JMSDigger User Guide

Author:

Gursev Singh Kalra
Senior Principal Consultant
Foundstone Professional Services

Table of Contents

Introduction

JMSDigger is a tool used for penetration testing ActiveMQ based JMS Applications and was originally introduced in "*A Pentesters Guide to Hacking ActiveMQ Based JMS Applications*". This user guide provides comprehensive details on JMSDigger's functionality and assumes that you either have prior knowledge of JMS concepts or you are aware about penetration testing techniques for JMS applications.

JMSDigger Features

JMSDigger is an open source, GUI based tool written in Java using JMS API with the following features:

Generic JMS Operations

1. Anonymous authentication check
2. Manual authentication check
3. Automated credential brute force and fuzzing.
4. Retrieve messages from Topics, Queues and Durable Subscribers
5. Create new Durable Subscribers
6. Erase existing Durable Subscribers

ActiveMQ Specific Operations

1. Retrieve ActiveMQ broker and destination statistics
2. Create new destinations (Topics or Queues)
3. ActiveMQ password decryption.

Compiling and Running JMSDigger

JMSDigger source code is available on OpenSecurityResearch's GitHub repository as a Maven[1] project that you can compile:

- https://github.com/OpenSecurityResearch/jmsdigger

A JAR file is also available on McAfee's website that you can download and run in case you decide not to compile the application. To execute the JAR file, run the following command:

```
java -jar jmsdigger-0.1.1.0.jar
```

[1] http://maven.apache.org

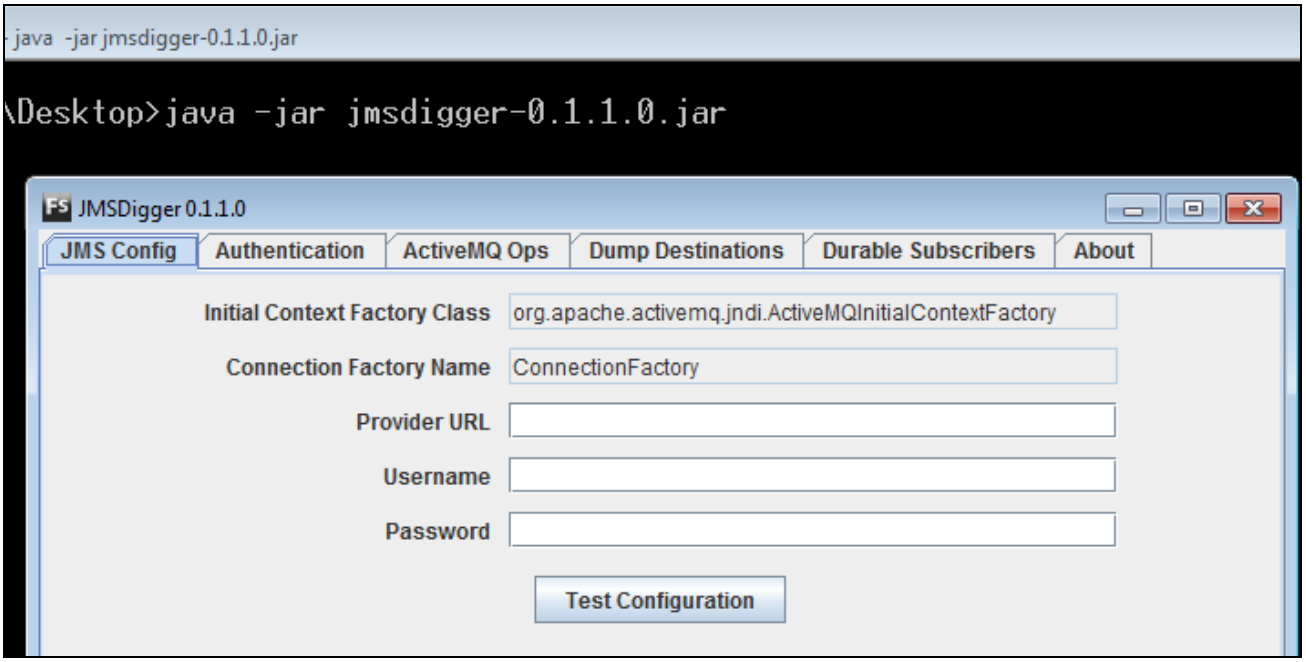

Figure 1: Image shows JMSDigger's landing screen

Bootstrapping JMSDigger

JMSDigger's is configured via the "JMS Config" tab. The first step to engage a JMS application using JMSDigger is to enter JMS broker's (ActiveMQ in this case) configuration as shown in the image below. JMSDigger requires a "Provider URL" containing the protocol, the hostname/IP address and the port number of the broker along with a valid username and password combination to establish a connection and communicate with the broker. Once you provide the required information, click on the "Test Configuration" button and JMSDigger will verify the configuration artifacts and signal success or failure.

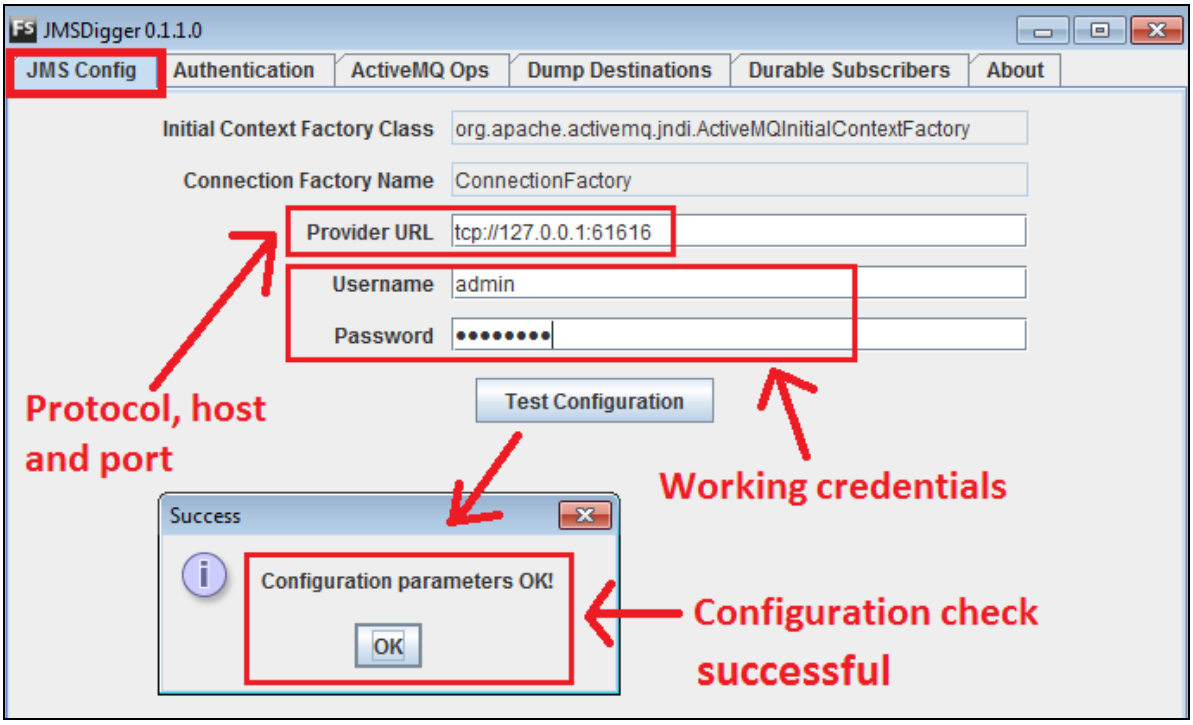

Figure 2: Image shows JMSDigger's Configuration tab

The protocol prefix allows JMSDigger to choose the transport protocol to be used when communicating with the broker. For example, when the protocol component of the URL is set to `tcp`, the protocol used is `OpenWire`; for `amqp`, the protocol is `Advanced Message Queuing Protocol`. The hostname and port components are self-explanatory.

The contents of the "Provider URL" field are shared across all JMSDigger functionalities and JMSDigger will also communicate to the provider supplied in this field. However, it is important to point out that JMSDigger will use the credentials provided in the configuration tab to connect to the message broker for all the functionalities (that require authentication), other than specifically performing Authentication testing done from the "Authentication" Tab. For example, if you want to dump messages from a Topic or a Queue, or you are trying to create new Topic, Queue or Durable Subscriber on the broker, the credentials used to perform those actions will be taken from the configuration tab.

Testing for Anonymous/No Authentication

The "JMS Config" tab allows you to check if the ActiveMQ instance under test supports anonymous authentication. To check for anonymous authentication, leave the username and password fields blank in the "JMS Config" tab and press the "Test Configuration" button. A success or failure signals presence or absence of anonymous authentication respectively.

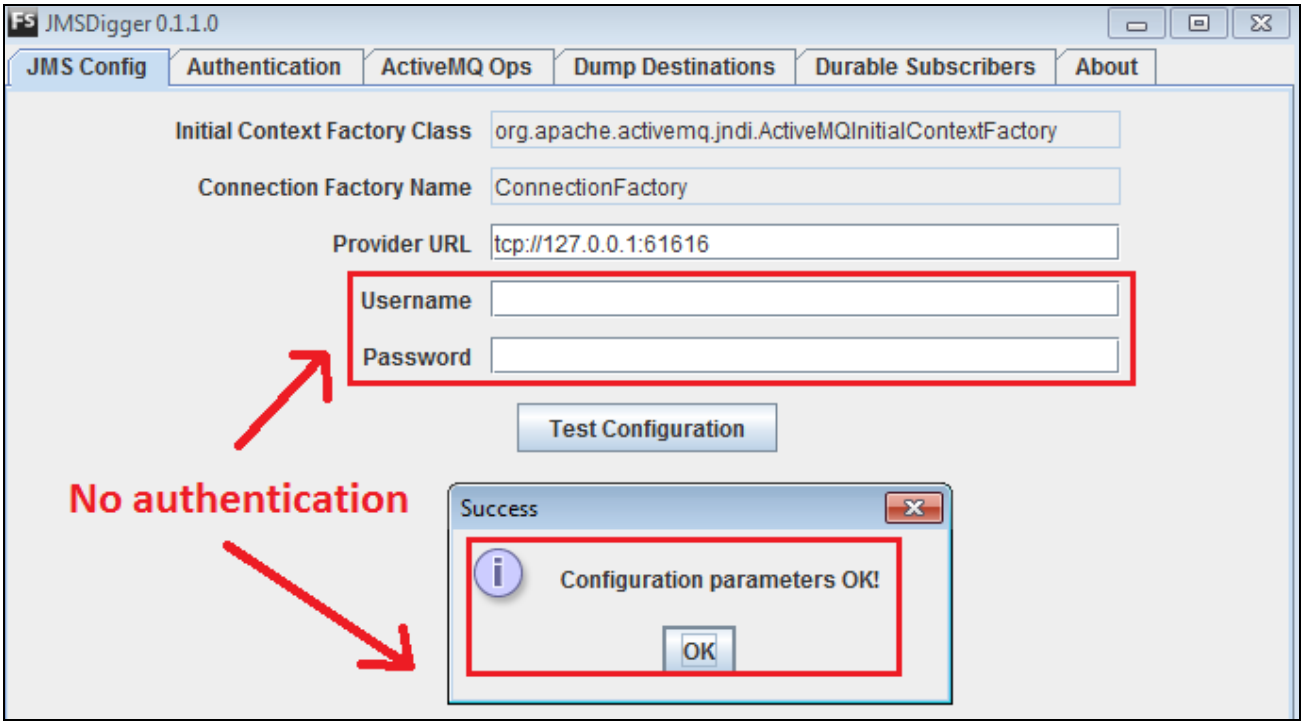

Figure 3: Image shows ActiveMQ instance accepts blank username and password

JMSDigger Error Logs

JMSDigger error logs are written to a file named `jmsdigger.log` inside the `jmsdigger` sub-directory in the current user's home directory. For example, in Windows 7, the logs will be written to `C:\Users\<Username>\jmsdigger\jmsdigger.log`.

Testing Authentication

The "Authentication" tab contains the functionality to perform authentication testing. It is the only tab that does not rely on user credentials from the "JMS Config" tab. The Authentication tab has two modes of operation.

The first mode allows you to test one set of credentials at a time provided in the "Credentials Check" control group against the message broker. Typical uses of this mode are fine tuning injection vectors for SQL injection, LDAP injection or for testing one credential set at a time.

The second mode allows you to test multiple credentials to facilitate credential brute force or automated fuzzing for fault discovery. You can load a large number of usernames and passwords from the file system and click "Go". JMSDigger will then combine each username from the Usernames list with all the passwords from the Passwords list, and try to connect to the broker with all the combinations.

The Results area in this tab shows you the exception details or success information for each attempted credential set. The Results area also shows a summary of working credentials at the end once JMSDigger runs through all the username and password combinations.

The three images below show the results of performing single credential testing and of the credential brute force.

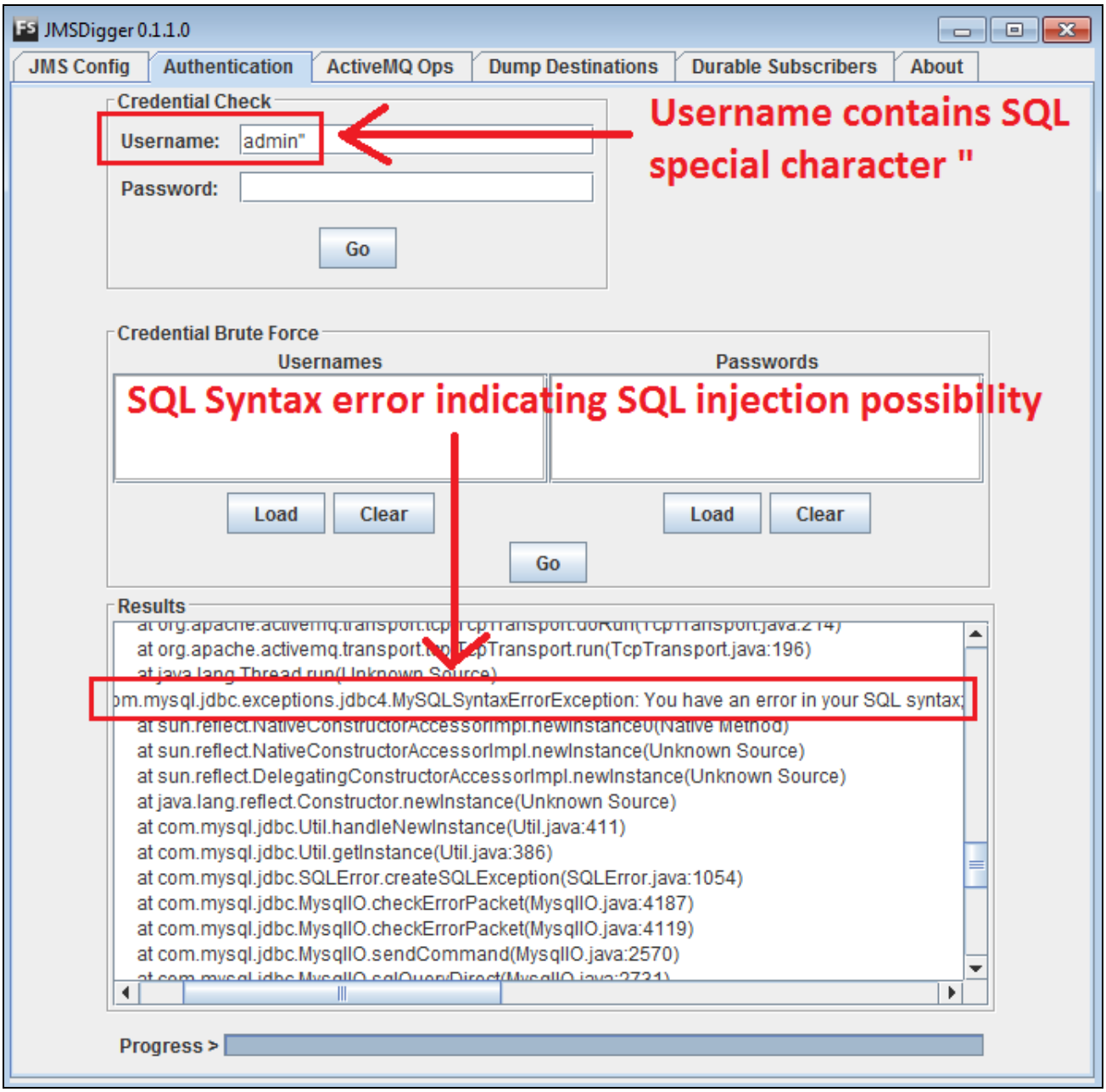

Figure 4: Image shows SQL error returned by the message broker when a SQL special character " is provided in the username

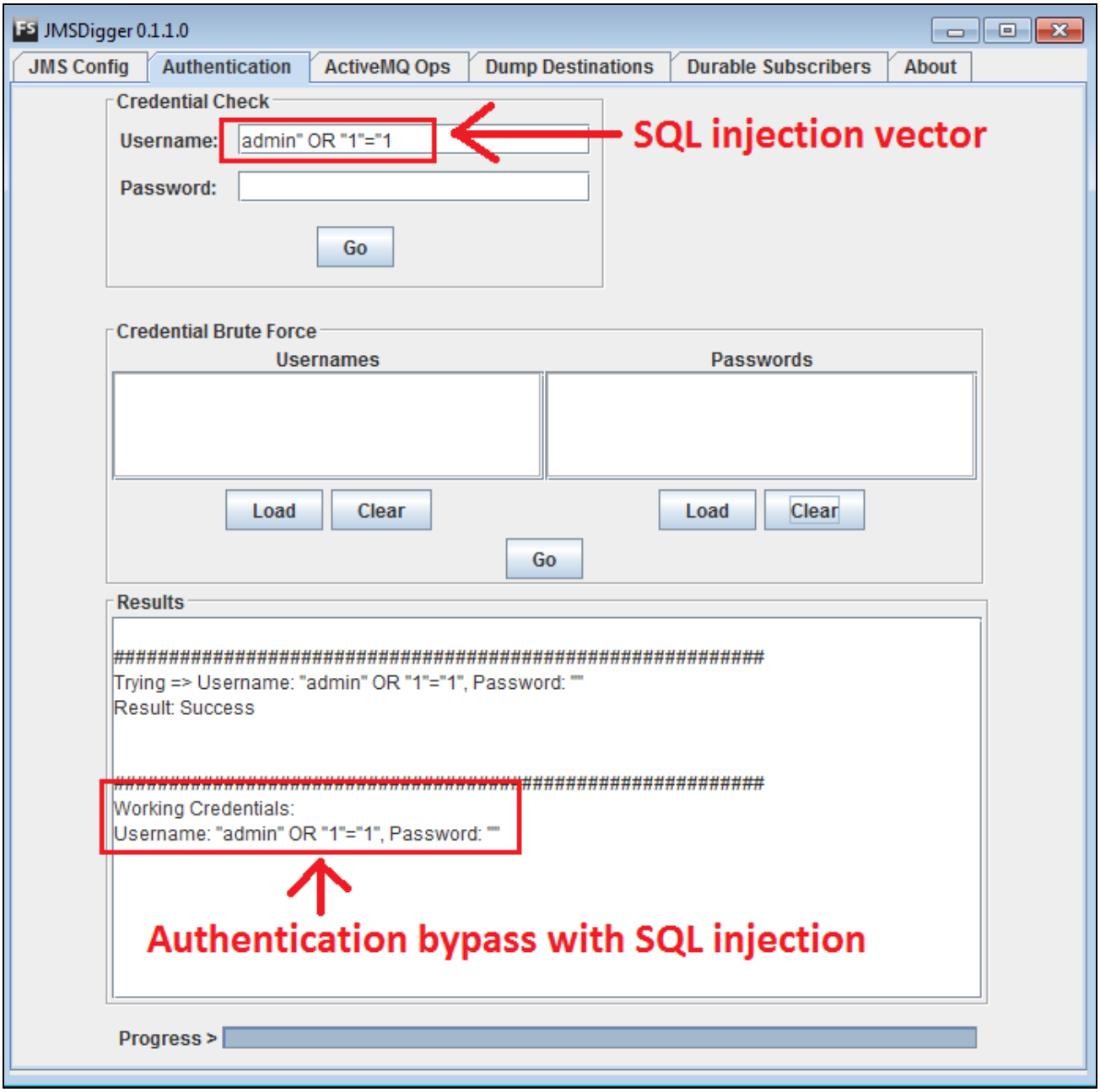

Figure 5: Image shows successful authentication using SQL injection vulnerability

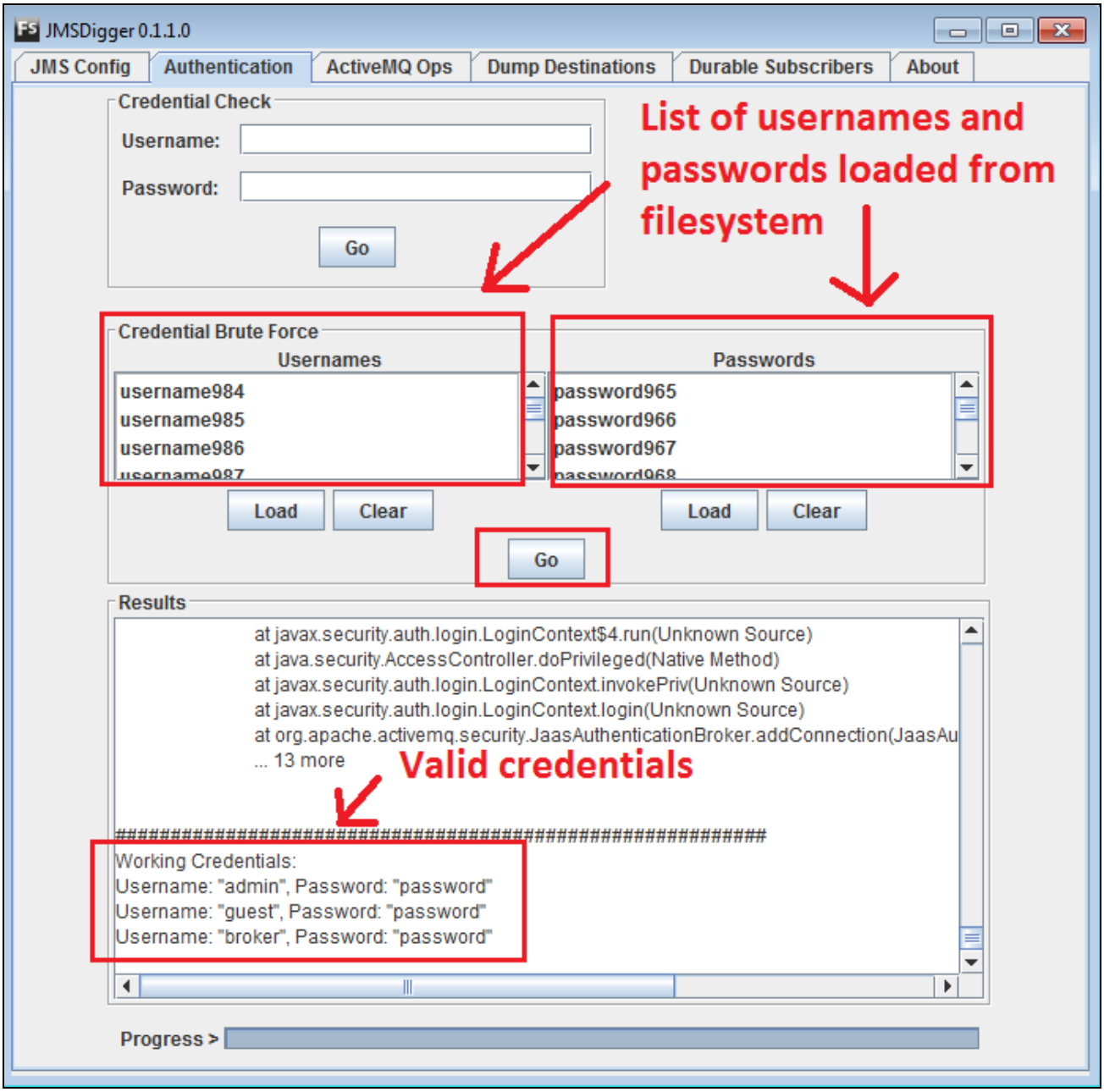

Figure 6: Image shows three working credentials as a result of a credential brute force test

ActiveMQ Specific Operations

The "ActiveMQ Ops" tab contains the functionalities that are specific to ActiveMQ testing. There are three distinct functionalities provided in this tab. Let us discuss each one of them.

Create New Destinations

Since ActiveMQ allows clients to create JMS destinations, JMSDigger provides you with a functionality to take advantage of this ActiveMQ feature to create new destinations on the message broker under the "Create Destinations" control group. Once you have the right configuration (URL and credentials), you can provide a

name for new destination, select appropriate Radio control to indicate that the destination is a Queue or a Topic and click the "Go" button to create new destination on the message broker.

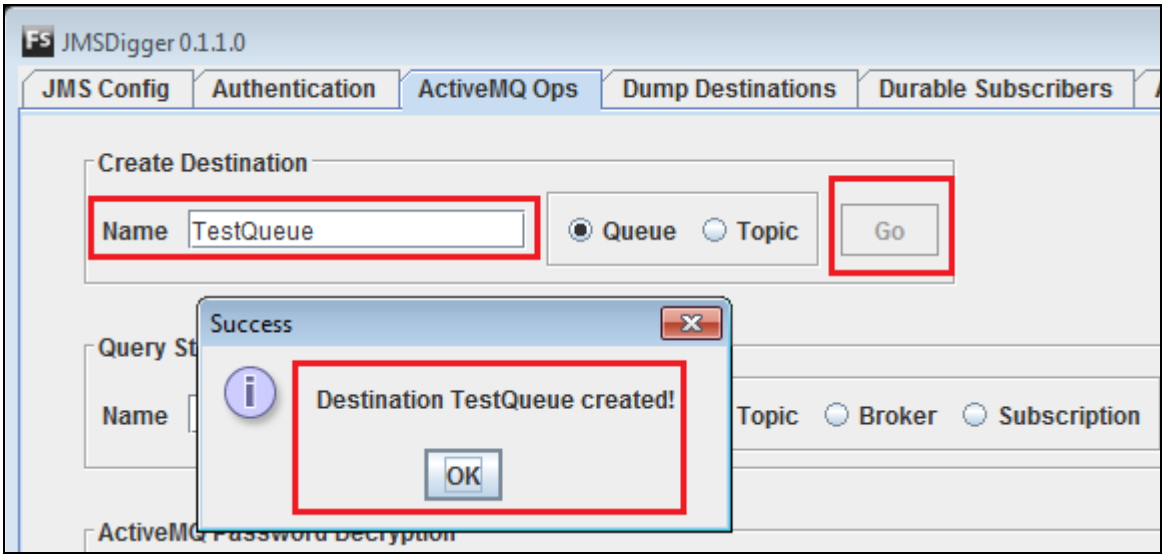

Figure 7: Image shows a success message after creation of a new Queue

Query Broker Statistics

ActiveMQ supports a statistics plugin, which, when enabled, allows the authorized clients to query broker statistics including statistics of the Queues, Topics, Broker and Subscriptions. You can provide name of the component for which you want to see the statistics, select the component type and hit "Go". JMSDigger will retrieve the statistics and display them in the Results area.

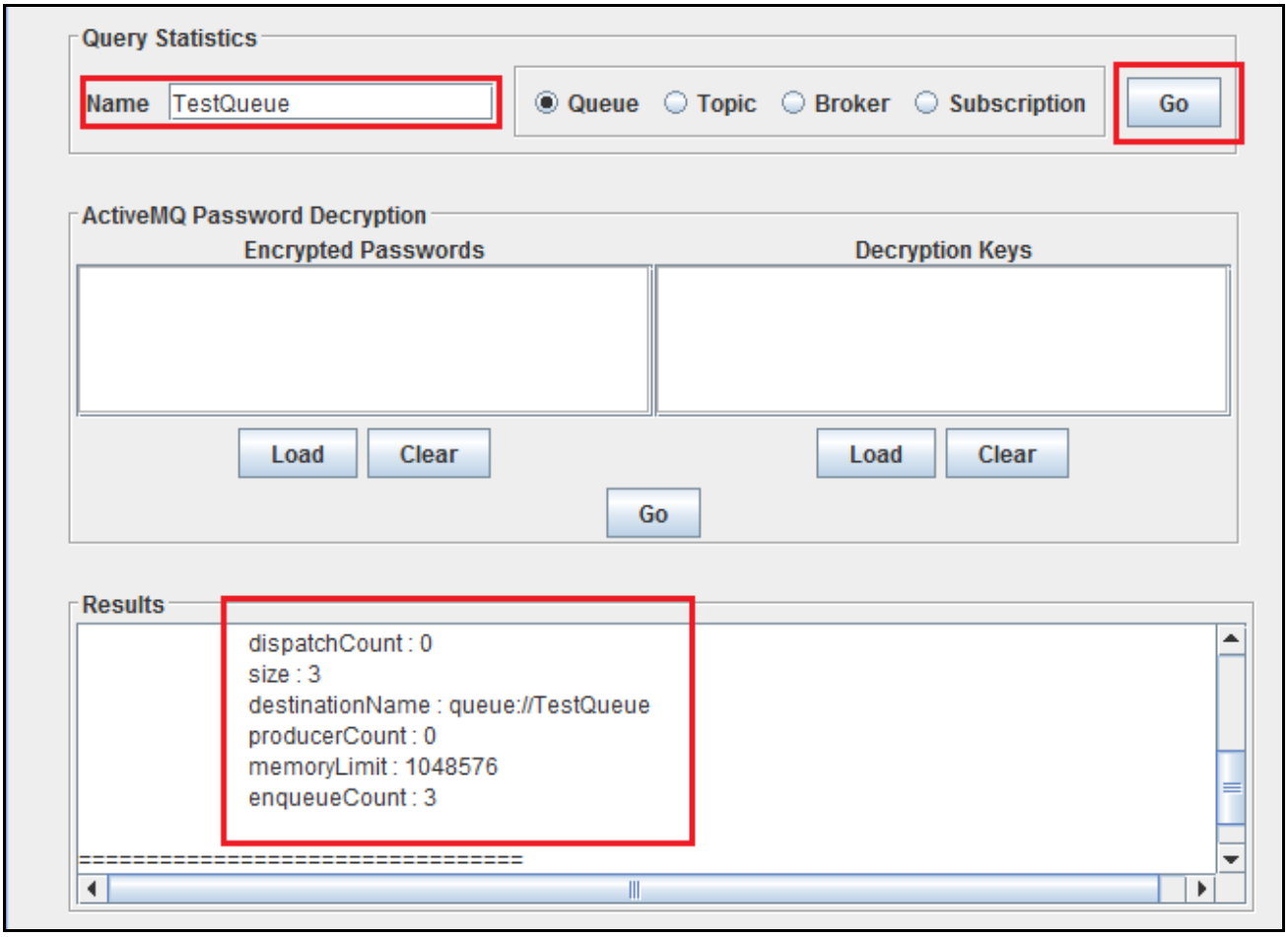

Figure 8: Image shows statistics for the TestQueue

Decrypt ActiveMQ Configuration Passwords

ActiveMQ optionally uses password based encryption (jascrpyt[2] library's `StandardPBEStringEncryptor` class) to encrypt and store login credentials in its configuration files. Using weak passwords to encrypt login credentials can potentially expose these credentials to anyone who has access to these configuration files. System administrators often rely on password cracking tools like John the Ripper[3] to audit password strength. Similarly, JMSDigger can assist with ActiveMQ password audits.

[2] http://www.jasypt.org/
[3] http://www.openwall.com/john/

```
# Defines credentials that will be used by components

activemq.username=system
activemq.password=ENC(mYRkg+4Q4hua1kvpCCI2hg==)
guest.password=ENC(Cf3Jf3tM+UrSOoaKU50od5CuBa8rxjoL)
```

Figure 9: Image shows sample encrypted passwords for ActiveMQ

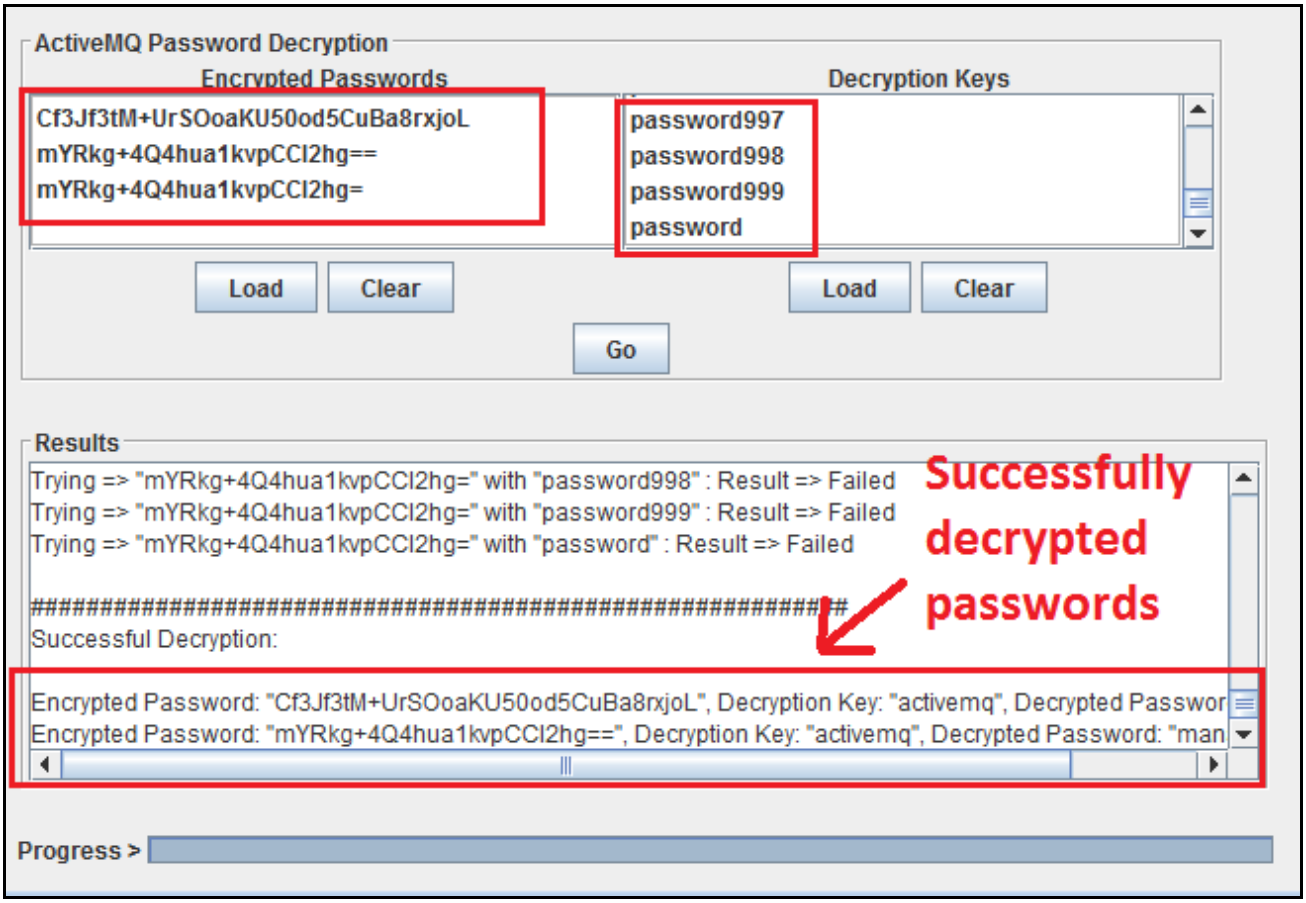

Figure 10: Image shows successful password decryption by JMSDigger

Steps to audit encrypted passwords:

1. Extract the encrypted credentials from inside the `ENC()` block from ActiveMQ configuration files
2. Paste the copied credentials to a new text file with one encrypted blob per line
3. Import the encrypted passwords to "Encrypted Password" section
4. Import newline separated password list to "Decryption Keys" section
5. Hit "Go"

JMSDigger will show a summary of successful decryptions for each successful decryption attempt at the end of the run cycle. The test machine with a single core was able to make up to 1 million decryption attempts in 240 seconds.

Retrieving Messages from JMS Destinations

The "Dump Destinations" tab houses the functionality to retrieve messages from JMS destinations. JMSDigger provides a condensed UI for retrieving messages from three different types of destinations, namely Queues, Topics and Durable Subscribers. The underlying mechanisms for retrieving messages from different types of destinations are different and will be discussed separately.

From Topics

To retrieve messages from a Topic, you need to provide the Topic name in the "Destination Name" text field and set the "Destination Type" radio button to "Topic". Optionally, you can provide a message selector to filter only the messages you are interested in. You can also choose the number of messages to retrieve and the directory where JMSDigger shall write the message dump. Once you are set with the configuration, click "Start Dump" and JMSDigger will connect to the Topic and retrieve messages till the number of messages retrieved reaches the number you specified.

Setting the "Messages to Dump" count to 0 allows JMSDigger to retrieve messages from a Topic indefinitely or until you explicitly click on "Stop Dump" to stop the dumping process. The two images below show JMSDigger retrieving the messages and an example message retrieved from a Topic.

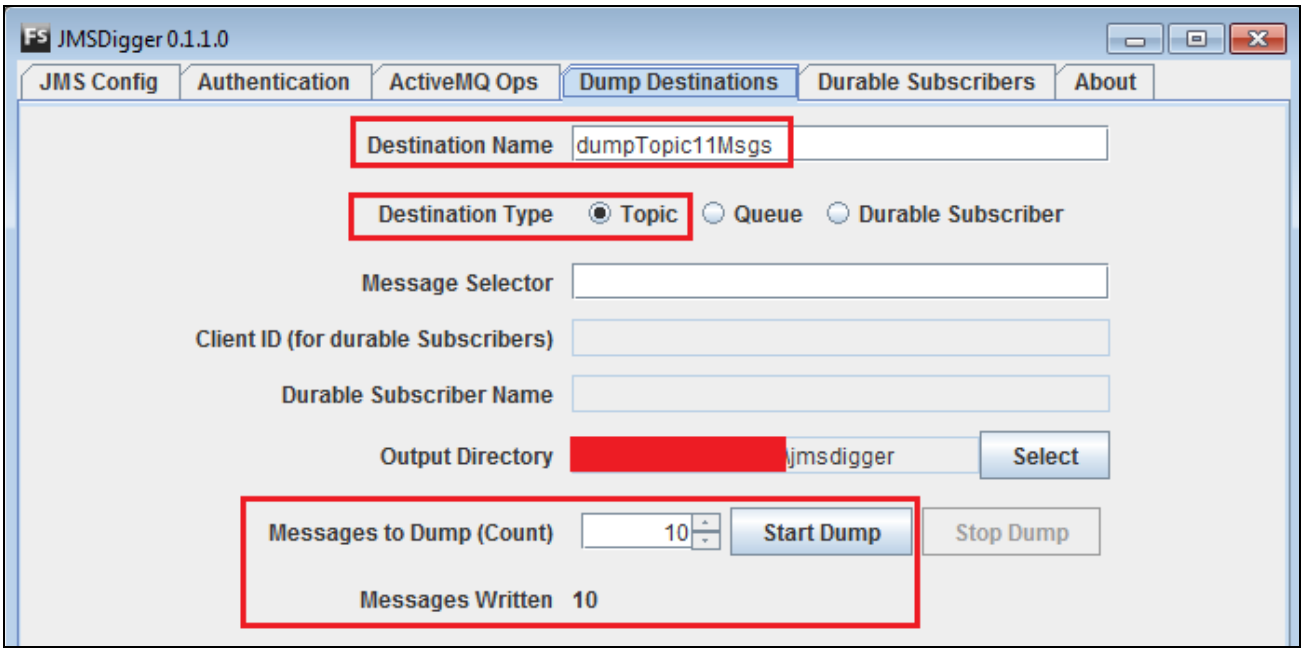

Figure 11: Image shows JMSDigger after successfully retrieving 10 messages from a Topic named dumpTopic11Msgs

Figure 12: Image shows a MapMessage retrieved from the Topic seen in image above

From Queues

The process of retrieving messages from a Queue is almost similar to a Topic with one minor difference. In my previous whitepaper we discussed how the `QueueBrowser` API is used to retrieve messages without disrupting a Queue's contents. When a `QueueBrowser` is used, the querying JMS client receives an enumeration of all messages on the Queue at a point in time, which the JMS client can iterate through. JMSDigger uses `QueueBrowser` for its obvious benefits to avoid detection and to not disrupt normal message flow.

However, using `QueueBrowser` has one disadvantage when it comes to message retrieval. If the messages from the Queue are consumed at a rate slower than that of the `QueueBrowser` instance, the `QueueBrowser` will receive some messages from the previous retrievals, leading to duplicates. The worst

case scenario is when no messages are consumed from the Queue. In this case, the retrieved messages will be duplicated in every `QueueBrowser` enumeration and hence in the message dump. So, if a Queue contains one message and that message is not consumed, JMSDigger will repeatedly retrieve and write that message to the message log.

JMSDigger's current release does not have any duplicate message detection mechanisms in place to address this issue.

From Durable Subscribers

The Durable Subscriber functionality requires you to select the "Durable Subscriber" radio button to enable to two fields to enter the Client ID and the Durable Subscriber Name in addition to the Topic Name in the "Destination Name" input field. Once you have keyed in the details, you can click on the "Start Dump" button to retrieve messages from the Durable Subscriber. Please note that once you initiate message retrieval from a Durable Subscriber, all the messages you retrieve will be discarded from the Durable Subscriber store on the message broker. Please use this feature with caution.

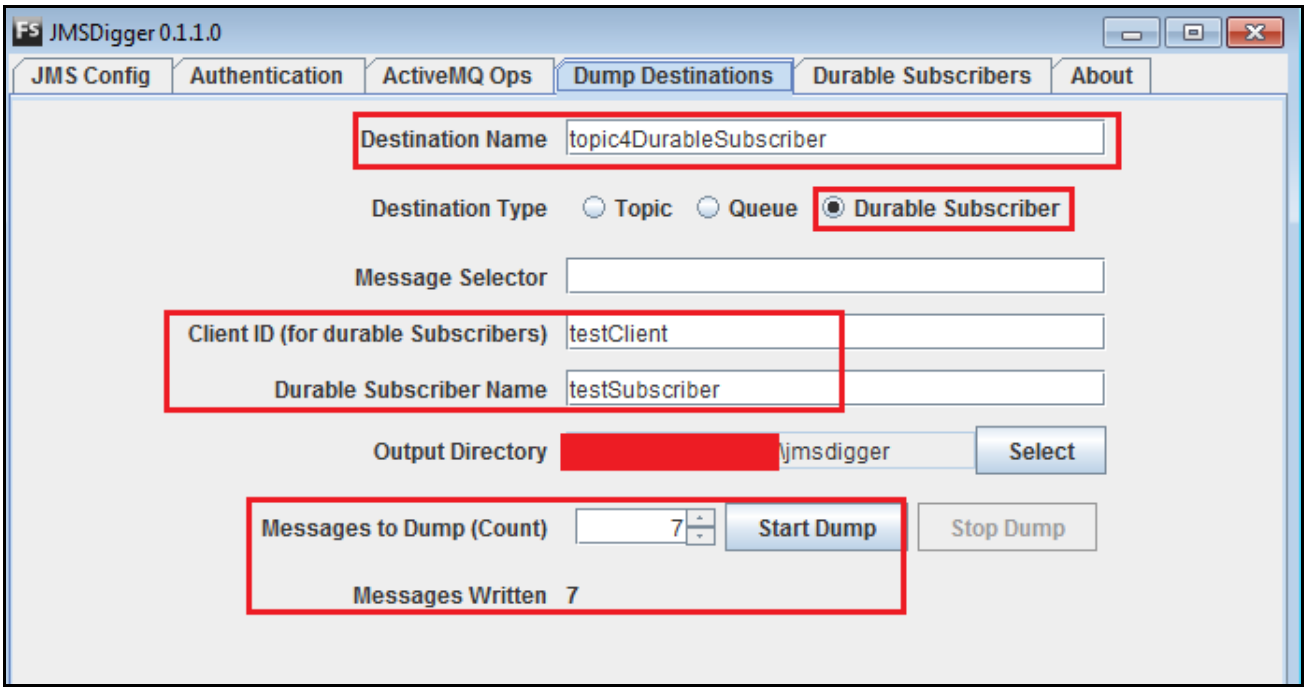

Figure 13: Image shows JMSDigger retrieving messages from a Durable subscriber

Playing with Durable Subscribers

The "Durable Subscribers" tab contains the functionality to create and erase durable subscribers. It allows you to create one or more durable subscribers and also to erase them.

Creating a Durable Subscriber

To create a durable subscriber for a Topic (on the message broker), supply JMSDigger with the Topic name, Client ID and Durable Subscriber name in the respective input areas and click "Create". The test result is updated in the Results area.

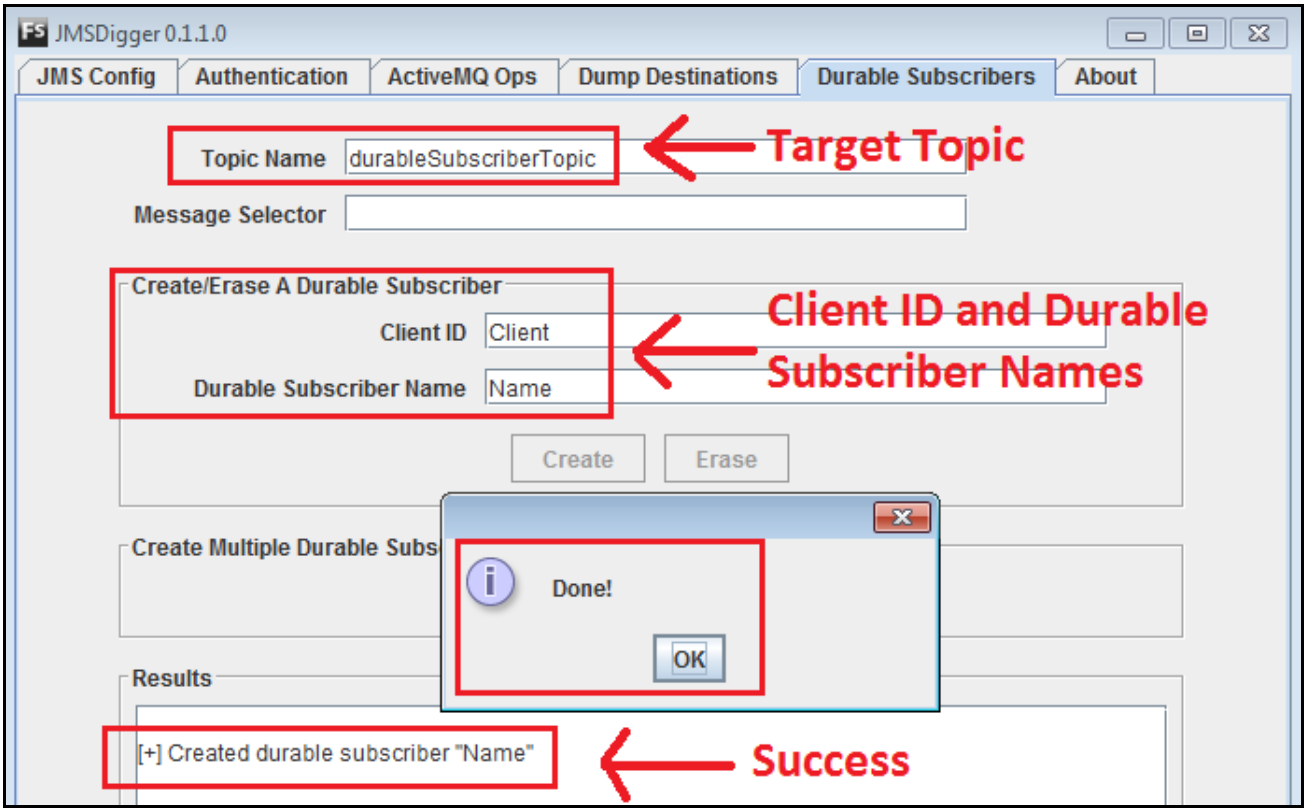

Figure 14: image shows successful durable subscriber creation with JMSDigger

Figure 15: image shows a durable subscriber created on ActiveMQ by JMSDigger

Erasing a Durable Subscriber

To erase a durable subscriber, you need to enter the Topic Name, Client ID and Durable Subscriber Name in respective input fields and click "Erase". If the durable subscriber exists, JMSDigger will erase the durable subscriber or show you the error message in the Results text area.

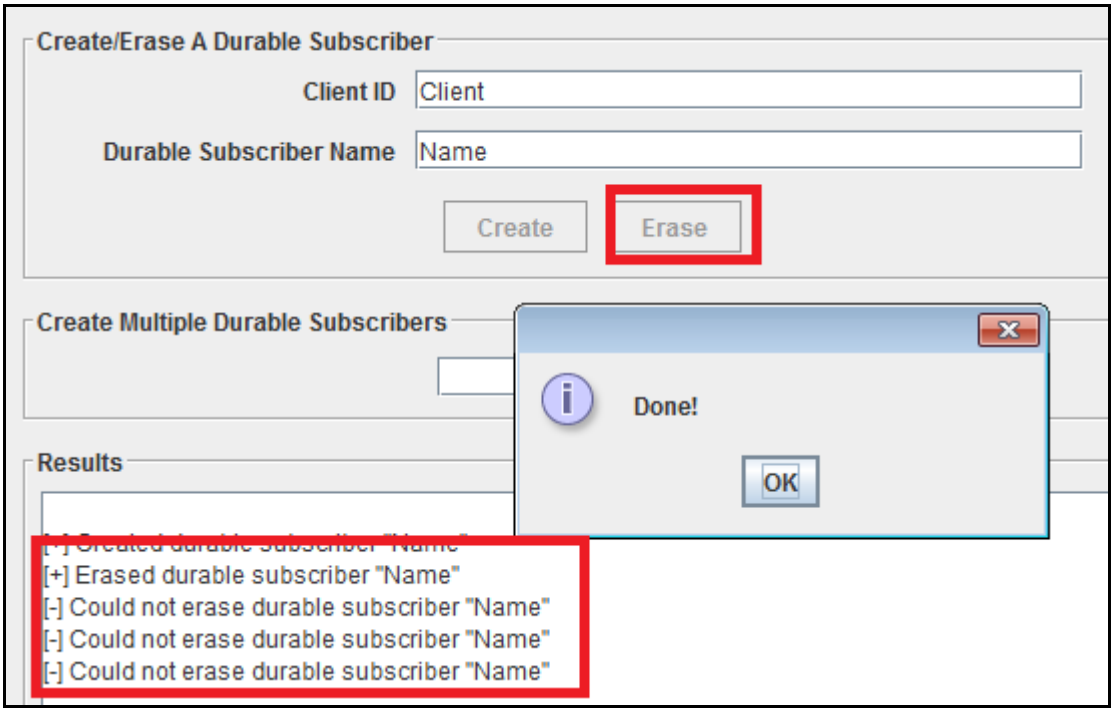

Figure 16: image shows successful durable subscriber erase operation followed by failures when the durable subscriber was no longer present

Creating Multiple (Random) Durable Subscribers

You can cause a resource starvation/Denial of Service condition on the message broker with high transactional loads by creating large number of durable subscribers on the message broker and disconnecting from the broker. The message broker will then start accumulating messages until the durable subscriber is erased, messages expire or messages are read from it.

With JMSDigger you can provide the configuration information, a Topic Name, select the number of random durable subscribers you want to create and click on "Create" button inside the "Create Multiple Durable Subscribers" control group to create a large number of durable subscribers with random names.

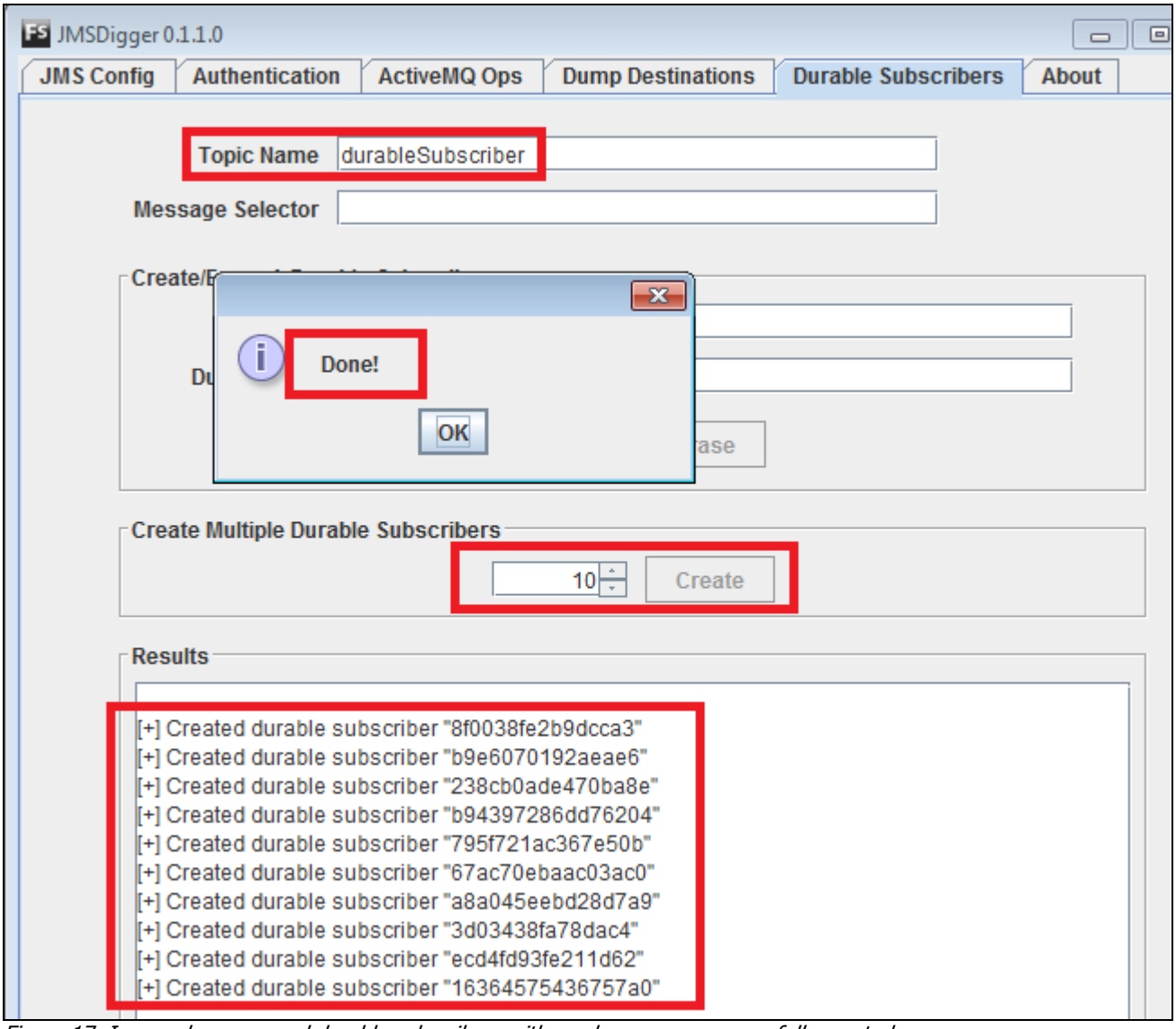

Figure 17: Image shows several durable subscribers with random names successfully created

About The Author

Gursev Singh Kalra serves as a Senior Principal with Foundstone Professional Services, a division of McAfee. Gursev has authored several security related whitepapers and his research has been voted among the top ten web hacks of 2011 and 2012. He loves to code and has authored several free security tools like JMSDigger, TesserCap, Oyedata, SSLSmart and clipcaptcha. He has spoken at conferences like BlackHat, OWASP AppSec ToorCon, NullCon and Infosec Southwest etc...

About Foundstone Professional Services

Foundstone® Professional Services, a division of McAfee. Inc., offers expert services and education to help organizations continuously and measurably protect their most important assets from the most critical threats. Through a strategic approach to security, Foundstone identifies and implements the right balance of technology, people, and process to manage digital risk and leverage security investments more effectively. The company's professional services team consists of recognized security experts and authors with broad security experience with multinational corporations, the public sector, and the US /military.

About McAfee

McAfee, a wholly owned subsidiary of Intel Corporation (NASDAQ:INTC), is the world's largest dedicated security technology company. McAfee delivers proactive and proven solutions and services that help secure systems, networks, and mobile devices around the world, allowing users to safely connect to the Internet, browse, and shop the web more securely. Backed by its unrivaled global threat intelligence, McAfee creates innovative products that empower home users, businesses, the public sector, and service providers by enabling them to prove compliance with regulations, protect data, prevent disruptions, identify vulnerabilities, and continuously monitor and improve their security. McAfee is relentlessly focused on constantly finding new ways to keep our customers safe. http://www.mcafee.com