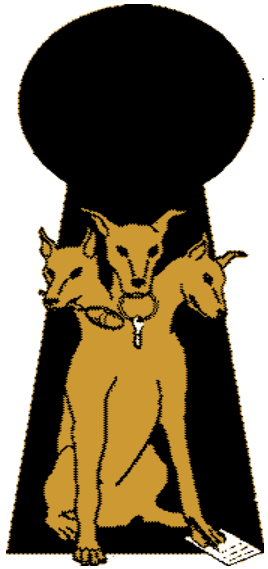
Implementing Kerberos (and Friends) on z/OS

with

Network Authentication Service

and

Resource Access Control Facility



+

Eric Rosenfeld, CISSP
z/OS Security Development
rosenfel@us.ibm.com

December 2011

Agenda

- General Kerberos Overview
- Base Kerberos Registry Support Overview
- Getting Started
  - Server Information
  - Registry set-up
- SAF Callable Services
- Dependencies and Considerations
- z/OS V1R12 & V1R13 Updates
- Session Summary

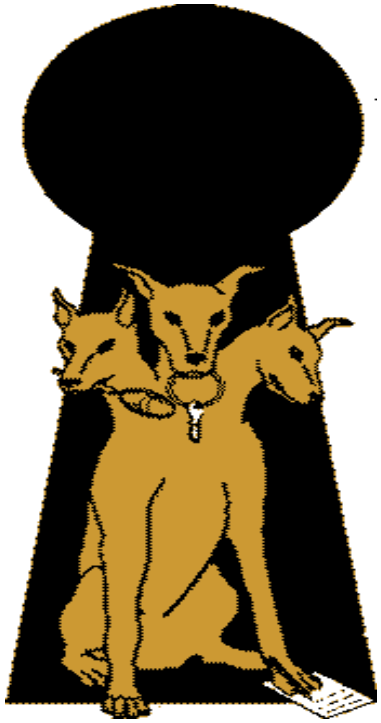
Trademarks

- The following are trademarks or registered trademarks of the International Business Machines Corporation:
  - IBM, DB2, OS/390, RACF, SecureWay, z/OS, AS/400, AIX
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- Microsoft, Windows, and Windows NT are trademarks of Microsoft Corporation in the United States, other countries, or both.
- SOLARIS is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries
- Kerberos is a trademark of MIT
- Other company, product, and service names may be trademarks or service marks of others.

Greek Mythology

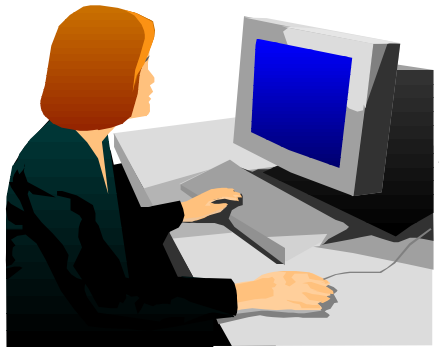
Kerberos (Cerberus) was the mythological three-headed dog that guarded the entrance to the underworld.

Unless you could get past Kerberos, you could not enter (or leave!) the underworld



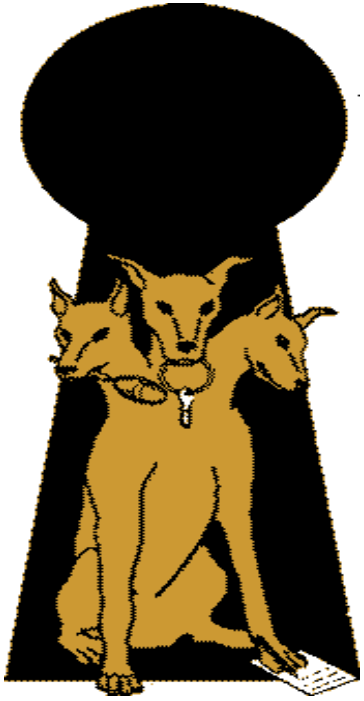
What is Kerberos?

- A distributed authentication service developed by MIT
- Allows user authentication over a physically untrusted network
- Tickets are issued by a Kerberos authentication server
  - Users and servers are required to have keys registered with server
- Flows to and from server covered by a session key
  - used in a direct exchange between a user and a service
- V5 implemented in z/OS, z/VM, AIX, AS/400, Windows, Linux, Solaris and others
  - **Network Authentication Service** component of Integrated Security Services on z/OS



Client

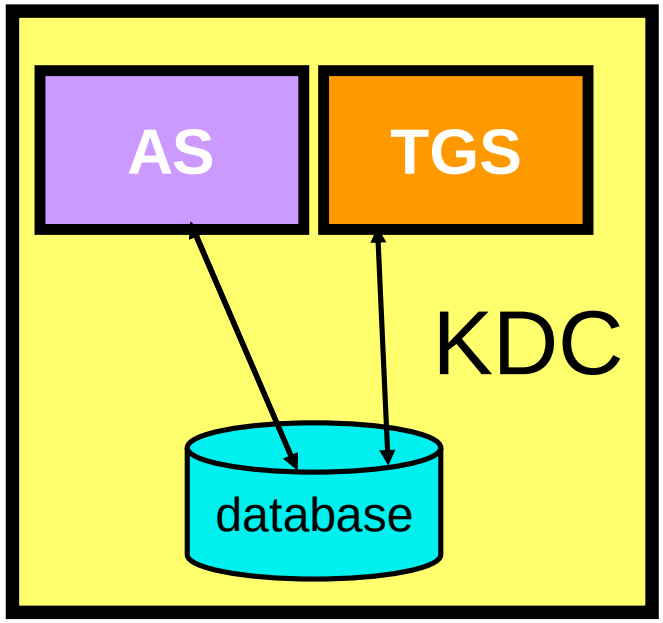
Trusted
Third
Party



Server

Key Distribution Center (KDC)

- Trusted "third party“
  - Both client and server trust the information in/decisions of the KDC
- Responsible for issuing user credentials and tickets
- Consists of
  - an authentication server (KAS)
    - Authenticates users
    - Grants Ticket Granting Tickets
  - a ticket granting server (TGS)
    - Generates session key
    - Grants service tickets
  - a Kerberos Data Base (KDB)
    - Contains keys for each user and server



Terms

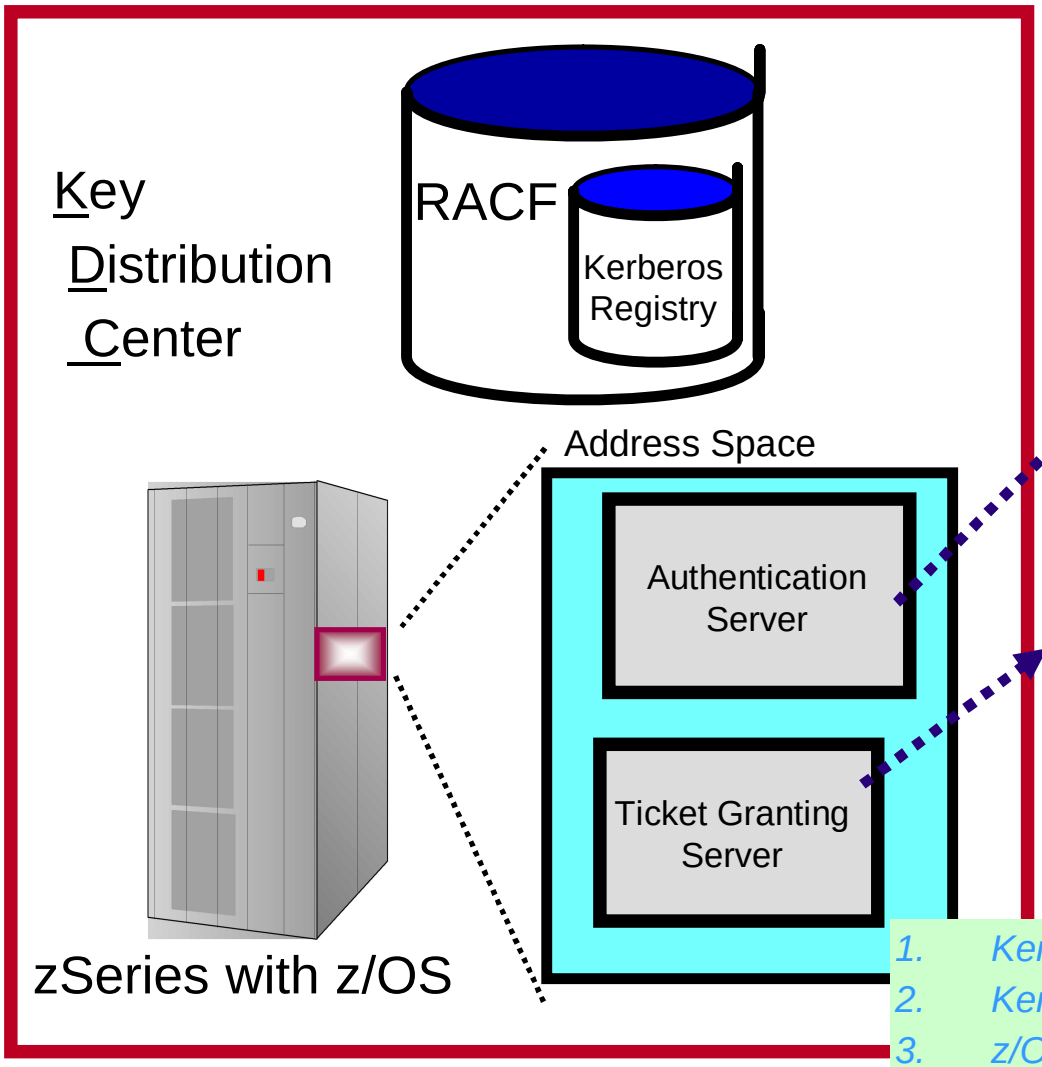
- Ticket
  - An encrypted electronic authentication token including:
    - client's identity
    - a dynamically created session key
    - a time stamp
    - lifetime for the ticket
    - a service name
- Realm
  - The Kerberos domain: the set of entities which authenticate using the domain of authority served by one KDC.
- Principal
  - Anything that is defined to a realm
  - *name@realm*
    - Can be a user, service or relationship

Ticket Use

- At logon (kinit) Ticket Granting Ticket returned
- To use a service, TGT presented w/request
- Server returns service ticket
  - Contains session key
  - Client presents service ticket to server as part of authentication protocol
    - GSS-API gss_init_sec_context method
  - Can be used until expiration
  - Avoids repeated authentication

Kerberos on z/OS

(Its own component, integrated with RACF via SAF)



Standards
RFC 1510 => Kerberos V5
RFC 1964 => GSS-API

(AS)

- Authenticates Users
- Grants TGTs

(TGS)

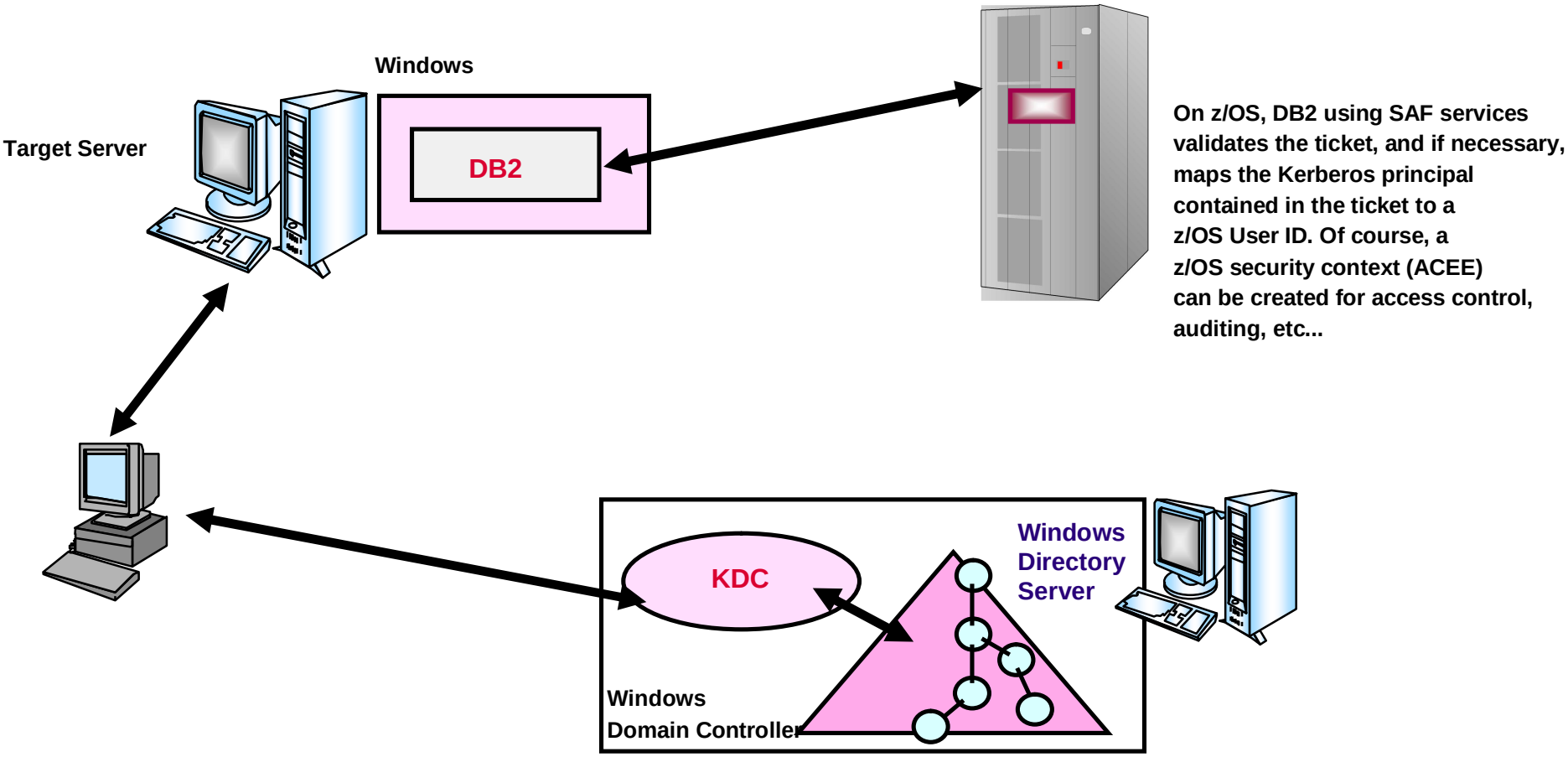
- Generates Session Keys
- Grants service tickets based on TGT

1. *Kerberos registry integrated into RACF registry*
2. *Kerberos KDC executes within z/OS address space*
3. *z/OS KDC behaves like any other Kerberos "Realm"*
4. *Kerberos Realm to Realm function supported*

z/OS and Windows Kerberos Domain

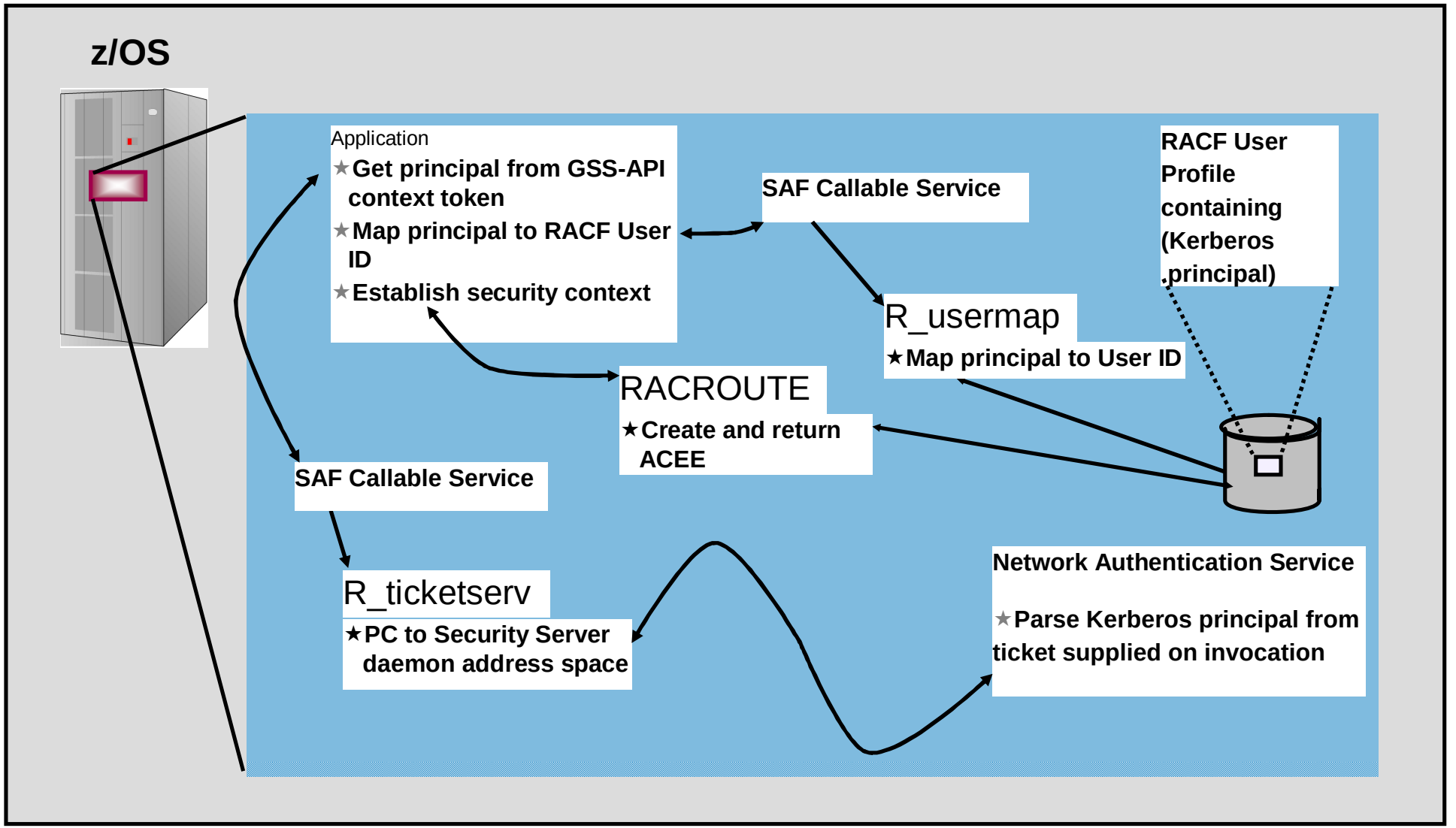
The client authenticates to the KDC, and obtains a ticket for the target server.

The assumption in this chart, is that the target server is Windows running DB2, and the target server makes a request to a DB2 instance on z/OS. The DB2 instance on the target server passes the ticket of the user client on the flow to the z/OS host.



z/OS and Windows Kerberos Domains...

This pictorial indicates that z/OS needs to be a viewed as a Kerberos peer domain. Administratively, a peer trust relationship has been established between the z/OS Kerberos domain and a Windows Kerberos domain. Local Kerberos principals must be defined to the z/OS Security Server and a user profile segment will hold the Kerberos principal name. Support is also provided to map a Kerberos principal name to a RACF User ID. Note that principal registration must be performed in two places, 1) to the Windows Kerberos domain, and 2) to the z/OS Kerberos domain.



Network Authentication Service – Commands

- kinit - obtains or renews the Kerberos ticket-granting ticket.
- klist - displays the contents of a Kerberos credentials cache or key table.
- kdestroy - destroys a Kerberos credentials cache.
- keytab - manages a key table (z/OS likely will use RACF).
- ksetup - manages Kerberos service entries in the LDAP directory for a Kerberos realm.
- kpasswd - allows principal to change password or password phrase
- kvno - returns key version number.
- kadmin - administer non RACF backed z/OS KDC with Kerberos commands
  - help, list_principals, add_principal, delete_principal, change_password, rename_principal, list_policies, add_policy, delete_policy, add_key, etc.

RACF is the Kerberos Registry

- The Network Authentication Server requires a registry of principal information, global information, etc.
- This security information is stored in RACF User and General Resource profiles
- Kerberos administration is done via RACF commands/panels
- The Network Authentication Server obtains it's registry information via SAF callable service
- Kerberos application servers can use SAF callable services to parse Kerberos tickets to obtain principal names, and to map from principal to RACF user and vice versa

RACF Classes

- KERBLINK
  - Maps Kerberos principal to RACF userid
    - ADDUSER/ALTUSER defines local profiles
    - RDEF/RALT used to define foreign profiles
- REALM
  - Defines default information for local realm (KERBDFLT)
  - Defines inter-realm trust
    - A TGT issued in one realm can be used in another

Kerberos Registry

- Local Kerberos principals are defined as RACF users with a KERB segment
- REALM class profiles are used to define information about the local Kerberos realm and foreign realms
  - Local realm information includes name, key, and ticket lifetime (MIN, MAX, and DEFAULT in seconds)
  - Foreign realm trust relationships are defined in pairs (A to B and B to A) which also include a key
- Foreign Kerberos principals are mapped to a RACF identity using KERBLINK class profiles

Kerberos Registry

- The RACF user password/password phrase and the Kerberos local principal's password are integrated
  - Kerberos key will be generated when the user's password or password phrase changes and is **not** expired
    - TSO/application logon
    - ALU NOEXPIRED
    - PASSWORD command
  - The Kerberos password and password phrase are subject to RACF SETROPTS rules and installation defined rules via exits

The GSS-API

- Generic Security Service Application Programming Interface (GSS-API) support is provided by the z/OS Network Authentication Service
  - The GSS-API is a set of programming interfaces which abstract identity authentication, message origin authentication and integrity, and message confidentiality
  - In concept, a secure application developed using the GSS-API should be able to work over different security mechanisms without changes to the application
- Originally, the z/OS Network Authentication Service GSS-API offering only supported the Kerberos security mechanism
- LIPKEY and SPKM-3 mechanisms were added as extensions to the GSS-API support

SAF Services

- R_kerbinfo is called by the server to
  - Retrieve principal information
  - Retrieve realm information
  - Update the count of invalid key attempts
    - similar to an invalid logon attempt
  - Reset the count of invalid key attempts
    - like when you remember your password, on your 2nd or 3rd try
- R_ticketserv is called by applications to determine the principal name associated with a credential
- R_usermap is called by applications to map from principal to RACF identifier

SAF Services (cont)

GSS-API support

- Allows Kerberos GSS-API function via non-LE interface
- R_GenSec service provides following GSS-API functions:
  1. GSEC_INIT_SEC_CONTEXT
  2. GSEC_CONT_SEC_CONTEXT
  3. GSEC_ACC_SEC_CONTEXT
  4. GSEC_DEL_SEC_CONTEXT
  5. GSEC_REL_CRED
  6. GSEC_GET_MIC
  7. GSEC_VER_MIC
  8. GSEC_WRAP_MSG
  9. GSEC_UNWRAP_MSG
  10. GSEC_EXPORT_SEC_CONTEXT
  11. GSEC_EXPORT_CRED
  12. GSEC_IMPORT_SEC_CONTEXT
  13. GSEC_IMPORT_CRED
  14. GSEC_ACQUIRE_CRED

Steps for Getting Started

- Install/Customize Network Authentication Server
- Set up registry
  - Define local realm
  - Define inter-realm relationships
  - Define local principals
  - Define foreign principals

Network Authentication Service - Installation

- Installs into
  - UNIX file system
    - executables in directory /usr/lpp/skrb
    - /etc/skrb files need access 755
    - /var/skrb/creds needs access 1777
  - System datasets
    - EUVF.SEUVFLPA
    - SYS1.SIEALNKE
    - EUVF.SEUVFEXC for SYSEXEC DD concatenation for TSO

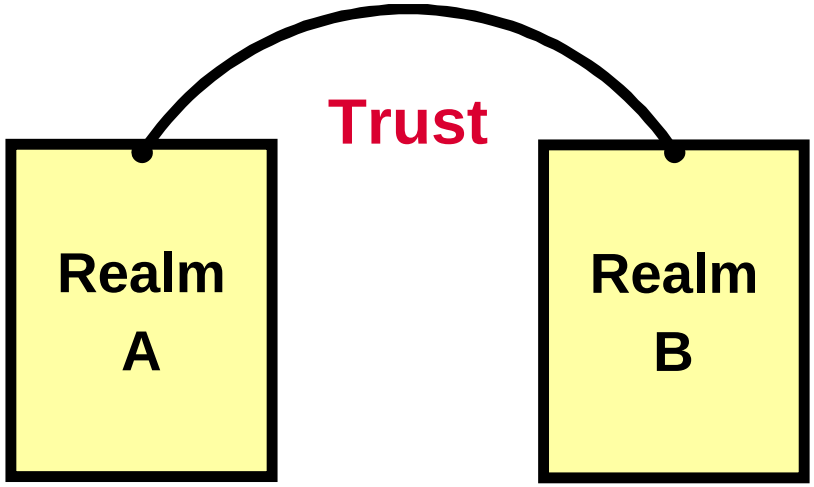
Network Authentication Service - Installation

- Configuration in krb5.conf file
  - KRB5_CONFIG environment variable
  - default is /etc/skrb/krb5.conf
  - sample in /usr/lpp/skrb/examples/krb5.conf
  - permissions should be read for everyone, only administrator may modify
  - modified only in code page 1047

Network Authentication Service - Installation ...

- Set-up RRSF(RACF Remote Sharing) in local mode
- Define SKRBKDC application and USERID as started task
- Copy SKRBKDC environment variables definitions to /etc/skrb/home/kdc/envar
- Set TZ and RESOLVER_CONFIG for your installation

Registry Definitions



Commands must be entered to define:

A local realm
Inter-realm trust relationships (between KDCs)
Local and foreign principals

Realm Commands

- Realm definition with RDEFINE/RALTER
  - Realm class profile
  - Ticket life values
    - DEFTKTLFE - default ticket life
    - MAXTKTLFE - maximum ticket life
    - MINTKTLFE - minimum ticket life
    - Only valid for local realm
    - If one is specified all three values must be for RDEFINE
    - All three values must be on command or in DB for RALTER
    - Range from 1 to 2,147,483,647 seconds

Realm Commands ...

- **KERBNAME** - unqualified name of the local Kerberos realm
  - Max length of 117 characters
  - Can not contain '/'
  - EBCDIC variant characters should not be used
- **PASSWORD** - realm password
  - Max length of 128 characters
  - EBCDIC variant characters should not be used
- **ENCRYPT** – Supported encryption types
  - DES, Triple DES, DES with Derivation, AES 128 and AES 256
- **NODEFTKTLFE, NOMAXTKTLFE, NOKERBNAME, NOMINTKTLFE, NOPASSWORD**, **NOENCRYPT** and **NOKERB** only for RALTER

Realm Commands ...

- Profile naming
  - Defining a local realm
    - Profile name must be KERBDFLT
    - KERBNAME field has unqualified local realm name
    - Realm name is rolled to upper case
  - Defining an inter-realm trust relationship
    - Can consist of two REALM class profiles
      - Profile name: /.../LOCAL_REALM/krbtgt/REALM_2
        - krbtgt/REALM_2@LOCAL_REALM
      - Profile name: /.../REALM_2/krbtgt/LOCAL_REALM
        - krbtgt/LOCAL_REALM@REALM2

Realm Command *Examples*

- Local Realm example:
  - RDEFINE REALM KERBDFLT KERB(KERBNAME(KRB390.IBM.COM) PASSWORD(xxxx) MINTKTLFE(15) DEFTKTLFE(36000) MAXTKTLFE(86400))
- Inter-realm trust example:
  - RDEFINE REALM /.../KRB390.IBM.COM/krbtgt/KRB2000.IBM.COM KERB(PASSWORD(passwrd1))
  - RDEFINE REALM /.../KRB2000.IBM.COM/krbtgt/KRB390.IBM.COM KERB(PASSWORD(passwrd 2))

User Commands

- Local principal definition with ADDUSER/ALTUSER
  - Local realm must exist before issuing command
  - **MAXTKTLFE** specifies the local principal maximum ticket life
  - **KERBNAME** is the unique name of a local principal.
    - Can not contain '@'
    - Variant characters should not be used
    - Can not exceed 240 characters when fully qualified with the local realm name
      - /.../local_realm/kerbname_1
    - Must be entered unqualified
  - **ENCRYPT** specifies supported encryption types
    - Choice of DES, Triple DES, DES with Derivation, AES 128 and AES 256
  - **NOMAXTKTLFE, NOKERBNAME, NOENCRYPT, NOKERB** only valid on ALTUSER
  - Kerberos keys generated at non-expired password setting
  - KERBLINK mapping profile created/updated

LISTUSER - Key information

When the initial KERB segment is added via

ADDUSER USER1 KERB(KERBNAME(User1))

the password is not yet synchronized with the Kerberos local principal's password:

LISTUSER USER1 KERB NORACF

USER=USER1
KERB INFORMATION

KERBNAME= User1

After a password change, the key is generated !

USER=USER1
KERB INFORMATION

KERBNAME= User1
KEY VERSION= 001

Mapping Foreign Users

- Foreign Kerberos principals are mapped to a RACF identity using KERBLINK class profiles
- RDEFINE KERBLINK /.../foreign_realm/foreign_principal APPLDATA('racf_user')
  - Maps single foreign principal to a RACF userid
- RDEFINE KERBLINK /.../foreign_realm/ APPLDATA('racf_user')
  - Maps all principals for a single realm to a RACF userid
- Realm names are rolled to upper case

Steps for Getting Started

- Install/Customize Server
- Define local realm
  - RDEFINE REALM KERBDFLT KERB(KERBNAME(realm) PASSWORD(realmpass))
- Define inter-realm relationships
  - RDEFINE REALM /.../realm1/krbtgt/realm2 KERB(PASSWORD(TrustP1))
  - RDEFINE REALM/.../realm2/krbtgt/realm1 KERB(PASSWORD(TrustP2))
- Define local principals
  - ALTUSER user1 KERB(KERBNAME(KerbUSER1)) PASSWORD(usrp) NOEXPIRED
- Define foreign principals
  - RDEFINE KERBLINK /.../foreign_realm/foreign_principal APPLDATA('racf_user')
    - maps single principal to a RACF user
  - RDEFINE KERBLINK /.../foreign_realm/ APPLDATA('racf_user')
    - Maps all principals for a single realm to a RACF userid

z/OS Tid Bits

- TCP/IP V6 supported
- NDBM (New DataBase Manager) support
  - UNIX backed SAF database alternative
  - Not shared by SYSPLEX
  - SAF still required to map principals to RACF IDs
  - kadmin used for administration
- Keytab merge utility
  - Enables keytabs to be consolidated and keys imported
- Keytab check utility
  - Enables validity checking of keytab entries

- **Implementation of SPKM-3/LIPKEY standards**
  - –RFC 2025 & RFC 2847
  - –Positions the z/OS Network Authentication Service to be able to interoperate with other non-z/OS GSS-API implementations using existing certificate infrastructure

Dependencies and Gotchas

- Network Authentication Service implements V5 standard
- Any application can use R_ticketserv and R_usermap to map Kerberos information to RACF
- Kerberos server required to be installed prior to any key generation
- RRSF local node must be defined to allow for keys to be generated for user password application updates
- Password or Password Phrase must be changed after user definition to generate initial keys

z/OS V1R12

RFC 4120

- Updated Network Authentication Service V5 standard
  - RFC 1510 was obsoleted by RFC 4120
  - Will produce more random data that is harder to predict
  - KDC will be able to connect with future Kerberos clients
    - Connections will not fail due to unknown functionality and options in the newer clients
    - Provides for a wider range of supported KDCs and clients for interoperation
  - KDC will reject requests from newer clients with mandatory security checks that fail or cannot be issued
    - Provides toleration support - Kerberos Clients and KDCs do not have to be upgraded at the same time

Sysplex Enablement

- **Problem Statement:**
  - z/OS Kerberized applications would not work when connections were re-directed with DVIPA.
- **Solution:**
  - With a configuration option, an application server can now accept AP-REQs for another instance of the same application server provided access is granted via the RACF KERBLINK class.
- **Benefits:**
  - z/OS Kerberized applications can now benefit from a DVIPA environment.
- **Invoked by:**
  - Specifing “use_dvipa_override=1” in the libdefaults section of the krb5.conf file.

Sysplex Enablement - Usage

- Examples of set up to allow principals to function on multiple images

  The commands below need to be issued for each server principal

  – Same started task ID (STC1) on all images (no KERB segment)

  KERBLINK profile created via ADDUSER/ALTUSER for ID APP1 – app/sys1
  KERBLINK profile created via ADDUSER/ALTUSER for ID APP2 - app/sys2
  PERMIT app/sys1 CLASS(KERBLINK) ID(STC1) ACCESS(READ)
  PERMIT app/sys2 CLASS(KERBLINK) ID(STC1) ACCESS(READ)

  – Different started task ID on each image

  KERBLINK profile created via ADDUSER/ALTUSER for ID STC1 - app/sys1
  KERBLINK profile created via ADDUSER/ALTUSER for ID STC2 - app/sys2
  KERBLINK profile created via ADDUSER/ALTUSER for ID STC3 - app/sys3
  PERMIT app/sys1 CLASS(KERBLINK) ID(STC2 STC3) ACCESS(READ)
  PERMIT app/sys2 CLASS(KERBLINK) ID(STC1 STC3) ACCESS(READ)
  PERMIT app/sys3 CLASS(KERBLINK) ID(STC1 STC2) ACCESS(READ)

z/OS V1R13

Kerberos Cryptosystem Negotiation Extension

- **Application client and server encryption type negotiation for more secure communication.**
- Problem Statement
  - KDC selects encryption type for application client and server based on encryption types set in the configuration file and settings of individual principals.
  - Client and/or Server will use a weak encryption type selected by the KDC when they support a stronger encryption type
- Solution
  - Allow application client and server to negotiate an encryption types independent of KDC and configuration file and principal settings.
- Benefits
  - Higher level of security in communication between application client and server

Exploit CPACF for Ciphertext Stealing

- **Exploit ICSF's implementation of Ciphertext Stealing (CTS) for AES (ICSF HCR7780)**
- Problem Statement
  - CTS implemented in Kerberos with two calls to ICSF
  - Less than optimal performance
- Solution
  - Call ICSF callable services for symmetric encryption and decryption
- Benefit
  - Fewer service calls
  - Exploits hardware crypto

References…

➢IBM Books

- SA22-7691 z/OS Security Server RACF Callable Services
- SA22-7687 z/OS Security Server RACF Command Language Reference
- GA22-7680 z/OS Security Server RACF Data Areas
- SA22-7682 z/OS Security Server RACF Macros and Interfaces
- SA22-7686 z/OS Security Server RACF Messages and Codes
- SA22-7683 z/OS Security Server RACF Security Administrator's Guide
- SC24-5926 z/OS Integrated Security Services Network Authentication and Privacy Service Administration
- SC24-5927 z/OS Integrated Security Services Network Authentication and Privacy Service Programming
- SC24-5901 Cryptographic Services System Secure Sockets Layer Programming
- GA22-7800 z/OS Unix System Services Planning
- SA22-7803 z/OS Unix System Services Programming: Assembler Callable Services Reference
- SG24-6540 Red Book: Putting the Latest z/OS Security Features to Work
- SG24-6448 Red Book: z/OS 1.6 Security Services Update
- SG24-6986 Red Book: ABCs of z/OS System Programming Volume 6

➢Internet

- http://web.mit.edu/kerberos/www/

References

➢RFCs

- RFC 1510 - The Kerberos Network Authentication Service (V5)
- RFC 4120 - The Kerberos Network Authentication Service (V5)
- RFC 1964 - The Kerberos Version 5 GSS-API Mechanism
- RFC 2078 - Generic Security Service Application Program Interface (V2)
- RFC 2744 - Generic Security Service Application Program Interface (V2): C Bindings
- RFC 3962 - Advanced Encryption Standard (AES) Encryption for Kerberos
- RFC 4121 - The Kerberos V5 GSSAPI Mechanism: Version 2
- RFC 4537 – Kerberos Cryptosystem Negotiation Extension

- RFC 2025 - The Simple Public-Key GSS-API Mechanism (SPKM)
- RFC 2847 - LIPKEY - A low infrastructure mechanism Using SPKM
- RFC 3962 - Advanced Encryption Standard (AES) Encryption for Kerberos
- RFC 4121 - The Kerberos V5 GSSAPI Mechanism: Version 2
- RFC2253 UTF-8 String Representation of Distinguished names
- RFC2459 X.509 Public Key Infrastructure

Session Summary

- What we have covered:
  - ►What Kerberos is and does
  - ►How SAF/RACF interacts with the Network Authentication Service
  - ►How an application would interact with SAF to map Kerberos constructs to RACF constructs
  - ►How to install and configure Kerberos support
  - ►An overview of newer support

Questions ?

Reference

SPKM-3

- **The Simple Public-Key GSS-API Mechanism (SPKM) is based on a public key infrastructure, not the Kerberos symmetric-key infrastructure**
  - **SPKM-3 does not use secure timestamps, enabling secure authentication in environments without access to secure time**
  - **Designed to be flexible, for example providing Algorithm Identifiers for specifying various algorithms to be used by communicating peers**
  - **Provides support for asymmetric algorithm-based digital signatures**
  - **Data formats and procedures are designed to be as similar to the Kerberos mechanism as possible for ease of implementation by applications which are already Kerberos enabled**
- **SPKM-3 uses the same certificate infrastructure as SSL**

LIPKEY

- **LIPKEY (a Low Infrastructure Public Key Mechanism using SPKM) is a GSS-API security mechanism which can be used when the initiator (client) does not have a certificate and instead uses user ID and password for authentication**
- **It consists of a client with no public key certificate, accessing a server with a public key certificate (in contrast, in SPKM-3, both client and server require access to certificates)**
- **The server must have access to a user ID/password repository (we use the __passwd system routine, with setup/restrictions documented in the z/OS Network Authentication Service Programming Guide)**

How LIPKEY works

A client using the LIPKEY mechanism

- Obtains the server's certificate
- Verifies that it was signed by a trusted CA
- Generates a random session symmetric key
- Encrypts the session key with the server's public key
- Sends the encrypted session key to the server
- At this point, the client and server have a secure channel, so the client can provide a user name and password for authentication

R_ticketserv (IRRSPK00)

- Parse or extract Kerberos principal
  - Function code
    - TKTS_RETURN_NAME (1) - Parse specified ticket and return Kerberos principal name
      - GSS-API context token is input
      - Principal name is output

R_usermap (IRRSIM00)

- Map application user
  - Function codes:
    - UMAP_R_TO_K (5) -- return the Kerberos application user identity for the supplied RACF user ID
    - UMAP_K_TO_R (6) -- return the RACF user ID associated with the supplied Kerberos application user identity

R_admin (IRRSEQ00)

- Functions supported
  - ADMN_ADD_USER, ADMN_ALT_USER, ADMN_LST_USER ADMN_ADD_GENRES, ADMN_ALT_GENRES, ADMN_LST_GENRES to support KERB segment fields
- Fields
  - KERBNAME - realm or principal name
  - MAXTKTLF - realm or principal maximum ticket life
  - MINTKTLF - realm wide minimum ticket life
  - DEFTKTLF - realm wide default ticket life
  - PASSWORD - realm password