# Inhalt



In vielen ActiveDirectory-Domains gibt es etliche Benutzerkonten, die hochprivilegiert sind. Nicht wenige davon können durch verschiedene Faktoren ein Risiko darstellen. Der Klassiker sind dabei Benutzer, die mit nicht ablaufenden Kennwörtern als ServiceAccount eingesetzt werden. Und wie oft habe ich diese Konten schon als Mitglied der Domain-Admin-Gruppe gesehen... :-(

Wenn die Gefährdung von den Administratoren verstanden wurde kommen meist diese Fragen auf:

- Welche Konten sind denn eigentlich betroffen?
- Wo werden diese Konten überall eingesetzt?

Diese Fragen möchte ich mit meinem neuen PowerShell-Script „PrivilegedADUser-Analyse.ps1“ beantworten.

# Aufbau des Scripte

## Arbeitsweise

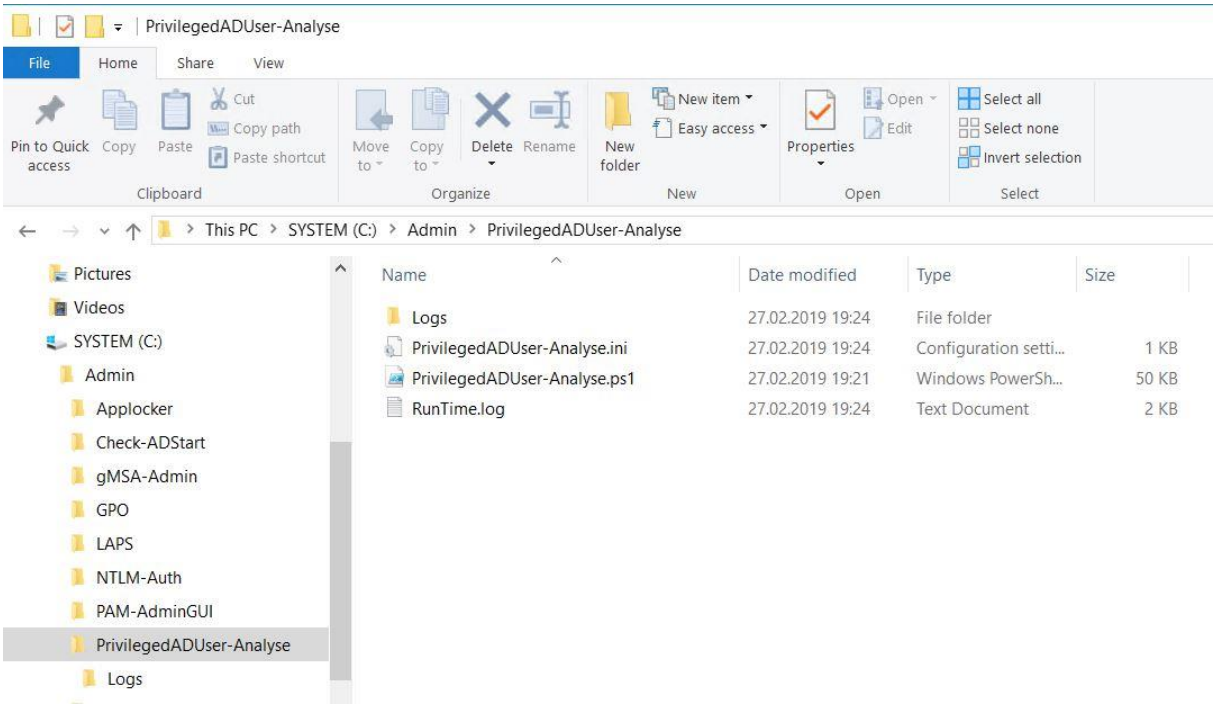
1. Das Script ermittelt aus dem ActiveDirectory alle sensiblen, hochprivilegierten Gruppen automatisch. Zusätzlich kann die Liste durch Suchfilter und Ziel-OUs auf eigene Gruppen erweitert werden.
2. Im nächsten Schritt werden die AD-Benutzer ermittelt, welche in diesen Gruppen Mitglied sind. Dabei werden auch Gruppenverschachtelungen berücksichtigt. Diese Benutzer sind unsere Administratoren.
3. Nun werden alle Windows-Server der Domäne ermittelt.
4. Diese Server werden via PowerShell-Remoting kontaktiert. Für jeden Server wird die Liste der geplanten Aufgaben und die Liste der Services ermittelt und zum Scriptserver übertragen.
5. In den Listen werden die Sicherheitsprinzipale der geplanten Aufgaben und der Services bestimmt und mit der Liste der Administratoren aus Schritt 2 abgeglichen.
6. Alle bisher gesammelten Daten werden bewertet und für die Ausgabe wird eine HTML-Datei generiert.
7. Die HTML-Datei wird je nach Konfiguration als Email versandt oder im Internet Explorer angezeigt.

## Voraussetzungen

- Ich habe das Script mit der PowerShell V4 und V5 programmiert. Daher sollte es auf Windows Server 2012R2 und höher gut laufen.
- PowerShell-Remoting über Windows-RemoteManagement (WinRM) ist zu den anderen Servern im Netzwerk erforderlich.
- Ebenso benötigt der Benutzer, der das Script ausführt, das Remoting-Recht auf die anderen Server. Dort angekommen muss er die geplanten Aufgaben und die Services auslesen können.
- Für die Mailfunktionalität sind entsprechende Anpassungen am Mailserver denkbar.

## Setup

Das Script selber benötigt ein Arbeitsverzeichnis für sich, seine Konfigurationsdatei und seine Analyseergebnisse. Das könnte auf einem Server so aussehen:




Innerhalb der PowerShell-Scriptdatei sind keine Anpassungen erforderlich. Somit wird auch die digitale Signatur des Scriptcodes nicht beschädigt. Sämtliche Anpassungen an die eigene Infrastruktur werden über die ini-Datei vorgenommen:

```
1  [Mailing]
2  MailAn     = logmails@crashwork.global
3  MailVon    = admin@crashwork.global
4  MailServer = email.crashwork.global
5  MailAktiv  = nein
6
7  [Filter: Organisationseinheiten mit Adminkonten]
8  AdminGruppenOUs += OU=crashwork,DC=crashwork,DC=global
9
10 [Filter: sensible ADGruppen]
11 AdminGruppenFLT += Name -like "*admin*"
12 AdminGruppenFLT += Name -like "*service"
13 AdminGruppenFLT += Name -like "*task"
14 AdminGruppenFLT += Name -like "*WinRM"
15 AdminGruppenFLT += Name -like "*-JEA-*"
16
17 [Filter: ausgenommene AdminAccounts]
18 'AdminAusnahmen += Service-SQL    # geht nicht anders
19 AdminAusnahmen += Paul.Paulsen   # HoneyToken
20
21 [Grenzwerte]
22 KRBTGT_PWDLaufzeit = 90
23 PWDAblauf_Warnung  = 10  # Tage    .. weniger wird gelb angezeigt
24 PWDAblauf_Achtung  = 5   # Tage    .. weniger wird rot angezeigt
25 PWsichereLaenge    = 15  # Zeichen .. auch bei statischen Passwörtern nicht rot angezeigt
26 PWmindestLaenge    = 8   # Zeichen
27 PWhistory_Warnung  = 3   # Tage    .. weniger wird rot angezeigt
28 POS_SCRequire      = j   # n|j .. ist SmartCardRequire positiv?
29 POS_ProtectedUser  = n   # n|j .. ist ProtectedUser positiv?
```

Jede Zeile kann mit einem Semikolon auskommentiert werden (Zeile 18). Viele Angaben sind aber verpflichtend zu definieren! Ebenso müssen die Werte dahinter korrekt sein, da ich nicht alle möglichen Eingaben und daraus resultierende Fehler abgefangen habe. Die Konfiguration ist in 5 Sektionen unterteilt:

- **[Mailing]**:
  Hier könnt ihr die Mailfunktion steuern. Dabei stehen die üblichen Parameter zur Verfügung. Der Parameter MailAktiv reagiert auf die Werte „ja“ und „nein“. Bei „nein“ wird das Ergebnis im Internet-Explorer angezeigt (der ja immer noch der Standard-Browser auf Serversystemen ist).

- **[Filter: Organisationseinheiten mit Adminkonten]**:
  Das ist ein Array, also eine Sammlung von verschiedenen Organisationseinheiten im AD, in denen eure AdminGruppen gesucht werden sollen. Ihr könnt die Zeile 8 einfach duplizieren und den DN anpassen. In jeder OU (inklusive Unter-OUs) werden eure AdminGruppen gesucht.

- **[Filter: sensible ADGruppen]**:
  Vielleicht habt ihr eure AdminGruppen mit anderen Gruppen in den OUs gemischt? Dann könnt ihr mit diesem Array den Suchfilter definieren. Alle Zeilen (11 bis 15) werden mit ODER verknüpft. Die Eingaben aus meiner im Bild gezeigten Konfiguration entsprechen dieser AD-Abfrage:

```
Get-ADGroup `
    -Filter     'Name -like "*admin*" -or Name -like "*service" -or Name -like "*task" -or Name -like "*WinRM" -or Name -like "*-JEA-*"' `
    -SearchBase 'OU=crashwork,DC=crashwork,DC=global'
```

- **[Filter: ausgenommene AdminAccounts]**:
  Mit diesem Array könnt ihr Accounts definieren, die problematisch sind, an denen aber nichts geändert werden kann. Das Script wird die Benutzer weiter überprüfen, aber in der Ausgabe werden sie in blauen statt roten Zeilen dargestellt4.

- **[Grenzwerte]**: Hier kommen die Grenzwerte für einige Bewertungen rein:
  - **KRBTGT_PWDLaufzeit**: Normal wird das Passwort des Users KRBTGT niemals ablaufen. Nur genau das ermöglicht die Erstellung von Kerberos-GoldenTickets… Also sollte dieses Passwort genauso altern, wie alle anderen. Hier könnt ihr das Alter in Tagen eingeben.
  - Wenn Passworte ablaufen, dann wird euch das Script (regelmäßige Ausführung vorausgesetzt) vorher informieren. Mit den beiden Schwellwerten **PWDAblauf_Warnung** und **PWDAblaufAchtung** wird definiert, ab wann die Ausgabe gelb oder rot wird.
  - **PWsichereLaenge**: ab dieser Passwortlänge könnte ein Passwort als sicher(er) gelten. Das Script wird bei der Bewertung darauf reagieren und keine Warnung ausgeben, selbst wenn das Passwort nicht abläuft. Falls ihr das nicht benötigt: setzt einfach den Wert auf 999.
  - **PWmindestLaenge**: Sind Passworte zu kurz, dann könnten Sie erraten werden. Könnte ein Benutzer ein Passwort mit einer kürzeren Länge haben, dann wird dies als kritische Warnung in der Bewertung ausgegeben.
  - **PWhistory_Warnung**: Passwortrichtlinien können eine PasswortHistory erzwingen, mit der die Wiederverwendung der X letzten Kennworte nicht erlaubt wird. Mit diesem Parameter könnt ihr angeben, welche Mindestanzahl an nicht wiederverwendbaren Kennworten konfiguriert sein muss. Definiert ihr beispielsweise den Wert 3, dann muss die KennwortHistory mindestens 3 betragen, sonst wird die Konfiguration als Fehler gewertet.
  - **POS_SCRequire**: Man kann sich darüber streiten: ist ein Passwortverfahren sicherer als ein SmartCardlogon. Mit diesem Schalter könnt ihr eure Präferenz eintragen: Ist für euch RequireSmartCard sicherer, dann tragt ein J ein. Dann wird ein Benutzer mit diesem Attribut immer als sicher betrachtet. Seid ihr anderer Meinung, dann schreibt ein N in die Konfiguration.
  - **POS_ProtectedUser**: Gleiches gilt für die Mitgliedschaft in der Gruppe Protected Users: ein J wird Mitglieder immer als sicher behandeln. Protected User können durchaus einen Sicherheitsgewinn darstellen. Nutzt ihr aber noch sehr viele Windows 7 / 2008R2, dann wird die Gruppenmitgliedschaft keinen Effekt haben! Dann wäre ein N als Wert sinnvoll.

Probiert es einfach mal durch. Für die Wertefindung wäre am Anfang ein MailAktiv = nein empfehlenswert.

## Konfiguration als geplante Aufgabe

Meiner Erfahrung nach ist die Ausführung des Scriptes und die anschließende, mögliche Korrektur der Admin-Konfigurationen kein einmaliger Prozess. „Morgen“ werden ggf. neue Admins erstellt oder deren Nutzung wird negativ angepasst.

Daher empfehle ich die wiederholte Ausführung des Scriptes. Dies kann durch eine geplante Aufgabe auf einem Server erledigt werden. Die Berichte werden dann z.B. wöchentlich generiert und per Mail an euch versendet.

Der TaskAccount benötigt natürlich selber sehr hohe Rechte. Daher empfehle ich die Ausführung der Aufgabe mit einem gMSA – einem Group Managed Service Account.

# Analyse der Ergebnisse

Nun kommen wir zur eigentlichen Analyse. Das Script kann nach Anpassung der Konfigurationsdatei manuell gestartet werden. In der Ausgabe der PowerShell werden dabei schon einige Details sichtbar:

```
finde MemberServer
   4 Hosts gefunden
finde AdminGruppen
   19 Gruppen gefunden
finde Administratoren
   9 Admins gefunden
finde ScheduledTasks
   sammle Task-Informationen über WinRM
      S-SVR3
      S-SVR2
      S-DC1
      S-SVR1
   4/4 Server ausgelesen
   filtere Task-Informationen
finde Admins mit ScheduledTasks
   2 Benutzer mit ScheduledTask gefunden
finde Services
   sammle Service-Informationen über WinRM
      S-SVR3
      S-SVR2
      S-SVR1
      S-DC1
   4/4 Server ausgelesen
   filtere Service-Informationen
finde Admins, die Services ausführen
   2 Benutzer mit ScheduledTask gefunden
bewerte Ergebnis
erstelle HTML-Bericht
zeige Report
```

In der HTML-Datei finden wir die Ergebnisse in verschiedenen, farbigen Tabellen.

## Admingruppen Level 1 und Level 2

Das könnte so aussehen:

**AdminGruppen Level 1 und 2**

| Gruppe | Level | Mitglieder |
|---|---|---|
| Account Operators | 1 | |
| Administrators | 1 | Administrator, Service-SQL, Service-STM, Task-Monitoring |
| Backup Operators | 1 | |
| Domain Admins | 1 | Administrator, Service-SQL, Service-STM, Task-Monitoring |
| Domain Controllers | 1 | |
| Enterprise Admins | 1 | Administrator |
| Print Operators | 1 | |
| Read-only Domain Controllers | 1 | |
| Replicator | 1 | |
| Schema Admins | 1 | Administrator |
| Server Operators | 1 | |
| GG-Admin-AD | 2 | |
| GG-Admin-Clients | 2 | Frank.Furt |
| GG-Admin-Drucker | 2 | Frank.Furt |
| GG-Admin-Freigaben | 2 | Olga.Ordnung |
| LD-Admin-AD | 2 | Fred.Fredsen, Paul.Paulsen |
| LD-Admin-Clients | 2 | Frank.Furt, Fred.Fredsen, Paul.Paulsen |
| LD-Admin-Drucker | 2 | Frank.Furt, Fred.Fredsen, Paul.Paulsen |
| LD-Admin-Freigaben | 2 | Fred.Fredsen, Olga.Ordnung, Paul.Paulsen |

Alle gefundenen Gruppen werden gelistet. Level 1 sind alle Gruppen mit dem AD-Attribut AdminCount – also sensible, hochberechtigte Systemgruppen:

```
PS C:\Admin\PrivilegedADUser-Analyse> Get-ADGroup -Filter 'Admincount -eq 1' | Format-Table -Property name,sid

name                        sid
----                        ---
Administrators              S-1-5-32-544
Print Operators             S-1-5-32-550
Backup Operators            S-1-5-32-551
Replicator                  S-1-5-32-552
Domain Controllers          S-1-5-21-1741355641-1558180958-3330877669-516
Schema Admins               S-1-5-21-1741355641-1558180958-3330877669-518
Enterprise Admins           S-1-5-21-1741355641-1558180958-3330877669-519
Domain Admins               S-1-5-21-1741355641-1558180958-3330877669-512
Server Operators            S-1-5-32-549
Account Operators           S-1-5-32-548
Read-only Domain Controllers S-1-5-21-1741355641-1558180958-3330877669-521
```

Mitglieder dieser Standard-ADGruppen haben weitreichende Systemrechte. Alle sonstigen, also eure benutzerdefinierten Gruppen haben das Level 2.

Für jede Gruppe werden die aktuellen Mitglieder angezeigt. In den Gruppen Enterprise-Admins und Schema-Admins sollten keine dauerhaften Mitglieder vorhanden sein. Daher werden deren Mitglieder immer rot dargestellt.

## privilegierte Accounts mit Gefährdungen

Sofern vorhanden werden hier alle Admins aufgelistet, zu denen mögliche Gefährdungen gefunden wurden:

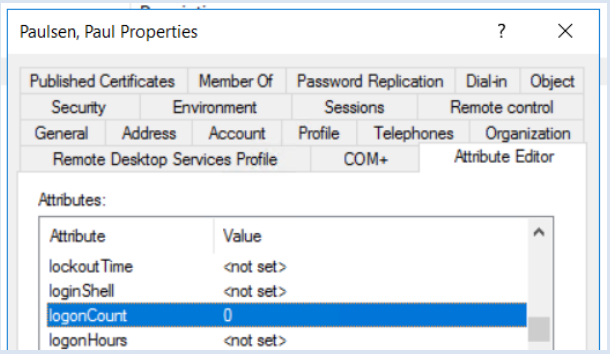
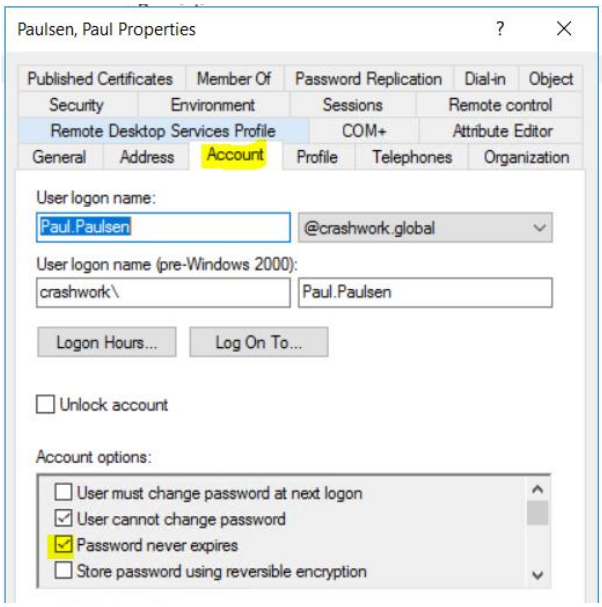
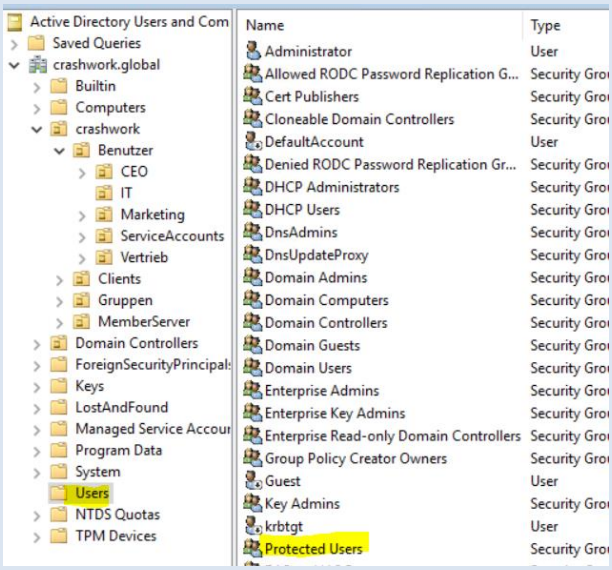
9/9 privilegierte Accounts sind gefaehrdet

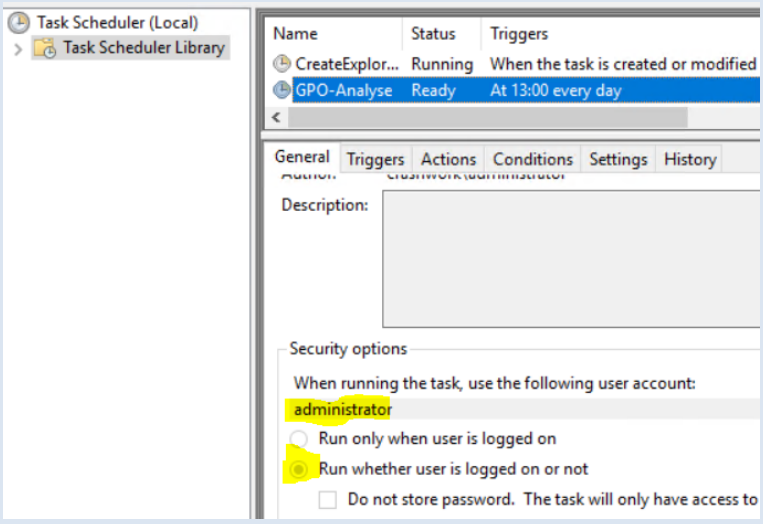
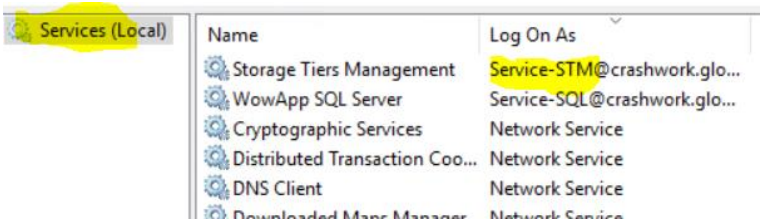
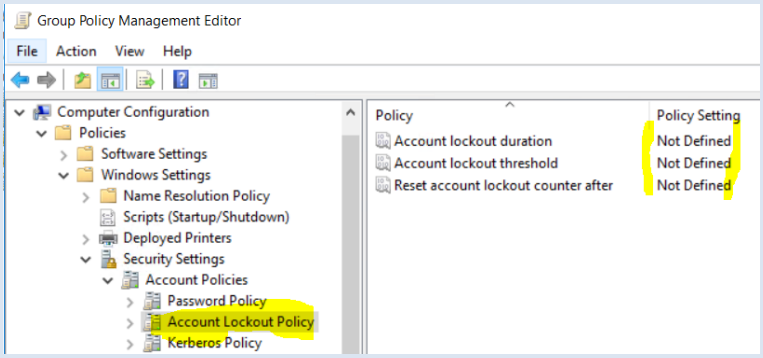
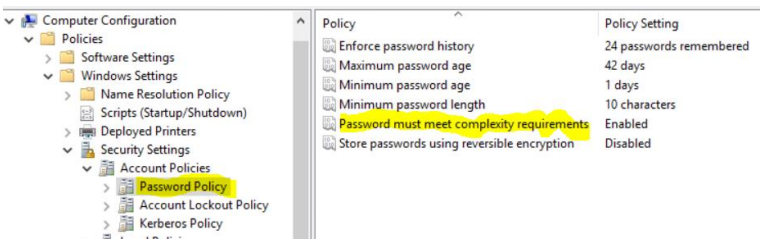
| Benutzer | Level | Policy | Lockout | ProtectedUser | Enabled | Description | SchedTasks | ServiceUser | LastLogon | PWfixed | PWendless | PWlastset | PWexpire | PWcomplex | PWreversible | PWlength | PWhistory | PWminAge | KRBpreAuth | requireSC | Bewertung |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Administrator | 1 | GPO | 0 | False | True | Built-in account for administering the computer/domain | 1 | | 2019-02-24 18:39:06 | False | True | 2018-06-26 07:11:09 | never | True | False | 7 | 24 | 1 | True | False | -kein Lockout, -kurzes PW, -ProtectedUser, -statisches PWD, -Taskuser |
| krbtgt | 1 | GPO | 0 | False | False | Key Distribution Center Service Account | | | always | False | True | 2018-06-26 07:15:36 | 2018-09-24 07:15:36 | True | False | 7 | 24 | 1 | True | False | -kurzes PW, -PWD abgelaufen |
| Service-SQL | 1 | GPO | 0 | False | True | ServiceAccount für SQL-Services | | 1 | 2019-02-24 18:49:26 | False | True | 2019-02-24 18:47:57 | never | True | False | 7 | 24 | 1 | True | False | -kein Lockout, -kurzes PW, -ProtectedUser, -ServiceUser, -statisches PWD |
| Service-STM | 1 | GPO | 0 | False | True | ServiceAccount für StorageTierManagement | | 3 | 2018-06-26 17:57:03 | False | True | 2018-10-15 07:46:45 | never | True | False | 7 | 24 | 1 | True | False | -kein Lockout, -kurzes PW, -ProtectedUser, -ServiceUser, -statisches PWD |
| Task-Monitoring | 1 | GPO | 0 | False | True | TaskAccount fürs Monitoring | 1 | | 2019-02-24 18:52:20 | False | True | 2019-02-24 18:51:34 | never | True | False | 7 | 24 | 1 | True | False | -kein Lockout, -kurzes PW, -ProtectedUser, -statisches PWD, -Taskuser |
| Frank.Furt | 2 | GPO | 0 | False | True | | | | never | True | True | 2018-06-26 07:17:40 | never | True | False | 7 | 24 | 1 | True | False | -kein Lockout, -kurzes PW, -nicht verwendet, -statisches PWD |
| Fred.Fredsen | 2 | GPO | 0 | False | True | | | | never | True | True | 2018-06-26 07:17:41 | never | True | False | 7 | 24 | 1 | True | False | -kein Lockout, -kurzes PW, -nicht verwendet, -statisches PWD |
| Olga.Ordnung | 2 | GPO | 0 | False | True | | | | never | True | True | 2018-06-26 07:17:41 | never | True | False | 7 | 24 | 1 | True | False | -kein Lockout, -kurzes PW, -nicht verwendet, -statisches PWD |
| Paul.Paulsen | 2 | GPO | 0 | False | True | | | | never | True | True | 2018-06-26 07:17:40 | never | True | False | 7 | 24 | 1 | True | False | -kein Lockout, -kurzes PW, -nicht verwendet, -statisches PWD |

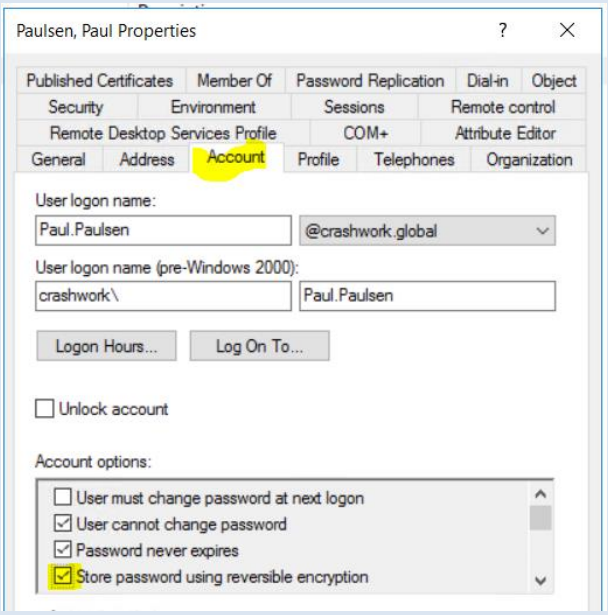
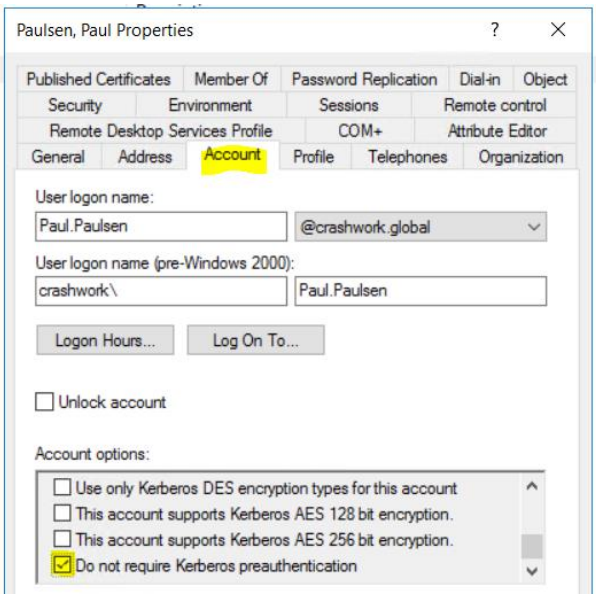
Die Zeilen sind mit einem roten Muster versehen. Etliche Spalten verweisen auf Eigenschaften mit Gefährdungs- oder Problempotential, z.B. die Spalte PWEndless. Hat ein Benutzer ein nicht ablaufendes Kennwort, dann wird diese „Verfehlung“ mit einer dunkelroten Zelle dargestellt. So kann man recht einfach die Probleme überschauen. Ebenso ist es aber auch möglich, dass Zellen grün dargestellt werden. Damit wird ein positiver Zustand hervorgehoben. So könnte z.B. die Mitgliedschaft in der Gruppe „Protected Users“ gewertet werden.

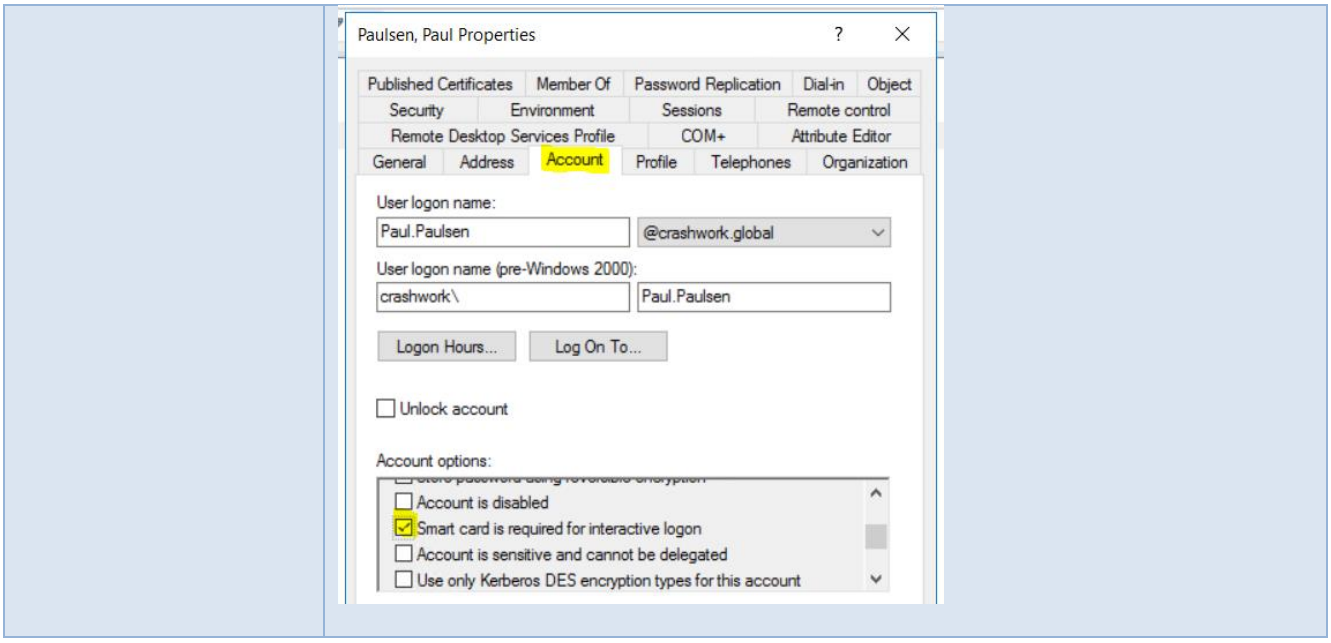
Die letzte Spalte „Bewertung“ listet noch einmal alle Probleme als Stichpunkte auf. Folgende Bewertungen sind möglich:

| Bewertungstext | Problem |
|---|---|
| -PWD laeuft ab | Das Passwort des Benutzers läuft in kürze ab. Hier gilt der Grenzwert PWAblauf_Warnung. |
| -PWD laeuft bald ab | Das Passwort des Benutzers läuft in weniger Tagen als der Grenzwert PWAblauf_Achtung ab. |
| -PWD abgelaufen | Das Passwort ist abgelaufen. |
| -kurzes PW | Das Passwort könnte kürzer als der Grenzwert PWmindestLaenge sein, da die Passwortrichtlinie zu schwach ist. |

| | |
|---|---|
| -nicht verwendet | Dieser privilegierte Account wurde noch nie verwendet:<br>  |
| -statisches PWD | Das Passwort ist statisch und muss nicht geändert werden:<br>  |
| -ProtectedUser | Der Benutzer ist Mitglied in einer Level-1-Gruppe, ist aber kein Mitglied der Gruppe „Protected Users“:<br>  |
| -kein PWD erforderlich | Der Benutzer kann auch ohne Passwort verwendet werden (Passwortrichtlinie mit 0 Zeichen Mindestlänge) |

| | |
|---|---|
| -Taskuser | Der Benutzer wird für mindestens eine geplante Aufgabe auf einem erreichten Server verwendet:<br>  |
| -ServiceUser | Der Benutzer wird für mindestens einen Dienst auf einem erreichten Server verwendet:<br>  |
| -kein Lockout | Für den Benutzer ist keine Kontosperre nach x fehlerhaften Anmeldungen definiert. Ein Online-BruteForce wäre möglich:<br>  |
| -schwaches PW | Für das Passwort schreibt die resultierende Passwortrichtlinie keine Komplexität vor und die Mindestlänge für sichere Passworte wurde nicht erzwungen:<br>  |
| -reversibles PW | Passworte können im ActiveDirectory umkehrbar gespeichert werden. Das ist natürlich nicht der Standardwert. Es gibt auch keinen guten Grund (mehr) das zu ändern: |

| | |
|---|---|
| |  |
| -kein PreAuth | Kerberos 5 hat die Kerberos PreAuthentication eingeführt. Ein Benutzer muss sich also anmelden, bevor er ein KerberosTicket erhält. Das ist der Systemstandard. So ist es Mist:<br>  |
| +SmardCard-protected | Vielleicht nutzt ihr SmartCards als alleiniges Anmeldeelement. Dann wird dieser Aspekt positiv gewertet: |




## privilegierte Accounts mit Gefährdungen, aber manuell ausgenommen

Manche Accounts können nicht weiter geschützt werden. Es ist ärgerlich, aber ganz offensichtlich stand Sicherheit bei Microsoft, den Softwareherstellern und auch bei den Admins nicht sonderlich hoch im Kurs. ☹

Damit diese (hoffentlich wenigen) Accounts nicht jedesmal die Anzeige rot färben könnt ihr sie in der Konfigurationsdatei ausnehmen. Die Admins werden dann blau statt rot in einer eigenen Tabelle angezeigt:

**1/5 privilegierte Accounts sind gefaehrdet, aber manuell ausgenommen**

| Benutzer | Level | Policy | Lockout | ProtectedUser | Enabled | Description | SchedTasks | ServiceUser | LastLogon | PWfixed | PWendless | PWlastset | PWexpire | PWcomplex | PWreversible | PWlength | PWhistory | PWminAge | KRBpreAuth | requireSC | Bewertung |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Paul.Paulsen | 2 | GPO | 5 | False | True | | | | never | True | True | 2018-06-26 07:17:40 | never | True | False | 10 | 24 | 1 | True | False | -nicht verwendet, -statisches PWD |

Dies gilt natürlich nur, wenn Gefährdungen erkannt wurden. Ist der Account sicher, dann wird er selbstverständlich in der nächsten Tabelle in grün dargestellt.

## privilegierte, sichere Accounts

Hier liegt das Ziel: möglichst alle privilegierten Accounts sollten gefährdungsfrei sein. Dann werden sie in dieser Ausgabetabelle angezeigt:

**1/5 privilegierte Accounts sind sicher**

| Benutzer | Level | Policy | Lockout | ProtectedUser | Enabled | Description | SchedTasks | ServiceUser | LastLogon | PWfixed | PWendless | PWlastset | PWexpire | PWcomplex | PWreversible | PWlength | PWhistory | PWminAge | KRBpreAuth | requireSC | Bewertung |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| krbtgt | 1 | GPO | 5 | False | False | Key Distribution Center Service Account | | | always | False | True | 2019-02-27 19:43:48 | 2019-05-28 19:43:48 | True | False | 10 | 24 | 1 | True | False | |

Mit einer gelben Level-Zelle werden alle Accounts der speziellen, systemeigenen Admingruppen dargestellt.

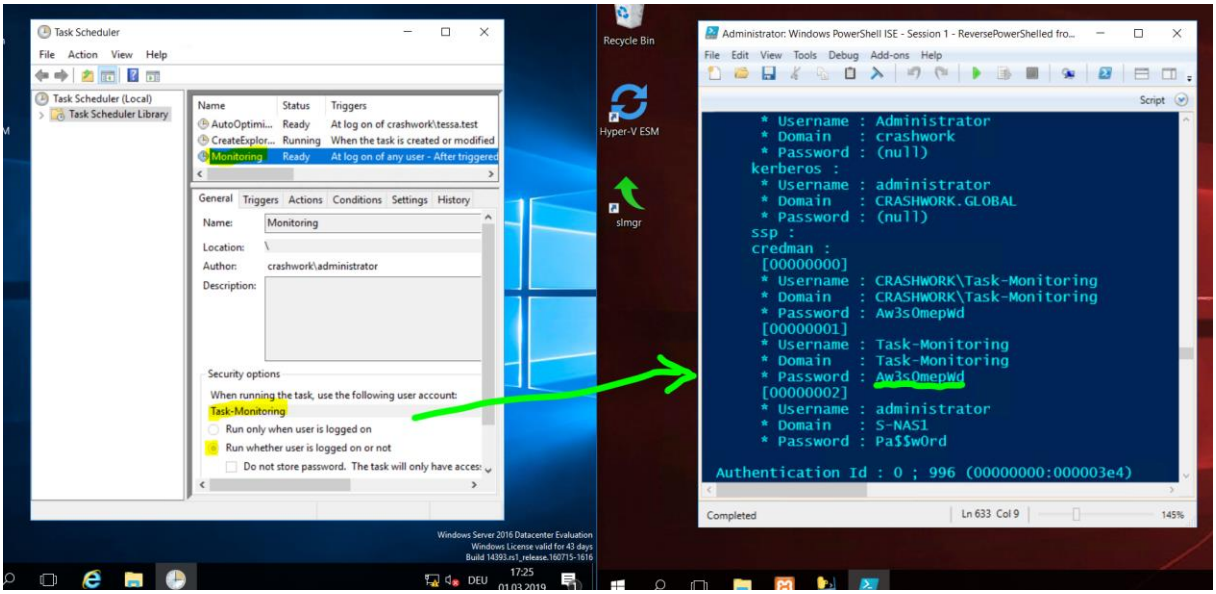
## geplante Aufgaben, die mit AdminKennungen laufen

Falls auf einem Server remote eine geplante Aufgabe gefunden wurde, die mit einer der AdminKennungen automatisch ausgeführt wird, dann wird diese Tabelle sichtbar:

**geplante Aufgaben, die mit AdminKennung laufen**

| Benutzer | Level | Hostname | TaskName | TaskAction | LastRunTime | NextRunTime | Description |
|---|---|---|---|---|---|---|---|
| Administrator | 1 | S-DC1 | GPO-Analyse | powershell.exe C:\Admin\GPO\GPO-Analyse.ps1 | 02/28/2019 17:28:39 | 03/01/2019 13:00:00 | |

Denkt immer daran, dass dieses Kennwort unter den richtigen Voraussetzungen im Klartext ausgelesen werden kann. Hier mal ein Taskuser auf einem Windows Server 2016 (links), dessen Kennwort im Hacking-System (rechts) angezeigt wird:




Arbeitet einfach die Liste ab und „entschärft“ die Rechte der eingesetzten Accounts!

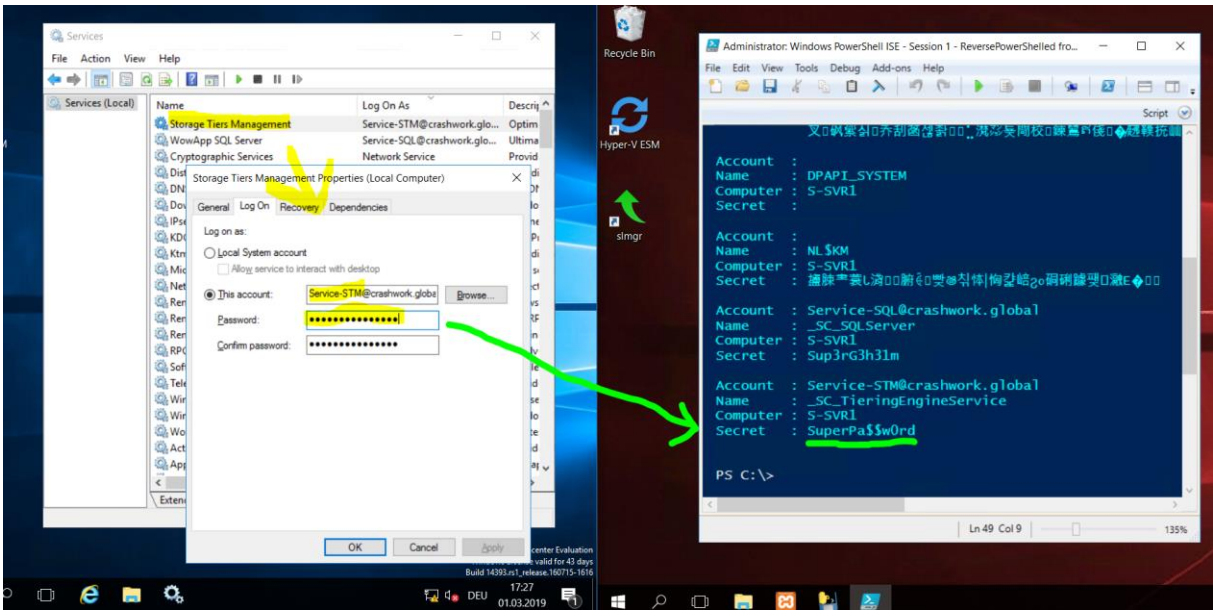
## Services, die mit AdminKennungen laufen

Auch diese Tabelle wird nur gezeigt, wenn durch das Remoting auf einem Server ein Dienst gefunden wurde, der eine AdminKennung verwendet:

**Services, die mit AdminKennung laufen**

| Benutzer | Level | Server | Service | State | StartMode | Pfad |
|---|---|---|---|---|---|---|
| Service-SQL | 1 | S-SVR1 | WowApp SQL Server | Running | Auto | C:\Program Files (x86)\Windows WowApp\SQL Bin\SQLServer.exe |

Auch diese Passworte müssen natürlich im System gespeichert werden. Und so können sie auch wieder ausgelesen werden. Links seht ihr die Adminkonsole der Dienste mit dem „geschützten“ Passwort. Rechts die Ansicht der Hacker…




Also gilt auch hier: arbeitet die Liste ab und „entschärft“ die Benutzerrechte!

## Zusammenfassung der Scriptausführung

Die letzte Tabelle enthält die Zusammenfassung. Hier kann abgelesen werden, ob es beim Remoting Probleme gab. Sollten ein oder mehrere Server nicht erreichbar sein, dann wird natürlich die Auswertung der ServiceAdmins und TaskAdmins lückenhaft sein:

**Zusammenfassung:**

| | |
|---|---|
| Generiert auf: | S-DC1 |
| Scriptversion: | V1.14 |
| Scan-Dauer | 8 sec |
| Anzahl Server: | 4 |
| ...davon erreichbar: | 4 |
| ...mit Fehler: | 0 |
| Anzahl Gruppen: | 19 |
| Anzahl Admins: | 9 |
| ...mit Level 1: | 5 |
| ...mit Level 2: | 4 |
| ...gefaehrdet: | 9 |
| ...sicher: | 0 |
| Anzahl Admins mit Task: | 2 |
| Anzahl Admins mit Service: | 4 |

# Möglichkeiten zur Vermeidung von Gefährdungen

## Prinzip Least Privilege

Das ist der ultimative Anspruch: so wenig Accounts wie möglich sollten mit höheren Rechten unterwegs sein! Wie oft habe ich schon Accounts gesehen, die einen Task oder einen Service befeuern und selber Mitglieder der Gruppe DomainAdmins sind!!!

Seit Windows Server 2012 können Group Managed Service Accounts eingesetzt werden. Diese werden im Betriebssystem wesentlich besser geschützt und zusätzlich wird das Kennwort vollautomatisch regelmäßig geändert! OK, nicht jede Software kann damit umgehen und die Konfiguration ist (auch in Windows Server 2019) immer noch nur über die PowerShell möglich. Aber hier kann vielleicht mein öffentliches Script „gMSA-Admin“ weiterhelfen!

Überlegt bei der Einrichtung einer neuen Software, welche Rechte wo benötigt werden. Fragt beim Hersteller nach. Klar läuft das Produkt mit DomainAdmin-Rechten… wie eine Firewall, in der alle Ports offen sind! Aber das geht bestimmt besser. 😊

## Konfigurationen im Active Directory

Viele von den zusätzlichen Schaltern, wie „Passwort kann nicht geändert werden“, „Passwort läuft nicht ab“, … sind einfach tabu. Ich selber habe einen Account in meinem Netzwerk, der hohe Rechte benötigt und nicht anders geschützt werden kann. Hier hilft mir dieses Script, denn es informiert mich, wenn das Passwort ausläuft. Dann generiere ich ein neues und trage es zeitgleich im AD und in der Software ein. Tada.

Bitte informiert euch über die Gruppe Protected Users.

Password-Setting-Objects können zusätzlich zur Kennwort-Richtlinie die Anforderungen für Passworte und die Sperrdefinitionen verwendet werden. Erzwingt an den richtigen Stellen maximal sichere Richtlinien.

Ebenfalls interessant könnte mein Security-Scoping sein. Dabei gliedere ich die zu administrierenden Systeme in einzelne Bereiche, denen jeweils neue administrative Gruppen zugewiesen werden.

Eine Besonderheit stellt der Benutzer KRBTGT dar. Dessen Passwort(hash) wird verwendet, um die Kerberos-TGT’s zu verschlüsseln. Auf diese baut die gesamte Sicherheit des AD auf. Dann muss doch dieses Passwort besonderen Ansprüchen genügen, oder? Leider sieht das im Default anders aus, denn das Passwort des KRBTGT-Users läuft NIEMALS ab! Ändert es bitte ebenso regelmäßig, wie die anderen Passworte. Dabei ist aber besondere Vorsicht geboten! Diesem Thema widme ich mich separat.

Probiert es einfach aus. Dann wird bald alles schön grün:

# Script PrivilegedADUser-Analyse.ps1
# 2019-02-28

**AdminGruppen Level 1 und 2**

| Gruppe | Level | Mitglieder |
|---|---|---|
| Account Operators | 1 | |
| Administrators | 1 | Administrator |
| Backup Operators | 1 | |
| Domain Admins | 1 | Administrator |
| Domain Controllers | 1 | |
| Enterprise Admins | 1 | |
| Print Operators | 1 | |
| Read-only Domain Controllers | 1 | |
| Replicator | 1 | |
| Schema Admins | 1 | |
| Server Operators | 1 | |
| GG-Admin-AD | 2 | |
| GG-Admin-Clients | 2 | |
| GG-Admin-Drucker | 2 | |
| GG-Admin-Freigaben | 2 | |
| LD-Admin-AD | 2 | |
| LD-Admin-Clients | 2 | |
| LD-Admin-Drucker | 2 | |
| LD-Admin-Freigaben | 2 | |

**2/2 privilegierte Accounts sind sicher**

| Benutzer | Level | Policy | Lockout | ProtectedUser | Enabled | Description | SchedTasks | ServiceUser | LastLogon | PWfixed | PWendless | PWlastset | PWexpire | PWcomplex | PWreversible | PWlength | PWhistory | PWminAge | KRBpreAuth | requireSC | Bewertung |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Administrator | 1 | GPO | 5 | True | True | Built-in account for administering the computer/domain | | | 2019-02-24 18:39:06 | False | False | 2019-02-27 19:42:23 | 2019-04-10 20:42:23 | True | False | 10 | 24 | 1 | True | False | +ProtectedUser |
| krbtgt | 1 | GPO | 5 | False | False | Key Distribution Center Service Account | | | always | False | True | 2019-02-27 19:43:48 | 2019-05-28 19:43:48 | True | False | 10 | 24 | 1 | True | False | |