



# HACKTHEBOX

## Noxious - Writeup

---



Prepared by: Cyberjunkie

Machine Author(s): Cyberjunkie

Difficulty: **Very Easy**

## Scenario

---

The IDS device alerted about a possible ROGUE device in internal Active directory network. The Intrusion detection device also showed signs of LLMNR Traffic which is not something we see everyday, so it is suspected that LLMNR Poisoning attack occurred. The LLMNR Traffic was directed towards Forela-WKstn002 which has the IP Address 172.17.79.136. Limited Packet Capture of surrounding time is provided to you, our Network Forensics expert. Since this occurred in Active Directory VLAN, it's suggested that perform network threat hunting while keeping the Active directory attacks vector in mind such as LLMNR Poisoning.

## Artefacts provided

1- Toxic.zip (zip file), sha1 : F474CECDF5A984EC5208D7B3E4AE705FD25C2D99

# Initial Analysis :

Let's first start by discussing the given artifact. We are given a zip file which upon extraction gives us a PCAP file. The incident concerns the Credential stealing technique using responder tool in an active directory network. You may find the full details of the attack and detection techniques in our blog.

## What is PCAP? :

PCAP files are data files created using a program. These files contain packet data of a network and are used to analyze the network characteristics. They also contribute to controlling the network traffic and determining network status. Using PCAP files, teams can attend to detect network problems and resolve data communications using various programs. Security teams can use a network packet capture tool to identify, analyze, inspect, and monitor network traffic. Unusual traffic spikes can be due to a faulty application or a security breach. The packet capture tool allows IT teams to identify the root cause of the issues by tracking network packets.

Let's start by opening the PCAP file in Wireshark. Going over Statistics-> Endpoints we can see top most IP Addresses.

Ethernet · 19 IPv4 · 231 IPv6 · 11 TCP · 545 UDP · 711										
Address	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes	Country	City	AS Number	AS
172.17.79.136	17,483	9.436 MiB	7,296	1.212 MiB	10,187	8.224 MiB				
172.17.79.129	9,091	119.796 MiB	4,072	323.272 KiB	5,019	119.481 MiB				
92.122.95.210	8,090	119.408 MiB	4,510	119.198 MiB	3,580	214.406 KiB				
2.16.135.186	2,394	1.911 MiB	1,404	1.846 MiB	990	65.973 KiB				
104.18.21.126	2,232	1.570 MiB	1,511	1.446 MiB	721	126.871 KiB				
20.191.45.158	1,892	1.821 MiB	1,293	1.771 MiB	599	50.964 KiB				
172.17.79.4	1,407	223.817 KiB	679	95.312 KiB	728	128.506 KiB				
2.16.135.216	1,101	759.409 KiB	648	719.615 KiB	453	39.794 KiB				
172.17.79.130	1,078	1,002.980 KiB	373	21.736 KiB	705	981.244 KiB				
151.101.2.49	1,029	991.275 KiB	679	972.008 KiB	350	19.268 KiB				
2.16.135.218	737	452.538 KiB	504	379.985 KiB	233	72.553 KiB				
2.16.135.194	679	430.578 KiB	415	271.113 KiB	264	159.465 KiB				
2.16.135.224	665	571.273 KiB	397	553.249 KiB	268	18.024 KiB				
172.17.79.2	579	88.025 KiB	246	56.773 KiB	333	31.252 KiB				
172.17.79.1	565	43.160 KiB	564	42.986 KiB	1	178 bytes				
224.0.0.22	478	28.016 KiB	0	0 bytes	478	28.016 KiB				
216.58.204.142	396	186.053 KiB	204	110.263 KiB	192	75.790 KiB				
216.58.209.42	239	189.211 KiB	164	177.021 KiB	75	12.190 KiB				
216.58.205.46	199	118.404 KiB	111	86.612 KiB	88	31.792 KiB				
216.58.204.132	160	85.044 KiB	94	73.190 KiB	66	11.854 KiB				
216.58.205.34	150	71.224 KiB	83	50.743 KiB	67	20.480 KiB				
23.13.53.67	149	41.988 KiB	77	12.146 KiB	72	29.842 KiB				
172.17.79.135	140	14.188 KiB	128	12.009 KiB	12	2.180 KiB				
230.255.255.250	140	30.027 KiB	0	0 bytes	140	30.027 KiB				

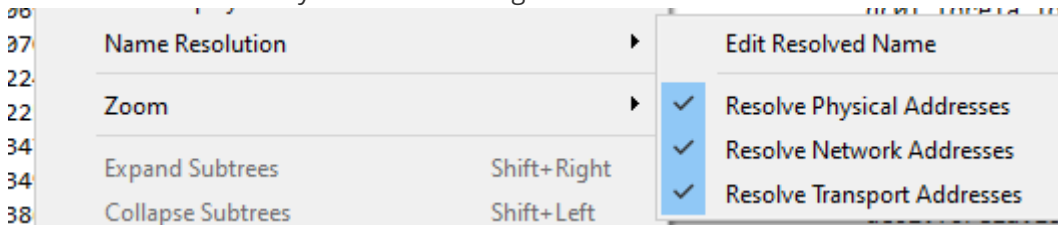
The highlighted IPs are the internal IP Address that we are interested in. Let's start our analysis based on the questions being asked.

## Questions Answers

Q1 Its suspected by the security team that there was a rogue device in forela's internal network running responder tool to perform LLMNR Poisoning attack. Please find the malicious IP Address of the machine.

Hint: First identify the IP Address of the domain controller. Any LLMNR requests originating from some other machine to another machine is a sign of a rogue machine pretending to be Domain controller to capture hashes. Filter for udp port 5355 in wireshark using `udp.port == 5355` . This will display all the LLMNR traffic. We see only 1 ip address responding to our victim machine to their query . This IP Address does not belong to the domain controller.

To perform the attack, the adversary needs to spoof as a Domain controller acting as a Man in the middle in the network. So our first step should be to find the True domain controller's IP address so we can ignore its traffic for this part of the analysis and identify the spoofed (Attacker's) IP Address. The easiest way to find this is to go to view->Name resolution and enable all 3 options.



Now filter for "DNS". We do this because in domain environments domain controller acts as a DNS server hence all network traffic goes through here.

A screenshot of the Wireshark packet list. The filter 'dns' is applied. The table shows several DNS packets. The first packet (No. 7421) is a 'Standard query response' from 'dc01.forela.local' to 'Forela-Wkstn002.forela.local'. The second packet (No. 7422) is a 'Standard query response' from 'dc01.forela.local' to 'Forela-Wkstn002.forela.local'. The third packet (No. 7765) is a 'Standard query' from 'Forela-Wkstn002.forela.local' to 'dc01.forela.local'. The fourth packet (No. 7766) is a 'Standard query' from 'dc01.forela.local' to '172.17.79.2'. The fifth packet (No. 7767) is a 'Standard query' from 'Forela-Wkstn002.forela.local' to 'dc01.forela.local'. The sixth packet (No. 7768) is a 'Standard query' from 'dc01.forela.local' to '172.17.79.2'. The seventh packet (No. 7971) is a 'Standard query response' from 'dc01.forela.local' to '172.17.79.2'. The eighth packet (No. 7972) is a 'Standard query response' from 'dc01.forela.local' to '172.17.79.2'. The ninth packet (No. 7973) is a 'Standard query response' from 'Forela-Wkstn002.forela.local' to 'Forela-Wkstn002.forela.local'.

We see lots of traffic and we all see FQDN for DC01 which is our domain controller. We can get to know the IP Address in the details tab. Now you can go ahead and untick the view options we set if you want the default view or leave it as it is whatever you prefer.

```
> Frame 7007: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
> Ethernet II, Src: dc01.forela.local (00:0c:29:56:44:f9), Dst: VMware_f7:80:51 (00:50:56:80:51:f7)
> Internet Protocol Version 4, Src: dc01.forela.local (172.17.79.4), Dst: 172.17.79.2
  0100 .... = Version: 4
  .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 72
    Identification: 0x6751 (26449)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
    Time to Live: 128
    Protocol: UDP (17)
    Header Checksum: 0xdd2a [validation disabled]
    [Header checksum status: Unverified]
    Source Address: dc01.forela.local (172.17.79.4)
    Destination Address: 172.17.79.2 (172.17.79.2)
  > User Datagram Protocol, Src Port: 64486 (64486), Dst Port: domain (53)
```

Doing some external research and from the given hint we learn that LLMNR uses UDP Port 5355. Since we are dealing with a potential LLMNR Poisoning attack we should certainly take a look at this stream of traffic. Add a Wireshark filter to filter for this port. Alternatively, we can also filter by protocol name "llmnr".

udp.port == 5355									
No.	Time	Source	Destination	Protocol	Length	Info			
9262	2024-06-24 11:18:30.895613	fe80::7994:1860:711:c243	ff02::1:3	LLMNR	85	Standard query 0xe708 A DCC01			
9263	2024-06-24 11:18:30.895766	172.17.79.136	224.0.0.252	LLMNR	65	Standard query 0xe708 A DCC01			
9264	2024-06-24 11:18:30.896012	fe80::7994:1860:711:c243	ff02::1:3	LLMNR	85	Standard query 0x2c01 AAAA DCC01			
9265	2024-06-24 11:18:30.896077	172.17.79.136	224.0.0.252	LLMNR	65	Standard query 0x2c01 AAAA DCC01			
9268	2024-06-24 11:18:30.900217	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	LLMNR	106	Standard query response 0xe708 A DCC01 /			
9269	2024-06-24 11:18:30.902155	172.17.79.135	172.17.79.136	LLMNR	86	Standard query response 0xe708 A DCC01 /			
9274	2024-06-24 11:18:30.903430	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	LLMNR	118	Standard query response 0x2c01 AAAA DCC01 /			
9277	2024-06-24 11:18:30.907809	172.17.79.135	172.17.79.136	LLMNR	98	Standard query response 0x2c01 AAAA DCC01 /			
9301	2024-06-24 11:18:30.932480	fe80::7994:1860:711:c243	ff02::1:3	LLMNR	85	Standard query 0x3ca4 A DCC01			
9302	2024-06-24 11:18:30.932481	172.17.79.136	224.0.0.252	LLMNR	65	Standard query 0x3ca4 A DCC01			
9303	2024-06-24 11:18:30.932919	fe80::7994:1860:711:c243	ff02::1:3	LLMNR	85	Standard query 0x2dd6 AAAA DCC01			
9304	2024-06-24 11:18:30.933150	172.17.79.136	224.0.0.252	LLMNR	65	Standard query 0x2dd6 AAAA DCC01			
9305	2024-06-24 11:18:30.935982	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	LLMNR	106	Standard query response 0x3ca4 A DCC01 /			
9307	2024-06-24 11:18:30.938257	172.17.79.135	172.17.79.136	LLMNR	86	Standard query response 0x3ca4 A DCC01 /			
9309	2024-06-24 11:18:30.941272	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	LLMNR	118	Standard query response 0x2dd6 AAAA DCC01 /			

Now that we know the real domain controller IP Address, it should be the one responding to llmnr queries in a legitimate case. But it doesn't. Instead, we see another IP Address "172.17.79.135" responding to queries made by 172.17.79.136 which according to the given scenario is the victim IP Address.

We also see that the query performed by the victim machine is DCC01 which seems like a typo of DC01. This suggests that the victim performed a typo when navigating to a network share(the typo share does not exist and Windows falls back to llmnr protocol to resolve that) which made the LLMNR Poison attack possible and the attacker captured the credentials. Ans 172.17.79.135

Q2 What is the hostname of the rogue machine?

Hint: Add a filter for the ip address and dhcp. In wireshark it will be ip.addr == 172.17.79.135 && dhcp. Then look for hostname value in one of the packets. Reason why we use this is because the device used dhcp to get a ip address assigned to itself and map it to its hostname.

Now that we have the IP address, we can look for DHCP traffic related to this IP which will reveal its hostname.

ip.addr == 172.17.79.135 && dhcp									
Packet bytes	Narrow & Wide	Case sensitive	String	dc01					
Time	Source	Destination	Protocol	Length	Info				
1666	2024-06-24 11:17:33.972330	172.17.79.135	DHCP	311	DHCP Discover - Transaction ID 0xa7ea9ba0				
12714	2024-06-24 11:27:57.928506	172.17.79.135	DHCP	324	DHCP Request - Transaction ID 0x481fcd6				
12715	2024-06-24 11:27:57.928506	172.17.79.254	DHCP	342	DHCP ACK - Transaction ID 0x481fcd6				

```

Parameter Request List Item: (17) Root Path
  Option: (57) Maximum DHCP Message Size
    Length: 2
    Maximum DHCP Message Size: 65535
  Option: (12) Host Name
    Length: 4
    Host Name: kali
  Option: (255) End
    Option End: 255

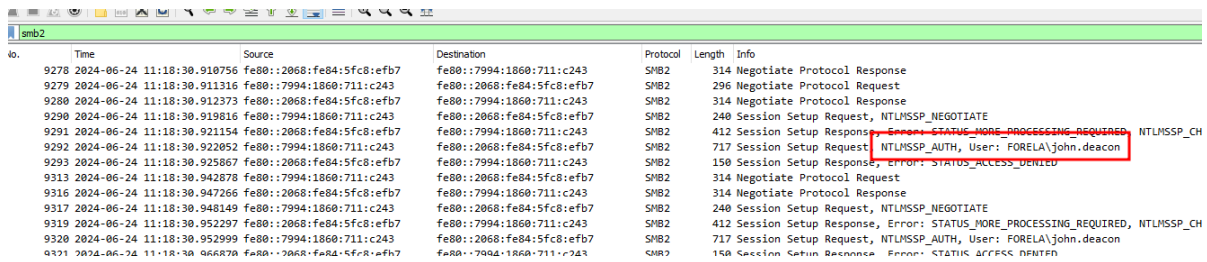
```

Ans kali

Q3 Now we need to confirm whether the attacker captured the hash of the user and is it really crackable!! What is the username whose hash was captured?

Hint: First filter for smb traffic. In wireshark it is "smb2". Here we see some ntlmssp negotiate and auth which means the hash indeed got captured. To further narrow down this add the filter "ntlmssp". We can see the user name in NTLMSSP\_AUTH packets

Add a filter for SMB Traffic.



No.	Time	Source	Destination	Protocol	Length	Info
9278	2024-06-24 11:18:30.910756	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	SMB2	314	Negotiate Protocol Response
9279	2024-06-24 11:18:30.911316	fe80::7994:1860:711:c243	fe80::2068:fe84:5fc8:efb7	SMB2	296	Negotiate Protocol Request
9280	2024-06-24 11:18:30.912373	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	SMB2	314	Negotiate Protocol Response
9290	2024-06-24 11:18:30.919816	fe80::7994:1860:711:c243	fe80::2068:fe84:5fc8:efb7	SMB2	240	Session Setup Request, NTLMSSP_NEGOTIATE
9291	2024-06-24 11:18:30.921154	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	SMB2	412	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
9292	2024-06-24 11:18:30.922052	fe80::7994:1860:711:c243	fe80::2068:fe84:5fc8:efb7	SMB2	717	Session Setup Request, NTLMSSP_AUTH, User: FORELA\john.deacon
9293	2024-06-24 11:18:30.925067	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	SMB2	150	Session Setup Response, Error: STATUS_ACCESS_DENIED
9313	2024-06-24 11:18:30.942078	fe80::7994:1860:711:c243	fe80::2068:fe84:5fc8:efb7	SMB2	314	Negotiate Protocol Request
9316	2024-06-24 11:18:30.947266	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	SMB2	314	Negotiate Protocol Response
9317	2024-06-24 11:18:30.948149	fe80::7994:1860:711:c243	fe80::2068:fe84:5fc8:efb7	SMB2	240	Session Setup Request, NTLMSSP_NEGOTIATE
9319	2024-06-24 11:18:30.952297	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	SMB2	412	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
9320	2024-06-24 11:18:30.952999	fe80::7994:1860:711:c243	fe80::2068:fe84:5fc8:efb7	SMB2	717	Session Setup Request, NTLMSSP_AUTH, User: FORELA\john.deacon
9321	2024-06-24 11:18:30.956870	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	SMB2	150	Session Setup Response, Error: STATUS_ACCESS_DENIED

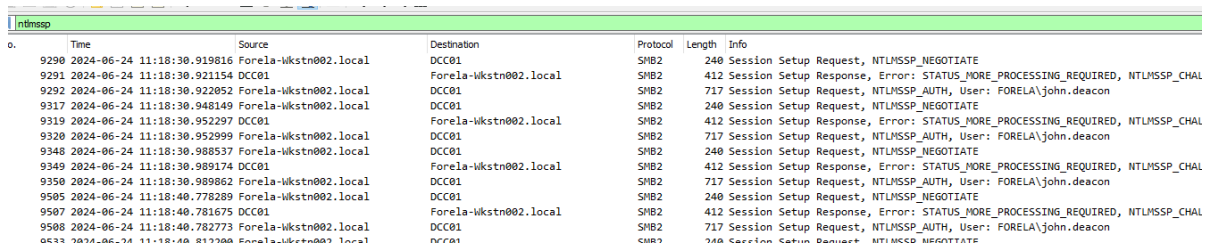
We see the user named john.deacon used during NTLM authentication negotiations. If we see the timestamps, we see that they also line up according to the timeline, as the NTLM process starts just after the typo DNS request we saw in the llmnr traffic.

Ans john.deacon

Q4 In ntlm traffic we can see that the victim credentials were relayed multiple time to the attacker's machine. When was the hashes captured the First time?

Hint : First we need to modify the wireshark view option. Go to View > Time display format > UTC DATE . Now the time coloumn will show the time in utc format. Filter for "ntlmssp" and look at the the time of the first 3 packets (starting from NTLMSSP\_NEGOTIATE and ending in NTLMSSP\_AUTH packet).

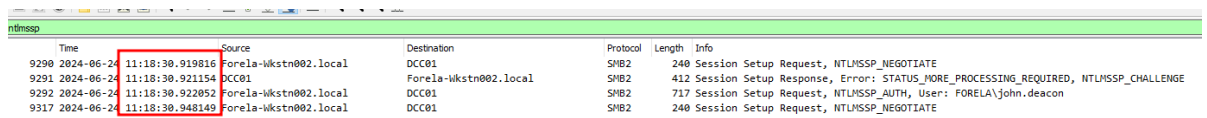
Let's add a filter for NTLMSSP which will only display the NTLM negotiations packets. We see multiple NTLM negotiations in a small amount of time and they all are directed towards a single host from our victim host.



No.	Time	Source	Destination	Protocol	Length	Info
9290	2024-06-24 11:18:30.919816	Forela-wkstrn002.local	DCC01	SMB2	240	Session Setup Request, NTLMSSP_NEGOTIATE
9291	2024-06-24 11:18:30.921154	DCC01	Forela-wkstrn002.local	SMB2	412	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
9292	2024-06-24 11:18:30.922052	Forela-wkstrn002.local	DCC01	SMB2	717	Session Setup Request, NTLMSSP_AUTH, User: FORELA\john.deacon
9317	2024-06-24 11:18:30.948149	Forela-wkstrn002.local	DCC01	SMB2	240	Session Setup Request, NTLMSSP_NEGOTIATE
9319	2024-06-24 11:18:30.952297	DCC01	Forela-wkstrn002.local	SMB2	412	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
9320	2024-06-24 11:18:30.952999	Forela-wkstrn002.local	DCC01	SMB2	717	Session Setup Request, NTLMSSP_AUTH, User: FORELA\john.deacon
9348	2024-06-24 11:18:30.988537	Forela-wkstrn002.local	DCC01	SMB2	240	Session Setup Request, NTLMSSP_NEGOTIATE
9349	2024-06-24 11:18:30.989174	DCC01	Forela-wkstrn002.local	SMB2	412	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
9350	2024-06-24 11:18:30.989862	Forela-wkstrn002.local	DCC01	SMB2	717	Session Setup Request, NTLMSSP_AUTH, User: FORELA\john.deacon
9505	2024-06-24 11:18:40.778289	Forela-wkstrn002.local	DCC01	SMB2	240	Session Setup Request, NTLMSSP_NEGOTIATE
9507	2024-06-24 11:18:40.781675	DCC01	Forela-wkstrn002.local	SMB2	412	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
9508	2024-06-24 11:18:40.782773	Forela-wkstrn002.local	DCC01	SMB2	717	Session Setup Request, NTLMSSP_AUTH, User: FORELA\john.deacon
9533	2024-06-24 11:18:40.817300	Forela-wkstrn002.local	DCC01	SMB2	240	Session Setup Request, NTLMSSP_NEGOTIATE

It's important to note that Wireshark is showing the supposed attacker destination hostname here as "DCC01" thinking it's the same hostname that which victim accidentally requested and the attacker's machine is masquerading as.

Let's see the time stamp for the first set of negotiations.



No.	Time	Source	Destination	Protocol	Length	Info
9290	2024-06-24 11:18:30.919816	Forela-wkstrn002.local	DCC01	SMB2	240	Session Setup Request, NTLMSSP_NEGOTIATE
9291	2024-06-24 11:18:30.921154	DCC01	Forela-wkstrn002.local	SMB2	412	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
9292	2024-06-24 11:18:30.922052	Forela-wkstrn002.local	DCC01	SMB2	717	Session Setup Request, NTLMSSP_AUTH, User: FORELA\john.deacon
9317	2024-06-24 11:18:30.948149	Forela-wkstrn002.local	DCC01	SMB2	240	Session Setup Request, NTLMSSP_NEGOTIATE

If we go back and notice the time when llmnr responses were sent by the rogue machine and when hashes were sent (and probably captured by the attackers) are the same so the timeline makes sense and confirms that something fishy is going on. Ans 2024-06-24 11:18:30

Q5 What was the typo made by the victim when navigating to the file share which caused his credentials being leaked?

Hint : When looking at LLMNR traffic we saw that attacker's machine responded to a query "DCC01" which means that the victim typed DCC01 instead of DC01 which cause the dns to fail and the machine fall back to LLMNR protocol to resolve the query and that's where attacker's rogue machine responded to the query pretending to be a domain controller. We can further confirm this typo in ntlmssp packets where we see that the netname value is DCC01.

Source	Destination	Protocol	Length	Info
.919 fe80::7994:1860:711:c243	ff02::1:3	LLMNR	85	Standard query 0x2dd6 AAAA DCC01
150 172.17.79.136	224.0.0.252	LLMNR	65	Standard query 0x2dd6 AAAA DCC01
.982 fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	LLMNR	106	Standard query response 0x3ca4 A DCC01 A 172.17.79.135
1257 172.17.79.135	172.17.79.136	LLMNR	86	Standard query response 0x3ca4 A DCC01 A 172.17.79.135
.702 fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	LLMNR	118	Standard query response 0x2dd6 AAAA DCC01 AAAA fe80::2068:fe84:5fc8:efb7
1700 172.17.79.135	172.17.79.136	LLMNR	98	Standard query response 0x2dd6 AAAA DCC01 AAAA fe80::2068:fe84:5fc8:efb7
581 fe80::7994:1860:711:c243	ff02::1:3	LLMNR	85	Standard query 0x9970 A DCC01
1734 172.17.79.136	224.0.0.252	LLMNR	65	Standard query 0x9970 A DCC01
.939 fe80::7994:1860:711:c243	ff02::1:3	LLMNR	85	Standard query 0xc75b AAAA DCC01
898 172.17.79.136	224.0.0.252	LLMNR	65	Standard query 0xc75b AAAA DCC01
.065 fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	LLMNR	106	Standard query response 0x9970 A DCC01 A 172.17.79.135
602 172.17.79.135	172.17.79.136	LLMNR	86	Standard query response 0x9970 A DCC01 A 172.17.79.135
.773 fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	LLMNR	118	Standard query response 0x775b AAAA DCC01 AAAA fe80::2068:fe84:5fc8:efb7

Ans DCC01

Hint: In details of the NTLMSSP\_CHALLENGE PACKET (Packet # 9291 in our case) expand SMB2 (Server Message Block Protocol Version 2) -> Session Setup Response (0x1) -> Security Blob -> GSS-API Generic -> Simple Protected Negotiation -> negTokenTarg -> NTLM Secure Service Provider -> NTLM Server Challenge.

Time	Source	Destination	Protocol	Length	Info
9290 2024-06-24 11:18:30.919816	fe80::7994:1860:711:c243	fe80::2068:fe84:5fc8:efb7	SM2	246	Session Setup Request, NTLMSSP_NEGOTIATE
9291 2024-06-24 11:18:30.921154	fe80::2068:fe84:5fc8:efb7	fe80::7994:1860:711:c243	SM2	417	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
9292 2024-06-24 11:18:30.922052	fe80::7994:1860:711:c243	fe80::2068:fe84:5fc8:efb7	SM2	712	Session Setup Request, NTLMSSP_AUTH, User: FORELA\john.deacon

```
netbios session service
SMB2 (Server Message Block Protocol version 2)
> SMB2 Header
  > Session Setup Response (0x01)
    [Preamble Hash: 00624b1bfc6563c355447089f853d38ebb73f8f4a33369aa6978cff0a8bbab4d831b92a...]
    > StructureSize: 0x0009
    > Session Flags: 0x0000
      Blob Offset: 0x00000048
      Blob Length: 262
    > Security Blob: a18201023081ffa0030a0101a10c060a2b06010401823702020aa281e90481e64e544cd...
      > 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 (NTLMSSP - Microsoft NTLM Security Support Provider)
            responseToken: 4e544cd5353500002000000000000038000000158289e2601019d191f054f100000000...
          > NTLM Secure Service Provider
            NTLMSSP identifier: NTLMSSP
            NTLM Message Type: NTLMSSP_CHALLENGE (0x00000002)
            > Target Name: NBFY
            > Negotiate Flags: 0xe2096215, Negotiate 56, Negotiate Key Exchange, Negotiate 128, Negotiate Version, Negotiate Target Info, Negotiate Extended Security, Targe...
            > NTLM Server Challenge: 601019d191f054f1
              Reserved: 0000000000000000
            > Target Info
            > Version 6.3 (Build 9600); NTLM Current Revision 15
```

Q7 Now doing something similar find the NTPProofStr value.

Hint: In details of the NTLMSSP\_AUTH Packet(Packet # 9292) packet details expand SMB2 (Server Message Block Protocol  
Version 2) -> Session Setup Response (0x1) -> Security Blob -> GSS-API Generic -> Simple Protected Negotiation -> negTokenTarg -> NTLM Secure Service Provider -> -> NTLM Response ->



NTLMv2 Response ->

NTPProofStr.

We repeat the process in the third packet of the first set and expand the fields as detailed in the hint.

```
-----
Blob Length: 551
▼ Security Blob: a18202233082021fa0030a0101a2820202048201fe4e544c4d535350003000000180018...
  ▼ GSS-API Generic Security Service Application Program Interface
    ▼ Simple Protected Negotiation
      ▼ negTokenTarg
        negResult: accept-incomplete (1)
        responseToken: 4e544c4d53535000300000018001800980000003e013e01b0000000c000c0058000000...
      ▼ NTLM Secure Service Provider
        NTLMSSP identifier: NTLMSSP
        NTLM Message Type: NTLMSSP_AUTH (0x00000003)
        > Lan Manager Response: 0000000000000000000000000000000000000000000000000000000000000000
        LMv2 Client Challenge: 0000000000000000
      ▼ NTLM Response: c0cc803a6d9fb5a9082253a04dbd4cd401010000000000000e4d59406c6da01cc3dcfc0...
        Length: 318
        Maxlen: 318
        Offset: 176
      ▼ NTLMv2 Response: c0cc803a6d9fb5a9082253a04dbd4cd401010000000000000e4d59406c6da01cc3dcfc0...
        NTPProofStr: c0cc803a6d9fb5a9082253a04dbd4cd4
        Response Version: 1
        Hi Response Version: 1
        Z: 000000000000
        Time: Jun 24, 2024 07:17:33.000000000 UTC
        NTLMv2 Client Challenge: cc3dcfc0de9b5f26
        7. 00000000
```

Ans c0cc803a6d9fb5a9082253a04dbd4cd4

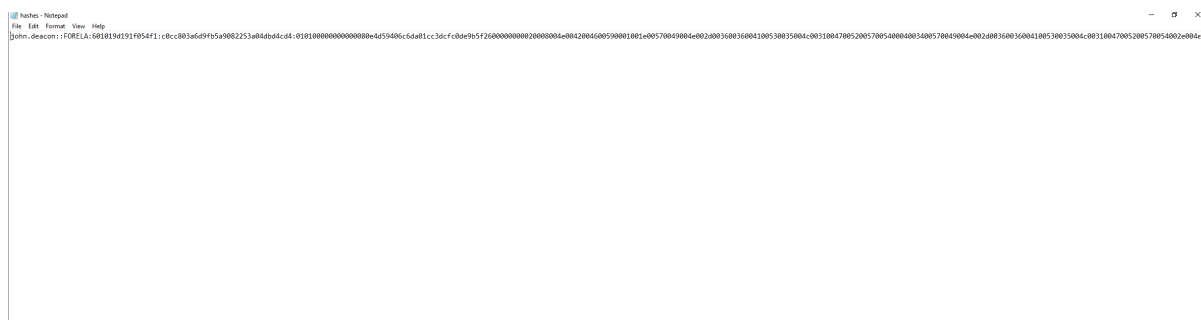
Q8 To test the password complexity, try recovering the password from the information found from packet capture. This is an crucial step as this way we can find whether attacker was able to crack this and how quickly.

Hint: Create a new file and plug in the values as follows .

User::Domain:ServerChallenge:NTPProofStr:NTLMv2Response(without first 16 bytes).

The NTLMv2 Response value can be found from where we found NTPProofStr. Remove the first 16 bytes(32 characters) from the value. Then crack the hash using hashcat. Hashcat syntax will be as following . Hashcat -a0 -m5600 hashfile.txt rockyouwordlist.txt

We now need to create a formula that will be used to crack the hash. It is detailed in the hint of the question. We plug in all the values as we find them from traffic, we just have to be careful when plugging in the NTLMV2Response value. We need to remove the first 32 characters from its value and plug in the rest of the value. This is because the first 32 characters of NTLMv2Response is the value of NTPProofStr which we should have already plugged in before.



The formula becomes

```
john.deacon::FORELA:601019d191f054f1:c0cc803a6d9fb5a9082253a04dbd4cd4:01010000000000000080e4d59406c6da01cc3dcfc0de9b5f2600000000020008004e0042004600590001001e00570049004e002d00360036004100530035004c003100470052005700540004003400570049004e002d00360036004100530035004c003100470052005700540002e004e004200460059002e004c004f004300410
```

04c00030014004e004200460059002e004c004f00430041004c00050014004e004200460059002e00  
4c004f00430041004c000700080080e4d59406c6da01060004000200000008003000300000000000  
00000000000000200000eb2ecbc5200a40b89ad5831abf821f4f20a2c7f352283a35600377e1f294f1c  
90a00100  
0300031000

Now lets crack it using hashcat.

```
Stopped: Tue Jul 09 09:49:11 2024

C:\Users\pc\Downloads\hashcat-6.2.6>hashcat.exe -a0 -m5600 H:\Project-Sherlock\Toxic\hash.txt H:\Project-Sherlock\Toxic\password.txt
hashcat (v6.2.6) starting

Successfully initialized the NVIDIA main driver CUDA runtime library.
Failed to initialize NVIDIA RTC library.
* Device #1: CUDA SDK Toolkit not installed or incorrectly installed.
```

```
Session.....: hashcat
Status.....: Exhausted
Hash.Mode.....: 5600 (NetNTLMv2)
Hash.Target.....: JOHN.DEACON::FORELA:601019d191f054f1:c0cc803a6d9fb5...000000
Time.Started....: Tue Jul 09 09:49:43 2024 (0 secs)
Time.Estimated...: Tue Jul 09 09:49:43 2024 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.....: File (H:\Project-Sherlock\Toxic\password.txt)
Guess.Queue.....: 1/1 (100.00%)
Speed.#1.....: 1012 H/s (0.09ms) @ Accel:1024 Loops:1 Thr:64 Vec:1
Recovered.....: 0/1 (0.00%) Digests (total), 0/1 (0.00%) Digests (new)
Progress.....: 1/1 (100.00%)
Rejected.....: 0/1 (0.00%)
Restore.Point....: 1/1 (100.00%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: NotMyPassword0K? -> NotMyPassword0K?
Hardware.Mon.#1..: Temp: 53c Fan: 34% Util: 29% Core:1770MHz Mem:9501MHz Bus:16
```

Ans NotMyPassword0K?

Q9 Just to get more context surrounding the incident, what is the actual file share which the victim was trying to navigate to?

Hint : Filter for smb traffic. In wireshark it is "SMB2" . Then keep scrolling until you see a tree connect/disconnect of a NON-DEFAULT File share name. This will make sense if you keep in mind the typo victim made in Question 5.

Filter for SMB traffic once again. Find the keyword tree connect. Here you will find the full file share path which is the only likely Share path when considered in context to the typo name.

172.17.79.136	172.17.79.4	SMB2	146 Close Request File: srvsvc
172.17.79.4	172.17.79.136	SMB2	182 Close Response
172.17.79.136	172.17.79.4	SMB2	174 Tree Connect Request Tree: \\DC01\DC-Confidential
172.17.79.4	172.17.79.136	SMB2	138 Tree Connect Response
172.17.79.136	172.17.79.4	SMB2	234 Create Request File:

Here we see that the victim connects to the file share on the domain controller. If we see the timeline it occurred around 40 seconds after the user john deacon made a typo.

Ans \\DC01\DC-Confidential