Automatisierte Korrelation und Aggregation von Syslog-Nachrichten in NoSQL-basierten Datenbanken

Sven Reißmann, Dustin Frisch, Sebastian Rieger

Fachbereich Angewandte Informatik
Hochschule Fulda
Marquardtstraße 35
36039 Fulda
[vorname.nachname]@informatik.hs-fulda.de

Abstract: Logging-Informationen von Systemen und Diensten nehmen in ihrem Umfang durch den Detaillierungsgrad komplexer Anwendungen, aber auch durch die zunehmende Anzahl von verwendeten Systemen (etwa aufgrund der vermehrten schnellen Bereitstellung von virtuellen Maschinen z.B. in Cloud Umgebungen) immer weiter zu. Um die zeitnahe Auswertung und Reaktion auf relevante Logging-Informationen zu gewährleisten sind automatisierte Korrelations- und Aggregationsverfahren erforderlich. In der Vergangenheit wurden diese z.B. durch die Zentralisierung von Logging-Systemen unterstützt. In der vorliegenden Arbeit wird darauf aufbauend ein Prototyp für eine automatisierte inhaltliche Korrelation und Aggregation bzw. Verdichtung der Logging-Informationen präsentiert. Hierfür werden insbesondere die Anforderungen an Logging-Systeme bei der dynamischen Bereitstellung von Systemen und Anwendungen bzw. Diensten (vgl. Cloud Umgebungen) vorgestellt.

1 Einleitung

Nicht zuletzt durch die schnelle Verfügbarkeit von virtuellen Maschinen hat in den vergangenen Jahren der Umfang von Logging-Daten stetig zugenommen. Neben virtuellen Maschinen können z.B. in Cloud-Infrastrukturen schnell neue Dienste und Anwendungen bereitgestellt werden, die Logging-Daten erzeugen anhand derer z.B. der Zustand der Anwendung überwacht werden soll. Dies führt dazu, dass die Quellen aus denen Logging-Daten stammen zunehmen und neue Quellen oft schnell z.B. in ein bestehendes Monitoring-System aufgenommen werden müssen. Wenn die einzelnen Quellen zusätzlich umfangreiche Logging-Informationen zur Verfügung stellen, nimmt die Menge von Logging-Daten weiter zu. In der Regel werden Log-Files, insbesondere bei enthaltenen personenbezogenen Daten, in kurzen Zeitabständen anonymisiert und komprimiert oder alte Log-Einträge verworfen. Die Anzahl der Quellen, die Logging-Daten zur Verfügung stellen lässt sich allerdings häufig nicht reduzieren. Beispielsweise soll in virtualisierten Umgebungen (sowohl Server-, Storage- als auch Netzwerk-Virtualisierung) häufig mindestens ein Minimalsatz an Logging-Daten von jedem System bzw. jeder Anwendung erfasst werden.

Um in den Logging-Daten Analysen durchführen zu können, bieten sich bei zunehmender Datenmenge bzw. steigender Anzahl von Quellen Korrelationsverfahren an, die z.B. inhaltlich zusammengehörige Systeme und Anwendungen gruppieren. Zusätzlich kann die Korrelation bei der Aggregation von Logging-Daten für eine höhere Verdichtung der enthaltenen Informationen anhand von deren Relevanz eingesetzt werden. In der vorliegenden Arbeit wird eine Lösung beschrieben, die Syslog-Daten unterschiedlicher Systeme in einer skalierbaren NoSQL-Datenbank ablegt, für die Korrelation und Aggregation mit Metadaten anreichert und für verteilte Analysen bereitstellt.

In den nachfolgenden Abschnitten werden kurz die Grundlagen aktueller Syslog-basierter zentraler bzw. netzwerkbasierter Logging-Verfahren sowie die Eignung von NoSQL-Datenbanken für die Speicherung der Logging-Daten beschrieben. In Abschnitt 2 werden die Anforderungen an ein zentralisiertes Logging von virtuellen Systemen und Anwendungen sowie der Analyse der Logging-Daten definiert. Darüber hinaus werden verwandte Arbeiten in diesem Bereich genannt und als Ausgangspunkt für die im vorliegenden Papier durchgeführten Betrachtungen erläutert. Anschließend wird in den Abschnitten 3 und 4 eine Lösung für die automatisierte Korrelation und Aggregation sowie dessen prototypische Umsetzung vorgestellt und bezogen auf die in Abschnitt 2 genannten Anforderungen bewertet. Abschnitt 5 zieht ein Fazit und beschreibt Ergebnisse sowie Anknüpfungspunkte für weitere Verbesserungen des vorgestellten Prototyps. Zusätzlich wird ein Ausblick auf die laufende Forschung und Entwicklung in diesem Umfeld aufgezeigt.

1.1 Zentralisiertes Logging mit Syslog

Bestehende Syslog-Implementierungen [Lon01] realisieren eine Lösung für das Logging von Host- und Netzwerkereignissen und gestattet die Trennung der Systeme bzw. Software, die Nachrichten generiert, von der Software die für das Verarbeiten und Persistieren der Nachrichten verantwortlich ist. Das Syslog-Protokoll hat sich seit seiner Entwicklung zum de-facto Standard für die Verarbeitung von Logging-Events auf UNIX-Systemen sowie einer Vielzahl von Netzwerkgeräten entwickelt. Syslog-Nachrichten bestehen grundsätzlich aus drei Teilen. Der sogenannte PRI-Part enthält die nummerisch codierte Priorität sowie die Facility der Nachricht. Im unmittelbar folgenden Header befinden sich ein Zeitstempel und der Hostname des Erzeugers der Nachricht. Letzterer erlaubt die Zuordnung verschiedener Nachrichten zu einem identischen Host. Der abschließende MSG-Part beinhaltet die eigentliche Nachricht und kann ggf. zusätzliche Informationen, wie die Prozess-ID des erzeugenden Prozesses enthalten. Im herkömmlichen Betrieb werden Syslog-Nachrichten eines Hostsystems in entsprechenden Dateien im Dateisystem abgelegt. Berücksichtigt man jedoch die einleitend beschriebene, z.B. durch das Hinzufügen virtueller Maschinen rasant wachsende Anzahl von Systemen, die Daten an einen Syslog-Server übermitteln, wird deutlich, dass eine zentralisierte Erfassung und Analyse von Syslog-Informationen von essentieller Bedeutung ist. Eine Bewertung der Nachrichten über mehrere Hostsysteme hinweg kann anders nicht gewährleistet werden.

Zentralisiertes Logging und die Verwendung von Relay-Servern zur Kaskadierung von Logging-Servern in großen Umgebungen wurden daher bereits bei der Entstehung von

Syslog vorgesehen. Wurde in [Lon01] zunächst noch das *User Datagram Protocol (UDP)* als Transportmechanismus benannt, zeigt sich heute ein Wechsel zum zuverlässigen *Transmission Control Protocol (TCP)* und der Verwendung von *Transport Layer Security (TLS)* zur Sicherung und Gewährleistung der Authentizität [Naw03] [Ger09]. Abbildung 1 zeigt ein Beispiel für eine zentralisierte Logging-Umgebung.

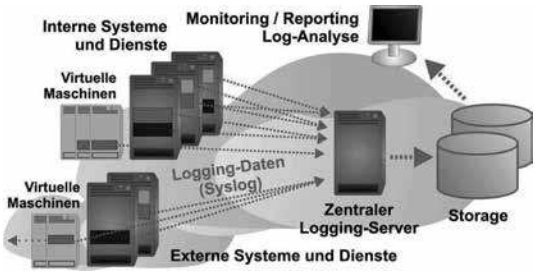

Abbildung 1: Zentralisiertes Logging von dezentralen Systemen und Diensten

Der Syslog-Server *rsyslog* [Ger12a] stellt eine quelloffene Implementierung des Syslog-Protokolls zur Verfügung und gehört neben *syslog-ng* [Sec12] zu den verbreitetsten Syslog-Servern. Eine Reihe von Plugins zur Normalisierung von Syslog-Nachrichten und zur Unterstützung einer Vielzahl neuartiger *Storage-Backends* wie MongoDB, HDFS oder ElasticSearch haben in den letzten Jahren zu einer starken Verbreitung von *rsyslog* geführt. Für unsere Untersuchungen wurde *rsyslog* als zentraler Syslog-Server gewählt.

1.2 Verwendung von NoSQL-Datenbanken für das Logging

Der Begriff NoSQL (Not only SQL) bezeichnet eine Art von Datenbanken, die in den letzten Jahren stark an Bedeutung gewonnen hat. Verschiedene Implementierungen solcher Datenbanken verfolgen sehr unterschiedliche Ansätze. Gemeinsam ist ihnen, dass sie keine relationalen (SQL-)Datenbanken sind und in der Regel hohe Performance durch Skalierbarkeit bieten. Ein Grund für die Entwicklung und die Beliebtheit von NoSQL-Datenbanken dürfte in der Vermeidung von Einschränkungen, die der Einsatz von SQL mit sich bringt, liegen. Die Struktur der Daten, die in relationale Datenbanken abgespeichert werden, muss bereits bei deren Design feststehen und kann im Nachhinein nur schwer verändert oder erweitert werden. Ebenso ist es schwierig, relationale Datenbanken für Daten zu entwerfen, die sich nicht in einer Tabellenstruktur abbilden lassen. Auch in Bezug auf die neuartige Anforderung immer größere Mengen unstrukturierter Daten (z.B. Logging-Daten) abzuspeichern und durchsuchen zu können tragen spezialisierte NoSQL-Datenbanken Rechnung. Betrachtet man die verfügbaren Lösungen im Bereich der NoSQL-Datenbanken, lassen diese sich in drei Typen klassifizieren [Lea10].

Key-Value-Stores erlauben die Verwaltung strukturierter oder unstrukturierter Daten in der Form von Schlüssel-Wert-Paaren. Diese Art der Indizierung erlaubt einen äußerst effizien-

ten Zugriff auf die Daten bei der Suche nach deren Schlüsseln, während jedoch die Suche über Werte nur mit sehr hohem Aufwand möglich ist.

Einen Schritt weiter gehen spaltenorientierte Datenbanken, die das Speichern komplexerer Datenstrukturen erlauben, ohne dabei die Beschränkungen relationaler Datenbanken mitzubringen. Dies wird beispielsweise durch die Verschachtelung von Key-Value-Paaren realisiert.

Den dritten Typ stellen dokumentenbasierte Datenbanken dar. Sie erlauben das Speichern von Daten in Form von Dokumenten, die selbst eine beliebige Anzahl an Datenfeldern bzw. Key-Value-Paaren in variabler Länge und Verschachtelung enthalten können. Häufig kommt der JSON-Standard zur Beschreibung der Dokumente zum Einsatz, wodurch ein hoher Grad an Flexibilität und Kompatibilität erreicht wird.

Basierend auf der im vorherigen Abschnitt genannten Struktur von Syslog-Daten und deren Anforderungen an die Speicherung, sind nicht alle genannten Typen gleichermaßen geeignet für ein zentralisiertes Logging. Key-Value-Stores und spaltenorientierte Datenbanken bieten beispielsweise nur die Möglichkeit, Inhalte auf Basis ihrer Schlüssel aufzufinden, und sind somit für eine flexible Suche und Korrelation weniger geeignet.

Dokumentenorientierte Datenbanken bieten die Möglichkeit, Inhalte mittels ihrer Schüssel aufzufinden, erlauben darüber hinaus jedoch auch die Suche und Korrelation über darin gespeicherte Werte. Die dokumentenbasierte Suchmaschine *ElasticSearch*, die auf der Programmbibliothek Apache Lucene basiert, weist zudem weitere interessante Eigenschaften für die Analyse der Daten, wie z.B. komplexe Suchen, auf [ela12]. Für die vorliegende Arbeit wird daher eine ElasticSearch basierte dokumentenorientierte Datenbank verwendet.

2 Aggregation und Reduktion von Logging-Informationen

Die zentralisierte Speicherung von Syslog-Informationen führt, wie einleitend dargelegt, zu einem sehr hohen Aufkommen von Logging-Events am Log- und Analyse-System. Um die entstehende Datenmenge zu bewältigen lassen sich strukturierte NoSQL-Datenbanken, wie im vorherigen Abschnitt genannt, einsetzen. Auch die Analyse der gespeicherten Daten kann durch den Einsatz dieser Technologien und der resultierenden Strukturierung der Daten beschleunigt werden. Darüber hinaus führt die steigende Datenmenge zu der Frage, ob eine automatisierte Bewertung der Relevanz von Syslog-Nachrichten zugunsten späterer Analysen möglich ist oder durch eine Reduktion der zu speichernden Daten eine weitere Steigerung der Performance erzielt werden sollte. Eine Reduktion ist jedoch nur dann realisierbar, wenn gewährleistet ist, dass enthaltene wertvolle Informationen hierbei nicht verfälscht werden oder verloren gehen. Dies kann erneut durch die Berücksichtigung von in der Struktur gespeicherten Attributen der Daten (z.B. Prioritäten) erreicht werden. Im Folgenden werden beide genannten Anforderungen detailliert erläutert.

2.1 Anforderungen an die Aggregation und Analyse

Das Hauptziel der vorliegenden Arbeit ist die Aufbereitung der durch Syslog gelieferten Informationen, sodass wichtige Vorfälle im Netzwerk oder auf einzelnen Hostsystemen auch in einer hohen Anzahl von Syslog-Meldungen sofort erkannt werden können. Betrachtet man beispielsweise den Ablauf eines SSH-Brute-Force-Angriffs anhand der in dessen Verlauf entstehenden Syslog-Nachrichten und das Vorgehen bei einer späteren Analyse des Vorfalls, so wird die Forderung nach einer automatisierten Bewertung deutlich. Während des Angriffs generiert der SSH-Daemon für jeden fehlerhaften Anmeldeversuch eine Syslog-Meldung, die zu einem zentralen Syslog-Server übertragen wird. Die Meldungen weisen zwar jeweils auf einen fehlerhaften Login-Versuch hin und gestatten somit die Bewertung der Situation beim Eintreffen des SSH-Pakets, jedoch führt die Flut an Meldungen gleichzeitig dazu, dass relevante Informationen darin verloren gehen oder in den Hintergrund treten können.

Für einen Analysten ist beispielsweise nicht die detaillierte Auflistung sämtlicher Anmeldeversuche des selben Benutzers von der selben Quelle relevant, sondern vielmehr die Information, ob es während des Brute-Force-Angriffs abschließend zu einer erfolgreichen Anmeldung kam. Zur Bewertung dieser Frage besteht das Vorgehen in der Regel zunächst in der Filterung der aufgezeichneten Syslog-Nachrichten nach solchen des zugehörigen SSH-Daemons, gefolgt von der Suche nach einer eventuell auf mehrfach fehlerhafte Login-Versuche schließlich folgende erfolgreiche Anmeldung. Wird eine erfolgreiche Anmeldung gefunden, die von der gleichen Quelle stammt, aus der zuvor eine Flut fehlerhafter Anmeldeversuche stammte, so kann mit zunehmender Anzahl der Fehlversuche davon ausgegangen werden, dass es sich mit hoher Wahrscheinlichkeit um einen schließlich erfolgreichen Brute-Force-Angriff handelt. Die zeit- und arbeitsaufwändige Suche nach einem solchen Angriffsverhalten kann durch die automatisierte Bewertung der Syslog-Meldungen vereinfacht werden. Voraussetzung ist die normalisierte Speicherung von Syslog-Nachrichten durch den zentralen Syslog-Server, um zu erreichen, dass Syslog-Nachrichten, die das gleiche Ereignis beschreiben, unabhängig von der Art oder Softwareplattform des generierenden Gerätes identisch sind. Die gesuchten fehlerhaften und erfolgreichen Anmeldeversuche können jeweils anhand einer relevanten Syslog-Nachricht erkannt werden, was die automatisierte Suche nach erfolgreichen Anmeldungen in unmittelbarer Folge auf eine Reihe fehlerhafter Anmeldeversuche erlaubt. Das Auffinden eines solchen Ereignisses kann schließlich zur Generierung einer neuen Syslog-Nachricht mit höherer Priorität genutzt werden, um die sofortige Erkennung erfolgreicher Angriffe auf überwachte Systeme zu ermöglichen.

2.2 Anforderungen an die Reduktion von Logging-Daten

Ein zweites Ziel, das im Zuge der vorliegenden Arbeit untersucht wurde, ist die Reduktion der Menge an zu persistierenden Syslog-Nachrichten durch die im vorherigen Abschnitt genannte Aggregation. Diese mag vor dem Hintergrund zunehmend skalierbarer Verarbeitung von großskaligen Daten (vgl. BigData) zweitrangig erscheinen, jedoch ist auch

hierbei eine Steigerung der Performance späterer Analysen möglich, wenn die Gesamtmenge an Daten, über die gesucht werden muss, verringert werden kann. Die Reduktion von Syslog-Nachrichten zu Gunsten der Performance wird daher in der Praxis schon seit langem eingesetzt. Ansätze wie das Entfernen von Syslog-Meldungen, die ein gewisses Alter überschritten haben oder unter einer gewissen Priorität (severity) eingeordnet werden, führen nur teilweise zu einem akzeptablen Ergebnis. Zwar kann hierdurch eine Reduktion der Daten und damit eine Erhöhung der Performance bei der Analyse erzielt werden, jedoch führt die Vorgehensweise zu einem Verlust von Informationen.

Unser Ansatz sieht hier zunächst die Gruppierung gleichartiger bzw. wiederkehrender Ereignisse basierend auf deren Struktur bzw. Attributen vor. Auf Basis dieser Gruppen ist uns die Generierung neuer Syslog-Ereignisse und das anschließende Verwerfen der in einer Gruppe befindlichen Ereignisse möglich. In der nun abgespeicherten, neuen Syslog-Nachricht befinden sich alle Information über die ursprünglichen, in einer Gruppe zusammengefassten, Nachrichten. Es kann also eine Reduktion der Daten um wiederkehrende Nachrichten ohne den Verlust der relevanten Information erreicht werden. Gleichzeitig kann die Manipulation (z.B. Steigerung) der Priorität solcher neu erzeugter Nachrichten wertvollere Hinweise über den Status der überwachten Server-Systeme bieten.

Am Beispiel des in Abschnitt 2.1 beschriebenen SSH-Brute-Force-Angriffs wurde bereits deutlich, dass eine relevante Nachricht hoher Priorität für die Bewertung der Situation höheren Wert besitzt, als die Flut gleichartiger Nachrichten, die eine manuelle Analyse nach sich zieht. Denkbar ist daher, die große Menge von Nachrichten durch den zentralen Syslog-Server nicht zu persistieren und dadurch die Performance von Analysen zu steigern. Eine Betrachtung der einzelnen Syslog-Events ist in den Log-Dateien der entsprechenden Host-Systeme unabhängig davon weiterhin möglich. Darüber hinaus ist die Reduktion wiederkehrender Nachrichten z.B. von Cron- oder SNMP-Diensten realisierbar.

2.3 Verwandte Arbeiten

Die Herausforderung der Speicherung und Auswertung von dezentralen Logging-Informationen wurde in einigen Veröffentlichungen bereits betrachtet. Ein Beispiel hierfür bildet die Auswertung von dezentralen Logging-Informationen in IaaS, PaaS und SaaS Cloud-Infrastrukturen, wie sie in [Mar11] beschrieben wird. Für das Logging von Syslog-Daten verteilter Cloud-Anwendungen befindet sich zudem ein Internet-Draft in Entwicklung [GBJ12]. Neben den Anforderungen durch diese zunehmend verteilten Anwendungen, besteht eine wesentliche Herausforderung der Analyse von Logging-Informationen in deren Strukturierung. Existierende automatisierte Log-Analyzer lösen hierbei nur einen Teil der Anforderungen [Jay12]. In [Jay12] wird daher die Strukturierung der in den Logging-Daten enthaltenen Informationen empfohlen. Hierfür werden aufgrund variabler Felder verschiedener aggregierter Logging-Formate NoSQL-Datenbanken vorgeschlagen, die eine adaptive Strukturierung der Daten durch unterschiedliche Tabellen-Schemata (bzw. document-based NoSQL-Datenbanken mit key-value Paaren) erlauben. Die eigentlich Analyse der Daten kann hierbei durch *Event Correlation* und *Event Detection*, wie sie in [MGT+09] und [GMMP12] beschrieben werden, automatisiert erfolgen. Beide

Veröffentlichungen beschreiben auch die Verwendung der in diesem Paper verwendeten Korrelations-Lösung *Drools*. Die Korrelation ermöglicht hierbei auch eine Reduktion (bzw. Konsolidierung) der Logging-Daten, wobei nur für die spätere Analyse relevante Informationen erhalten, und somit die Logging-Daten verdichtet werden. In [MGM11] und [GMMP12] wird die Reduktion von eingehenden dezentralen Syslog-Daten mit bis zu 99% angegeben. Eine auf der NoSQL-Datenbank *mongoDB* basierende Lösung, die mittels Map-Reduce eine Korrelation und Aggregation von Logging-Daten in verteilten Cloud-Analyse-Farmen ermöglicht, wird in [WZJ+11] vorgestellt. Hierbei werden jedoch keine Event Correlation und Detection Verfahren umgesetzt.

3 Automatisierte Aggregation und Reduktion von Logging-Daten

Die von uns umgesetzte Implementierung verwendet *rsyslog* [Ger12a] als zentralen Syslog-Server, für den Empfang und die Normalisierung von Logging-Nachrichten. Nach Durchführung der Normalisierung, die eine einheitliche Weiterverarbeitung der Syslog-Nachrichten gestattet, werden die Nachrichten im JSON-Format an den Log-Korrelator weitergeleitet. Unser in Java implementierter Korrelator nimmt die Analyse und Persistierung von Nachrichten in einem ElasticSearch-Cluster vor. Im Korrelator kommt die *Complex Event Processing* Engine *Drools Fusion* [Com12] zum Einsatz, die eine Formulierung von Regeln mit Bezug auf die zeitliche Relation von Nachrichten ermöglicht.

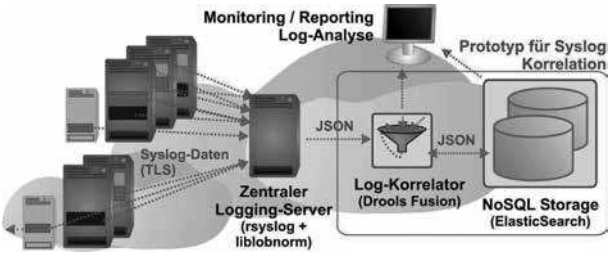

Abbildung 2: Korrelation und Aggregation von zentralisierten Logging-Daten

Abbildung 2 zeigt den Aufbau unserer Testumgebung. Syslog-Nachrichten werden von einem Clientsystem über eine gesicherte Verbindung an den zentralen Syslog-Server geleitet und dort zunächst mittels *liblognorm* [Ger12b] normalisiert. Der ausschließlich als Relay arbeitende Syslog-Server leitet die normalisierten Nachrichten anschließend in Form von JSON-Datensätzen an den in Java implementierten Korrelator weiter. Im Korrelator werden die Nachrichten für die Analyse vorgehalten und gleichzeitig in einem ElasticSearch-Cluster persistiert. Die Vorhaltung der Daten durch Drools im Arbeitsspeicher geschieht zur Bewertung der Nachrichten anhand der von uns festgelegten, erweiterbaren Regeln. Datensätze, die zeitlich oder thematisch nicht auf Regeln zutreffen können, werden durch Drools automatisch aus dem Arbeitsspeicher verworfen, sind jedoch für später durchzuführende Analysen in ElasticSearch verfügbar.

4 Korrelation und Aggregation von Logs in NoSQL-Datenbanken

Die konkrete Umsetzung unseres Korrelationsansatzes von Syslog-Nachrichten soll am Beispiel einer SSH-Brute-Force Attacke, wie sie bereits für die Anforderungen an unsere Lösung in Abschnitt 2.1 erläutert wurde, verdeutlicht werden. Ziel ist die Erzeugung einer neuen Syslog-Nachricht mit hoher Priorität, für den Fall, dass ein erfolgreicher SSH-Login in unmittelbarer Folge eines Passwort-Brute-Force-Angriffs erkannt wird. Die Bewertung dieses Sachverhaltes soll dabei nach initialer Definition der entsprechenden Regeln automatisch erfolgen. Für die Korrelation von Syslog-Ereignissen, die auf einen erfolgreichen Brute-Force-Angriff auf das Passwort hinweisen, müssen zunächst die entsprechenden Syslog-Meldungen ausgewählt und in den am zentralen Syslog-Server ankommenden Daten aufgefunden werden. Die Meldungen des SSH-Daemon für eine fehlerhafte bzw. erfolgreiche Anmeldung können nach der Normalisierung mittels der in Listing 1 dargestellten Syslog-Nachrichten erkannt werden. Drools Fusion erlaubt basierend auf diesen Informationen die Formulierung einer Regel, die genau dann zutrifft, wenn ein erfolgreicher SSH-Login innerhalb einer vorgegebenen Zeitspanne auf eine bestimmte Anzahl fehlerhafter SSH-Loginversuche folgt.

```
Failed password for root from 192.168.1.1 port 34201 ssh2
Accepted password for root from 192.168.1.1 port 34201 ssh2
```

Listing 1: Syslog-Nachricht eines fehlerhaften bzw. erfolgreichen SSH-Login

Listing 2 veranschaulicht die von uns definierten Drools Regeln, wobei erkennbar ist, dass die *failed*-Regel das Auftreten der zuvor definierten success-Regel innerhalb von 20 Minuten (`this before[0,20m] success`) nach Auftreten einer Anzahl zutreffender Syslog-Nachrichten (`size >= 10`) fordert.

```
success : Message(
  message matches
    "Accepted password for [^\\s]* from [^\\s]* port [^\\s]* ssh2")
failed : ArrayList( size >= 10 ) from collect(
  Message(this before[0,20m] success,
    message matches
      "Failed password for [^\\s]* from [^\\s]* port [^\\s]* ssh2") )
```

Listing 2: Drools-Regeln zum Erkennen erfolgreicher SSH-Brute-Force-Angriffe

Das Zutreffen der Regel führt in unserer derzeitigen Implementierung zum Erzeugen einer neuen Syslog-Nachricht, die der Facility *Security* mit der Priorität *Emergency* zugeordnet ist und die Nachricht *Möglicher erfolgreicher SSH-Brute-Force-Login* enthält. Denkbar ist statt des Einfügens einer neuen Nachricht auch die Aufwertung der Severity der zu persistierenden Nachrichten, oder der Verzicht auf die Persistierung dieser Nachrichten zugunsten der neu erzeugten, relevanten Nachricht. Syslog-Nachrichten, die für die Auswertung der angegebenen Regeln benötigt werden, hält Drools selbständig im dafür vorgesehenen Speicherbereich. Anschließend werden die Nachrichten automatisch aus dem Arbeitsspeicher entfernt.

5 Fazit und Ausblick

In den vorangegangenen Abschnitten wurde eine Lösung für die automatisierte Korrelation und Aggregation von Syslog-Nachrichten vorgestellt. Neben der Definition der Anforderungen und eines Konzepts für die Umsetzung wurde darüber hinaus eine prototypische Implementierung realisiert. Der implementierte Prototyp ist bereits in der Lage einige der im Abschnitt 2 definierten Anforderungen zu erfüllen. Er erlaubt sowohl die Reduktion (Verdichtung) von Nachrichten durch eine entsprechende Gruppierung und beschleunigt dadurch auch die Auswertung der Logging-Daten. Dies unterstützt die Analyse von Logging-Informationen insb. bei einer großen Anzahl von Systemen (vgl. die schnelle Bereistellung von Systemen und Diensten in Form von virtuellen Maschinen), die Logging-Daten erzeugen. Hierbei ließe sich der Prototyp auch beispielsweise in bestehende Monitoring-Systeme integrieren, um z.B. verdichtete Logging-Daten anhand deren Relevanz innerhalb des Monitorings z.B. durch Traps bzw. Versand von priorisierten Messages (Events) eskalieren zu lassen (vgl. Abschnitt 2.1 und 4). Eine Integration in bestehende Monitoring-Umgebungen (z.B. OpenNMS, splunk) bildet eines der nächsten Ziele für die Weiterentwicklung der hier vorgestellten Lösung. Hier können darin existierende Schnittstellen für die Korrelation (z.B. für die Root-Cause-Analyse) ggf. erweitert werden. Diese können auch bidirektional an den Prototypen angebunden werden, um die Logging-Daten mit weiteren Informationen (z.B. aus dem Asset-Management, bzw. Informationen zum System im Monitoring z.B. Wartungs-Intervalen etc.) anzureichern.

Eine Einschränkung des prototypischen Ansatzes bildet die Auswertung der Regeln für die Korrelation und Aggration von Logging-Daten im Arbeitsspeicher und anschließende Persistierung im angebundenen NoSQL-Storage. Dadurch ist die Korrelation und Aggregation der Logging-Daten auf die Größe des zugewiesenen Arbeitsspeichers begrenzt. Theoretisch können die vom Prototypen definierten Attribute innerhalb der NoSQL-Datenbank auch für die Korrelation und Aggregation über alle gespeicherten Logging-Informationen verwendet werden. Hierfür sind für eine zukünftige Weiterentwicklung entsprechende Evaluierungen geplant, die den zusätzlichen Overhead (Latenz, Datentransfer) für die Suche zu korrelierender Einträge im nachgelagerten NoSQL-Storage betrachten. Eine zusätzliche Erweiterung könnte die automatisierte Erkennung von Anomalien, basierend auf den persistierten Daten, bilden. Dies würde die Definition von Regeln für die Korrelation und Aggregation erleichtern. Durch Syslog-basiertes Event Forecasting, für das der NETdradamus Ansatz [CH10] einen interessanten Ansatz liefert, könnte die Definition von Regeln sogar teilweise entfallen oder automatisiert werden. Diese Ansätze ermöglichen auch eine effektive Aggregation von Logging-Daten durch die Erkennung häufig wiederkehrender Messages. Durch die Realisierung des vorgestellten Prototypen als skalierbaren Cloud Service (mittels OpenStack) soll dessen Einsatz auch für großskalige Logging-Daten und komplexen Regelwerke evaluiert werden. Eine Anbindung weiterer Logging-Systeme, die als Alternativen zu Syslog zur Verfügung stehen (vgl. applikationsspezifisches Logging wie log4j) könnte ebenfalls den Einsatzbereich des Prototypen erweitern.

Literatur

[CH10] A. Clemm und M. Hartwig. NETradamus: A forecasting system for system event messages. In *Network Operations and Management Symposium (NOMS), 2010 IEEE*, Seiten 623–630, 2010.

[Com12] JBoss Community. Drools - JBoss Community. `http://www.jboss.org/drools/`, 2012. abgerufen am 12.12.2012.

[ela12] elasticsearch. elasticsearch - - Open Source, Distributed, RESTful, Search Engine. `http://www.elasticsearch.org`, 2012. abgerufen am 12.12.2012.

[GBJ12] G. Golovinsky, D. Birk und S. Johnston. Syslog Extension for Cloud Using Syslog Structured Data - draft-golovinsky-cloud-services-log-format-03. *Internet-Draft, IETF*, 2012.

[Ger09] R. Gerhards. RFC 5424: The Syslog Protocol'. *Request for Comments, IETF*, 2009.

[Ger12a] R. Gerhards. The enhanced syslogd for Linux and Unix rsyslog. `http://www.rsyslog.com`, 2012. abgerufen am 12.12.2012.

[Ger12b] R. Gerhards. A Syslog normalization library. `http://www.liblognorm.com`, 2012. abgerufen am 23.12.2012.

[GMMP12] M.R. Grimaila, J. Myers, R.F. Mills und G. Peterson. Design and analysis of a dynamically configured log-based distributed security event detection methodology. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, 9(3):219–241, 2012.

[Jay12] D. Jayathilake. Towards structured log analysis. In *Computer Science and Software Engineering (JCSSE), 2012 International Joint Conference on*, Seiten 259 –264, 30 2012-june 1 2012.

[Lea10] N. Leavitt. Will NoSQL databases live up to their promise? *Computer*, 43(2):12–14, 2010.

[Lon01] C. Lonvick. RFC 3164: The BSD syslog protocol. *Request for Comments, IETF*, 2001.

[Mar11] R. Marty. Cloud application logging for forensics. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, Seiten 178–184. ACM, 2011.

[MGM11] Justin Myers, Michael R. Grimaila und Robert F. Mills. Log-Based Distributed Security Event Detection Using Simple Event Correlator. *Hawaii International Conference on System Sciences*, 0:1–7, 2011.

[MGT+09] A. Müller, C. Göldi, B. Tellenbach, B. Plattner und S. Lampart. Event Correlation Engine. *Department of Information Technology and Electrical Engineering - Master's Thesis, Eidgenössische Technische Hochschule Zürich*, 2009.

[Naw03] Kenneth E. Nawyn. A Security Analysis of System Event Logging with Syslog. 2003.

[Sec12] BalaBit IT Security. syslog-ng - Multiplatform Syslog Server and Logging Daemon. `http://www.balabit.com/network-security/syslog-ng`, 2012. abgerufen am 23.12.2012.

[WZJ+11] Jianwen Wei, Yusu Zhao, Kaida Jiang, Rui Xie und Yaohui Jin. Analysis farm: A cloud-based scalable aggregation and query platform for network log analysis. In *Cloud and Service Computing (CSC), 2011 International Conference on*, Seiten 354 –359, dec. 2011.