MDT, where are you?

TLDR

Search for the `intellimirrorSCP` object class and its `netbootServer` attribute to find a WDS server that can serve as a MDT share or a Distribution Point.

What's MDT, and MDT background

According to the Microsoft's documentation:

> MDT is a unified collection of tools, processes, and guidance for automating desktop and server deployment. You can use it to create reference images or as a complete deployment solution. MDT is one of the most important tools available to IT professionals today.
>
> In addition to reducing deployment time and standardizing desktop and server images, MDT enables you to more easily manage security and ongoing configurations. MDT builds on top of the core deployment tools in the Windows Assessment and Deployment Kit (Windows ADK) with more guidance and features designed to reduce the complexity and time required for deployment in an enterprise environment.
>
> MDT supports the deployment of Windows 10, and Windows 7, Windows 8.1, and Windows Server. It also includes support for zero-touch installation (ZTI) with Microsoft Configuration Manager.

Recently, TrustedSec has dropped a pretty interesting blog post about MDT shares that has caught my attention.

To resume it really quickly, Microsoft Deployment Toolkit (MDT) is often overlooked compared to SCCM, but as a "Lite-Touch Installation" tool, it frequently stores plaintext credentials (e.g., domain join, network access, local admin) within plaintext files on its deployment share—most notably in `Bootstrap.ini` and `CustomSettings.ini`.

Typically, misconfigured shares with overly permissive access enable any authenticated AD user to browse and read these files, exposing critical credentials, like Domain Admin's ones. Additional

sensitive data may also reside in:

- Task sequence XML files (`ts.xml`)
- Unattend files (`unattend.xml`)
- Custom scripts within the share

These exposed credentials (e.g., `DomainAdmin`, `UserID`, `AdminPassword`, `DBPwd`) can facilitate domain compromise or lateral movement.

I really encourage you to read the TrustedSec's article before going any further in this blog post.

But, where ?

The article ends by saying that a good technique to identify that a MDT server is deployed on the network, is to search for the `HKLM\Software\Microsoft\Deployment 4\` registry key on already compromised machines. This key doesn't specify where the MDT service lies, but only indicate that a MDT server is probably somewhere.

Even if its already a good starting point, in large network finding a specific server can be a real pain. Maybe there is some attributes or exposed services that could be searched for, in order to more easily snipe the server?

LDAP is a good candidate for this. Generally, when services are deployed in an Active Directory, LDAP operations are performed in order to setup attributes, containers, schema modifications and

so on. So, the plan is really basic: let's deploy a MDT server, and monitor all the LDAP activity to find some stuff that could be targeted during a recon.

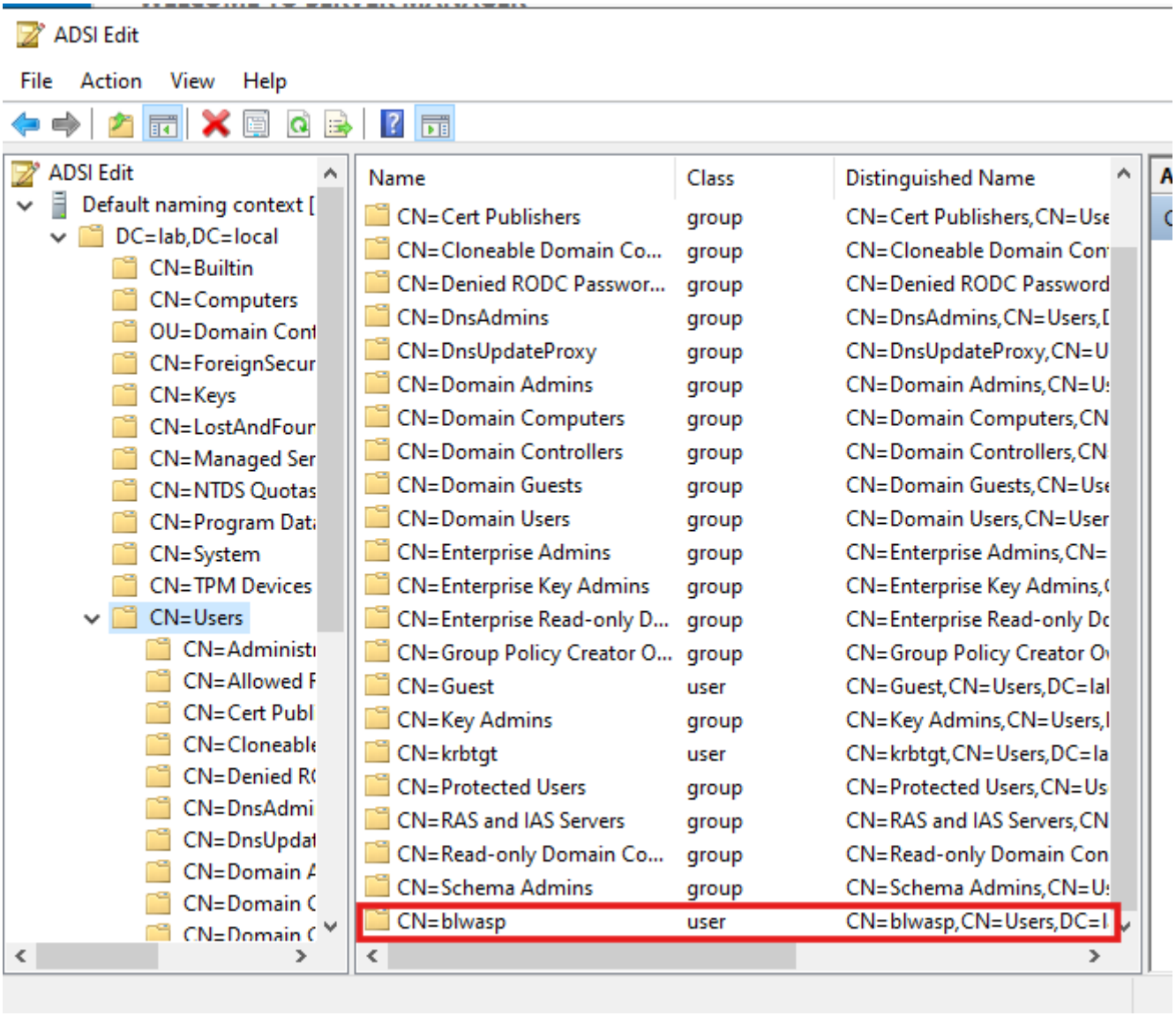
The lab environment is as follow:

- DC1.lab.local, IP: 192.168.56.10, the Domain Controller
- MDTServer.lab.local, IP: 192.168.56.20, the server that will host MDT
- Kali machine, IP: 172.22.110.222, a WSL VM that will be used for monitoring and execute tools

LDAPMonitor got your back

To best way I have found to live monitor the LDAP activity is LDAPMonitor. This tool, developped in Python by @podalirius_, allows to view in real time the modifications performed in the LDAP directory.

To check the setup, I just create a new standard user in the ADSI:



And the addition shows up:

```
$ python3 pyLDAPmonitor.py -u Administrator -p Password123! -d LAB --dc-ip 192.168.56.10 --
ignore-user-logon
[+]=====================================================
[+]    LDAP live monitor v1.3        @podalirius_
[+]=====================================================

[>] Trying to connect to 192.168.56.10 ...
[>] Listening for LDAP changes ...
[2025-06-27 13:42:33] 'CN=blwasp,CN=Users,DC=lab,DC=local' was added.
[2025-06-27 13:42:33] CN=RID Set,CN=DC1,OU=Domain Controllers,DC=lab,DC=local
 | Attribute "rIDNextRID" changed from '1103' to '1104'
```

Good, time to move to MDT.

MDT deployment

WDS installation

Microsoft WDS (Windows Deployment Services) is a server role provided by Microsoft to facilitate the network-based deployment of Windows operating systems to client machines. It allows system administrators to deploy Windows OS images to computers without physical media (DVD/USB) by using PXE boot (Preboot Execution Environment). It automates the installation process, reducing manual setup effort in large-scale environments.

In the case of MDT deployment, WDS will be mandatory to deploy Windows images. The WDS installation and setup can be performed with 3 PowerShell commands on the server that will host it:

```
PS C:\> Install-WindowsFeature -Name WDS -IncludeManagementTools
PS C:\> WDSUTIL.exe /Verbose /Progress /Initialize-Server /Server:MDTServer
/RemInst:"C:\RemoteInstall"
PS C:\> WDSUTIL.exe /Set-Server /AnswerClients:All
```

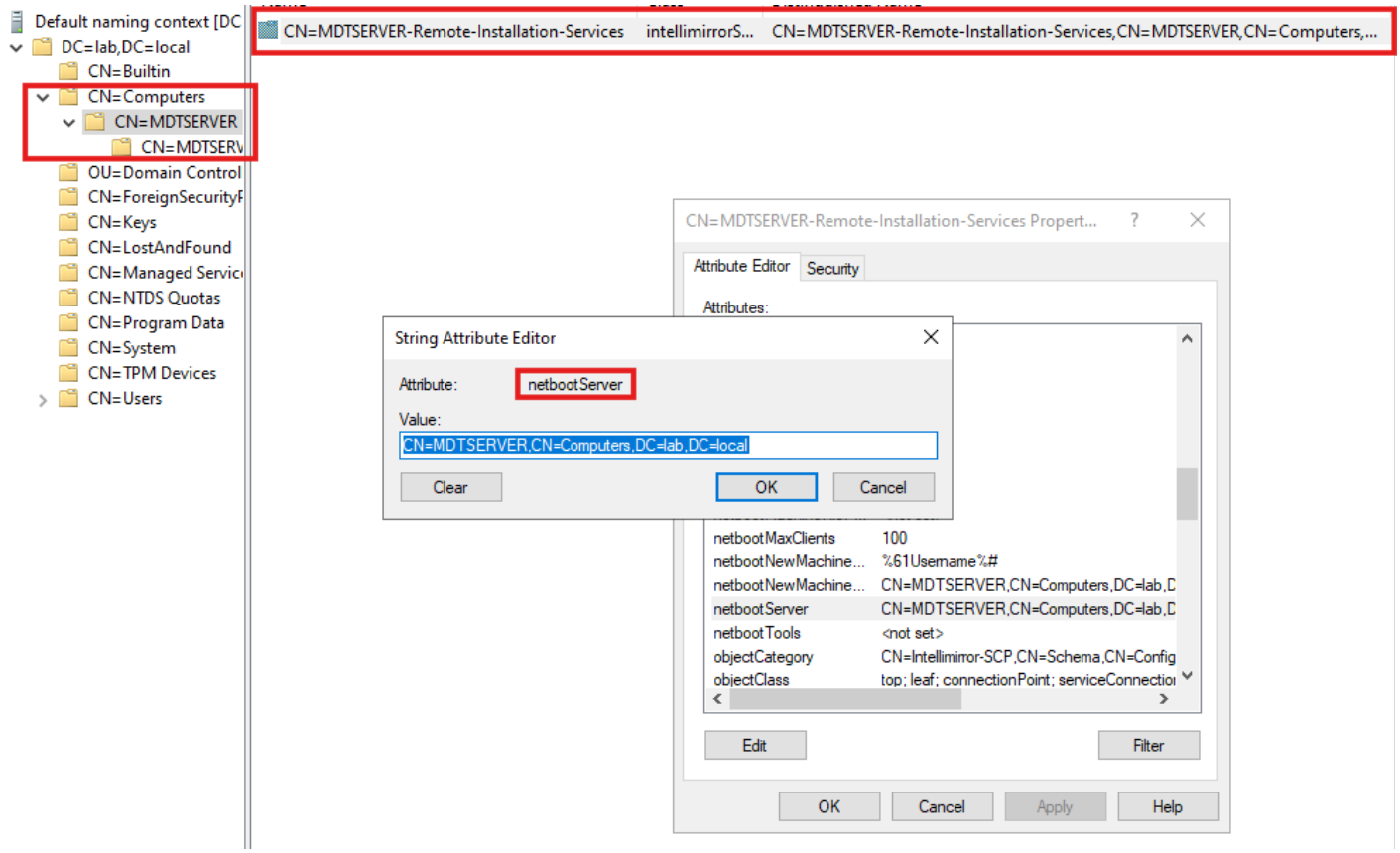
During the WDS initialization, the following LDAP operations are visible:

```
[2025-06-27 13:51:43] 'CN=MDTSERVER-Remote-Installation-
Services,CN=MDTSERVER,CN=Computers,DC=lab,DC=local' was added.
[2025-06-27 13:51:43] CN=MDTSERVER,CN=Computers,DC=lab,DC=local
 | Attribute "netbootSCPBL" = '['CN=MDTSERVER-Remote-Installation-
Services,CN=MDTSERVER,CN=Computers,DC=lab,DC=local']' was created.
[2025-06-27 13:53:06] CN=MDTSERVER-Remote-Installation-
Services,CN=MDTSERVER,CN=Computers,DC=lab,DC=local
 | Attribute "whenChanged" changed from '2025-06-27 11:51:00+00:00' to '2025-06-27
11:52:57+00:00'
 | Attribute "uSNChanged" changed from '28758' to '28760'
 | Attribute "netbootAnswerRequests" changed from 'False' to 'True'
```

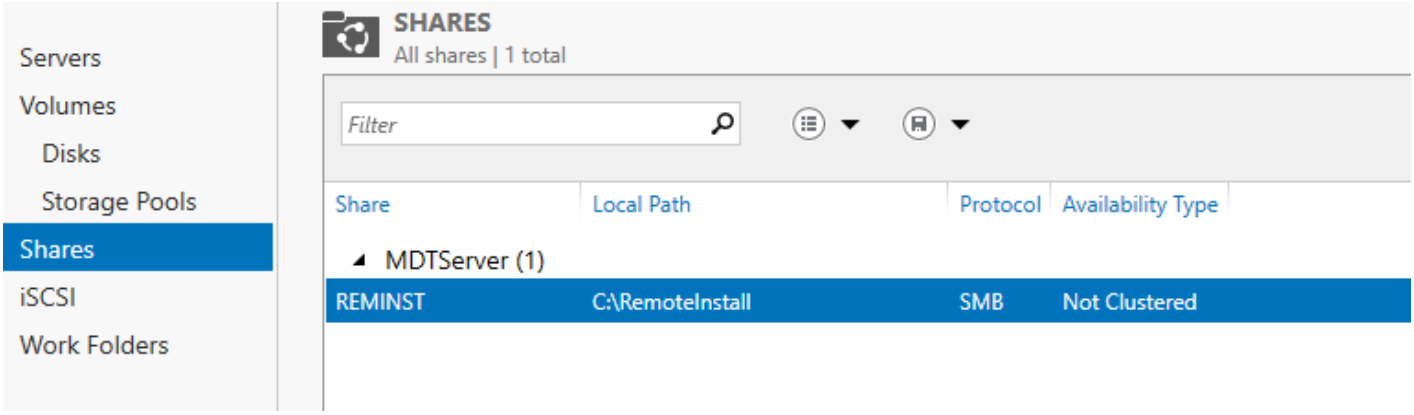
A new container, named `MDTSERVER-Remote-Installation-Services`, is added under the `MDTSERVER` server container.

By looking at the new `MDTSERVER-Remote-Installation-Services` object's attributes, we find `netbootServer`

. According to the Microsoft Documentation, this attribute holds the distinguished name of a Netboot server, and in our case, the DN of the WDS server.



Additionally, a new `REMINST` share appears on the server:



Also visible with Netexec, obviously:

```
$ nxc smb 192.168.56.20 -u BlWasp -p Password123! --shares
SMB         192.168.56.20   445    MDTSERVER        [*] Windows Server 2022 Build 20348 x64
(name:MDTSERVER) (domain:lab.local) (signing:False) (SMBv1:False)
SMB         192.168.56.20   445    MDTSERVER        [+] lab.local\BlWasp:Password123!
SMB         192.168.56.20   445    MDTSERVER        [*] Enumerated shares
SMB         192.168.56.20   445    MDTSERVER        Share           Permissions     Remark
```

```
SMB         192.168.56.20   445    MDTSERVER        -----           -----------     ------
SMB         192.168.56.20   445    MDTSERVER        ADMIN$                          Remote
Admin
SMB         192.168.56.20   445    MDTSERVER        C$                              Default
share
SMB         192.168.56.20   445    MDTSERVER        IPC$            READ            Remote IPC
SMB         192.168.56.20   445    MDTSERVER        REMINST         READ            Windows
Deployment Services Share
```

And, to be sure, we can look at the RPC endpoints via *epmapper* with Impacket:

```
$ rpcdump.py LAB/BlWasp:'Password123!'@'192.168.56.20' |grep -i WDS -A 5 -B 5
UUID    : D4051BDE-9CDD-4910-B393-4AA85EC3C482 v1.0
Bindings:
          ncalrpc:[LRPC-5433bff7016b7e884b]
          ncalrpc:[OLE6B13C9DAA4D10A03B4288A5131B1]

Protocol: [MS-WDSC]: Windows Deployment Services Control Protocol
Provider: wdssrv.dll
UUID    : 1A927394-352E-4553-AE3F-7CF4AAFCA620 v1.0
Bindings:
          ncacn_ip_tcp:192.168.56.20[5040]

Protocol: [MS-FASP]: Firewall and Advanced Security Protocol
```

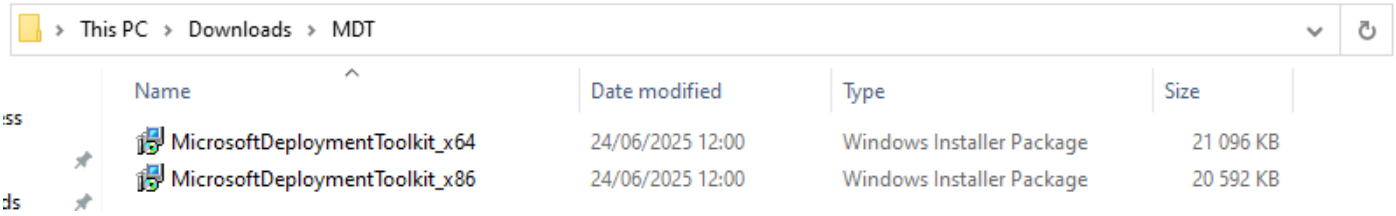
At the end, a new endpoint for the protocol `MS-WDSC` appears! Also, it is worth to note that 521 endpoints are present:

```
$ rpcdump.py LAB/BlWasp:'Password123!'@'192.168.56.20'

[SKIP]
Protocol: N/A
Provider: N/A
UUID    : 650A7E26-EAB8-5533-CE43-9C1DFCE11511 v1.0 Vpn APIs
Bindings:
          ncacn_np:\\MDTSERVER[\PIPE\ROUTER]
          ncalrpc:[RasmanLrpc]
          ncalrpc:[VpnikeRpc]
          ncalrpc:[LRPC-19efb95e3fd58adb94]
```

```
Protocol: N/A
Provider: N/A
UUID    : 7DF1CEAE-DE4E-4E6F-AB14-49636E7C2052 v1.0
Bindings:
          ncalrpc:[LRPC-31e36bccf5e7616642]

[*] Received 521 endpoints.
```

MDT installation

We are now ready to install the MDT environment. To do this, we just have to run the MSI package.



At this point, no new LDAP operation is visible. Additionally, nothing to view regarding the SMB shares. However, by looking at the RPC endpoints, it looks like new ones appear:

```
Protocol: N/A
Provider: N/A
UUID    : 7DF1CEAE-DE4E-4E6F-AB14-49636E7C2052 v1.0
Bindings:
          ncalrpc:[LRPC-31e36bccf5e7616642]

Protocol: N/A
Provider: N/A
UUID    : D4051BDE-9CDD-4910-B393-4AA85EC3C482 v1.0
Bindings:
          ncalrpc:[OLE6B13C9DAA4D10A03B4288A5131B1]
          ncalrpc:[LRPC-5433bff7016b7e884b]

[*] Received 524 endpoints.
```

However, by checking the new endpoints, I was not able to identify anything linked to MDT. By doing a `diff` on the outputs, I got the following new UUID. If someone is able to identify them, and tell if they are, yes or not, related to MDT?

```
$ diff rpc521 rpc524
502a503,508
> UUID    : A4B8D482-80CE-40D6-934D-B22A01A44FE7 v1.0 LicenseManager
> Bindings:
>           ncalrpc:[LicenseServiceEndpoint]
>
> Protocol: N/A
> Provider: N/A
1003c1009
< UUID    : D4051BDE-9CDD-4910-B393-4AA85EC3C482 v1.0
---
> UUID    : AE2DC901-312D-41DF-8B79-E835E63DB874 v1.0 appxsvc
1005,1006c1011,1017
<           ncalrpc:[LRPC-5433bff7016b7e884b]
<           ncalrpc:[OLE6B13C9DAA4D10A03B4288A5131B1]
---
>           ncalrpc:[LRPC-cf7276638bd260fa9f]
>
> Protocol: N/A
> Provider: N/A
> UUID    : FF9FD3C4-742E-45E0-91DD-2F5BC632A1DF v1.0 appxsvc
> Bindings:
>           ncalrpc:[LRPC-cf7276638bd260fa9f]
1083c1094,1101
< [*] Received 521 endpoints.
\ No newline at end of file
---
> Protocol: N/A
> Provider: N/A
> UUID    : D4051BDE-9CDD-4910-B393-4AA85EC3C482 v1.0
> Bindings:
>           ncalrpc:[OLE6B13C9DAA4D10A03B4288A5131B1]
>           ncalrpc:[LRPC-5433bff7016b7e884b]
>
> [*] Received 524 endpoints.
\ No newline at end of file
```

MDT build account creation

> When creating a reference image, you need an account for MDT. The *MDT build account* is used for Windows Preinstallation Environment (Windows PE, not related to Portable Executable) to connect to *[the MDT server]*.

So basically, the goal of this account will be to access the deployment share to retrieve the files to deploy.

```
PS C:\> New-ADUser -Name MDT_BA -UserPrincipalName MDT_BA -path "CN=Users,DC=LAB,DC=LOCAL" -
Description "MDT Build Account" -AccountPassword (ConvertTo-SecureString "Password123!" -
AsPlainText -Force) -ChangePasswordAtLogon $false -PasswordNeverExpires $true -Enabled $true
```

A second service account is mentionned in the documentation. Since MDT should permit to deploy new machines, this account will be used to add the new machines to the Active Directory, in the desired OU.

```
PS C:\> New-ADUser -Name MDT_JD -UserPrincipalName MDT_JD@lab.local -path
"CN=Users,DC=LAB,DC=LOCAL" -Description "MDT join domain account" -AccountPassword (ConvertTo-
SecureString "Password123!" -AsPlainText -Force) -ChangePasswordAtLogon $false -
PasswordNeverExpires $true -Enabled $true
```

Then, the documentation advise to configure the following permissions for this account on the OU where the machines will be added:

> The following list is of the permissions being granted:
>
> Scope: This object and all descendant objects Create Computer objects Delete Computer objects Scope: Descendant Computer objects Read All Properties Write All Properties Read Permissions Modify Permissions Change Password Reset Password Validated write to DNS host name Validated write to service principal name

From an attacker point of view, finding an account with a name or a description mentionning MDT is a pretty good indicator that a MDT deployment service is in place. The command I have specified above are copied from the documentation (only the DN are changed), so it's maybe probable that sysadmins also copy paste them, and use the same usernames and descriptions, who knows?

Log folder

> By default MDT stores the log files locally on the client. In order to capture a reference image, you'll need to enable server-side logging and, to do that, you'll need to have a folder in which to store the logs.

```
PS C:\> New-Item -Path C:\Logs -ItemType directory
PS C:\> New-SmbShare -Name Logs$ -Path C:\Logs -ChangeAccess EVERYONE
PS C:\> icacls C:\Logs /grant '"MDT_BA":(OI)(CI)(M)'
```

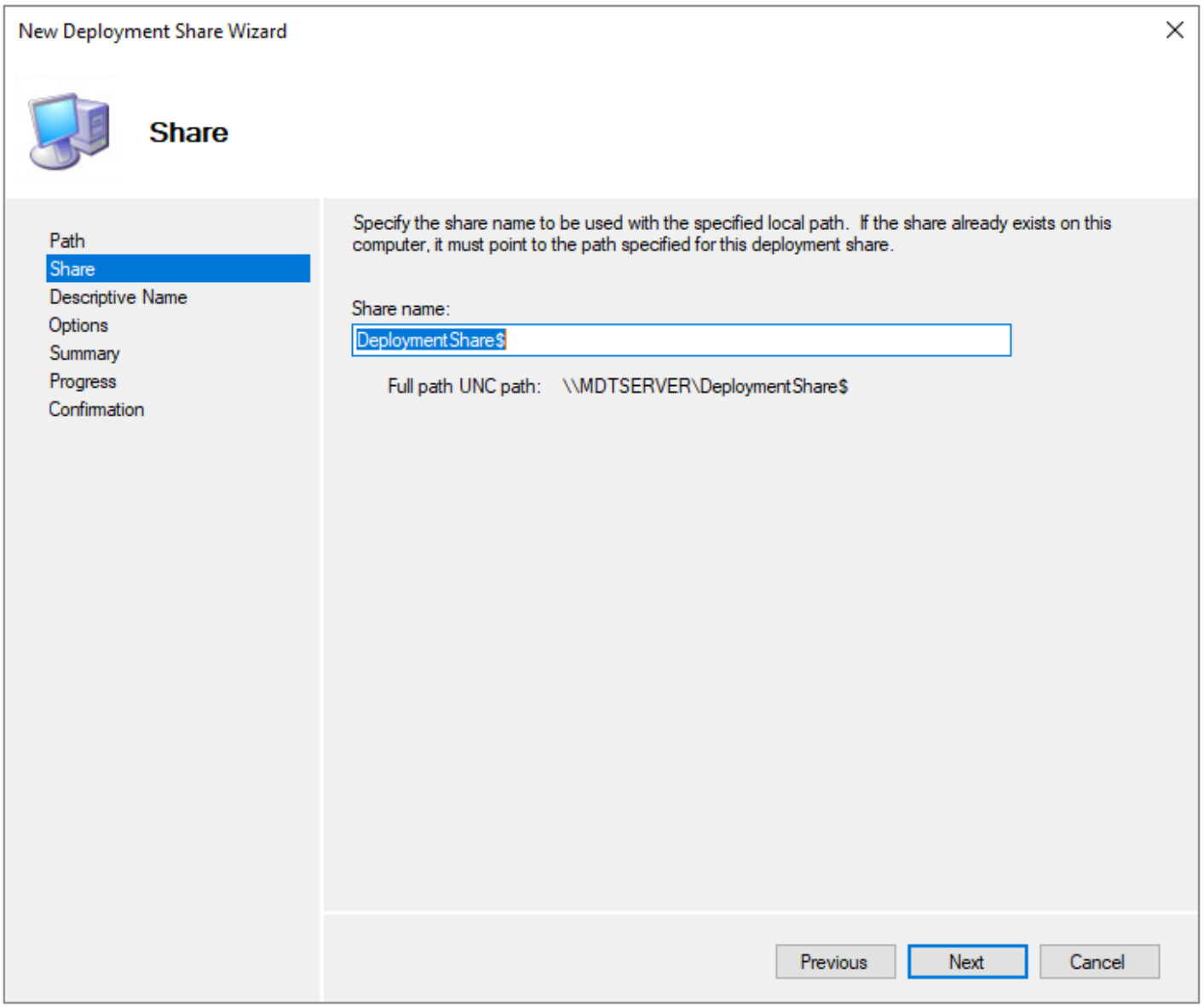
So, yeah, the official documentation recommend to set the permissions to full access for `EVERYONE` ... If we look at it with NXC and our standard user:

```
$ nxc smb 192.168.56.20 -u BlWasp -p Password123! --shares
SMB         192.168.56.20   445    MDTSERVER        [*] Windows Server 2022 Build 20348 x64
(name:MDTSERVER) (domain:lab.local) (signing:False) (SMBv1:False)
SMB         192.168.56.20   445    MDTSERVER        [+] lab.local\BlWasp:Password123!
SMB         192.168.56.20   445    MDTSERVER        [*] Enumerated shares
SMB         192.168.56.20   445    MDTSERVER        Share           Permissions     Remark
SMB         192.168.56.20   445    MDTSERVER        -----           -----------     ------
SMB         192.168.56.20   445    MDTSERVER        ADMIN$                          Remote
Admin
SMB         192.168.56.20   445    MDTSERVER        C$                              Default
share
SMB         192.168.56.20   445    MDTSERVER        IPC$            READ            Remote IPC
SMB         192.168.56.20   445    MDTSERVER        Logs$           READ,WRITE
SMB         192.168.56.20   445    MDTSERVER        REMINST         READ            Windows
Deployment Services Share
```

I'm not really sure that providing full access to everyone on the log folder is a best security practice, but that a good point to know, from an attacker perspective.

MDT deployment share setup

Ok, we are now ready to create the deployment share on the MDT server. The setup is performed via the DeploymentWorkbench utility, and the default name for the share in the tool is `DeploymentShare$`:



However, the Microsoft documentation indicates `MDTProduction$` as an example. So, maybe there is a chance that one of these names is used in production environment.

When the setup is finished, the new share is obviously visible with NXC:

```
$ nxc smb 192.168.56.20 -u BlWasp -p Password123! --shares
SMB         192.168.56.20   445    MDTSERVER        [*] Windows Server 2022 Build 20348 x64
(name:MDTSERVER) (domain:lab.local) (signing:False) (SMBv1:False)
SMB         192.168.56.20   445    MDTSERVER        [+] lab.local\BlWasp:Password123!
SMB         192.168.56.20   445    MDTSERVER        [*] Enumerated shares
SMB         192.168.56.20   445    MDTSERVER        Share           Permissions     Remark
SMB         192.168.56.20   445    MDTSERVER        -----           -----------     ------
```

```
SMB         192.168.56.20   445    MDTSERVER        ADMIN$                         Remote
Admin
SMB         192.168.56.20   445    MDTSERVER        C$                             Default
share
SMB         192.168.56.20   445    MDTSERVER        DeploymentShare$                MDT
Deployment Share
SMB         192.168.56.20   445    MDTSERVER        IPC$            READ           Remote IPC
SMB         192.168.56.20   445    MDTSERVER        Logs$           READ,WRITE
SMB         192.168.56.20   445    MDTSERVER        REMINST         READ           Windows
Deployment Services Share
```

Note that, for the moment, we do not have access to the share. The next step is to configure the permissions on it. The documentation recommend to only provide access to the `MDT_BA` account via NTFS and SMB permissions. But, as highlighted by TrustedSec, it is not uncommon to come across too permissive rights. Here are the permissions indicated by Microsoft:

```
PS C:\> icacls.exe "C:\DeploymentShare" /grant '"LAB\MDT_BA":(OI)(CI)(M)'
PS C:\> Grant-smbshareaccess -Name DeploymentShare$ -AccountName "LAB\MDT_BA" -AccessRight
Full -force
```

At this point, no new LDAP operation is visible.

We should now add images, application, task sequences, and so on to the share, and configure the share to perform dynamic deployment. This is out of scope for this short blog post.

That was a cool sysadmin tuto, but where is the server?

So, since the point of this blog post is to identify to MDT server in the network, let's do it! If we look back at the installation, we have seen that a specific LDAP operation was performed during the WDS server deployment. In fact, this service's installation create a new node under the container of the server supporting the server, `CN=MDTSERVER-Remote-Installation-Services`, with a specific `objectclass` `intellimirrorSCP`. And since this service is mandatory to run MDT, it should be a good starting point.

However, it is important to note that **WDS servers can be used for other stuff than MDT shares!** In fact, they are also used in SCCM environments for the Distribution Point and PXE servers for example (even if WDS is no more recommended for SCCM, and Microsoft advises to use SCCM OSD (*Operating System Deployment*)). So, based on this, **the detection of a WDS server is a good indicator, but not the assurance of having found a MDT share.**

Let's run our favorite tool to perform LDAP operations, ldeep, and search for the specific `classObject`:

```
$ pdm run ldeep ldap -u BlWasp -p Password123! -d LAB.LOCAL -s 192.168.56.10 search
'(objectclass=intellimirrorSCP)'
[{
  "cn": "MDTSERVER-Remote-Installation-Services",
  "dSCorePropagationData": [
    "1601-01-01T00:00:00+00:00"
  ],
  "distinguishedName": "CN=MDTSERVER-Remote-Installation-
Services,CN=MDTSERVER,CN=Computers,DC=lab,DC=local",
  "dn": "CN=MDTSERVER-Remote-Installation-
Services,CN=MDTSERVER,CN=Computers,DC=lab,DC=local",
  "instanceType": 4,
  "name": "MDTSERVER-Remote-Installation-Services",
  "netbootAllowNewClients": true,
  "netbootAnswerOnlyValidClients": false,
  "netbootAnswerRequests": true,
  "netbootCurrentClientCount": 0,
  "netbootLimitClients": false,
  "netbootMaxClients": 100,
  "netbootNewMachineNamingPolicy": [
    "%61Username%#"
  ],
  "netbootNewMachineOU": "CN=MDTSERVER,CN=Computers,DC=lab,DC=local",
  "netbootServer": "CN=MDTSERVER,CN=Computers,DC=lab,DC=local",
  "objectCategory": "CN=Intellimirror-SCP,CN=Schema,CN=Configuration,DC=lab,DC=local",
  "objectClass": [
    "top",
    "leaf",
    "connectionPoint",
    "serviceConnectionPoint",
    "serviceAdministrationPoint",
```

```
    "intellimirrorSCP"
  ],
  "objectGUID": "{5ece04db-3566-4dbf-b453-2aa81d919239}",
  "showInAdvancedViewOnly": true,
  "uSNChanged": 28760,
  "uSNCreated": 28758,
  "whenChanged": "2025-06-27T11:52:57+00:00",
  "whenCreated": "2025-06-27T11:51:00+00:00"
}]
```

It looks to work. We could also filter the JSON output to only extract the server's DN:

```
$ pdm run ldeep ldap -u BlWasp -p Password123! -d LAB.LOCAL -s 192.168.56.10 search
'(objectclass=intellimirrorSCP)' | jq '.[].netbootServer'
"CN=MDTSERVER,CN=Computers,DC=lab,DC=local"
```

As indicated previously, a WDS server could be another thing than a MDT share, like a SCCM Distribution Point. But, with this information, and by checking manually the exposed SMB shares, it should be sufficient.

Tools and PR

To simplify the MDT research via WDS, I have made a quick pull request on ldeep which add the `wds` command. There wasn't much to do, since the Distribution Point research for the `sccm` command was already implemented, by searching for `"(cn=*-Remote-Installation-Services)"` in the LDAP directory. This is a valid alternative to `(objectclass=intellimirrorSCP)`.

Now, when a WDS server is identified, ldeep just says that a potential Distribution Point or MDT share has been found.

I have also coded a simple tool in Rust, that is able to identify all the WDS servers, and enumerate their shares. To be honest, I have essentially developed it to learn how to work with the `ldap3` library in Rust...

Conclusion

As indicated by TrustedSec, there is no magic technique to identify MDT shares on the network.Or, at least, I couldn't find it either. However, WDS servers are a good indicator of their presence,and can be easily found in LDAP.

Revision #3
Created 7 July 2025 16:51:19 by BlackWasp
Updated 10 July 2025 15:13:39 by BlackWasp