Certificate Management or:

How I learned to stop worrying and love the expiry bomb

T.Rob Wyatt

t.rob@ioptconsulting.com

Change is the only constant

This presentation reflects…

- **My current opinions regarding WMQ security**
- **The product itself continues to evolve (even in PTFs)**
- **Attacks only get better with time**
- **This version of the presentation is based on WebSphere MQ v7.1 & v7.5**
- **This content will be revised over time so please be sure to check for the latest version at https://t-rob.net/links**
- **Your thoughts and ideas are welcome**

WebSphere MQ Security Presentation Series

- This presentation is part of a series authored by T.Rob Wyatt. The introductory decks are available through https://t-rob.net.
- For the most current version of this deck, please see https://t-rob.net/links or contact the author presentations@t-rob.net.
- Ask me about on-site and remote training tailored for your organization.

Why Certificate Management?

- **WebSphere MQ's native authentication is based on certificates.**
  - At the very least, administrators probably need to use certs.
  - If using exits to exchange credentials, the channel should be encrypted.
  - Non-interactive connections (apps & other QMgrs)

At IMPACT we had 25 slots for WebSphere Messaging content. Here we have 70! But that means we will need to get a feel for the level of presentation that best suits this audience – please provide feedback as to content!

Also, the sessions I'm presenting have fewer slides because we always tend to run out of time for Q&A.

If you wanted more written content & less Q&A, please let me know!

Agenda

- What is a certificate anyway?
- Why sign certificates?
- Self-signed or CA-signed, what's the difference?
- How does certificate authentication work?
- Start your own Center of Mediocrity using these Security Worst Practices!
- Digital Locksmith 101: GSKit, Keytool & OpenSSL
- Workin' on the (cert) chain gang
- Revocation is not a book of scripture (but it should be)
- "He's dead, Jim!" No, he's just expired.

What is a certificate anyway?

- **Certificate is a container for…**
  - Asymmetric key pair
  - Identity attributes
  - Policy attributes
- **Asymmetric key pair is crypto primitive that supports privacy and integrity.**
- **Identity attributes support authentication and non-repudiation.**
- **Policy attributes strengthen the certificate security model.**

Why sign certificates?

- **Binds the certificate attributes and the keys together.**
- **The public key and certificate attributes are hashed to form a unique fingerprint.**
  - Same key pair with different attributes or policy results in different fingerprint.
  - CA's job to guard against someone with same key pair registering a duplicate cert.
- **A bare key can encrypt and integrity check.**
- **A certificate adds identity and policy.**
- ***The signature ensures that these are not modified after issuance*.**

Self-signed or CA-signed, what's the difference?

- **A certificate has a public key.**
- **A certificate is signed using a private key.**
  - If the private key is from the same certificate, the result is self-signed.
  - If the private key is from another certificate, the result is CA signed.
- **Which means that…**
  - A certificate signed by a company's internal CA is *not* self-signed!
  - The signer of a CA-signed certificate is not necessarily a commercial CA!
  - The terminology describes a specific technical aspect of the certificate and not an organizational relationship of signer and signee.
- **Self-signed certs have the same value in the Issuer Distinguished Name as in the Subject Distinguished name.**
- **Every CA root certificate is a self-signed certificate.**

How does certificate authentication work?

Start your own Center of Mediocrity using these Security Worst Practices!

- **Define your certificates in one place, then move them to where they will be used. Use email if you want to be really unsecure.**
- **Use the same certificate to represent multiple identities.**
- **Make sure the certificate files are group- and world-readable.**
- **Run your own CA without properly securing it.**
  - On someone's laptop.
  - No control over dupe DN's, policy restrictions, usage restrictions.
  - Don't use intermediate signers.
  - Same signers for Prod and non-Prod.
  - Put lots of signer roots in the KDB or JKS.
- **Accept self-signed certs without checking path length & IsCA flag.**
  - Special category for accepting any cert without dumping it first.

Digital Locksmith 101: GSKit, Keytool & OpenSSL

- **Different tools with different distributions.**
  - GSKit with WMQ, WMQ Client, Broker, etc.
  - Keytool with Java-based products like FTE/MFT.
  - OpenSSL is part of the UNIX/Linux standard distribution.
- **All use X.509 certs**
- **Keytool talks to JKS, JCEKS and standard PKCS formats.**
- **OpenSSL talks to standard PKCS formats.**
- **GSKit talks to these plus the proprietary KDB format.**
- **Standard-based PEM file = GSKit ARM file.**

Workin’ on the (cert) chain gang

- **Walking the cert chain:**
  - Dump a certificate to determine the signer.
  - Self-signed? No? Continue.
  - Dump the signer to determine its signer.
  - Continue until reaching a self-signed root.
- **The self-signed root must always be in the KDB/Trust Store.**
- **Depending on your version of WMQ, the intermediate certs may or may not need to be in the trust store. Assume always that they do.**

```
runmqakm –cert –details –db key.kdb –stashed –label twyatt

cat cert.pem | openssl –text –noout

keytool -list -v -keystore key.jks -alias twyatt
```

Revocation is not a book of scripture (but it should be)

- **Certificates are about provisioning access.**
  Revocation is about taking it away.
- **Imagine a hotel where they never change the door key code.**
  - Previous guests who stayed in your room can walk in at any time.
  - Hotel staff cannot tell without close inspection who is the legitimate guest.
  - Imposter can impersonate you and run up room service and spa charges.
  - Guest who left on bad terms can return and trash the room.
  - Current guest can repudiate the damages (“It wasn’t me!”) with impunity.
- **As the guest, it’s not so bad so long as you personally are not damaged.**
  - Just say “it wasn’t me!” and hope nobody shows up while you are present.
- **As the MQ admin, you are the hotel manager, not the guest.**
- **CHLAUTH will assist until real revocation can be implemented.**

“He’s dead, Jim!” No, he’s just expired.

- **You can check your local QMgr cert expiry with a script.**
- **What happens when the remote QMgr’s or app’s cert expires?**
  - Nothing. As in no messages move, no channels start, nothing.
- **Your business partner doesn’t have a clue what’s wrong or how to fix it.**
- **But *you* take a hit on revenue and possibly customer satisfaction, SLAs.**
- **Anything we can to do prevent this situation?**
- **Absolutely! WMQ uses the standard TLS so OpenSSL is your friend.**

He’s dead Jim! - continued

- **TLS handshake includes server side sending its cert to client.**
  - Client in this case means “thing that requested the connection” – might be a QMgr.
- **So we can initiate the TLS handshake to get the server’s certificate.**
- **But abandoning the socket before the handshake completes generates errors and an FDC file. Oops! Can’t do that in Production! (We hope.)**
- **Need a way to make sure the connection fails gracefully.**
  - Sending an unrecognized cert will cause a failure.
  - Sending an expired cert will cause a failure.
  - Belt & Suspenders – send an expired, unrecognized cert.
- **Fortunately, this is easy to do. Attached is a cert and script.**

```
#!/usr/bin/perl
use strict;
# ------------------------------------------------------------------------ #
# mqCertRpt.pl
#
# Script to print out the certificates and expiration dates form a cluster
# of QMgrs
# ------------------------------------------------------------------------ #

# Make sure that SupportPac MO72 'mqsc' command is in the PATH.  In this case
# it's in . but best to put it in /opt/mqm/bin and leave $ENV{PATH} alone.
$ENV{PATH} = "$ENV{PATH}:.";

# Shell one-liner that fetches a local QMgr name. If there's more than one it gets
the first one reported.
my $QMgr = `dspmq 2>&1| grep -i '(Running)' | tr ')' '\n' | grep QMNAME | tr '('
'\n' | grep -v QMNAME | head -1`;

if ($QMgr) {        # Did we find one?
    chomp $QMgr;
    print "Found '$QMgr' local queue manager\nStarting certificate report on local
cluster.\n\n";
```

```
# Now go get the list of QMgrs in the cluster.  This is intended to be run from the
repository QMgr.
    # Uses the mqsc executable from SupportPac MO72 that prints output on one line
per object.
    foreach (`echo 'dis clusqmgr(*) cluster(*) CONNAME' | ./mqsc -m $QMgr -p wrap=no
| grep CLUSQMGR`) {

        # Grab the QMgr name and the CONNAME from the mqsc output. Leave out any
trailing ) chars.
        my ($thisQMgr, $thisCONNAME) = /CLUSQMGR\((\S+?)\) .* CONNAME\((\S+?)\)/;

        # We need to convert the CONNAME to host:port for openssl.
        if ($thisCONNAME =~ /\(/) {
            # If the CONNAME has a ( then convert to : and keep the port
            $thisCONNAME =~ s/\(/:/;
        } else {
            # Otherwise add the default port.
            $thisCONNAME .= ":1414";
        }

        # Print the QMgr name and the CONNAME, then run the openssl commands
        print "$thisQMgr Queue Manager hosted at $thisCONNAME\n";
        # Make sure to have a private key to send which will _NOT_ be known to the
QMgr
```

```
# or else openssl will hang and the QMgr will cut an fdC file.
        print `openssl s_client -connect $thisCONNAME -cert key.pem -prexit 2>&1 |
openssl x509 -enddate -issuer -subject -noout`, "\n\n";

    }
} else {
    # Could not find a QMgr. Print output of dspmq for diagnostic.
    print "ERROR: No local running QMgr found.\n\n";
    print `dspmq 2>&1`;
}
```

```
Bag Attributes
    localKeyID: 31 33 37 30 34 36 33 36 39 33 31 35 31
    friendlyName: dummy cert
Key Attributes: <No Attributes>
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAjRt2xZubTfa9LfFAK4++GpXl+49N3fBa4swrj2gxT2QPv8pT
+jJBGo308BljE5UztidAEtgMyQUI412KjrUE9V8vaOcrr3na1krjL/zu8kJauTY9
qGRTBVq8Us94SCTXh/uDWLCz654ZDHOvXKIJvECEK4IsQ0tF+OYF1soh3yn4jCkU
r+IaMORlCoK6paig0IK+0ImjonQjTu5yPM7TxcS3rb964dl1Fo62tdkpW/Jz+xC9
WA4ARxnhA+Wyr9x+HVQzeozfJKAsVX0aRC3iferk/VWq/BEtl4oJ3qUwTkD3yejC
i8yypmo8zsQDsqovyGKlX9p3bSXrKpK1lrySZQIDAQABAoIBAGdNou9aUyGJsMfw
pcT+b/NwqDU5nY0MyMzfIkjTw73WiH0ld8NkhSsWoCoiu9j7UydYptzk1L9Z8z7Z
K0cdYg6SNlO2VCYtyRu/3805vc+H0Aa6f5bek1uRMDHc/17xNuCMgZ/5I5XBI1ul
9j4/T/HFRenPJlbtgoMwMkQBJ1t3zovhX0mWfKFrxthIb25DCr2t/UYBJ9931EE5
3Yzi1Vi7pIk5Dy3L1+SdZ6dCvwwcFSO3Mo7o81ECgYA3JAuVcwkCJ/oehltYFPzw
FHte8HS1dHxCcDomYVz+KCzX3aH0woFsG7cngnpn31uDcJnQhQmQsMLu0knN2y6U
[Lines removed for brevit]
GlOauILwX0cgNO8bfid3lSLUhQ4vgzbyG7jwjCq90aMHuOoLQ440dpwMJ/v5BgJt
tHVKChbvGT4T2uJTdBdzlQKBgQCLdT+DXbtFPJa58CFupubxxWPN/hsjS5quUO+8
vlhEy+6MOfdmnLJ22Jho61SFky1daborQobhNdIOAvG4EvU8sNryDQfmtJfpkqHS
91hwP71EMI2YaJ1NSG3IkDsuAei10Xky2T5dwxzB9KCupdmwj9m8EdjXT0/IVEvz
J//OkQKBgQDkulMlElhKhq/8JxCpx+B/SZeid7Gn9hLR7nYc99y9+qIr3ZCCK3jM
ThPuTTymug3JHZBew59MerdAUc7Ya6iOTS8i7ZO8ut20/GiE5ZFUToTiUVL1n1qK
TUtSsVeMPeZKCIRDsohZj1m+qtcA9UFsBVi/gzcGRLmJN1nwAxkbyA==
-----END RSA PRIVATE KEY-----
```

```
Bag Attributes
    localKeyID: 31 33 37 30 34 36 33 36 39 33 31 35 31
    friendlyName: dummy cert
subject=/CN=Dummy cert designed to fail validation
issuer=/CN=Dummy cert designed to fail validation
-----BEGIN CERTIFICATE-----
MIIC3jCCAcagAwIBAgIEUa+doTANBgkqhkiG9w0BAQsFADAxMS8wLQYDVQQDEyZE
dW1teSBjZXJ0IGRlc2lnbmVkIHRvIGZhaWwgdmFsaWRhdGlvbjAeFw0xMzA2MDUy
MDIwNDlaFw0xMzA2MDYyMDIwNDlaMDExLzAtBgNVBAMTJkR1bW15IGNlcnQgZGVz
aWduZWQgdG8gZmFpbCB2YWxpZGF0aW9uMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8A
MIIBCgKCAQEAjRt2xZubTfa9LfFAK4++GpXl+49N3fBa4swrj2gxT2QPv8pT+jJB
Go308BljE5UztidAEtgMyQUI412KjrUE9V8vaOcrr3na1krjL/zu8kJauTY9qGRT
BVq8Us94SCTXh/uDWLCz654ZDHOvXKIJvECEK4IsQ0tF+OYF1soh3yn4jCkUr+Ia
MORlCoK6paig0IK+0ImjonQjTu5yPM7TxcS3rb964dl1Fo62tdkpW/Jz+xC9WA4A
RxnhA+Wyr9x+HVQzeozfJKAsVX0aRC3iferk/VWq/BEtl4oJ3qUwTkD3yejCi8yy
pmo8zsQDsqovyGKlX9p3bSXrKpK1lrySZQIDAQABMA0GCSqGSIb3DQEBCwUAA4IB
AQA3USqov8n3RPGobXPMmP26jbXcw9ikK8Oj0eES5tw6Hf5xXC+UC8LsfH0+Fzxo
8tIDzTy/hLRUBN/yKo53YQ4y+y7pbjrW3wQHdBck3bNR4mjbrBl7QYxglBnf4FVc
cdUQ5idA+5YWdg/IplBinOVSWzXYj8/2Z2ft71mpxR+8o+9+m2UFGKMb8qNx0STW
N1xiXXOLoUt1+KVvK7WM/4ZHz8VswjMDSGkfeOnIorkZHsEeJW9a92kRehU3M1K/
bjWDqjkwQhB2bc8tskpSWrPmA4c9Ocifrvf3ZLwpgnGikzHKvO2eaaUADXL1aJVG
8e4fCtZVbjRdf+sOUtyef9Q0
-----END CERTIFICATE-----
```

Time for a demo?

- We have several asymmetric keyed demo tools available.
- See if you can spot the weak points and the ways in which they are mitigated by the protocol.
- I need a volunteer.

Questions & Answers