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 ausgeben.

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

VirtualBox Der von uns verwendete Server wurde durch die Virtualisierungslösung VirtualBox realisiert. Neben der einfachen Installation neuer Gast-Systeme besteht die Möglichkeit Sicherheitspunkte zu erstellen. Bei einem Ausfall des System kann so der ursprüngliche Sicherungspunkt wiederhergestellt werden. Als Grundlage dient die Servervariante von Ubuntu 14.04 LTS, welche durch die fehlende grafische Oberfläche zum einsparen von Ressourcen dient.

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
Beschreibung	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.
Testparameter	
Befehlszeile	`Amqpstress -dm No -c 5 -i 1 -ms 1048576 -mp -u amqp://testc:testp@localhost:5672/%2f`
Beobachtungen	
Anmerkungen	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
Beschreibung	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.
Testparameter	
Befehlszeile	`Amqpstress -dm REJECT -c 5 -i 1 -ms 1048576 -mp -u amqp://testc:testp@localhost:5672/%2f`
Beobachtungen	
Anmerkungen	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

Beschreibung	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 gewzungen alle Nachrichten zwischenzuspeichern und stoßweise alle Nachrichten bis zu einer gewissen Nachrichtenkennung erneut zuzustellen.
Testparameter	
Befehlszeile	`Amqpstress -dm NACK -c 5 -i 1 -ms 1048576 -mp -u amqp://testc:testp@localhost:5672/%2f`
Beobachtungen	
Anmerkungen	

A4 Versenden von Nachrichten mit großem Header

Beschreibung	RabbitMQ bietet die Möglichkeit im Header der Nachricht bestimmt 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.
Testparameter	-lh (Aufrufparameter für Headererzeugung) -ms 10000 (Messagegrösse in Byte - hier: 10000) -u (URI für Verbindung mit Server)
Befehlszeile	`Amqpstress -lh -ms 10000 -u amqp://testc:testp@localhost:5672`
Beobachtungen	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.
Anmerkungen	Die Headergröße ist bei 2500 Weiterleitungsoptionen begrenzt. Wir diese überschritten lässt das System die Verbindung fallen, aufgrund einer zu großen Framegröße.

A5 Aufbauen mehrerer Channel über eine einzelne Verbindung

Beschreibung	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.
Testparameter	-mc (Aufrufparameter für Generierung mehrerer Channel) -ms 10000 (Messagegrösse in Byte - hier: 10000) -u (URI für Verbindung mit Server) -p 100 (Anzahl Producer - hier:100) -c 10 (Anzahl Consumer - hier: 10)
Befehlszeile	`Amqpstress -mc -p 100 -c 10 -ms 10000 -u amqp://testc:testp@localhost:5672`
Beobachtungen	Das System ist stark ausgelastet. Ähnelt 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.
Anmerkungen	Zeit für Aufbau der Channel hängt stark von der Anzahl von Producer und Consumer ab.

A6 Commit mehrerer Nachrichten im Transaktionsmodus

Beschreibung	Im Transaktionsmodus ist es möglich mehrere Nachrichten als Folge zu übertragen, die aber vergleichbar mit Datenbanken als Einheit betrachtet werden. Nur durch die Commit()-Funktion wird der neue Zustand angenommen und die Nachrichten im System zur Verfügung gestellt. Ferner kann mit der Rollback()-Funktion der ursprüngliche Zustand wiederhergestellt werden.
Testparameter	-tx (Aufrufparameter für Transaktionsmodus) -ms 10000 (Messagegrösse in Byte - hier: 10000) -mct 10000 (Nachrichtenanzahl pro Producer - hier: 10000) -co false (Commit der Transaktion - hier: Auslassen) -p 100 (Anzahl Producer - hier: 100)
Befehlszeile	`-tx -p 100 -ms 10000 -mct 10000 -co false -u amqp://testc:testp@localhost:5672`
Beobachtungen	Durch Auslassen des Commits steigt der Speicherverbrauch stark an. Nach wenigen Sekunden ist die Speichergrenze erreicht und keine weiteren Nachrichten werden übertragen. Daraufhin verharren alle Producer bis der Speicher wieder freigegeben wird (Falls Consumer vorhanden)
Anmerkungen	Sporadisch friert der RabbitMQ-Server bei diesem Versuch ein. Ein Aufbau einer neuen Verbindung schlägt fehl und die Weboberfläche ist nicht mehr erreichbar.

Auswirkungen der Angriffe

Zusammenfassung und Fazit