Apache Qpid Denial of Service

06/02/2015

Software:	Apache Qpid
Affected Versions:	All versions prior and including 0.30 are affected.
CVE Reference:	CVE-2015-0224
Author:	Georgi Geshev - MWR Labs (http://labs.mwrinfosecurity.com/)
Severity:	High
Vendor:	Apache
Vendor Response:	Fix Released

Description:

Apache Qpid is an open source message-oriented middleware messaging broker. Qpid provides Java and C++ implementations of the Advanced Message Queuing Protocol (AMQP).

The C++ broker implementation is susceptible to several denial of service conditions caused by mishandling of invalid or unsupported AMQP packets.

Impact:

Three distinct bugs allow for an unauthenticated adversary to crash the broker process, thus causing a denial of service and preventing legitimate users from exchanging messages.

Cause:

Apache Qpid performs incomplete and insufficient sanity checks on incoming AMQP packets.

Interim Workaround:

Apache has developed a patch that can be manually applied to Qpid 0.30. This patch is available under the following link: https://issues.apache.org/jira/browse/QPID-6310

Solution:

Upgrade to Apache Qpid version 0.31 or later.

Technical details

Three distinct denial of service conditions were identified in the way Qpid handles malformed, unsupported or unexpected AMQP packets.

The AMQP 0-10 specification allows for certain commands and controls to transfer additional segments, namely header and body segments that can be used for carrying message content.

Apache Qpid handles additional segments only for the AMQP message-transfer command. Any other command that includes header and/or body segments would cause a segmentation fault in the broker process due to an out-of-bounds read condition. This will cause the broker process to exit.

The following patch was developed to discard frames with additional segments, unless the specified AMQP command is message-transfer.

```
diff --git a/qpid/cpp/src/qpid/broker/MessageBuilder.cpp
b/qpid/cpp/src/qpid/broker/MessageBuilder.cpp
index 7cb9951..f5e9332 100644
--- a/qpid/cpp/src/qpid/broker/MessageBuilder.cpp
+++ b/qpid/cpp/src/qpid/broker/MessageBuilder.cpp
@@ -45,6 +45,9 @@ void MessageBuilder::handle(AMQFrame& frame)
     switch(state) {
     case METHOD:
         checkType(METHOD_BODY, type);
+        if (!frame.getMethod()->isA<qpid::framing::MessageTransferBody>())
+            throw NotImplementedException(QPID_MSG("Unexpected method: " <<
*(frame.getMethod())));
+
         exchange = frame.castBody<qpid::framing::MessageTransferBody>()-
>getDestination();
         state = HEADER;
         break;
```

Another vulnerability relates to the way Apache Qpid handles out-of-session AMQP controls.

The AMQP 0-10 specification defines two distinct data units, namely commands and controls, that can be transferred in AMQP. Commands can only be sent on established sessions, while there is no such requirement for the controls.

One of the AMQP defined controls is session-gap. This control is not supported by Qpid and an appropriate error message is returned to a client when such a control is received on an established session. However, the session-gap control is not properly handled when requested before a session is opened, which triggers an assert statement and causes the process to exit.

The vendor fix was to check whether this control is received out-of-session, discard the AMQP frame and notify the client.

```
diff --git a/qpid/cpp/src/qpid/amqp_0_10/SessionHandler.cpp
b/qpid/cpp/src/qpid/amqp_0_10/SessionHandler.cpp
```

```
index 43f39c2..bd0dcbf 100644
--- a/qpid/cpp/src/qpid/amqp_0_10/SessionHandler.cpp
+++ b/qpid/cpp/src/qpid/amqp_0_10/SessionHandler.cpp
@@ -276,6 +276,7 @@ void SessionHandler::flush(bool expected, bool confirmed, bool
completed) {
 }

 void SessionHandler::gap(const SequenceSet& /*commands*/) {
+    checkAttached();
     throw NotImplementedException("session.gap not supported");
 }
```

Another flaw was found in the way Apache Qpid handles AMQP sequence-set variables.

The AMQP 0-10 specification defines the variable width sequence-set type. According to the AMQP standard, the sequence-set type is a set of pairs of RFC-1982 numbers representing a discontinuous range within an RFC-1982 sequence. Each pair represents a closed interval within the list. The messaging broker service can be crashed when handling a sequence-set which contains a pair, also referred to as a range, where the range expressed is the maximum possible one. This would trigger an assert statement which will cause the process to exit.

The following patch resolves the vulnerability and supersedes an incomplete fix related to CVE-2015-0203.

```
diff --git a/qpid/cpp/src/qpid/framing/SequenceSet.cpp
b/qpid/cpp/src/qpid/framing/SequenceSet.cpp
index 845bf8b..6510842 100644
--- a/qpid/cpp/src/qpid/framing/SequenceSet.cpp
+++ b/qpid/cpp/src/qpid/framing/SequenceSet.cpp
@@ -33,7 +33,18 @@ namespace framing {

 namespace {
 //each range contains 2 numbers, 4 bytes each
-uint16_t RANGE_SIZE = 2 * 4;
+uint16_t RANGE_SIZE = 2 * 4;
+int32_t MAX_RANGE = 2147483647;//2^31-1
+
+int32_t gap(const SequenceNumber& a, const SequenceNumber& b)
+{
+    return a < b ? b - a : a - b;
+}
+
+bool is_max_range(const SequenceNumber& a, const SequenceNumber& b)
+{
+    return gap(a, b) == MAX_RANGE;
+}
```

```
 }

 void SequenceSet::encode(Buffer& buffer) const
@@ -58,7 +69,17 @@ void SequenceSet::decode(Buffer& buffer)
         SequenceNumber b(buffer.getLong());
         if (b < a)
             throw IllegalArgumentException(QPID_MSG("Invalid range in sequence set:
" << a << " -> " << b));
-        add(a, b);
+        if (is_max_range(a, b)) {
+            //RangeSet holds 'half-closed' ranges, where the end is
+            //one past the 'highest' value in the range. So if the
+            //range is already the maximum expressable with a 32bit
+            //sequence number, we can't represent it as a
+            //'half-closed' range, so we represent it as two ranges.
+            add(a, b-1);
+            add(b);
+        } else {
+            add(a, b);
+        }
     }
 }
```

Detailed Timeline

Date:	Summary:
22/12/2014	Reported to Apache and Red Hat
23/12/2014	Apache confirms reception
07/01/2015	Red Hat suggests fix
12/01/2015	MWR acknowledges fix
13/01/2015	Public fix released
18/01/2015	MWR notifies the vendor of fix and advisory inconsistencies
19/01/2015	Red Hat confirms inconsistencies
23/01/2015	Red Hat suggests new fix
26/01/2015	Public fix released
06/02/2015	Advisory published