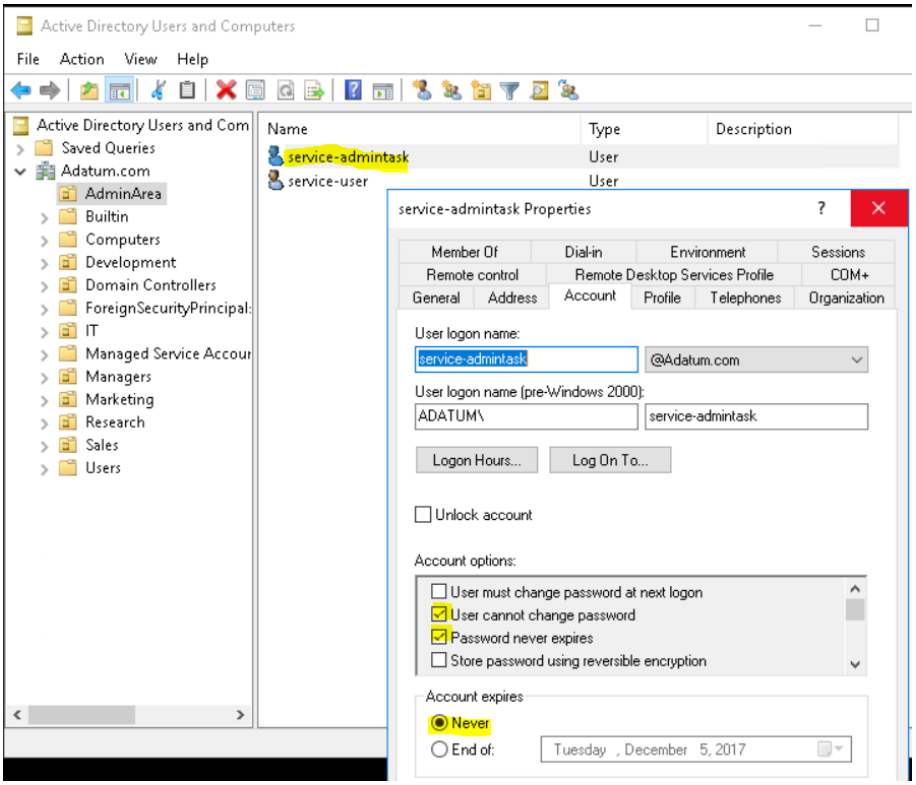
Inhalt

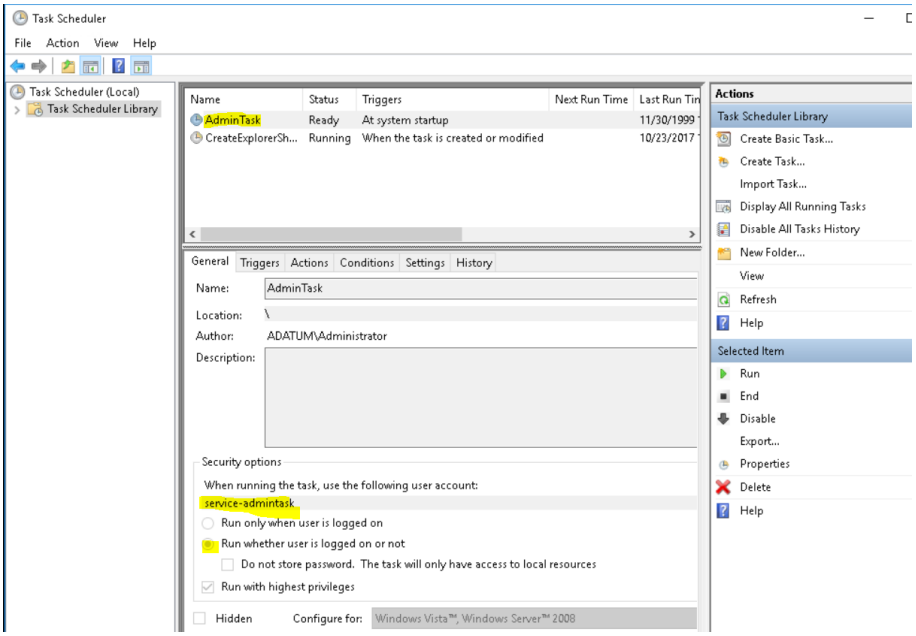
1. Automation durch Benutzer mit statischen Kennwörtern

Scheduled-Tasks

Um wiederkehrende Aufgaben zu automatisieren bieten sich geplante Aufgaben an. Diese müssen in einem Sicherheitskontext ausgeführt werden. Für viele Aufgaben genügt der lokale Systemkontext. Aber wenn eine Aufgabe rechnerübergreifend arbeiten muss oder auf bestimmte Services zugreifen soll, dann muss ein Benutzeraccount verwendet werden. Dieser wird dann etwa so aussehen:



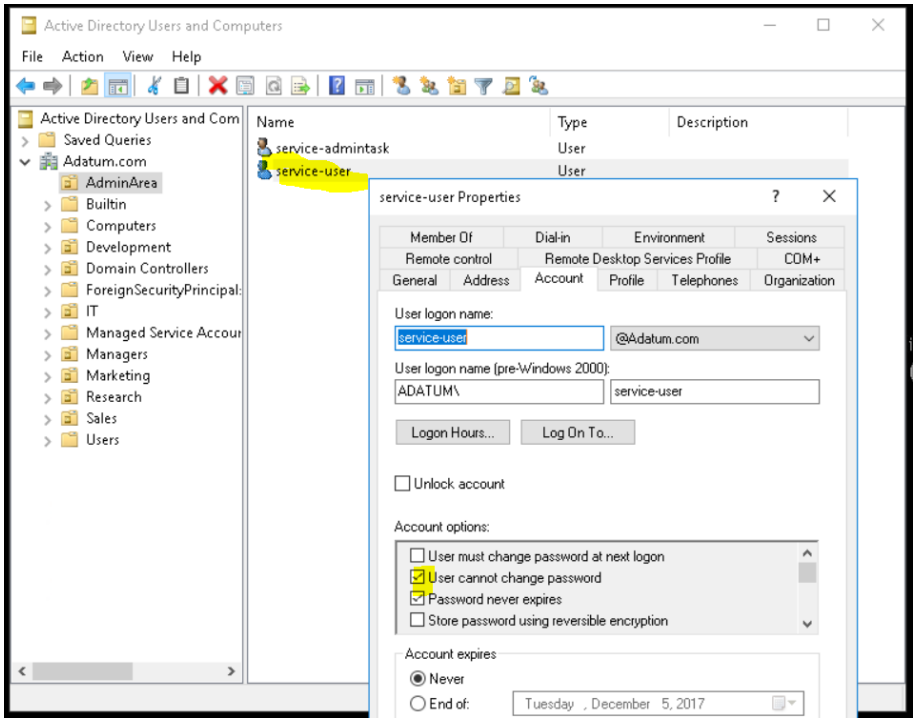
In der Aufgabenplanung wird der Benutzer dann mit seinem Passwort hinterlegt:



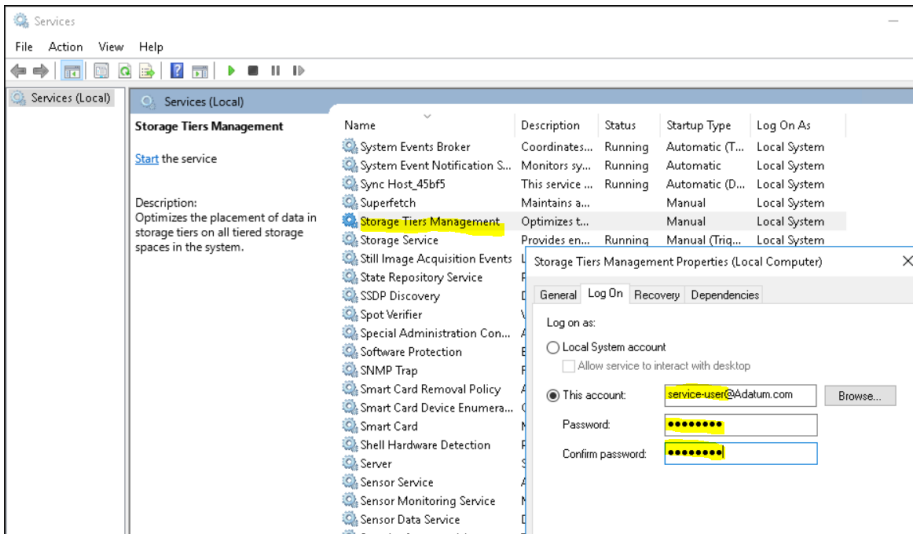
Das Kennwort darf nicht einfach ablaufen und ungültig werden, da eine geplante Aufgabe sonst nicht mehr ausgeführt wird…

Services

Mit Diensten ist es meist genauso. Wenn ein Benutzerkontext der Domäne benötigt wird, findet man eben solche Benutzer im AD:



Und in der Dienstekonsole wird dieser Benutzer dann hinterlegt:



Kommt euch das etwa bekannt vor?

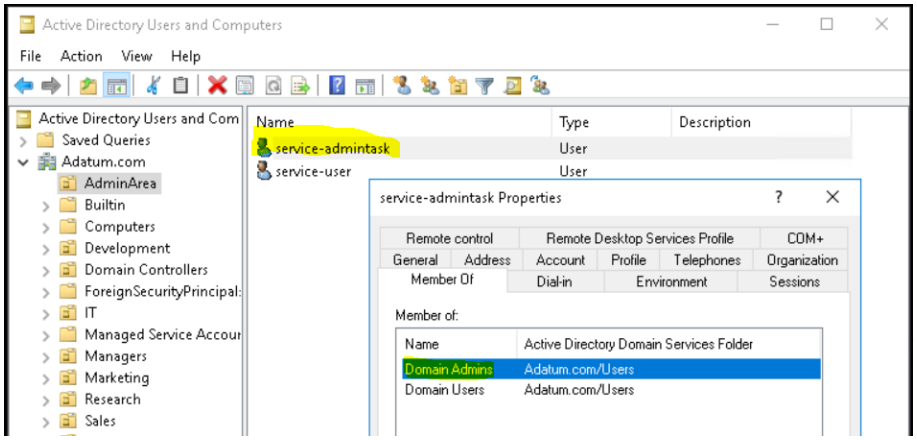
Wo ist das Problem?

Es liegt auf der Hand: ein Benutzer im AD mit statischem Kennwort (der zumeist auch nicht überprüft wird) ist für Angreifer eine lohnende Beute. Kapert er einmal dessen Anmeldeinformationen, dann kann er sie nahezu endlos verwenden.

Für jeden anderen Benutzer im AD würde eine Kennwortrichtlinie vorschreiben, dass dessen Kennwort regelmäßig geändert wird. Nur wie will man das mit den vielen Service-Usern unternehmensweit vornehmen? Meist weiß nach Monaten niemand mehr, wo der Useraccount eingesetzt wird.

Ebenso habe ich auch schon oft (genug) verwaiste Service-User gefunden, die nirgends mehr eingesetzt wurden. Dennoch traute sich niemand, den Service-User zu löschen. Wer weiß schon, ob er nicht doch noch eingesetzt wird??

Und wenn dann die Benutzerkonten noch erhöhte Rechte haben, dann ist das Durchfallen beim nächstem Pentest fast garantiert:



2. Was sind gMSA?

Definition

Group Managed Service Accounts – kurz gMSA – sind die Lösung für diese Probleme. Es sind Benutzerkonten-ähnliche Objekte im Active Directory, die durch Gruppenmitgliedschaften berechtigt werden können. Deren Kennwort wird aber regelmäßig geändert!

Dafür braucht ihr „nur“ ein AD mit der Funktionsebene WindowsServer2012 (ohne R2 genügt).
Den Vorgänger „Managed Service Account“ beleuchte ich hier nicht, denn das war leider nix… 😊

Wie funktioniert ein gMSA?

Es sind folgende Arbeitsschritte notwendig:

1. Das AD muss einmalig vorbereitet werden. Neben der richtigen Domänen-Funktionsebene ist ein KDS-RootKey erforderlich. Dieser wird von den DCs benötigt, um die Kennworte zu generieren und zu schützen. Dazu gleich mehr!

2. Dann kann der erste gMSA erstellt werden. Leider gibt es auch mit Windows Server 2016 keine Möglichkeit, dies über die grafische Oberfläche zu erledigen. Es bleibt nur die PowerShell. Beim Erstellen des gMSA wird angegeben, welche Domänen-Computer den Account verwenden dürfen.

3. Auch Gruppenmitgliedschaften können über die PowerShell konfiguriert werden.

4. Zuletzt wird der Account als Service-User oder als Task-User auf den Zielservern eingesetzt.

Doch wie funktioniert nun der Kennwortaustausch?

1. Beim Erstellen eines gMSA vergibt der DC ein komplexes Kennwort.
2. Wird ein gMSA auf einem Server eingesetzt, der berechtigt ist, diesen gMSA zu verwenden, dann erfragt der Server über den SecureChannel beim DC das aktuelle Kennwort und speichert es.
3. Der DC überwacht das Kennwortalter aller gMSA. Läuft von einem gMSA das Kennwort aus, dann wird der DC dieses ändern.
4. Verliert danach auf einem Server das TGT (Ticket Granting Ticket) des eingesetzten gMSA seine Gültigkeit, dann wird der Server dessen Anmeldung erneuern. Dabei kontaktiert er den DC über den SecureChannel und erfährt das neue Kennwort.

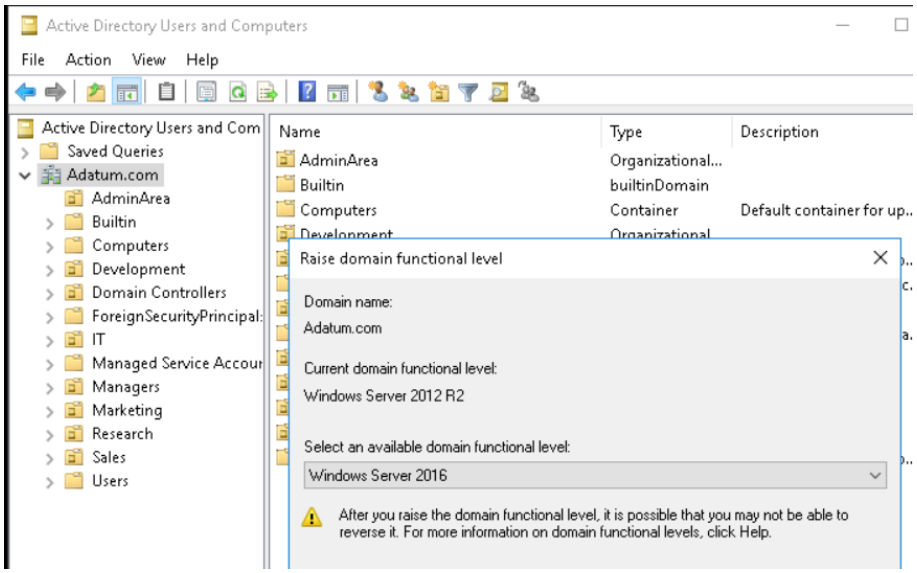
So einfach ist die Idee. 😊

Ein gMSA kann von einem oder von mehreren AD-Computern verwendet werden. Ebenso kann ein AD-Computer einen oder mehrere gMSA verwenden.

3. Administration mit gMSA-Accounts ohne mein Script

Einrichtung eines gMSA

Zunächst muss das AD vorbereitet werden:



Es wird ein KDS-RootKey benötigt. Dieser kann mit der PowerShell erstellt und geprüft werden. Ist kein Key vorhanden, dann bleibt die PS-Ausgabe leer:

```
PS C:\Users\Administrator> Get-KdsRootKey

PS C:\Users\Administrator>
```

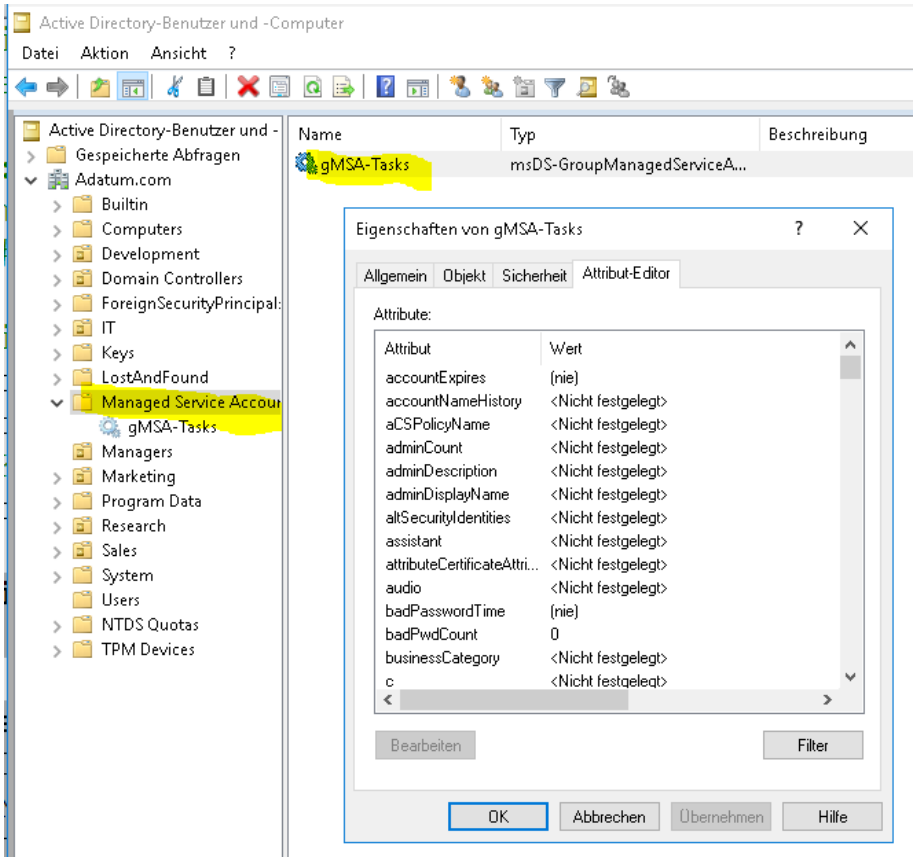
Einen neuen Key kann man in LAB-Umgebungen von seiner Gültigkeit her zurückdatieren. In Produktionsumgebungen würde ich den Parameter EffectiveImmediately empfehlen und abwarten, bis der Key repliziert wurde:

```
PS C:\> Add-KdsRootKey -EffectiveTime (get-date).AddDays(-10)

Guid
----
8fc4bb35-f4a8-b1a1-d182-953a64969c0a


PS C:\>
```

Ist das AD bereit, dann kann man den ersten gMSA erstellen. Dabei muss neben dessen Namen auch mindestens ein Server angegeben werden, der das Kennwort abfragen darf:

```
1  $Domain = (Get-ADDomain).dnsroot
2  $gMSA   = 'gMSA-Tasks'
3
4  $MemberServer = 'LON-DC1','LON-SVR4' | Get-ADComputer
5  New-ADServiceAccount `
6      -Name $gMSA `
7      -DNSHOSTNAME "$gMSA.$Domain" `
8      -PrincipalsAllowedToRetrieveManagedPassword $MemberServer
```

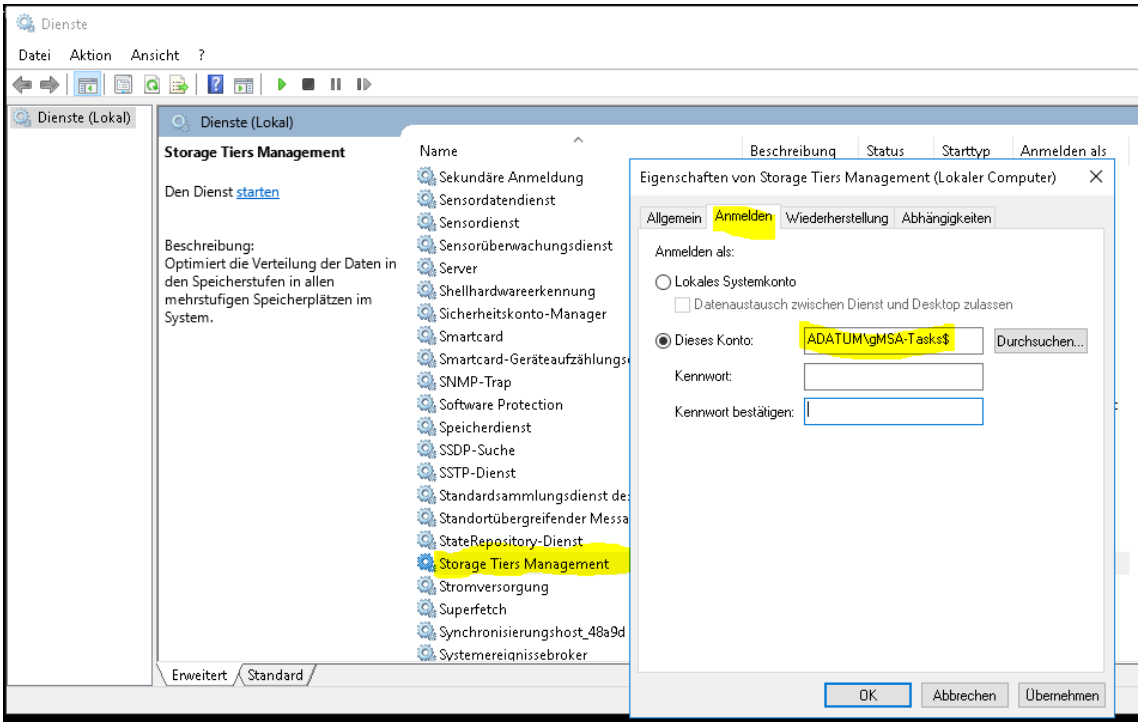
In Zeile 1 frage ich die aktuelle Domain ab. Zeile 2 speichert den Namen des neuen Accounts. Zeile 4 ermittelt die Computerkonten von 2 Servern. Und Zeile 5++ erstellt den gMSA. Hier sieht man das Ergebnis in dsa.msc:



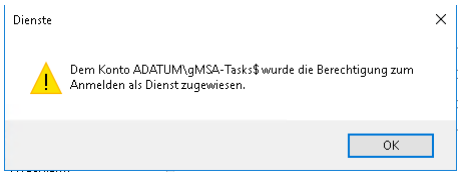
Jede Änderung wird über die PowerShell vorgenommen: in der MMC gibt es dafür keine Funktionen… Selbst der Attributeditor ist nur bedingt hilfreich. ☹

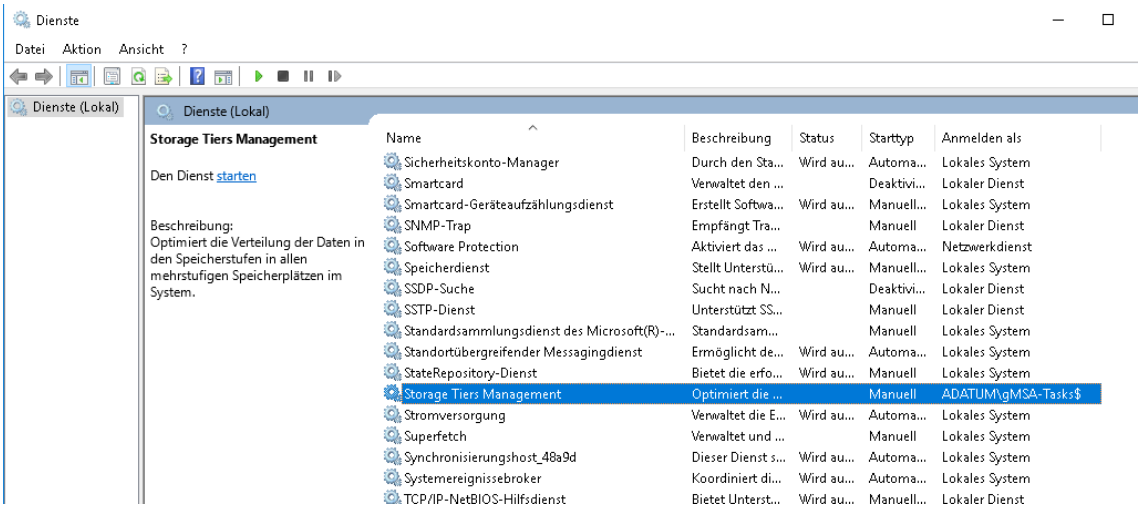
Konfiguration eines Services mit gMSA

Wie bekomme ich nun den Account als Service-User eingetragen? In der klassischen Services.msc gibt es die Option „Anmelden“:



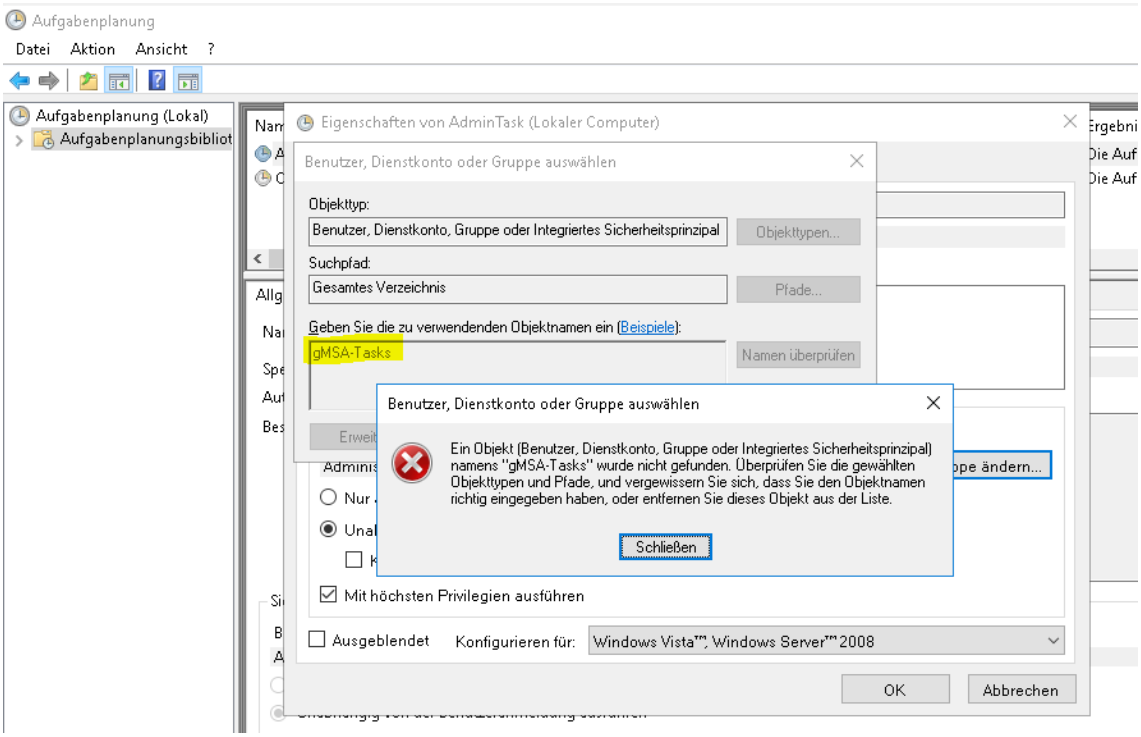
Das Kennwort ist nicht bekannt, denn das wird ja vom KDC verwaltet. Der Trick in der MMC besteht darin, das Kennwortfeld leer zu lassen. Eine Erweiterung der Berechtigung später ist der Account bereit:





Konfiguration eines Scheduled-Tasks mit gMSA

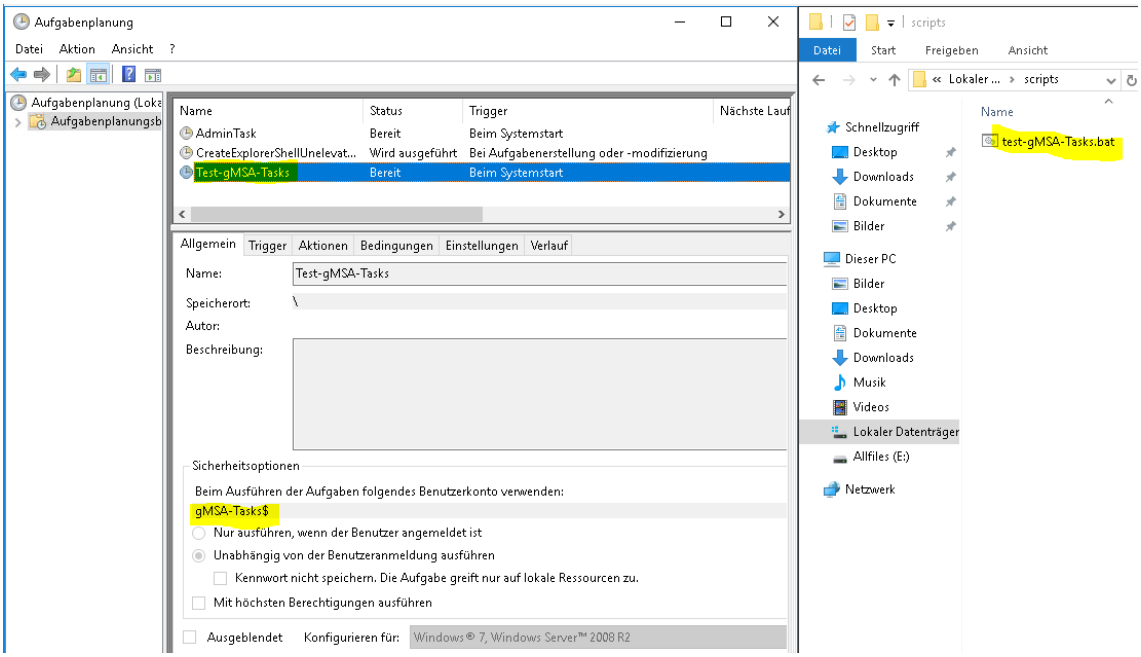
Diese Änderung ist deutlich schwerer, denn die MMC findet den Benutzeraccount nicht:



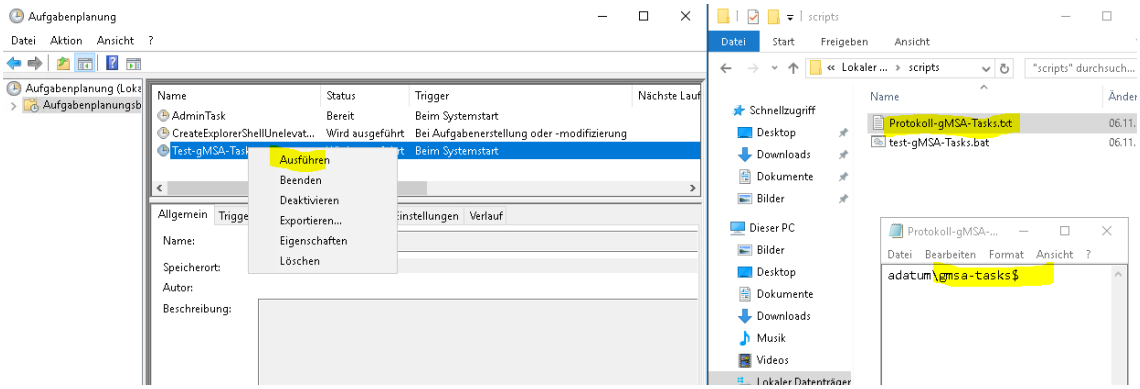
Da hilft nur die PowerShell:

```
1  $action    = New-ScheduledTaskAction  "c:\scripts\test-$gMSA.bat"
2  $trigger   = New-ScheduledTaskTrigger -AtStartup
3  $principal = New-ScheduledTaskPrincipal -UserID "$Domain\$gMSA$" -LogonType Password
4  Register-ScheduledTask `
5          -TaskName "Test-$gMSA" `
6          -Action $action `
7          -Trigger $trigger `
8          -Principal $principal `
9          -ErrorAction SilentlyContinue |
10     Out-Null
11
12 New-Item -Path "c:\scripts"  -ItemType directory -ErrorAction SilentlyContinue | Out-Null
13 "whoami > c:\Scripts\Protokoll-$gMSA.txt" |
14     Out-File -FilePath "c:\scripts\test-$gMSA.bat" -Encoding default
```

Dieser Code erstellt eine neue Aufgabe, die von dem gMSA-Account beim Systemstart ein Script startet. Das Script wird samt Verzeichnis ab Zeile 12 erstellt:



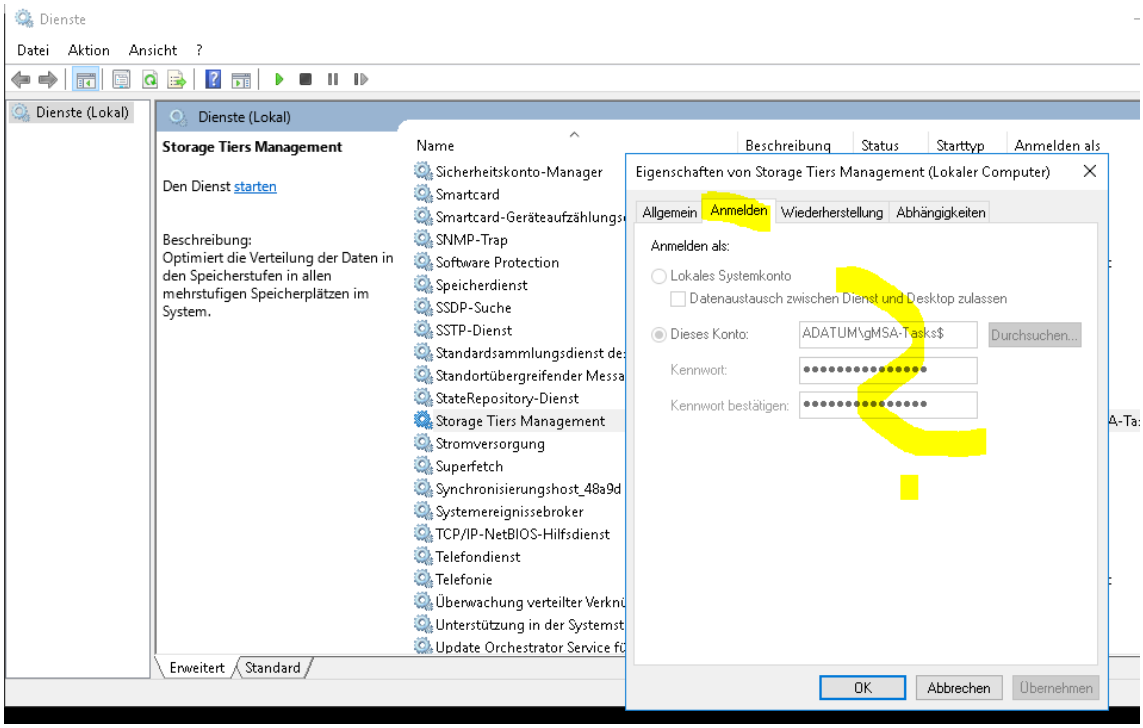
Beim manuellen Start der Aufgabe wird dann die batch-Datei ein whoami in eine Textdatei umleiten. Und hier sieht man den aktiven gMSA:



Das Problem

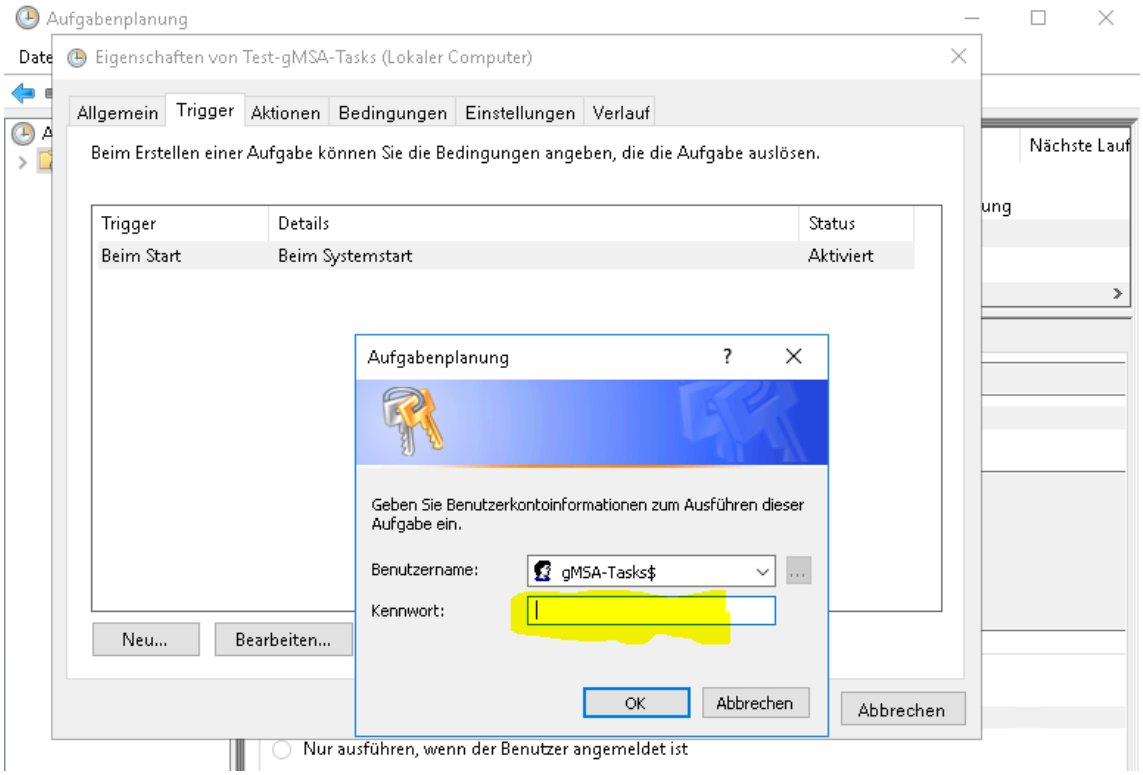
Das passt doch soweit. Wo ist nun die Problematik? Ganz einfach:

- ändert doch einmal den Benutzer des Services:

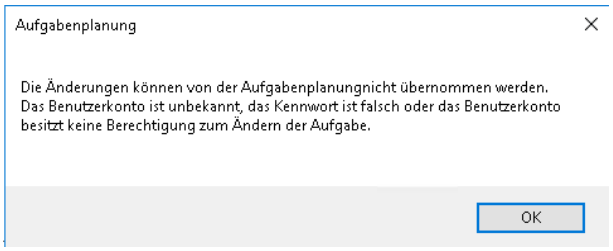


Der Dialog kann nicht erneut aufgerufen werden!!!

- ändert doch einmal die Aufgabe:



Das Kennwort kann eben NICHT leergelassen werden:



Änderungen sind also nur durch die PowerShell möglich… Und das schließt auch die Erweiterung der Server mit Passwortleserecht und die Änderung der Gruppenmitgliedschaften mit ein.

So wird das bestimmt nichts mit gMSA!!!

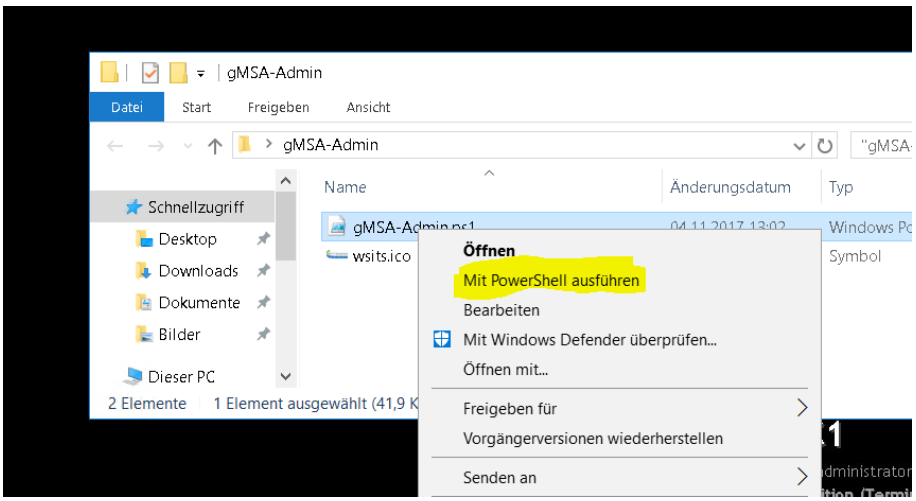
4. Einsatz des gMSA-Admin-Scriptes

Die Lösung

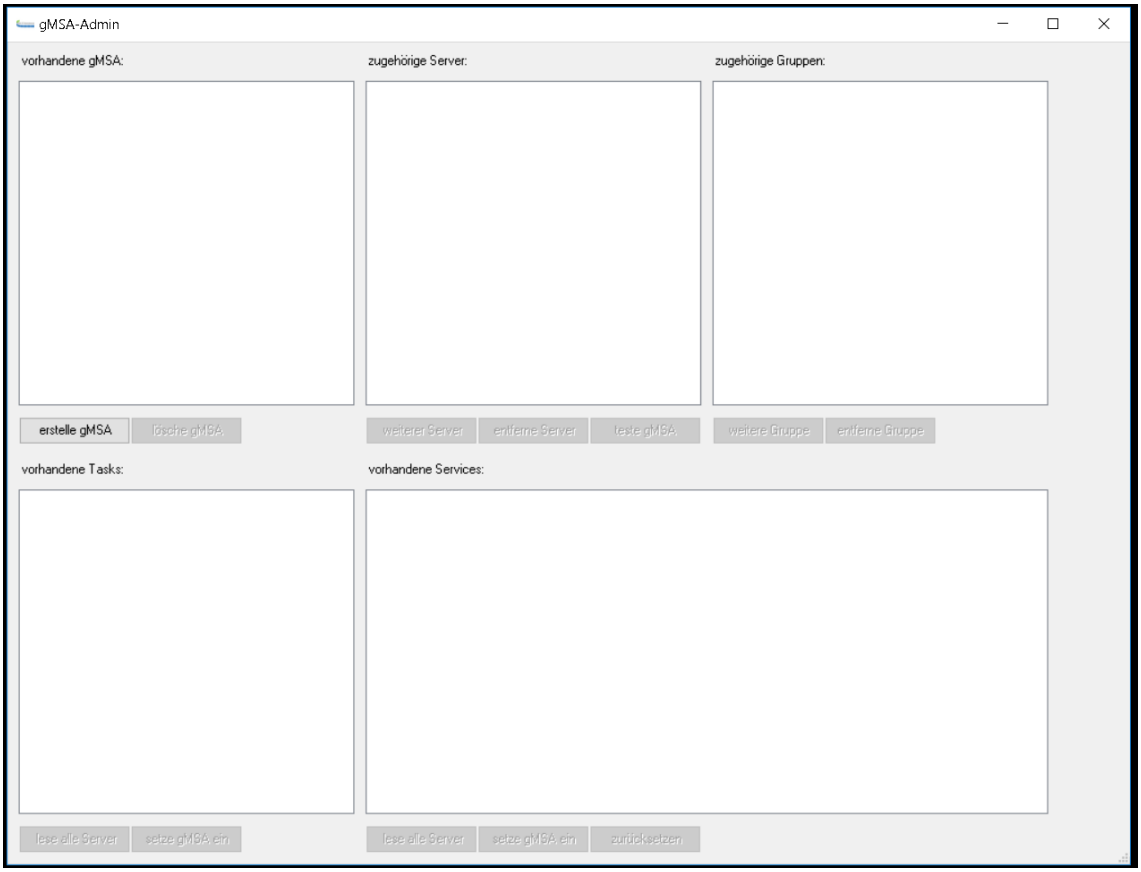
Ich habe für genau diese Aufgabenstellung ein kleines PowerShell-Script geschrieben. Dieses beinhaltet die erforderlichen Funktionen zur gMSA-Administration:

- Erstellen eines gMSA
- Löschen eines gMSA
- Änderung der Server für die gMSA-Verwendung (hinzufügen und entfernen)
- Änderung der Gruppenmitgliedschaft eines gMSA
- Konfiguration einer bestehenden Aufgabe für gMSA
- Konfiguration eines bestehenden Services für gMSA – inklusive Entfernen des gMSA
- Vorbereitung des AD
- Testen eines gMSA (erstellt eine Aufgabe, bindet den gMSA ein startet die Aufgabe und löscht sie wieder)

Und das ganze mit grafischer Oberfläche! So wird es gestartet:

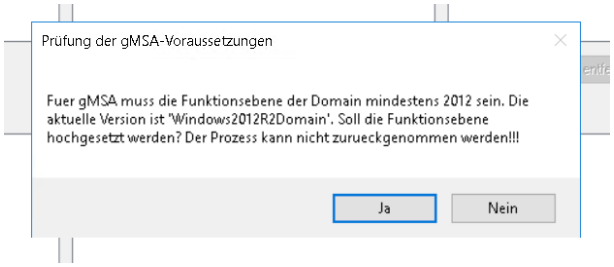


Und das ist die GUI:

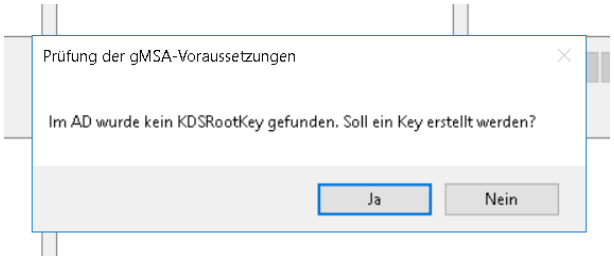


Einrichtung eines gMSA mit Vorbereitung der Domäne

Sollte die Funktionsebene der Domäne niedriger sein als Windows Server 2012, dann wird das Script nachfragen, ob es die Erhöhung durchführen darf.

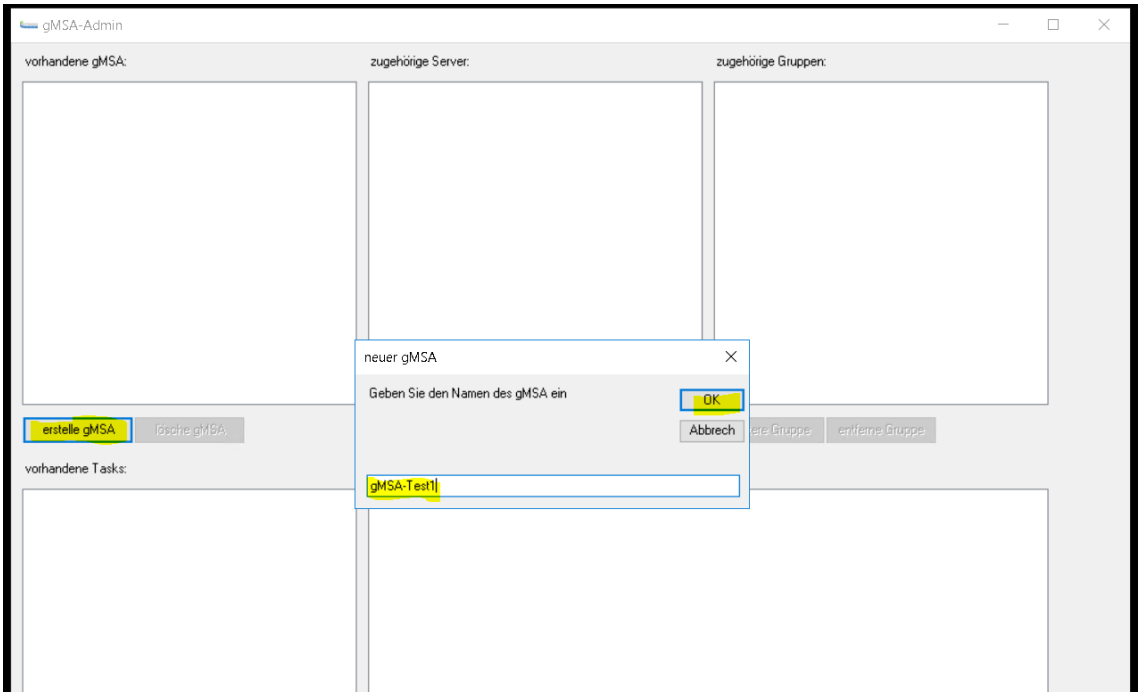


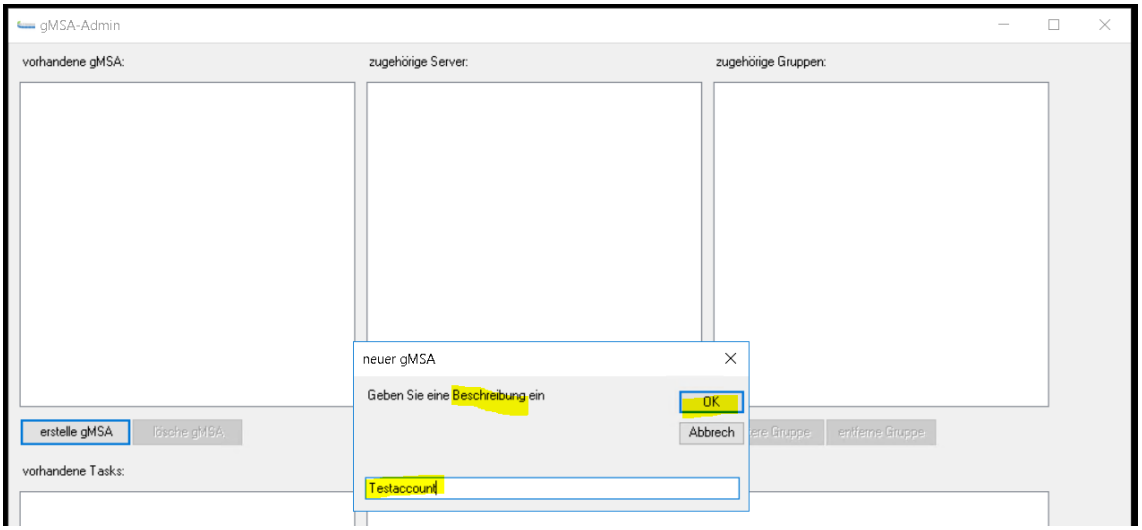
Sollte das Script keinen KDS-Rootkey finden, dann wird es fragen, ob es einen erstellen darf:



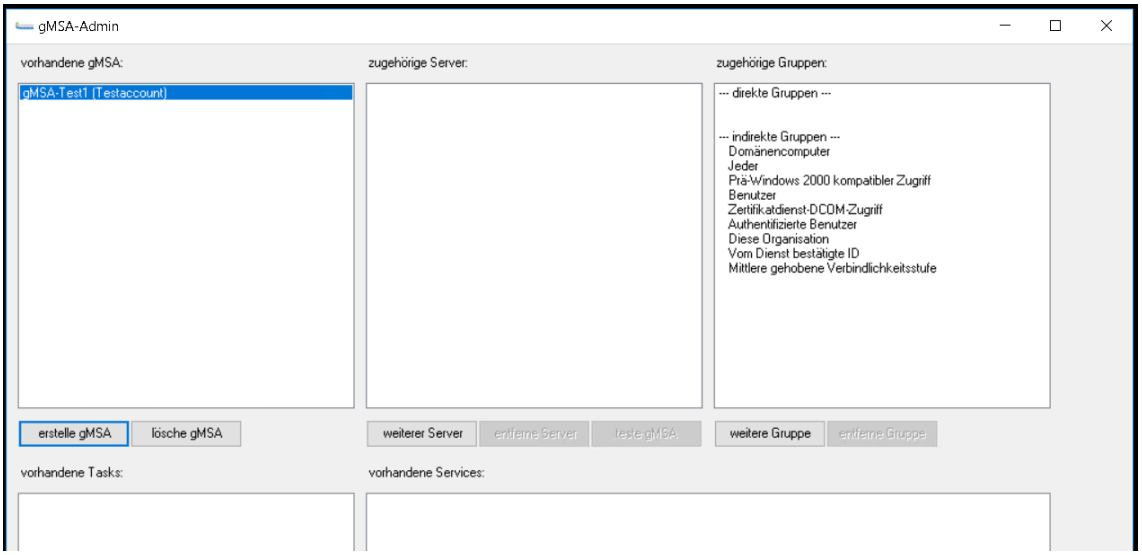
Wenn die Voraussetzungen nicht erfüllt sind, dann bleiben alle Schalter deaktiviert.

Über den Schalter „neuer gMSA“ kann mit 2 Eingaben ein neuer gMSA erzeugt werden:

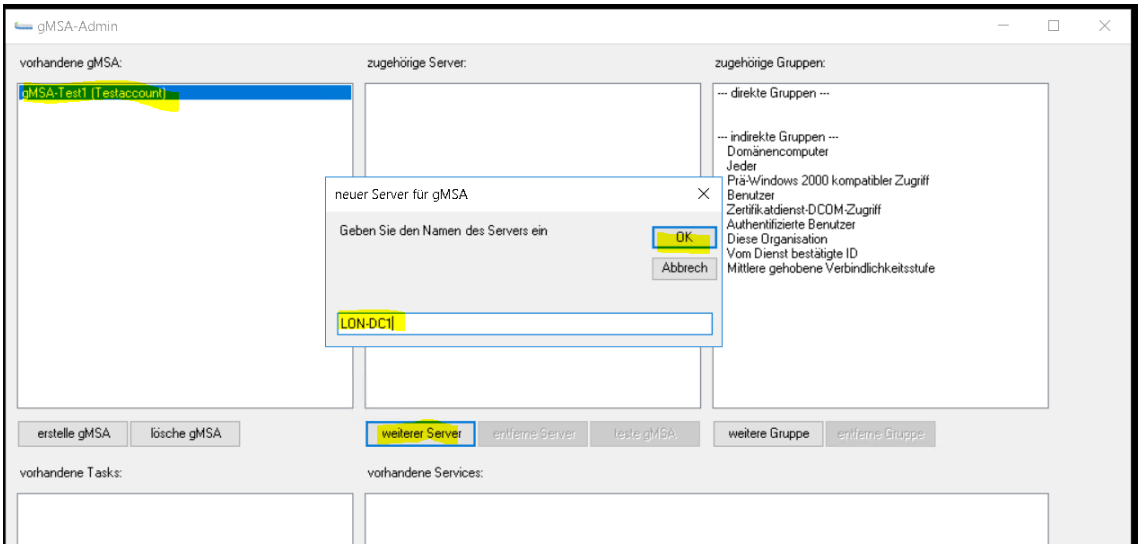




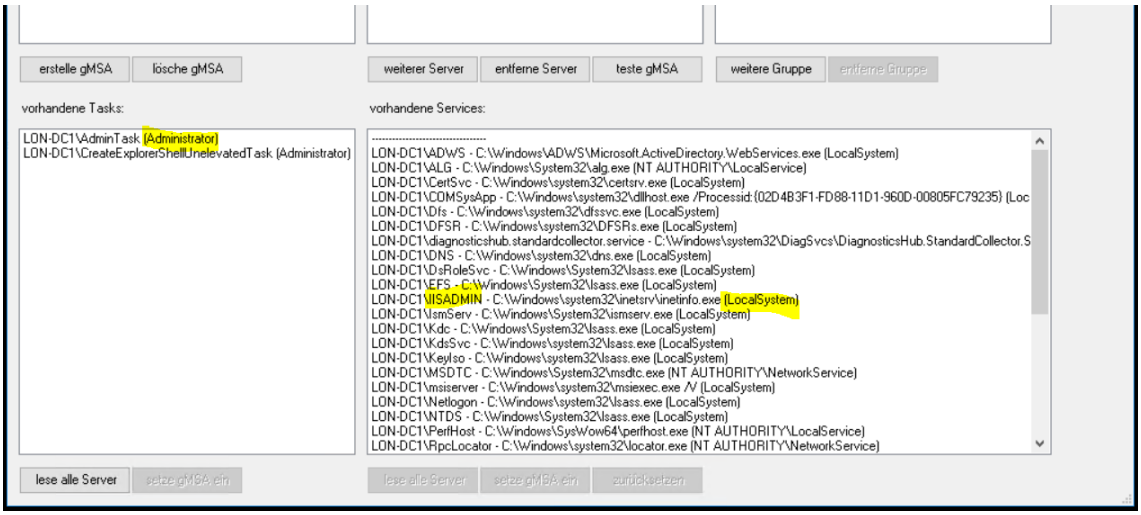
Ist der Account erzeugt, dann wird er direkt ausgewählt. Zu dem selektierten gMSA werden alle weiteren Felder aktualisiert:



Nun können Server an den gMSA gebunden werden:

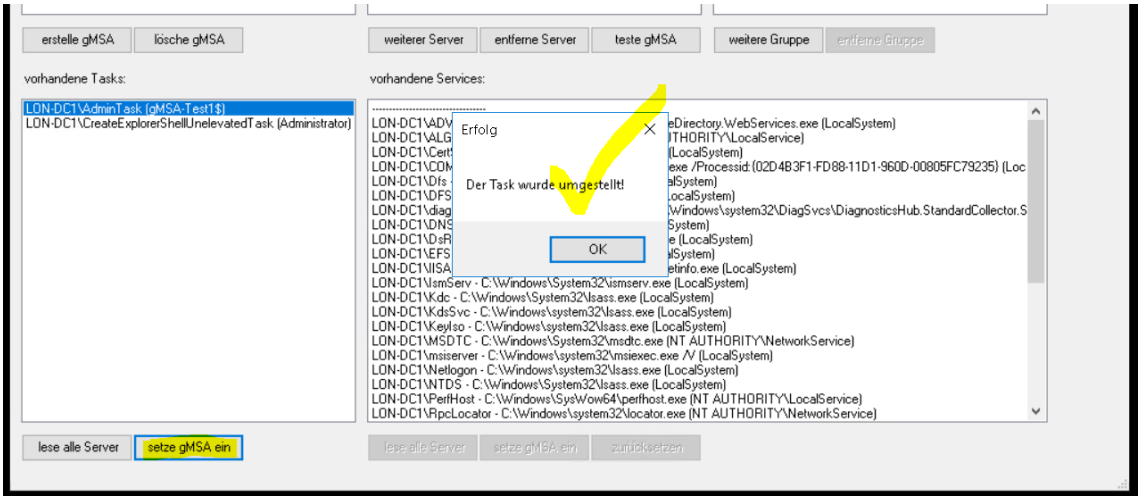


Für den ausgewählten Server wird dann ein Powershell-Remoting versucht. Ist dieses erfolgreich, dann werden in den unteren Boxen alle Aufgaben und Dienste des Servers angezeigt:

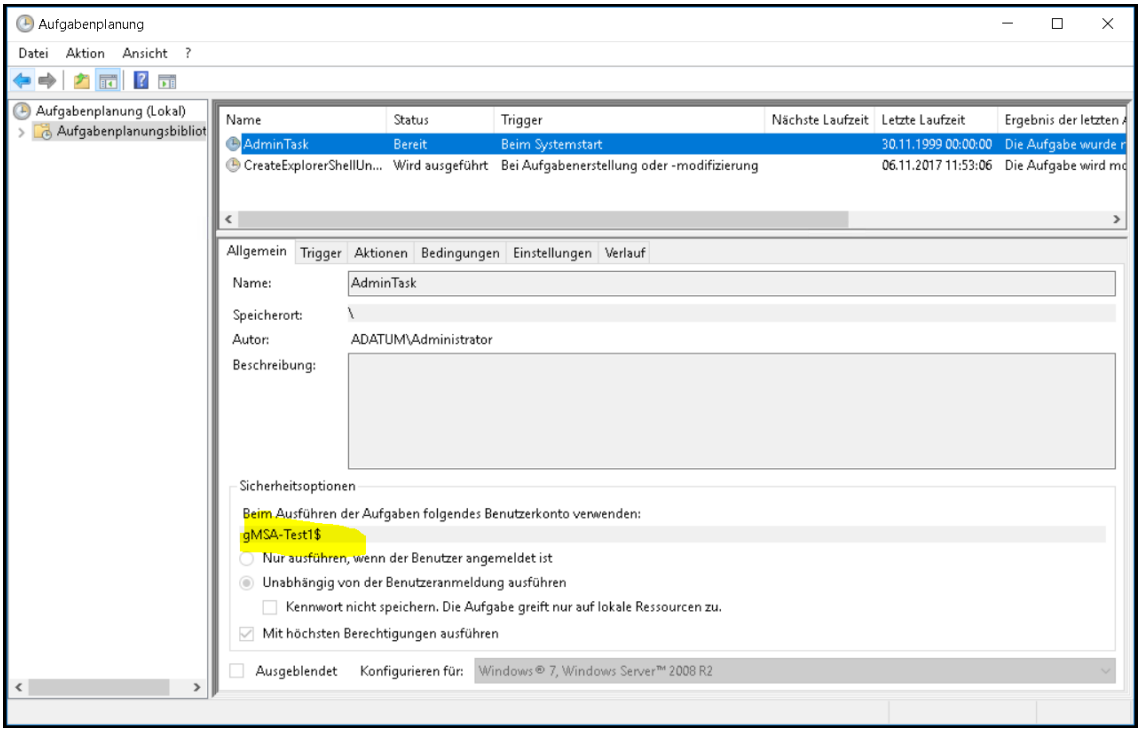


Konfiguration eines Scheduled-Tasks mit gMSA

Eine fertige Aufgabe in der Aufgabenplanung kann nun einfach ausgewählt werden. Mit dem Schalter „setze gMSA ein“ wird dann der gewählte gMSA auf dem gewählten Server aus Account für die gewählte Aufgabe geschrieben. Wenn das funktioniert wird der Erfolg mitgeteilt:



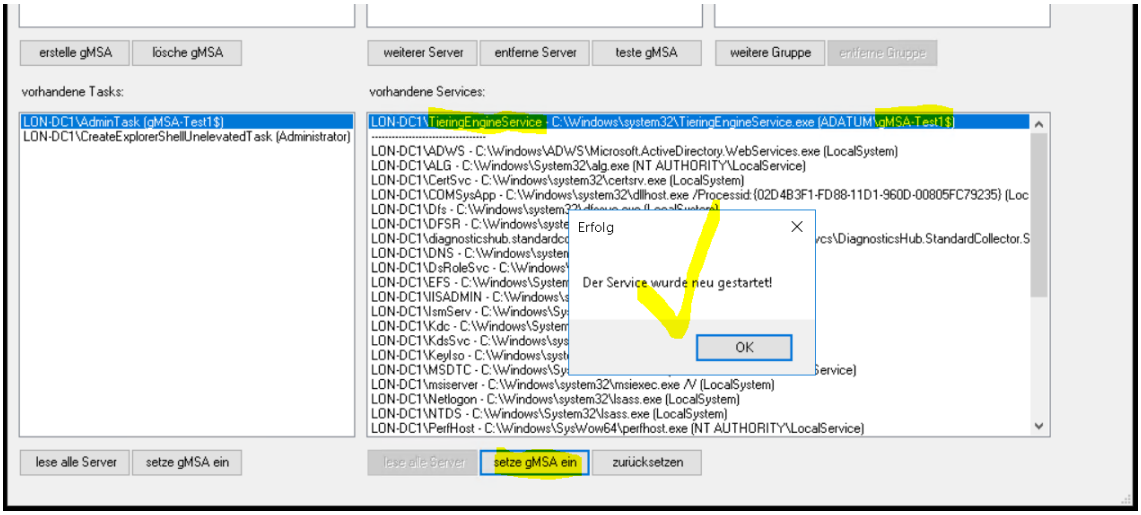
In der Aufgabenplanung kann das Ergebnis geprüft werden:



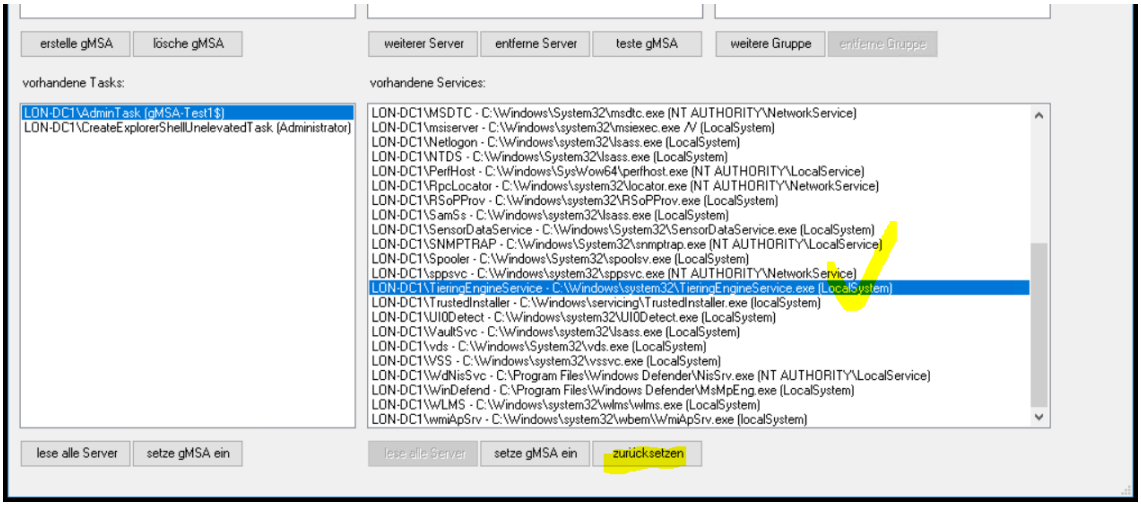
Wenn eine Aufgabe verändert werden muss, dann trägt man temporär einen Benutzer als Konto ein, ruft den gMSA-Admin auf und überschreibt den User mit dem passenden gMSA. 😊

Konfiguration eines Services mit gMSA

Dienste können ebenso leicht modifiziert werden: such den Service aus der Liste aus und klicke auf „setze gMSA ein“. Der Dienst wird danach in der Liste nach oben einsortiert:

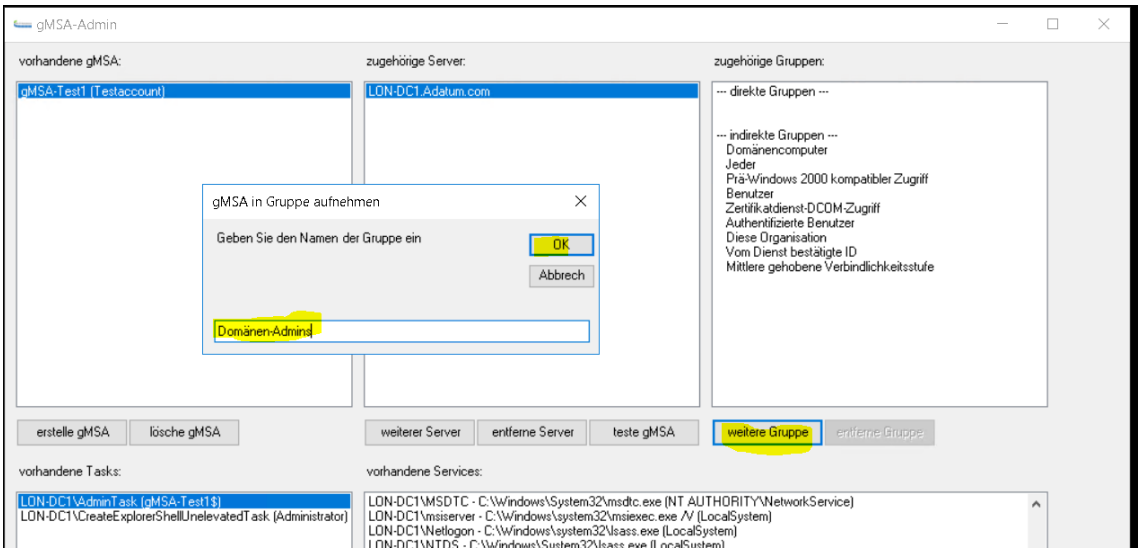


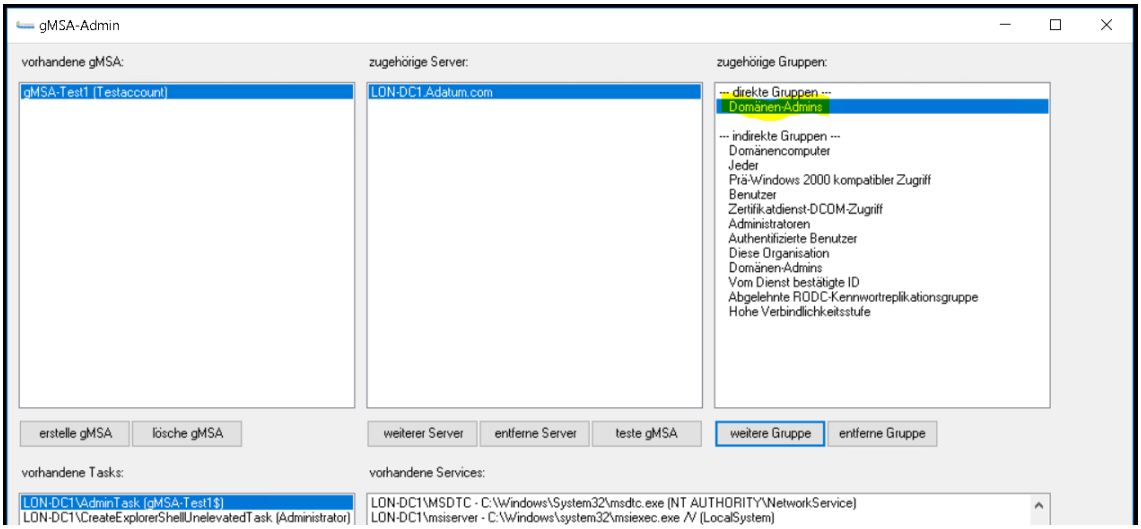
Keine Lust mehr auf gMSA? Dann wähle den Service aus und klicke auf Zurücksetzen:



Konfiguration von Berechtigungen eines gMSA

Der gMSA benötigt höhere Rechte? Kein Problem:





Zusammenfassung

Das Script steht für euch in der ersten Generation bereit und wurde bereit in mehreren Produktionsumgebungen getestet. Nun gibt es keine Ausreden mehr! Viel Spaß beim Evaluieren!!!