# Kerberos

Martin Stanek

2025

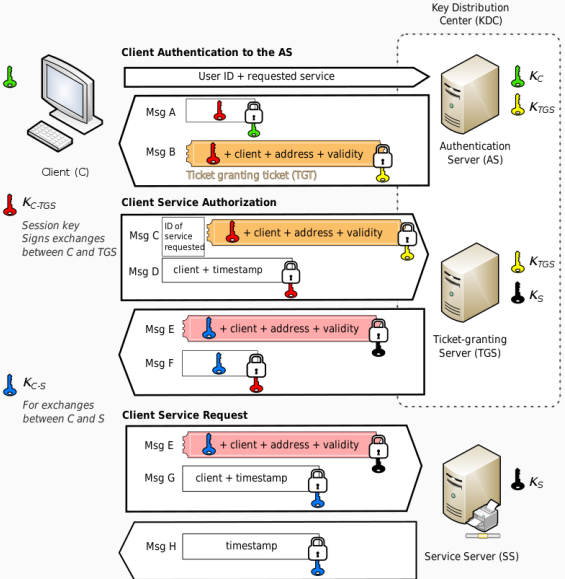
# Introduction to Kerberos

- What is Kerberos:
  - authentication protocol
  - uses symmetric key cryptography
- Why Kerberos:
  - mutual authentication using trusted third party
  - passwords are not send over the network
  - single point of defining and enforcing access/policies
  - security, when properly implemented, configured and operated
- Potential problems:
  - time synchronization
  - single point of failure
  - symmetric keys required for each principal/service

## Kerberos Architecture – key components

- KDC (Key Distribution Center) – Domain controller in Active Directory
  - AS (Authentication Server/Service)
  - TGS (Ticket-granting Server/Service)
- client
- service server
- password transformed into a symmetric key (NT hash)
  - client's symmetric key $K_C$
  - KDC's symmetric key $K_{\mathtt{TGT}}$, derived from `krbtgt` password
- ticket – a claim to a service

# Kerberos Architecture – diagram




Jeran Renz
(https://commons.wikimedia.org/wiki/File:Kerberos_protocol.svg),
https://creativecommons.org/licenses/by-sa/4.0/legalcode

# Message flow 1: AS-REQ + AS-REP

- client sends an Authentication Service Request (AS-REQ) to the KDC
  - pre-authentication data: current time encrypted with $K_C$
  - the client's identity (usually the username)
  - the name of the target service (`krbtgt`)
  - *client authenticated to the KDC*
- the KDC responds with an Authentication Service Reply (AS-REP)
  - session key $K_{C,\mathrm{TGT}}$ and some metadata encrypted with $K_C$
    - *KDC authenticated to the client*
  - Ticket Granting Ticket (TGT) encrypted with $K_{\mathrm{TGT}}$
    - $K_{C,\mathrm{TGT}}$, username, address, validity, group membership, SID, etc.
- client decrypts the session key using $K_C$, and stores TGT

## Client wants to access a service

- services have their own accounts in the domain
- Service Principal Name (SPN)
  - MSSQLSvc/sqlsrvr.contoso.com:1433: SQL Server on a specific host and port
  - HTTP/www.example.com: HTTP service on a specified web server
  - HOST/hostname.domain.com: generic SPN for host services on a specific machine

# Message flow 2: TGS-REQ + TGS-REP

- client sends a Ticket-Granting Service Request (TGS-REQ) to the KDC
  - the TGT, the name of the target service, and an authenticator
  - authenticator: client's identity and timestamp, encrypted with $K_{C,\mathrm{TGT}}$
    - *client proves the knowledge of the session key*
- KDC decrypts and verifies the TGT, and the authenticator
- KDC responds with a Ticket-Granting Service Response (TGS-REP)
  - session key $K_{C,S}$ for the client and the service.
  - Service Ticket (ST) for the requested service, encrypted with $K_S$
    - $K_S$ is the symmetric key of the service
    - ST: $K_{S,\mathrm{TGT}}$, username, service name, validity, group membership, etc.

## Message flow 3: AP-REQ + AP-REP

- client sends an Application Request (AP-REQ) to the service
  - ST and an authenticator
  - authenticator: client's identity and timestamp, encrypted with $K_{C,S}$
  - *client proves the knowledge of the session key*
- the service decrypts and verifies the ST, and the authenticator
  - the service decides to grant access or nots
- the service responds with Application Response (AP-REP)
  - timestamp encrypted with $K_{C,S}$
  - *the service proves the knowledge of the session key*
- subsequent communication between the client and the service can use $K_{C,S}$

# AS-REP Roasting

- assumption:
  - (1) account does not require pre-authentication (misconfiguration), or
  - (2) attacker is able to catch an AS-REP message, MITM
- KDC sends back an AS-REP for that user (1)
- AS-REP contains session key and some metadata encrypted with $K_C$
- off-line dictionary or brute-force attack possible

# AS-REQ Roasting

- assumption: attacker in the MITM position, gets AS-REQ
- timestamp encrypted with $K_C$
- off-line dictionary or brute-force attack possible

# Pass The Ticket

- assumption: attacker with system-level privileges on a host
- exporting valid Kerberos TGTs and STs from memory
- tickets can be reused to authenticate to other services – lateral movement
- similar to pass the hash for NTLM
- compromised host as a client in Kerberos (or in general)
  - other attack possibilities

# Kerberoasting

- targeting service accounts
- anyone can ask KDC for ST for any service
- ST are encrypted with $K_S$
- off-line dictionary or brute-force attack possible
- Kerberoasting is low-profile and hard to detect
- usually the services accounts are machine accounts
  - very long, complex, and random passwords, regularly changed
- risk for the services with human-defined passwords

# The Golden Ticket Attack

- assumption:
  - attacker with administrator access to domain controller or replication privileges
- attacker gets `krbtgt` account's password hash/symmetric key
  - NT hash, AES-128, AES-256
- Golden Ticket – forged TGT, usually:
  - user added to privileged groups
  - long validity
- full domain control (domain admins)
- long-term persistence
  - golden tickets are valid until `krbtgt` password changes
  - attacker can create other persistence points meanwhile
- golden tickets are hard to detect

# The Silver Ticket Attack

- similar to the golden ticket attack, attack targets service accounts
- assumption:
  - attacker compromises a service account's password hash/symmetric key
  - often easier to get than compromising `krbtgt`
- Silver Ticket – forget ST (Service Ticket), usually:
  - user added to privileged groups
  - long validity
- allows "just" access to this particular service
- even harder detection
  - using silver ticket does not need communication with a domain controller

# Mitigation

Follow best practices:

- configuration
- patching
- monitoring
- incident response