

# **Breaking the Ticket: A Beginner's Guide to Kerberos Attacks**

EKARUCH PURATHANANONT

07 Feb 2025

# Agenda

---

## Background of Kerberos

Background knowledge for kerberos authentication

## Roasting attack

Stealing encrypted Kerberos tickets to crack passwords offline.

## Delegation attack

Exploiting Kerberos to impersonate users and access restricted resources.

## Ticket Abuse

Using stolen or forged Kerberos tickets to gain unauthorized access.

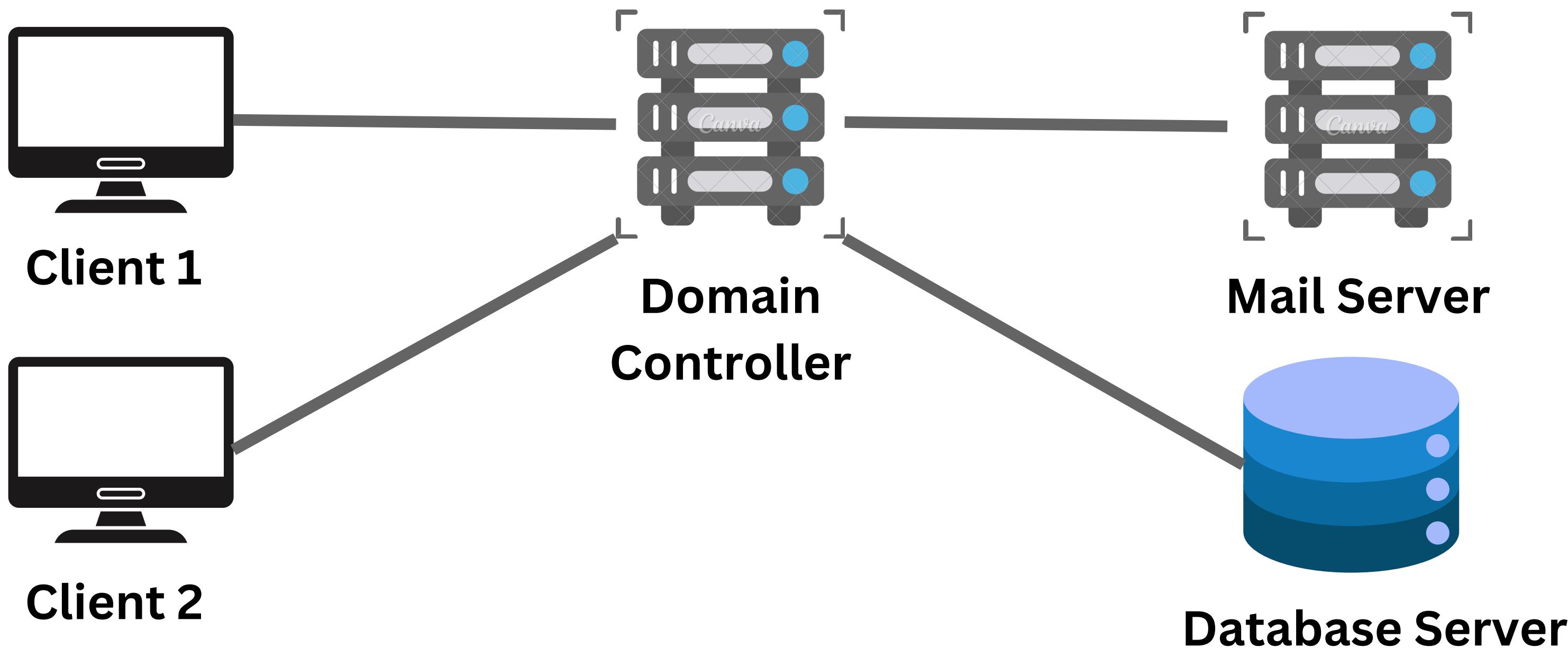
# Background of Kerberos

## About Active Directory

- Management system for window domain network
- Centralize management



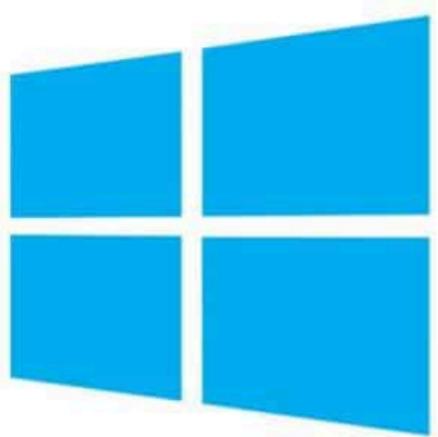
Active Directory



# Background of Kerberos

## Active Directory authentication protocol

- Kerberos
- NTLM (NT LAN Manager)



Active Directory



NTLM AUTHENTICATION  
ACTIVE DIRECTORY

# Background of Kerberos

---

## About Kerberos

- Ticket Base Authentication
- Use ticket to proof identity
- Got **KDC (Key Distribution Center)** as centralize server management

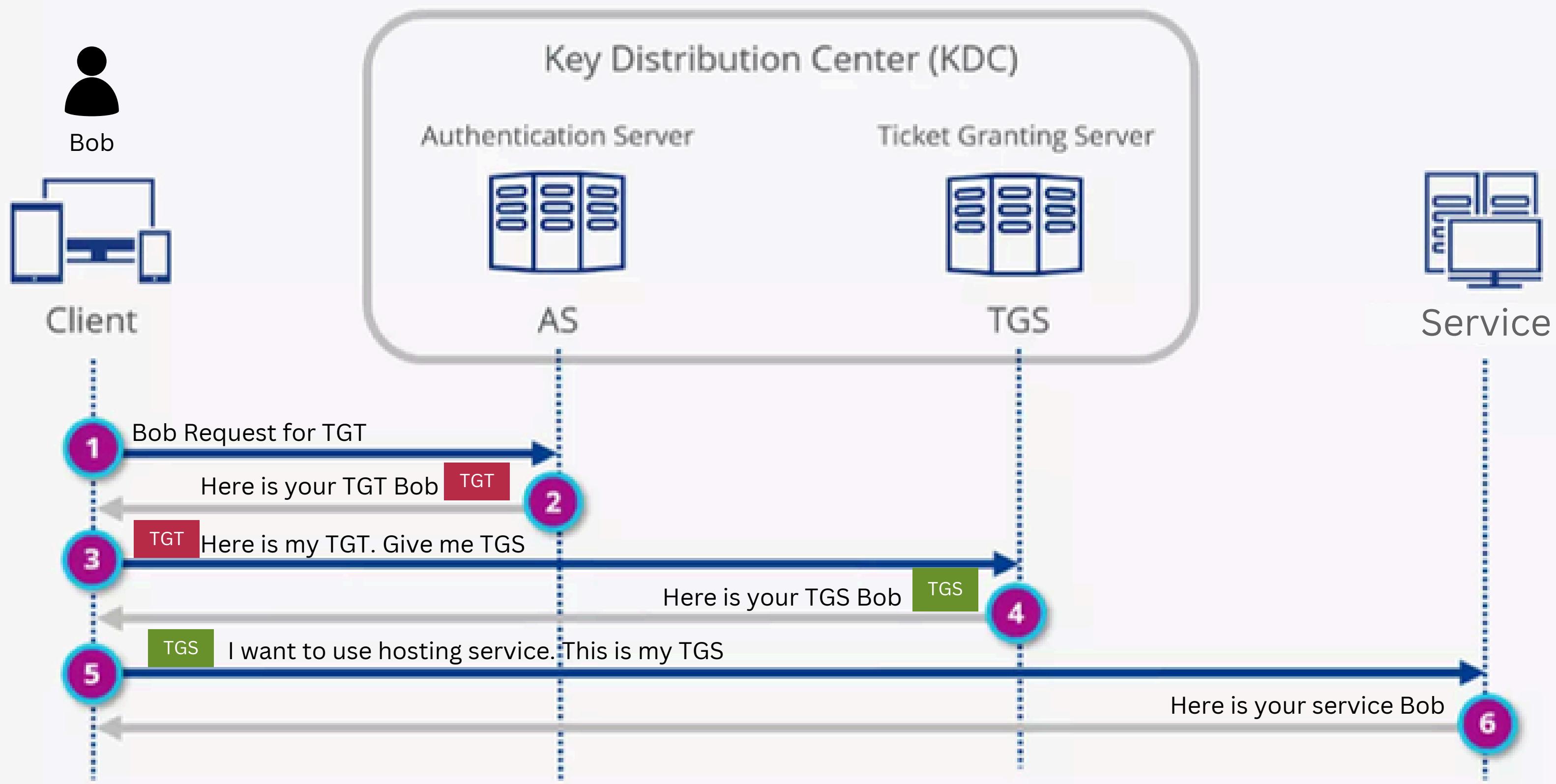
### Ticket

- TGT (Ticket Granting Ticket)
- TGS (Ticket Granting Service)



# Background of Kerberos

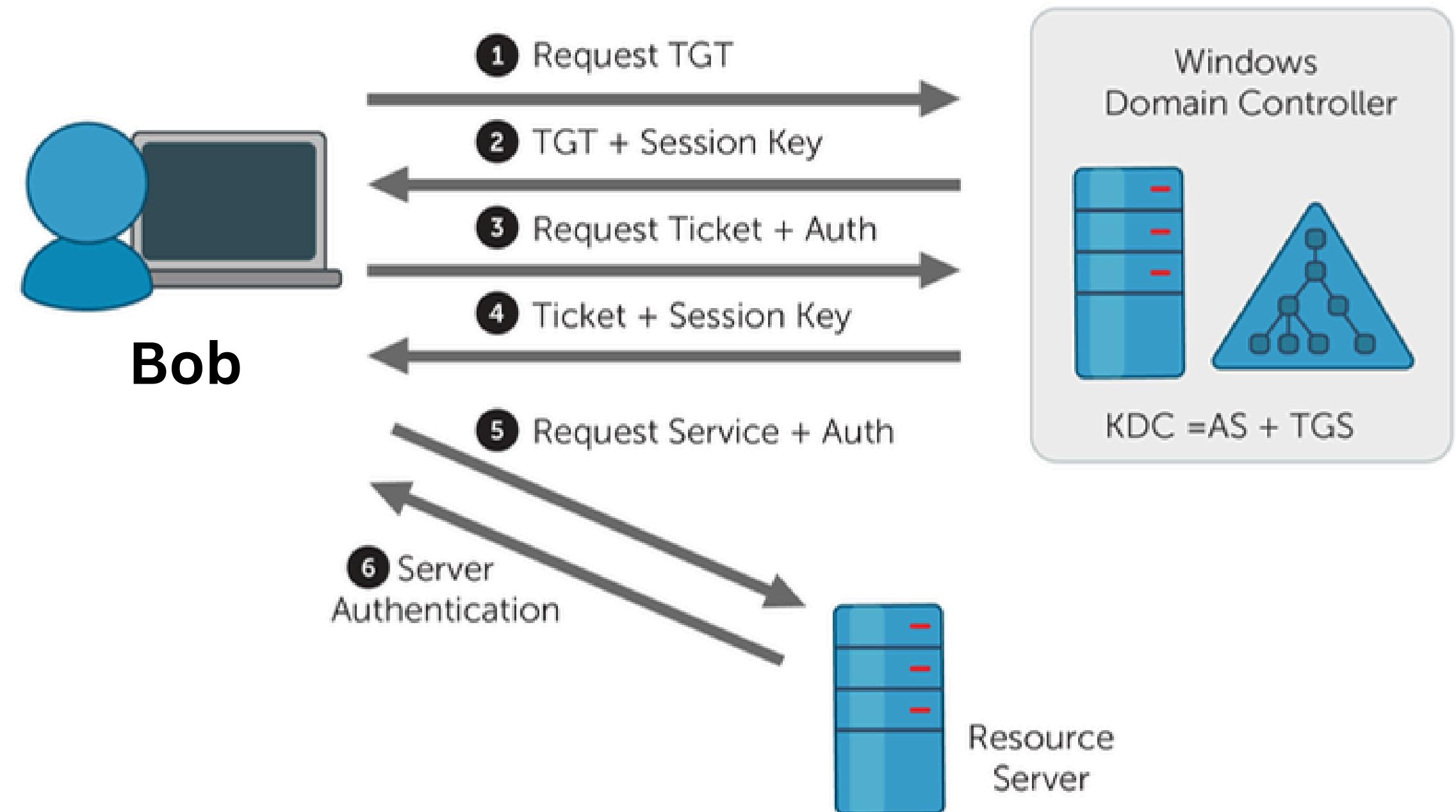
## Kerberos authentication process



# Background of Kerberos

## Kerberos Authentication Flow

1. AS-REQ
2. AS-REP
3. TGS-REQ
4. TGS-REP
5. AP-REQ
6. AP-REP



# **Roasting Attack**

# AS-REP Roasting

## What is AS-REP Roasting

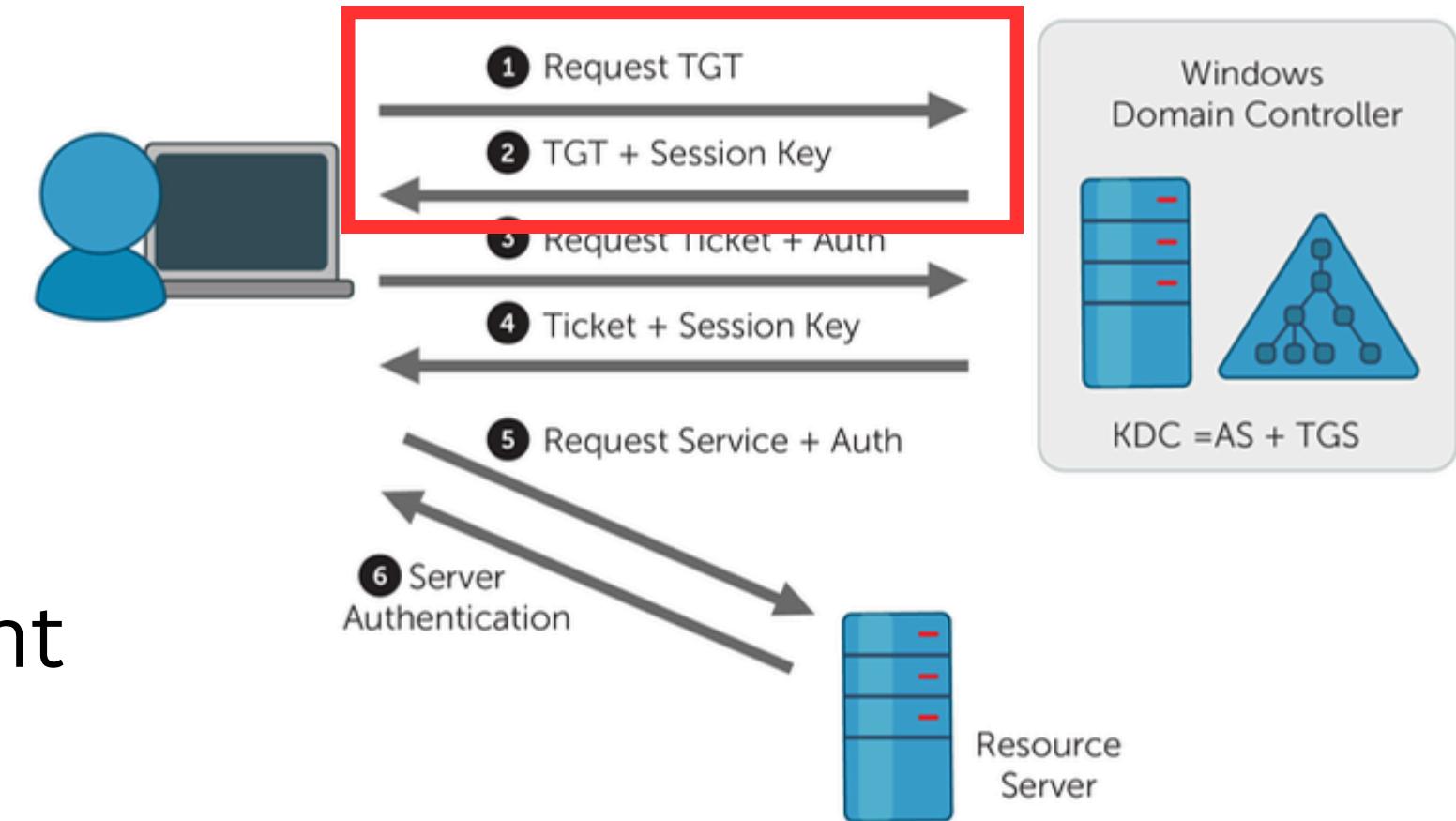
- An attack that aim to cracking the **user password** with AS-REP



# AS-REP Roasting

Normal flow for get TGT ticket

1. Client sent **AS-REQ** to **KDC**
2. **KDC** validate the **AS-REQ** from client
3. **KDC** issue TGT ticket, session key and sent **AS-REP** to Client



## AS-REQ structure

- **Authenticator** (encrypt with user password)
  - timestamp
- **Username**

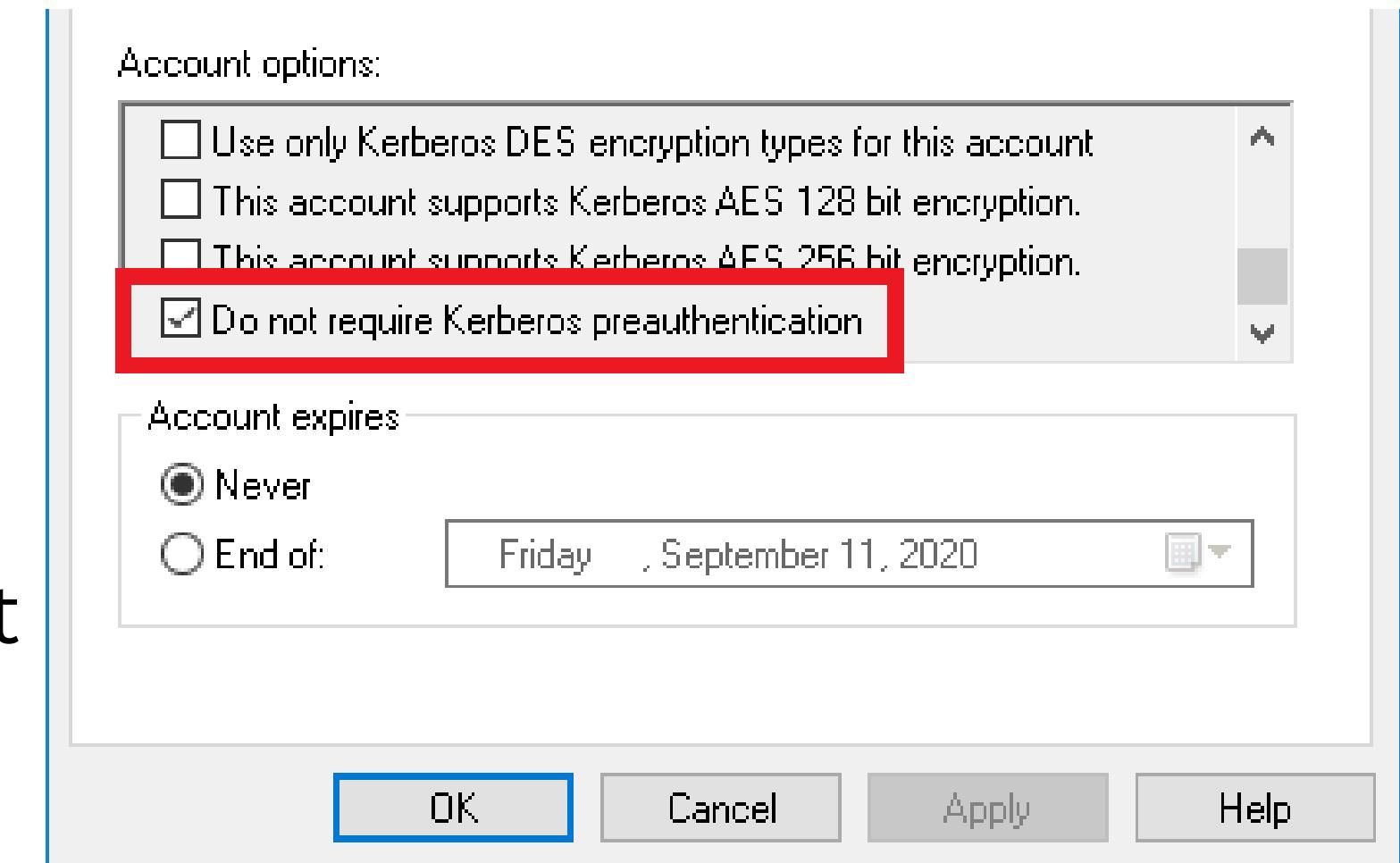
## AS-REP structure

- **Session key A** (encrypt with user password)
- **TGT ticket** (encrypt with KDC key)
  - user information
  - session key A

# AS-REP Roasting

Not require pre-authen flow

1. Client sent **AS-REQ** to **KDC**
2. ~~KDC validate the AS-REQ from client~~
3. KDC issue TGT ticket, session key and sent **AS-REP** to Client



## AS-REQ structure

- Username

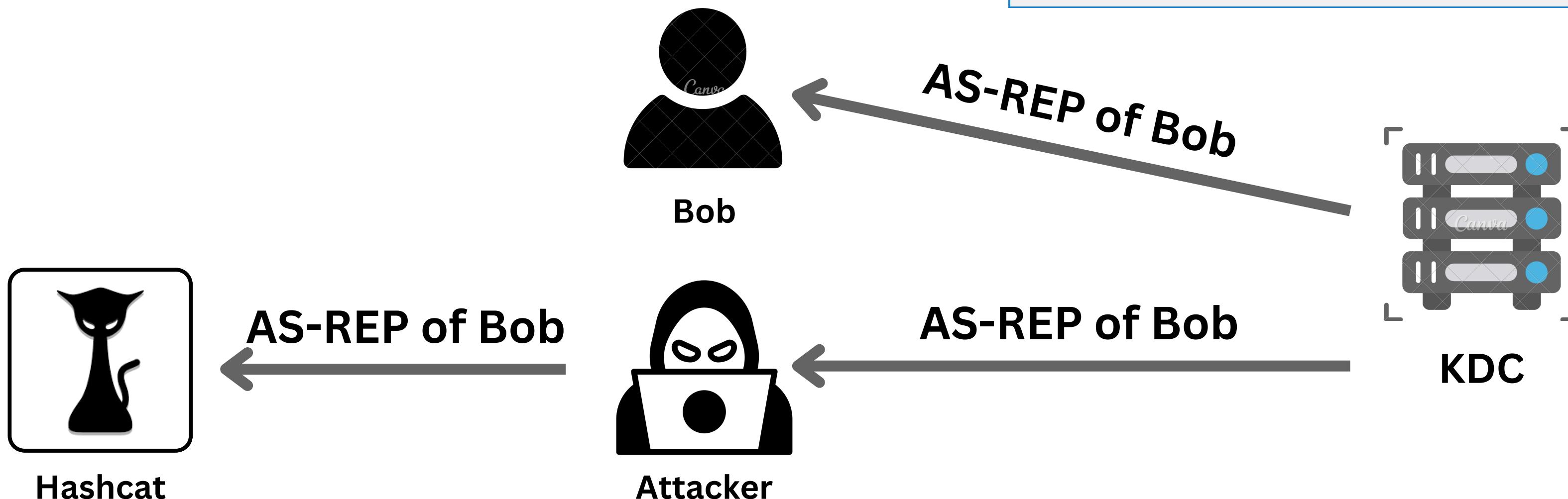
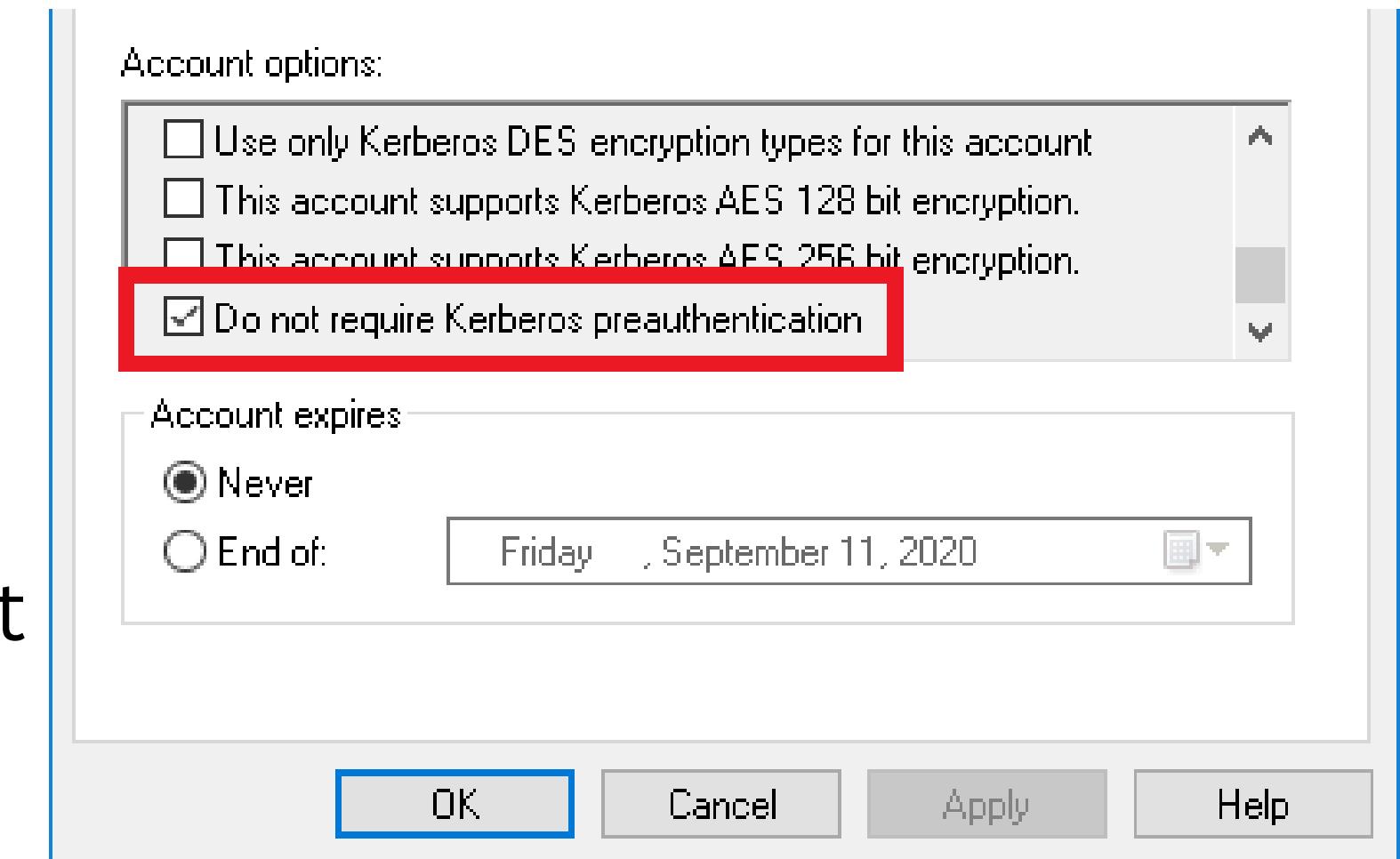
## AS-REP structure

- **Session key A** (encrypt with user password)
- **TGT ticket** (encrypt with KDC key)
  - user information
  - session key A

# AS-REP Roasting

Not require pre-authen flow

1. Client sent **AS-REQ** to KDC
2. ~~KDC validate the AS-REQ from client~~
3. KDC issue TGT ticket, session key and sent **AS-REP** to Client



# AS-REP Roasting

## Condition

- Weak password user
- Not require pre-authentication user
- Valid domain joined user

## Window Tools



## Tools for enumerate

- PowerView, Rubeus (For window)
- GetNPUsers.py (For linux)
- Hashcat

## Linux Tools

**fortra/impacket**

Impacket is a collection of Python classes for working with network protocols.



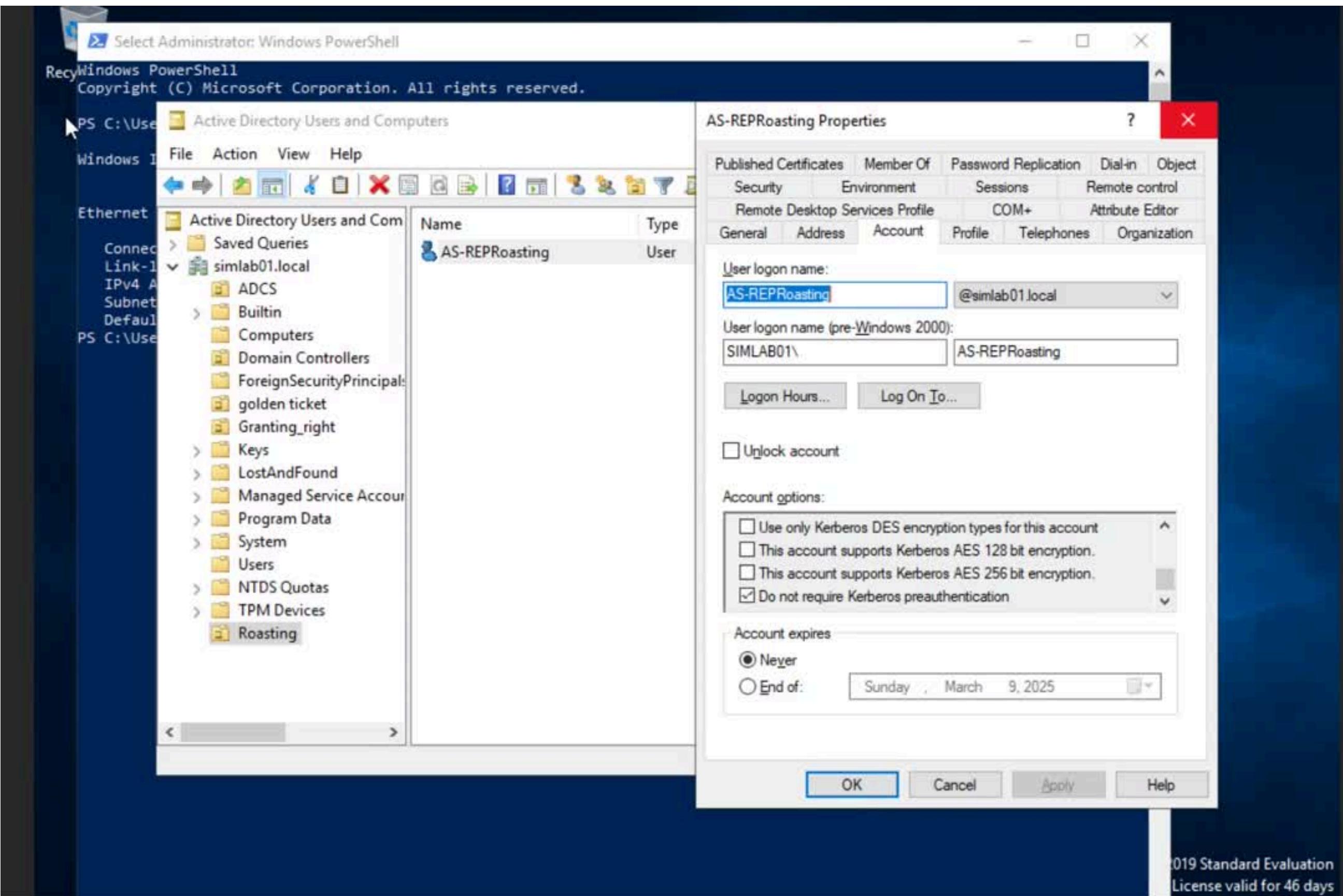
## Impact

- Gain user password

## Hash cracking Tools



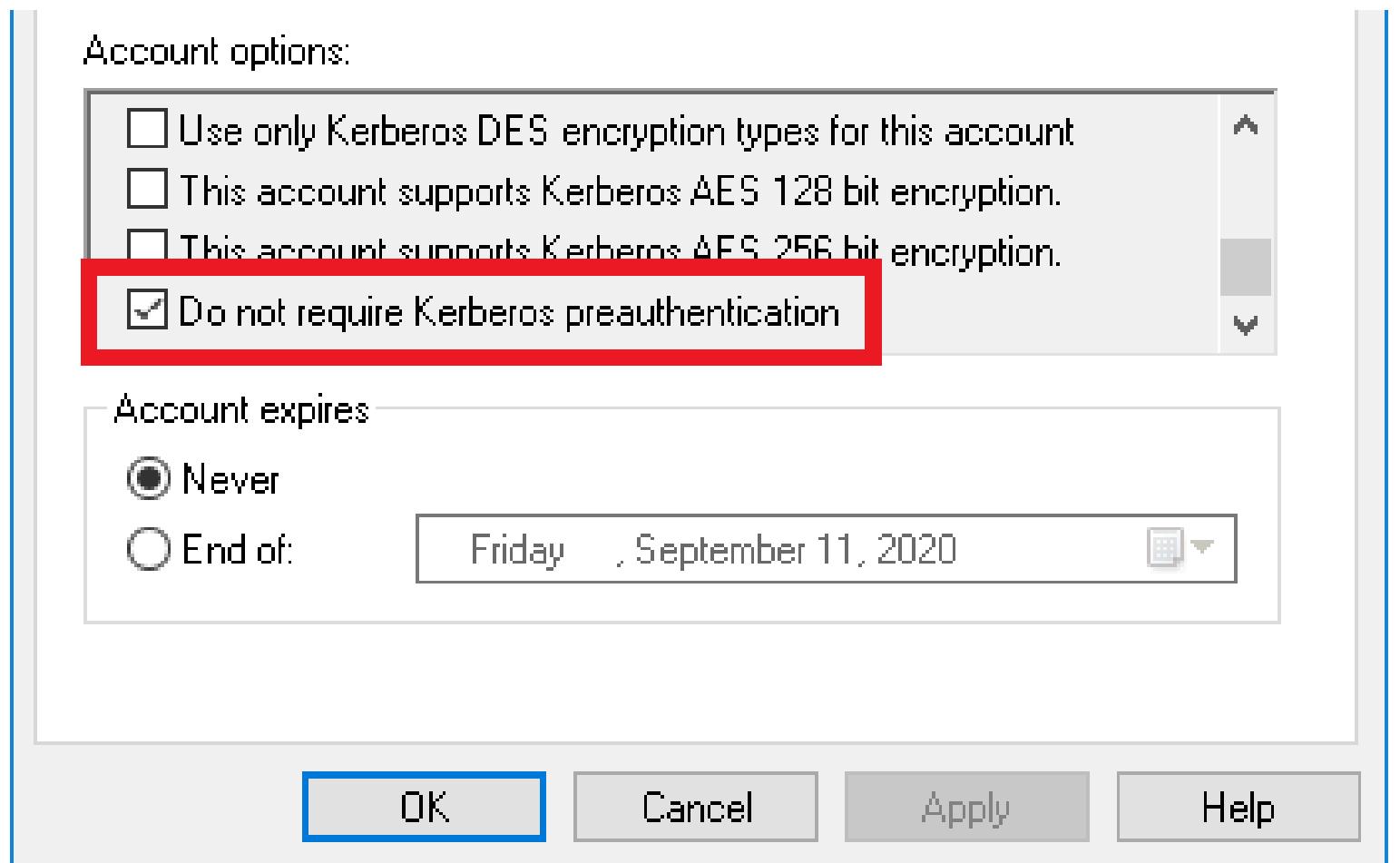
# AS-REPRoasting



# AS-REP Roasting

## Mitigations

- use strong password for user account
- set not require pre-authentication



# Kerberoasting

## What is Kerberoasting

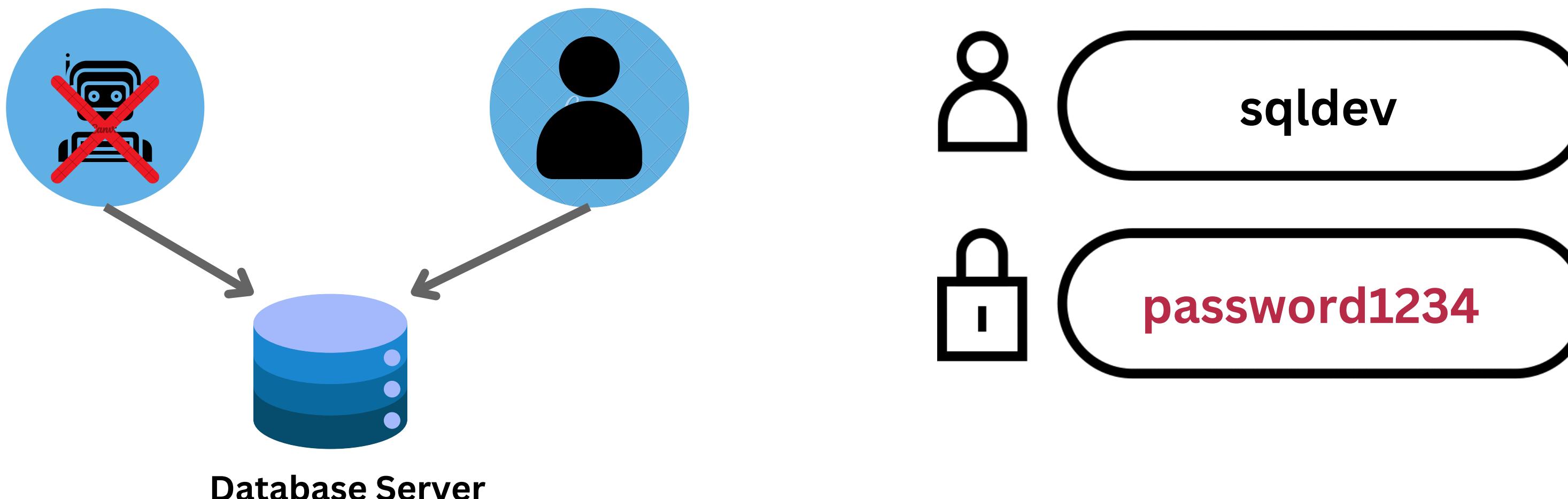
- An attack that aim to cracking the service password on **Ticket Grain Service** (TGS)



# Kerberoasting

## Misconfiguration

- Use **user account** instead of **machine account** to manage the services
- Use **weak password** on user account



# Kerberoasting

## What target that we will focus

- Find user that **SPN is not empty** and manually check the result

```
● ● ●  
$search = New-Object DirectoryServices.DirectorySearcher([ADSI]"")  
$search.filter = "(&(objectCategory=person)(objectClass=user)(servicePrincipalName*))"  
$results = $search.Findall()  
foreach($result in $results)  
{  
    $userEntry = $result.GetDirectoryEntry()  
    Write-host "User"  
    Write-Host "===="  
    Write-Host $userEntry.name "(" $userEntry.distinguishedName ")"  
    Write-host ""  
    Write-host "SPNs"  
    Write-Host "===="  
    foreach($SPN in $userEntry.servicePrincipalName)  
    {  
        $SPN  
    }  
    Write-host ""  
    Write-host ""  
}
```

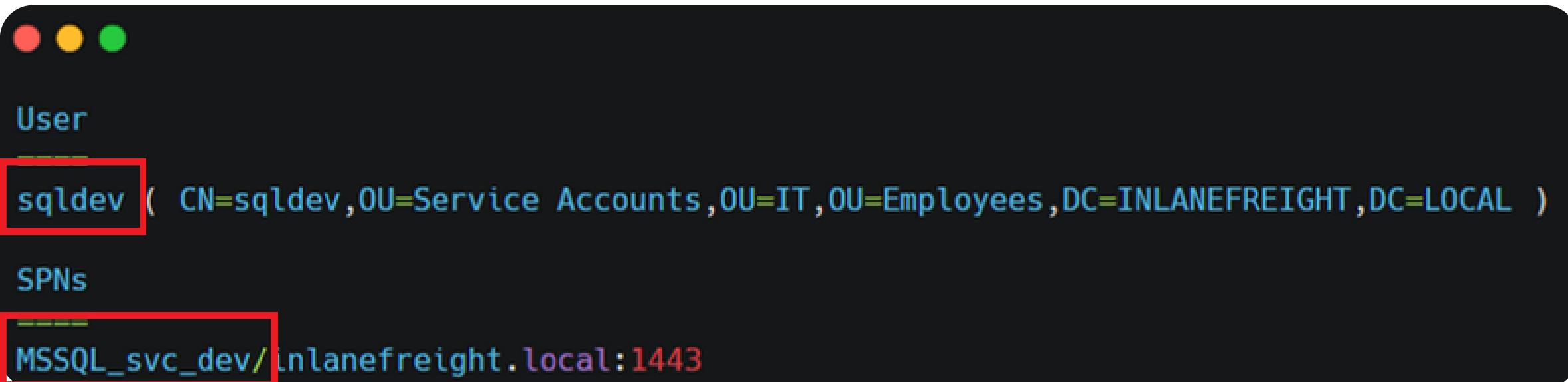
### SPN (Service Principal Name)

- Identify service
- Account that got SPN
  - Service account
  - Computer account

# Kerberoasting

What target that we will focus

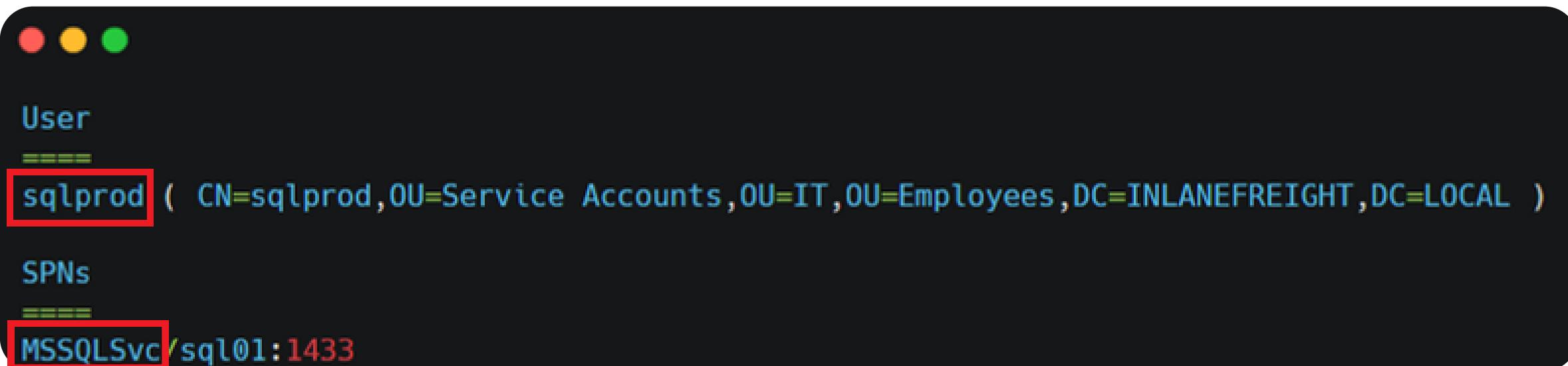
- Service account that **SPN** is associate with **service account**



A terminal window displaying LDAP search results. The results are organized into sections: 'User' and 'SPNs'. The 'User' section contains one entry: 'sqldev ( CN=sqldev,OU=Service Accounts,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL )'. The 'SPNs' section contains one entry: 'MSSQL\_svc\_dev/inlanefreight.local:1443'. Both the 'User' and 'SPN' entries are highlighted with red boxes.

```
User
sqldev ( CN=sqldev,OU=Service Accounts,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL )
SPNs
MSSQL_svc_dev/inlanefreight.local:1443
```

Example 1



A terminal window displaying LDAP search results. The results are organized into sections: 'User' and 'SPNs'. The 'User' section contains one entry: 'sqlprod ( CN=sqlprod,OU=Service Accounts,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL )'. The 'SPNs' section contains one entry: 'MSSQLSvc/sql01:1433'. Both the 'User' and 'SPN' entries are highlighted with red boxes.

```
User
sqlprod ( CN=sqlprod,OU=Service Accounts,OU=IT,OU=Employees,DC=INLANEFREIGHT,DC=LOCAL )
SPNs
MSSQLSvc/sql01:1433
```

Example 2

# Kerberoasting

## Request for TGS with Powerview

- Service account that **SPN** is associate with **service account**

```
PS C:\Tools> Import-Module .\PowerView.ps1
PS C:\Tools> Invoke-Kerberoastq

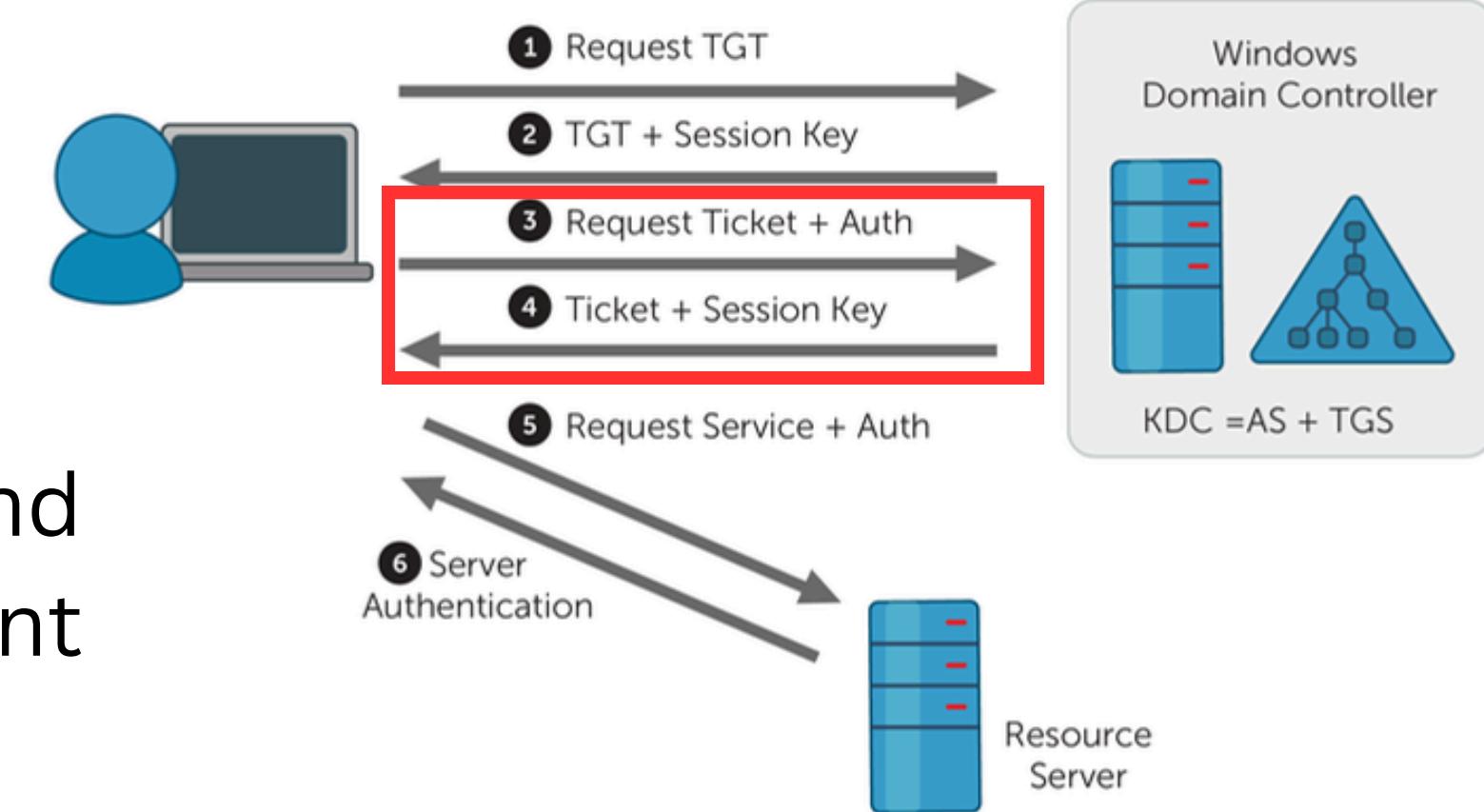
SamAccountName      : sqldev
DistinguishedName   : CN=sqldev,OU=Service Accounts,OU=Roasting,DC=SIMLAB01,DC=LOCAL
ServicePrincipalName : MSSQL_svc_dev/simlab01.local:1443
TicketByteHexStream  :
Hash                :

$krb5tgs$23$*sqldev$SIMLAB01.LOCAL$MSSQL_svc_dev/inlanefreight.local:1443*$29A78F89AC
```

# Kerberoasting

## Normal flow for get TGS ticket

1. Client sent **TGS-REQ** to KDC
2. KDC validate the **TGS-REQ** from client and
3. KDC issue TGS ticket, session key and sent **TGS-REP** to Client



### TGS-REQ structure

- **Authenticator** (encrypt with session key A)
  - timestamp
- **TGT** (encrypt with KDC key)
  - user information
  - Session key A
- **Name of service that will access**

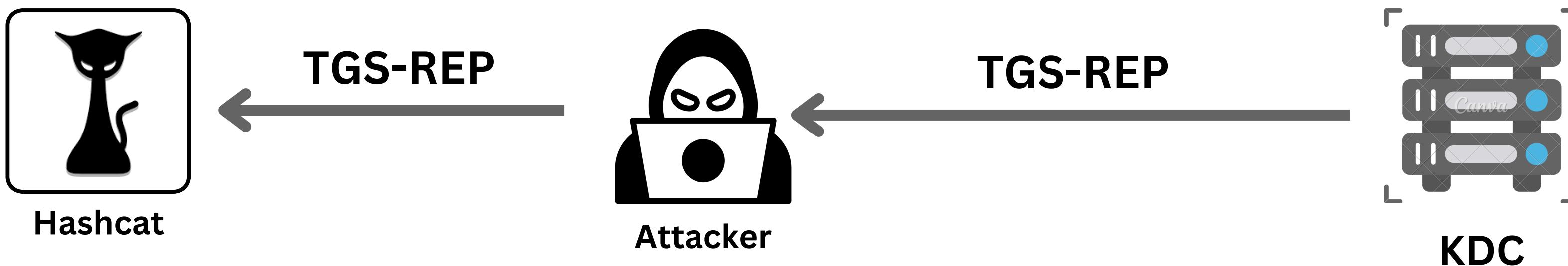
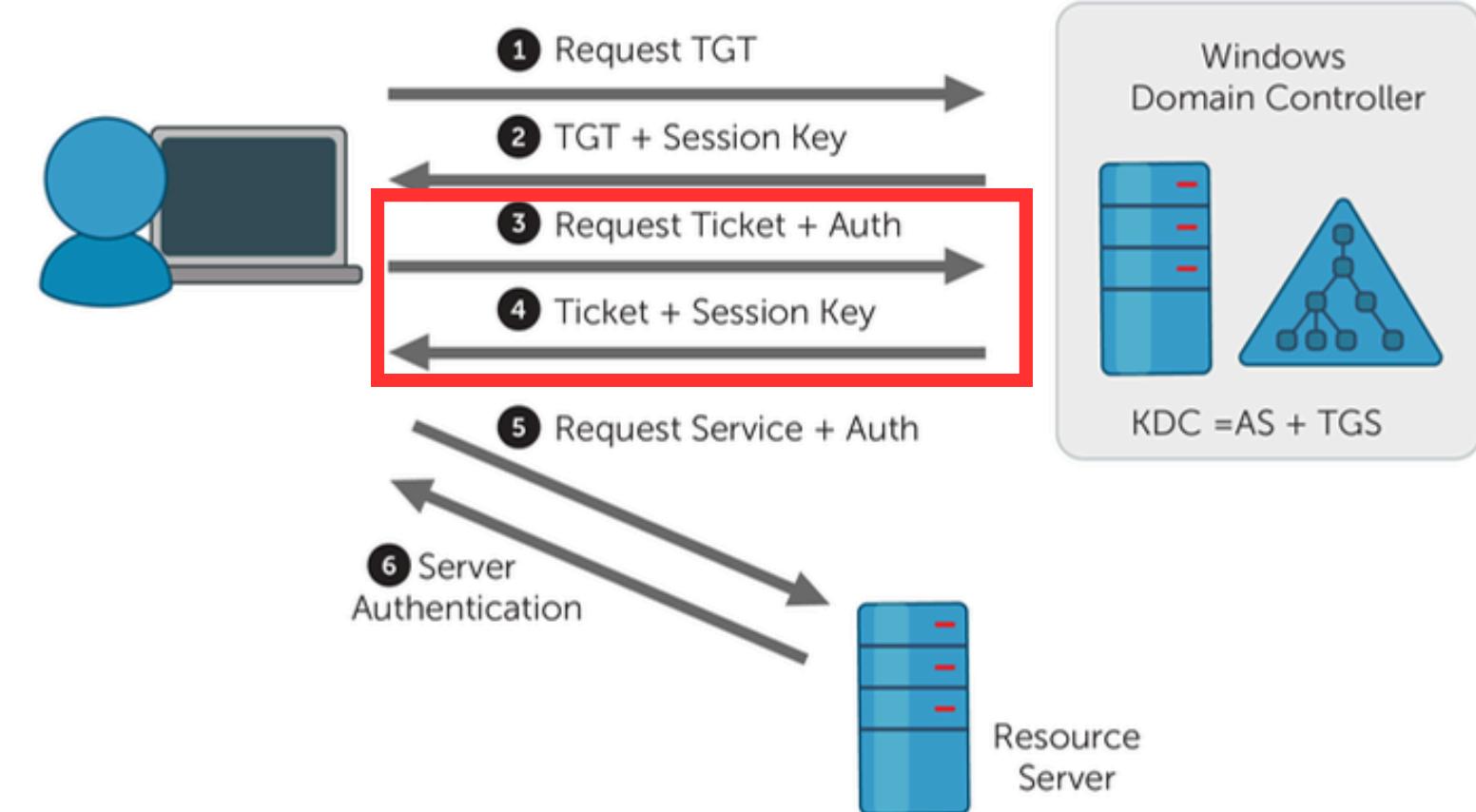
### TGS-REP structure

- **Session key B**
- **TGS ticket** (encrypt with service key)
  - user information
  - Session key B
- SPN of service

# Kerberoasting

## Attack flow of kerberoasting

1. Attacker find target account
2. Attacker get TGS of target from TGS-REP
3. Attacker crack the TGS with password cracking tools

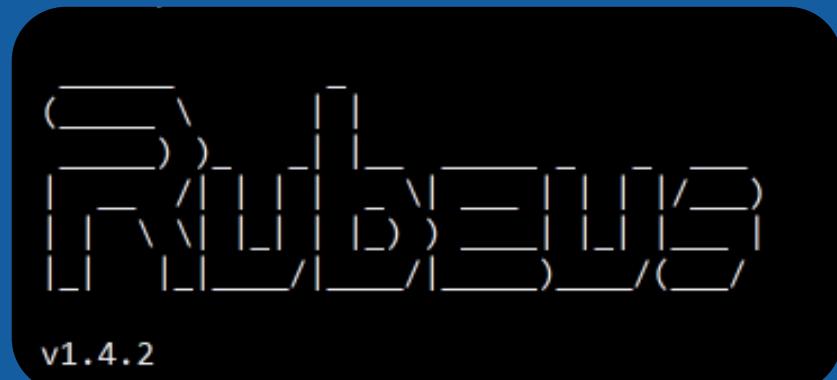


# Kerberoasting

## Condition

- Weak password service account
- Valid domain joined user

## Window Tools



## Tools for enumerate

- PowerView, Rubeus (For window)
- GetNPUsers.py (For linux)
- Hashcat

## Impact

- Gain user password

## Linux Tools

**fortra/impacket**

Impacket is a collection of Python classes for working with network protocols.



## Hash cracking Tools



# Kerberoasting

## Mitigations

- Use strong password for service account
- Try to use **machine account** for manage services instead of use user account if possible

---

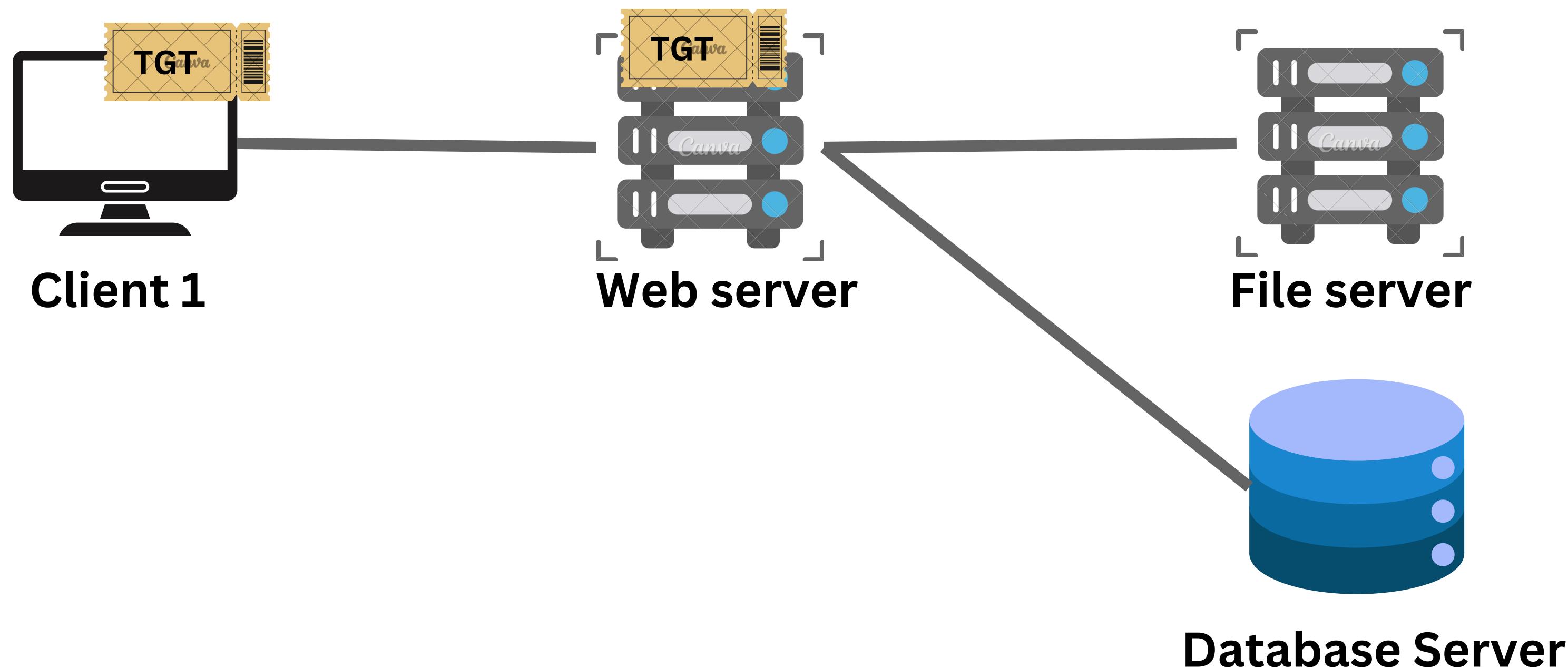
# Delegation Attacks

---

# Unconstrained Delegation

## What is Unconstrained Delegation

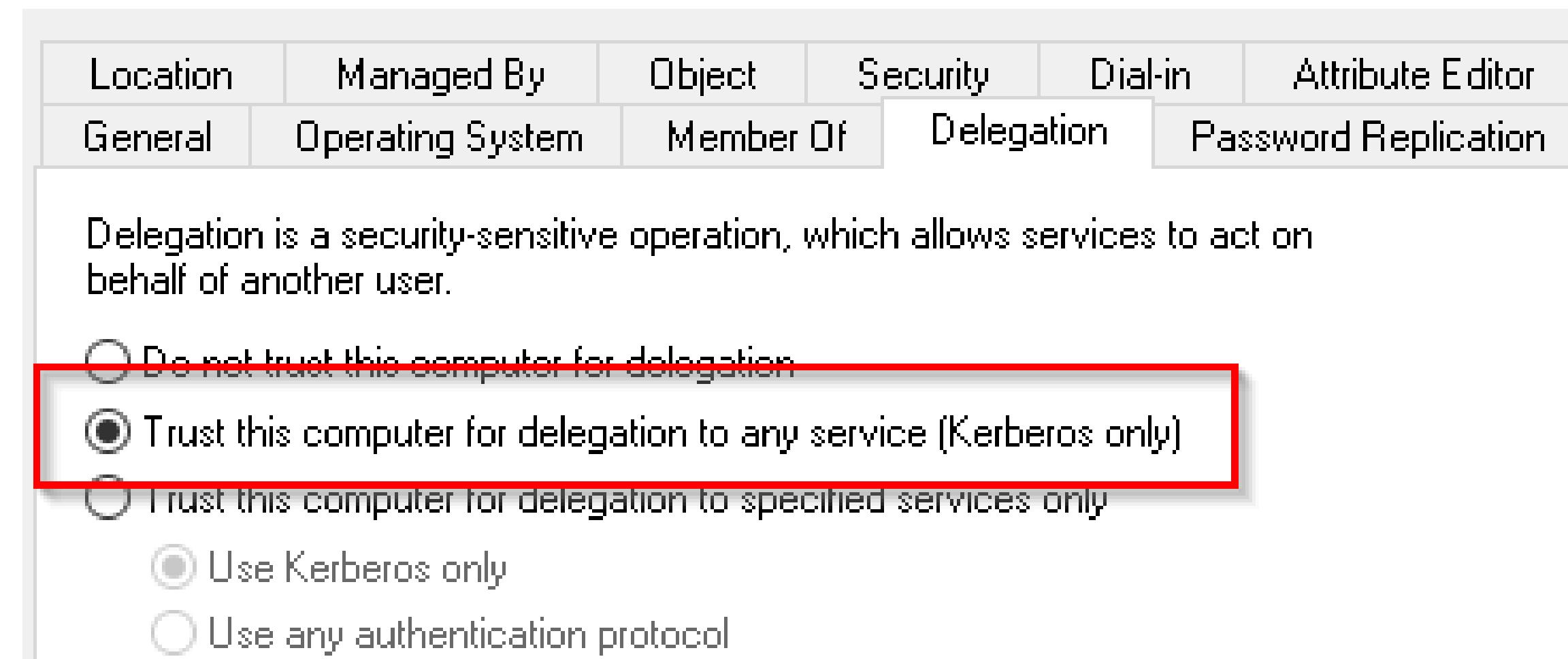
- An attack that aim steal TGT ticket that forwarded to service machine



# Unconstrained Delegation

## What is Unconstrained Delegation

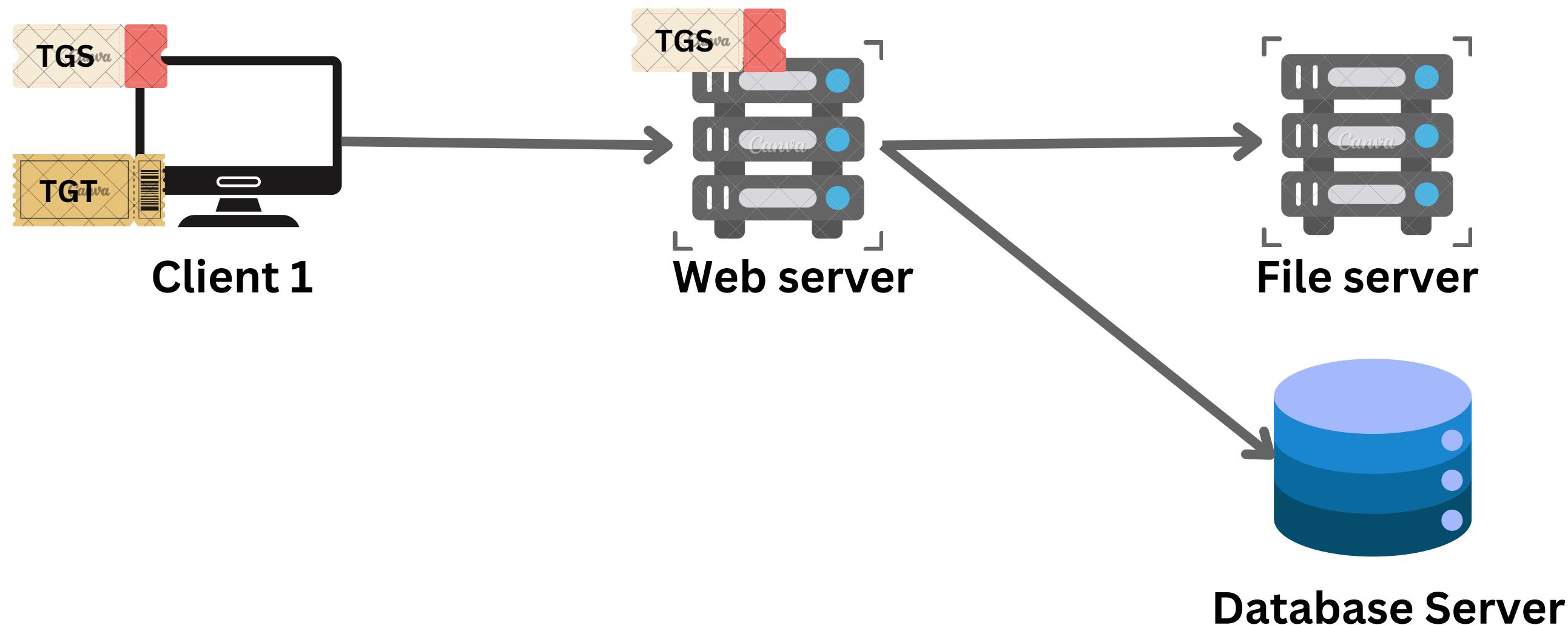
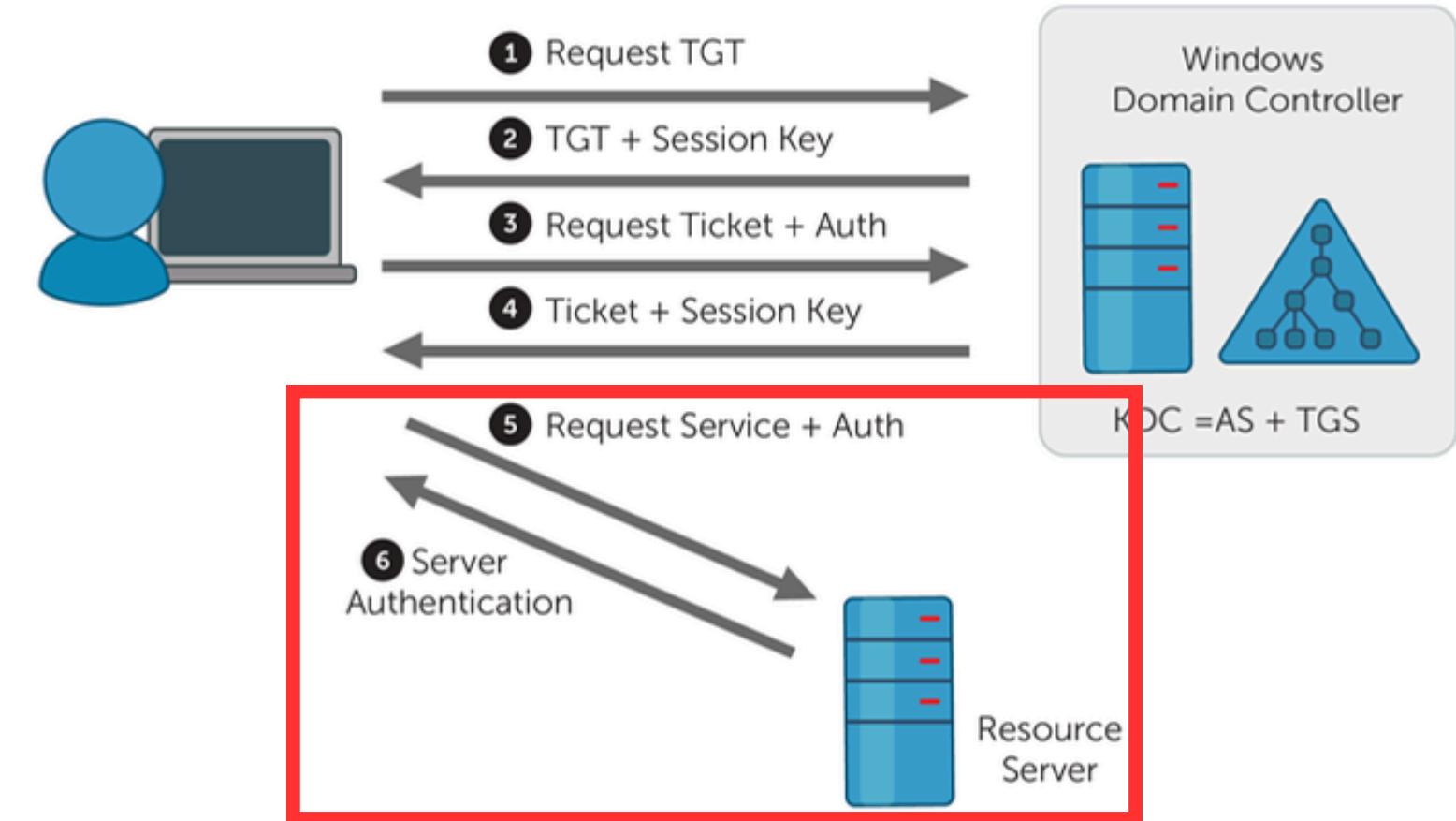
- An attack that aim steal TGT ticket that forwarded to service machine



# Unconstrained Delegation

Use service on non-constrain delegation

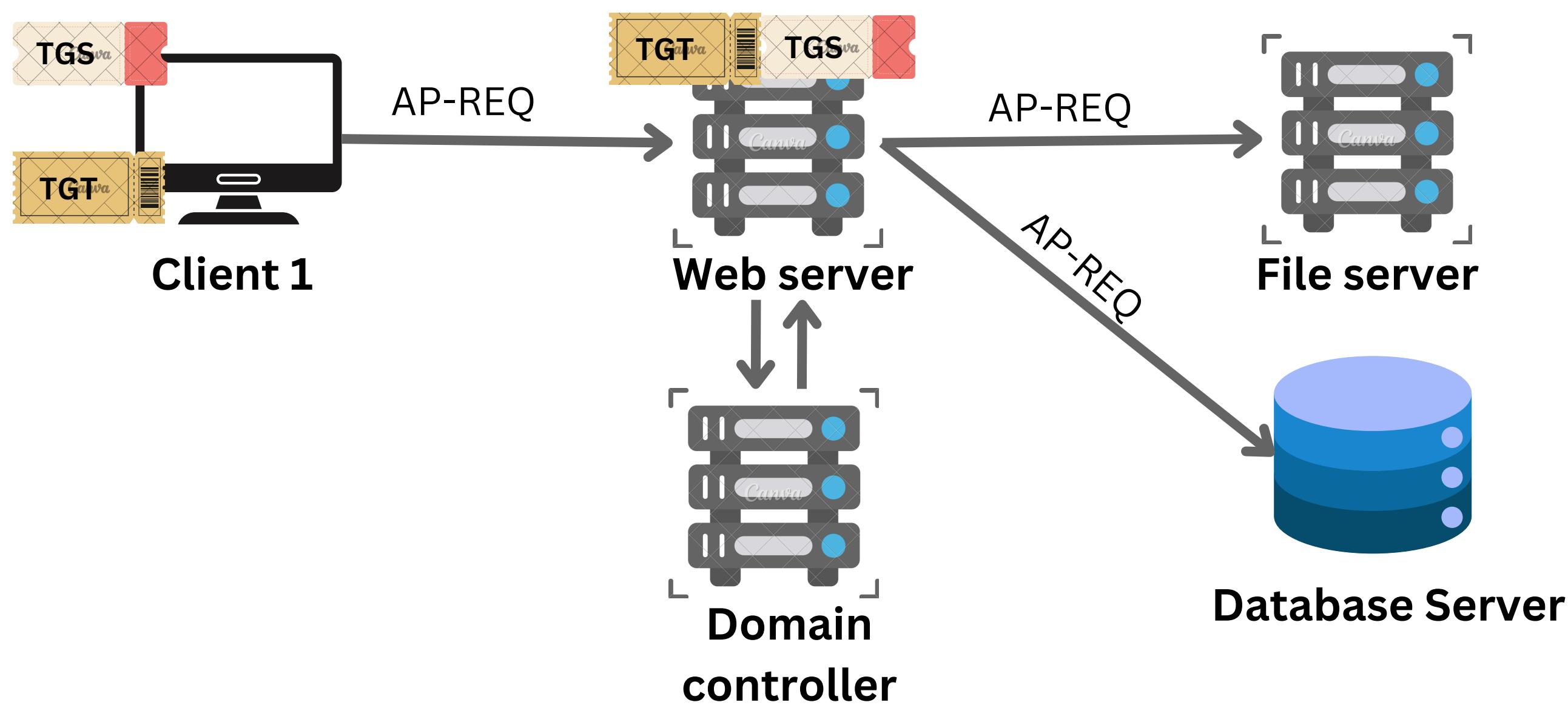
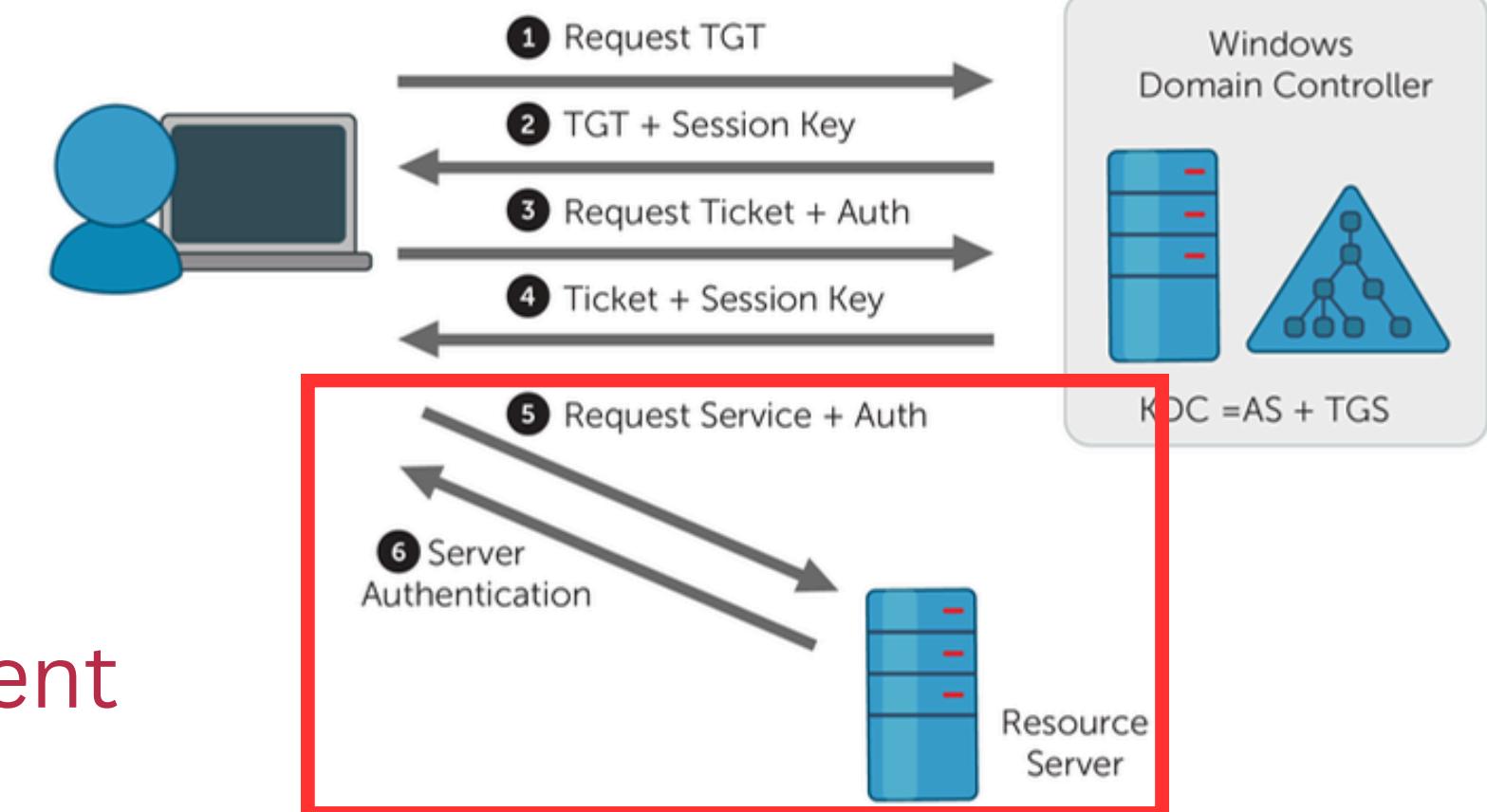
- Client sent TGS for request service



# Unconstrained Delegation

## Use service on unconstrained delegation

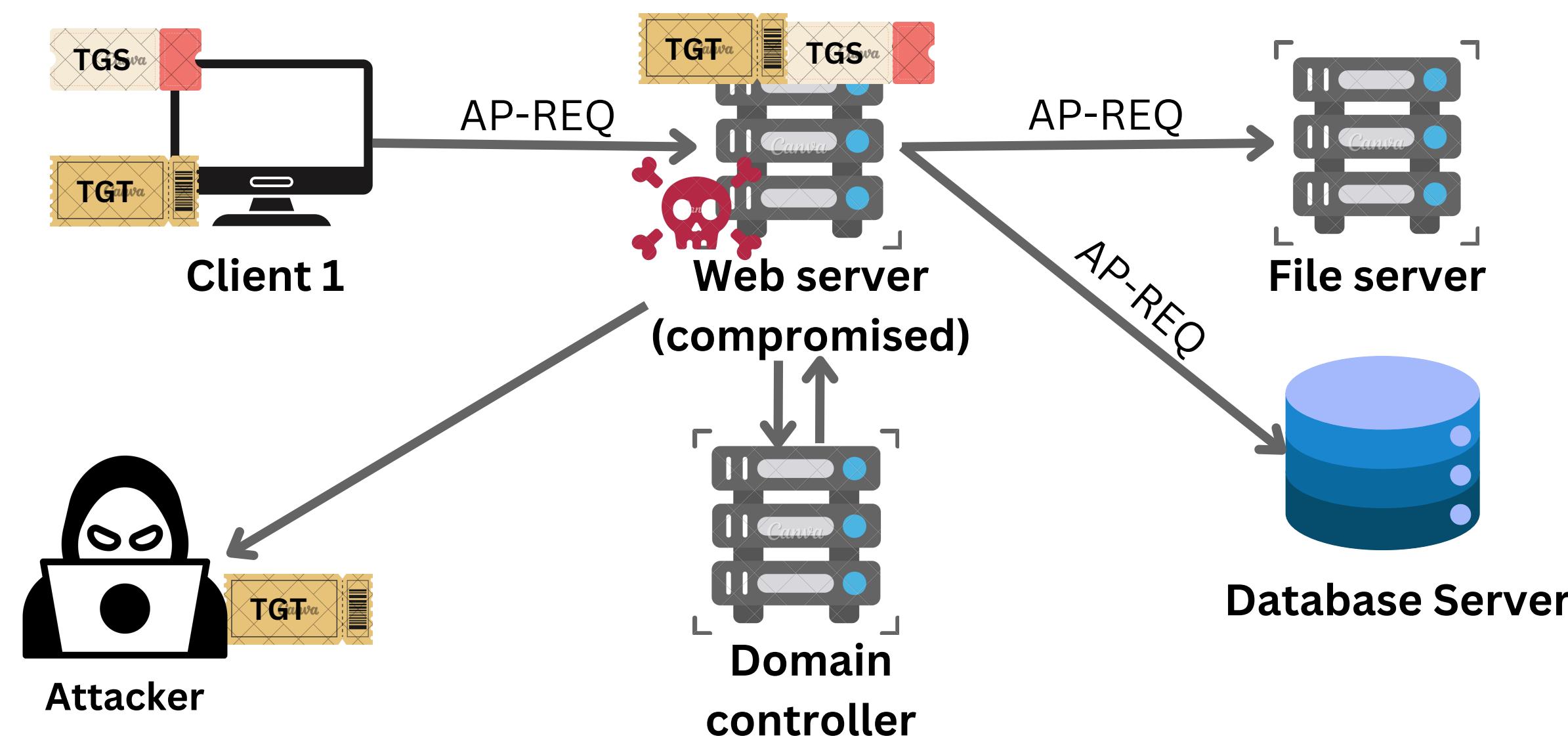
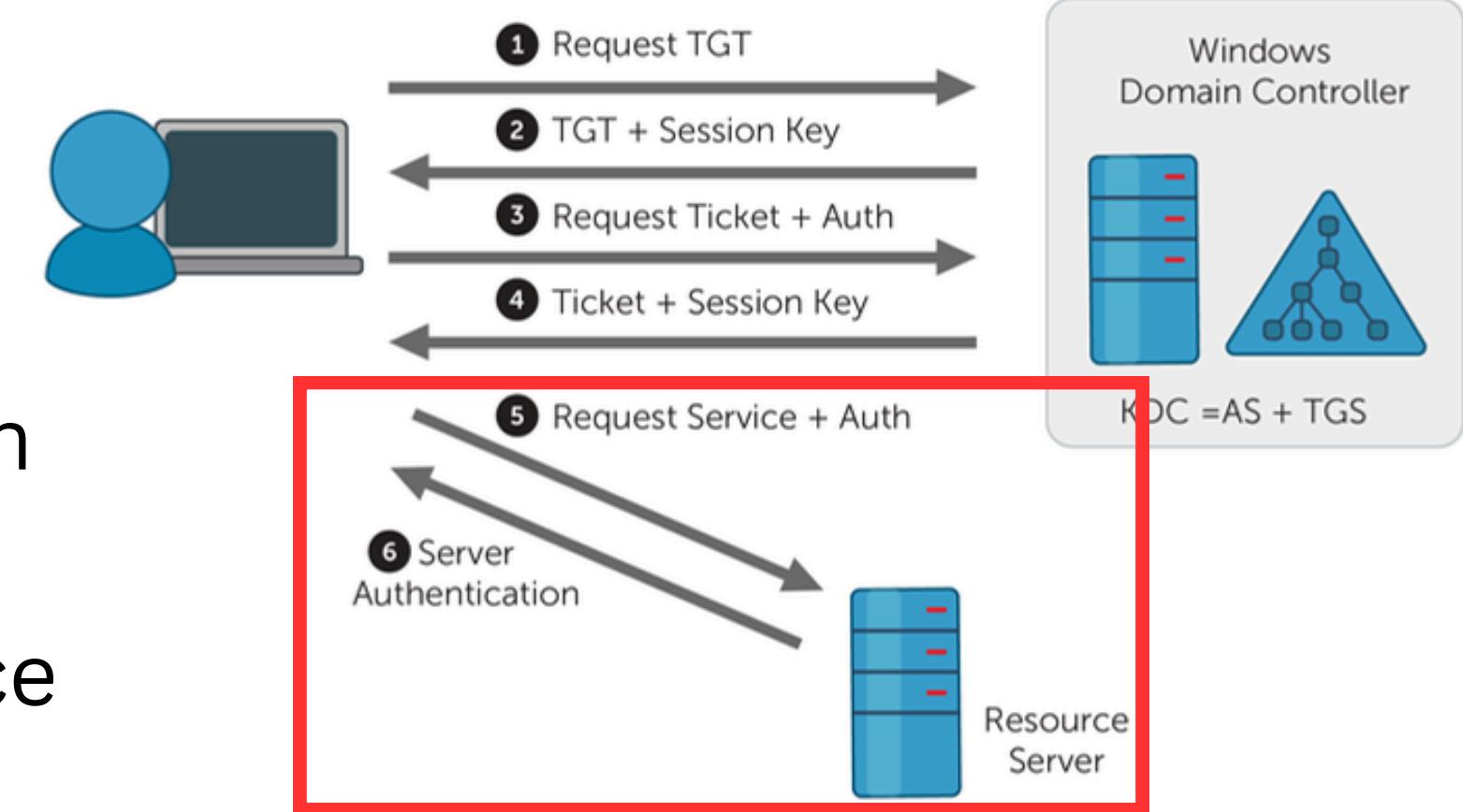
- Client sent TGS for request service
- And client also sent a TGT to service for request access to another service as a client



# Unconstrained Delegation

## Attack flow for unconstrained delegation

- Attacker compromised service that allow Unconstraint Delegation
- Attacker can **dump TGT ticket** from service machine memory



# Unconstraint Delegation

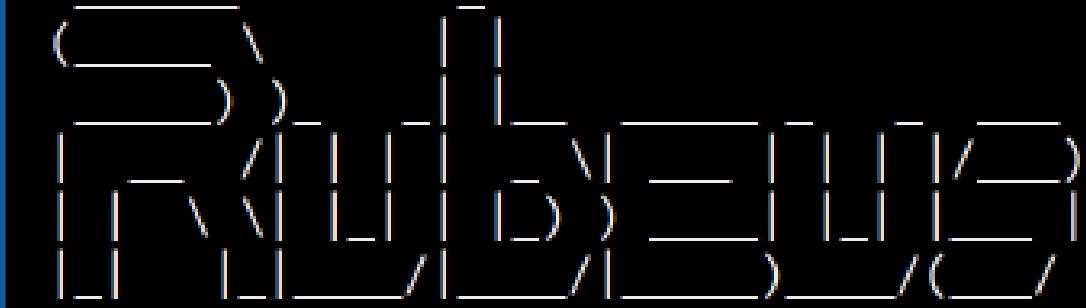
## Condition

- Compromise machine that allow unconstrained delegation

## Tools for enumerate

- Rubeus

## Window Tools



v1.4.2

## Impact

- Allow attacker to impersonate as user that use compromised machine service

# **Unconstraint Delegation**

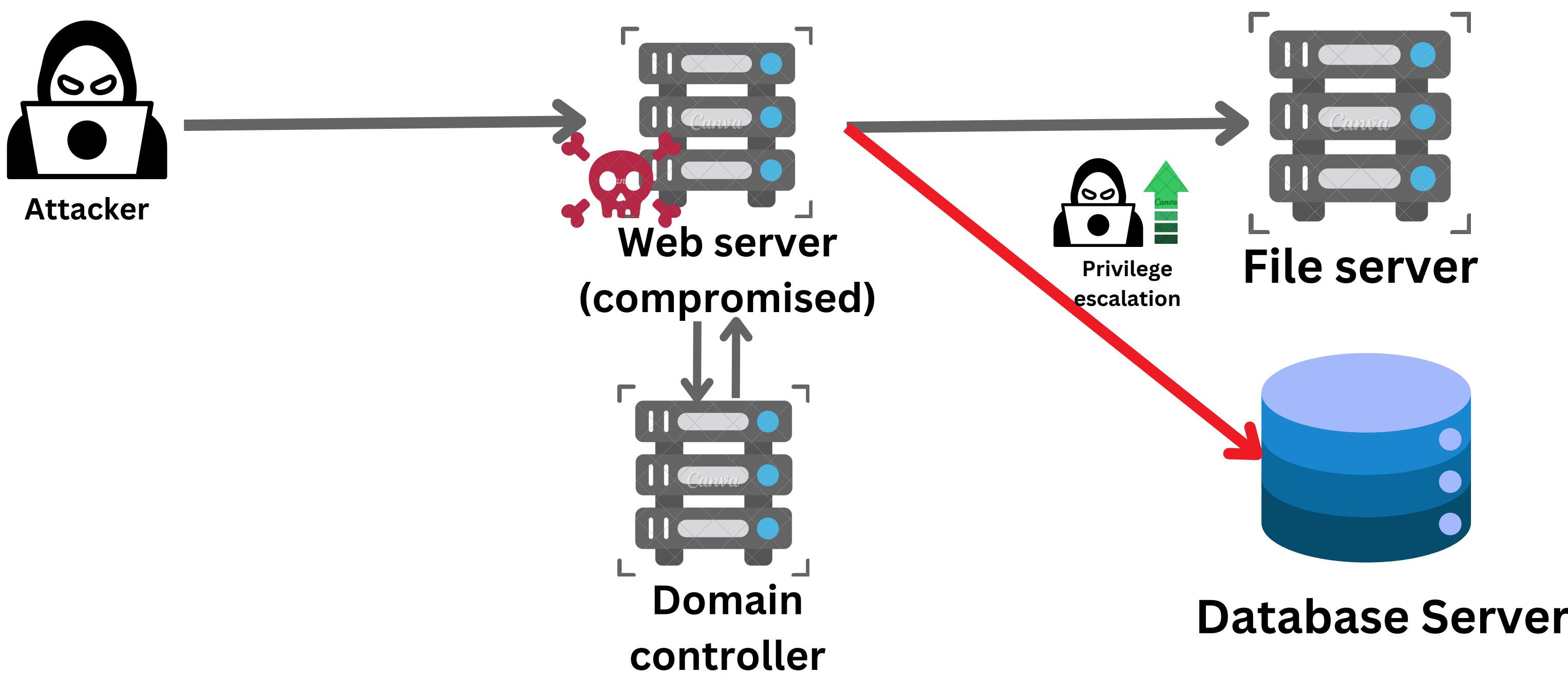
## **Mitigations**

- Disable Unconstrained delegation If Possible
- Monitor and Detect Delegation Misuse

# Constrained Delegation

## What is Constrained Delegation

- An attack that aim to compromise **constrained delegation service account** to perform privilege escalation and lateral movement



# Constrained Delegation

## What is Constrain Delegation

- An attack that aim to compromise **constrained delegation service account** to perform privilege escalation and lateral movement

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

- Do not trust this computer for delegation
- Trust this computer for delegation to any service (Kerberos only)
- Trust this computer for delegation to specified services only
  - Use Kerberos only
  - Use any authentication protocol

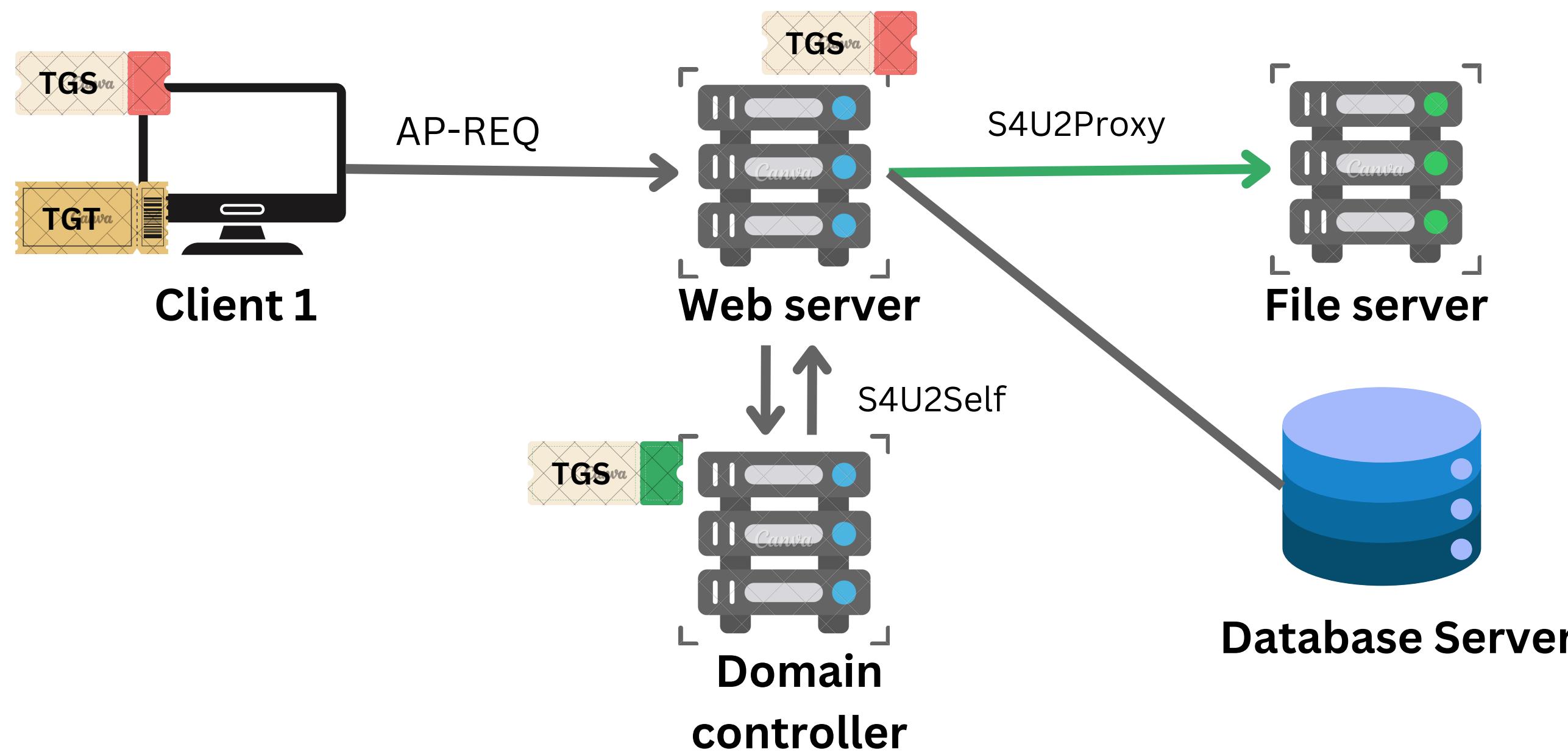
Services to which this account can present delegated credentials:

Service Type	User or Computer	Port	Service N
SQL	DBSRV		

# Constrained Delegation

## Normal flow of constrained delegation

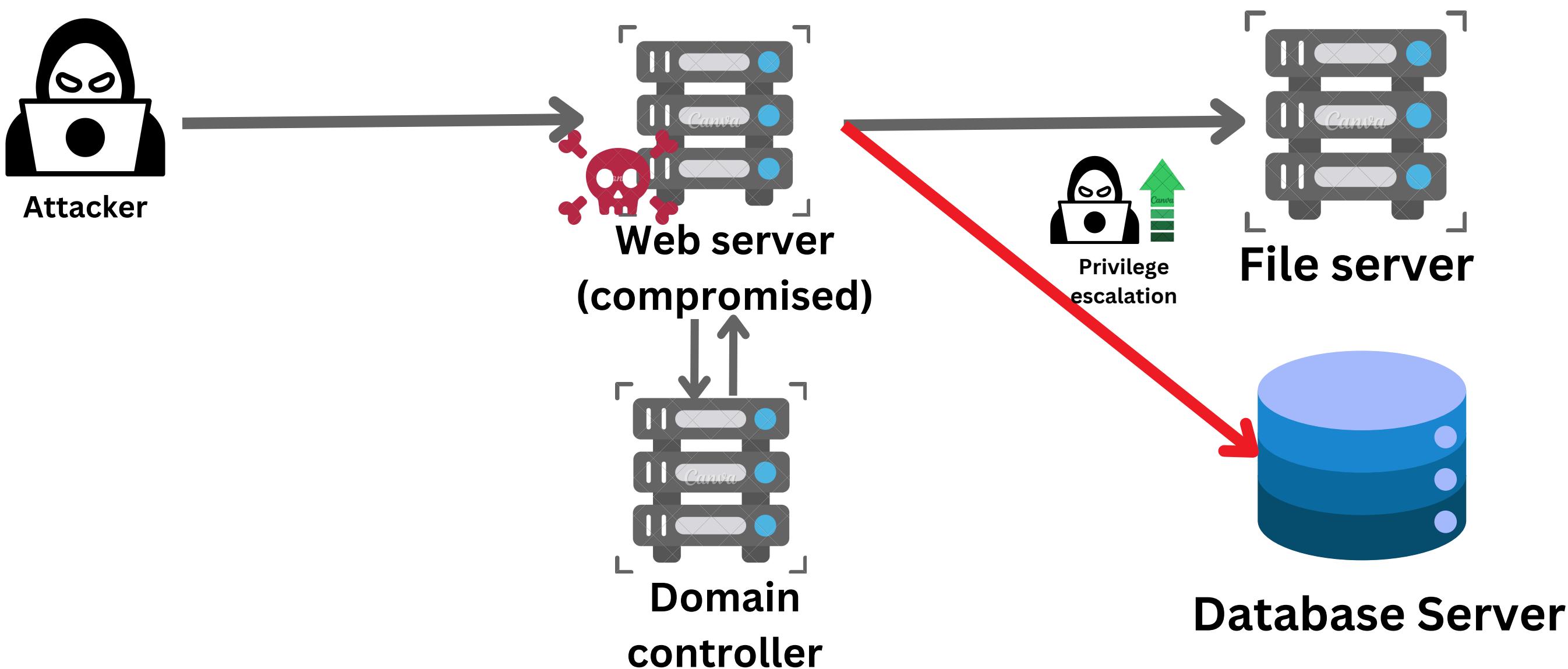
- Client sent AP-REQ to service
- Service use **S4U2Self** to request new TGS to access another service
- Service account use **S4U2Proxy** to access another service as client



# Constrained Delegation

## Attack flow for constrained delegation

- Find constrained delegation service account
- Compromise target service account
- Impersonate Any User (only work for allow service)



# Constrained Delegation

## Condition

- Compromise machine that allow unconstrained delegation

## Tools for enumerate

- PowerView, Mimikatz, Rubeus
- findDelegation.py, getST.py, psexec.py

## Impact

- Allow attacker to impersonate as any user with specific service

## Window Tools



## Linux Tools

### fortra/impacket

Impacket is a collection of Python classes for working with network protocols.



# **Constrained Delegation**

---

## **Mitigations**

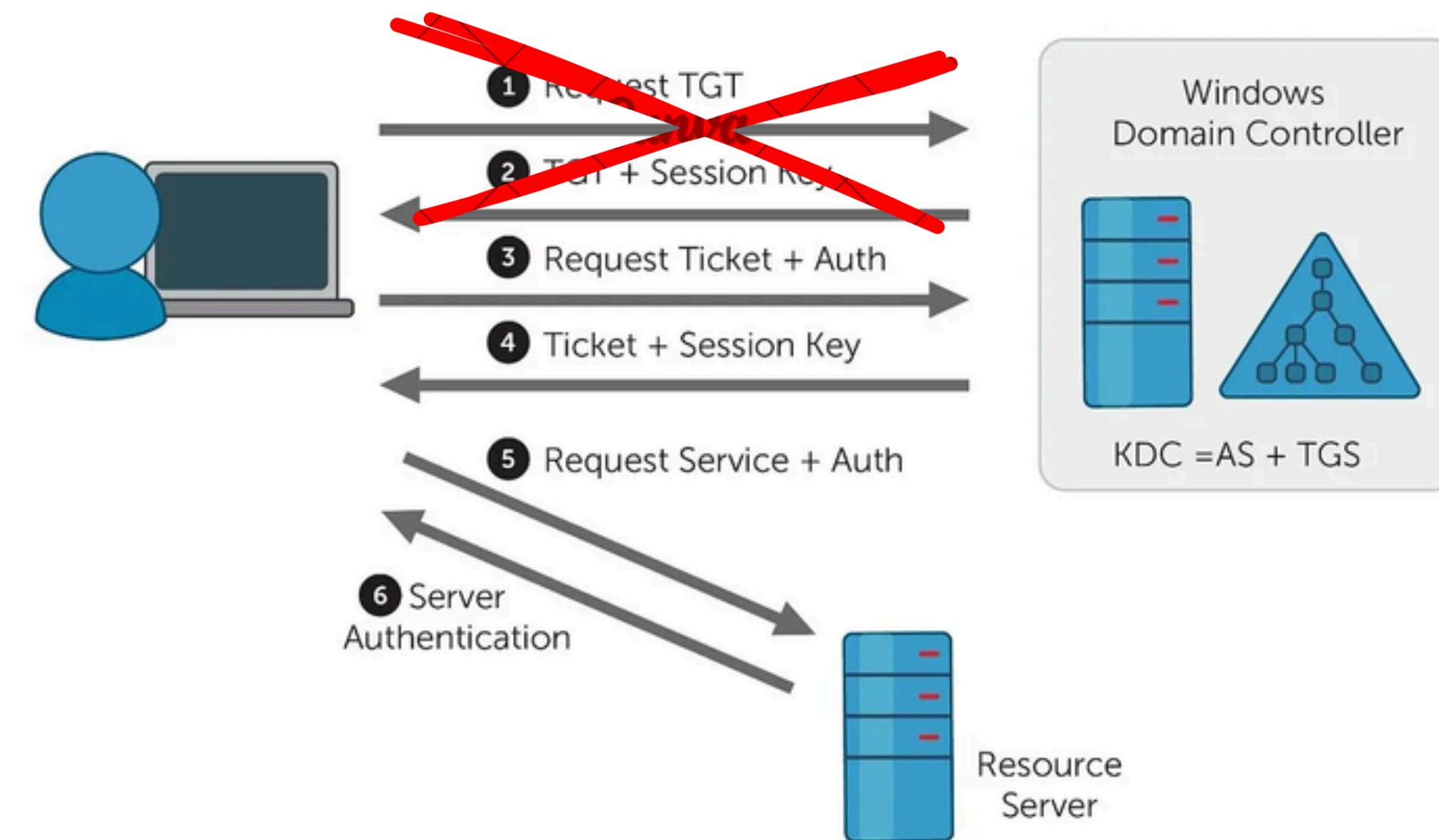
- Disable Constrained delegation If Possible
- Monitor and Detect Delegation Misuse

# **Ticket Abuse**

# Golden ticket

## What is Golden ticket

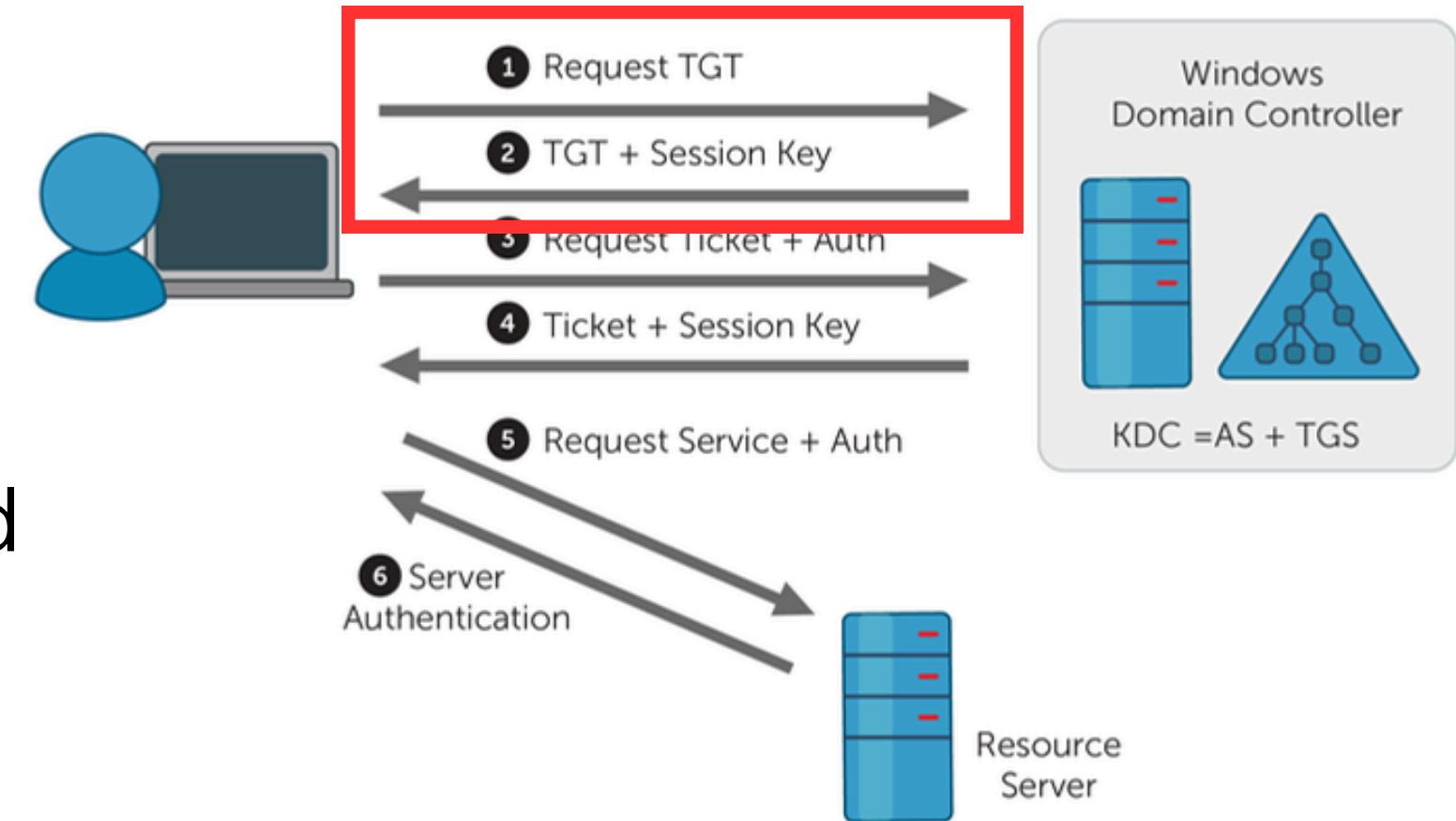
- An attack that aim to forging the TGT ticket by use privilege of **krbtgt account**



# Golden ticket

## Normal flow for get TGT ticket

1. Client sent **AS-REQ** to KDC
2. KDC validate the **AS-REQ** from client and
3. KDC use key of **krbtgt** to encrypt TGTs
4. KDC sent **AS-REP** to Client



## AS-REQ structure

- **Authenticator** (encrypt with user password)
  - timestamp
- **Username**

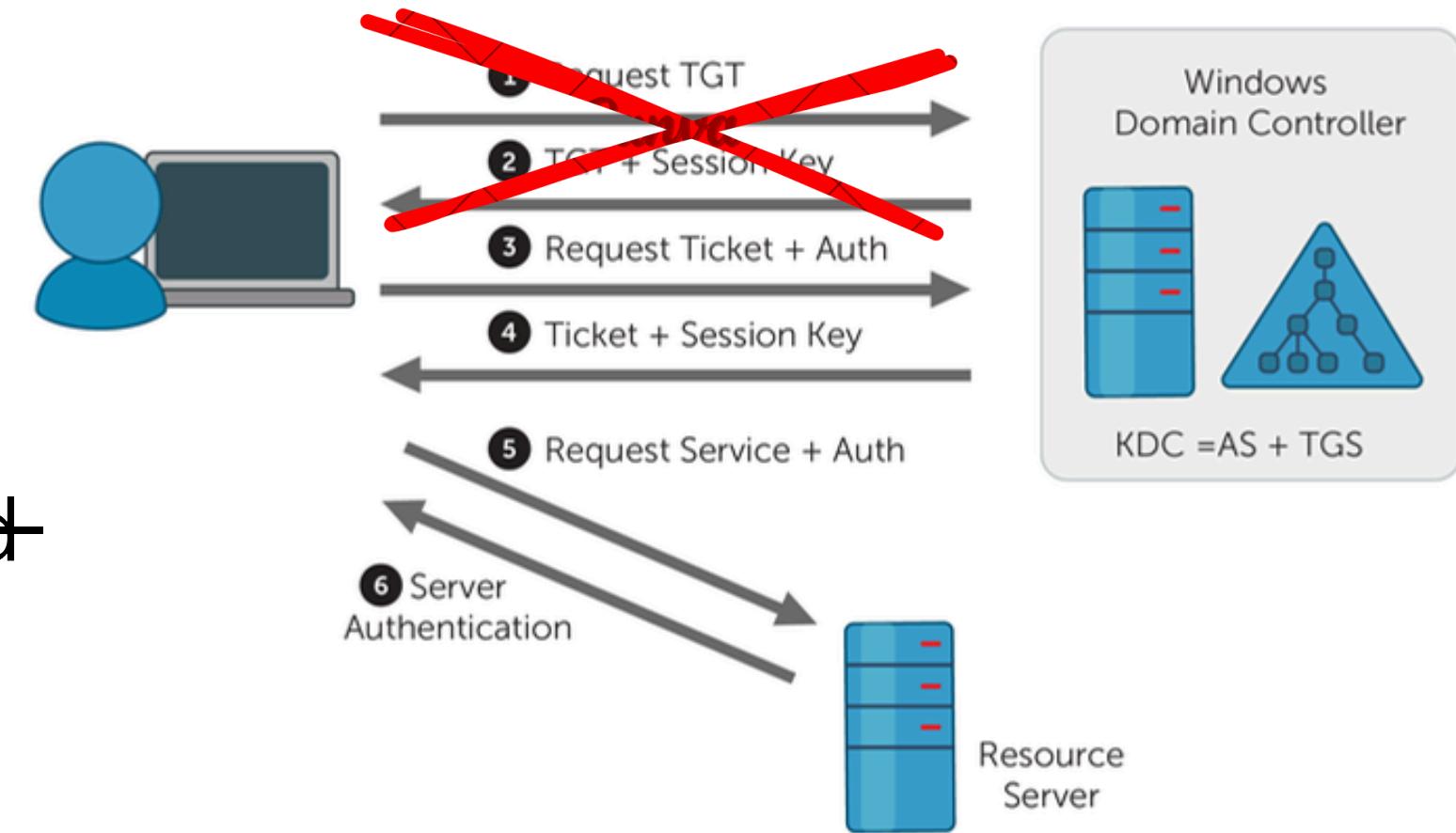
## AS-REP structure

- **Session key** (encrypt with user password)
- **TGT ticket** (encrypt with KDC key)
  - user information
  - Session key

# Golden ticket

## Goldent ticket flow for get TGT ticket

1. Client sent **AS-REQ** to **KDC**
2. **KDC** validate the **AS-REQ** from client and
3. **KDC** use key of **krbtgt** to encrypt TGTs
4. Client got **krbtgt** key
5. Client issue TGTs ticket by use **krbtgt** key to encrypt data



## Elements that require for forge golden ticket

1. Domain name
2. Domain SID
3. Username to Impersonate
4. KRBTGT's hash

# **Golden ticket**

## **Condition**

- krbtgt account
- valid domain joined user

## **Tools for enumerate**

- PowerView, mimikatz (For window)
- lookupsid.py, ticketer.py (For linux)

## **Impact**

- Can issue any TGTs in this domain
- Can be any user in this domain

## **Window Tools**



## **Linux Tools**

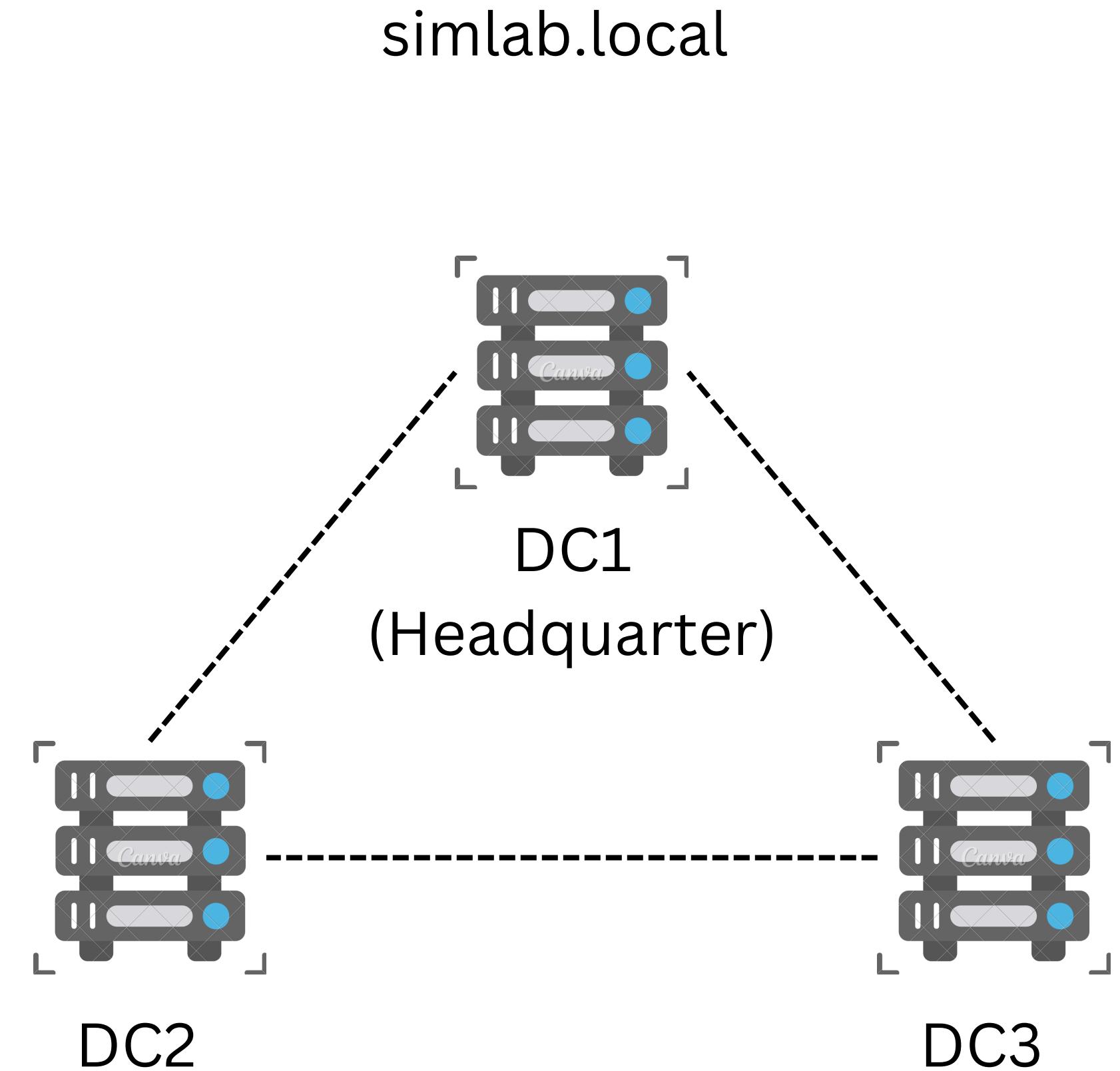
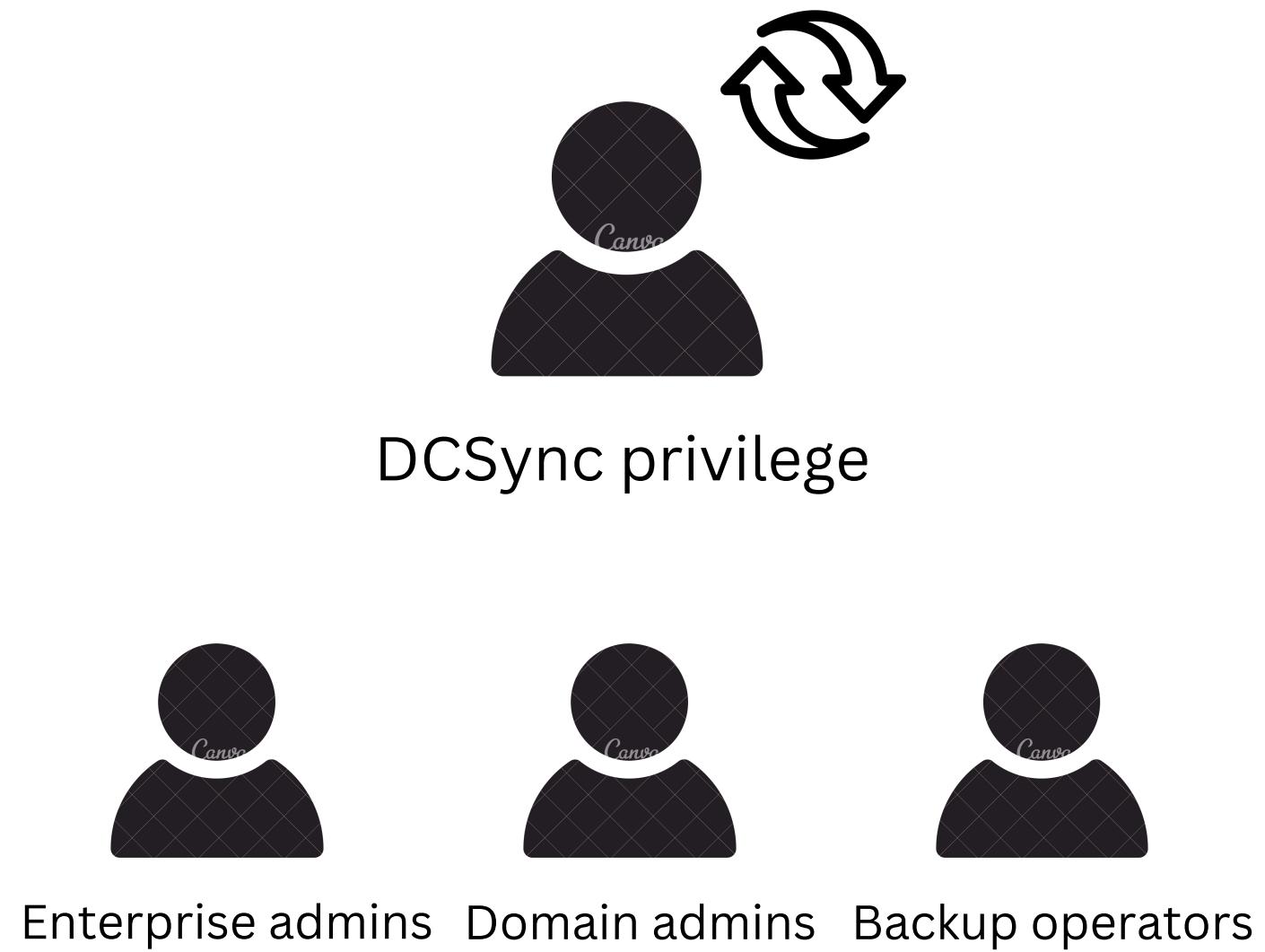
### **fortra/impacket**

Impacket is a collection of Python classes for working with network protocols.



# Golden ticket

- DCSync privilege



# Golden ticket

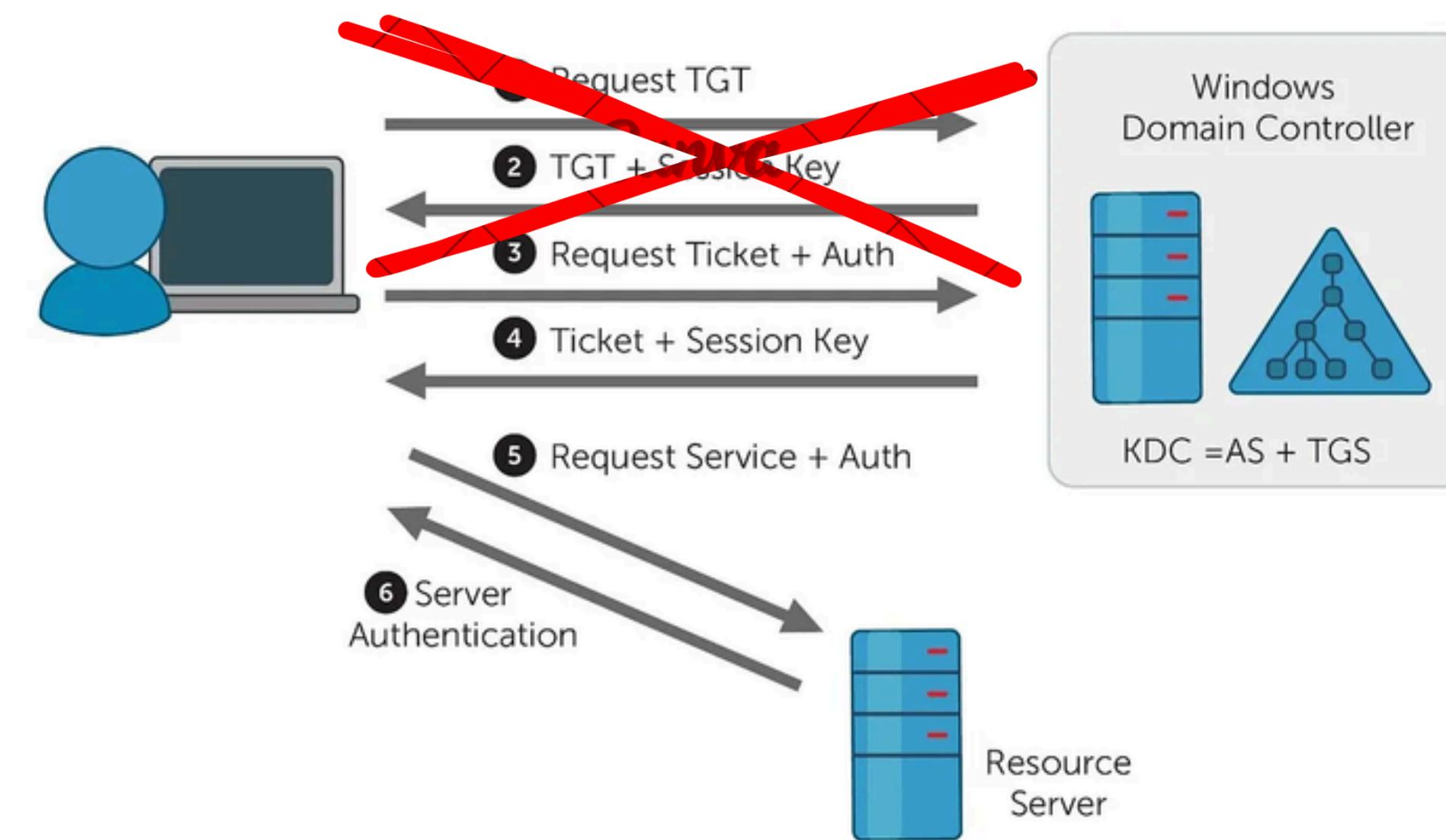
## Mitigations

- Use Endpoint Detection and antivirus for prevent and detect tools like Mimikatz
- Implement a least privilege access model

# Silver ticket

## What is Silver ticket

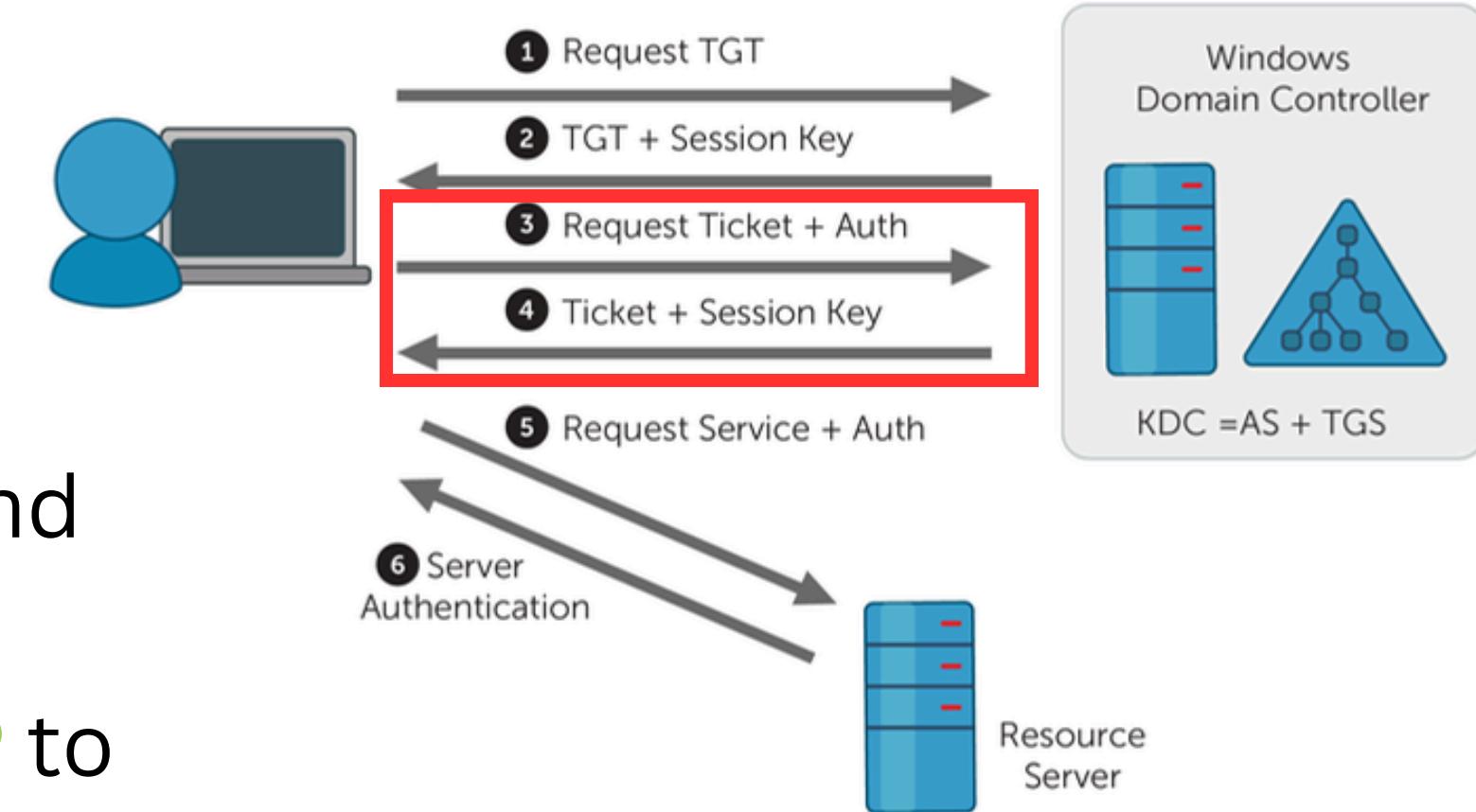
- An attack that aim to forging the TGS ticket by get NTLM hash of machine account (service account)



# Silver ticket

## Normal flow for get TGS ticket

1. Client sent **TGS-REQ** to KDC
2. KDC validate the **TGS-REQ** from client and
3. KDC encrypt TGS with service key
4. KDC issue session key and sent **TGS-REP** to Client



### TGS-REQ structure

- **Authenticator** (encrypt with session key A)
  - timestamp
- **TGT** (encrypt with KDC key)
  - user information
  - Session key A
- **Name of service that will access**

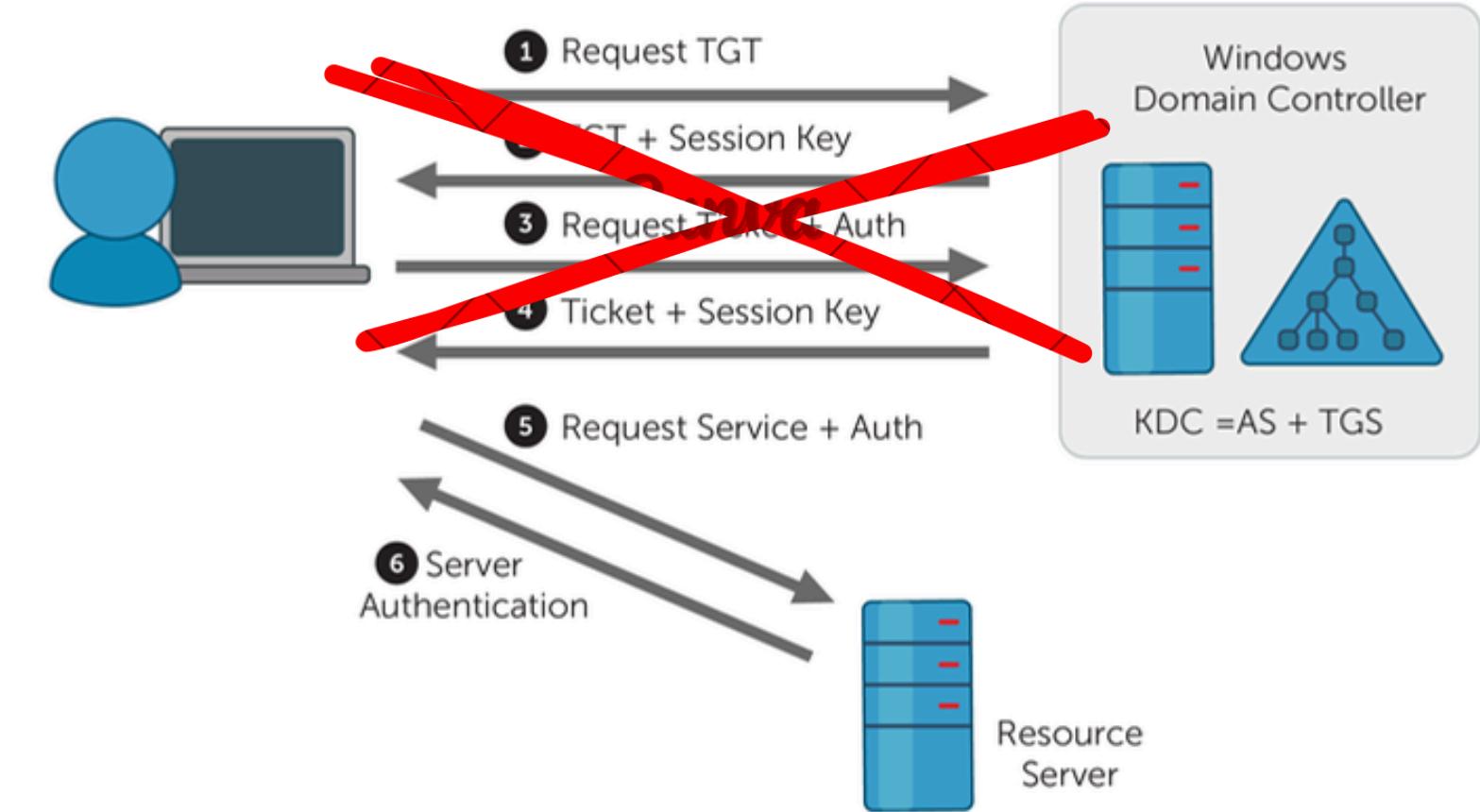
### TGS-REP structure

- **Session key B**
- **TGS ticket** (encrypt with service key)
  - user information
  - Session key B

# Silver ticket

## Silver ticket flow for get TGS ticket

1. Client sent TGS REQ to KDC
2. KDC validate the TGS REQ from client and
3. KDC encrypt TGS with service key
4. KDC issue session key and sent TGS REP to Client
5. Compromise service account and get NTLM hash
6. Issue TGS ticket



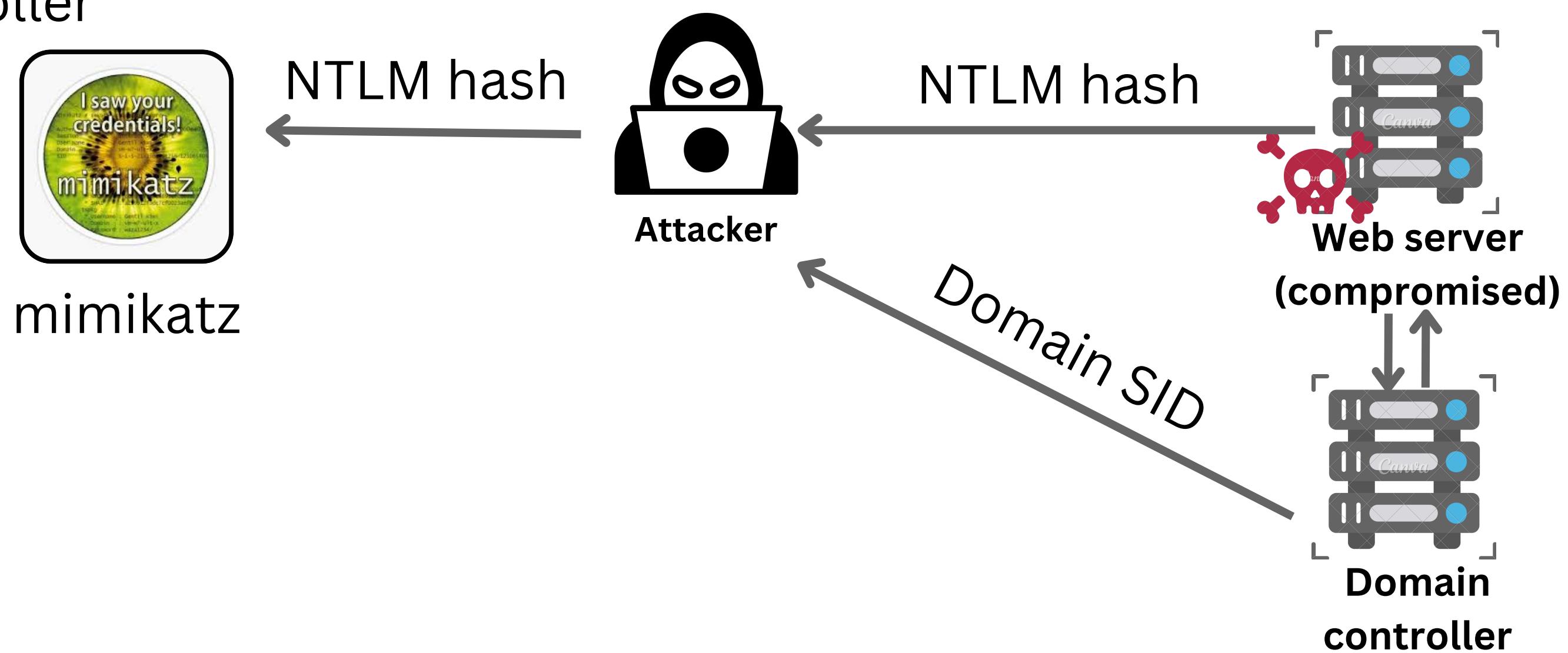
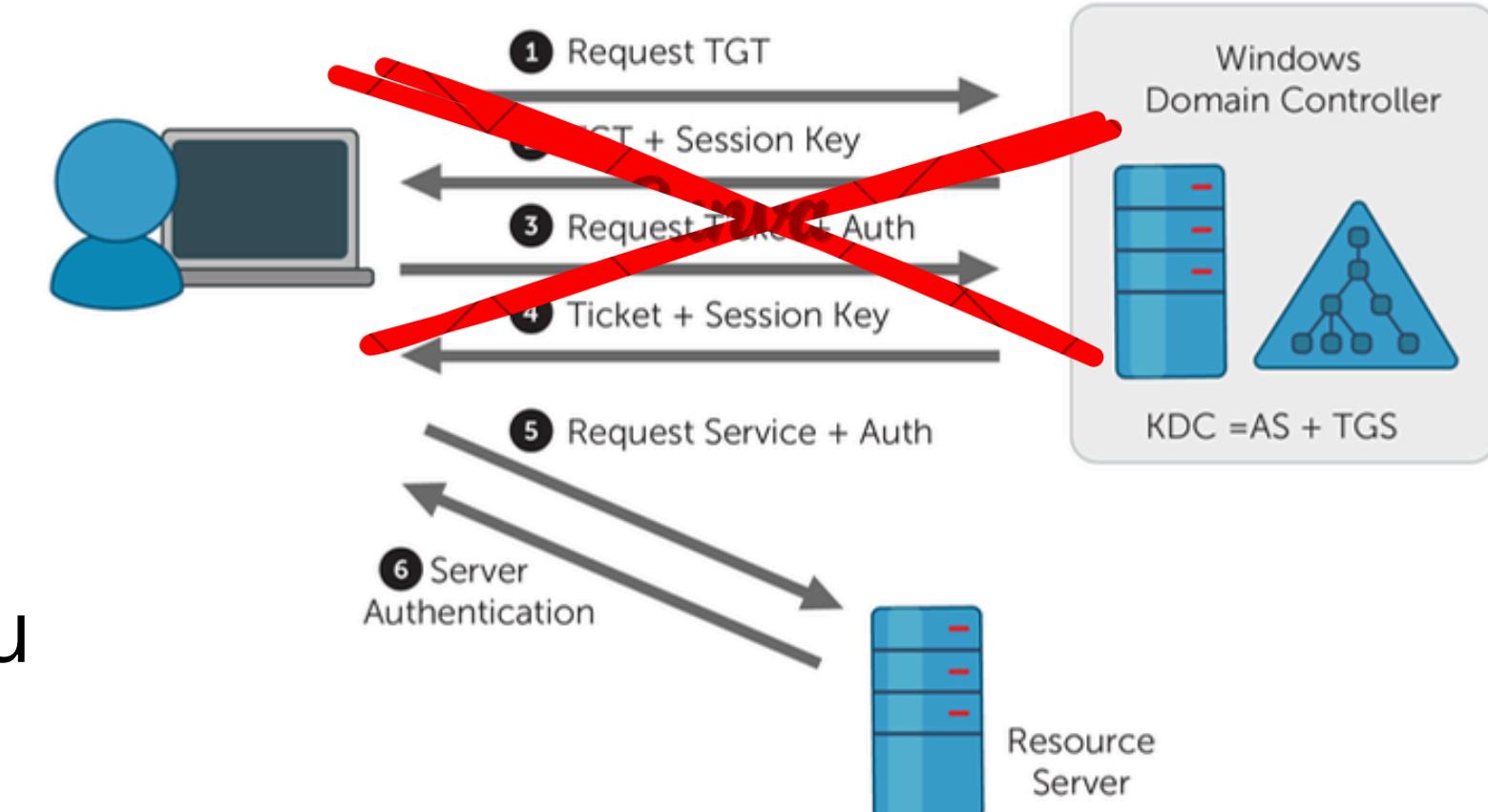
## Elements that require for forge silver ticket

1. Domain name
2. Domain SID
3. NTLM hash (service key)
4. Target service

# Silver ticket

## Attack flow for forging silver ticket

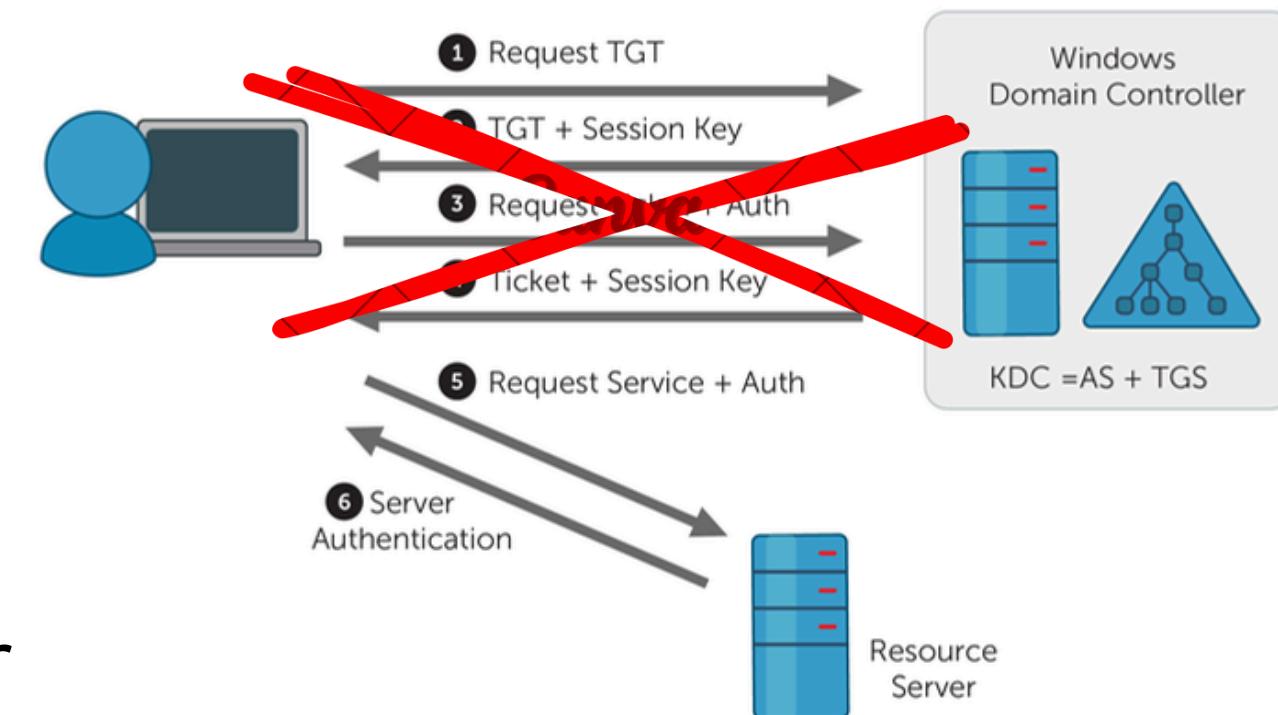
- Attacker compromised service service account
- Attacker Get NTLM hash of service accou
- Attacker Get Domain SID from Domain controller



# Silver ticket

## Attack flow for forging silver ticket

- Attacker compromised service service account
- Attacker Get NTLM hash of service account
- Attacker Get Domain SID from Domain controller



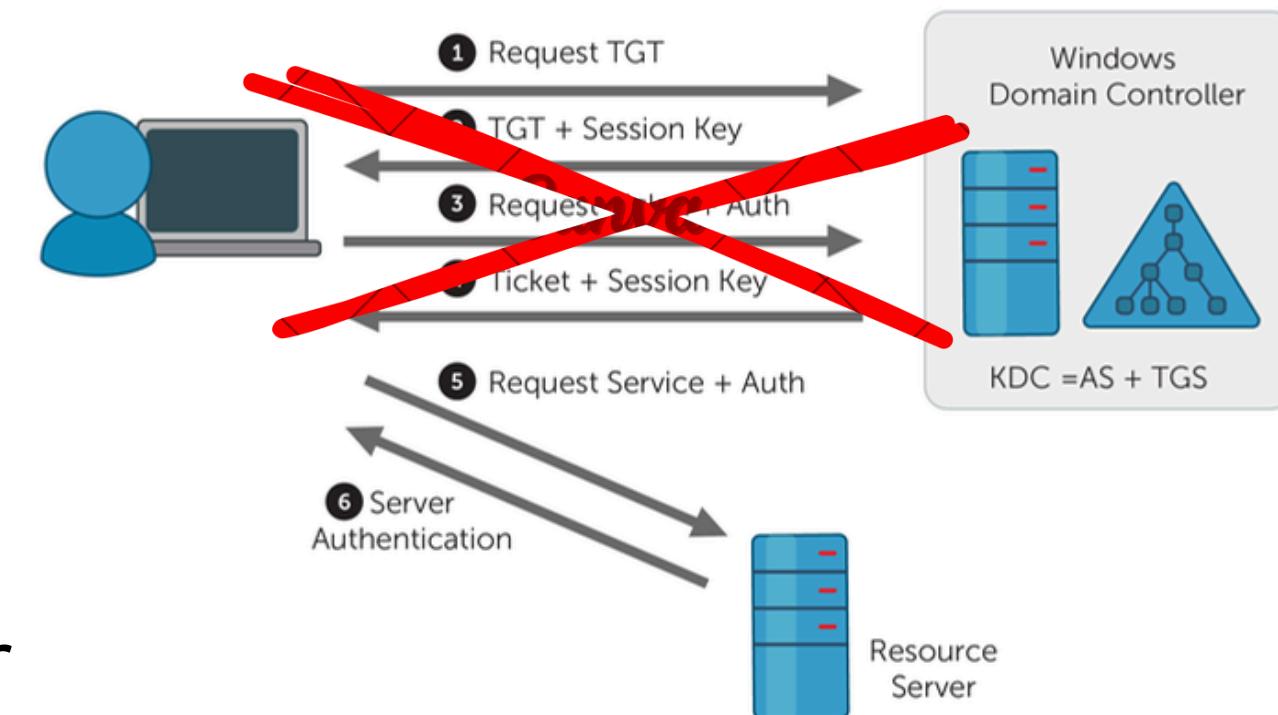
```
PS C:\Tools> mimikatz.exe
mimikatz # kerberos::golden /domain:inlanefreight.local /user:Administrator /sid:S-1-5-21-2974783224-3764228556-2640795941
/rc4:ff955e93a130f5bb1a6565f32b7dc127 /target:sql01.inlanefreight.local /service:cifs /ptt
User      : Administrator
Domain    : inlanefreight.local (INLANEFREIGHT)
SID       : S-1-5-21-2974783224-3764228556-2640795941
User Id   : 500
Groups Id : *513 512 520 518 519
ServiceKey: ff955e93a130f5bb1a6565f32b7dc127 - rc4_hmac_nt
Service   : cifs
Target    : sql01.inlanefreight.local
Lifetime  : 8/17/2020 3:22:27 PM ; 8/15/2030 3:22:27 PM ; 8/15/2030 3:22:27 PM
-> Ticket : ` Pass The Ticket `
* PAC generated
* PAC signed
* EncTicketPart generated
* EncTicketPart encrypted
* KrbCred generated

Golden ticket for 'Administrator @ inlanefreight.local' successfully submitted for current session
```

# Silver ticket

## Attack flow for forging silver ticket

- Attacker compromised service service account
- Attacker Get NTLM hash of service account
- Attacker Get Domain SID from Domain controller



```
PS C:\Tools> klist
Current LogonId is 0x0x3f22d82
Cached Tickets: (1)

#0> Client: Administrator @ inlanefreight.local
    Server: cifs/sql01.inlanefreight.local @ inlanefreight.local
    Kerbticket Encryption type: RSADSI RC4-HMAC(NT)
    Ticket Flags 0x40a00000 -> forwardable renewable pre_authent
    Start Time: 8/17/2020 15:22:27 (local)
    End Time: 8/15/2030 15:22:27 (local)
    Renew Time: 8/15/2030 15:22:27 (local)
    Session Key Type: RSADSI RC4-HMAC(NT)
    Cache Flags: 0
    Kdc Called:
```

# Silver ticket

## Condition

- compromised service account
- valid domain joined user

## Tools for enumerate

- PowerView, mimikatz, rubeus (For window)
- lookupsid.py, ticketer.py (For linux)



## Impact

- Can direct access to compromised service without exist log on Domain controller

## Linux Tools

**fortra/impacket**

Impacket is a collection of Python classes for working with network protocols.



# **Silver ticket**

---

## **Mitigations**

- Do not place service accounts within privileged groups like domain administrators
- Use strong password for service accounts
- Utilize Managed Service Accounts and ensure passwords rotate regularly

# Q&A

---

**Thank you**

---