Analyse von IAM-Anwendungslogs

Tobias Hülsken, Roland H. Steinegger, Dr. Sebastian Abeck, Karlsruher Institut für Technologie

Florian Röser, Dr. Nadina Hintz, iC Consult Gesellschaft für Systemintegration und Kommunikation mbH

Die Anwendungslogs der Identitäts- und Zugriffs-Managementsysteme (engl. Identity and Access Management, IAM) einer Web-Anwendung können zentrale Informationen über Zugriffsprobleme oder Angriffe liefern. Der Artikel zeigt die Konfiguration von ElasticSearch, Logstash und Kibana für ein großes Automobil-Unternehmen. Zudem ist dargestellt, wie Logs mit Watcher live überwacht werden können und wie Timelion zur grafischen Aufbereitung und Auswertung genutzt werden kann. Abschließend sind die Erfahrungen mit den Werkzeugen sowie Lösungen für aufgetretene Probleme zusammengefasst.

Wer kennt die Situation nicht? Der Kunde meldet, dass er sich nicht an der Web-Anwendung anmelden kann oder dass ihm der Zugriff verwehrt wird. Vermutlich liegt ein Problem in den Identitäts- und Zugriffs-Managementsystemen vor. Jetzt heißt es, den Fehler möglichst schnell zu lokalisieren und zu beheben.

Die Identifikation der Problemstelle in einer verteilten Web-Anwendung kann sich als schwieriges Unterfangen erweisen. Meist sind Anwendungslogs die Hauptquelle für Informationen über die vergangenen Anfragen. Im schlimmsten Fall sind diese nicht zentral gespeichert und der Fehler bezieht sich nicht nur auf ein einzelnes System. Im besten Fall ruft der Kunde gar nicht erst an, weil der Fehler automatisch erkannt und bereits behoben wurde.

Anhand eines Szenarios wird das Zusammenführen von Logs verteilter Anwendungen an einer zentralen Stelle gezeigt und ebenso, wie durch eine Bedrohungsanalyse die Auswertungsgrafiken angepasst werden können. In diesem Beispiel kommen die bekannten Open-Source-Vertreter ElasticSearch, Logstash und Kibana, der ELK-Stack, zum Einsatz. Sie sind ergänzt um Watcher, um automatisch Probleme der Web-Anwendung zu erkennen, sowie um das Kibana-Plug-in Timelion, um das Auffinden von Problemen durch den Vergleich von Zeitreihen zu erleichtern.

ElasticSearch, Logstash und Kibana

Der ELK-Stack besteht im Grundaufbau aus den Tools ElasticSearch, Logstash und Kibana. Diese drei Tools kommunizieren jeweils über eine Web-Schnittstelle miteinander. Damit sind sie austauschbar und mit Eigen- oder Fremd-Entwicklungen kombinierbar. Zudem ist die Einbindung von Plug-ins zur Erweiterung der Funktionalitäten möglich *(siehe Abbildung 1)*. Ihr Aufgabenbereich beginnt beim Einlesen der Log-Dateien und endet mit der visuellen Darstellung.

Die Verarbeitungspipeline beginnt mit Logstash *[1]*. Dessen Zielsetzung ist die Vorbereitung der Logdaten für die Analyse. Dazu ist das Tool in eine dreistufige Pipeline eingeteilt. Die erste Stufe ist „Input". Auf ihr werden die Daten zunächst mithilfe von Filebeat (einer leichtgewichtigen Logstash-Variante) zentralisiert. Unterstützte Eingabe-Quellen sind beispielsweise Sensoren, IoT-Komponenten, Logs und Datenstreams. Zudem sind bereits grundlegende Funktionen wie das Zusammenfügen von mehrzeiligen Logs möglich.

Die zweite Stufe ist „Filter". Dieser Bereich ist für die Überführung der Log-Dateien in das einheitliche und normalisierte Datenaustausch-Format JSON gedacht. Um dies zu bewerkstelligen, gibt es zahlreiche Logstash-Plug-ins *[2]*. Es existieren unter anderem Plug-ins zum Anwenden von Regex-Ausdrücken, Parsen von Zeitstempeln oder auch zum Zuordnen von geografischen Lagen anhand der IP-Adresse.

Die letzte Stufe ist „Output". Auf dieser wird das zuvor erzeugte JSON-Dokument weitergeleitet. Hier lassen sich neben der reinen Weiterleitung noch Konfigurationen für die Übertragungsart (UDP, TCP), das Ziel (ElasticSearch, Datei) oder auch für die nachstehenden Dienste, wie die Anzahl der Worker-Threads von ElasticSearch, einstellen. Um noch nicht umgesetzte Funktionalitäten einbinden zu können, ist es möglich, eigene Plug-ins mittels Ruby zu entwickeln und über den Logstash-Plug-in-Manager einzubinden.

Die nächste Stufe der Verarbeitungspipeline ist der Datenspeicher ElasticSearch *[3]*. Er basiert auf Apache Lucene und bietet als NoSQL-Datenbank nahezu Echtzeit-Funktionalitäten. Zu den unterstützten Funktionen zählen das Aggregieren, Suchen und Verwalten der Daten. ElasticSearch ist dezentral aufgebaut. Hierfür besitzt jedes Cluster einen Namen als Kennung. Über diesen können sich neue Knoten selbstständig anschließen. Die Daten haben jeweils

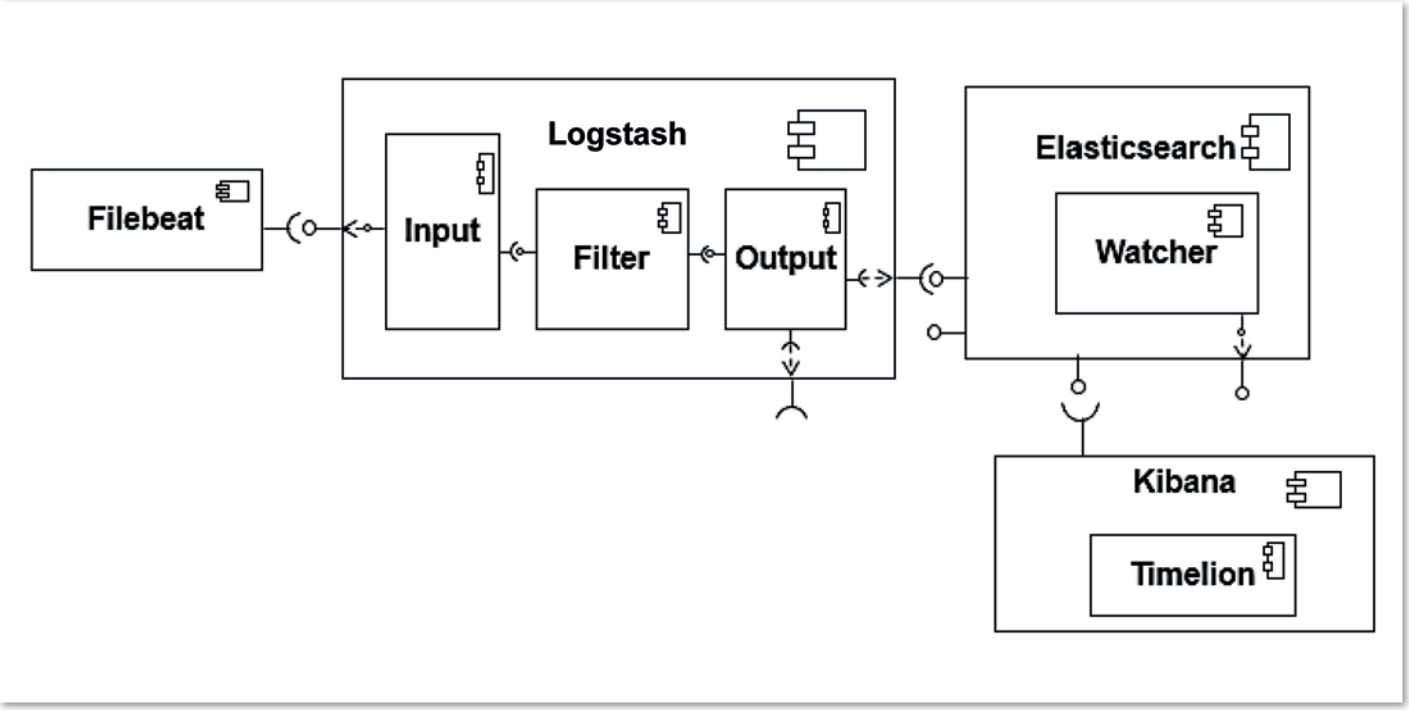

Abbildung 1: Komponenten-Diagramm des ELK-Stacks mit vorgeschaltetem Filebeat zur Logdaten-Sammlung und den Plug-ins Watcher und Timelion

einen Index, der ihre Zugehörigkeit definiert. Zur Verteilung und Replizierung von Daten verwendet ElasticSearch für jeden Index sogenannte „Shards" und „Replikas". Shards sind Teile, auf die die Daten eines Index aufgeteilt sind. Replikas sind Kopien dieser Shards. Shards und Replikas werden dann entsprechend der Konfiguration über der Gesamtanzahl an Knoten dezentral angelegt. Dies erhöht sowohl die Ausfallsicherheit als auch die Geschwindigkeit.

Die letzte Stufe der Verarbeitungspipeline ist das Anzeigetool Kibana *[4]*. Es besitzt eine Web-Oberfläche zum Analysieren und Visualisieren von ElasticSearch-Indizes. Die Oberfläche ist in die drei Bereiche „Discover", „Visualize" und „Dashboard" aufgeteilt. Mithilfe der Discover-Ansicht lassen sich einzelne Einträge anzeigen und filtern. Zudem werden die JSON-Einträge anhand ihrer Schlüssel vorausgewertet, um einen Überblick über die häufigsten Werte zu liefern.
In der Visualize-Ansicht lassen sich Diagramme anfertigen – Schritt für Schritt durch die GUI geleitet. Die Datenbankanfragen können durch das Interface spezifiziert oder manuell mit einer Anfrage in der Apache-Lucene-Query-Syntax erstellt werden.

Dashboard, die letzte Ansicht, dient zur Zusammenstellung mehrerer, in der Visualize-Ansicht erstellter Diagramme. Dies erleichtert den Vergleich verschiedener Daten und das Erkennen von Zusammenhängen.

Watcher

Watcher *[5]* ist ein Plug-in für ElasticSearch, um Anomalien zu erkennen. Diese werden durch sogenannte „Watcher" identifiert. Sie bestehen aus den vier Bestandteilen „Trigger", „Input", „Condition" und „Action". Die Conditions werden periodisch oder nach bestimmten Ereignissen für die aus dem Input stammenden Daten überprüft. Sind die Conditions erfüllt, werden die Actions ausgeführt. Dies kann beispielsweise das Senden einer E-Mail oder auch das Ausführen eines Web-Hook sein, eine auf HTTP aufbauende Server-Kommunikation. Generell lassen sich auch Skripte einbinden. Unterstützte Formate sind unter anderem Groovy oder über weitere Plug-ins auch JavaScript und Python.

Timelion

Timelion *[6]* ist ein Anzeige-Plug-in für Kibana, das im Gegensatz zu den meisten Plug-ins nicht nur neue Diagramm-Typen einführt, sondern eine eigenständige App innerhalb von Kibana ist. Es kann dazu genutzt werden, verschiedene Datenquellen in einem Diagramm zusammenzuführen. Ergänzend zu den bordeigenen Mitteln ist es mit Timelion möglich, mathematische Operationen auf verschiedene Datensätze anzuwenden. So sind einfache Operationen, wie die Addition von Diagramm-Verläufen oder das Hinzufügen von zeitlichen Offsets, möglich. Daten lassen sich auf diese Weise einfacher vergleichen, da ihre Beziehungen und Relationen in einem Diagramm sichtbar sind. So kann beispielsweise das Login-Verhalten bei einem Server wochenweise verglichen werden.

Der ELK-Stack zur Log-Aufarbeitung

Werden die Logs dezentral erzeugt, müssen sie an eine zentrale Logstash-Instanz weitergeschickt oder abgeholt werden, um den späteren Verwaltungsaufwand, beispielsweise das Anpassen der Konfigurationen, zu minimieren. Um dies zu realisieren, kann Filebeat benutzt werden, wobei es ausreicht, den Speicherort der Logs und die Logstash-Instanz anzupassen. In *Listing 1* sind die zu ändernden Stellen (Zeile 2 und 5) als Auszug der Filebeat-Konfiguration aufgelistet.

```
paths:
    %PathToLogfiles%/*.log
output:
  logstash:
    hosts: ["127.0.0.1:5044"]
```

Listing 1

```
input {
  beats {
    port => 5044
    codec => multiline {    #For Multiline Log Entrys
      pattern => "^%{TIMESTAMP_ISO8601} "
      negate => true
      what => "previous"
    }
  }
}
```

Listing 2

```
filter {
  if [message] =~ /^[\s]*$/   #removing empty lines
  { drop {} }
  grok {
    match => { "message" => "%{TIMESTAMP_
ISO8601:timestamp} %{GREEDYDATA:logmessage}" }
  }
  grok {
    patterns_dir => "./custom-grok-patterns"
    match => { "logmessage" =>"\
AR{%{REQUESTID:[producer][requestId]}?}
T{%{TRACKINGID:[producer][trackingId]}
IP{%{IP:clientIP}?}- %{GREEDYDATA:logmessage}" }
    overwrite => ['logmessage']
  }
  yamlgrok {
    patterns_dir => "./custom-grok-patterns"
    path => "./yamlgrok-files/backend.yml"
  }
  mutate {  #Converts the Thread Number and adds the
appname
    convert => { "[producer][thread]" => "integer" }
    add_field => { "[producer][appname]" => "backend"}
  }
}
```

Listing 3

Zudem muss Logstash heruntergeladen und konfiguriert werden. Es reicht aus, das Programm zu entpacken; eine Installation ist nicht nötig. Für die Konfiguration empfiehlt es sich, dieses in drei Bereiche „Input", „Filter" und „Output" aufzuspalten, um die jeweiligen Konfigurationen für andere Projekte oder Indizes wiederverwenden zu können.

Für den ersten Bereich „Input" muss angegeben werden, aus welcher Quelle die Logs stammen (Filebeat, Datei etc.). Als Beispiel wird eine Input-Konfiguration für Filebeat herangezogen, die Daten vom Port 5044 empfängt. Optional können mehrzeilige Einträge in diesem Bereich anhand von Zeitstempeln zusammengefasst werden (siehe *Listing 2*, Zeilen 4 bis 8). Danach kann der Log-Eintrag im Bereich „Filter" geparst werden. *Listing 3* zeigt Auszüge aus einer Filter-Konfiguration.

Zu Beginn werden Leerzeilen entfernt und die Log-Nachricht vom Zeitstempel getrennt. Danach wird die Log-Nachricht in ein hierarchisches Format geparst. Die dafür notwendigen Muster sind der Übersicht und Wiederverwendbarkeit wegen ausgelagert. Danach werden mithilfe von Yamlgrok, einem Logstash-Plug-in zum Identifizieren von String-Bausteinen anhand der Yet Another Markup Language (YAML), erst ein String- und dann ein Regex-Vergleich durchgeführt. Dies erlaubt die Volltextsuche durch Grok *[7]*, einem Logstash-Plug-in, um Daten anhand eines Musters zu strukturieren, zu vermeiden und somit Rechenaufwand zu minimieren. In den Zeilen 20 und 21 wird dem Feld „Thread" der Datentyp „Integer" zugewiesen und das Feld „Appname" erhält den Wert „Backend"; beide sind Attribute des Producer-Objekts, das seinerseits eine Sammlung von Informationen zu der Maschine enthält, die die Logs erstellt. Die Konvertierung zum Datentyp „Integer" spart zum einen Speicherplatz in ElasticSearch, zum anderen ermöglicht es weitere Operationen während der Auswertung in Kibana. Die Zuweisung des Wertes „Backend" als Applikationsname bietet bei der Auswertung das Zuordnen der verschiedenen Log-Quellen zu den einzelnen Applikationen.

Zuletzt ist der Output zu konfigurieren. Im Beispiel in *Listing 4* wird als Zielsystem eine ElasticSearch-Instanz auf dem „localhost" eingestellt. Zudem wird der Index „ciam-access-<Datum>" gesetzt. Die Wahl eines guten Index ist wichtig, da nur über diesen performant aggregiert werden kann.

Der nächste Schritt ist das Installieren und Konfigurieren von ElasticSearch. Dieses ist nach dem Download bereits voll funktionsfähig. Zur Laufzeit können mittels REST-API weitere Einstellungen wie die Anzahl der Shards oder das Löschen von alten Indizes getroffen werden. Als Beispiel soll das Benutzen einer Vorlage dienen, die zur Konfiguration der Datenstruktur und der Feld-Eigenschaften in ElasticSearch zum Einsatz kommt. Damit wird zum einen Typsicherheit garantiert und zum anderen Speicherplatz gespart.

Die Vorlage wird per HTTP-PUT an ElasticSearch geschickt. Sie definiert das Datenformat der Felder des jeweiligen Index und ob diese Felder analysiert werden. Das Analysieren ermöglicht später eine schnelle Volltextsuche nach einzelnen Begriffen innerhalb des jeweiligen Feldes, benötigt jedoch ein Vielfaches an Speicher. Ein nicht analysiertes Feld erlaubt nur noch einen Volltextvergleich *(siehe Listing 5)*.

Um mit Kibana fortzufahren, muss ElasticSearch gestartet sein. Es empfiehlt sich, erste Log-Einträge mit Logstash in ElasticSearch eingespielt zu haben, um die Kibana-Funktionalitäten bereits nutzen zu können. Danach ist Kibana standardmäßig unter dem Port 5601 im Webbrowser verfügbar.

Auf der Startseite von Kibana, etwa „localhost:5601", ist zunächst ein Index anzugeben. Dieser besteht aus einer Zeichenkette und optional einem „*" (Asterisk), um verschiedene Indizes zu vereinen. Die Benutzung ist intuitiv und einfach gehalten.

Im Bereich „Discover" sind links die verschiedenen Felder des Index aufgelistet. Zu jedem Feld lässt sich eine Statistik der fünf häufigsten Werte anzeigen. Zudem werden die einzelnen Einträge aufgelistet, die expandiert und somit übersichtlicher dargestellt werden können.

```
output {
  elasticsearch {
    action => "index"
    hosts => "localhost"
    index => "ciam-backend-%{+YYYY.MM.dd}"
  }
}
```

Listing 4

```
PUT /_template/template_1
{
  "template" : "ciam-backend*",
  "mappings" : {
    "_default_" : {
      "properties" : {
        "clientip" : {"type": "string", "index" :
"not_analyzed" },
      }
    }
  }
}
```

Listing 5

Im „Visualize"-Bereich lassen sich nun diverse Diagramme erstellen. Hierzu müssen als Erstes der Diagramm-Typ (wie Balken- oder Kuchendiagramm) und das darzustellende Feld einer Datenquelle ausgewählt werden. Für das gewählte Feld muss zudem die Aggregationsmethode (etwa als Histogramm) eingestellt sein. Zusätzliche Anpassungen sind der anzuzeigende Wertebereich oder das Ausblenden bestimmter Werte gemäß einem Muster. Das erstellte Diagramm kann daraufhin gespeichert und in der Dashboard-Ansicht genutzt werden. Sowohl in der Visualize- als auch in der Dashboard-Ansicht lassen sich die Zeit-Intervalle der dargestellten Daten einstellen.

Überwachung der Logs mit Watcher

Watcher ist ein kommerzielles Plug-in für ElasticSearch. Dementsprechend ist eine Lizenz zur Installation erforderlich. Nach der Installation wird Watcher beim Ausführen von ElasticSearch automatisch gestartet. Die Watches lassen sich mit dem REST-API oder übersichtlicher mit dem Kibana-Plug-in Sense verwalten.

„Trigger" bestimmt, in welchen Intervallen eine Watch ausgeführt wird. „Inputs" können sowohl HTTP-Anfragen auf einen Webserver als auch Anfrage-Ergebnisse von ElasticSearch sein. Diese lassen sich verketten und aggregieren. „Condition" überprüft, ob die durch „Actions" definierten Aktionen ausgeführt werden oder nicht. Das können einfache numerische Operationen wie „größer oder gleich", textuelle Vergleiche oder komplexere Überprüfungen sein, die als Skript definiert sind. „Actions" beschreiben schließlich die auszuführenden Aktionen. Das können beispielsweise das Senden einer Mail, das Schreiben in eine Log-Datei oder auch das Senden einer Nachricht an einen Webhook sein.

Manuelle Anomalie-Erkennung mit Timelion

Um Timelion zu benutzen, müssen bereits Daten in ElasticSearch abgelegt sein. Zur Installation reichen der Befehl „%Kibana_DIR%/bin/kibana plugin -i elastic/timelion" und der anschließende Neustart von Kibana aus. Timelion wird nun in der Rubrik „Apps" in der Webansicht von Kibana aufgeführt. Die Ansicht in Timelion ist in ein

Feld für Anfragen sowie in einen Bereich aufgeteilt, der maximal 16 x 16 Diagramme (Visualisierungen von Einträgen) enthalten kann.

Um mit einem Diagramm zu arbeiten, muss dieses erst mit „add Chart" erstellt und anschließend ausgewählt werden. Nun lässt sich das Diagramm konfigurieren, indem man die Anfrage inkrementell aufbaut. Die Anfrage beginnt mit dem Index, der mathematische Operationen wie das Dividieren oder Addieren von Anfragen oder auch das Anzeigen mehrerer Graphen innerhalb eines Diagramms erlaubt. Eine der interessantesten Anwendungen ist es, Daten derselben Quelle zu verschiedenen Zeiten, beispielsweise im Abstand von einer Woche, miteinander zu vergleichen.

Die Anfragen in *Listing 6* zeigen die Funktionsweise. In der ersten Anfrage wird die Anzahl der eintreffenden Logs zum aktuellen Zeitpunkt und vor einem Monat in einem Diagramm angezeigt. In der zweiten Anfrage wird die Differenz gebildet und visualisiert. Anhand dessen lassen sich Visualisierungen erzeugen, die eine schnelle und übersichtliche manuelle Auswertung großer Daten ermöglichen.

Erweiterung der Konfiguration

Durch den Umfang des ELK-Stacks und dessen Plug-ins sind zahlreiche Erweiterungen für die Zukunft denkbar und vor dem Produktiv-Einsatz auch erforderlich. Zum einen war es bereits notwendig, eine portable Entwicklungsumgebung zu schaffen, da der Aufwand zum Projekteinstieg und zur korrekten Konfiguration stetig steigt. Dafür wurde auf eine Kombination aus Vagrant und Puppet gesetzt, um die Entwicklungsumgebung auf verschiedenen Systemen ressourcenschonend aufzusetzen und zu verwenden.

Bevor das System für den Produktiv-Einsatz bereit ist, sind noch zahlreiche Konfigurationsänderungen nötig. Zum einen muss die ElasticSearch-Konfiguration aufgrund der Datenmenge angepasst werden, sodass diese auf mehrere Shards und Replikas verteilt sind. Zudem müssen alte Daten aus dem System als Aggregationen gespeichert und ihre detaillierten Informationen gelöscht werden, um das System auf Dauer stabil und performant halten zu können.

Ein anderer Aspekt ist die Sicherheit. Momentan sind alle Verbindungen und Daten frei zugänglich. Hier ist ein Ansatz der Einsatz von Shield, um die Kommunikation zu verschlüsseln und die Daten mit einem Passwort zu schützen.

Fazit

Der ELK-Stack bietet eine gute Grundlage zur Erfassung, Verarbeitung und Darstellung von Logdaten. Durch seine Architektur ist er skalierbar und auf allen verbreiteten Systemen ausführbar. In seiner reinen Form ist es möglich, einfache Überwachungsaufgaben und Spitzen oder Anomalien zu erkennen. Mithilfe von Timelion wird die Analyse durch den Vergleich von Daten erleichtert. Dies bietet beispielsweise die manuelle Auswertung großer Datenmengen. Watcher ergänzt die manuelle Analyse um die Überwachung, basierend auf Regeln in Echtzeit.

Um diese Werkzeuge effizient einsetzen zu können, sind jedoch ein tiefgreifendes Verständnis der Software und ein strukturiertes und durchdachtes Logkonzept nötig. Für die Überwachung und Korrelation mehrerer Applikationen ist es ratsam weitere Werkzeuge wie Drools Fusion *[8]*, ein auf Java basierendes Eclipse-Plug-in, das Log-Ereignisse korreliert und zu einem Event zusammenfasst, zu benutzen.

Weitere Informationen

[1] *https://www.elastic.co/guide/en/logstash/current/introduction.html*
[2] *https://www.elastic.co/guide/en/logstash/current/filter-plugins.html*
[3] *https://www.elastic.co/de/products/elasticsearch*
[4] *https://www.elastic.co/guide/en/kibana/current/introduction.html*
[5] *https://www.elastic.co/guide/en/watcher/current/introduction.html*
[6] *https://www.elastic.co/de/blog/timelion-timeline*
[7] *https://www.elastic.co/guide/en/logstash/current/plugins-filters-grok.html*
[8] *http://www.drools.org*

Tobias Hülsken
tobias.huelsken@student.kit.edu

Roland H. Steinegger
roland.steinegger@kit.edu

Florian Röser
florian.roeser@ic-consult.de

Dr. Nadina Hintz
nadina.hintz@ic-consult.de

Dr. Sebastian Abeck
sebastian.abeck@kit.edu

Tobias Hülsken ist Hauptautor des Artikels. Er ist Student mit Schwerpunkten der IT-Sicherheit und Rechnerstrukturen am KIT. Im Rahmen seiner Tätigkeit ist er für die Sicherheit und Instandhaltung der Industrie-4.0-Software-Landschaft zuständig.

Roland H. Steinegger forscht seit vier Jahren im Bereich des Identity und Access Management. Schwerpunkt seiner Arbeit ist der Einsatz von Sicherheitsmustern beim domänenorientierten Entwurf von (Micro-) Serviceorientierten Web-Anwendungen.

Florian Röser ist als Consultant und Software-Engineer bei iC Consult für einen deutschen Automobilhersteller im Bereich Identity und Access Management tätig.

Dr. Nadina Hintz ist als Key-Account Manager für einen Kunden der Automobilbranche bei der IC Consult, einem systemunabhängigen Integrator im Bereich Identity und Access Management tätig. Promoviert hat sie im Bereich IT-Security an der Universität Erlangen-Nürnberg.

Dr. Sebastian Abeck leitet am Karlsruher Institut für Technologie die Forschungsgruppe Cooperation & Management, mit der er in den Bereichen der serviceorientierten und mobilen Web-Anwendungen, dem Identity and Access Management und dem Internet of Things Forschungs- und Projektarbeiten durchführt.