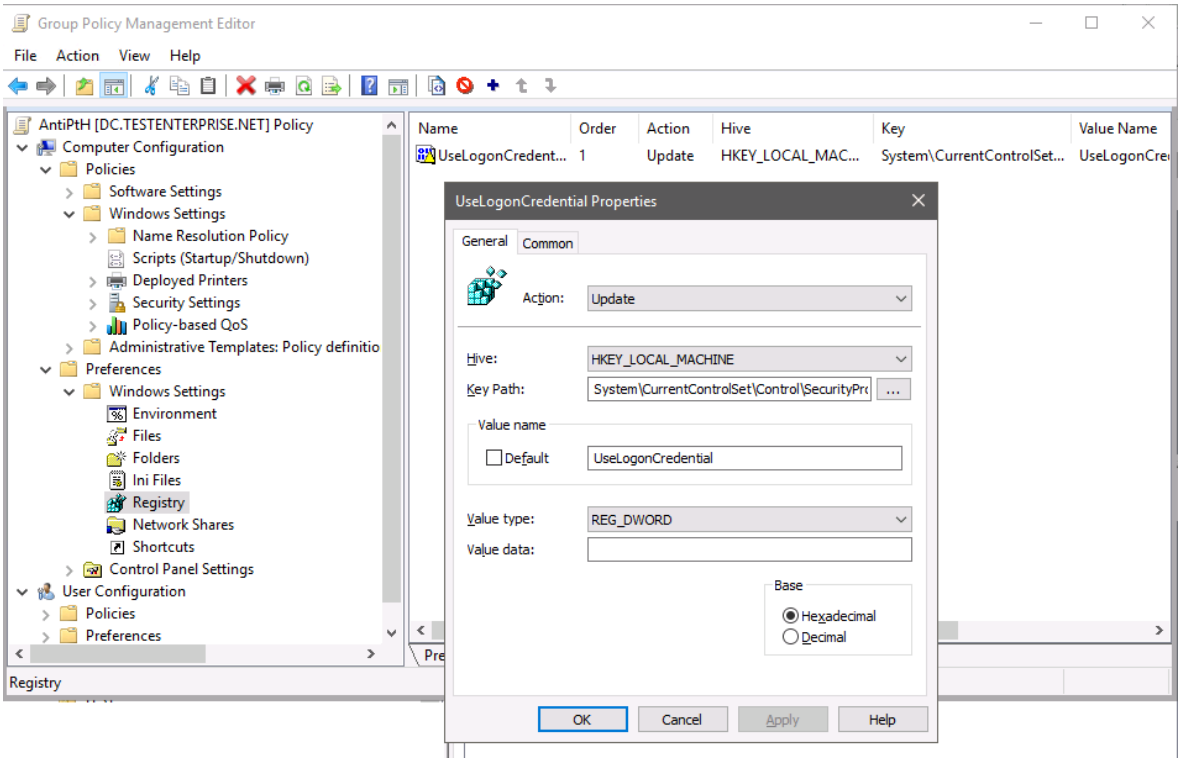
1) *Store password using reversible encryption* – disabled

Computer Configuration\Windows Settings\Security Settings\Account Policies\Password Policy



2) Disabling WDigest

a) Setting the **UseLogonCredential** parameter in this registry branch HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\SecurityProviders\WDigest to 0 will prevent WDigest credentials from being cached in memory.

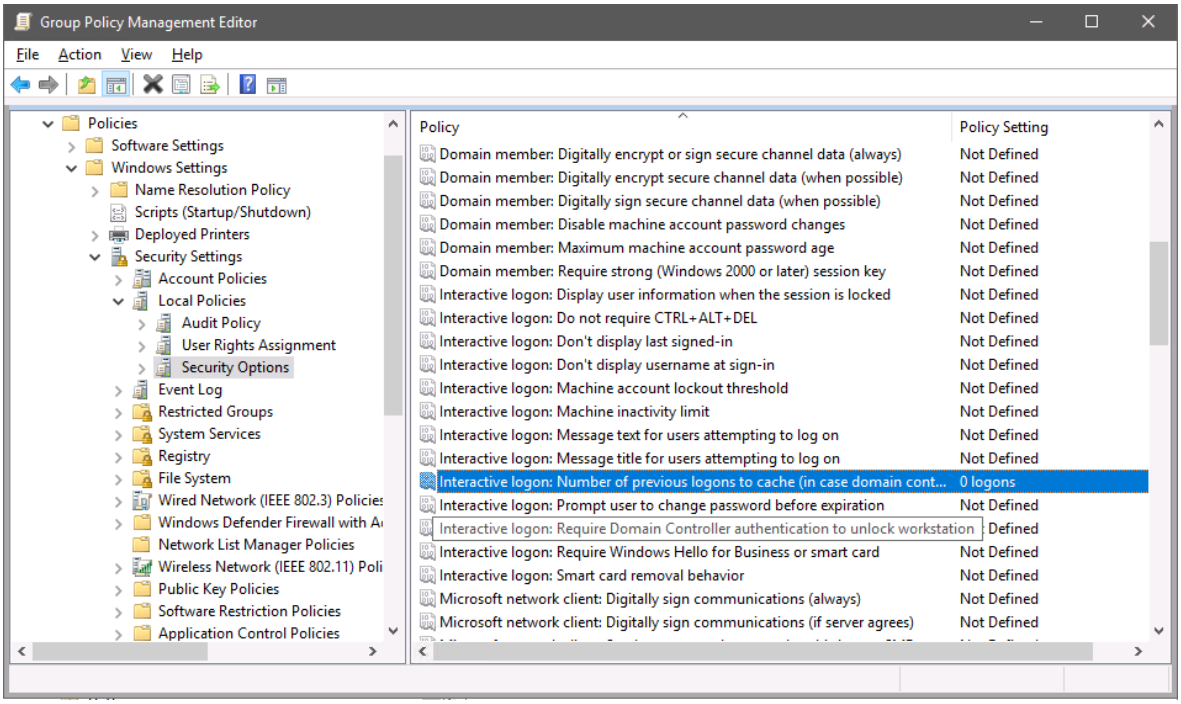


b) setting the **Negotiate** parameter to 0 here will disable WDigest completely – I'm not going to add this parameter to GPO yet – additional testing is needed to make sure IIS clients can authenticate properly.

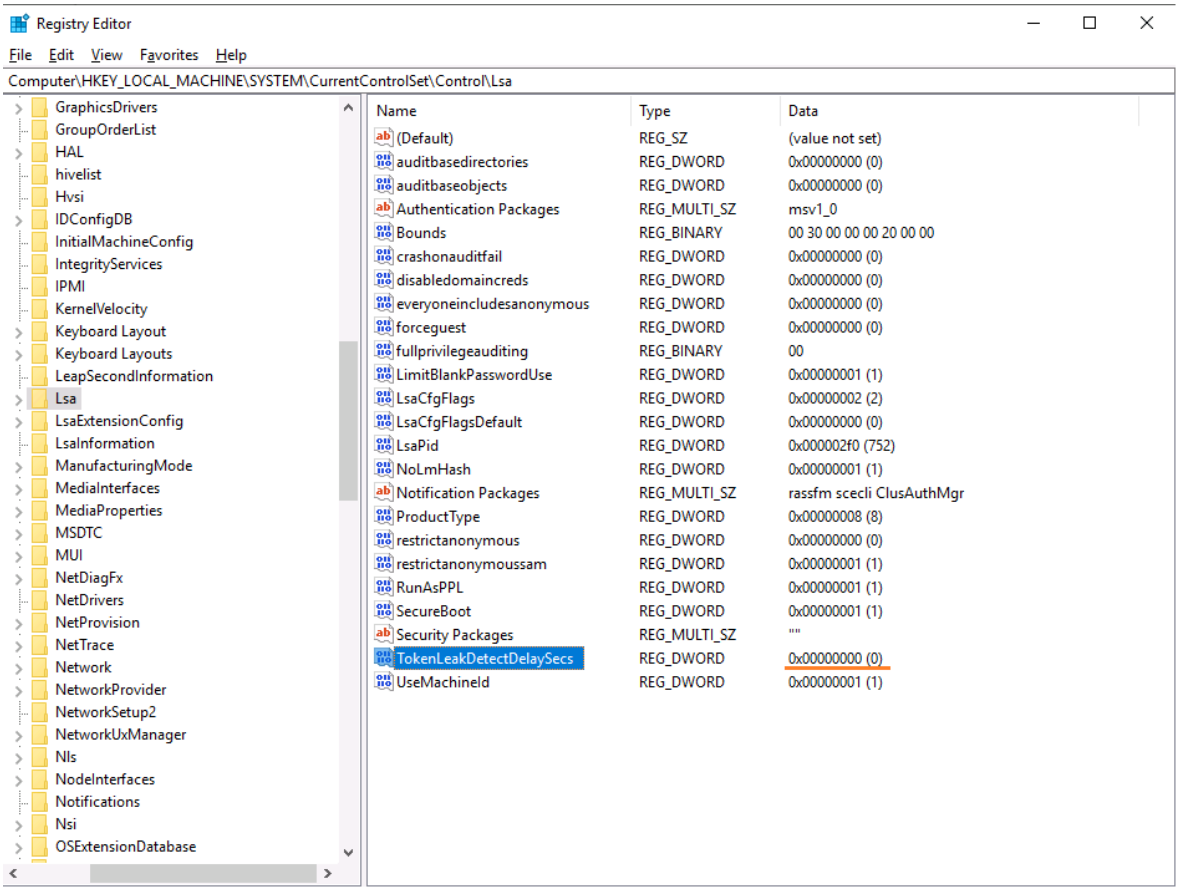
3) **setting parameter**

a) *Interactive Logon: Number of previous logons to cache (in case domain controller is not available)*

to 0 effectively disables caching of user credentials:



b) **some articles on Internet also advise to set the parameter** HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa**TokenLeakDetectDelaySecs** to 30 to define the number of second the credentials of logged-off users can stay in memory, but according to MS documentation this is not required for systems starting from Windows 8.1/Windows Server 2012 R2. In spite of the MS's recomendation of setting this parameter to 30 it's set to 0 by default on Windows Server 2019 so I won't be changing that value in gpo:

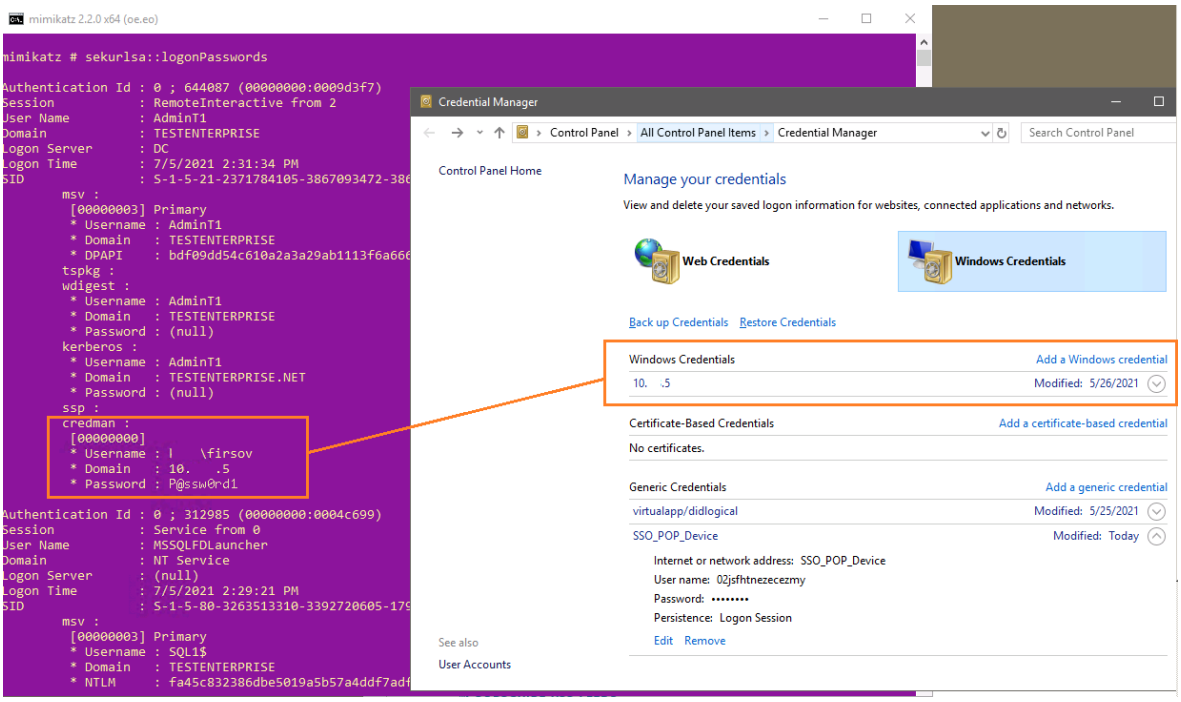


There're also a couple of options – 1) *removing debug privileges from all users icluding administrators* and 2) *preventing users from saving passwods* – that can be set in this gpo but for now I'll abstain from using them because

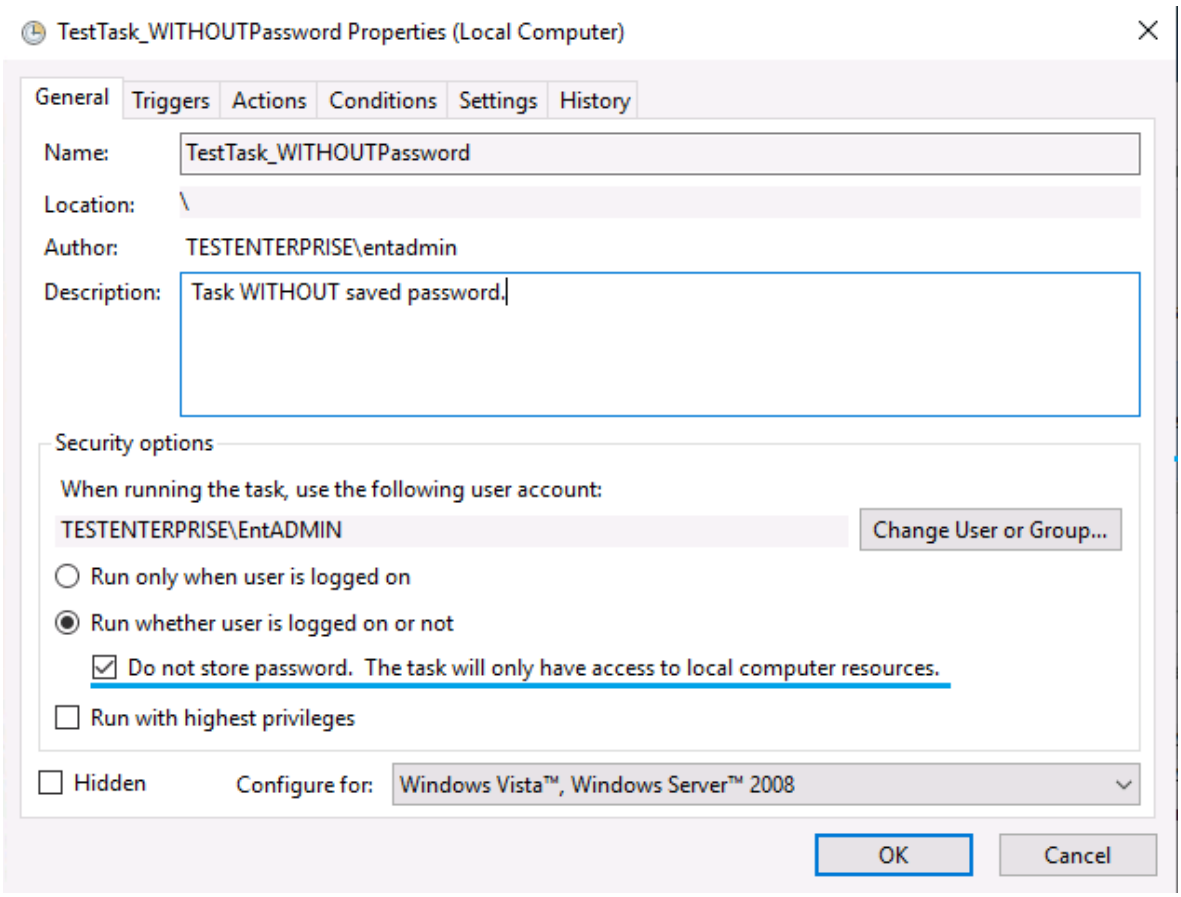
1) *removing debug ptivileges from administrators* (Computer Configuration\Windows Settings\Security Settings\Local Policies\User Rights Assignment – **Debug Program** policy) may prevent some programs to install properly – more information here)

2) *blocking the possibility for users to save passwords* (Computer Configuration\Windows Settings\Security Settings\Local Policies\Security Options – **Network access: Do not allow storage of passwords and credentials for network authentication** policy) will prevent Task Scheduler from keeping passwords (more information here).

Nevertheless, without the second GPO setting (*Do not allow storage of passwords and credentials for network authentication*) – the process of getting the password may be as simple as this:

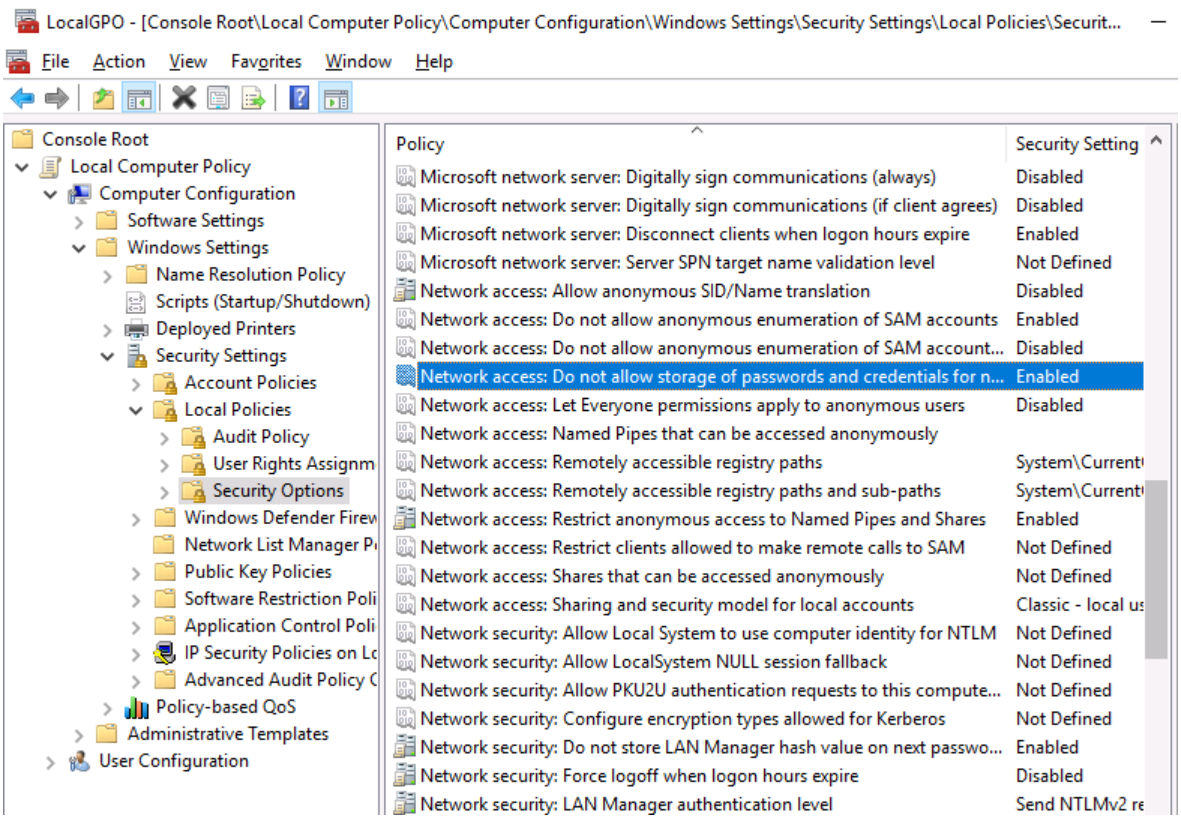


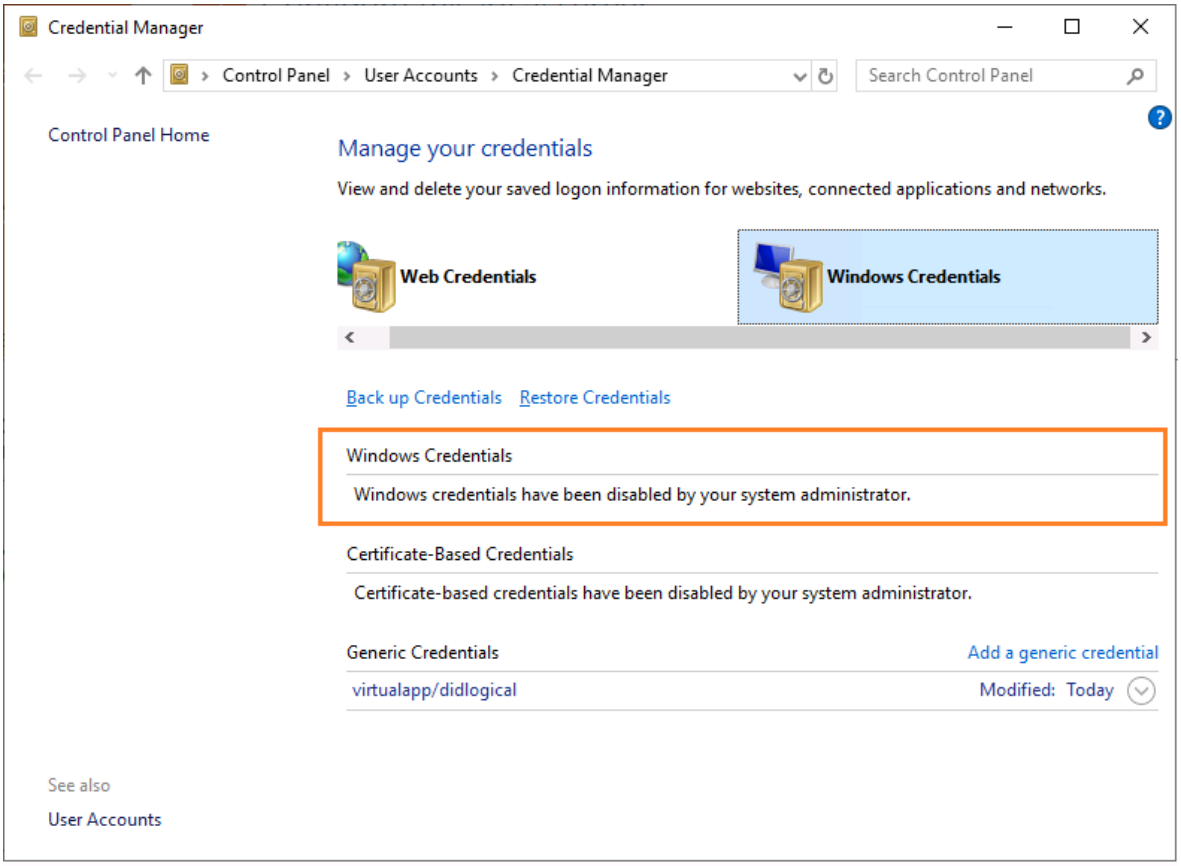
I think the following technique may be used to workaround the problem with keeping passwords in Task Scheduler (at least in some cases, of course): only tasks that operate locally should be created on the servers to which the policy is applied:



If there's a task that must access other computers – do not apply the policy with *Do not allow storage of passwords and credentials for network authentication* to that computer account.

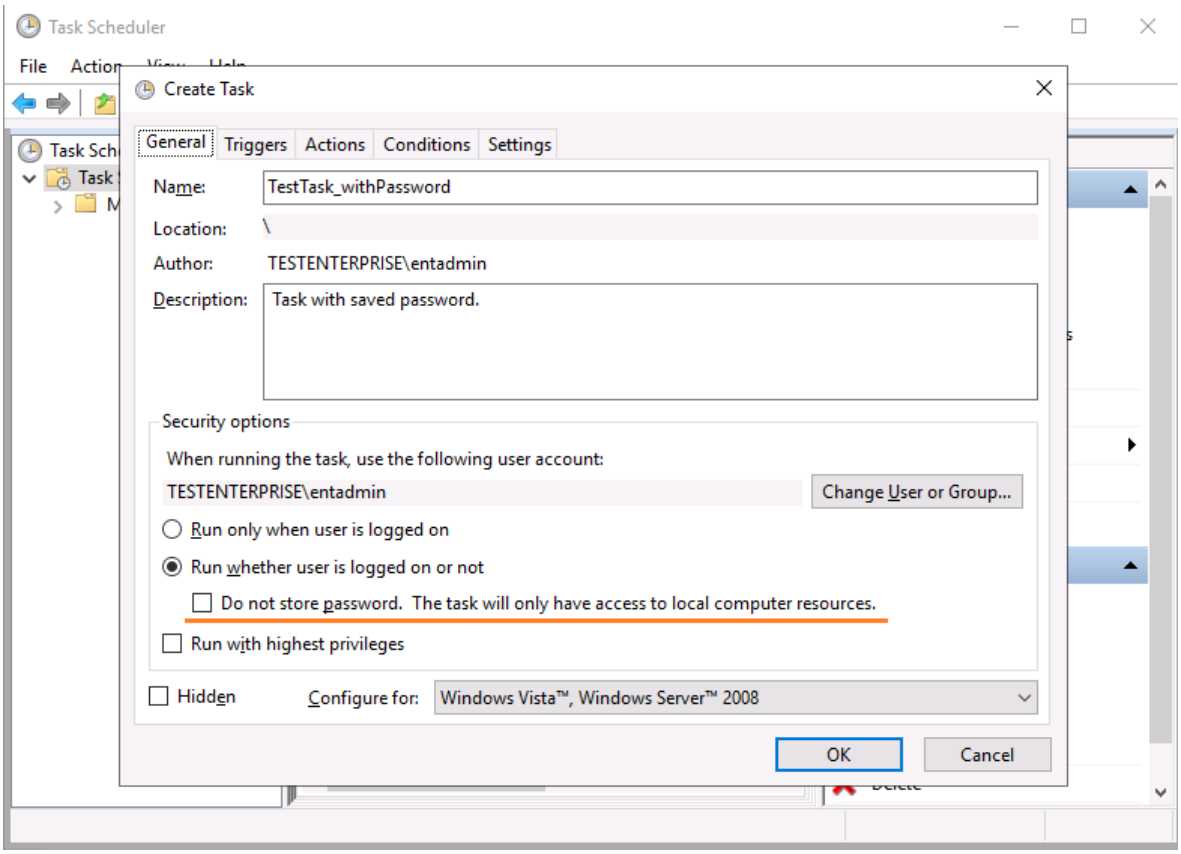
Once none of your scripts on the respective servers require storing passwords you can apply the **Network access: Do not allow storage of passwords and credentials for network authentication** policy setting to those servers:

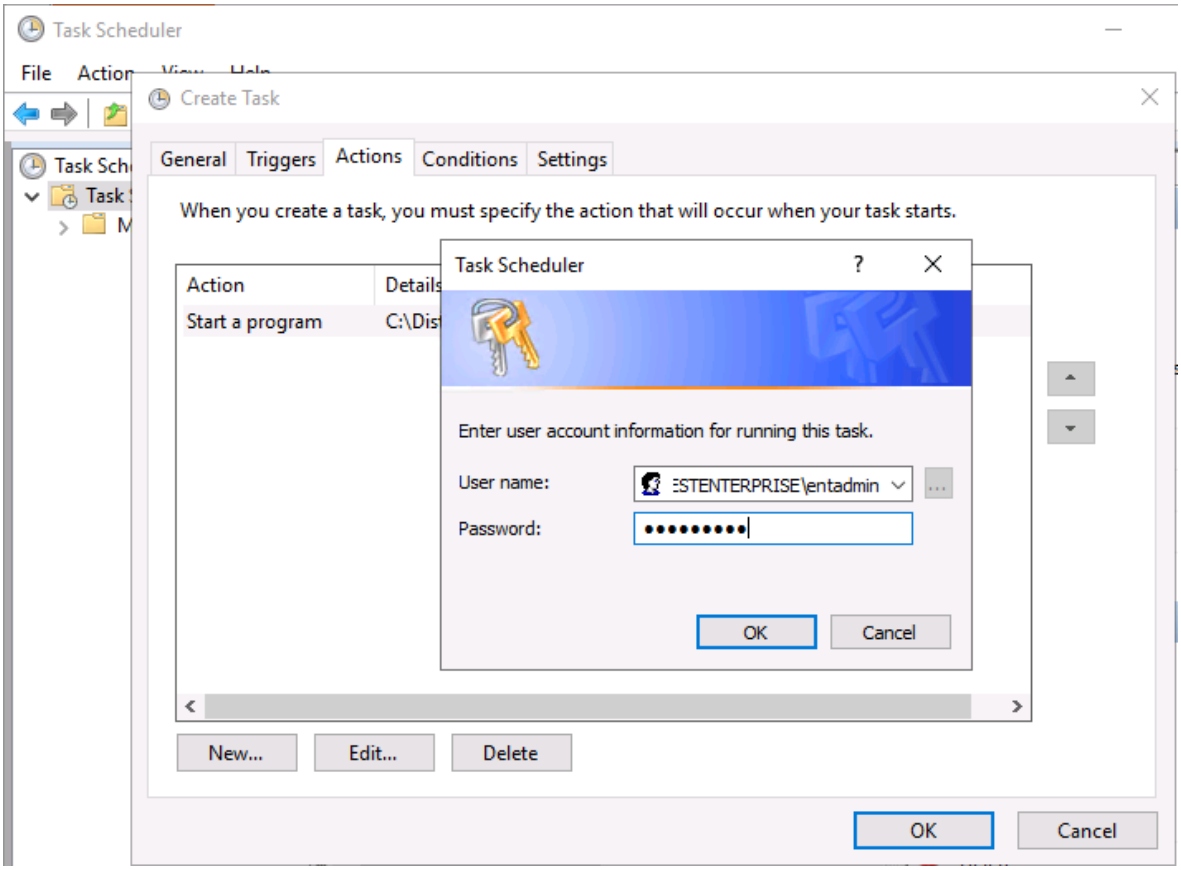


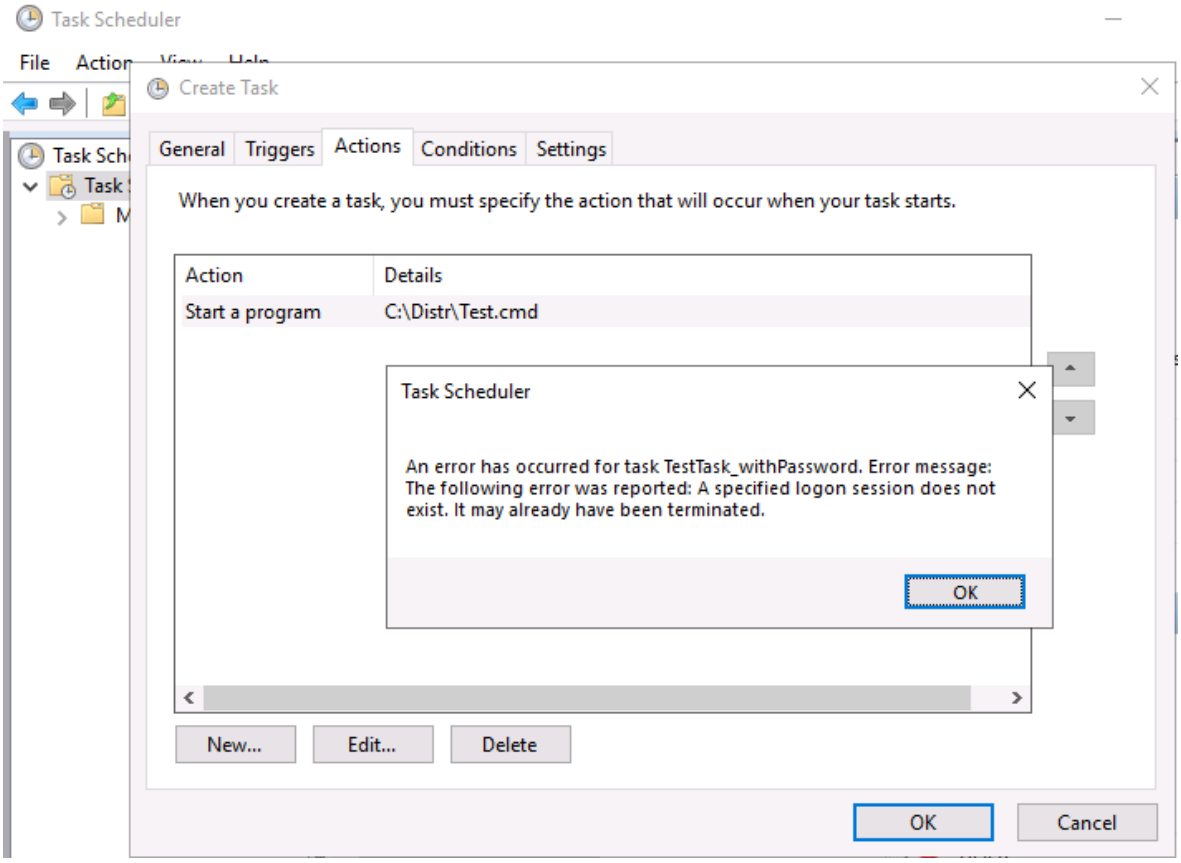


From now on no new passwords can be saved on the target server(s) but the **password** saved previously – NOT the username AND the resource name !!! – will be encrypted:

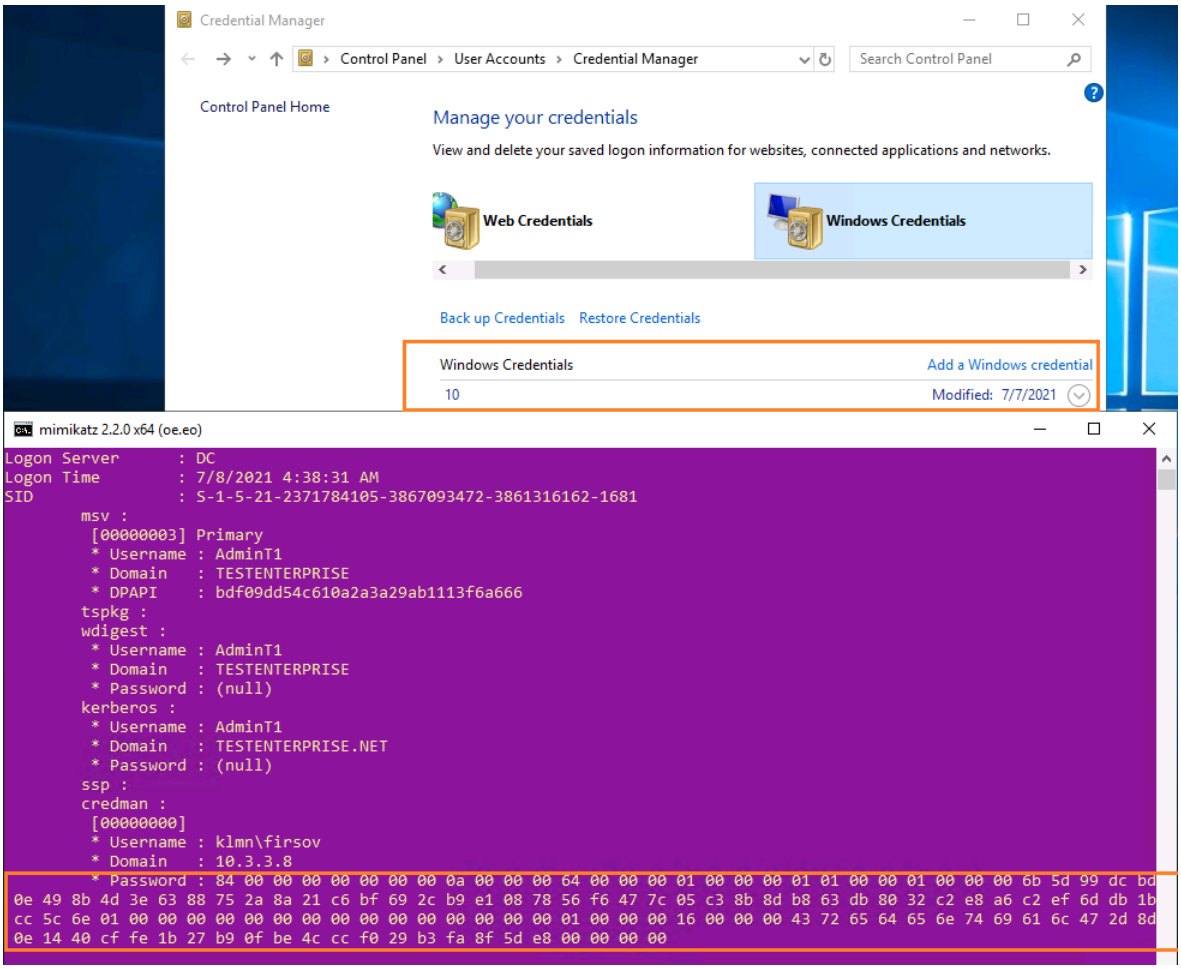
You will see the following error should you try to create a scheduled task with the saved password:



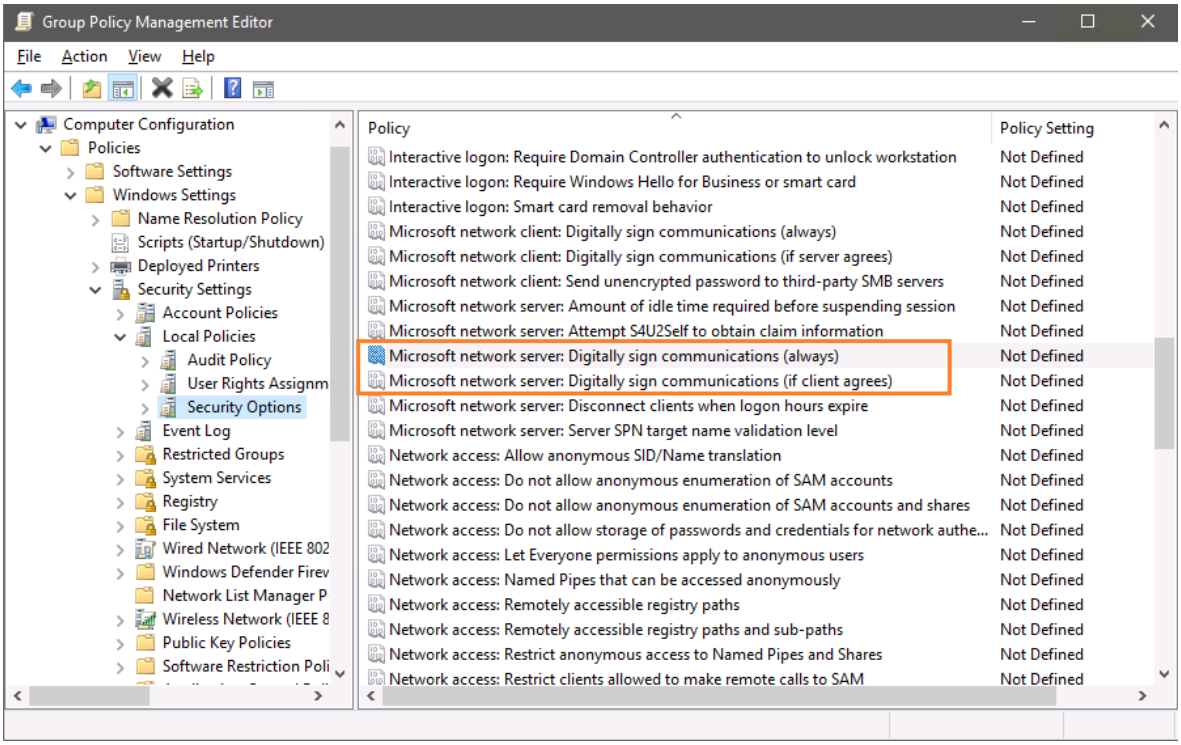




Interestingly enough: if you re-enable the possibility to save passwords the already saved passwords will be kept encrypted:



SMB signing (server and client side) should also be enabled to better protect smb traffic but it also requires thorough testing before implementation – I'll post a screenshot of these settings but won't apply them right now:



Here is the great example of the security settings to deploy in organizations.

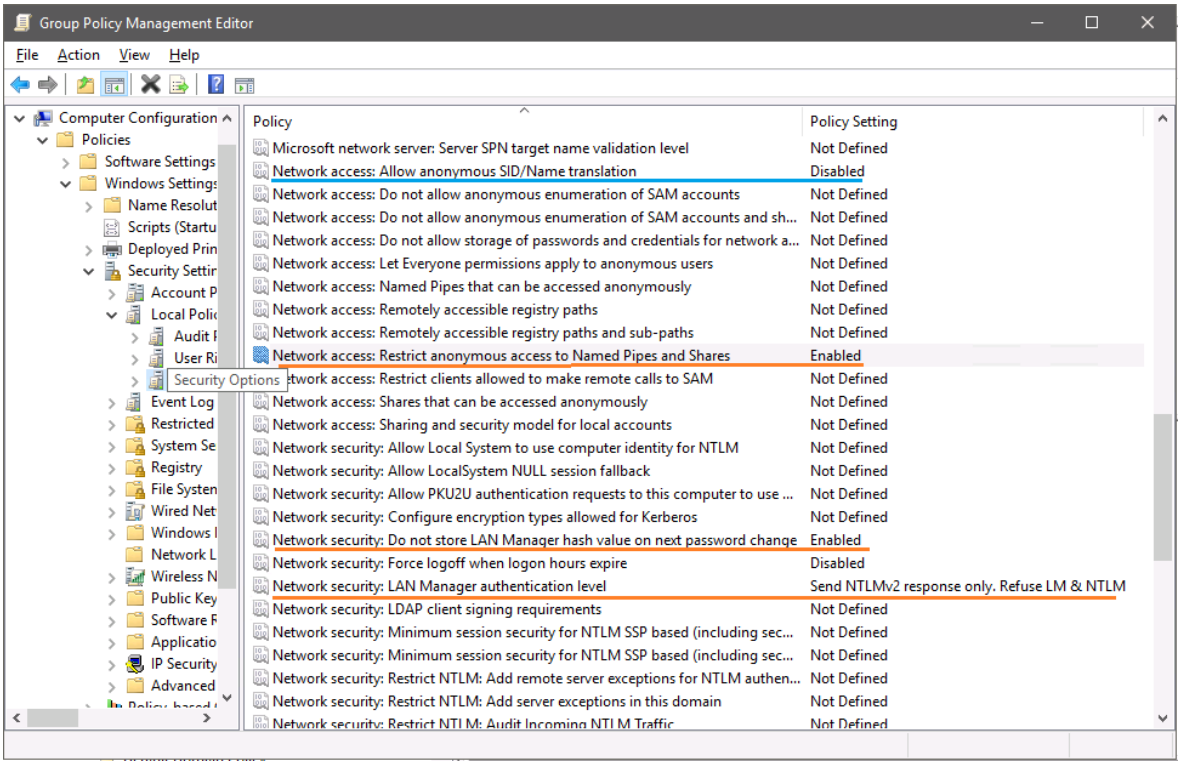
And there’re at least four settings (including Allow Anonymous SID/Name translation) that must be applied at the domain level (*Default Domain Policy*):

Computer Configuration\Windows Settings\Security Settings\Local Policies\Security Options

Network Security: Do Not Store LAN Manager Hash Value On the next Password Change

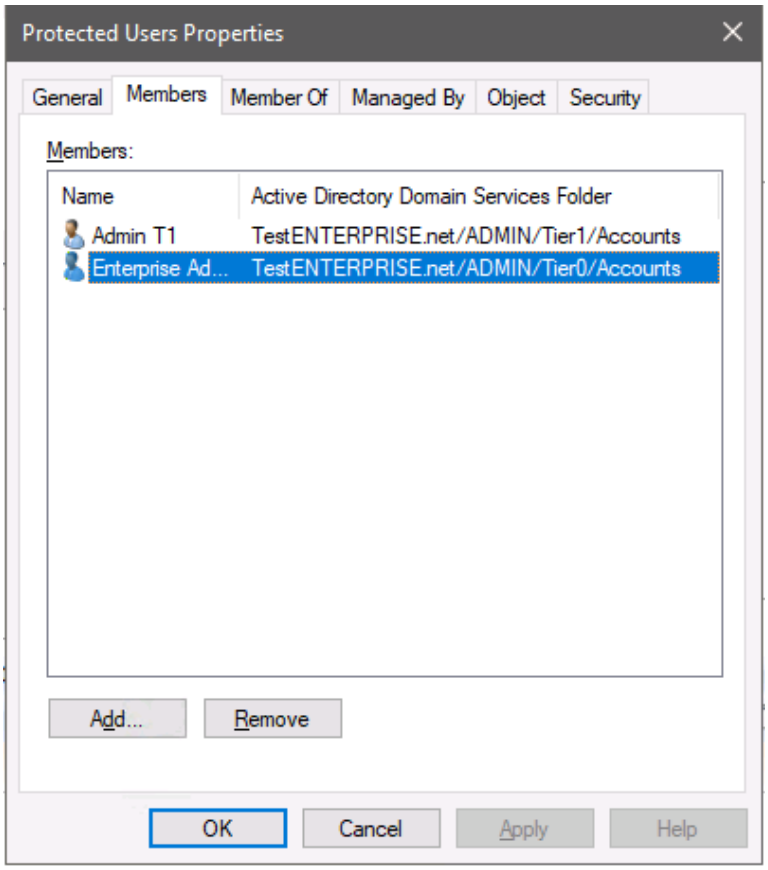
Send LM & NTLM – use NTLMv2 session security if negotiated.

Anonymous access to Named Pipes and Shares must be restricted.

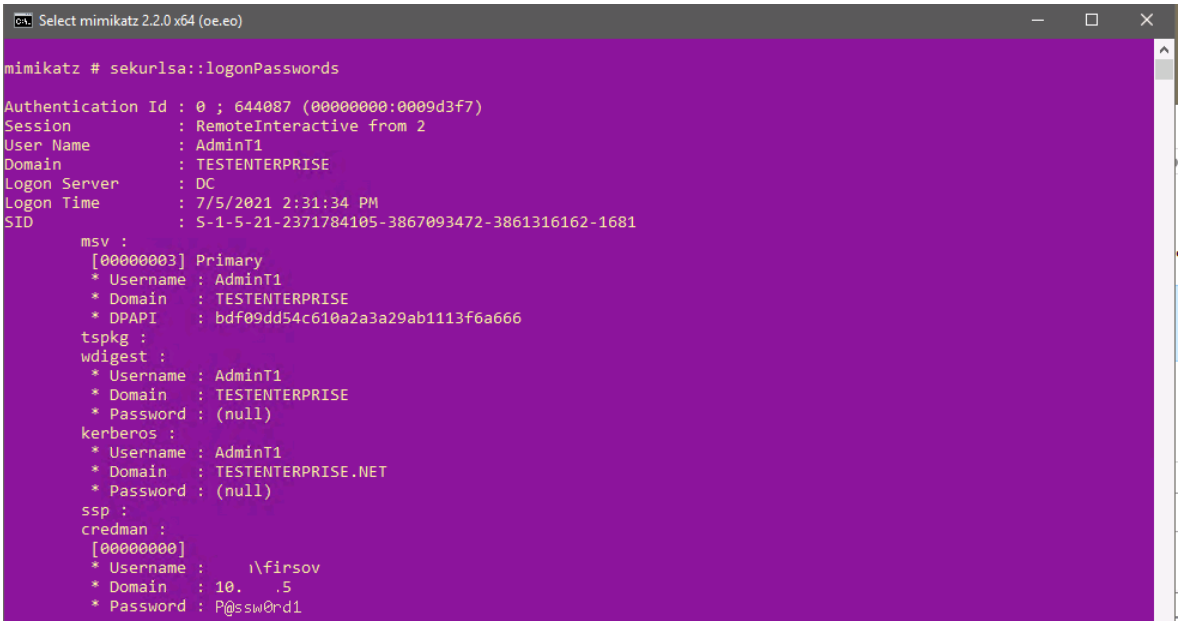


Protected USERS

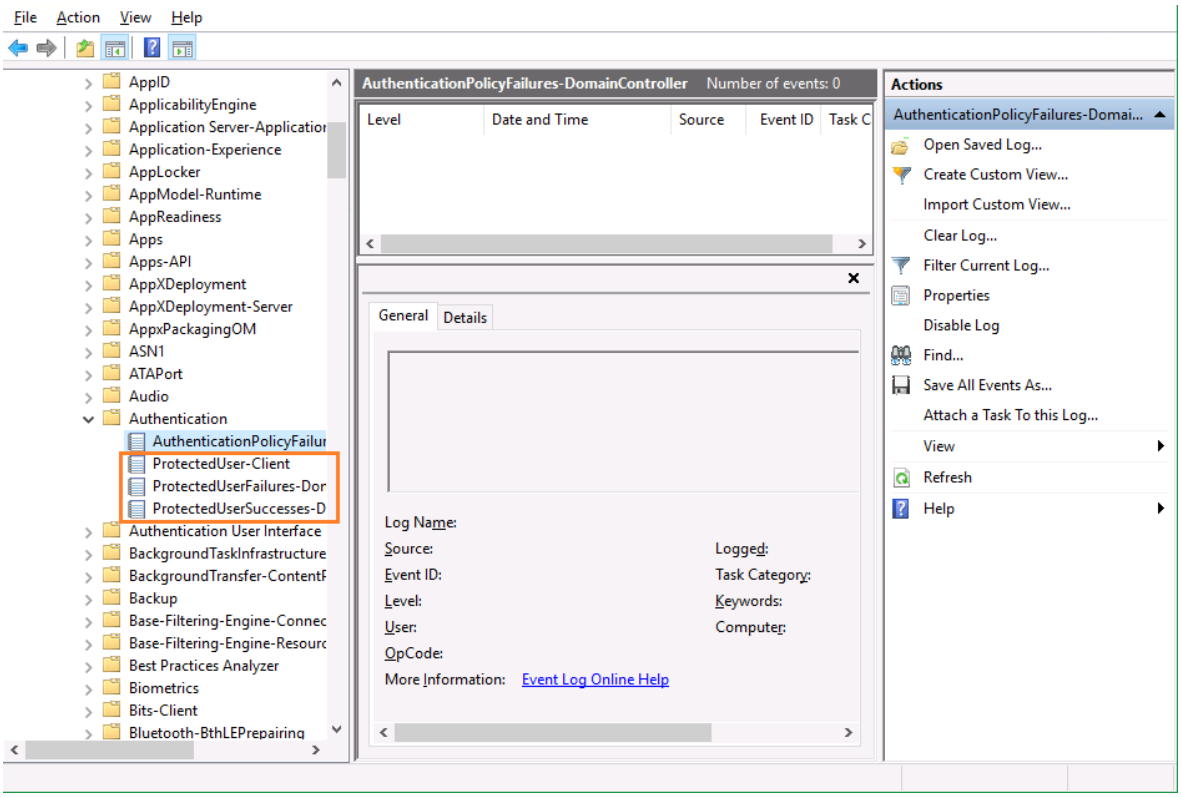
The most valuable user (not computer or service!) accounts may also be added to the *Protected Users* group (you can read about protection it provides here).



No NTLM hash will be created for the account:

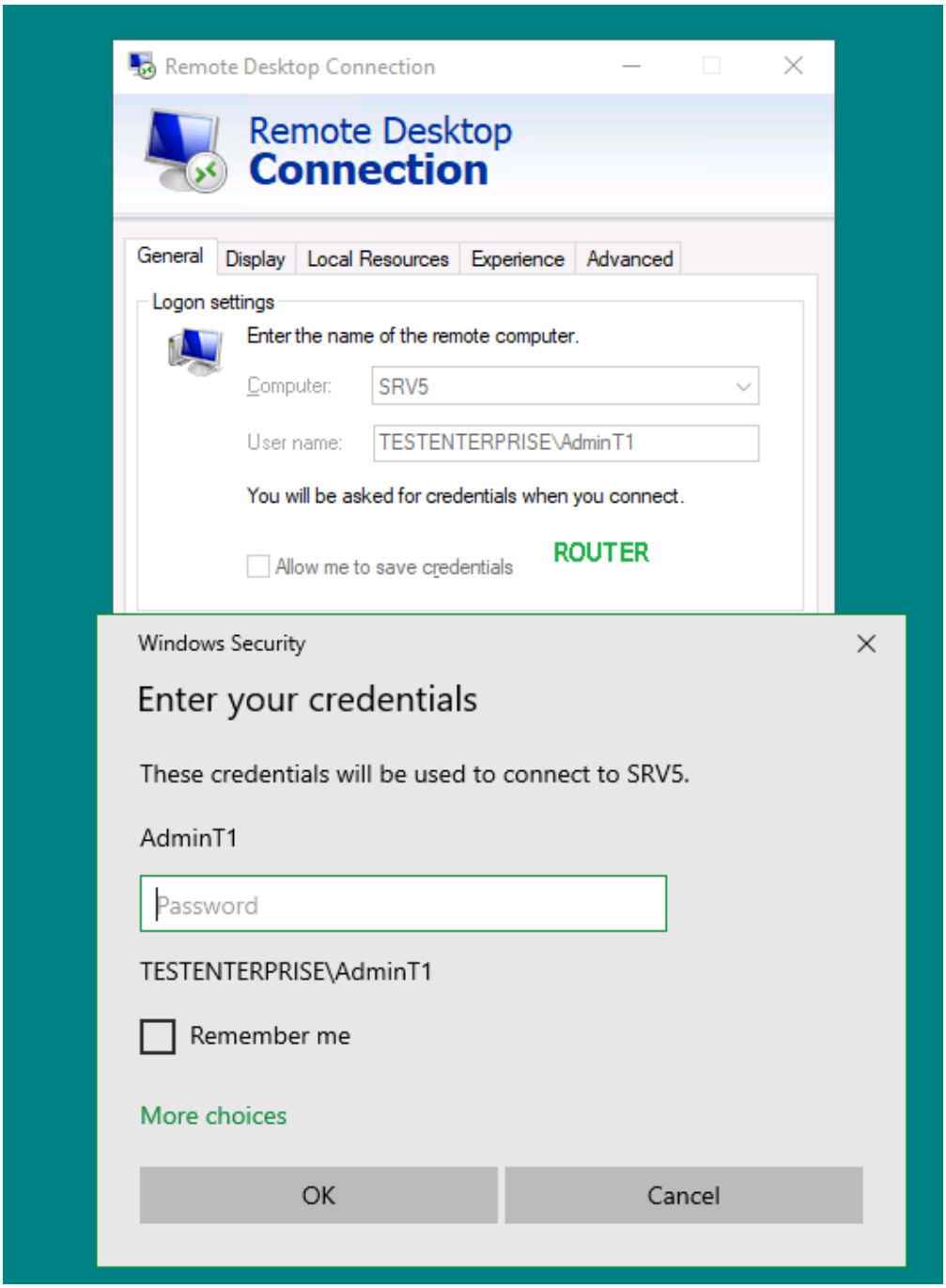


Once the accounts are added to the Protected Users group you should enable the Protected Users logs:

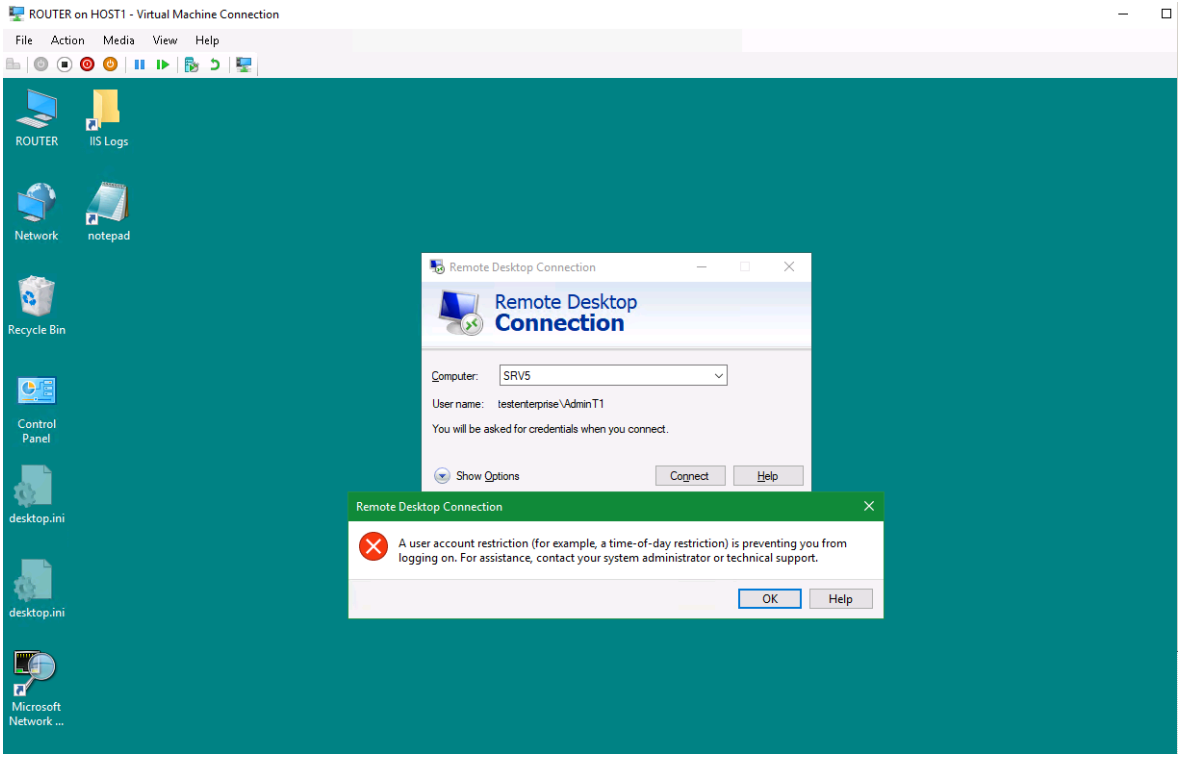


Navigate to
Application and Services Logs\Microsoft\Authentication and enable the logs by right-clicking and selecting *Enable Log*.

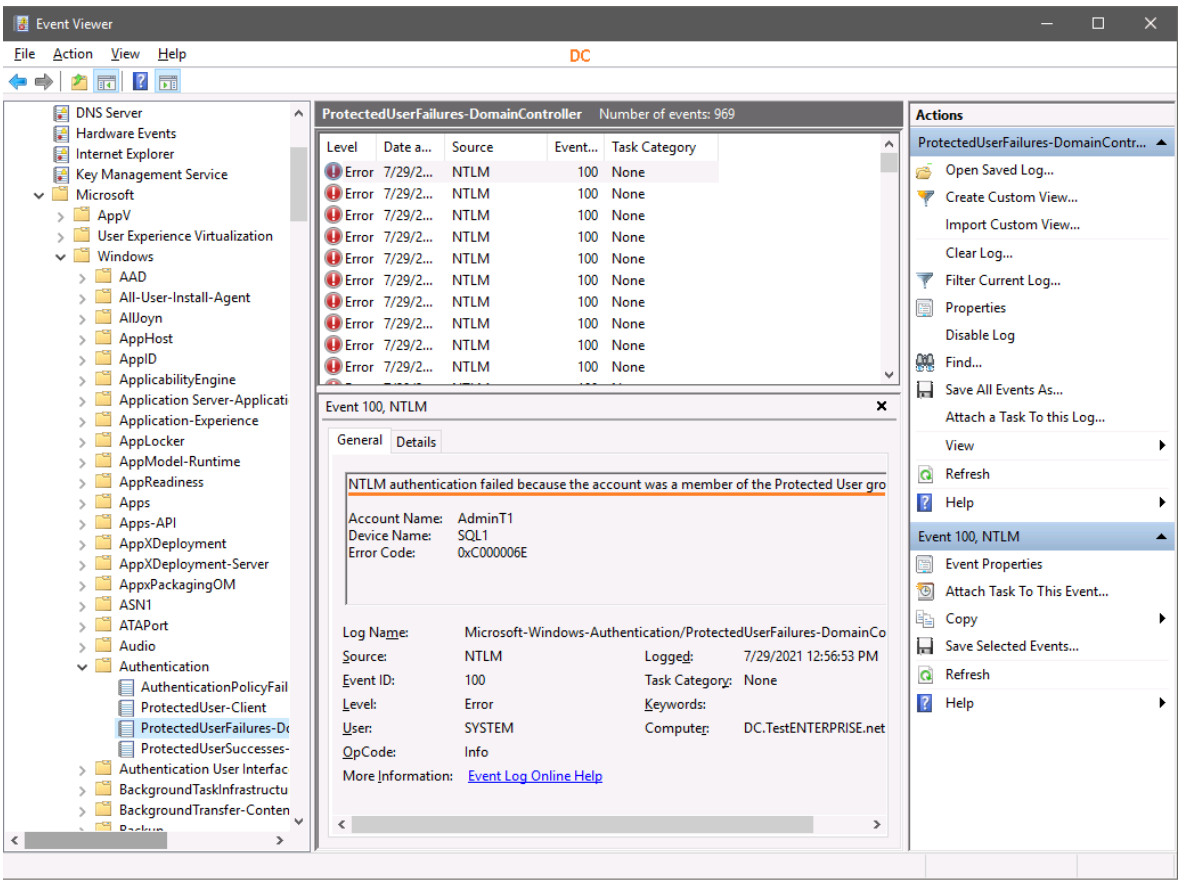
Please note that from now on you won't be able to connect to domain computers using accounts that are members of the Protected Users group from standalone computers – even when there's no any restrictions applied to target machines, for example: AdminT1 is the account made for administering Tier1 servers but if I try to RDP to some domain server (SRV5 in this case) from non-domain ROUTER…



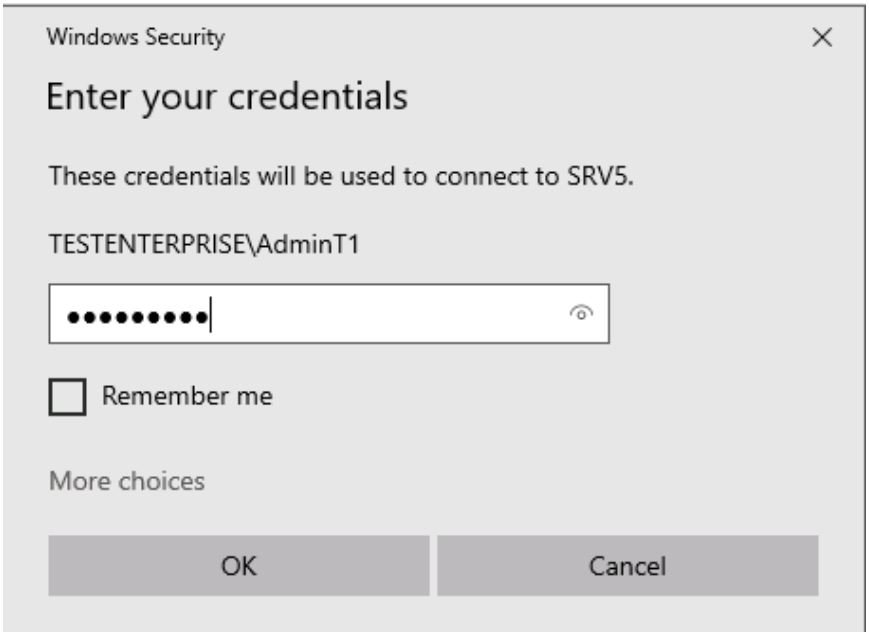
…I will get this error:

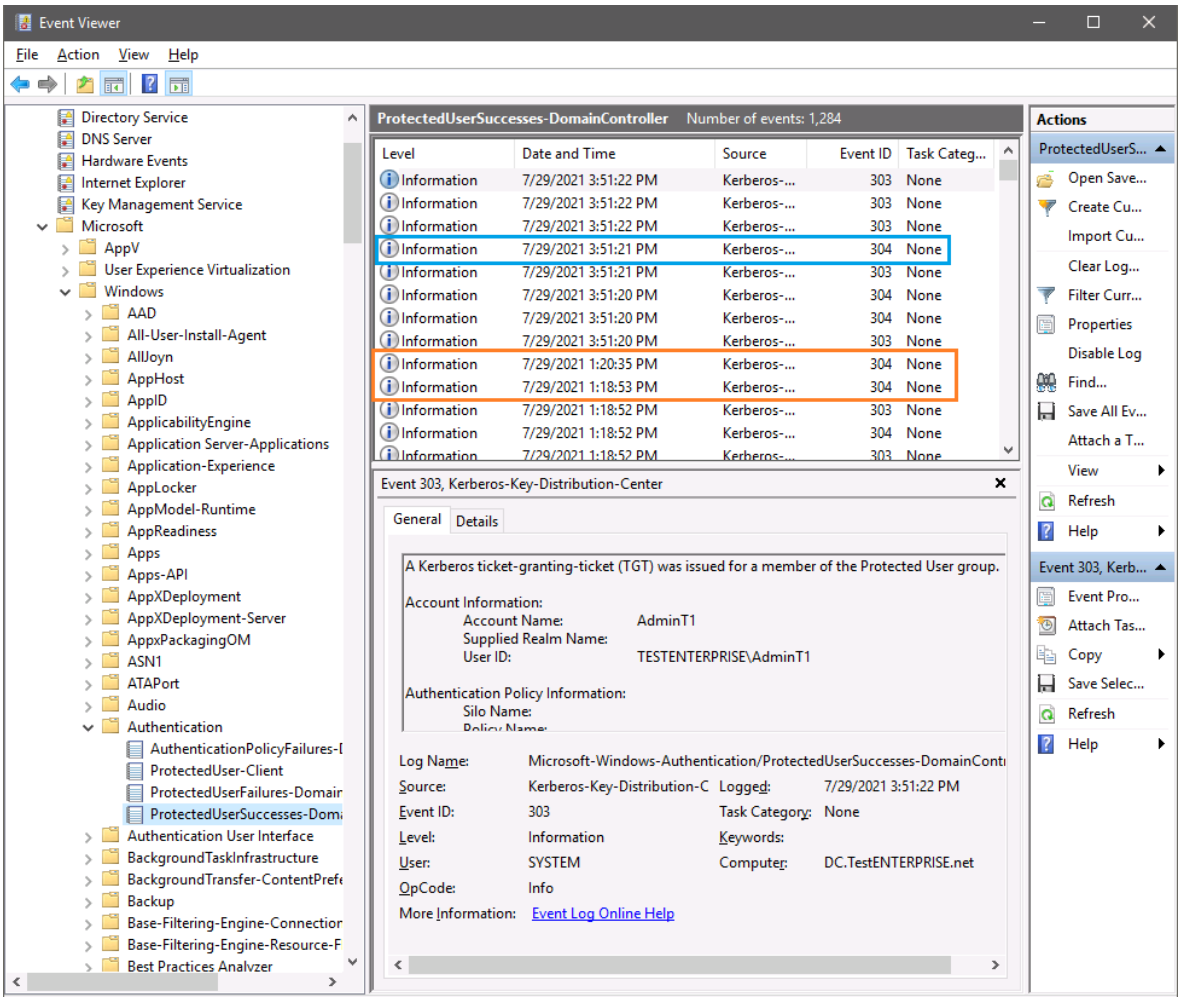


The ProtectedUsers-Client log on ROUTER will stay empty while the ProtectedUsersFaillures-DomainController on SRV5 will log the following information:

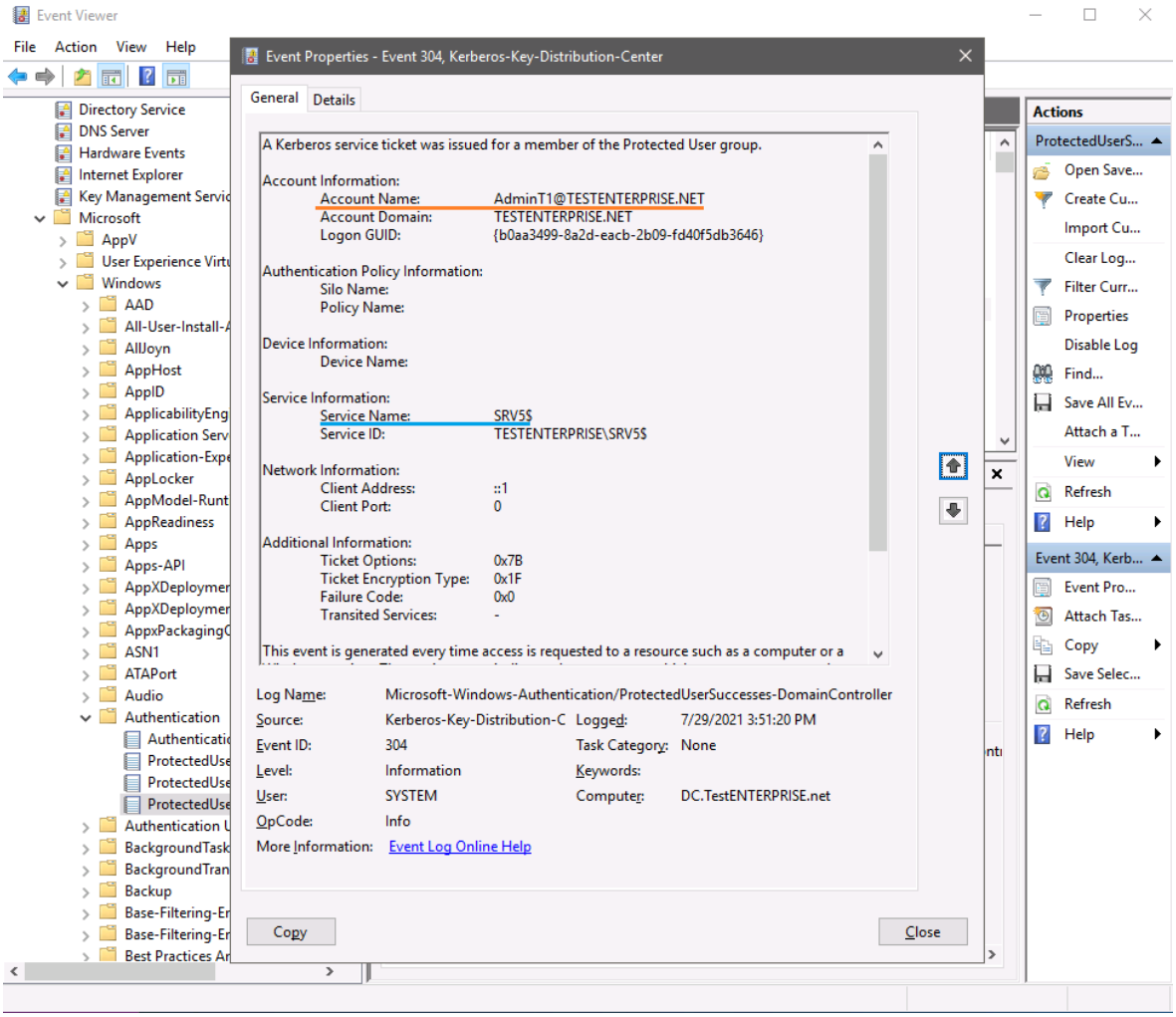


If I then try to connect to SRV5 from other domain member the connection will succeed and the following events will be generated on both the server and the client:

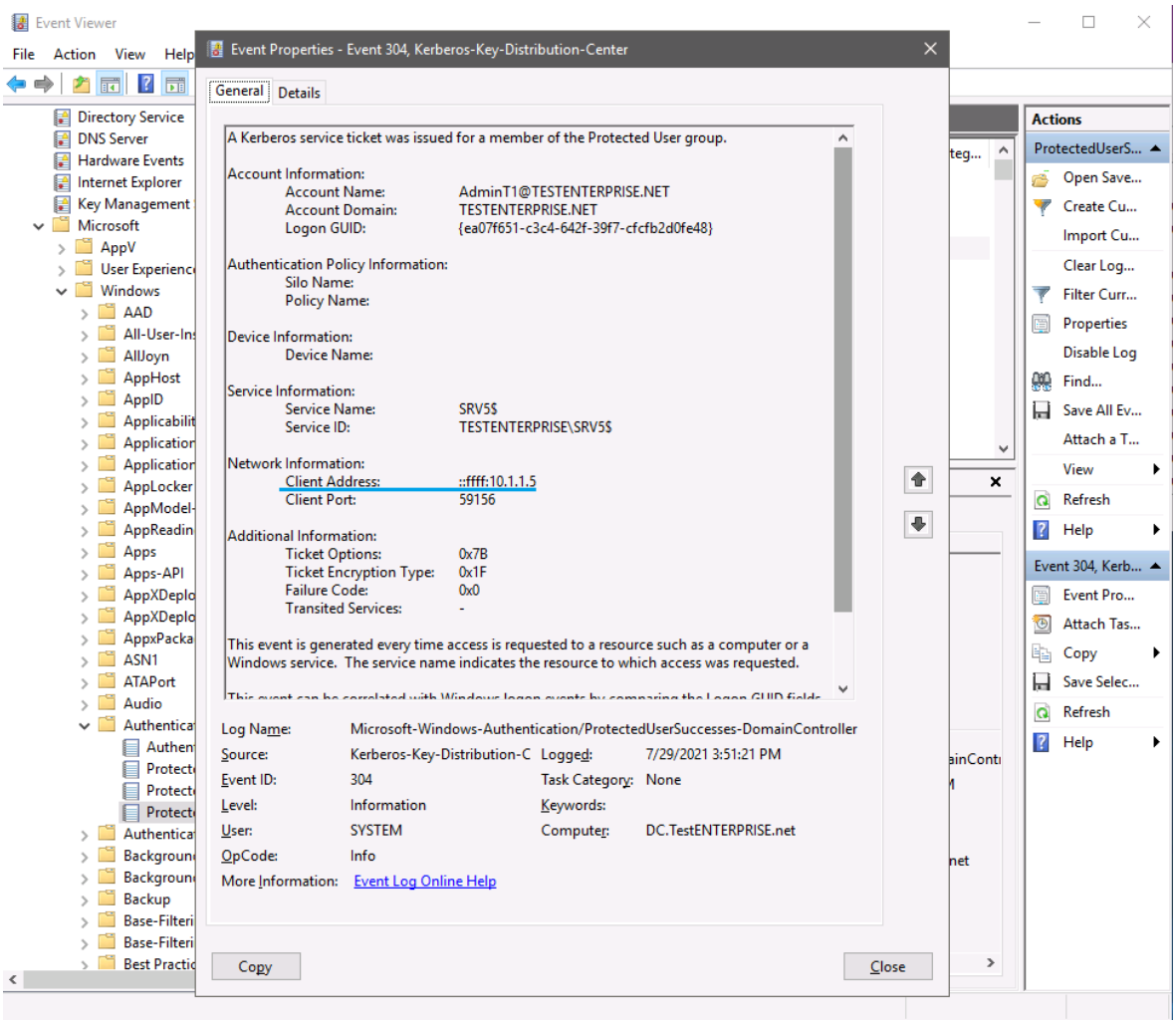




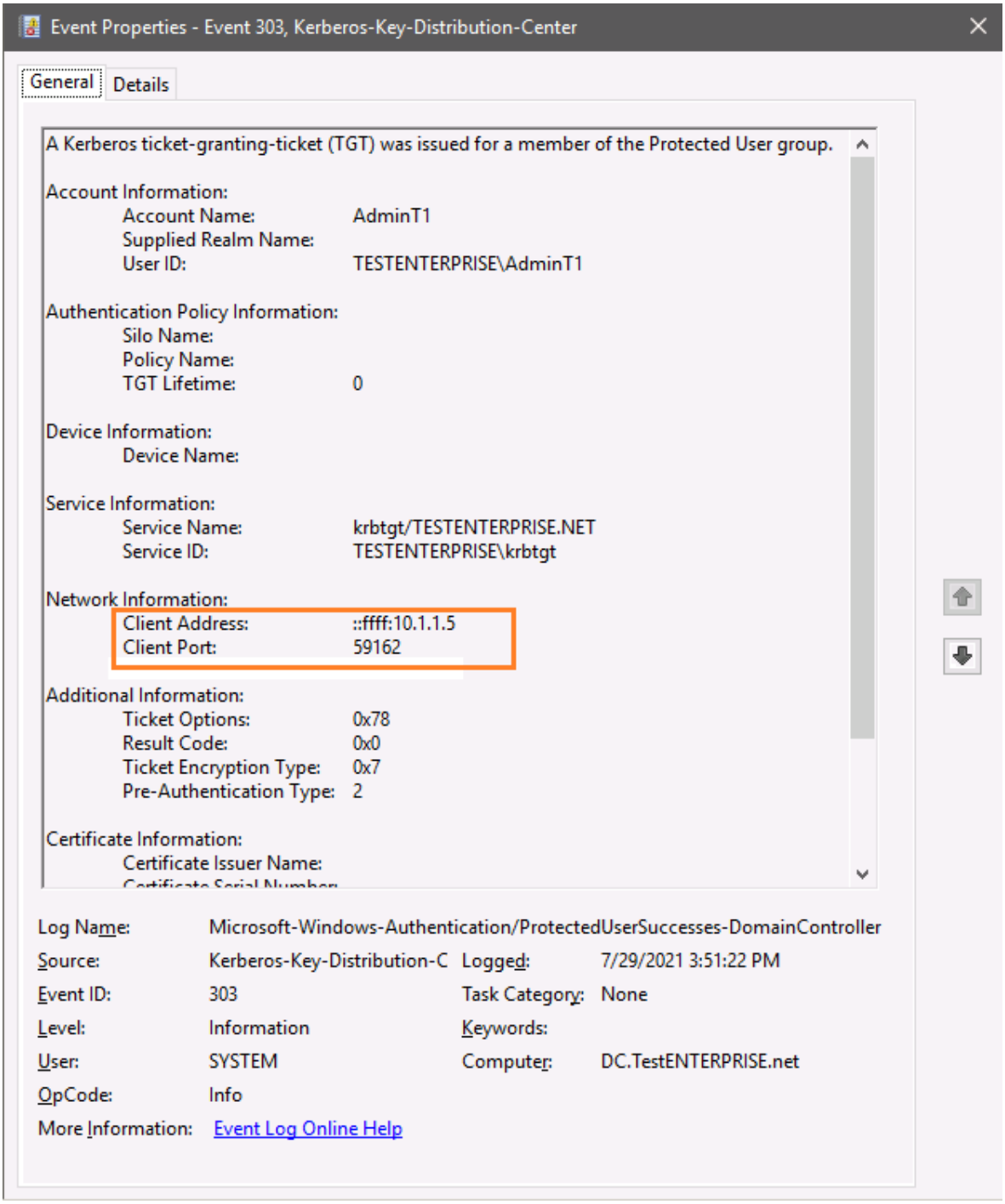
I've highlighted the first two 304 events because they are completely the same -.



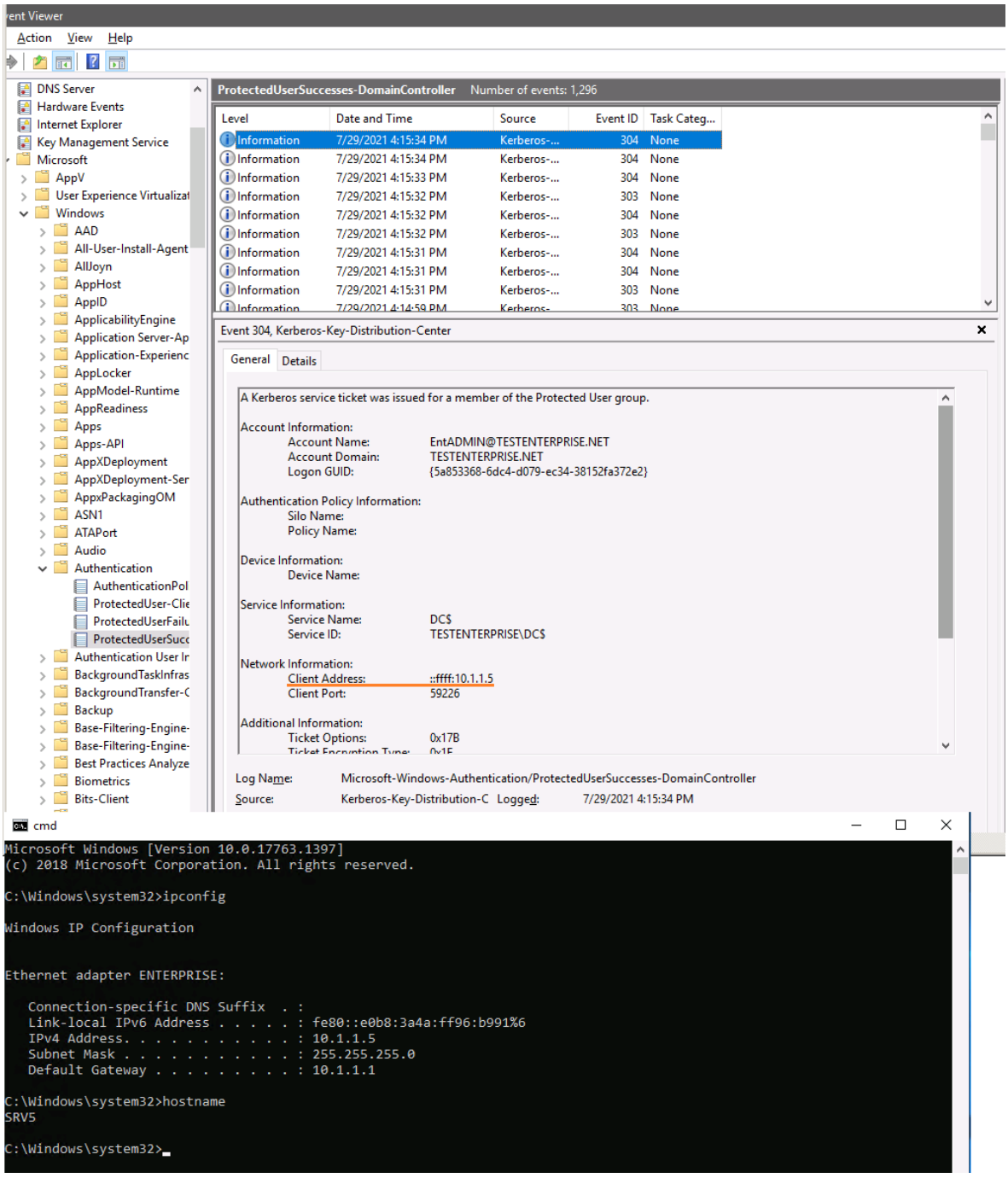
…and the third event 304 contains the target network address of SRV5:



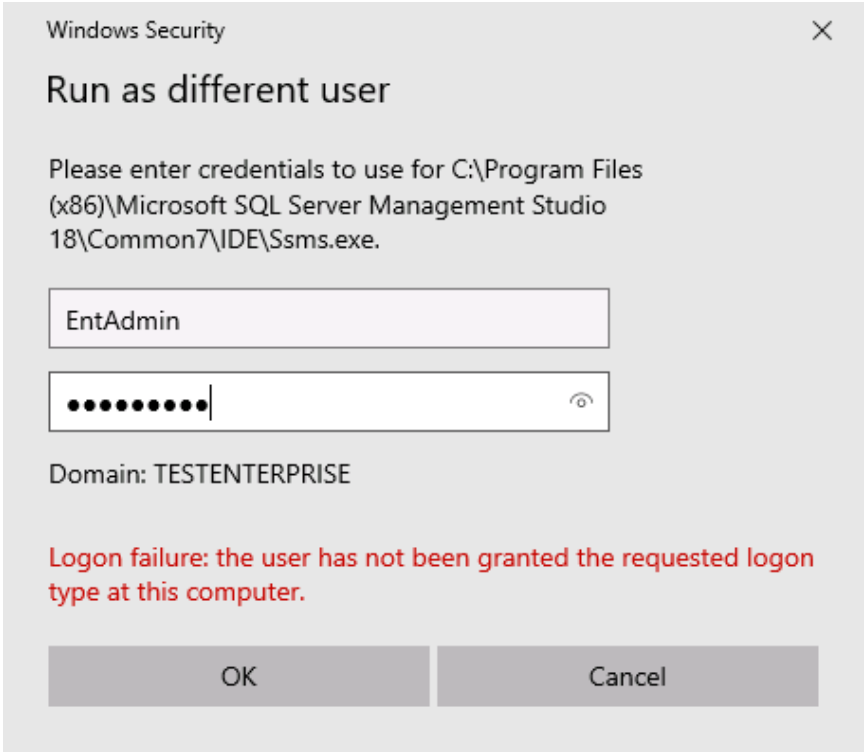
Here's the example of the event 303:



Note: Client Address here is the address of target computer (SRV5), NOT the address from which the connection was made:

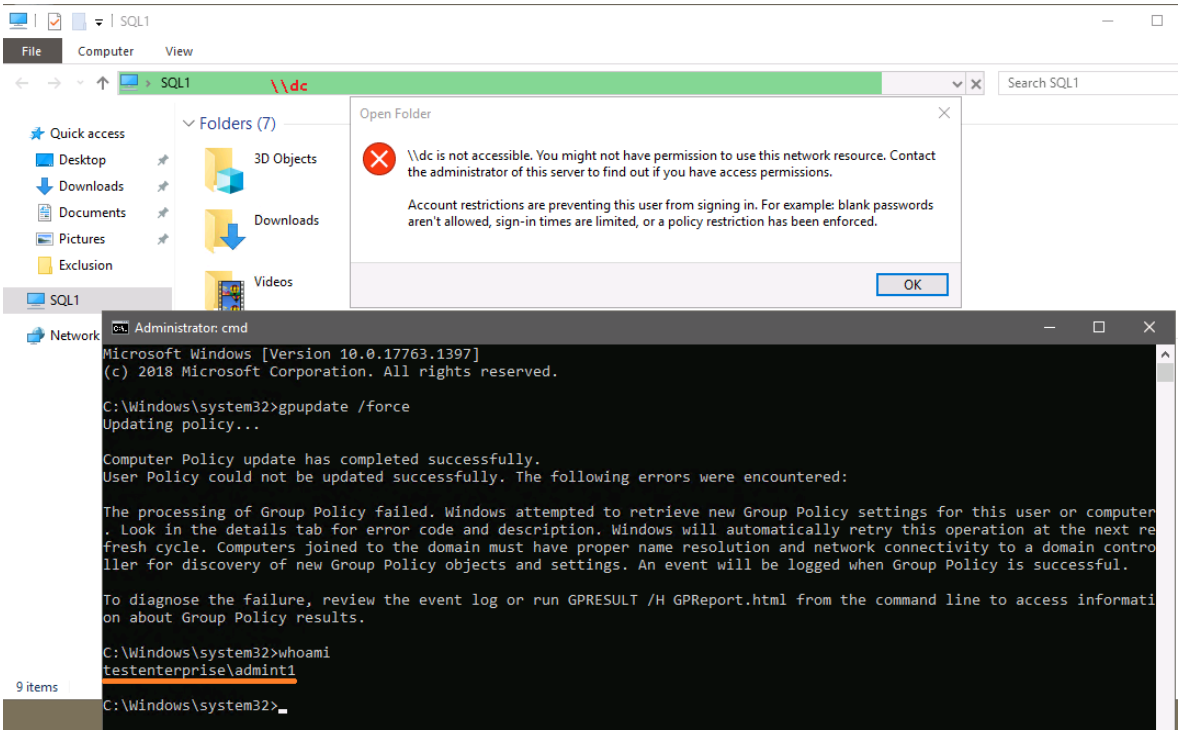


Thourogh planning is needed before implementing tiered AD model and security-based GPOs – it's easy to bump into the issues after applying the aforementioned policies (including the ones described in part 3 and part 4), for example: if the *EntAdmin* is the only account with administrative privileges on SQL server and this highly sensitive account becomes prohibited from logging onto tier1 servers, it would be impossible to connect to SQL server while logged on as tier1 *AdminT1* account, even with the Run as another user command:



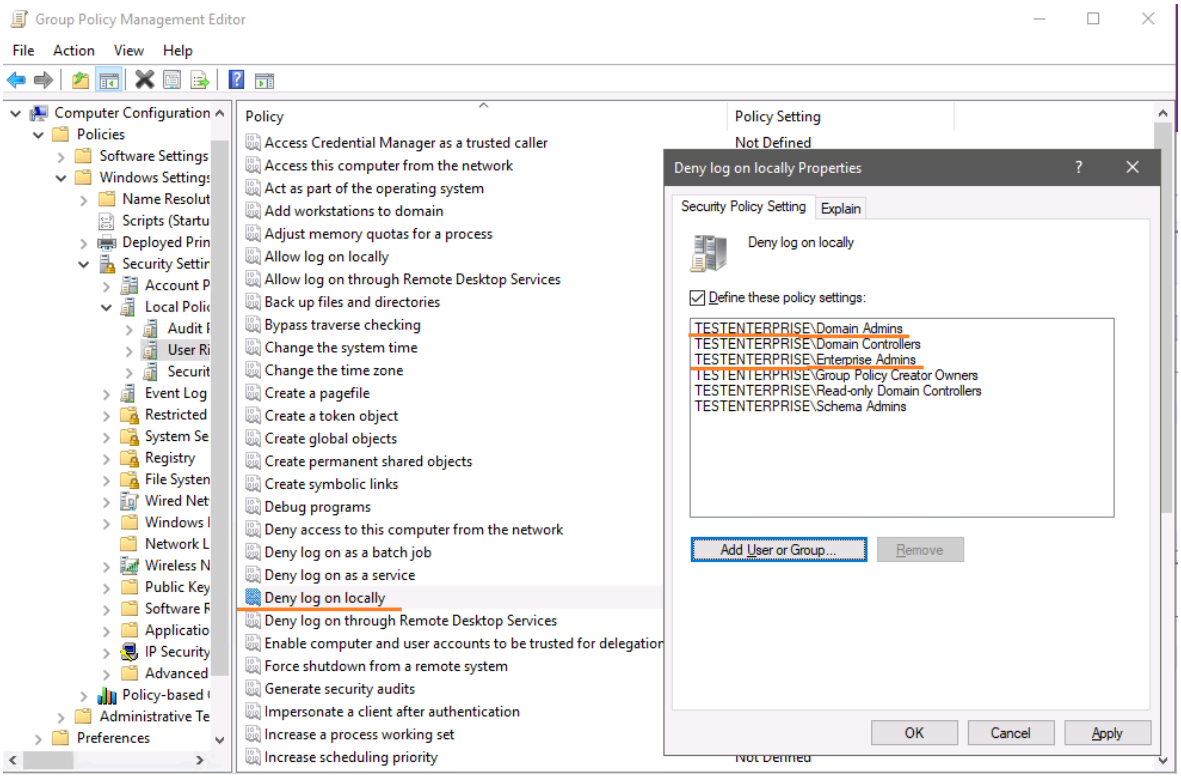
While this problem is “by design”, the other is not: there’s one more issue that seems to be rather weird:

once all GPOs have been applied to Tier1 servers I’ve noticed that I can’t access domain controller while logged on to Tier1 servers (servers from the SERVERS OU) as AdminT1 (Tier1 admin), for example:

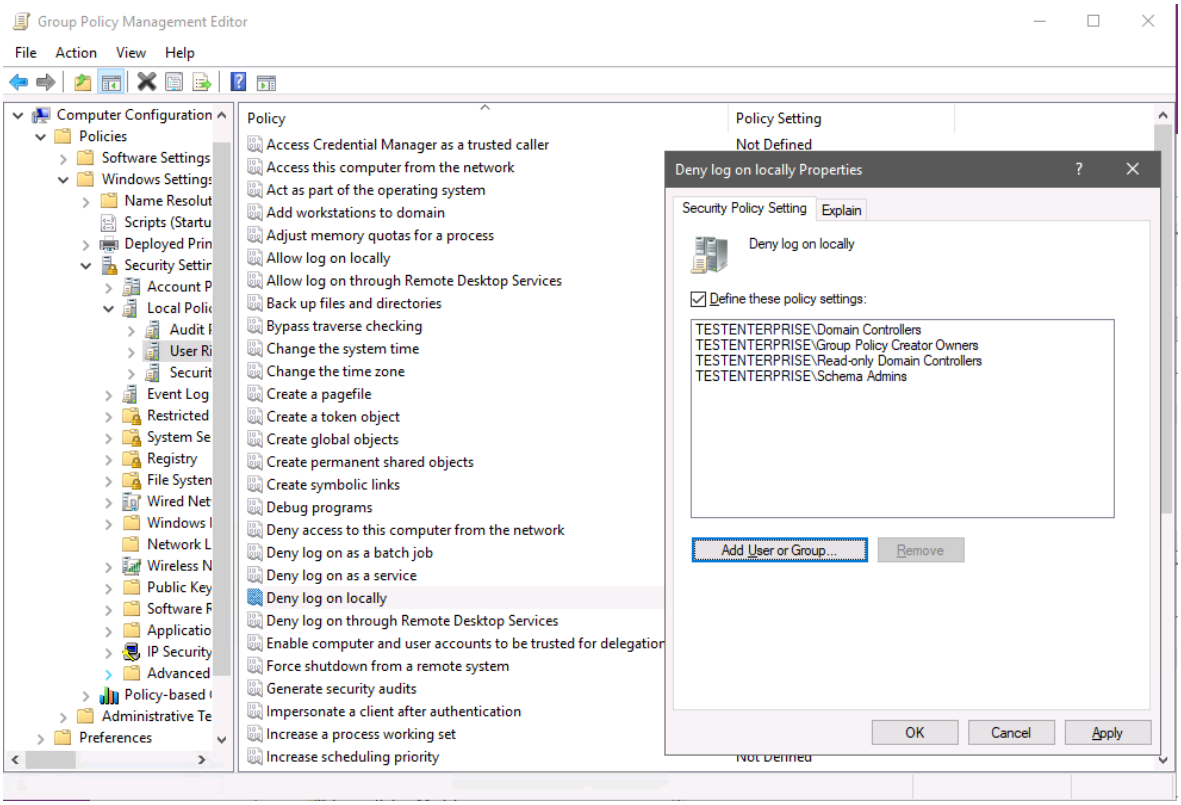


I still don’t know what’s the connection between this issue and the *Deny Log On locally user right* – one of the key policy settings in the tiered AD model that was configured in the RestrictSERVERLogon GPO to prevent lgging on to servers from one tier using accounts from other tiers – but deleting the Domain Admins and Enterprise Admins from this policy does really resume access to \\DC:

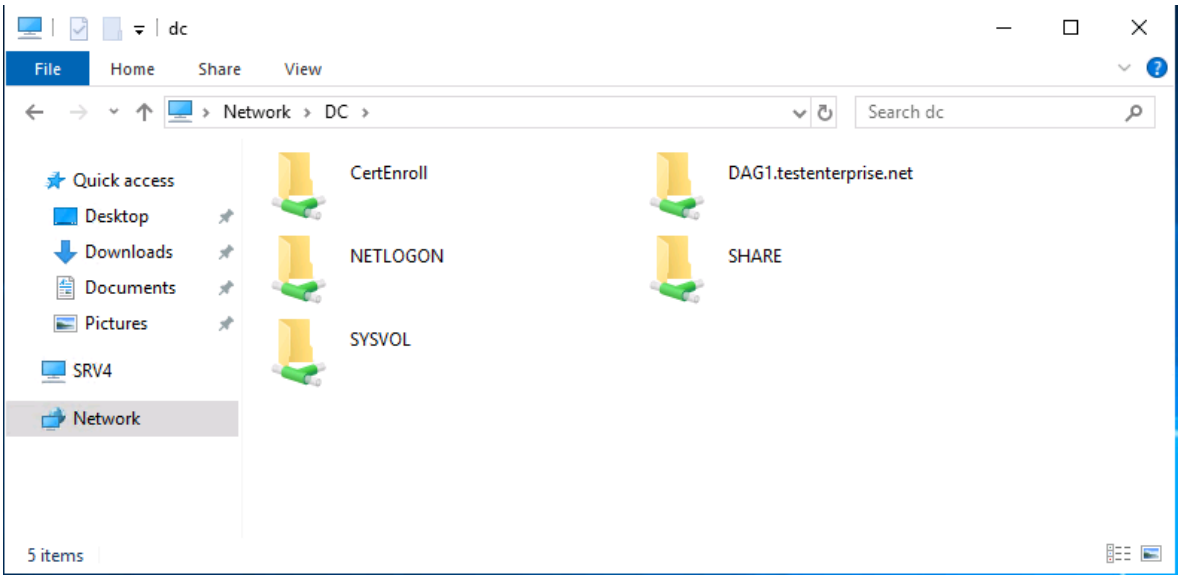
a) RestrictSERVERLogon GPO current settings – \\DC is not accessible:



b) after deleting Domain Admins and Enterprise Admins fromthe list…



…and restarting (as GPUPDATE /FORCE did not work for the user configuration either) the domain controller was accessible again:



I've seen this problem on the two servers, but after removing and re-adding Domain Admins and Enterprise Admins to the list of banned accounts the problem did not reappear. This problem just proves once again that the tiered AD infrastructure should be implemented step by step, ideally, implementing one GPO (or even a GPO setting) at a time and testing the new configuration before proceeding to other settings/parameters.