

## Inhalt

Die aktuelle Situation in vielen Infrastrukturen .....	2
Privilegierte Benutzer 24/7? .....	2
Lösungsansatz MIM? .....	2
Die Lösung mit neuen Funktionen in Windows Server 2016 .....	2
Privileged Access Management .....	2
Just Enough Administration.....	3
Das Problem mit PAM und JEA .....	3
Die WS-ITS Lösung: PAM-AdminGUI .....	3
Voraussetzungen .....	3
Vorstellung der Dateien .....	4
Installation und Konfiguration.....	4
Vorbereitung vom Active Directory .....	4
Vorbereitung der Dateien.....	5
Setup von PAM.....	6
Setup von JEA .....	7
Vorbereitung des Clients .....	9
Betrieb .....	10
Test – Nutzung des AD-Accounts ohne Berechtigung .....	10
Test – Nutzung des AD-Accounts nach dem Verwenden der PAM-AdminGUI.....	11
Details.....	13
Das JEA-Logging.....	13
Die JEA-ProxyFunktionen.....	14
JEA-VirtualAccounts.....	15
Was passiert nach dem Ablauf der MemberTimeToLive? .....	15

## Die aktuelle Situation in vielen Infrastrukturen

### Privilegierte Benutzer 24/7?

In den vergangenen Jahren entwickelte sich ein Standard, der IT-Infrastrukturen schützen soll: administrative Benutzer verwenden 2 Benutzerkonten:

- einen normalen Benutzeraccount, der die Anmeldung an der Arbeitsstation ermöglicht, die Mails abrufen kann und im Internet unterwegs ist
- einen AD-Account (AdminAccount). Dieser ist hochberechtigt und wird für administrative Tätigkeiten auf Servern und Clients verwendet.

Die Idee ist schon mal nicht verkehrt. ABER:

- Der AD-Account hat hohe Rechte. Und diese hat er rund um die Uhr!
- Der AD-Account könnte sich auf einem kompromittierten System anmelden und dort könnten seine Berechtigungen missbraucht werden

Mit klassischen Boardmitteln konnte nur die Vorsicht des Admins schützen...

### Lösungsansatz MIM?

Mit Forefront Identity Management (FIM – RIP...) und nun Microsoft Identity Management 2016 (MIM) kann eine Lösung aufgebaut werden. Diese Lösung stellt über eine Vielzahl von Schnittstellen und Services einen Workflow bereit, mit dem Benutzeraccounts für eine definierte Zeit eine erhöhte Berechtigung erhalten können.

Dafür sind „nur“ folgende Komponenten erforderlich:

- Ein SharePoint-Server (ideal als Farm (2+) mit entsprechender hochverfügbarer SQL-Anbindung)
- Ein separates Active Directory (mind. 2 DCs) als Bastion-AD (gut, es ginge auch ohne)
- ein MIM-Server

Abgesehen vom Preis für die Beschaffung und den Betrieb ist das Konzept dahinter nicht verkehrt. Aber für viele Umgebungen wird das einfach zu groß sein... ☹

## Die Lösung mit neuen Funktionen in Windows Server 2016

### Privileged Access Management

Wenn alle DCs in der Gesamtstruktur auf Windows Server 2016 oder höher umgestellt wurden, dann steht ein neues Feature im AD zur Verfügung: Privileged Access Management.

Dieses bietet eine ähnliche Funktion wie der MIM: ein Benutzer kann als Gruppenmitglied auf Zeit konfiguriert werden. Während dieser Zeit kann er seine Aufgaben ausführen. Ist die Zeit abgelaufen, dann wird er vom AD automatisch aus der Gruppe entfernt. Wie cool ist das denn? 😊

Damit der Benutzer durch eine aktive Anmeldung nicht einfach weiter administriert wird zusätzlich noch das Kerberos Ticket Granting Ticket (TGT) von der Gültigkeitszeit her auf den Zeitraum der temporären Gruppenmitgliedschaft beschränkt. Der User kann danach also nicht mehr weiterarbeiten.

PAM ist aber nur die funktionale Implementierung der ablaufenden Gruppenmitgliedschaft. Der Workflow für die Aufnahme in die gewünschte Gruppe ist damit nicht abgebildet (Das wäre beim MIM eben in der Sharepoint-Seite der Fall). Ebenso fehlt eine Protokollierung der Aufnahme eines Benutzers in eine Gruppe.

Und leider kann aktuell keine grafische Oberfläche von Microsoft einen Benutzer temporär in eine Gruppe aufnehmen. Oder eine temporäre Mitgliedschaft auch als solche anzeigen.

Raten wir einmal, wie denn das sonst funktionieren könnte ... klar: mit der PowerShell!

Nur da wird's knifflig: WER soll den meinen AD-Account (ohne interessante Gruppenmitgliedschaften) mit mehr Rechten versehen? Ein Domänen-Admin? Toll, dann würde ein „Workflow“ wie folgt aussehen:

- Ein Admin ist mit seinem Standardbenutzer am Client angemeldet.
- Ein anderer (privilegierter Admin) verbindet sich mit dem DC als Domänen-Admin und führt dort den Powershell-Befehl für PAM aus. Der AD-Account des Admins ist nun privilegiert
- Der Admin meldet sich mit seinem AD-Account am Zielsystem an und legt los...

Das wird so bestimmt nichts. Vor allem, wenn es nur wenige Admins gibt. Im Worst-Case muss sich der Admin selbst als Domänen-Admin am DC anmelden um dann seinen AD-Account zu elevaten... Klar! Dann nimmt er doch gleich seinen Dom-Admin-Account für alles.

### Just Enough Administration

Da kann nun JEA helfen. Die Idee dahinter ist folgende:

- Auf einem System wie einem DC kann sich nicht jeder anmelden. Und das betrifft nicht nur die lokale Anmeldung oder RDP, sondern auch das PowerShell-Remoting.
- Auf diesem System möchte ich nun aber ganz bestimmte Aufgaben an wenig privilegierte Benutzer delegieren. Über eine Rollenbeschreibung kann ich exakt die PowerShell-Befehle (und anderes) bestimmen, die ausgeführt werden sollen.
- Auf dem gleichen System definiere ich nun noch einen PowerShell-Endpunkt. Dieser umfasst eine Rollenbeschreibung und eine Identität – z.B. eine Gruppe mit wenig privilegierten Benutzern
- Verbindet sich nun ein solcher Benutzer mit einer PowerShell remote mit dem Server, dann kann er dort genau die Befehle ausführen, die ich vorher erlaubt habe.
- Und protokolliert wird's auch noch

Wie genial ist das denn!!!

Man könnte nun einen Endpunkt und eine Rolle definieren, mit der ein Admin als Standardbenutzer seinen AD-Account über den DC elevaten kann, ohne dass er selbst zu hohe Rechte hat.

### Das Problem mit PAM und JEA

Beide arbeiten ausschließlich mit der PowerShell. Und Microsoft hat hier nur das Framework bereitgestellt. Die Lösungen muss nun jeder für sich aufbauen.

Viel Spass übrigens bei den Microsoft Kursen zur PowerShell oder Windows Server 2016 – da werden diese beiden Punkte bestenfalls erwähnt... außer ihr kommt in meine Kurse 😊

Oder ihr verwendet gleich meine fertige Lösung: ein Script, das über JEA die temporären Gruppenmitgliedschaften steuert. Natürlich mit einer grafischen Oberfläche und richtigen ProxyFunktionen! Das ist meine PAM-AdminGUI.

## Die WS-ITS Lösung: PAM-AdminGUI

### Voraussetzungen





Die Messlatte liegt schon ordentlich hoch. Daher werden nun bestimmt viele Leser enttäuscht sein. Aber vielleicht ermutigt es euch auch, die Migration auf Active Directory unter Windows Server 2016 voranzubringen.

Was braucht man für die PAM-AdminGUI:

- Active Directory mit der Gesamtstrukturfunktionsebene Windows Server 2016 oder höher
- Im AD ist eine geeignete Ablage und Benennung von AD-Accounts und AdminGruppen sehr empfehlenswert.
- JEA sollte auf einem Windows Server 2016 laufen. Dieser benötigt die RSAT-AD-PowerShell installiert und muss natürlich Mitglied im AD sein. Ein DC ist natürlich machbar.
- Der Admin sollte das die GUI im täglichen Betrieb auf einem System starten, auf dem die PowerShell V5 installiert ist. Ein Windows 10 oder ein Windows Server 2016 sind da fein.

### Vorstellung der Dateien

Die Lösung besteht aus mehreren Dateien:

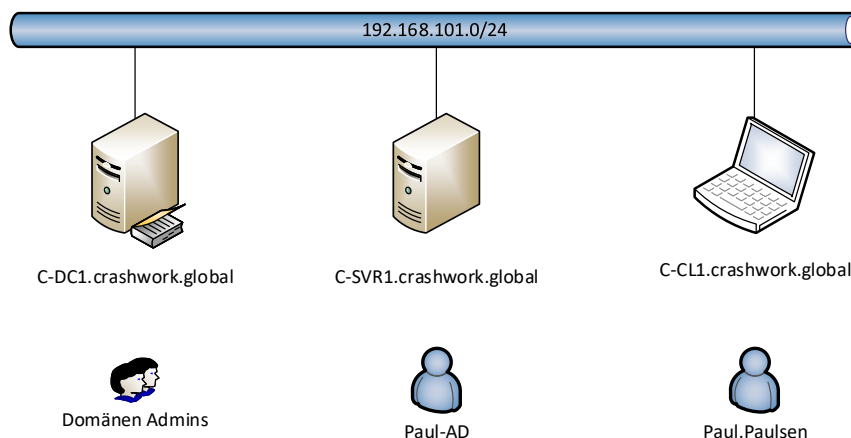
Name	Änderungsdatum	Typ	Größe
 PAM-AdminGUI.ini	15.11.2017 17:02	Konfigurationseins...	2 KB
 PAM-AdminGUI.ps1	15.11.2017 07:49	Windows PowerSh...	14 KB
 PAM-AdminGUI-Functions.ps1	14.11.2017 17:31	Windows PowerSh...	7 KB
 PAM-Tester.ps1	15.11.2017 07:49	Windows PowerSh...	2 KB
 Setup-PAM&JEA.ps1	15.11.2017 07:49	Windows PowerSh...	6 KB
 wsits.ico	18.04.2016 18:05	IrfanView ICO File	33 KB

Dabei hat jede Datei natürlich eine Aufgabe:

Datei	Funktion
PAM-AdminGUI.ini	In der ini-Datei sind einige Parameter deklarierbar. Diese werden für das Setup und später auch für den GUI-Aufruf benötigt
PAM-AdminGUI.ps1	Diese Datei gehört später auf den Computer des Admins. Sie erstellt zur Laufzeit die grafische Oberfläche und ruft über JEA die Proxyfunktionen der Datei PAM-AdminGUI-Functions.ps1 auf.
PAM-AdminGUI-Functions.ps1	Diese Datei liegt auf dem Server, der über JEA die PAM-Funktionen anbietet. Um den Prozess abzusichern erstellt die Datei sogenannte Proxyfunktionen, die ein Benutzer über JEA verwenden kann. Dazu später mehr.
PAM-Tester.ps1	Falls die GUI nicht korrekt arbeitet oder falls ihr die Funktion gerne ohne GUI sehen wollt: diese Datei enthält verschiedene Aufrufe zum Testen.
Setup-PAM&JEA.ps1	Die Einrichtung und Konfiguration von JEA und PAM wird mit diesem Script vorgenommen.
wsits.ico	Die GUI zeigt ein Logo an. Diese Datei enthält mein Firmenlogo. 😊

### Installation und Konfiguration

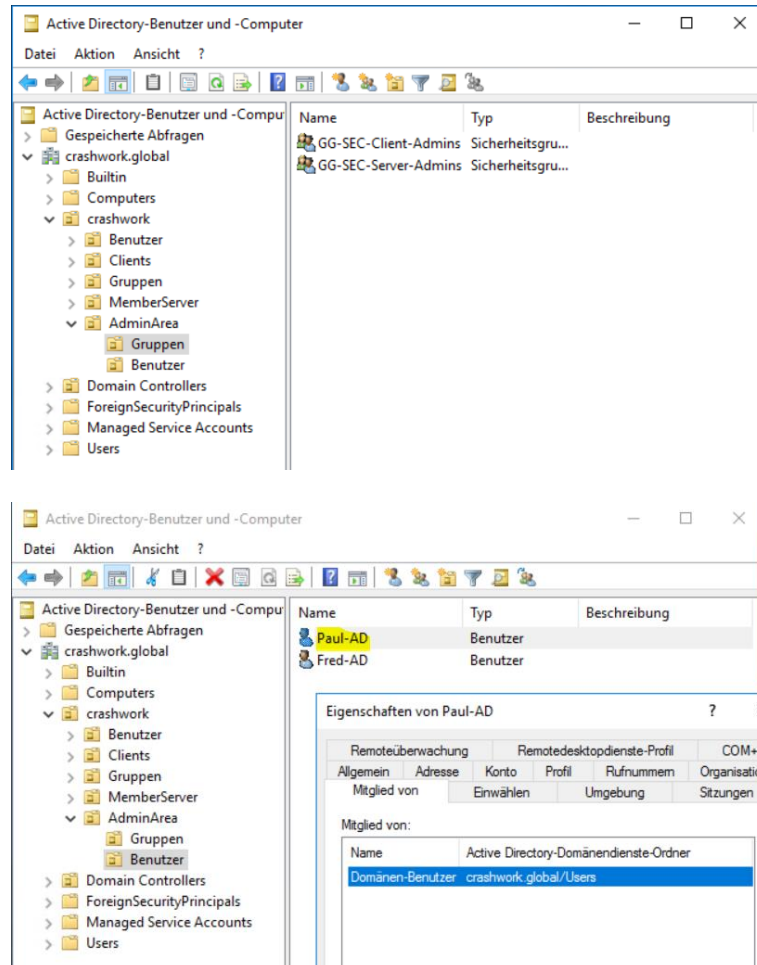
Es sind mehrere Schritte erforderlich, um die Lösung zu implementieren. Dafür hab ich fix ein kleines LAB hochgezogen:



Der Benutzer Paul.Paulsen ist der StandardBenutzer, der mit dem Account Paul-AD administrativ tätig werden soll. Dafür soll er zuerst auf seinem Client C-CL1 die PAM-AdminGUI verwenden, um seinen Benutzeraccount Paul-AD über einen JEA-Endpunkt auf C-DC1 temporär in die Gruppe „Domänen Admins“ aufzunehmen. Danach verwendet er eine RDP-Sitzung zum Server C-SVR1 mit seinem AD-Account – und legt los. Alles klar? 😊

### Vorbereitung vom Active Directory

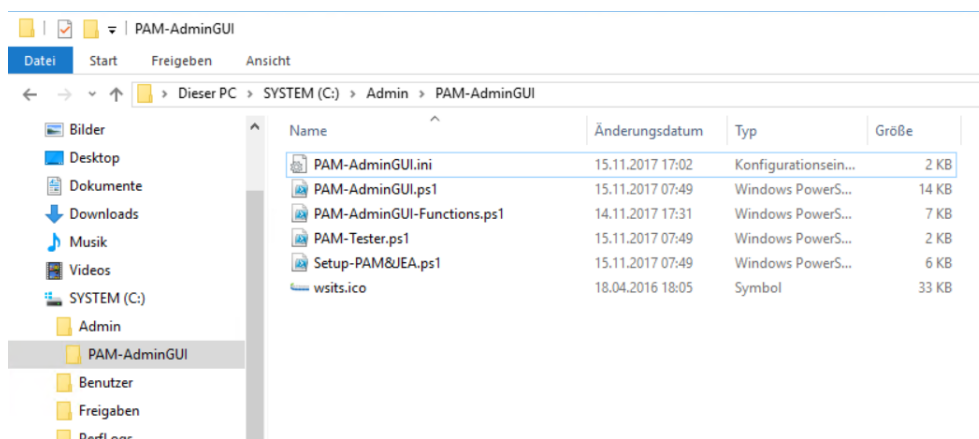
Ich erwähnte bereits: das AD sollte über ein wenig Struktur verfügen. In meinem Fall habe ich eine Organisationseinheit AdminArea erstellt. Diese ist weiter unterteilt in eine OU Gruppen und eine OU Admins. Und darin liegen meine benutzerdefinierten Admingruppen und die AD-Accounts:



Wie man sieht: Paul-AD ist ein gewöhnlicher Benutzer ohne administrative Rechte. Und das soll so lange so bleiben, bis Paul die Rechte wirklich benötigt. 😊

### Vorbereitung der Dateien

Zuerst kopiere ich das Verzeichnis mit allen Dateien auf den Server, der über JEA die PAM-Funktionen bereitstellen soll. In meinem Fall ist das der C-DC1:



Dann wird die Datei Setup-PAM&JEA.ps1 in einer administrativen PowerShell-ISE geöffnet. Für die Einrichtung von PAM werden die Rechte eines Enterprise-Admins erforderlich (Anhebung der Funktionsebene des AD). Und für die Einrichtung von JEA sind lokal administrative Rechte erforderlich. Ich führe alles mit dem Domänen-Administratorkonto aus.

In der Setup-Datei sind verschiedene Blöcke mit PowerShell-Code enthalten. Diese dürfen nur einzeln ausgeführt werden! F5 ist keine gute Idee!!!

Alle weiteren Zeilenangaben beziehen sich auf meine Version V1.04. Sollte ich später etwas am Code ändern, dann sucht bitte die richtigen Zeilen zur Ausführung! Der Code ändert einiges in der Infrastruktur. ein Praktikant darf bei der Ausführung gerne zusehen, aber das doing bleibt bitte bei einem ausgeschlafenen und mit Kaffee versorgten Administrator!

Der erste Block (Zeile 9-10) zeigt die Datei PAM-AdminGUI.ini mit einem Notepad an. Editiert hier unbedingt die Werte passend zu eurer Infrastruktur:

Variable	Bedeutung
PAMAdminRolleGruppeName	Im AD wird eine Gruppe mit diesem Namen aufgebaut. Mitglieder dürfen dann über den JEA-Endpunkt die PAM-Funktionen starten.
PAMAdminRolleGruppeOU	Die Gruppe braucht ein Zuhause. Hier könnt ihr den DN der OU eintragen.
FilterAdminGruppen	Meine PAM-AdminGUI zeigt nachher nicht alle Gruppen des gesamten AD an, sondern nur jene, die diesem AD-Filter entsprechen. Die Gruppen werden unterhalb der OUAdminGruppen gesucht.
OUAdminGruppen	
FilterAdmins	Gleiches gilt für die AD-Accounts: diese werden unterhalb der OU OUAdmins gesucht und müssen dem AD-Filter entsprechen
OUAdmins	
MitDomainAdmins MitEnterpriseAdmins MitSchemaAdmins	Zusätzlich können diese 3 StandardAdminGruppen zur Auswahl hinzugefügt werden. Gültige Variablenwerte sind „an“ oder „aus“
AdminZeitInMinuten	Die PAM-AdminGUI bietet später über ein Dropdown-Feld diese Zahlenwerte an. Daraus wird der Minutenwert für die temporäre Gruppenmitgliedschaft gebildet.

Achtet bitte auf die Anführungszeichen. Nicht alle Variablen dürfen diese verwenden! Wenn alle Zeilen angepasst wurden, dann kann es weitergehen. Schließt Notepad.

Führt nun diese Zeilen aus. Damit werden die Laufzeitvariablen für das Setup geladen:

```

8 # Variablen
9 notepad.exe PAM-AdminGUI.ini
10 pause
11
12 Get-Content -Path "PAM-AdminGUI.ini"
13   ForEach-Object {
14     if ($_.length -gt 0) {
15       if ([$_ -notcontains ($_) 0] {
16         $Name, $wert = ($_.split '#')[0].Trim().split '='
17
18         $Name = "WS_" + $Name.trim()
19
20         Get-Variable -Scope global | Where-Object {$_.name -eq $Name} | Remove-Variable -Scope gl
21         New-Variable -Name $Name.trim() -Value $wert.trim() -Scope global
22       }
23     }
24   }
25

```

### Setup von PAM

Diese Zeilen starten die Konfiguration des Features „Privileged Access Management“:

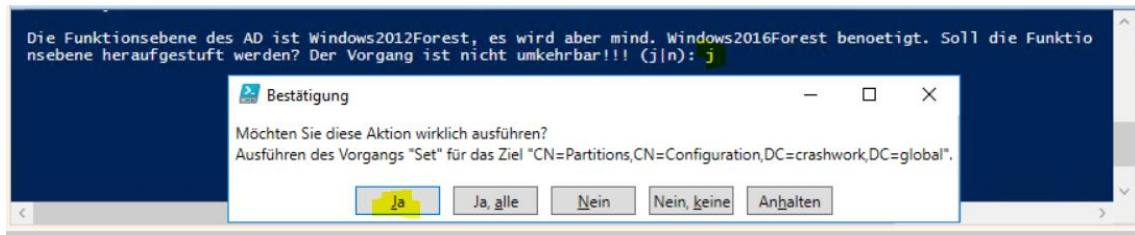
```

27 # Vorbereitung PAM
28 Import-Module -Name ActiveDirectory
29 $ADFunktionsEbene = (Get-ADForest).ForestMode
30 if ($ADFunktionsEbene -replace '\D' -lt 2016) {
31   $Antwort = Read-Host -Prompt "Die Funktionsebene des AD ist $ADFunktionsEbene, es wird aber mind.
32   if ($Antwort -eq 'j') {
33     Set-ADForestMode -ForestMode "Windows2016Forest" -Identity (Get-ADForest).name
34   } else {
35     exit
36   }
37 }
38 if ((Get-ADOptionalFeature -Identity 'Privileged Access Management Feature').EnabledScopes.count -eq

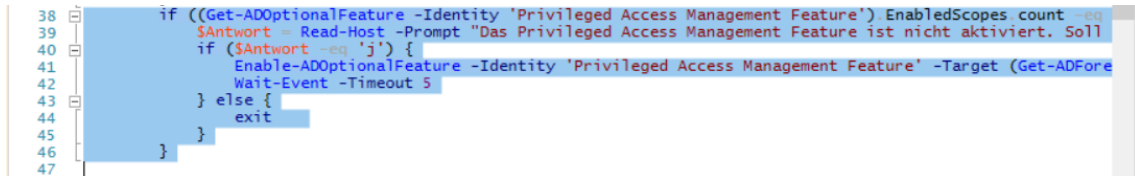
```



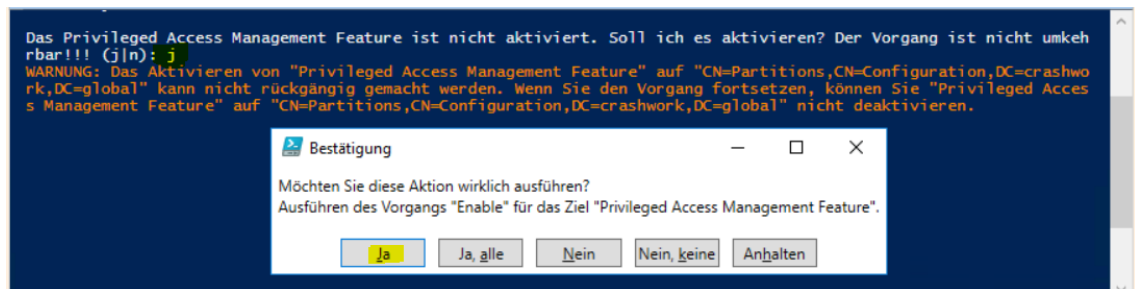
Damit wird geprüft, ob die Funktionsebene der Gesamtstruktur mindestens WindowsServer2016 ist. Falls nicht fragt euch das Script 2x, ob ihr das anpassen wollt. Ich bin mir der Konsequenzen bewusst und bestätige die Anhebung der Ebene:



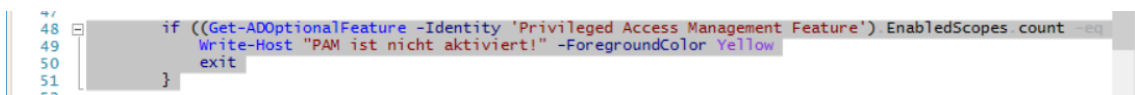
Der nächste Block prüft, ob das ADOptionalFeature PAM bereits aktiviert ist:



Falls nicht, dann wird euch der Dialog wieder 2x fragen, ob das geändert werden darf:



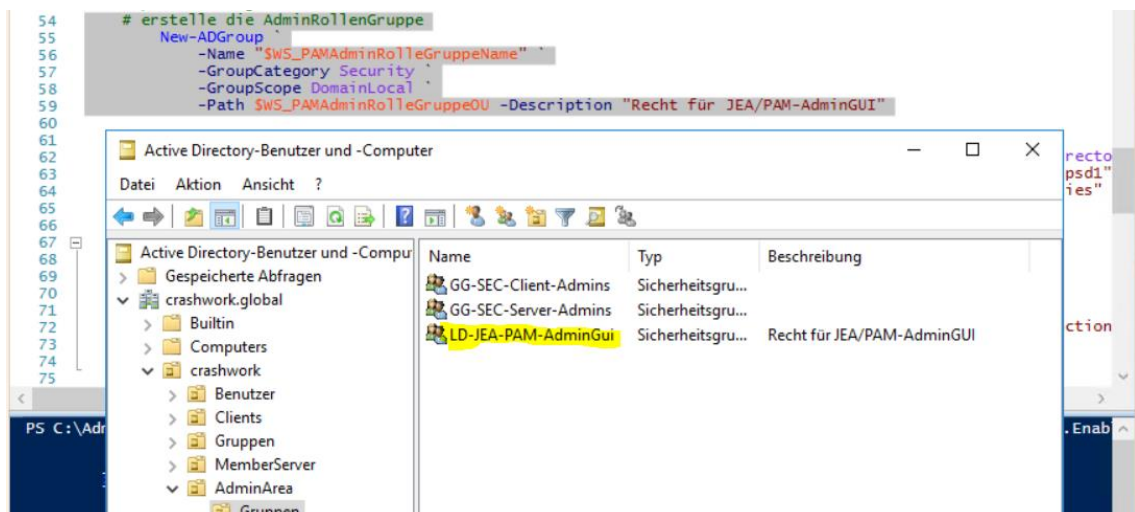
Wartet bitte in Umgebungen mit mehreren DCs, bis diese repliziert haben. Danach kann mit diesen Zeilen das Ergebnis geprüft werden:



PAM ist nun einsatzbereit.

### Setup von JEA

Im gleichen Script geht es auf dem C-DC1 weiter. Diese Zeile erstellen die neue Rollengruppe. Mitglieder dürfen später diesen JEA-Endpunkt verwenden:



Die nächsten Zeilen erstellen das erforderliche PowerShell-Modulverzeichnis und die Description-File:

```

61 # erstelle PS-Module für die neue Rolle
62 New-Item -Path "$env:ProgramFiles\WindowsPowerShell\Modules\PAM-AdminGUI" -ItemType Directory
63 New-ModuleManifest -Path "$env:ProgramFiles\WindowsPowerShell\Modules\PAM-AdminGUI\PAM-AdminGUI.psd1"
64 New-Item -Path "$env:ProgramFiles\WindowsPowerShell\Modules\PAM-AdminGUI\RoleCapabilities"
65

```

In dem neuen Verzeichnis RoleCapabilities wird mir diesen Zeilen die ResourceConfiguration-File erzeugt. Diese regelt, welche Befehle ein Benutzer über den JEA-Endpoint ausführen darf:

```

66 # Definition der Administrations-Rolle
67 $Rolle = @{
68     Author = $env:username
69     CompanyName = (Get-ADDomain) NetBIOSName
70     ModulesToImport = 'ActiveDirectory'
71     VisibleCmdlets = 'get-ADGroupMember', 'add-member'
72     ScriptsToProcess = "C:\Program Files\WindowsPowerShell\Modules\PAM-AdminGUI\PAM-AdminGUI-Function
73     FunctionDefinitions = @{ Name = 'Get-UserInfo'; ScriptBlock = { $PSSenderInfo } }
74 }
75 # erzeuge die Rollenbeschreibungdatei
76 New-PSRoleCapabilityFile -Path "$env:ProgramFiles\WindowsPowerShell\Modules\PAM-AdminGUI\RoleCapabili
77

```

Folgende Befehle sind nun erlaubt:

Befehl	Zweck
<b>Get-ADGroupMember</b>	Der JEA-User soll die bestehenden Gruppenmitglieder auslesen können.
<b>Add-Member</b>	Dieser Befehl dient nur der Erweiterung von PowerShell-Objekten. Er hat keine Funktion im ActiveDirectory. Sempel betrachtet wird er von mir für Ausgabeformatierungen verwendet.
<b>Alle Befehle, die in der Scriptdatei „PAM-AdminGUI-Functions.ps1“ deklariert sind</b>	Die Scriptdatei erzeugt zur Laufzeit einige Funktionen. Und diese stehen den JEA-User zur Verfügung. Das Feintuning bzw. die Konfiguration ermittelt diese Scriptdatei aus der PAM-AdminGUI.ini

Natürlich muss dann auch die PAM-AdminGui-Functions.ps1 mit ins Modul-Verzeichnis. Das übernehmen diese Zeilen:

```

78 # kopiere FunctionsFile
79 Copy-Item PAM-AdminGUI-Functions.ps1 -Destination "$env:ProgramFiles\WindowsPowerShell\Modules\PAM-Ad
80 Copy-Item PAM-AdminGUI.ini -Destination "$env:ProgramFiles\WindowsPowerShell\Modules\PAM-Ad
81

```



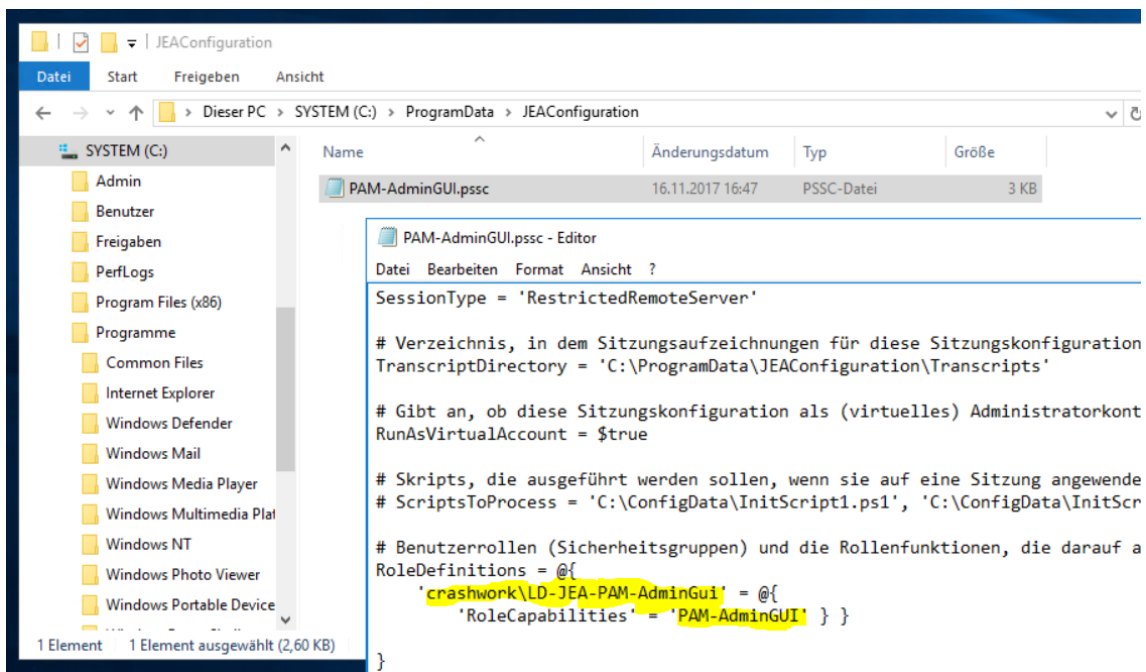
Bis jetzt ist beschrieben, WAS über den JEA-Endpoint ausgeführt werden darf. Nun fehlt noch das WER. Das übernehmen diese Zeilen:

```

82 # erstelle Endpunkt-Sessionconfiguration
83 $JEAConfigParameter = @{
84     SessionType = 'RestrictedRemoteServer'
85     RunAsVirtualAccount = $true
86     RoleDefinitions = @{"$((Get-ADDomain) NetBIOSName)\$WS_PAMAdminRolleGruppeName" = @{ RoleCapabil
87     TranscriptDirectory = "$env:ProgramData\JEAConfiguration\Transcripts"
88 }
89
90 # erstelle Verzeichnis für die Session-Konfigurationen
91 New-Item -Path "$env:ProgramData\JEAConfiguration" -ItemType Directory -ErrorAction SilentlyContinue
92
93 # erstelle JEA-Endpoint
94 if (Get-PSSessionConfiguration -Name 'PAM-AdminGUI' -ErrorAction SilentlyContinue) {
95     Unregister-PSSessionConfiguration -Name 'PAM-AdminGUI' -ErrorAction Stop
96 }
97 New-PSSessionConfigurationFile -Path "$env:ProgramData\JEAConfiguration\PAM-AdminGUI.pssc" @JEAConfig

```

Das Ergebnis findet man in diesem SessionConfiguration-File:



Zuletzt muss der neue JEA-Endpoint noch aktiviert werden. Optional schalte ich dazu die Protokollierung ein:

```

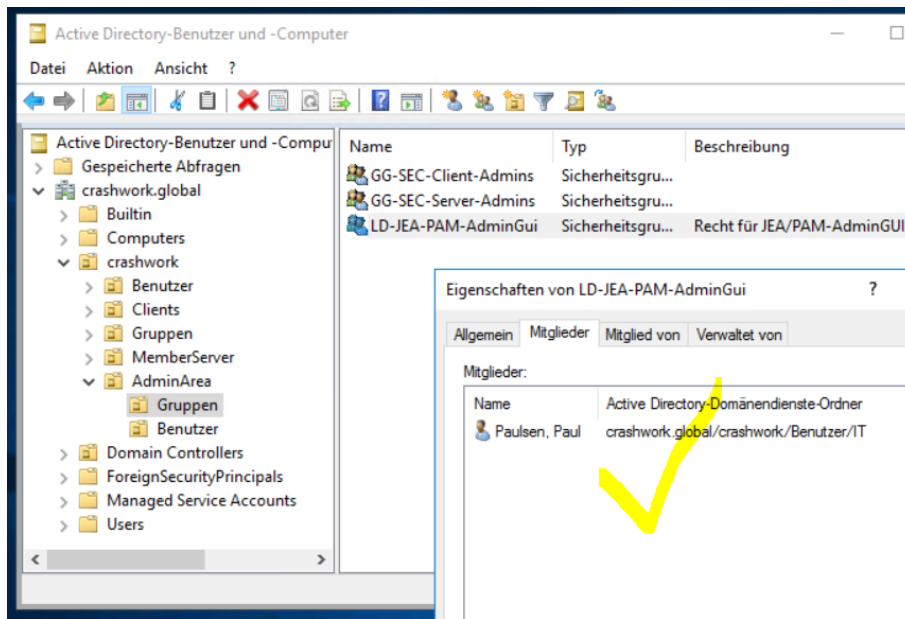
99 # lade die JEA-Session-Konfiguration
100 Register-PSSessionConfiguration -Name 'PAM-AdminGUI' -Path "$env:ProgramData\JEAConfiguration\PAM-Adm
101
102 # aktiviere ModuleLogging (optional)
103 New-Item -Path HKLM:\SOFTWARE\Policies\Microsoft\Windows\PowerShell -ErrorAction SilentlyContinue | o
104 New-Item -Path HKLM:\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ModuleLogging -ErrorAction Silent
105 New-Item -Path HKLM:\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ModuleLogging\ModuleNames -ErrorA
106 Set-ItemProperty -Path HKLM:\SOFTWARE\Policies\Microsoft\Windows\PowerShell\ModuleLogging\ModuleNames
107

```

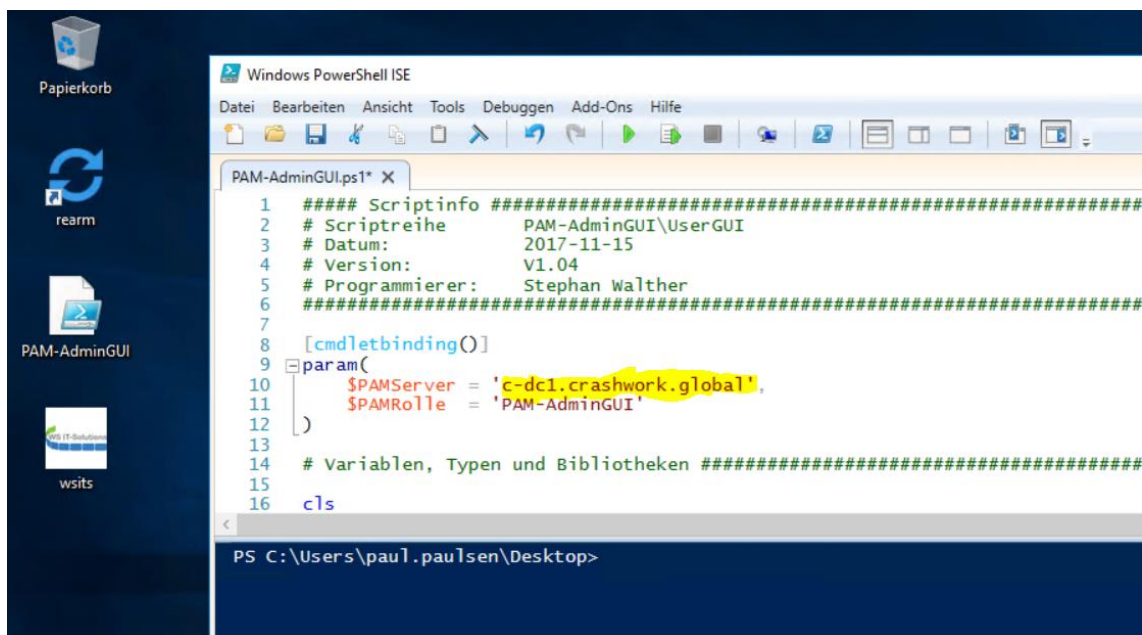
Das war die Vorbereitung des PowerShell-JEA-Endpunktes.

### Vorbereitung des Clients

Natürlich muss der Standard-Benutzer Paul.Paulsen noch in die JEA-Gruppe aufgenommen werden. Sonst darf er JEA und damit auch PAM nicht verwenden:



Dann bekommt mein Paul noch die Scriptdatei auf seinen Desktop kopiert (gerne auch mit der Icondatei). In der ps1-Datei kann der Servername angepasst werden. Dann ist der Start des Scriptes einfacher:

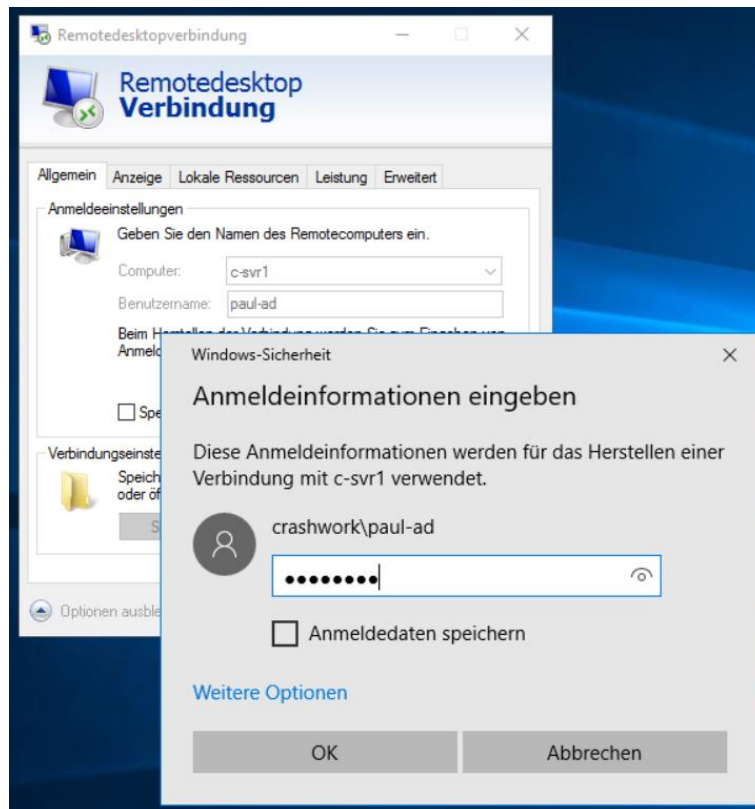


Und das war's auch schon.

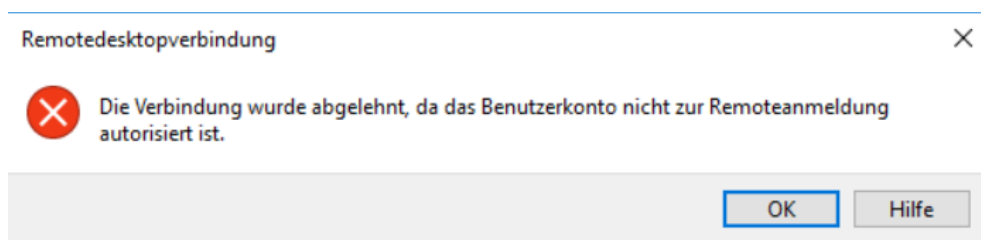
### Betrieb

#### Test – Nutzung des AD-Accounts ohne Berechtigung

Wenn Paul von seinem Client eine RDP-Sitzung auf einen Server startet und dabei seinen AD-Account verwendet...



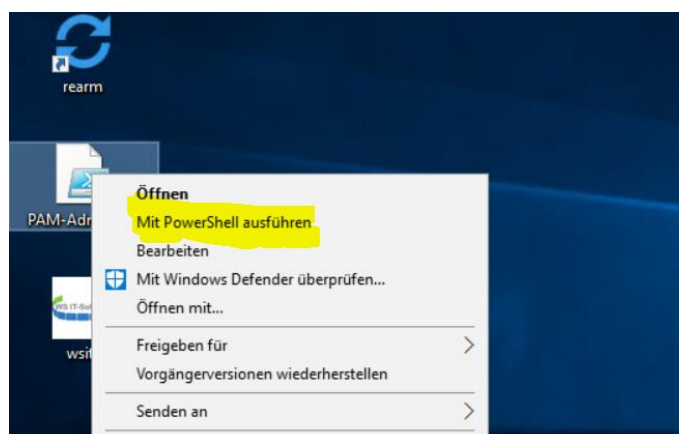
... dann wird das aktuell nichts werden:



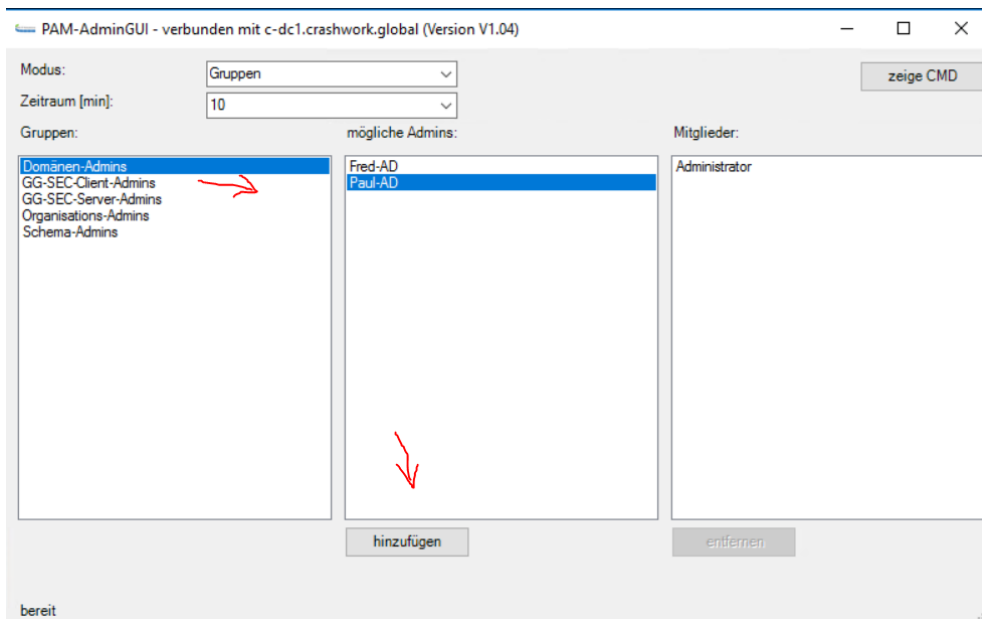
Klar: der AD-Account ist ja auch nicht mehr höher berechtigt.

Test – Nutzung des AD-Accounts nach dem Verwenden der PAM-AdminGUI

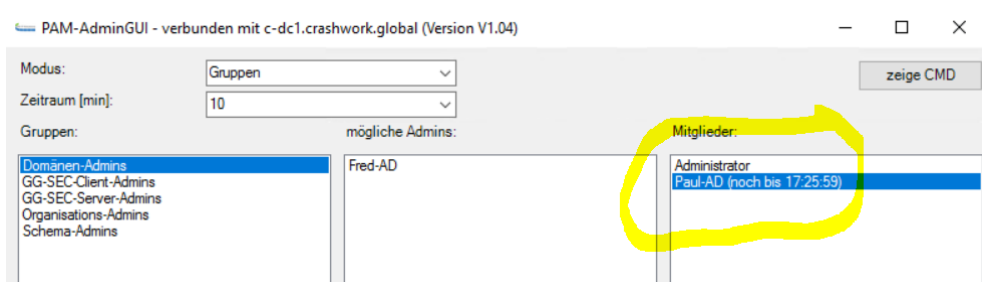
Dann ändern wir das einmal. Paul ruft als Paul.Paulsen das Script PAM-AdminGUI.ps1 auf:



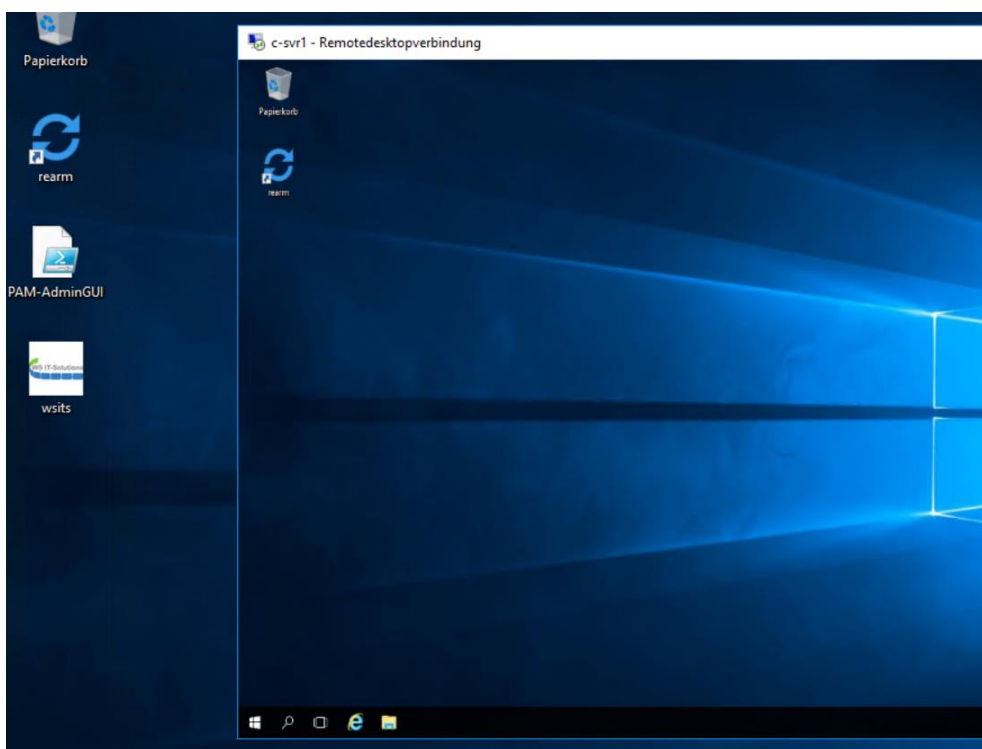
Das Fenster selber ist (hoffentlich) übersichtlich gestaltet. Paul wählt die richtige Gruppe aus, dann den richtigen AD-Account und klickt auf hinzufügen...



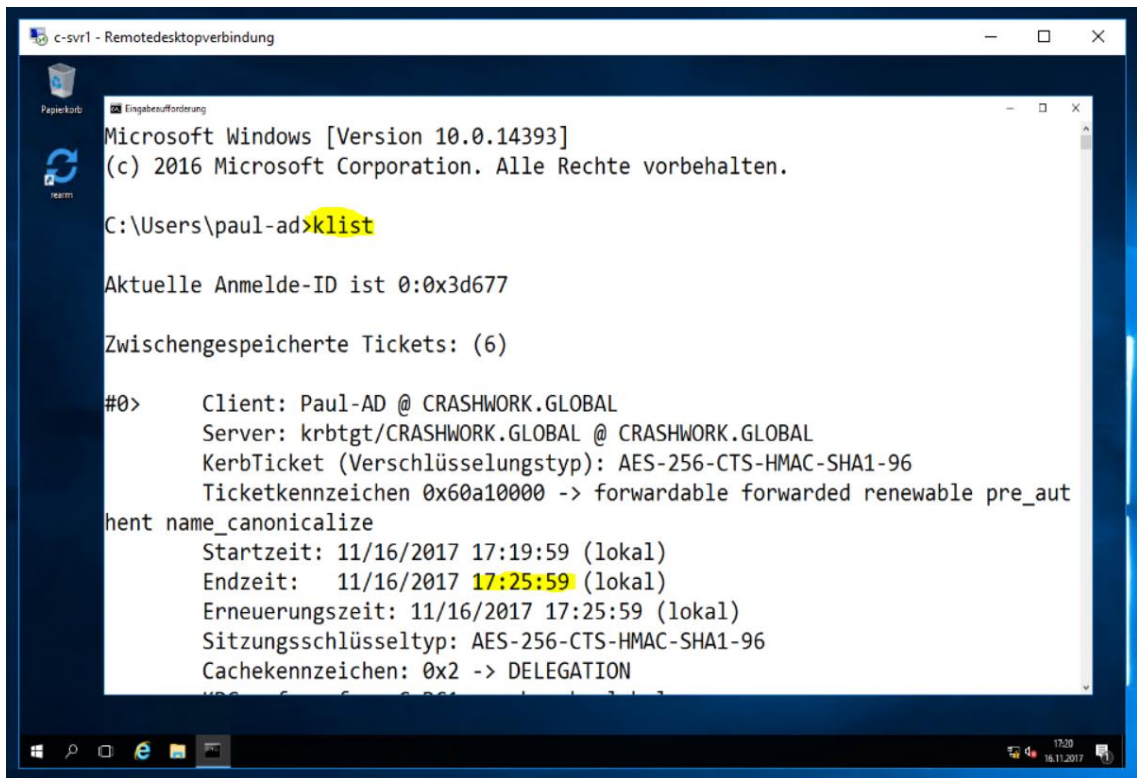
Und dann ist (nach einer kurzen AD-Replikation) sein AD-Account höher berechtigt:



Wenn er nun wieder die RDP-Verbindung versucht, dann klappt es auch:



Klar, sein AD-Account ist nun berechtigt. Aber nur auf Zeit. Und auch das Kerberos-Ticket läuft pünktlich ab:

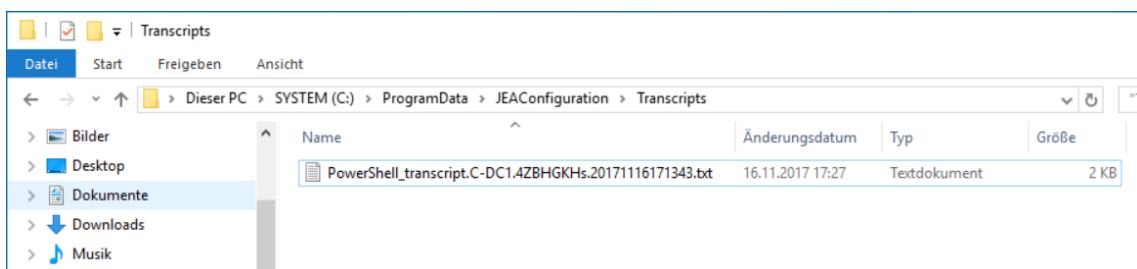


Cool? Easy? Na bitte! ☺

## Details

### Das JEA-Logging

JEA-Verwendungen werden aufgezeichnet. Dabei wird jeder (erfolgreiche) Scriptstart eine Datei auf dem JEA-Server erzeugen:



In dieser Datei ist alles vom User protokolliert. Das hier ist der Mitschnitt von meinem Test weiter oben:

```

*****
Start der Windows PowerShell-Aufzeichnung
Startzeit: 20171116171343
Benutzername: crashwork\Paul.Paulsen
RunAs-Benutzer: WinRM Virtual Users\WinRM VA_1_crashwork_Paul.Paulsen
Computer: C-DC1 (Microsoft Windows NT 10.0.14393.0)
Hostanwendung: C:\Windows\system32\wsmprovhost.exe -Embedding
Prozess-ID: 5936
PSVersion: 5.1.14393.693
PSEdition: Desktop
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.14393.693
BuildVersion: 10.0.14393.693
CLRVersion: 4.0.30319.42000
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
  
```

```

SerializationVersion: 1.1.0.1
*****
PS>CommandInvocation(liste-Zeitspannen): "liste-Zeitspannen"
10
15
30
45
60
120
180
PS>CommandInvocation(liste-Admins): "liste-Admins"
Fred-AD
Paul-AD
PS>CommandInvocation(liste-Gruppen): "liste-Gruppen"
Domänen-Admins
GG-SEC-Client-Admins
GG-SEC-Server-Admins
Organisations-Admins
Schema-Admins
PS>CommandInvocation(zeige-GruppenMitglieder): "zeige-GruppenMitglieder"
>> ParameterBinding(zeige-GruppenMitglieder): Name="Gruppe"; Wert="Domänen-Admins"

User          TTL bis
----          -
Administrator

PS>CommandInvocation(erstelle-PAMGruppenMitglied): "erstelle-PAMGruppenMitglied"
>> ParameterBinding(erstelle-PAMGruppenMitglied): Name="Admin"; Wert="Paul-AD"
>> ParameterBinding(erstelle-PAMGruppenMitglied): Name="Gruppe"; Wert="Domänen-Admins"
>> ParameterBinding(erstelle-PAMGruppenMitglied): Name="Zeitspanne"; Wert="10"
OK
PS>CommandInvocation(zeige-GruppenMitglieder): "zeige-GruppenMitglieder"
>> ParameterBinding(zeige-GruppenMitglieder): Name="Gruppe"; Wert="Domänen-Admins"

User          TTL bis
----          -
Paul-AD       600 17:25:59
Administrator

*****
Ende der Windows PowerShell-Aufzeichnung
Endzeit: 20171116172700
*****

```

Darin sieht man einige interessante Funktionsnamen, z.B. erstelle-PAMGruppenMitglied. Diese Funktionen werden beim JEA-Start durch meine Datei PAM-AdminGUI-Functions.ps1 dynamisch erzeugt. Wozu das Ganze? Das erklärt der nächste Abschnitt...

### Die JEA-ProxyFunktionen

Ich hätte dem Benutzer auch die Berechtigung für die Standardfunktion **Add-ADGroupMember** geben können, denn genau diese steckt hinter der Funktion **erstelle-PAMGruppenMitglied**. Dann könnte meine GUI mit der Benutzerauswahl folgenden Befehl an den JEA-Server senden:

```

Invoke-Command -ComputerName C-DC1 -ScriptBlock {
    Add-ADGroupMember `
        -Identity 'Domänen-Admins' `
        -Members 'Paul-AD' `
        -MemberTimeToLive (New-TimeSpan -Minutes 10)
} -ConfigurationName PAM-AdminGUI

```



Und alles wäre gut. Wirklich? Was wäre, wenn der Benutzer Paul.Paulsen stattdessen diesen Befehl über die PowerShell an den JEA-Server sendet:

```
Invoke-Command -ComputerName C-DC1 -ScriptBlock {
    Add-ADGroupMember `
        -Identity 'Domänen-Admins' `
        -Members 'Paul-AD'
} -ConfigurationName PAM-AdminGUI
```

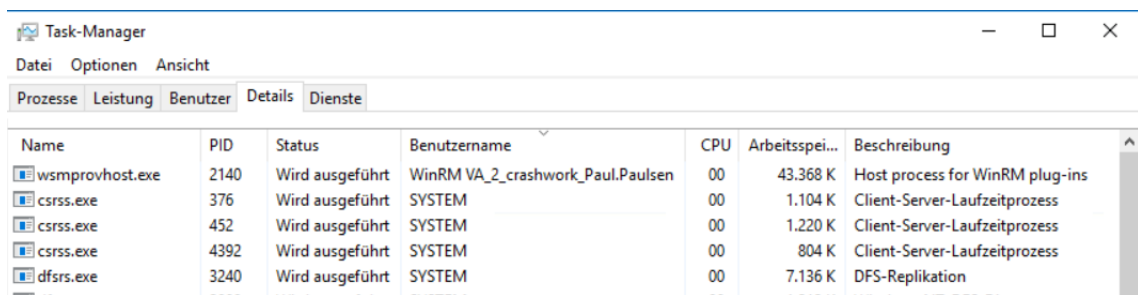
Wäre dann der Benutzer Paul-AD immer noch nur 10 Minuten DomänenAdmin? Eben nicht! Denn das Cmdlet Add-ADGroupMember bewertet den Parameter MemberTimeToLive nicht als Pflichtparameter. **Er kann also weggelassen werden.** Und dazu könnte Paul JEDE andere Gruppe und JEDEN andere Benutzer auswählen. Das wäre fatal.

Die Proxyfunktionen habe ich so geschrieben, dass eben nur bestimmte Gruppen und bestimmte Benutzer erlaubt sind (diese werden über die ini-Datei identifiziert). Und der Parameter für die Mitgliedschaftszeit ist eben nicht optional! Dazu kann auch nicht jede beliebige Zeit ausgewählt werden.

Proxyfunktionen filtern also die möglichen Aufrufe und erzwingen eine entsprechende Logik für den Workflow.

### JEA-VirtualAccounts

Ist Paul.Paulsen über JEA tatsächlich auf dem DC angemeldet? Das ist natürlich nicht der Fall. In der JEA-Konfiguration habe ich die virtuellen Accounts aktiviert. Im Taskmanager auf dem DC sieht man also nur diesen Paul – einen virtuellen Account:



Name	PID	Status	Benutzername	CPU	Arbeitsspei...	Beschreibung
wsmpvhost.exe	2140	Wird ausgeführt	WinRM VA_2_crashwork_Paul.Paulsen	00	43.368 K	Host process for WinRM plug-ins
csrss.exe	376	Wird ausgeführt	SYSTEM	00	1.104 K	Client-Server-Laufzeitprozess
csrss.exe	452	Wird ausgeführt	SYSTEM	00	1.220 K	Client-Server-Laufzeitprozess
csrss.exe	4392	Wird ausgeführt	SYSTEM	00	804 K	Client-Server-Laufzeitprozess
dfsrs.exe	3240	Wird ausgeführt	SYSTEM	00	7.136 K	DFS-Replikation

### Was passiert nach dem Ablauf der MemberTimeToLive?

Das Kerberos-Ticket wird so ausgestellt, dass dessen Gültigkeit mit der kleinsten MemberTimeToLive abläuft. Nach dieser Zeit ist das Ticket dann ungültig und weitere Autorisierungen schlagen fehl:

