

## Zielstellung und Rahmenbedingungen

Diese Dokumentation beschäftigt sich mit dem Thema „Angriffe auf AMQP-Messagebroker“. Begleitend zur Vorlesung „Betriebliche Informationssysteme“ galt es von uns in einem Praktikum zu identifizieren, welche potenziellen Angriffsvektoren existieren. Eingeschlossen ist hier auch das System „RabbitMQ“ welches das zu untersuchende Protokoll einschließt.

Ziel ist es bestimmt Angriffsvektoren aufzuspüren und zu bewerten. Durch Implementierung von einzelnen Clients auf Basis der RabbitMQ Java-Bibliothek sollen die Angriffe veranschaulicht werden. Mit veränderten Parametern richtet sich die Suche gezielt nach Faktoren die das System negativ beeinflussen und so die Verfügbarkeit stören, fokussiert auf „Denial of Service“. DoS kann dabei auf verschiedene Ressourcen bezogen sein (wie CPU, Arbeitsspeicher, Netzwerkbandbreite, ...).

Um die Begrenzungen ausfindig zu machen, gehört es zu Beginn zu unsere Aufgabe angemessene Werkzeuge zur Beobachtung der Ressourcen zu finden. Nur dann ist es möglich die Angriffe zu bewerten und ihre tiefere Auswirkung zu untersuchen.

Ferner sollen Vorschläge zur Schadensbegrenzung gegeben werden. Dazu gehört die Angabe auf welcher Ebene (Netzwerkebene, Protokollebene, ...) sich die Gefahren beseitigen lassen.

## Testumgebung

## Verwendete bzw. Erstellte Programme

RabbitMQ bietet einige Anwendungen, mit denen sich die Leistungsfähigkeit der Server messen lässt. Alle Hilfsprogramme sind innerhalb der *rabbitmq-client-tests.jar* vorzufinden. Diese JAR-Datei enthält weiterhin zahlreiche kleine Beispielprogramme für das Testen der Funktionalität des eigenen Servers.

**Perftest** ist ein Performance-Test-Tool, welches beliebig viele Producer und Consumer erstellt und die Sende-/Empfangsrate misst. Zusammen mit der Latenzzeit werden alle Angaben auf der Konsole ausgegeben.

**HTML Performance Tools** bieten ebenfalls ein breites Spektrum an Funktionalität. Mit einer Reihe von Tools und Unterstützung von perftest lassen sich automatisierte Benchmarks erstellen. Die gelieferten Daten dienen dem Vergleich der Systems vor und während des Angriffs. Alle Ergebnisse werden in einer JSON-Datei gesichert und ansprechend in einer HTML-Seite dargestellt

## Beschreibung der Anwendungsszenarien

### **S1**      Direkte Nachrichten 1:1

---

*Beschreibung*      Ein Producer erzeugt kontinuierlich Nachrichten, die von einem Consumer kontinuierlich korrekt entnommen werden.

*Aktion*              `perfTest -h`

*Anmerkungen*

### **S2**

---

*Beschreibung*

*Aktion*

*Anmerkungen*

### **S3**

---

*Beschreibung*

*Aktion*

*Anmerkungen*

## Beschreibung der Angriffe

---

### A1 Ignorieren von Nachrichten

---

<i>Beschreibung</i>	Ein Producer erzeugt kontinuierlich Nachrichten, die von einem oder mehreren Consumer empfangen, aber nicht quittiert werden. Der RabbitMQ-Server ist somit gezwungen, die Nachrichten in der Queue zwischenzuspeichern.
<i>Testparameter</i>	
<i>Befehlszeile</i>	<code>Amqpstress -dm No -c 5 -i 1 -ms 1048576 -mp -u amqp://testc:testp@localhost:5672/%2f</code>
<i>Beobachtungen</i>	
<i>Anmerkungen</i>	Durch Verwendung einer höheren Anzahl an Consumer, kleineren Sendeintervallen, größeren Nachrichtengrößen sowie durch Verwendung von persistenten Queues kann die Auswirkung des Angriffs erhöht werden.

---

### A2 Sofortiges Abweisen von Nachrichten

---

<i>Beschreibung</i>	Ein Producer erzeugt kontinuierlich Nachrichten, die von einem oder mehreren Consumer empfangen, aber sofort abgewiesen (basic.Reject) werden. Der RabbitMQ-Server ist somit gezwungen, die Nachrichten in der Queue zwischenzuspeichern und erneut an den Consumer zu senden.
<i>Testparameter</i>	
<i>Befehlszeile</i>	<code>Amqpstress -dm REJECT -c 5 -i 1 -ms 1048576 -mp -u amqp://testc:testp@localhost:5672/%2f</code>
<i>Beobachtungen</i>	
<i>Anmerkungen</i>	Durch Verwendung einer höheren Anzahl an Consumer, kleineren Sendeintervallen, größeren Nachrichtengrößen sowie durch Verwendung von persistenten Queues kann die Auswirkung des Angriffs erhöht werden.

A3	Gebündeltes Abweisen von Nachrichten
<i>Beschreibung</i>	Ein Producer erzeugt kontinuierlich Nachrichten, die von einem oder mehreren Consumer empfangen, zunächst ignoriert werden, um sie bei Erreichen eines Schwellwertes gebündelt abzuweisen (basic.NACK). Dadurch ist der RabbitMQ-Server gezwungen alle Nachrichten zwischenspeichern und stoßweise alle Nachrichten bis zu einer gewissen Nachrichtenkennung erneut zuzustellen.
<i>Testparameter</i>	<pre>Amqpstress -dm NACK -c 5 -i 1 -ms 1048576 -mp -u amqp://testc:testp@localhost:5672/%2f</pre>
<i>Beobachtungen</i>	
<i>Anmerkungen</i>	
A4	Versenden von Nachrichten mit großem Header
<i>Beschreibung</i>	RabbitMQ bietet die Möglichkeit im Header der Nachricht bestimmte Parameter für die Weiterleitung zu deklarieren. Dieser Test beschäftigt sich mit der Auswirkung, wenn der Header unnötig ausgelastet wird. Das System ist gezwungen alle Weiterleitungsoptionen zu prüfen, auch wenn diese keinem Ziel entsprechen.
<i>Testparameter</i>	<pre>-LH (AUFRUFPARAMETER FÜR HEADERERZEUGUNG) -MS 10000 (MESSAGEGRÖSSE IN BYTE - HIER: 10000) -U (URI FÜR VERBINDUNG MIT SERVER)</pre>
<i>Befehlszeile</i>	<pre>Amqpstress -lh -ms 10000 -u amqp://testc:testp@localhost:5672</pre>
<i>Beobachtungen</i>	Die Anwendung generiert zu Beginn 1000 Weiterleitungsoptionen und schreibt sie in den Header jeder Nachricht. Hierdurch ist das System stark ausgelastet und der Durchsatz der Nachrichten schrumpft stark auf 3 - 5 Nachrichten pro Sekunde.
<i>Anmerkungen</i>	Die Headergröße ist bei 2500 Weiterleitungsoptionen begrenzt. Wird diese überschritten, lässt das System die Verbindung fallen, aufgrund einer zu großen Framegröße.

---

**A5****Aufbauen mehrerer Channel über eine einzelne Verbindung**

---

<i>Beschreibung</i>	Neben einzelnen Verbindungen können in RabbitMQ auch mehrerer Kanäle aufgebaut werden. Hier stellt sich die Frage, wie das System mit einer Vielzahl von Kanäle zurecht kommt. Auf Basis einer einzelnen Verbindung wird das System so ausgelastet und beobachtet.
<i>Testparameter</i>	<code>-mc</code> (AUFRUFPARAMETER FÜR GENERIERUNG MEHRERER CHANNEL) <code>-ms 10000</code> (MESSAGEGRÖSSE IN BYTE - HIER: 10000) <code>-u</code> (URI FÜR VERBINDUNG MIT SERVER) <code>-p 100</code> (ANZAHL PRODUCER - HIER:100) <code>-c 10</code> (ANZAHL CONSUMER - HIER: 10)
<i>Befehlszeile</i>	<code>Amqpstress -mc -p 100 -c 10 -ms 10000 -u amqp://testc:testp@localhost:5672</code>
<i>Beobachtungen</i>	Das System ist stark ausgelastet. Ähneln aber der Auslastung unter der Erstellung mehrerer Verbindungen. Allerdings beansprucht der Aufbau der Channel extrem viel Zeit. Nach Aufbau aller Kanäle bricht die Übertragungsrate stark ein.
<i>Anmerkungen</i>	Zeit für Aufbau der Channel hängt stark von der Anzahl von Producer und Consumer ab.

## Auswirkungen der Angriffe



## Zusammenfassung und Fazit