

Inhalt

Theorie	1
Zweck	1
NTLM im Active Directory?	2
NTLM-Audit.....	3
NTLM-Deaktivierung	5
Umstellung	8
IST-Situation.....	8
Umstellung auf NoNTLM.....	9
Nachkontrollen und Troubleshooting	10
Exchange Server (2016)	10
VPN-Einwahl mit Windows RemoteAccess	10
RemoteDesktop in einer RDS-Farm	11
PKI-Sperrlistenveröffentlichung	12

Theorie

Zweck

Anmeldeinformationen sollen vertrauliche Daten und Dienste schützen. In unseren Infrastrukturen finden wir an allen relevanten Punkten Authentifizierungslösungen. Die meisten verwenden dabei Benutzernamen und Kennworte. Wer die Kombination aus beiden kennt, kann auf die Services und Daten zugreifen.

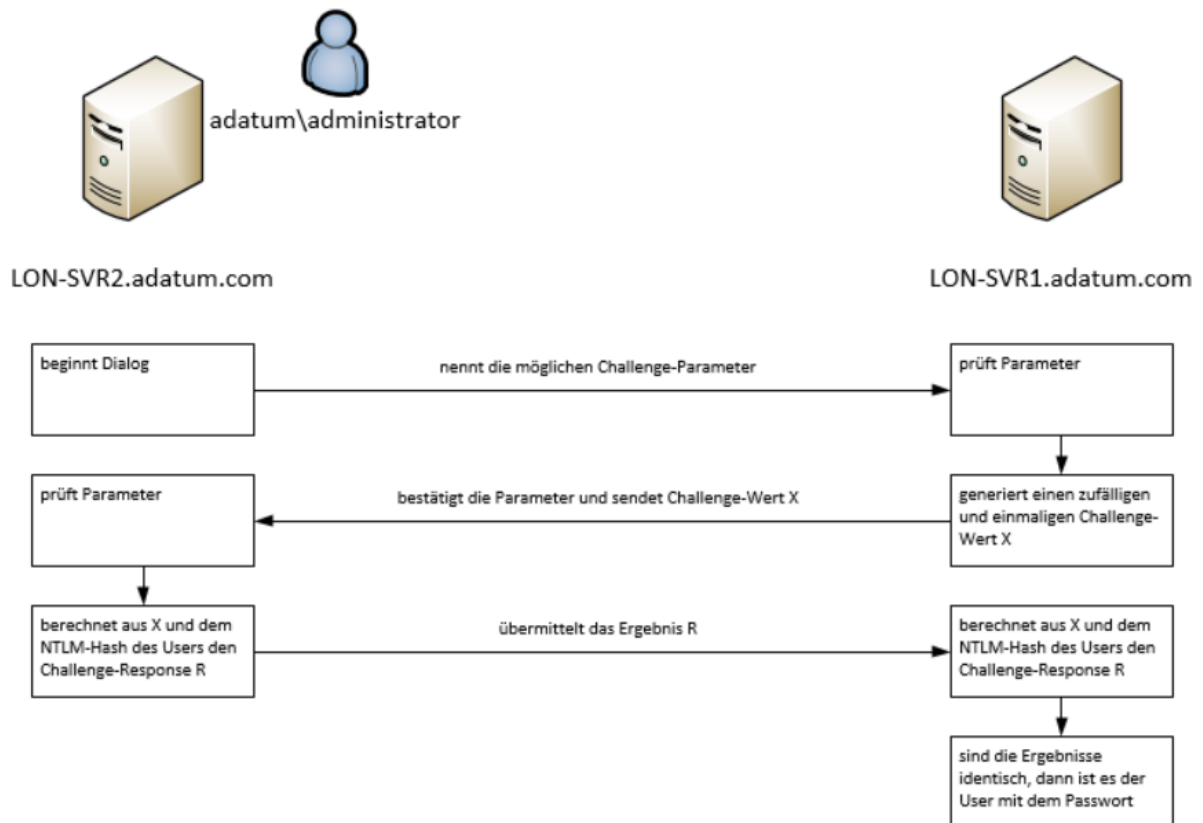
Wir bemühen uns, dass diese Anmeldeinformationen unser Geheimnis bleiben: wir schreiben keine Kennworte frei zugänglich auf, nutzen Password-Safes und jeder Benutzeraccount hat ein anderes Kennwort (oder? 😊)

Dabei ist das nur ein Teil der Lösung. Denn insgesamt sind 3 Punkte in einer Authentifizierung zu schützen:

1. Die Identität – also der Benutzer, der seine Kennworte geheimhält.
2. Der Authentifizierungsdienst, der die Anmeldeinformationen prüfen soll – auch dieser muss die Anmeldeinformationen kennen
3. Und zuletzt: der Übertragungsweg der Anmeldeinformationen von der Identität zum Authentifizierungsdienst.

Der Übertragungsweg ist generell als unsicher zu betrachten – bitte auch in internen Netzwerken! Ich habe schon einige Beispiele in meinem Hacking-Blog vorgestellt, mit denen es gelingt, die Anmeldeinformationen während der Authentifizierung abzugreifen. Sehr gerne nutze ich dazu den Responder. In Kombination mit MultiRelay.py kann sehr einfach die Anmeldung eines Administrators umgeleitet werden, um als Angreifer ein Zielsystem zu übernehmen. Aber auch ein Responder alleine kann bereits Anmeldeinfos aufzeichnen. Diese könnten offline geknackt werden.

Jedesmal, wenn das Passwort im Klartext oder gehashed übertragen wird, kann ein Angriff erfolgreich sein. Über viele Jahre war NTLM eine Möglichkeit, die Anmeldung vorzunehmen. Bei der aktuellen Version NTLMv2 wird nicht das Passwort bzw. der Hash des Passwortes übertragen. Stattdessen wird eine Challenge (eine Zufallszahl) mit dem Hash vom Client berechnet und zum Server übertragen. Der Server kann die Challenge ebenfalls berechnen (er muss dazu Zugriff auf den Hash des Passwortes vom Benutzer haben). Sind die Ergebnisse der Challenge identisch, dann kann der Server dem Client einen Zugriff gestatten:



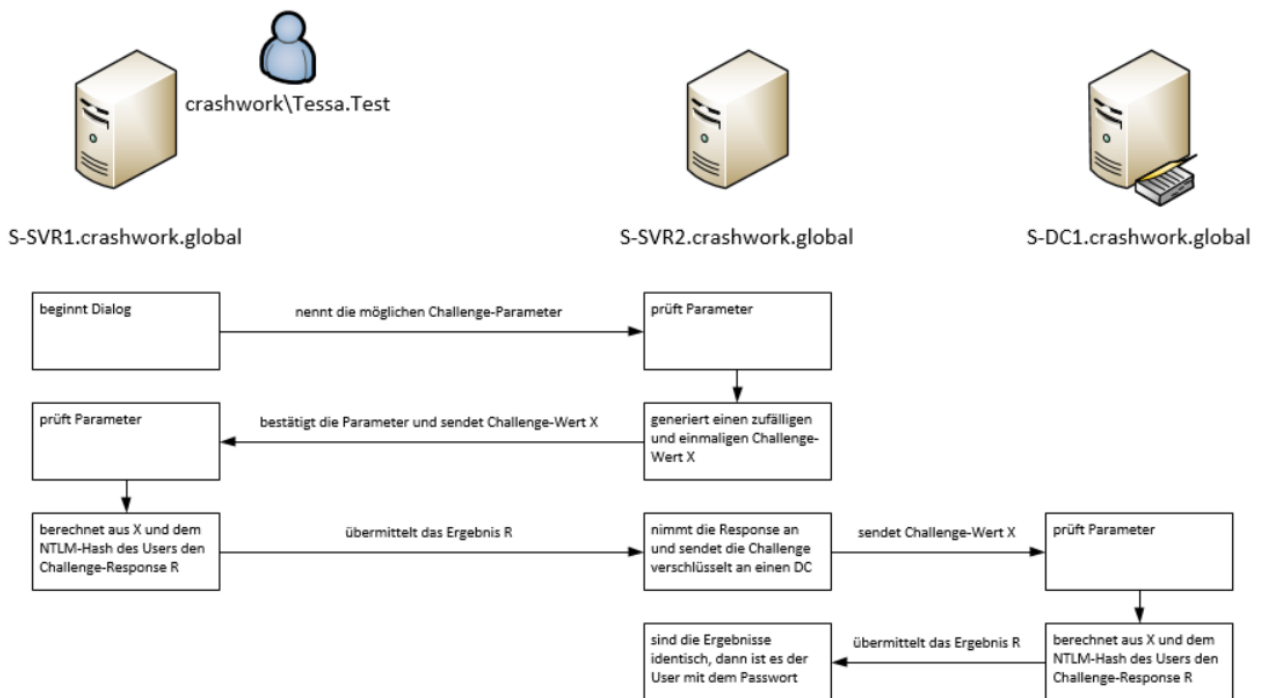
Aber ein Angreifer kann aus der Zufallszahl und dem Ergebnis der Challenge den NTLM-Hash bzw. das dazugehörige Passwort OFFLINE berechnen.

Seit vielen Jahren empfiehlt Microsoft nun schon, NTLM im Active Directory zu verbieten und statt dessen vollständig auf Kerberos zu setzen. Dieses Protokoll ist wesentlich sicherer.

Natürlich müssen dafür alle Abhängigkeiten von NTLM geklärt werden, damit es keine Anmeldeprobleme oder Zugriffsprobleme im Netzwerk gibt. Dies beginnt bei der Nutzung kompatibler Clients und Services und umfasst auch die korrekte Konfiguration aller Dienste. Das Standardverfahren zur Deaktivierung von NTLM sieht daher vor, dass über einen Referenzzeitraum ein Audit der Nutzung von NTLM auf allen **DomainControllern** mitläuft. Erst wenn alle Services angepasst sind, wird die Verwendung von NTLM abgeschaltet.

NTLM im Active Directory?

Aber warum werden die DomainController überwacht? Ganz einfach: der Zielservers kann nicht von jedem Benutzer den PasswortHash kennen. Er muss einen DomainController fragen. Das sieht dann so aus:



Im Wireshark kann man die Pakete sehr schön erkennen. Ich habe das Szenario einmal mit 4 Servern nachgestellt:

- Auf S-SVR1 (172.16.1.11) ist die Benutzerin Tessa.Test angemeldet.
- Sie greift auf den Server S-SVR2 (172.16.1.12) mit dem Windows Explorer zu. Dabei verwendet Sie den UNC-Pfad [\\172.16.1.12](#) – Kerberos kann also nicht verwendet werden, denn dafür sind DNS-Namen erforderlich.
- S-SVR2 antwortet im Paket 18 mit der NTLM-Challenge – der Zufallszahl.
- S-SVR1 berechnet aus dem NTLM-Hash von Tessa (dieser liegt nach der Anmeldung im LSA vor) die Challenge-Response und sendet diese an S-SVR2 im Paket 19
- S-SVR2 baut nun eine Verbindung zum S-DC1 (172.16.1.1) auf. Im Paket 26 überträgt er mit dem Longterm-Service-Key aus seiner Kerberos-Anmeldung die Challenge und den Challenge-Response vom S-SVR1 an den Domain Controller.
- Der DC kennt alle Benutzer und hat deren NTLM-Hashes gespeichert. Er entschlüsselt die ChallengeWerte aus Paket 26, berechnet selbst die Response und antwortet S-SVR2 in Paket 27, dass die Response korrekt ist. Somit weiß S-SVR2 nun, dass Tessa.Test auch die echte Tessa ist.
- Im Paket 28 bestätigt S-SVR2 nun die erfolgreiche Anmeldung an S-SVR1 – Tessa kann nun mit dem Fileserver arbeiten

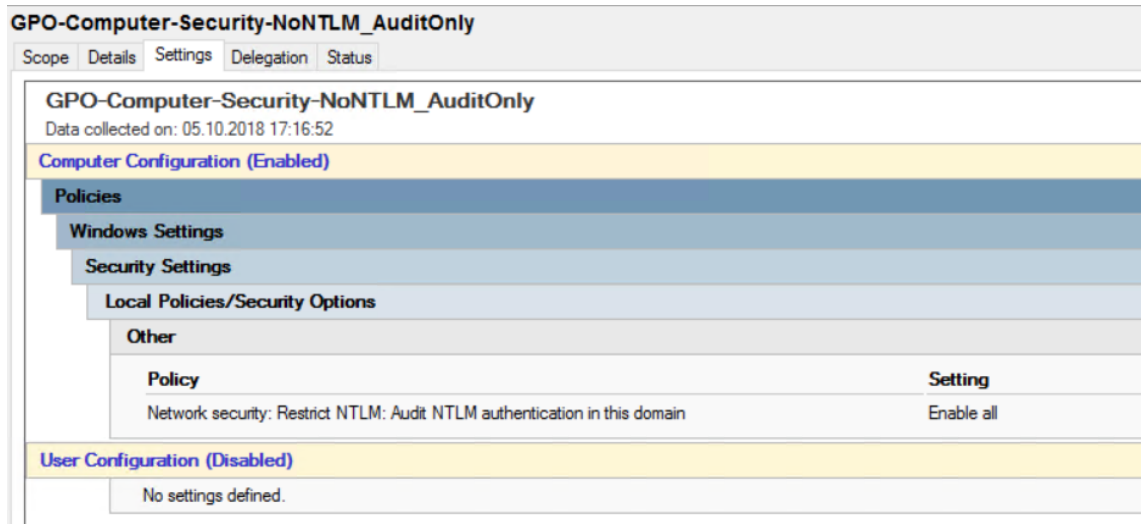
No.	Time	Source	Destination	Protocol	Length	Info
10	1.854978	172.16.1.11	172.16.1.12	TCP	66	49741 → 445 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
11	1.860633	172.16.1.12	172.16.1.11	TCP	66	445 → 49741 [SYN, ACK, ECN] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
12	1.862006	172.16.1.11	172.16.1.12	TCP	54	49741 → 445 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
13	1.862022	172.16.1.11	172.16.1.12	SMB	213	Negotiate Protocol Request
14	1.863072	172.16.1.12	172.16.1.11	SMB2	306	Negotiate Protocol Response
15	1.863416	172.16.1.11	172.16.1.12	SMB2	232	Negotiate Protocol Request
16	1.865096	172.16.1.12	172.16.1.11	SMB2	366	Negotiate Protocol Response
17	1.876397	172.16.1.11	172.16.1.12	SMB2	220	Session Setup Request, NTLMSSP_NEGOTIATE
18	1.877144	172.16.1.12	172.16.1.11	SMB2	413	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
19	1.878492	172.16.1.11	172.16.1.12	SMB2	709	Session Setup Request, NTLMSSP_AUTH, User: crashwork\Tessa.Test
20	1.892126	172.16.1.12	172.16.1.1	TCP	66	49775 → 49670 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
21	1.892610	172.16.1.1	172.16.1.12	TCP	66	49670 → 49775 [SYN, ACK, ECN] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
22	1.892639	172.16.1.12	172.16.1.11	TCP	54	445 → 49741 [ACK] Seq=924 Ack=1159 Win=2101248 Len=0
23	1.893534	172.16.1.12	172.16.1.1	TCP	54	49775 → 49670 [ACK] Seq=1 Ack=1 Win=262656 Len=0
24	1.893559	172.16.1.12	172.16.1.1	DCERPC	273	Bind: call_id: 10, Fragment: Single, 3 context items: RPC_NETLOGON V1.0 (32bit)
25	1.895558	172.16.1.1	172.16.1.12	DCERPC	182	Bind_ack: call_id: 10, Fragment: Single, max_xmit: 5840 max_recv: 5840, 3 results
26	1.908072	172.16.1.12	172.16.1.1	RPC_NE...	1022	NetrLogonSamLogonEx request
27	1.910866	172.16.1.1	172.16.1.12	RPC_NE...	974	NetrLogonSamLogonEx response
28	1.912900	172.16.1.12	172.16.1.11	SMB2	159	Session Setup Response
29	1.917292	172.16.1.11	172.16.1.12	SMB2	166	Tree Connect Request Tree: \\172.16.1.12\IPC\$
30	1.918534	172.16.1.12	172.16.1.11	SMB2	138	Tree Connect Response

Was ist nun daran problematisch? Jedes Paket wie #19 stellt ein Risiko dar, da es abgefangen und geknackt werden kann!

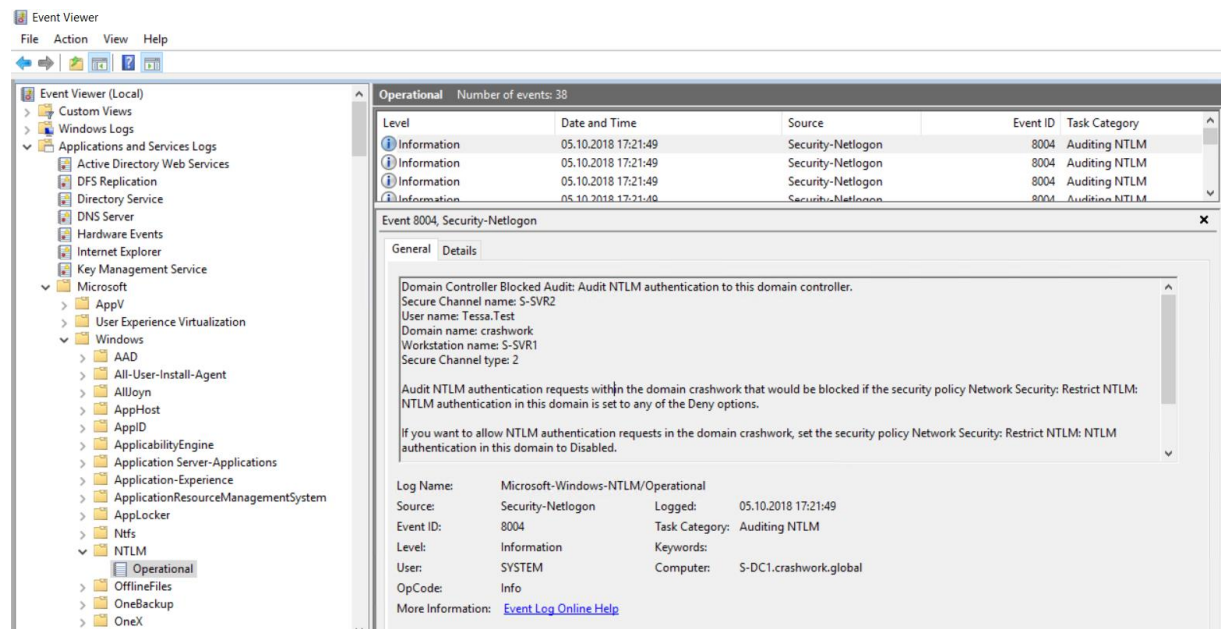
NTLM-Audit

Aktivieren wir nun das Audit auf allen DomainControllern, dann schreiben diese in ein spezielles Eventlog alle eingehenden Anfragen von DomainMembern nieder. Diese können dann zentral ausgewertet werden.

Die Aktivierung wird über eine GPO vorgenommen:



Diese wirkt auf alle DomainController. Ergebnisse des Audits findet man in diesem Eventlog:



Das Auslesen der Eventlogs über ALLE DomainController und Analysieren kann man mit der PowerShell etwas vereinfachen:

```
Invoke-Command -ComputerName (Get-ADDomain).ReplicaDirectoryServers -ScriptBlock {
    Get-WinEvent -Path C:\Windows\System32\Winevt\Logs\Microsoft-Windows-NTLM%4Operational.evtx |
    Select-Object -Property @{ n='DC' ; e={ $env:COMPUTERNAME } },
    @{ n='Datetime' ; e={ (Get-Date -Date $_.TimeCreated -Format u) -replace 'z' } },
    @{ n='Client' ; e={ (($_.Message -split "`n" | select-string 'Workstation') -split ':')[1].trim() } },
    @{ n='Server' ; e={ (($_.Message -split "`n" | select-string 'Secure Channel name') -split ':')[1].trim() } },
    @{ n='Domain' ; e={ (($_.Message -split "`n" | select-string 'Domain name') -split ':')[1].trim() } },
    @{ n='User' ; e={ (($_.Message -split "`n" | select-string 'User name') -split ':')[1].trim() } }
} | Format-Table -Property DC,Datetime,Client,Server,Domain,User
```

Die Ergebnisse werden tabellarisch dargestellt:

DC	Datetime	Client	Server	Domain	User
--	-----	-----	-----	-----	----
S-DC1	2018-10-05 17:21:49	S-SVR1	S-SVR2	crashwork	Tessa.Test
S-DC1	2018-10-05 17:21:49	S-SVR1	S-SVR2	crashwork	Tessa.Test
S-DC1	2018-10-05 17:21:49	S-SVR1	S-SVR2	crashwork	Tessa.Test
S-DC1	2018-10-05 17:21:49	S-SVR1	S-SVR2	crashwork	Tessa.Test
S-DC1	2018-10-05 17:20:31	S-SVR1	S-SVR2	crashwork	Tessa.Test
S-DC1	2018-10-05 17:20:31	S-SVR1	S-SVR2	crashwork	Tessa.Test
S-DC1	2018-10-05 17:20:31	S-SVR1	S-SVR2	crashwork	Tessa.Test
S-DC1	2018-10-05 17:20:31	S-SVR1	S-SVR2	crashwork	Tessa.Test
S-DC1	2018-10-05 17:20:03	S-SVR1	S-SVR2	crashwork	Tessa.Test
S-DC1	2018-10-05 17:20:03	S-SVR1	S-SVR2	crashwork	Tessa.Test
S-DC1	2018-10-05 17:20:03	S-SVR1	S-SVR2	crashwork	Tessa.Test
S-DC1	2018-10-05 17:20:03	S-SVR1	S-SVR2	crashwork	Tessa.Test
S-DC1	2018-10-05 17:17:52	S-SVR1	S-SVR2	crashwork	Administrator
S-DC1	2018-10-05 17:17:52	S-SVR1	S-SVR2	crashwork	Administrator
S-DC1	2018-10-05 17:17:52	S-SVR1	S-SVR2	crashwork	Administrator
S-DC1	2018-10-05 17:16:13	S-SVR1	S-SVR2	crashwork	Administrator
S-DC1	2018-10-05 17:16:13	S-SVR1	S-SVR2	crashwork	Administrator
S-DC1	2018-10-05 17:16:13	S-SVR1	S-SVR2	crashwork	Administrator
S-DC1	2018-10-05 17:14:31	S-SVR1	S-SVR2	crashwork	Administrator
S-DC1	2018-10-05 17:14:31	S-SVR1	S-SVR2	crashwork	Administrator
S-DC1	2018-10-05 17:14:31	S-SVR1	S-SVR2	crashwork	Administrator
S-DC1	2018-10-05 17:12:37	S-SVR1	S-SVR2	crashwork	Administrator
S-DC1	2018-10-05 17:12:37	S-SVR1	S-SVR2	crashwork	Administrator

NTLM-Deaktivierung

Wurden alle Konfigurationen an Kerberos angepasst, dann kann mit einer weiteren GPO NTLM auf den DomainControllern deaktiviert werden. Clients werden zwar weiterhin ihre Challenge-Responses bei Bedarf verwenden und Server werden diese wie bisher an den DomainController weiterleiten. Aber der DC wird sich weigern, die Response für den Server zu berechnen. Der Server kann also nicht mehr bestimmen, ob der Client für den Zugriff autorisiert ist – und daher lehnt er den Verbindungsaufbau ab!

Nur dieses Nichtzustandekommen der Verbindung wird der IT-Abteilung gemeldet. Und diese kann sich dann dem Problem annehmen und Konfigurationen zur Verbesserung der Sicherheit anpassen. Das klingt nach Arbeit? Jup – Sicherheit hat eben ihren Preis!

Für Systeme, die nicht ohne NTLM-Auth auskommen kann in der GPO eine Ausnahme definiert werden. Mit diesen sollte man aber sparsam umgehen – sonst bringt es nichts. Die Logfiles können zur Bestimmung der Ausnahmen dienen.

Die Deaktivierung (mit einer Ausnahme für den Server S-SVR3) könnte nun so aussehen:

GPO-Computer-Security-NoNTLM	
Scope	Details Settings Delegation Status
GPO-Computer-Security-NoNTLM	
Data collected on: 05.10.2018 18:04:54	
Computer Configuration (Enabled)	
Policies	
Windows Settings	
Security Settings	
Local Policies/Security Options	
Other	
Policy	Setting
Network security: Restrict NTLM: Add server exceptions in this domain	S-SVR3.crashwork.global
Network security: Restrict NTLM: NTLM authentication in this domain	Deny all
User Configuration (Disabled)	
No settings defined.	

Wenn nun NACH der Deaktivierung von NTLM ein Client eine NTLM-Session zu einem Server anfragt und dieser Server die Anmeldeinformationen bei einem DomainController prüfen möchte, dann wird dieser keine positive Auskunft mehr erteilen. Die Verbindung wird fehlschlagen:

Im WireShark kann man den Prozess wieder schön beobachten:

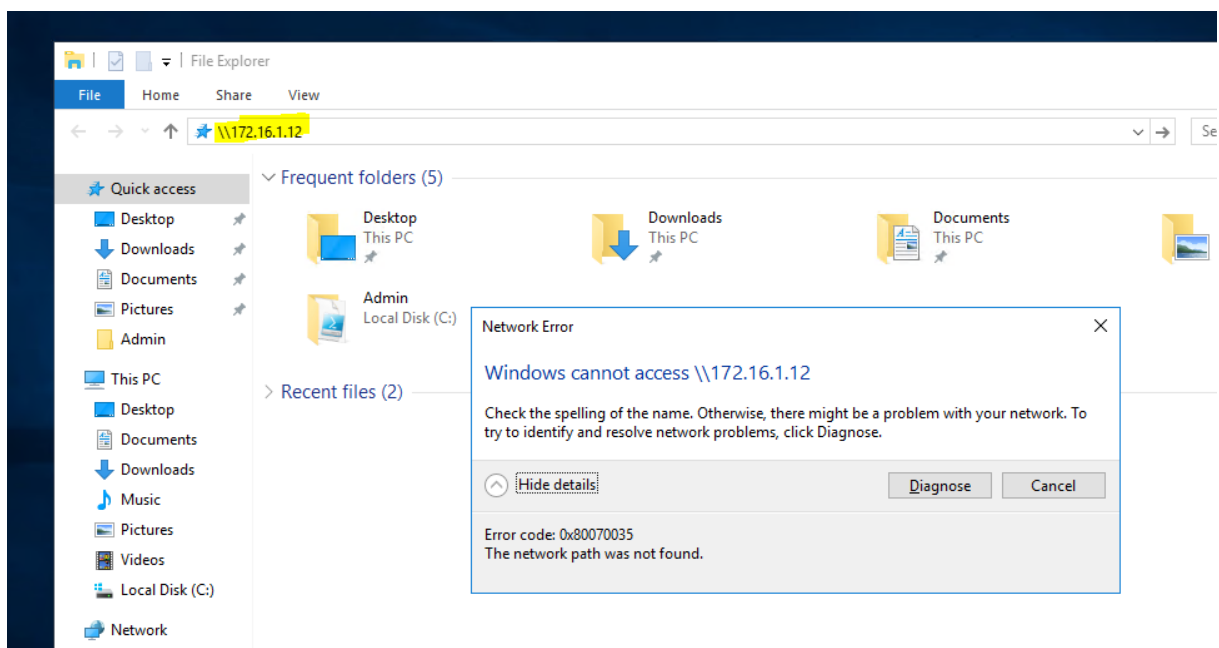
- Der Client mit der IP 172.16.1.11 fragt im Paket 14 eine SMB-Session beim Server 172.16.1.12 an.
- Im Paket 15 sendet der Server eine NTLM-Challenge zum Client
- Der Client nutzt den NTLM-Hash des Benutzers, um daraus und aus der Challenge die Challenge-Response zu berechnen. Diese sendet er im Paket 16 an den Server.

Man sieht hier ganz deutlich, dass die NTLM-Deaktivierung NICHT von den Clients und den Servern erzwungen wird! Die NTLM-Responses werden nach wie vor über das unsichere Netzwerk geleitet!!!

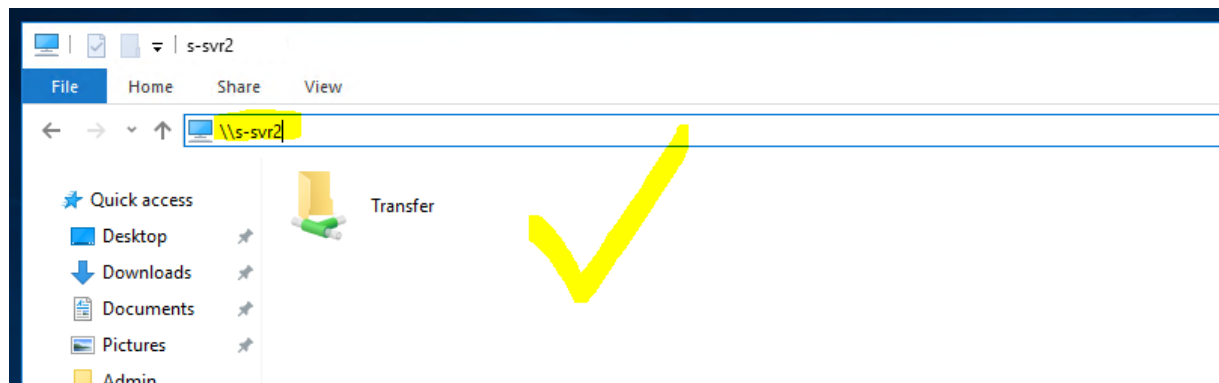
- Der Server fragt nun beim Domaincontroller (IP 172.16.1.1) nach, indem er die Challenge-Response mit Kerberos verschlüsselt im Paket 30 an den DC sendet.
- Der DC weiß, dass er keine NTLM-Authentifizierungen mehr prüfen soll. Nach einer kurzen Prüfung, ob der Server in der Ausnahmeliste steht, verweigert er in Paket 31 die Authentifizierung.
- Der Server informiert den Client im Paket 32, dass diese Anmeldeform nicht unterstützt wird

Time	Source	Destination	Protocol	Length	Info
7	172.16.1.11	172.16.1.12	TCP	66	49746 → 445 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
8	172.16.1.12	172.16.1.11	TCP	66	445 → 49746 [SYN, ACK, ECN] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
9	172.16.1.11	172.16.1.12	TCP	54	49746 → 445 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
10	172.16.1.11	172.16.1.12	SMB	213	Negotiate Protocol Request
11	172.16.1.12	172.16.1.11	SMB2	306	Negotiate Protocol Response
12	172.16.1.11	172.16.1.12	SMB2	232	Negotiate Protocol Request
13	172.16.1.12	172.16.1.11	SMB2	366	Negotiate Protocol Response
14	172.16.1.11	172.16.1.12	SMB2	220	Session Setup Request, NTLMSSP_NEGOTIATE
15	172.16.1.12	172.16.1.11	SMB2	375	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
16	172.16.1.11	172.16.1.12	SMB2	673	Session Setup Request, NTLMSSP_AUTH, User: crashwork\Tessa.Test
17	172.16.1.12	172.16.1.1	TCP	66	49772 → 135 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
18	172.16.1.12	172.16.1.1	TCP	54	445 → 49746 [ACK] Seq=886 Ack=1123 Win=2101248 Len=0
19	172.16.1.1	172.16.1.12	TCP	66	135 → 49772 [SYN, ACK, ECN] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
20	172.16.1.12	172.16.1.1	TCP	54	49772 → 135 [ACK] Seq=1 Ack=1 Win=262656 Len=0
21	172.16.1.12	172.16.1.1	DCERPC	214	Bind: call_id: 7, Fragment: Single, 3 context items: EPMV4 V3.0 (32bit NDR), EPMV4 V3.0 (64bit NDR), EPMV4 V3.0 (64bit NDR)
22	172.16.1.1	172.16.1.12	DCERPC	162	Bind_ack: call_id: 7, Fragment: Single, max_xmit: 5840 max_recv: 5840, 3 results: Provide
23	172.16.1.12	172.16.1.1	EPN	222	Map request, RPC_NETLOGON, 32bit NDR
24	172.16.1.1	172.16.1.12	EPN	322	Map response, RPC_NETLOGON, 32bit NDR, RPC_NETLOGON, 32bit NDR
25	172.16.1.12	172.16.1.1	TCP	66	49773 → 49670 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
26	172.16.1.1	172.16.1.12	TCP	66	49670 → 49773 [SYN, ACK, ECN] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
27	172.16.1.12	172.16.1.1	TCP	54	49773 → 49670 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
28	172.16.1.12	172.16.1.1	DCERPC	273	Bind: call_id: 6, Fragment: Single, 3 context items: RPC_NETLOGON V1.0 (32bit NDR), RPC_NETLOGON V1.0 (64bit NDR), RPC_NETLOGON V1.0 (64bit NDR)
29	172.16.1.1	172.16.1.12	DCERPC	182	Bind_ack: call_id: 6, Fragment: Single, max_xmit: 5840 max_recv: 5840, 3 results: Provide
30	172.16.1.12	172.16.1.1	RPC_NE-	990	NetrLogonSamLogonEx request
31	172.16.1.1	172.16.1.12	RPC_NE-	174	NetrLogonSamLogonEx response
32	172.16.1.12	172.16.1.11	SMB2	131	Session Setup Response, Error: STATUS_NOT_SUPPORTED

Am Client sieht meine Testuserin ein AccessDenied:



Nutzt meine Testuserin dagegen den Namen des Servers im Windows Explorer (und hängt der Client den DomainName an, damit ein FQDN gebildet werden kann), dann klappt die Verbindung:



Im Wireshark sieht man deutlich, dass nun der Client (172.16.1.11) zum DomainController (172.16.1.1) ein TGS-Request sendet, in dem er für den Service CIFS (SMB) auf dem Server S-SVR2 (172.16.1.12) ein SessionTicket erbittet. Da der Name passt und auch alle anderen Voraussetzungen erfüllt sind, erhält der Client das Ticket. Dieses sendet er dann an den Server. Und dieser bestätigt die Verbindung:

Time	Source	Destination	Protocol	Length	Info
5	1.618159	172.16.1.11	172.16.1.12	TCP	66 49752 → 445 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
6	1.620716	172.16.1.12	172.16.1.11	TCP	66 445 → 49752 [SYN, ACK, ECN] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
7	1.625468	172.16.1.11	172.16.1.12	TCP	54 49752 → 445 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
8	1.625487	172.16.1.11	172.16.1.12	SMB	213 Negotiate Protocol Request
9	1.636449	172.16.1.12	172.16.1.11	SMB2	306 Negotiate Protocol Response
10	1.638677	172.16.1.11	172.16.1.12	SMB2	232 Negotiate Protocol Request
11	1.652001	172.16.1.12	172.16.1.11	SMB2	366 Negotiate Protocol Response
12	1.653619	172.16.1.11	172.16.1.1	TCP	66 49753 → 88 [SYN, ECN, CWR]
13	1.654098	172.16.1.1	172.16.1.11	TCP	66 88 → 49753 [SYN, ACK, ECN]
14	1.654360	172.16.1.11	172.16.1.1	TCP	54 49753 → 88 [ACK] Seq=1 Ack=
15	1.654371	172.16.1.11	172.16.1.1	TCP	1514 49753 → 88 [ACK] Seq=1 Ack=
16	1.654372	172.16.1.11	172.16.1.1	KRB5	240 TGS-REQ
17	1.667657	172.16.1.11	172.16.1.11	TCP	54 88 → 49753 [ACK] Seq=1 Ack=
18	1.668761	172.16.1.11	172.16.1.12	TCP	54 49752 → 445 [ACK] Seq=338 A
19	1.669648	172.16.1.1	172.16.1.11	TCP	1514 88 → 49753 [ACK] Seq=1 Ack=
20	1.669648	172.16.1.1	172.16.1.11	KRB5	495 TGS-REP
21	1.670245	172.16.1.11	172.16.1.1	TCP	54 49753 → 88 [ACK] Seq=1647 A
22	1.670256	172.16.1.11	172.16.1.1	TCP	54 49753 → 88 [FIN, ACK] Seq=1
23	1.670321	172.16.1.11	172.16.1.12	TCP	1514 49752 → 445 [ACK] Seq=338 A
24	1.670321	172.16.1.11	172.16.1.12	SMB2	376 Session Setup Request
25	1.670633	172.16.1.12	172.16.1.11	TCP	54 445 → 49752 [ACK] Seq=565 A
26	1.670646	172.16.1.1	172.16.1.11	TCP	54 88 → 49753 [ACK] Seq=1602 A
27	1.670655	172.16.1.1	172.16.1.11	TCP	54 88 → 49753 [RST, ACK] Seq=1
28	1.671513	172.16.1.12	172.16.1.11	SMB2	314 Session Setup Response
29	1.672070	172.16.1.11	172.16.1.12	SMB2	156 Tree Connect Request Tree
30	1.683271	172.16.1.12	172.16.1.11	SMB2	138 Tree Connect Response
31	1.683853	172.16.1.11	172.16.1.12	SMB2	178 Ioctl Request FSCTL_QUERY_M
32	1.683872	172.16.1.11	172.16.1.12	SMB2	190 Create Request File: wkssvc

Es ist also nicht wirklich schwer, das Anmeldeverhalten zu analysieren.

Umstellung

Im folgenden Text beschreibe ich die Umstellung in meiner eigenen Infrastruktur.

IST-Situation

In meinem Netzwerk setze ich auf Windows Server 2012R2 und Windows Server 2016. Alle Clients laufen mindestens mit Windows 10 Enterprise V1709. Es existieren einige Nicht-Microsoft-Systeme. Die meisten Services kommen direkt von Microsoft und sind auf dem aktuellen Stand der Technik. Es sollte keine Probleme bei der Umstellung geben.

Dennoch habe ich das Audit aktiviert. Das ist generell auch bei einer Deaktivierung von NTLM empfehlenswert, denn vielleicht kommen morgen neue Services mit NTLM ums Eck. ☹️

Wie zu erwarten kamen einige Treffer zusammen (ein kleiner Auszug):

DC	Datetime	Client	Server	Domain	User
--	----	----	----	----	----
WS-DC3	2018-09-25 02:21:40	WS-IPM	WS-RDS3	ws	service-prtg
WS-DC3	2018-09-25 02:21:40	WS-IPM	WS-RDS3	ws	service-prtg
WS-DC3	2018-09-25 02:33:40	WS-IPM	WS-RDS3	ws	service-prtg
WS-DC3	2018-09-25 02:33:40	WS-IPM	WS-RDS3	ws	service-prtg
WS-DC3	2018-09-25 02:42:40	WS-IPM	WS-RDS3	ws	service-prtg
WS-DC3	2018-09-25 02:42:40	WS-IPM	WS-RDS3	ws	service-prtg
WS-DC3	2018-09-25 13:52:40	WS-IPM	WS-RDS3	ws	service-prtg
WS-DC3	2018-09-25 13:52:40	WS-IPM	WS-RDS3	ws	service-prtg
WS-DC3	2018-09-28 05:02:40	WS-IPM	WS-RDS3	ws	service-prtg
WS-DC3	2018-09-28 05:02:41	WS-IPM	WS-RDS3	ws	service-prtg
WS-DC3	2018-09-28 08:36:40	WS-IPM	WS-RDS3	ws	service-prtg
WS-DC3	2018-09-28 08:36:40	WS-IPM	WS-RDS3	ws	service-prtg
WS-DC2	2018-09-30 14:34:23	WS-CL1	WS-RDS1	ws	stephan
WS-DC2	2018-09-30 14:34:24	WS-CL1	WS-RDS1	ws	stephan
WS-DC2	2018-09-30 14:35:17	WS-CL1	WS-RDS1	ws	stephan
WS-DC2	2018-09-30 14:38:42	WS-CL1	WS-RDS1	ws	stephan
WS-DC2	2018-09-30 14:40:24	WS-CL1	WS-RDS1	ws	stephan
WS-DC2	2018-09-30 14:44:24	WS-CL1	WS-RDS1	ws	stephan
WS-DC2	2018-09-30 14:44:26	WS-CL1	WS-RDS1	ws	stephan
WS-DC2	2018-09-30 14:44:27	WS-CL1	WS-RDS1	ws	stephan
WS-DC2	2018-09-30 14:50:27	WS-CL1	WS-RDS1	ws	stephan
WS-DC2	2018-09-30 14:54:27	WS-CL1	WS-RDS1	ws	stephan
WS-DC2	2018-09-30 14:54:28	WS-CL1	WS-RDS1	ws	stephan
WS-DC2	2018-09-30 14:54:30	WS-CL1	WS-RDS1	ws	stephan
WS-DC2	2018-09-30 15:00:30	WS-CL1	WS-RDS1	ws	stephan
WS-DC2	2018-09-30 15:04:29	WS-CL1	WS-RDS1	ws	stephan
WS-DC2	2018-09-30 15:04:31	WS-CL1	WS-RDS1	ws	stephan
WS-DC2	2018-09-30 15:04:32	WS-CL1	WS-RDS1	ws	stephan
WS-DC3	2018-09-30 20:44:40	WS-IPM	WS-RDS3	ws	service-prtg
WS-DC3	2018-09-30 20:44:40	WS-IPM	WS-RDS3	ws	service-prtg
WS-DC3	2018-10-03 11:47:40	WS-IPM	WS-RDS3	ws	service-prtg
WS-DC3	2018-10-03 11:47:40	WS-IPM	WS-RDS3	ws	service-prtg
WS-DC1	2018-10-03 15:21:17	WS-DPM	WS-FS1	ws	stephan-ad
WS-DC1	2018-10-03 15:21:47	WS-DPM	WS-FS1	ws	stephan-ad
WS-DC1	2018-10-03 17:08:45	\\WS-IPM	WS-NAS1	ws	sysadm
WS-DC1	2018-10-03 17:08:46	\\WS-IPM	WS-NAS1	ws	sysadm
WS-DC1	2018-10-03 17:08:47	\\WS-IPM	WS-NAS1	ws	sysadm
WS-DC1	2018-10-03 17:08:48	\\WS-IPM	WS-NAS1	ws	sysadm
WS-DC1	2018-10-03 17:08:48	\\WS-IPM	WS-NAS1	ws	sysadm
WS-DC1	2018-10-04 17:09:57	\\WS-IPM	WS-NAS1	ws	sysadm
WS-DC1	2018-10-04 17:09:57	\\WS-IPM	WS-NAS1	ws	sysadm
WS-DC1	2018-10-04 17:09:58	\\WS-IPM	WS-NAS1	ws	sysadm

Betroffene Services sind

- mein PRTG-Monitoring, dass sich vom WS-IPM aus auf verschiedene Services zur Überwachung aufschaltet – darunter auch meine NAS WS-NAS1
- verschiedene Zugriffe auf meine RDS-Plattform
- Ein Zugriff meines AdminAccounts vom DPM-Server auf einen der Fileserver

Jeder Treffer im Logfile muss analysiert werden Dabei kann auch eine Zusammenfassung helfen:


```
Invoke-Command -ComputerName (Get-ADDomain).ReplicaDirectoryServers -ScriptBlock {
    Get-WinEvent -Path C:\Windows\System32\Winevt\Logs\Microsoft-Windows-NTLM%4Operational.evtx -ErrorAction SilentlyContinue |
        Select-Object -Property @{ n='DC' ; e={ $env:COMPUTERNAME } },
        @{ n='Datetime' ; e={ (Get-Date -Date $_.TimeCreated -Format u) -replace 'z' } },
        @{ n='Client' ; e={ (($_.Message -split "`n" | select-string 'Arbeitsstationsname') -replace '\\\' -split
':')[1].trim() } },
        @{ n='Server' ; e={ (($_.Message -split "`n" | select-string 'Name des sicheren Kanals') -split ':')[1].trim() } },
        @{ n='Domain' ; e={ (($_.Message -split "`n" | select-string 'Domänenname') -split ':')[1].trim() } },
        @{ n='User' ; e={ (($_.Message -split "`n" | select-string 'Benutzername') -split ':')[1].trim() } }
    } | Group-Object -Property Client,Server,User |
        Sort-Object -Property Count |
            Select-Object -Property count,@{n='Connection';e={
                $Val = $_.name -split ' '
                $Val[0] + " -> " + $Val[1] + " (" + $Val[2] + ")"
            }} |
                Format-Table -Property count,Connection
```

```
Count Connection
-----
1 WS-CL1 -> WS-FS2 (stephan)
1 WS-DC1 -> WS-MX2 (WS-DC1$)
1 WS-CL5 -> WS-RDS1 (stephan-ad)
2 WS-IPM -> WS-DPM (service-prtg)
2 WS-IPM -> WS-RA1 (service-prtg)
2 WS-DPM -> WS-MX1 (gMSA-Backup$)
2 WS-DPM -> WS-MX1 (WS-DPM$)
2 WS-CM -> WS-MX1 (anyone)
2 NULL -> WS-RA1 (stephan)
2 WS-DC1 -> WS-RDS2 (sysadm)
2 WS-IPM -> WS-RA2 (service-prtg)
2 WS-IPM -> WS-HV2 (service-prtg)
3 WS-CL1 -> WS-CM (stephan)
3 NULL -> WS-RA2 (stephan)
3 WS-CL1 -> WS-FS1 (stephan)
3 WS-CL1 -> WS-RDS2 (stephan)
3 WS-RDS2 -> WS-FS1 (stephan-ad)
4 WS-STEUER -> WS-FS2 (stephan)
6 WS-CM -> WS-FS2 (stephan-ad)
6 WS-RDS1 -> WS-FS1 (stephan-ad)
7 WS-STEUER -> WS-FS1 (stephan)
8 WS-IPM -> WS-MX1 (anonymous)
13 WS-CM -> WS-FS1 (stephan-ad)
13 WS-IPM -> WS-MX2 (anonymous)
13 WS-DPM -> WS-FS1 (stephan-ad)
13 WS-CL2 -> WS-RDS1 (Sandro)
16 WS-IPM -> WS-RDS3 (service-prtg)
24 WS-RDS2 -> WS-FS2 (stephan-ad)
25 WS-CL5 -> WS-RDS1 (stephan)
67 WS-CL1 -> WS-RDS1 (stephan)
75 WS-IPM -> WS-NAS1 (sysadm)
165 WS-CL1 -> WS-MX2 (stephan-jb)
196 WS-CL1 -> WS-MX1 (stephan-jb)
224 WS-CL1 -> WS-MX2 (stephan-privat)
244 WS-CL1 -> WS-MX1 (stephan-privat)
374 WS-CL1 -> WS-MX2 (stephan)
557 WS-CL1 -> WS-MX1 (stephan)
597 WS-CL3 -> WS-MX2 (jungbrunnen)
660 WS-CA1 -> WS-RA1 (WS-CA1$)
738 WS-CL3 -> WS-MX1 (jungbrunnen)
```

Nach diversen Tests konnte ich 4 Services ausmachen, die ich genauer kontrollieren wollte:

- meine Exchange Server (WS-MX1 und WS-MX2)
- meine Microsoft VPN-Lösung (WS-RA1 und WS-RA2)
- meine RDS-Farm (WS-RDS1, WS-RDS2)
- meinen PRTG-MonitoringService (WS-IPM)

Alle anderen sollten sich bei der Aushandlung auf Kerberos verständigen können.

Umstellung auf NoNTLM

Die Umstellung war durch eine scharfe GPO und deren Anwendung auf 3 DomainController recht schnell erledigt. Nun ging es an das Troubleshooting.

Und dabei habe ich immer das Logfile beobachtet!

Nachkontrollen und Troubleshooting

Exchange Server (2016)

Ich verwende Outlook im internen Netzwerk und auch außerhalb. Zudem nutze ich bei Bedarf OWA und selbstverständlich ActiveSync auf Smartphones.

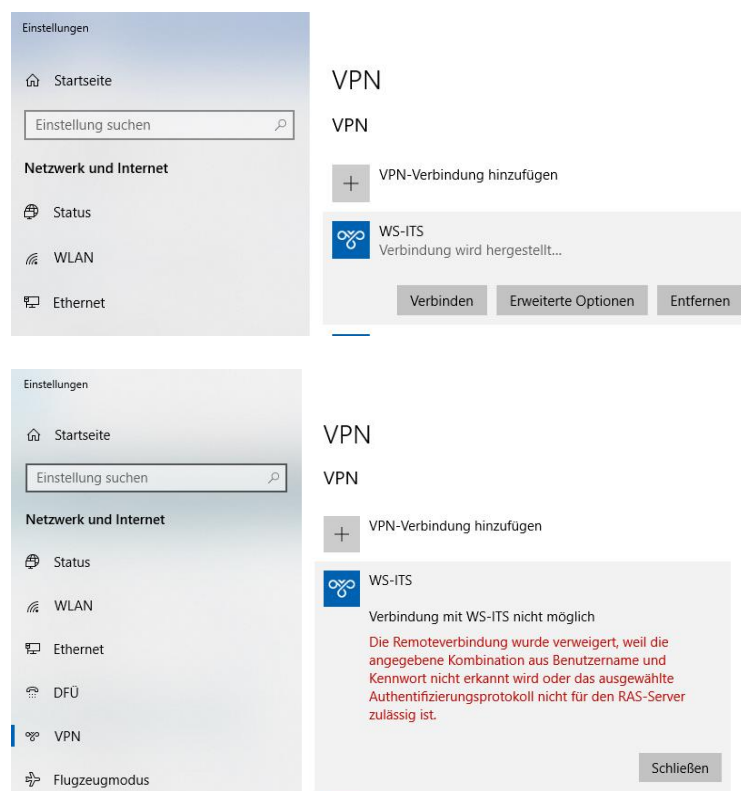
Mein Outlook hatte Intern sofort eine Verbindung. Ebenso klappt der Verbindungsaufbau von extern:

VID	SMTP-Adresse	Anzeigenname	Proxyserver	Servename	Status	Protokoll	Authn	Versc...	RPC-Port	Typ	Anfr/Fehler	Reaktio...	Bearb (0)	Sitzungstyp
36	Stephan.Walther@...	Stephan.Walther@...		https://email.ws-its.de/ma...	hergestellt	HTTP	Nego*	SSL		Exchan...	46/1	192	18	Hintergrund
39	stephan@widwal.de	stephan@widwal.de		https://email.ws-its.de/ma...	hergestellt	HTTP	Nego*	SSL		Exchan...	32/1	167	8	Hintergrund
40	stephan@jungbrun...	stephan@jungbrun...		https://email.ws-its.de/ma...	hergestellt	HTTP	Nego*	SSL		Exchan...	33/1	183	7	Hintergrund
43	Stephan.Walther@...	Onlinearchiv - Step...		https://email.ws-its.de/ma...	hergestellt	HTTP	Nego*	SSL		Exchan...	25/1	244	13	Hintergrund
46	stephan@jungbrun...	stephan@jungbrun...		https://email.ws-its.de/ma...	hergestellt	HTTP	Nego*	SSL		Exchan...	25/1	155	5	Cache
49	stephan@widwal.de	stephan@widwal.de		https://email.ws-its.de/ma...	hergestellt	HTTP	Nego*	SSL		Exchan...	22/1	186	7	Cache
53	Stephan.Walther@...	Stephan.Walther@...		https://email.ws-its.de/ma...	hergestellt	HTTP	Nego*	SSL		Exchan...	31/1	164	18	Cache
55	stephan@jungbrun...	Onlinearchiv - step...		https://email.ws-its.de/ma...	hergestellt	HTTP	Nego*	SSL		Exchan...	31/1	257	5	Spooler
59	stephan@widwal.de	Öffentliche Ordner ...		https://email.ws-its.de/ma...	hergestellt	HTTP	Nego*	SSL		Öffentli...	4/1		30	Cache
61	stephan@jungbrun...	Öffentliche Ordner ...		https://email.ws-its.de/ma...	hergestellt	HTTP	Nego*	SSL		Öffentli...	4/1		26	Cache
63	Stephan.Walther@...	Öffentliche Ordner ...		https://email.ws-its.de/ma...	hergestellt	HTTP	Nego*	SSL		Öffentli...	21/1	131	12	Cache
81	Stephan.Walther@...	Nicole@jungbrun...		https://email.ws-its.de/ma...	hergestellt	HTTP	Nego*	SSL		Exchan...	2/1			Cache

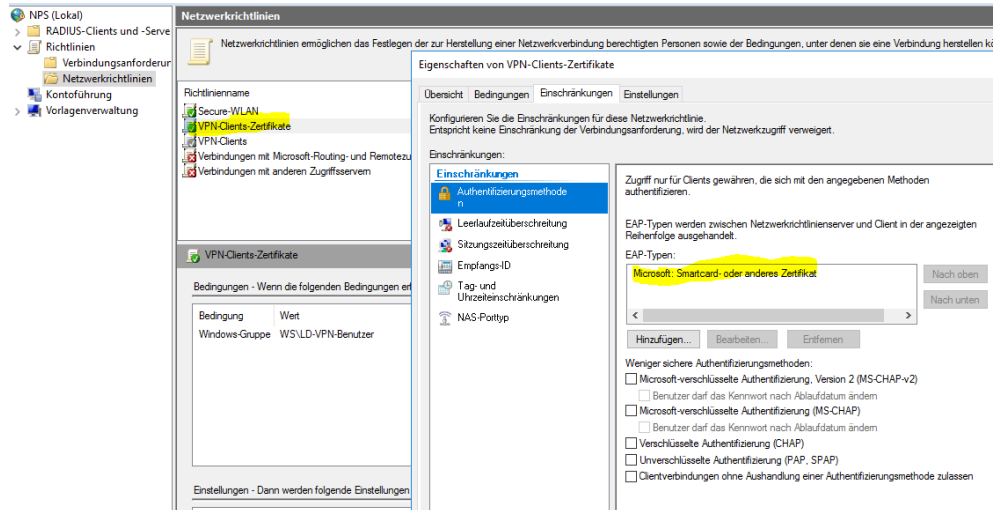
Auch ActiveSync scheint keine Probleme zu haben. Ebenso funktioniert OWA über FBA. ☺

VPN-Einwahl mit Windows RemoteAccess

Leider sieht das bei meiner VPN-Lösung nicht ganz so fein aus. Aktuell habe ich einen RemoteAccessCluster aus 2 Servern bereitgestellt. Beide bieten nach außen VPN mit SSTP über eine Zertifikatsabsicherung auf TCP Port 443 an. Für die Authentifizierung verwendete ich bisher Benutzername und Kennwort von den ActiveDirectory-Benutzern. Die RemoteAccessServer leiteten diese Infos an ihren NPS (NetworkPolicyServer) weiter. Und dieser prüfte gegen die DomainController - mit NTLM... ☹



OK, die Absicherung war eh nicht mehr zeitgemäß. Ich wollte generell auf Smartcard-Anmeldung umstellen. Dank vSmartcard mit Windows 10 und meinem TPM ging das recht einfach. Nun musste ich nur noch die Netzwerkrichtlinien auf meinen NPS anpassen:



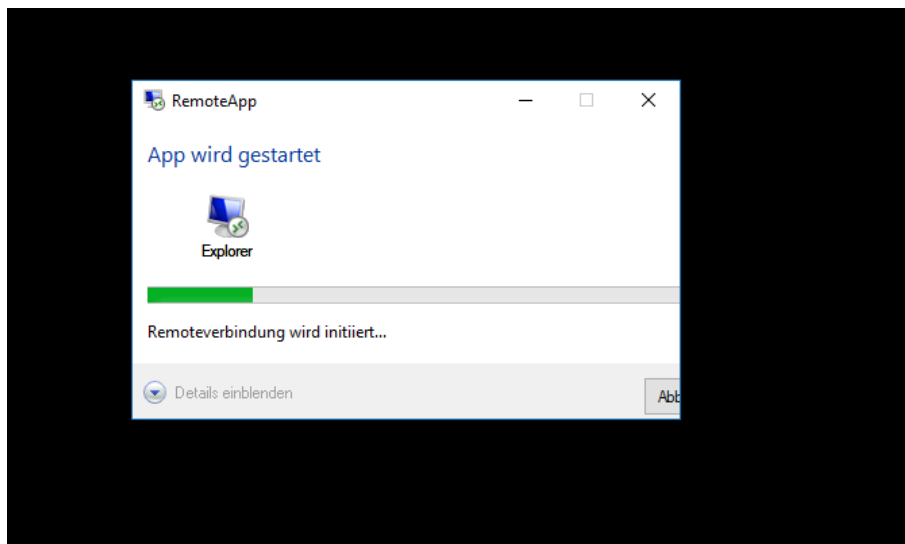
Und schon komm ich wieder per VPN in mein Netzwerk! So haben sich das wohl auch die Entwickler bei Microsoft vorgestellt:

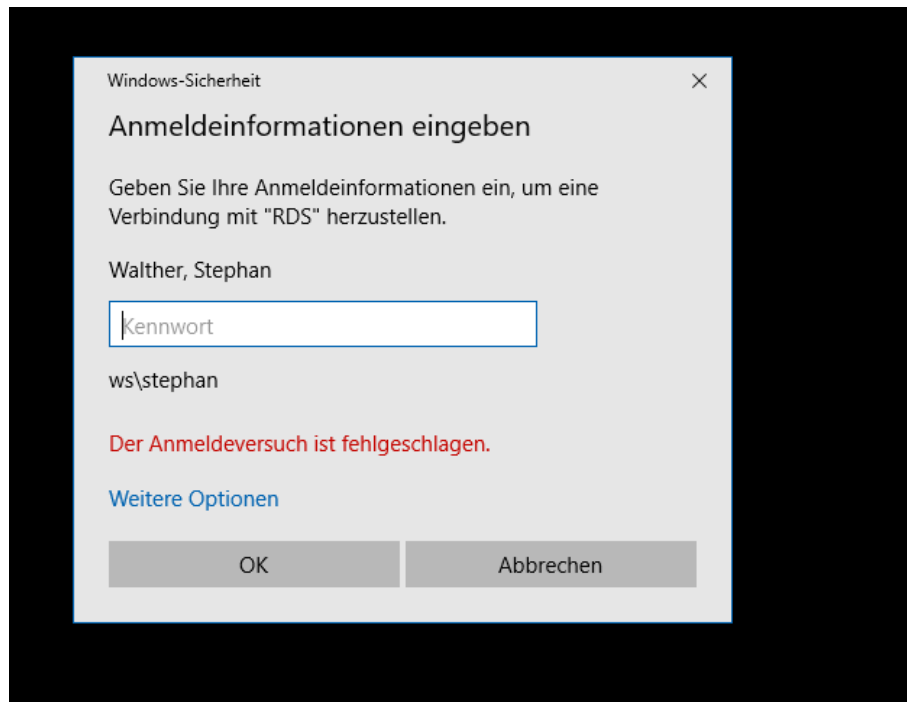
- NTLM wird deaktiviert
- unsichere Zugänge werden entdeckt
- unsichere Zugänge werden gegen sichere Zugänge ersetzt bzw. die Konfigurationen werden angepasst

RemoteDesktop in einer RDS-Farm

Bei meinem RemoteDesktopService (RDS) gab es eine ähnliche Überraschung. Im internen Netzwerk funktionierte alles wie gewohnt – da Kerberos zur Verfügung steht.

Komme ich aber von außen auf meine RDS-Infrastruktur, dann muss ich das RDS-Gateway verwenden. Dieses tunnelt das RDP-Protokoll in einem https-Datenstrom. Damit mich das Gateway erkennt muss ich mich an ihm anmelden. Und das Gateway fragt im Hintergrund beim DomainController via NTLM an... ☹



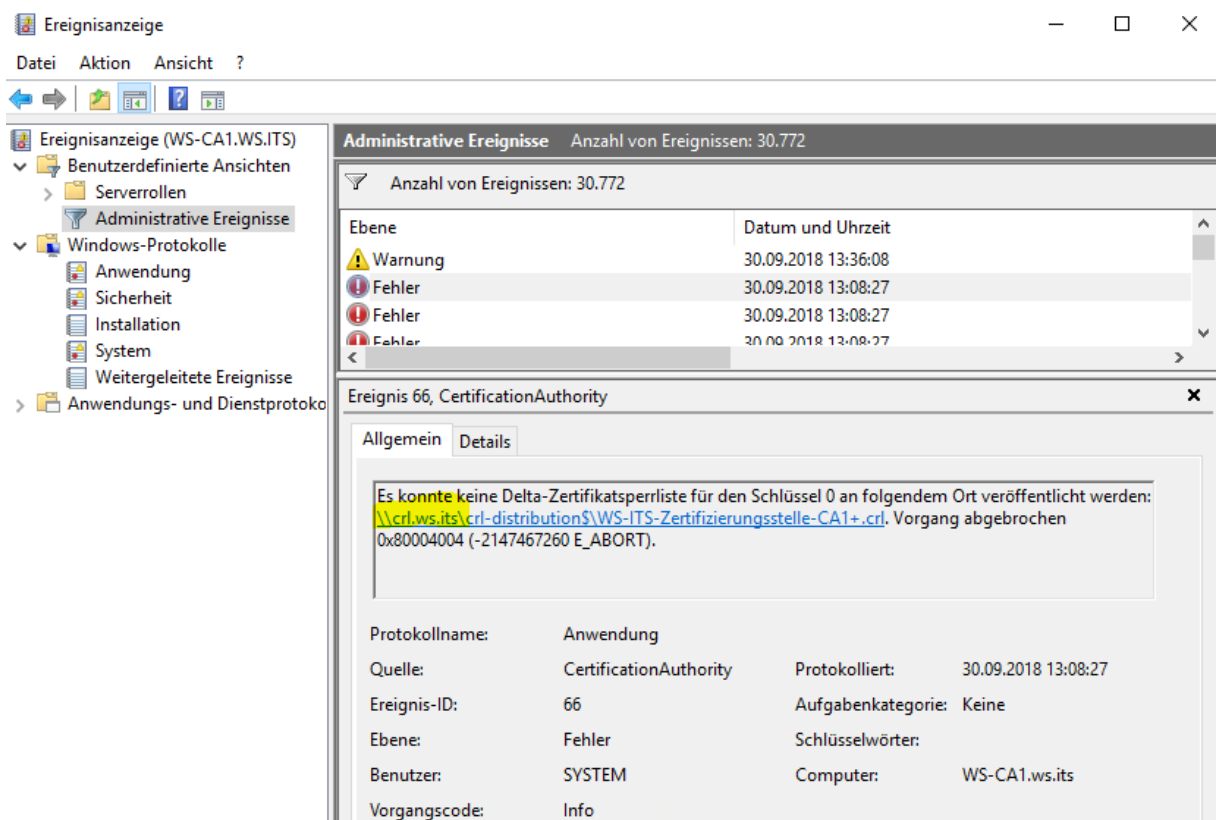


Der RD-Gateway verwendet ebenfalls einen lokalen NPS. In diesem gibt es eine Standard-Zugriffsrichtlinie. Diese verwendet aktuell ein Passwort (und damit NTLM). Alternativ kann ich auch hier auf eine Smartcard-Anmeldung umstellen. Da ich meine RDS-Anwendungen aber auch außerhalb meines eigenen Rechners verwenden möchte kann ich nicht auf Smartcard umstellen. Hier werde ich noch einmal gesondert nach einer Lösung suchen. Bis dahin verwende ich diese eine Maschine in der Ausnahmeliste – und lasse somit NTLM für einen Server zu.

PKI-Sperrlistenveröffentlichung

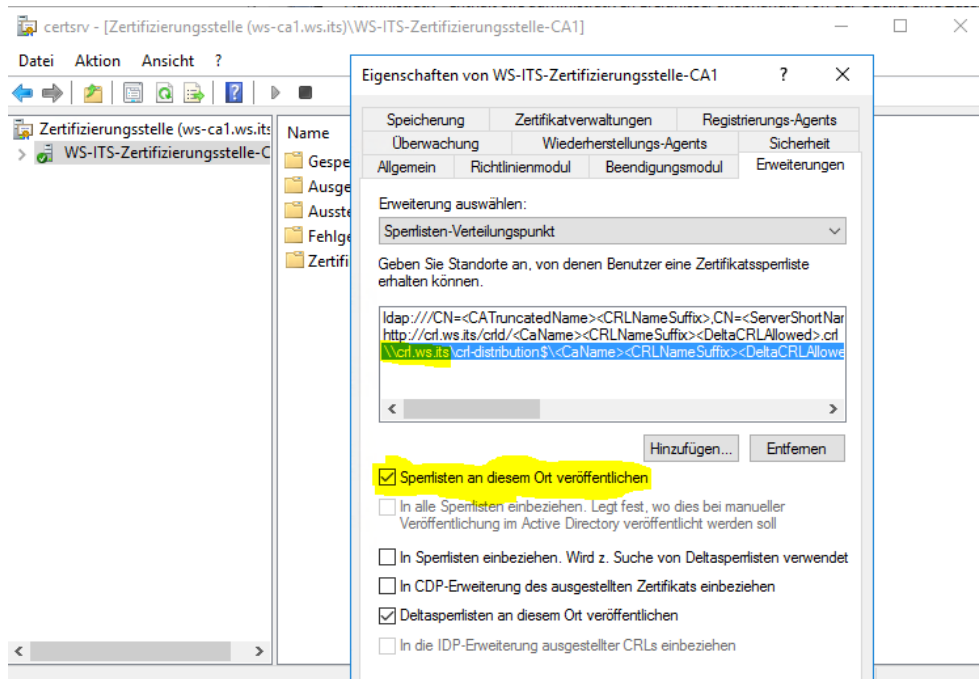
Mein Monitoring meldete mir, dass die Sperrlisten meiner Windows PKI nicht mehr aktualisiert werden.

Ein Blick ins Eventlog des Servers zeigte das Problem:



Was war passiert?

- In meiner CDP-Erweiterung hatte ich eine http-Seite für den Download der Sperrliste hinterlegt:



- Diese Website läuft auf einem anderen Server. Den Namen der Seite hatte ich mit **http://crl.ws.its/...** angegeben.
- Der Webserver ist mein WS-RA1.
- Über DNS hatte ich einen HostA-Record für **crl.ws.its** erstellt und auf die IP-Adresse von WS-RA1 zeigen lassen.
- Mein PKI-Server muss die Sperrlisten über SMB auf WS-RA1 übertragen.
- Dafür hatte ich die Freigabe **crl-distribution\$** auf WS-RA1 erstellt.
- Und meiner PKI hatte ich den Pfad **\\crl.ws.its\crl-distribution\$...** für die Sperrlistenveröffentlichung hinterlegt.

Bisher konnte das System den Namen **crl.ws.its** in die IP-Adresse des Servers WS-RA1 übersetzen und dann den Schreibzugriff auf die Freigabe anfordern. Da der Name aber nicht mit dem echten Namen übereinstimmte konnte meine PKI kein Kerberos verwenden (**crl.ws.its** hatte ich nur im DNS verwendet!). Also verwendete die PKI einfach NTLM. Und mit der Deaktivierung von NTLM war damit Schluss!

Hier war es recht einfach, den Anmeldeprozess auf Kerberos umzustellen, da es sich „nur“ um eine Fehlkonfiguration handelte. Ich nahm den falschen Eintrag aus den Sperrlistenverteilungspunkten raus und ersetzte ihn durch den FQDN des Zielservers:

