z/OS 3.2 IBM Education Assistant

Solution Name: RACF AES LDAPBIND Support

Solution Element(s): z/OS Security Server RACF

July 2025

Agenda

- Trademarks
- Objectives
- Overview
- Usage & Invocation
- Interactions & Dependencies
- Upgrade & Coexistence Considerations
- Installation & Configuration
- Summary
- Appendix

Trademarks

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

Objectives

The Challenge: Prior to this support RACF support for encrypting LDAPBIND credentials via the KEYSMSTR class only supports non-quantum safe algorithms (DES).

The Objective: Add support for quantum-safe encryption (AES) for RACF LDAPBIND authentication credentials stored in RACF.

Overview

- **Who (Audience)**
  - RACF installations who store external authentication credentials in RACF via the LDAPBIND and KEYSMSTR class profiles.

- **What (Solution)**
  - The RACF LDAPBIND and KEYSMSTR class support for storing encrypted authentication credentials is enhanced to support quantum-safe encryption (AES).

- **Wow (Benefit / Value, Need Addressed)**
  - Stronger protection for passwords being shared with other platforms.
  - Regulatory compliance.

Usage & Invocation (1)

Some applications have a need to connect to an external server such as LDAP. When connecting to an external server with stored user ID and password, it's best to encrypt those credentials.

KEYSMSTR Class:

- RACF/SAF provides functions for encrypting and decrypting passwords for external servers such as LDAP via the KEYSMSTR and LDAPBIND classes.
- Although the KEYSMSTR and LDAPBIND functions are expressed in terms of the LDAP application (and DCE), any type of non-RACF-user password can be encrypted, saved in the RACF database, decrypted and returned top be used as a password to authenticate to the external server.

Encryption Algorithm:

- The existing KEYSMSTR class functions use the DES encryption algorithm.
- Not NIST approved. Not quantum-safe.

Usage & Invocation (2)

Configuration steps for LDAPBIND / KEYSMSTR class support:

1. Configure the Encryption Key:

- The security administrator configures the encryption key by creating a profile in the KEYSMSTR class with a SSIGNON segment which identifies the key to be used to encrypt passwords
- `RDEFINE KEYSMSTR LDAP.BINDPW.KEY SSIGNON(KEYLABEL(ICSF.KEY.LABEL))`
- The key material can be masked or encrypted by use of keywords in the SSIGNON segment:
  - **KEYMASKED()** – Key material is provided on RACF command and masked in the RACF DB
  - **KEYENCRYPTED()** – Key material is provided on RACF command and encrypted in ICSF and RACF DB has the ICSF label
  - **KEYLABEL()** – Key material is created outside of RACF and RACF DB has the ICSF label

2. Configure the Application Password:

- Security administrator configures the application password in an application profile in the LDAPBIND class.
- `RDEFINE LDAPBIND APPL01.PROFILE PROXY(BINDPW('PASSWORD'))`
- This password is encrypted with the key identified in the KEYSMSTR class profile before being stored in RACF.
- When the password is changed the encryption key is also stored with the password to insulate it from a key change.

3. Application retrieves the password:

- Application calls the SAF **R_dcekey** callable service and asks for the password from a particular profile in the LDAPBIND class (APPL01.PROFILE in the example above).
- The password is decrypted with the key that was used to encrypt it (key from KEYSMSTR class not used for decryption).

Usage & Invocation (3)

Starting with OA66458 (z/OS 2.5) and z/OS 3.2 base the RACF KEYSMSTR class functions provide an option for quantum-safe encryption with support for the AES encryption algorithm.

Using an AES key with the KEYSMSTR class:

- With this support, the security administrator can now use the KEYLABEL field in the SSIGNON segment in the KEYSMSTR class profile to refer to an AES key in ICSF.
- No application changes are required:
  - Applications still call R_dcekey to retrieve the password in the clear.
  - The password is encrypted with the AES or DES key defined in the KEYSMSTR class profile.

Other KEYSMSTR class usage (which now also support AES):

- R_dcekey functions that encrypt DCE passwords.
  - Uses the DCE.PASSWORD.KEY profile in the KEYSMSTR class.
- R_Proxyserv callable service can also use KEYSMSTR class profiles.

Usage & Invocation (4)

Steps to exploit AES encryption for KEYSMSTR/LDAPBIND passwords:

1. **Confirm AES KEYSMSTR/LDAPBIND support is available:**
   - Be on 3.2 or apply PTF for OA66458 (z/OS 2.5) and IPL (all systems that share the RACF database)
   - A KEYSMSTR AES key should not be configured until all systems sharing the RACF database have the support available.
2. **Find all Encrypted Passwords:**
   - Find all profiles in the LDAPBIND class (SEARCH command or DBUNLOAD utility)
   - List each LDAPBIND profile and PROXY segment. RLIST will indicate when the profile contains a BINDPW password (but not the actual password value).
   - Determine the external password (your own documentation or call R_dcekey to recover it in the clear).
3. **Define and configure AES key:**
   - Create AES key in ICSF (REXX script or ICSF panels)
   - Update KEYSMSTR class profile to refer to the new AES key
4. **Reset the Passwords:**
   - Use the RALTER command to set the BINDPW in each the LDAPBIND profile.

Interactions & Dependencies

- Software Dependencies
  - ICSF required for AES (and DES) encryption
- Hardware Dependencies
  - None - The AES encryption supports both clear keys and secure keys
    - **Note:** If using secure keys, AES key must be marked as protected eligible via the SYMCPACFWRAP(YES) keyword in the CSFKEYs class profile which covers the ICSF key label.
    - **Note:** DES encryption requires a crypto coprocessor.
- Exploiters
  - Comm Server – CSSMTP Email Server
    - Plans to use this service to store encrypted credentials to an email server.

Upgrade & Coexistence Considerations

- To exploit this solution, all systems in the Plex:
  - Must be at the new z/OS level (or)
  - Must be on z/OS 3.1 or 2.5 with PTFs for APAR OA66458 applied
- Note that when RACF configuration is shared via shared database or is propagated in an RRSF network, behavior on each system depends on the release and service level.
  - When an AES ICSF key label is configured in a KEYSMSTR class profile a system without the RACF AES KEYSMSTR class support introduced by APAR OA66458 will fail when attempting to use the AES key to perform encryption.
  - When RACF data is encrypted with an AES key label configured in a RACF KEYSMSTR class profile a system without the RACF AES KEYSMSTR class support introduced by APAR OA66458 will fail when attempting to use the AES key to perform decryption.
- List any toleration/coexistence APARs/PTFs: None
- List anything that doesn't work the same anymore:
  - When configured, RACF can encrypt LDAPBIND credentials with the AES encryption algorithm.

Installation & Configuration

- Are any APARs or PTFs needed for enablement? Not on 3.2.
  - Function is also available on 2.5 via PTF for OA66458
- What jobs need to be run? None
- What hardware configuration is required? None
- What PARMLIB statements or members are needed? None
- Are any other system programmer procedures required? No
- Are there any planning considerations? No
- Are any special web deliverables needed? No
- Does installation change any system defaults? No

Summary

- RACF support for encrypting LDAPBIND credentials via the KEYSMSTR class is enhanced to support AES encryption for stronger protection of external server authentication credentials and regulatory compliance.

Appendix (1)

IBM Publications

- **SA23-2292-xx - Security Server RACF Command Language Reference**
  - Update RALTER/RDEFINE commands SSIGNON segment to indicate it can also be used to define encryption keys for profiles in KEYSMSTR class profiles.

- **SA23-2289-xx - Security Server RACF Security Administrator's Guide**
  - Update Chapter “General resources” to add AES key type for KEYSMSTR profiles.

APAR Details:

https://www.ibm.com/support/pages/apar/OA66458

Appendix (2)

Terminology:

- **DES** - **Data Encryption Standard** is as symmetric 56-bit key method of data encryption. It was adopted in 1977 for government agencies to protect sensitive data and was officially retired in 2005.
- **AES** - **Advanced Encryption Standard** is a symmetric block cipher chosen by the U.S. government to protect classified information. It supports three different key lengths: 128, 192 and 256 bits and was approved for use by the U.S. National Institute of Standards and Technology (NIST) in 2001.