IBM Education Assistance for z/OS V2R2

Item: NAS PKINIT
Element/Component: NAS (Kerberos)

Material current as of May 2015

Agenda

- Trademarks
- Presentation Objectives
- Overview
- Usage & Invocation
- Interactions & Dependencies
- Presentation Summary
- Appendix

Trademarks

- See url http://www.ibm.com/legal/copytrade.shtml for a list of trademarks.

Session Objectives

- Digital certificates usage has been growing
- Continuous enhancements to fulfill customer requirements
- Components on certificate support:
  - NAS (Network Authentication Services, aka Kerberos)
- At the end of this presentation, you should have an understanding of the support from NAS PKINIT support

Overview

- Problem Statement / Need Addressed
  - Kerberos is a distributed authentication service that allows user authentication over a physically untrusted network by using tickets issued by a mutually trusted authentication server which is run by a Key Distribution Center (KDC).
  - Before the client can communicate with the server, he needs to request an initial ticket from the KDC and use that ticket to obtain a service ticket from the KDC which will then be used for the subsequent communications with the server
  - The initial authentication in Kerberos is to acquire the ticket-granting-ticket (TGT)
  - The ticket is encrypted by a secret key which is typically derived from the client's password

Overview

- Problem Statement / Need Addressed
  - The attack resistance strength of the Kerberos protocol is no stronger than the strength of the passwords
  - Another problem is the need of using UID 0 for the Kerberos started task. It is a standard policy for many companies to avoid UID 0

Overview

- Solution
  - Make use of asymmetric cryptography in the form of X.509 certificates which is popular for facilitating data origin authentication and secrecy
  - Implement RFC4556 - Public Key Cryptography for Initial Authentication in Kerberos (PKINIT) for z/OS KDC and z/OS Kerberos client
  - Support PKINIT parameters for the kinit command
  - Add PKINIT APIs
  - Lift the requirement of using UID 0 for the Kerberos started task

Overview

- Benefit / Value
  - PKINIT lifts the burden to manage strong passwords for Kerberos authentication
  - Eliminating the need of UID 0 helps to enable least-privilege

Usage & Invocation

- z/OS KDC ↔ Kerberos client on z/OS
- z/OS KDC <–> Kerberos client on Linux
- z/OS KDC <–> Kerberos client on Windows
- Kerberos client on z/OS <-> Linux KDC
- Kerberos client on z/OS <-> Windows KDC

Usage & Invocation

- New configuration options for KDC, specified in envar

```
SKDC_PKINIT_REQUIRED=1, default is 0
SKDC_PKINIT_KEYRING=*TOKEN*/KDCTOKEN
OR
SKDC_PKINIT_KEYRING=SKRBKDC/KDCRING
OR
SKDC_PKINIT_KEYRING=/etc/skrb/home/kdc/kdcgsk.kdb
SKDC_PKINIT_KEYRING_STASH=/etc/skrb/home/kdc/kdcgsk.sth

SKDC_PKINIT_REQUIRE_EKU=1, default is 0

SKDC_PKINIT_REQUIRE_REVOCATION_CHECKING=none, default is none
    OR
SKDC_PKINIT_REQUIRE_REVOCATION_CHECKING=ocsp
    OR
SKDC_PKINIT_REQUIRE_REVOCATION_CHECKING=ldap
SKDC_PKINIT_LDAP_SERVER = sysmgmt.pok.ibm.com:389
   OR
SKDC_PKINIT_REQUIRE_REVOCATION_CHECKING=ocsp, crldp

SKDC_PKINIT_DH_MIN_BITS=2048, default is 2048
```

Usage & Invocation

- New configuration options for Kerberos client, specified in krb5.conf

```
pkinit_keyring = *TOKEN*/KRBTOKEN
OR
pkinit_keyring = KRBUSR/KRBRING
OR
pkinit_keyring = /etc/skrb/clientgsk.kdb
pkinit_keyring_stash = /etc/skrb/clientgsk.sth

pkinit_require_revocation_checking = none, default is none
    OR
pkinit_require_revocation_checking = ocsp
    OR
pkinit_require_revocation_checking = ldap
pkinit_ldap_server = sysmgmt.pok.ibm.com:389
    OR
pkinit_require_revocation_checking = ocsp, crldp

pkinit_dh_min_bits = 2048, default is 2048

pkinit_host_name = kdc1.krbzos.ibm.com
```

Usage & Invocation

- New kinit parameters
  - key ring, key token or the key database
  - stash file contains the password of the key database
  - desire way to encrypt the reply – DH(Diffie-Hellman) or RSA (default is DH)

    kinit –X keyring=KRBUSR/KRBRING -X rsa_protocol=yes …

    kinit –X keyring=/etc/skrb/clientgsk.kdb -X stash=/etc/skrb/clientgsk.sth -X rsa_protocol=no ...

Usage & Invocation

- New APIs

  - **krb5_init_context_pkinit**
    Add to a Kerberos context with values specified in the configuration file for public private key authentication

    krb5_error_code krb5_init_context_pkinit (
    krb5_context context,
    char* realm,
    krb5_error_code * warning_code)
    krb5_set_value_pkinit (set pkinit value)

  - **krb5_set_value_pkinit**
    Add to a Kerberos context for public private key authentication from values specified in the input attribute / value pair parameter

    krb5_error_code krb5_set_value_pkinit (
    krb5_context context,
    char* attr,
    char* value)

Usage & Invocation

-
- **krb5_get_in_tkt_with_pkinit**
  Gets an initial ticket using a public private key pair

  krb5_error_code krb5_get_in_tkt_with_pkinit (
  krb5_context context,
  const krb5_flags options,
  krb5_address * const * addrs,
  krb5_enctype * enctypes,
  krb5_ccache ccache,
  krb5_creds * creds,
  krb5_kdc_rep ** ret_as_reply)

Usage & Invocation

- KDC example: Set up existing z/OS KDC for PKINIT support
- 1. If you don't already have a issuing CA, generate a self-signed certificate to represent the local certificate authority, eg.

  RACDCERT CERTAUTH GENCERT SUBJECTSDN(OU('Local Certificate Authority') O('Network group') C('US')) KEYUSAGE(CERTSIGN) WITHLABEL('Network Local CA')

  2. Generate the KDC certificate, specify your KDC domain name as the subject alternate domain name, signed by your issuing CA, eg.

  RACDCERT ID(SKRBKDC) GENCERT SUBJECTSDN(CN('z/OS KDC') OU('Test') O('Network A') C('US')) ALTNAME(DOMAIN('kdc1.krbzos.ibm.com')) KEYUSAGE(HANDSHAKE) WITHLABEL('zOS KDC') SIGNWITH(CERTAUTH LABEL('Network Local CA'))

  3. Create the KDC key ring, eg

  RACDCERT ID(SKRBKDC) ADDRING(KDCRING)

  4. Connect the KDC certificate to the key ring and mark it as the default certificate, eg

  RACDCERT ID(SKRBKDC) CONNECT(LABEL('zOS KDC') RING(KDCRING) DEFAULT)

Usage & Invocation

5. Connect the local certificate authority certificate to the key ring as well, eg

RACDCERT ID(SKRBKDC) CONNECT(CERTAUTH LABEL(’Network Local CA’) RING(KDCRING))

- 6. Add and connect the CA certificate that signed the Kerberos client certificate, eg

RACDCERT CERTAUTH ADD .....WITHLABEL('KRBClient CA')
RACDCERT ID(SKRBKDC) CONNECT(CERTAUTH LABEL('KRBClient CA') RING(KDCRING))

7. Give SKRBKDC permission to read its own key ring by administering a profile in the RDATALIB class

RDEFINE RDATALIB SKRBKDC.KDCRING.LST UACC(NONE)

PERMIT SKRBKDC.KDCRING.LST CLASS(RDATALIB) ID(SKRBKDC) ACCESS(READ)

– If the RDATALIB class is not already active, activate and RACLIST it.
SETROPTS CLASSACT(RDATALIB) RACLIST(RDATALIB)

– If the RDATALIB class is already active and RACLISTed, refresh it.
SETROPTS RACLIST(RDATALIB) REFRESH

8. Edit /etc/skrb/home/kdc/envar to point to the KDC keyring

SKDC_PKINIT_KEYRING=SKRBKDC/KDCRING

Usage & Invocation

- Client example: Set up existing z/OS Kerberos client for z/OS KDC
- 1. Get a certificate with these attributes from a CA that support them:

  OtherName extension containing the client's principal name and the realm name:
  OID:1.3.6.1.5.2.2 id_pkinit-san, OR
  *OID: 1.3.6.1.4.1.311.20.2.3 id-ms-san-sc-logon-upn

  *PKI Services can generate a certificate with this attribute

  2. Create the client key ring, eg.

  RACDCERT ID(KRBUSR) ADDRING(KRBRING)

  3. Add and connect the client certificate to the key ring and mark it as the default certificate, eg

  RACDCERT ID(KRBUSR) ADD... WITHLABEL('KRBCert')
  RACDCERT ID(KRBUSR) CONNECT(LABEL('KRBCert') RING(KRBRING) DEFAULT)

  4. Add and connect the certificate authority certificate that signed the client certificate to the key ring as well, eg

  RACDCERT CERTAUTH ADD.... WITHLABEL('KRBClient CA')
  RACDCERT ID(KRBUSR) CONNECT(CERTAUTH LABEL('KRBClient CA') RING(KRBRING))

Usage & Invocation

5. Give KRBUSR permission to read its own key ring by administering a profile in the RDATALIB class.

- RDEFINE RDATALIB KRBUSR.KRBRING.LST UACC(NONE)

  PERMIT KRBUSR.KRBRING.LST CLASS(RDATALIB) ID(KRBUSR) ACCESS(READ)

6. Send the CA certificate which signed the client certificate to the KDC and add the KDC CA certificate to the client keyring

7a. Edit krb5.conf to point to the client keyring, eg

pkinit_keyring=KRBUSR/KRBRING

OR

7b. Specify -X in the kinit command, eg.

kinit KRBUSR@KRBZOS.IBM.COM -X keyring=KRBUSR/KRBRING

Interactions & Dependencies

- Software Dependencies
  - None
- Hardware Dependencies
  - None
- Exploiters
  - Customers running Kerberos on z/OS, Linux and Windows

Session Summary

- At the end of this presentation, you should have an understanding of the support from:
  - RACF:
    - RACDCERT granular administration (FP0131)
  - PKI Services:
    - OCSP currency(FP0489)
    - Nxm Authorization (FP0344)
  - NAS (kerberos):
    - PKINIT support (FP0219)

Appendix

- Publication references
  - *Integrated Security Services Network Authentication Service Administration* (SC23-6786)
  - *Integrated Security Services Network Authentication Service Programming* (SC23-6787)
- RFCs
  - RFC4120 – The Kerberos Network Authentication Service V5
  - RFC4556 - Public Key Cryptography for Initial Authentication in Kerberos (PKINIT)