IBM Software Group

Installation, Configuration and Basic Test of WebSphere MQ Advanced Message Security 7.5 in Linux

Angel Rivera (rivera@us.ibm.com)
WebSphere MQ Unix® Level 2 Support
11-Jun-2014

WebSphere® Support Technical Exchange

Agenda

- What is Advanced Message Security (AMS) and its benefits
- How to install AMS
- How to configure AMS
- How to perform a basic test using the amqsput and amqsget samples
- Exploration of scenarios

Tutorial techdoc

- This WSTE is based on the following tutorial techdoc:

- IBM Techdoc: 7041465
- http://www.ibm.com/support/docview.wss?uid=swg27041465

- Installation, Configuration and Basic Test of WebSphere MQ Advanced Message Security 7.5 in Linux

What is MQ AMS?

- IBM WebSphere MQ Advanced Message Security (WebSphere MQ AMS) is a separately licensed component of WebSphere MQ that provides a high level of protection for sensitive data flowing through the WebSphere MQ network.
- Protects the data not only as it flows across the network but **also when it is stored in a queue.**
- Verifies that a sender of message data is authorized to place signed messages on a queue, and that a receiver is authorized to get messages.

What is MQ AMS? Additional benefits

- Secures sensitive or high-value transactions processed by WebSphere MQ.
- Detects and removes rogue or unauthorized messages before they are processed by a receiving application.
- Verifies that messages were not modified while in transit from queue to queue.

What has changed from AMS 7.0 to 7.5?

- MQ AMS 7.0:
- A separate product that is installed on top of MQ.
- Need to explicitly configure and enable interceptors
- Requires the use of the cfgmqs command

- MQ AMS 7.5:
- AMS installation is a part of MQ installation process (shipped with MQ Server)
- AMS security capability are enabled with its installation. Interceptors already enabled.
- Does not require the use of cfgmqs command.

What has changed from AMS 7.0 to 7.5?

- MQ AMS 7.5
- The information regarding AMS is now included with the MQ 7.5 Knowledge Center:
- http://www.ibm.com/support/knowledgecenter/SSFKSJ_7.5.0/

- Instead of having a separate Information Center, like with AMS 7.0.

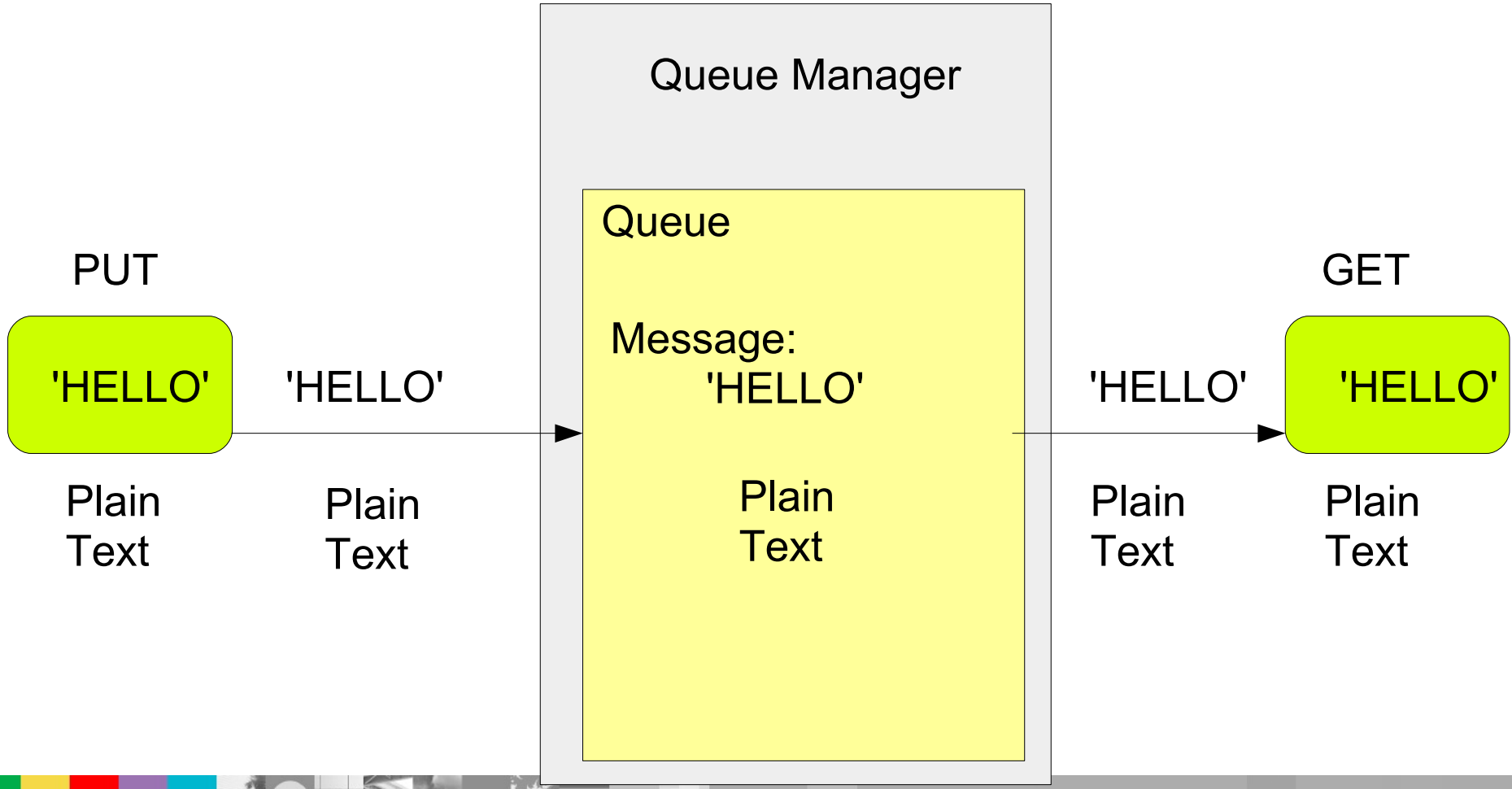
Scenario 1: No encryption at all

- Let's compare scenarios to better understand the value added of AMS

- The default behavior in MQ is to NOT use encryption when:
- MQ clients put/get messages to/from queue
- Messages stored in the queue

- That is:
- The messages are sent in Plain Text
- The messages are stored in Plain Text

Scenario 1: No encryption at all

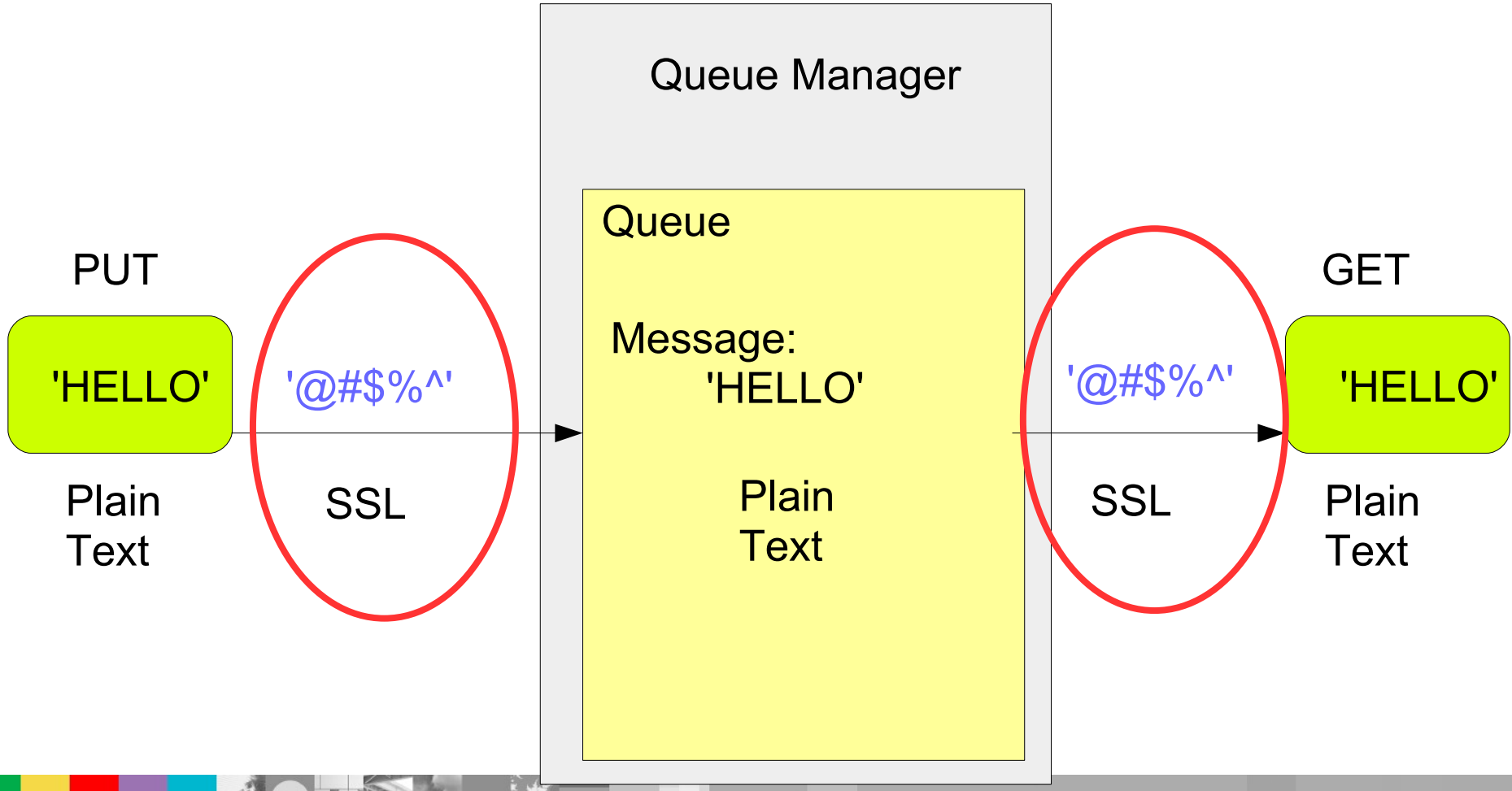


Scenario 2: Encryption during Put/Get

- It is possible to use Secure Sockets Layer (SSL) to encrypt the communication between an MQ client and the MQ queue manager during Puts/Gets.
- But the messages are stored in Plain Text in the queue.

- That is:
- The messages are sent **Encrypted**
- The messages are stored in Plain Text

Scenario 2: Encryption during Put/Get



Problem: root/mqm can read contents of q's

- For some teams, the fact that the user root or the user “mqm” (or another MQ administrator) can read the contents of the queue is an issue.
- Let's explore this issue in the next slides

- In this presentation there are 2 queues for comparison:
- Q1 = normal, NOT controlled by AMS
- TEST.Q = controlled by AMS, contents is encrypted and only selected users can access it

Notes: MQ Admin can browse messages

notes

- User “alice” puts a message into Q1:

- alice$ **amqsput Q1 QM_VERIFY_AMS**
- Sample AMQSPUT0 start
- target queue is Q1
- **this is a test**
- Sample AMQSPUT0 end

- User “mqm” can browse the contents of the queue and see the message in plain text:

- mqm$ **amqsbcg Q1 QM_VERIFY_AMS**
- AMQSBCG0 - starts here
- MQGET of message number 1, CompCode:0 Reason:0
- ****Message descriptor****
- StrucId : 'MD ' Version : 2
- ...
- **** Message ****
- length - 14 of 14 bytes
- 00000000: 5468 6973 2069 7320 6120 7465 7374 'this is a test '

Notes: MQ Admin can browse messages

notes

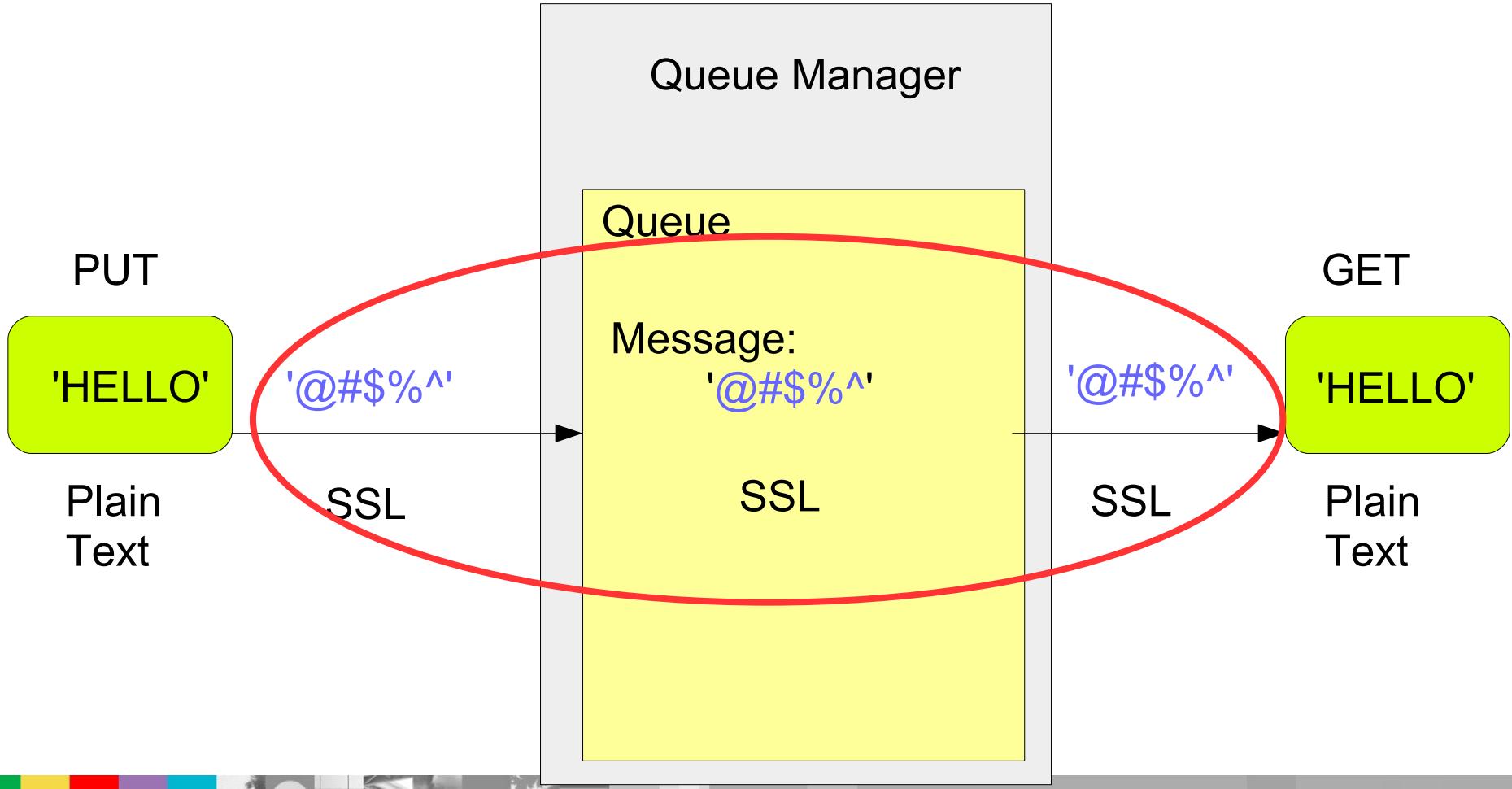
- User “mqm” has access to the physical file for the queue and can bypass MQ commands and use operating system commands to see inside the queue:
- mqm$ **cd /var/mqm/qmgrs/QM_VERIFY_AMS/queues**
- mqm$ **ls -l Q1**
- -rw-rw---- 1 mqm mqm 2560 2014-02-24 07:56 Q1
- mqm$ **cat Q1**
- + begin output
- AQQF°¸$Äÿÿ
- ÿÿ
- ÿÿÿÿÿÿÿÿÿÿÿLLQQ1
- 2014-02-24 07.30.42 @ ÿÉ;ÿÿÿÿ2014-02-24 07.30.42
- 8àcÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿAQRHÿÿÿÿÿÿÿÿÿÿÿÿÿÀAMQ QM_VERIFY_AM9S.
- ÿÿÿÿ&óY98}ÿÿÿÿMD "¸MQSTR QM_VERIFY_AMS
- rivera amqsput 2014022412554198 ÿÿ **this is a test**¸MQSTR
- + end output

Scenario 3: Encryption with AMS

- With MQ Advanced Message Security it is possible to use SSL to encrypt:
- The communication between an MQ client and the MQ queue manager during Puts/Gets,
- The messages are stored encrypted in the queue.

- That is:
- The messages are sent **Encrypted**
- The messages are stored **Encrypted**

Scenario 3: Encryption all the way



Not even root / MQ admins can read queue

- The messages are stored encrypted in the queue controlled by AMS.
- Not even the user “root” nor the MQ Administrators can view the unencrypted messages.
- They get error 2035 (MQRC_NOT_AUTHORIZED) if they use MQ commands to browse, get.
- They see the encrypted messages (not human readable) if they access directly the queue file.

Notes: MQ Admin cannot browse messages

notes

- User “alice” puts a message into TEST.Q (user has authorization)

- alice$ amqsput TEST.Q QM_VERIFY_AMS
- Sample AMQSPUT0 start
- target queue is TEST.Q
- **this is a test**
- Sample AMQSPUT0 end

- User “mqm” **cannot** browse the contents of the queue because it is not authorized!

- mqm$ amqsbcg TEST.Q QM_VERIFY_AMS
- AMQSBCG0 - starts here
- **********
- MQOPEN - 'TEST.Q'
- MQOPEN failed with CompCode:2, Reason:2035

- $ mqrc 2035
- 2035 0x000007f3 MQRC_NOT_AUTHORIZED

Notes: MQ Admin sees encrypted data

notes

- User “mqm” has access to the physical file for the queue and can bypass MQ commands and use operating system commands to see inside the queue:

- mqm$ **cd /var/mqm/qmgrs/QM_VERIFY_AMS/queues**

- mqm$ **ls -l *Q***
- -rw-rw---- 1 mqm mqm 2560 2014-02-24 07:56 Q1
- -rw-rw---- 1 mqm mqm 4096 2014-02-24 07:56 TEST!Q

- Notice that the queue TEST.Q has a file that is called TEST!Q.
- This might seems strange to Unix and Windows users to have in the file name, an exclamation point (!) instead of a period (.).
- However it is necessary due to the multiplatform support in MQ (iSeries, z/OS).

- Even though both queues have only 1 message, with the text “this is a test”,
- the file size for TEST!Q (protected by AMS) is larger, 4,096 bytes
- than for Q1 (normal Queue) 2,560 bytes

Notes: MQ Admin sees encrypted data

notes

- User 'mqm' issues Unix command to display the contents of the file, but data is encrypted (that is, cannot see the text “this is a test').

- mqm$ **cat 'TEST.Q'**
- + begin output
- AQQF°¸$Äÿÿÿ
- ÿÿÿÿÿÿÿÿÿÿÿÿLLQTEST.Q
- 2014-02-21 12.56.21 @ ÿÉ;ÿÿÿÿ2014-02-21 12.56.21 PÿÉ;ýÿÿÿýÿÿÿÿAQRHÿÿÿÿÿ
- ÿÿÿÀÖAMQ QM_VERIFY_AM9S, ÿÿÿÿ&ó°3qÿÿÿÿMD "¸ QM_VERIFY_AMS
- alice amqsput 20140224125220 36 ÿÿPDMQh¸MQS [0W1Ï0Ì050)1 0j *H÷
- IBM1 0 ;Ò\+¤´c9ë ã>¬õ´]ÂÑ\Ï0 S Ï,é(7¹ÇNÚÎYzÿÀqVñTì#×GÁGhsÙ´tQ3êZe²Ø
- ~¾ëµôÎ DÏnó>*~Ó]´4@{Z)%p q$h*ÄFgLÝç¤Møµ±ÏQãC+¦~Ïsdî2:FkÎr{Ur
- jº½:Ht@VÄÍssQCHíøÜ:ÝG@Ä9íÿ¥@þQæ!ÛþóôGNP9ÓÐOM,Ëntt²kS}íêfmkal?
- _:8GÊ1GÒ¯îöÏi¥Bºýbê©2§HªJ%I 6Ã]È¡Ë$-omþ=QYU]¡
- ÐpAS³¶oª:ú:YVTUk±¼ÐtS°Áñ3~»¦±zV#èÒ²?¤H{:Éjí°4Ï_ÁS¿
- à#$ª{õÜ'«çµ1Ì¶áØúÕIucMúÎøïù3·Y!ÕÇØ7Ä½
- £FH.Á)£·¯ÎxõÒÖºÝÓÌamqzfuma 20140221175642 27
- ÿÿ>zfdLISTENER.1444
- + end output

Using multiple windows

- Because this scenario describes the tasks done by multiple users on the same machine, to reduce confusion it is best to create at least three (3) separate command prompt windows:

- Window 1: for users “root”, “mqm”, “fulano”
- Window 2: for user “alice”
- Window 3: for user “bob”

Installation

- Window 1: You need to log in as user “root” to install the MQ filesets.

- The following free redbook has an overview of the installation steps.
- http://www.redbooks.ibm.com/redpieces/abstracts/sg248087.html?Open
- WebSphere MQ V7.1 and V7.5 Features and Enhancements

Installation - filesets

- Specifically the section:
- Section 16.1 (Page 232) WebSphere MQ Advanced Message Security installation

- The following names of the AMS packages on UNIX and Linux are used:
- AIX: mqm.ams.rte
- HP-UX: MQSERIES.MQM-AMS
- Linux: MQSeriesAMS
- Solaris: mqams

Creating queue manager for testing

- Window 1: Log in as user “mqm”
- Create the queue manager.
- **crtmqm -u DLQ QM_VERIFY_AMS**

- The -u flag indicates which queue is going to be the dead letter queue (DLQ).
- Hint: Many MQ Explorer users hide the SYSTEM* queues and thus, if you use as the DLQ the name SYSTEM.DEAD.LETTER.QUEUE, then it will be hidden and you might not notice if there are messages in the dead letter queue

Configure queue manager

- Start the queue manager
- $ **strmqm QM_VERIFY_AMS**

- Configure the queue manager
- $ **runmqsc QM_VERIFY_AMS**

- # Define testing queue to be protected by AMS
- **define qlocal(TEST.Q)**
- # Define a normal queue which will NOT be protected by AMS
- **define qlocal(Q1)**

Notes: additional configuration

notes

- # Define a listener. It is a good idea to specify the port number in the name in that
- way a quick look at the list of listeners will tell you the port number right away.
- The default port is 1414, however here the port 1444 will be used instead.
- **define listener(LISTENER.1444) trptype(tcp) control(qmgr) port(1444)**
- **start listener(LISTENER.1444)**

- # Define a channel to be used by a remote MQ Explorer
- **define channel(SYSTEM.ADMIN.SVRCONN) chltype(SVRCONN)**

- # Define the DLQ
- **define qlocal(DLQ) like(SYSTEM.DEAD.LETTER.QUEUE)**

- # For MQ 7.1 and 7.5 and if desiring to allow remote connections by an MQ
- Administrator, to avoid return code 2035:

- **set CHLAUTH(*) TYPE(BLOCKUSER) USERLIST('nobody','*MQADMIN')**
- **set CHLAUTH(SYSTEM.ADMIN.*) TYPE(BLOCKUSER) USERLIST('nobody')**

Notes: miscellaneous

notes

- # Display the attribute SPLCAP which shows if AMS is enabled
- **display qmgr SPLCAP**
- AMQ8408: Display Queue Manager details.
- QMNAME(QM_VERIFY_AMS) **SPLCAP(ENABLED)**

- # Display the 2 system queues used by AMS
- **display ql(SYSTEM.PROTECTION*)**
- AMQ8409: Display Queue details.
- QUEUE(**SYSTEM.PROTECTION.ERROR.QUEUE**) TYPE(QLOCAL)
- AMQ8409: Display Queue details.
- QUEUE(**SYSTEM.PROTECTION.POLICY.QUEUE**) TYPE(QLOCAL)

Creating and authorizing users

- Window 1: Log in as user “root”.
- Use line commands (useradd) or the YAST GUI or another administrative tool to create the following users:
- alice - will have access to AMS to put messages
- bob - will have access to AMS to get messages
- fulano - will NOT have access to AMS, and it will be used to show what happens when an unauthorized user tries to browse the AMS protected messages.
- Users are members of the group “mqusers”

Authorizing users for normal MQ access

- Window 1: Login as an MQ administrator
- Ensure to set the environment for MQ 7.5, such as:
- **. /opt/mqm/bin/setmqenv -n Installation1**
- Authorize users to connect to the queue manager
- This is “normal” (non-AMS) activity.
- $ **setmqaut -m QM_VERIFY_AMS -t qmgr -p alice -p bob -p fulano +connect +inq +dsp**

Authorizing users for normal MQ access

- Authorize users to work with the queue TEST.Q:

- User alice to put messages:
- $ **setmqaut -m QM_VERIFY_AMS -n TEST.Q -t queue -p alice -p fulano +put +browse +dsp**

- User bob to get messages:
- $ **setmqaut -m QM_VERIFY_AMS -n TEST.Q -t queue -p bob -p fulano +get +browse +dsp**

Authorizing users for AMS access

- Allow alice and bob to browse the AMS system policy queue and put messages on the AMS error queue. (User fulano does not have AMS access)
- **$ setmqaut -m QM_VERIFY_AMS -t queue -n SYSTEM.PROTECTION.POLICY.QUEUE -p alice -p bob +browse**
- **$ setmqaut -m QM_VERIFY_AMS -t queue -n SYSTEM.PROTECTION.ERROR.QUEUE -p alice -p bob +put**

Notes: Verifying that alice can put message

notes

- NOTE: At this time, TEST.Q is a “normal” queue and it is NOT yet under AMS control
- Window 2: Log in as user “alice”
- Select to work with the MQ 7.5 environment:
- Put a message
- $ **amqsput TEST.Q QM_VERIFY_AMS**
- Sample AMQSPUT0 start
- target queue is TEST.Q
- test-AMS
- Sample AMQSPUT0 end

Notes: Verifying that bob can get message

notes

- NOTE: At this time, TEST.Q is a “normal” queue and it is NOT yet under AMS control
- Window 3: Log in as user “bob”
- Select to work with the MQ 7.5 environment:
- Get a message
- $ **amqsget TEST.Q QM_VERIFY_AMS**
- Sample AMQSGET0 start
- message <test-AMS>
- no more messages
- Sample AMQSGET0 end

Creating key database and certificates

- To encrypt the message, the AMS interceptors require the public key of the sending users.
- Thus, the key database of user identities mapped to public and private keys must be created.

- In this scenario, we are using self-signed certificate which can be created without using a Certificate Authority.
- For production systems, it is advisable not to use self-signed certificates however instead rely on certificates signed by a Certificate Authority.

Notes: Creating key database for alice

notes

- Window 2 (alice):
- The umask used in this example is: 0022
- $ **umask**
- 0022

- Create a new key database for user alice
- The -p flag will create intermediate directories, if they do not yet exist. It is useful when dealing a deep dir tree.
- $ **mkdir /home/alice/.mqs -p**

- $ **runmqakm -keydb -create -db /home/alice/.mqs/alicekey.kdb -pw passw0rd -stash**

Notes: Creating key database for alice

notes

- The following are the directories and files that were created:

- $ **ls -dl /home/alice/.mqs**
- drwxr-xr-x 2 alice mqusers 4096 2014-02-24 06:27 /home/alice/.mqs

- $ **ls -l /home/alice/.mqs**
- -rw------- 1 alice mqusers 88 2014-02-24 06:27 alicekey.crl
- -rw------- 1 alice mqusers 88 2014-02-24 06:27 alicekey.kdb
- -rw------- 1 alice mqusers 88 2014-02-24 06:27 alicekey.rdb
- -rw------- 1 alice mqusers 129 2014-02-24 06:27 alicekey.sth

Notes: Creating key database for alice

notes

- Create a self-signed certificate identifying the user alice for use in encryption

- $ **runmqakm -cert -create -db /home/alice/.mqs/alicekey.kdb -pw passw0rd -label Alice_Cert -dn "cn=alice,o=IBM,c=GB" -default_cert yes**

- Notice the increase in size for alicekey.kdb
- $ **ls -l /home/alice/.mqs**
- -rw------- 1 alice mqusers 88 2014-02-24 06:27 alicekey.crl
- -rw------- 1 alice mqusers **5088** 2014-02-24 06:30 **alicekey.kdb**
- -rw------- 1 alice mqusers 88 2014-02-24 06:27 alicekey.rdb
- -rw------- 1 alice mqusers 129 2014-02-24 06:27 alicekey.sth

Notes: Creating key database for bob

notes

- Window 3 (bob):
- The umask used in this example is: 0022
- $ **umask**
- 0022

- Create a new key database for user bob
- $ **mkdir /home/bob/.mqs -p**

- $ **runmqakm -keydb -create -db /home/bob/.mqs/bobkey.kdb -pw passw0rd -stash**

Notes: Creating key database for bob

notes

- Create a self-signed certificate identifying the user bob for use in encryption
- $ **runmqakm -cert -create -db /home/bob/.mqs/bobkey.kdb -pw passw0rd -label Bob_Cert -dn "cn=bob,o=IBM,c=GB" -default_cert yes**

Creating keystore.conf

- You must point WebSphere MQ Advanced Message Security interceptors to the directory where the key databases and certificates are located. This is done via the keystore.conf file, which hold that information in the plain text form.
- Each user must have a separate keystore.conf file. This step should be done for both alice and bob.

Creating keystore.conf

- The content of keystore.conf must be of the form:

- cms.keystore = <dir>/keystore_file
- cms.certificate = certificate_label

- Notes:
- The path to the keystore file must be provided with no file extension.
- **HOME/.mqs/keystore.conf** is the default location where AMS searches for the keystore.conf file.

Notes: Creating keystore.conf for alice

notes

- Window 2 (alice)

- Create file:
- **$ vi /home/alice/.mqs/keystore.conf**
-
- The contents is:

- cms.keystore = /home/alice/.mqs/alicekey
- cms.certificate = Alice_Cert

Notes: Creating keystore.conf for bob

notes

- Window 3 (bob)

- Create file:
- **$ vi /home/bob/.mqs/keystore.conf**
-
- The contents is:

- cms.keystore = /home/bob/.mqs/bobkey
- cms.certificate = Bob_Cert

Sharing Certificates

- It is necessary to share the certificates between the two key databases so that each user can successfully identify each other.

- Because in this scenario the users are located in the same host, the directory /tmp will be used as the neutral directory to exchange the certificates between the users.
- But if they were located in different boxes, then you will need to use ftp and specify the file transfer as “ascii”.

Notes: Sharing Certificates for alice

notes

- Window 2 (alice)

- Export the certificate identifying alice to a file located in /tmp
- The resulting file will be written as ascii text, which is the default (-format ascii).

- $ **runmqakm -cert -extract -db /home/alice/.mqs/alicekey.kdb -pw passw0rd -label Alice_Cert -target /tmp/alice_public.arm**

- Allow the certificate to be read by others
- $ **chmod 644 /tmp/alice_public.arm**
- $ **ls -l /tmp/*.arm**
- -rw-r--r-- 1 alice mqusers 692 2014-02-24 06:55 /tmp/alice_public.arm

Notes: Sharing Certificates for alice

notes

- Window 3 (bob)
- Add the certificate from alice into bob's keystore:
- $ **runmqakm -cert -add -db /home/bob/.mqs/bobkey.kdb -pw passw0rd -label Alice_Cert -file /tmp/alice_public.arm**
- Notice size increase for bobkey.kdb:
- $ **ls -l /home/bob/.mqs**
- -rw------- 1 bob mqusers 88 2014-02-24 06:36 bobkey.crl
- -rw------- 1 bob mqusers **10088** 2014-02-24 06:56 bobkey.kdb

Notes: Sharing Certificates for bob

notes

- Window 3 (bob)
- Export the certificate identifying bob to a file located in /tmp:
- $ **runmqakm -cert -extract -db /home/bob/.mqs/bobkey.kdb -pw passw0rd -label Bob_Cert -target /tmp/bob_public.arm**
- Allow the certificate to be read by others
- $ **chmod 644 /tmp/bob_public.arm**
- $ **ls -l /tmp/*.arm**
- -rw-r--r-- 1 alice mqusers 692 2014-02-24 06:55 /tmp/alice_public.arm
- -rw-r--r-- 1 bob mqusers 684 2014-02-24 06:58 /tmp/bob_public.arm

Notes: Sharing Certificates for bob

notes

- Window 2 (alice)
- Add the certificate from bob into alice's keystore:
- $ **runmqakm -cert -add -db /home/alice/.mqs/alicekey.kdb -pw passw0rd -label Bob_Cert -file /tmp/bob_public.arm**
- **Notice size increase for alicekey.db:**
- **$ ls -l /home/alice/.mqs**
- -rw------- 1 alice mqusers 88 2014-02-24 06:27 alicekey.crl
- -rw------- 1 alice mqusers **10088** 2014-02-24 07:00 alicekey.kdb

Defining queue policy

- Let's define protection policies using the “setmqspl” command.
- Each policy name must be the same as the queue name it is to be applied to.

Notes: Defining queue policy

notes

- Window 1 (mqm)

- This is an example of a policy defined for the TEST.Q queue.
- The messages are signed by the user alice using the SHA1 algorithm, and encrypted using the 256-bit AES algorithm.
- The user alice is the only valid sender and the user bob is the only receiver of the messages on this queue:

- $ **setmqspl -m QM_VERIFY_AMS -p TEST.Q -s SHA1 -a "CN=alice,O=IBM,C=GB" -e AES256 -r "CN=bob,O=IBM,C=GB"**

- Note: The DNs need to match exactly those specified in the receptive user's certificate from the key database.

Notes: Defining queue policy

notes

- Verify the policy:

- $ **dspmqspl -m QM_VERIFY_AMS**
- Policy Details:
- Policy name: TEST.Q
- Quality of protection: PRIVACY
- Signature algorithm: SHA1
- Encryption algorithm: AES256
- Signer DNs:
- CN=alice,O=IBM,C=GB
- Recipient DNs:
- CN=bob,O=IBM,C=GB
- Toleration: 0

Notes: Defining queue policy - backup

notes

- Best Practice:

- The command “dmpmqcfg” was introduced in MQ 7.1 to provide a way to export the object definitions and authorizations of a queue manager, which could be used as backup or as a way to create a new queue manager with the same object definitions and authorizations.
- This is similar to the function provided by the MQ V6 and MQ 7.0 SupportPac MS03 “saveqmgr” (does not work with 7.1+)
- However, dmpmqcfg does NOT export the AMS Policies.
- Thus, in case that you want to export these AMS Policies into a file that is part of backup procedures, you can use the command “dspmqspl” with the -export flag, as shown in the next slide.

Notes: Defining queue policy - backup

notes

- Best Practice:
- Export the policies in the form of of a setmqspl command that could be executed later, in case that you need to restore the policies:
- $ **dspmqspl -m QM_VERIFY_AMS -export > restore_AMS_policies.bat**
- Display the output file:
- $ **cat restore_AMS_policies.bat**
- setmqspl -m QM_VERIFY_AMS -p TEST.Q -s SHA1 -a "CN=alice,O=IBM,C=GB" -e AES25r "CN=bob,O=IBM,C=GB" -t 0

Basic testing of the setup - put

- Let's test the setup by putting a message as user alice and reading the message as user bob.

- Window 2: User alice
- Put a message using a sample application.
- Type the text of the message, then press Enter.

- $ **amqsput TEST.Q QM_VERIFY_AMS**
- Sample AMQSPUT0 start
- target queue is TEST.Q
- **this is a test**
- Sample AMQSPUT0 end

Basic testing of the setup - get

- Window 3: User bob
- Get a message using a sample application:

- $ **amqsget TEST.Q QM_VERIFY_AMS**
- Sample AMQSGET0 start
- message <**this is a test**>
- no more messages
- Sample AMQSGET0 end

- **Conclusion:** User alice was able to put a message and bob was able to read it. Yeah!

Testing encryption

- To verify that the encryption is occurring as expected, create an alias queue which references the original queue TEST.Q.
- This alias queue will have no security policy and so no user will have the information to decrypt the message and therefore the encrypted data will be shown

Notes: Testing encryption - alias queue

notes

- Window 1 (mqm)
- Create an alias queue
- $ **runmqsc QM_VERIFY_AMS**
- **DEFINE QALIAS(TEST.ALIAS) TARGET(TEST.Q)**
- **end**

- Grant bob access to browse from the alias queue
- $ **setmqaut -m QM_VERIFY_AMS -n TEST.ALIAS -t queue -p bob +browse**

- Window 2 (alice)
- Put another message:
- $ **amqsput TEST.Q QM_VERIFY_AMS**

Notes: Testing encryption - alias queue

notes

- Window 3 (User bob) browse the message via the alias queue:
- $ **amqsbcg TEST.ALIAS QM_VERIFY_AMS**

- The output from amqsbcg application shows the encrypted data, proving that the message has been encrypted:

- + begin output
- MQOPEN - 'TEST.ALIAS'
- MQGET of message number 1, CompCode:0 Reason:0
- ****Message descriptor****
- StrucId : 'MD ' Version : 2
- Report : 0 MsgType : 8
- ...

Notes: Testing encryption - alias queue

notes

- **** Message ****
- length - 1238 of 1238 bytes

```
00000000:  5044 4D51 0200 0200 6800 0000 6800 0000          'PDMQ....h...h...'
00000010:  0800 0000 B804 0000 0E00 0000 0000 0000          '................'
00000020:  4D51 5354 5220 2020 0000 0000 0000 0000          'MQSTR   ........'
00000030:  0000 0000 0000 0000 2020 2020 2020 2020          '........        '
00000040:  2020 2020 2020 2020 2020 2020 2020 2020          '                '
00000050:  2020 2020 2020 2020 2020 2020 2020 2020          '                '
00000060:  2020 2020 2020 2020 3082 046A 0609 2A86          '        0..j..*.'
00000070:  4886 F70D 0107 03A0 8204 5B30 8204 5702          'H.........[0..W.'
...
000004A0:  D99F 93C9 1E16 535D 2B30 C141 495B 7548          'Ù....S]+0.AI[uH'
000004B0:  7264 7CFB A471 D6DD 6576 36BC FA99 5D95          'rd|..q..ev6...].'
000004C0:  0148 B66D 4FD4 B581 B1AB 63CC 11F8 5A0F          '.H.mOÔµ...c...Z.'
000004D0:  F0C2 6B2C E73E                                   '..k,.>          '
```

- No more messages
- MQCLOSE
- + end output

Testing unauthorized access - user fulano, 1

- Window 1: User fulano
- Log in as user fulano and ensure to set up the environment for using MQ 7.5.
- Try to put, browse or get a message from the queue. These actions will fail.
- Even though the setmqaut was given for user fulano to get messages from the queue TEST.Q, the AMS policies do not include user fulano as an authorized user.

Testing unauthorized access - user fulano, 2

- $ **amqsput TEST.Q QM_VERIFY_AMS**
- Sample AMQSPUT0 start
- target queue is TEST.Q
- MQOPEN ended with reason code 2035
- unable to open queue for output
- Sample AMQSPUT0 end

- $ **amqsbcg TEST.Q QM_VERIFY_AMS**
- AMQSBCG0 - starts here
- MQOPEN - 'TEST.Q'
- MQOPEN failed with CompCode:2, Reason:2035

- The Reason is: MQRC_NOT_AUTHORIZED

Notes: Looking at the error logs

notes

- Let's look at the error messages for the queue manager
- $ **cd /var/mqm/qmgrs/QM_VERIFY_AMS/errors**
- $ **tail AMQERR01.LOG**

- 02/24/2014 07:42:50 AM - Process(6877.1) User(fulano) Program(amqsput)
- Host(veracruz) Installation(Installation2)
- VRMF(7.5.0.3) Qmgr(QM_VERIFY_AMS)
- AMQ9062: The WebSphere MQ security policy interceptor could not read the keystore configuration file: /home/fulano/.mqs/keystore.conf.
- EXPLANATION:
- The WebSphere MQ security policy interceptor could not read the keystore
- configuration file: /home/fulano/.mqs/keystore.conf.
- ACTION:
- Make sure that the user who executes the WebSphere MQ application has
- permissions to read the configuration file. Check if the configuration file is
- not corrupted or empty.

Notes: MQ Admin sees encrypted data

notes

- User 'mqm' issues Unix command to display the contents of the file, but data is encrypted (that is, cannot see the text “this is a test').

- mqm$ **cat 'TEST.Q'**
- + begin output
- AQQF°¸$Äÿÿ
- ÿÿÿÿÿÿÿÿÿÿÿÿLLQTEST.Q
- 2014-02-21 12.56.21 @ ÿÉ;ÿÿÿÿ2014-02-21 12.56.21 PÿÉ;ýÿÿÿýÿÿÿÿAQRHÿÿÿÿÿ
- ÿÿÿÀÖAMQ QM_VERIFY_AM9S, ÿÿÿÿ&ó°3qÿÿÿÿMD "¸ QM_VERIFY_AMS
- alice amqsput 2014022412522036 ÿÿPDMQh¸MQS [0W1Ï0Ì050)1 0j *H÷
- IBM1 0 ;Ò\+¤´c9ë ã>¬õ´]ÂÑ\Ï0 S Ï,é(7¹ÇNÚÎYzÿÀqVñTì#×GÁGhsÙ´tQ3êZe²Ø
- ~¾ëµôÎ DÏnó>*~Ó]´4@{Z)%p q$h*ÄFgLÝç¤Møµ±ÏQãC+¦~Ïsdî2:FkÎr{Ur
- jº½:Ht@VÄÍssQCHíøÜ:ÝG@Ä9íÿ¥@þQæ!ÛþóôGNP9ÓÐOM,Ëntt²kS}íêfmkal?
- _:8GÊ1GÒ¯îöÏi¥Bºýbê©2§HªJ%I 6Ã]È¡Ë$-omþ=QYU]¡
- ÐpAS³¶oª:ú:YVTUk±¼ÐtS°Áñ3~»¦±zV#èÒ²?¤H{:Éjí°4Ï_ÁS¿
- à#$ª{õÜ'«çµ1Ì¶áØúÕIucMúÎøïù3·Y!ÕÇØ7Ä½
- £FH.Á)£·¯ÎxõÒÖºÝÓÌamqzfuma 2014022117564227
- ÿÿ>zfdLISTENER.1444
- + end output

Testing unauthorized access - bob, alice

- Only one AMS policy has been created in this test.
- In this policy user "alice" was explicitly indicated as a "signer" and user "bob" was indicated as a "reader".

- Let's explore the following scenario, which is NOT covered by the above policy:
- user "bob" puts a message as a signer and user "alice" tries to read it.

- Because there is no explicit policy for this case, the error message that we get will be 2063:
- 2063 0x0000080f MQRC_SECURITY_ERROR

Notes: Testing unauthorized access - 2

notes

- Window 3 (bob) - put a message into TEST.Q. This is successful.
- The message is encrypted and placed encrypted in the queue.
- $ amqsput TEST.Q QM_VERIFY_AMS

- Window 2 (alice) - try to browse message from TEST.Q. This is not successful.
- $ amqsbcg TEST.Q QM_VERIFY_AMS
- AMQSBCG0 - starts here
- MQGET 1, failed with CompCode:2 Reason:2063

- See queue manager error log to get more details:
- AMQ9017: WebSphere MQ security policy internal error: message could not be
- unprotected: GSKit error code 851968, reason 43.
- EXPLANATION:
- The WebSphere MQ security policy interceptor could not verify or decrypt a
- message because the indicated GSKit error occurred. This can happen for several
- reasons, all of which are internal failures:
- (1) the message is not a valid PKCS#7 message;
- (2) the sender's certificate does not have the required key
- usage bit to be able to encrypt the message;
- (3) the sender's certificate was not recognized as a trusted certificate;
- **(4) receiver is not among the recipients of the message. !!!!! This is the case for our test**

Testing unauthorized access - bob, bob

- This other scenario may seem a bit strange: unless there is a policy in place, user bob CANNOT browse the encrypted messages generated by himself!
- user "bob" puts a message as a signer and the same user "bob" tries to read it.
- Because there is no explicit policy for this case, the error message that we get will be 2063:
- 2063 0x0000080f MQRC_SECURITY_ERROR

Notes: Testing unauthorized access - 3

notes

- Window 3 (bob) - put a message into TEST.Q. This is successful.
- The message is encrypted and placed encrypted in the queue.
- $ amqsput TEST.Q QM_VERIFY_AMS

- Window 3 (bob) - try to browse message from TEST.Q. This is not successful.
- $ amqsbcg TEST.Q QM_VERIFY_AMS
- AMQSBCG0 - starts here
- MQGET 1, failed with CompCode:2 Reason:2063

- See queue manager error log to get more details:

- AMQ9035: Message signer is not in the list of authorised signers.
- EXPLANATION:
- The WebSphere MQ security policy interceptor detected that the message is
- signed by an unauthorised party.
- ACTION:
- Establish whether the identity associated with the sender of the message is
- authorized to send messages to this application. Ensure the sender is named in
- the list of allowed signers on the security policy for the queue.

Additional WebSphere Product Resources

- Learn about upcoming WebSphere Support Technical Exchange webcasts, and access previously recorded presentations at:
  http://www.ibm.com/software/websphere/support/supp_tech.html
- Discover the latest trends in WebSphere Technology and implementation, participate in technically-focused briefings, webcasts and podcasts at:
  http://www.ibm.com/developerworks/websphere/community/
- Join the Global WebSphere Community:
  http://www.websphereusergroup.org
- Access key product show-me demos and tutorials by visiting IBM Education Assistant:
  http://www.ibm.com/software/info/education/assistant
- View a webcast replay with step-by-step instructions for using the Service Request (SR) tool for submitting problems electronically:
  http://www.ibm.com/software/websphere/support/d2w.html
- Sign up to receive weekly technical My Notifications emails:
  http://www.ibm.com/software/support/einfo.html

Connect with us!

1. **Get notified on upcoming webcasts**
   Send an e-mail to wsehelp@us.ibm.com with subject line “wste subscribe” to get a list of mailing lists and to subscribe
2. **Tell us what you want to learn**
   Send us suggestions for future topics or improvements about our webcasts to wsehelp@us.ibm.com

Questions and Answers