Deep Learning und Graphenanalyse im Einsatz gegen Hacker

Hans Viehmann
ORACLE Deutschland B.V. & Co. KG
NL Hamburg

Schlüsselworte
Big Data, Graph-Datenbank, Graphenanalyse, Machine Learning, R, Network Intrusion Detection

Zusammenfassung
Modelliert man IT-Netzwerke als abstrakten Graphen, in dem Netzeinbauten wie Router, Server, usw. als Knoten und die Verbindungen zwischen den Komponenten als Kanten abgebildet werden, lassen sich bereits aus der Topologie des Graphen interessante Erkenntnisse ableiten. Erfasst man darüber hinaus auch den gesamten Netzwerkverkehr zwischen den Knoten samt Protokoll, Port, Status, Antwortzeitverhalten und weiterer Parameter, kann dieses Datenmaterial sehr gut als Grundlage für das Aufspüren von Cyberattacken verwendet werden.
In dem Beitrag wird aufgezeigt, wie die Kombination aus Graphenanalyse und Deep Learning bei der Erkennung von Auffälligkeiten hilft. Es wird die generelle Vorgehensweise auf Basis von Oracle Big Data Spatial and Graph in Kombination mit Skymind DL4J als Deep-Learning-Plattform dargestellt. Methodisch kommt neben Algorithmen aus der Graphentheorie vor allem Supervised Learning in einem neuronalen Netzwerk zum Einsatz. Abschließend werden die Ergebnisse noch visualisiert, um Ausgangspunkte und Ziele von Attacken im Zusammenhang sehen zu können und dadurch besonders gefährdete Komponenten zu erkennen.

Einleitung
Graph-Datenbanken (auch: graphenorientierte Datenbanken) unterscheiden sich von relationalen Technologien vor allem darin, dass sie Entitäten und deren Beziehungen nicht in Tabellen, sondern in Form eines Knoten-Kanten-Modells speichern, wie es schematisch in Abb. 1 dargestellt ist. Sie werden immer häufiger benutzt, um beispielsweise stark vernetzte Datenbestände zu verwalten und auszuwerten. Hierbei spielen insbesondere so genannte Property Graph-Datenbanken eine Rolle, bei denen sowohl Knoten, als auch Kanten mit Eigenschaften (properties) verknüpft sind und für die in der Literatur zahlreiche aus der Graphentheorie bekannte Algorithmen beschrieben sind. Dieser Ansatz unterscheidet sie etwa von Triple Stores (RDF Graph-Datenbanken), wie sie im Umfeld des Semantic Web eingesetzt werden.
Property Graph Technologien werden gern zur Modellierung und Analyse sozialer Netzwerke eingesetzt, werden aber auch zur Auswertung von Kunden-Lieferanten-Beziehungen, oder zur Analyse physikalischer Netze verwendet. Insbesondere für die Überwachung von IP Netzwerken, wie sie im vorliegenden Beitrag im Mittelpunkt stehen, können sie sehr interessant sein. Knowledge Graphs, wie sie Apple SIRI oder der Google Suche zugrunde liegen, basieren im übrigen ebenfalls darauf.
Im Gegensatz zu relationalen Technologien, die eine sorgfältige Modellierung erfordern, sind Property Graph Datenbanken sehr flexibel. Sie erfordern kein vordefiniertes Schema, sondern können mit einer einfachen Skizze vom Flipchart beginnen und sich im Verlauf des Projekts entwickeln. Neue Entitäten, Beziehungen oder Eigenschaften können fortlaufend hinzugefügt werden, ohne die bestehende Funktionalität zu beeinträchtigen. Da Entitäten keinen Platzhalter für alle möglicherweise auftretenden Eigenschaften oder Beziehungen benötigen, eignet sich dieses Schema der Speicherung insbesondere für „sparse data“, also Einsatzfälle, bei denen die Tabellen in relationalen Datenbanken zahlreiche NULL Werte enthalten würden.

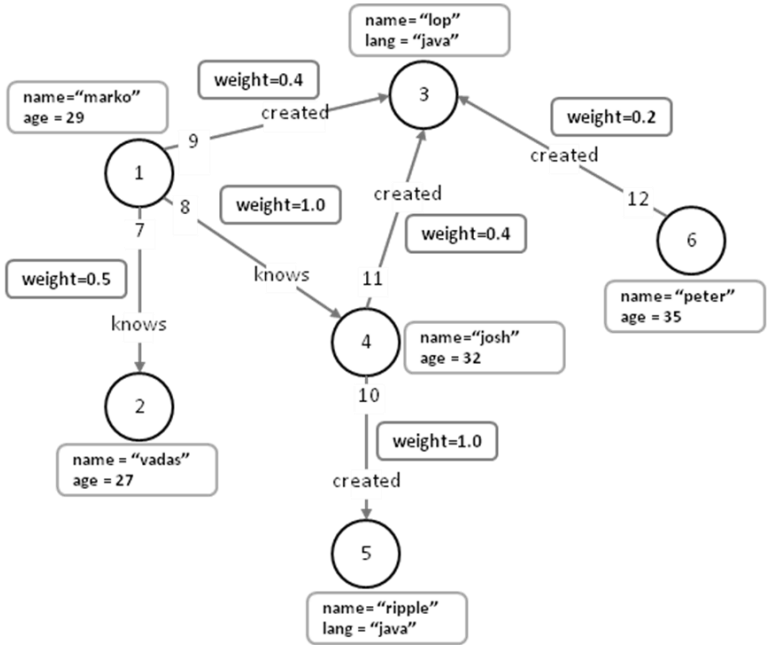

Abb. 1: Allgemeines Modell eines Property Graph (nach: github.com/tinkerpop/blueprints/wiki/Property-Graph-Model)

Der wesentliche Mehrwert von Graph-Datenbanken besteht darin, dass sie Analysen ermöglichen, die mit konventionellen Datenbanken gar nicht oder nur mit hohem rechnerischen Aufwand umsetzbar wären. Diese teilen sich grundsätzlich in zwei Kategorien – entweder handelt es sich um rechnerische Verfahren, die je Knoten bzw. Kante Indikatoren berechnen, die aus (meist wiederholter) Iteration über den gesamten betrachteten Graphen abgeleitet werden, oder es handelt sich um Pattern Matching Verfahren, bei denen ein vorgegebenes Muster genutzt wird, um alle gleichartigen Teilgraphen im Datenbestand zu identifizieren.

Erkennung von Ähnlichkeiten und Anomalien
Graph-Datenbanken lassen sich sehr elegant zur Auswertung von Beziehungsmustern nutzen, weil der Bezug zwischen zwei Objekten explizit gespeichert wird. Diese Eigenschaft kann man sich beispielsweise in Recommendation Engines zunutze machen. Modelliert man Kunden und Produkte als Knoten in einem Graphen und erzeugt jeweils zwischen Kunde und Produkt eine Kante, wenn der Kunde das entsprechende Produkt gekauft hat, lässt sich mittels bekannter Graphenalgorithmen (Collaborative Filtering, Personalized Pagerank) herausfiltern, welche Kunden einander ähnlich sind und darüber bestimmen, welche Produkte jeweils zusätzlich zu einem gegebenen Kunden passen könnten. Im Gegenzug lassen sich natürlich auch diejenigen Knoten in dem Graphen bestimmen, die auffallend „anders“ sind.

Einsatzfall Ähnlichkeitsanalyse für Recommendation Engines
Recommendation Engines sind im Umfeld von eCommerce sehr gängig und etwa bei Amazon seit Jahren im Einsatz. Sie basieren darauf, dass zu einem Nutzer mit gegebenen Kundenverhalten eine Gruppe ähnlicher Kunden aus dem Bestand gefunden wird und auf Basis der Präferenzen in der Gruppe dem Nutzer Produkte vorgeschlagen werden. Technisch lässt sich das über verschiedene Ansätze erreichen; im folgenden wird ein Matrix-basierter (Collaborative Filtering) und ein Ranking-basierter (Personalized Pagerank) Ansatz kurz dargestellt.

1. Recommendation Engines auf Basis Collaborative Filtering
Trägt man für eine eCommerce Plattform alle Produkte und alle Kunden in einer Matrix gegeneinander auf und markiert jeweils diejenigen Kombinationen von Produkt und Kunde, für die eine Interaktion vorhanden war (z.B. Produkt gekauft, Produkt angesehen, Produkt negativ bewertet, ...), entsteht eine sehr große Matrix, die ganz überwiegend aus leeren Zellen besteht. Diese Art von Matrix lässt sich nicht ohne weiteres in einer relationalen Datenbank abbilden, kann aber sehr kompakt als Graph modelliert und mittels Algorithmen aus der Graphentheorie ausgewertet werden. Der zugehörige Lösungsalgorithmus wird als Matrix Factorization bezeichnet. Er dient dazu, eine Art Fingerabdruck („taste signature“) zu berechnen, der die Präferenzen des Kunden widerspiegelt, bzw. komlementär dazu eine „taste signature“ des Produkts zu ermitteln, der die Kunden, die das Produkt bevorzugen repräsentiert.
Um diese Funktionalität zu nutzen, muss nur ein Graph aufgebaut werden, der alle Interaktionen von Kunde und Produkt enthält. Der Matrix Factorization Algorithmus kann dann einfach aufgerufen werden, um anschließend zu einem gegebenem Kunden genau das Produkt vorgeschlagen zu bekommen, das er/sie gemäß dem Algorithmus mit der höchsten Wahrscheinlichkeit kauft.

2. Recommendation Engines auf Basis Personalized Pagerank
Hat man alle Interaktionen von Kunde und Produkt in einem Graph modelliert, lassen sich auch andere Vorhersagemodelle verwenden, um Produkte vorzuschlagen. Insbesondere der Personalized Pagerank Algorithmus wäre hier zu nennen. Dieser Algorithmus startet bei einem gegebenen Knoten (Kunden) und folgt einem zufälligen Pfad von Kunde zu Produkt zu Kunde, usw., bis zu einem willkürlichen Abbruchzeitpunkt. Anschließend springt er zum Ausgangspunkt zurück und legt – wieder und wieder – einen zufälligen Weg entlang des Graphen zurück. Wählt man die Anzahl Iterationen groß genug und zählt die Anzahl der Male, die ein Produkt bzw. Kunde im Laufe der Iterationen auf dem Pfad gelegen hat, erhält man ein Maß dafür, wie „ähnlich“ der jeweilige Kunde bzw. wie „naheliegend“ das jeweilige Produkt ist. Diejenigen Produkte, die von Kunden gekauft wurden, die mit dem Startknoten über viele Produkt-Kunden Interaktionen verbunden sind, werden wesentlich häufiger auf dem zufälligen Pfad liegen, als ein Produkt, das nur über eine große Zahl von Zwischenschritten erreicht werden kann. In relationaler Technologie wäre dieser Algorithmus aufgrund der zahlreichen Joins jeweils zwischen Produkt und Kunde in Kombination mit den vielfachen Iterationen sehr langsam.

Einsatzfall Anomaly Detection
Der Personalized Pagerank Algorithmus ist nicht nur dafür geeignet, in einem Graph mit Beziehungen zwischen Kunden und Produkten die zu einem Knoten besonders „ähnlichen“ Knoten zu ermitteln. Er kann ebenso dazu genutzt werden, diejenigen Knoten in einem Graphen zu identifizieren, die auffallend „anders“ sind, als die übrigen Knoten. Sucht man zu einer Vergleichsgruppe diejenigen Knoten mit einem geringen Personalized Pagerank Wert, erhält man sozusagen die „Außenseiter“. In einem Graphen in dem beispielsweise Ärzte und deren Behandlungen in Beziehung gesetzt sind, kann man so diejenigen Ärzte identifizieren, die Behandlungen durchführen, die in ihrer Vergleichsgruppe besonders unüblich sind.

Oracle Property Graph Technologien – Grundlagen
Oracle liefert Property Graph Technologien samt Datenhaltung und in-memory Analytics Engine an, die für alle diese Einsatzfälle ausgelegt ist. Neben der Datenhaltung, die in Oracle NoSQL, Apache HBase oder Oracle 12cR2 (samt Spatial and Graph Option) erfolgt und die mit den in diesem Umfeld gängigen Programmierschnittstellen ausgestattet ist, besteht der wesentliche Wert vor allem in den knapp 50 vorgefertigten Algorithmen aus den Bereichen Ranking, Community Detection, Path Finding, Recommendation, usw., die bereits mit dem Produkt ausgeliefert werden. Die Ergebnisse von Analysen im Hauptspeicher lassen sich in der Datenhaltung persistieren und und stehen im Falle der Speicherung in HBase oder NoSQL über Big Data SQL oder External Tables (Big Data Connectors) auch für den Zugriff aus der Oracle Datenbank zur Verfügung. Im Falle der Speicherung in Oracle

12.2 kann mittels SQL auch direkt auf die Datenstrukturen zugegriffen werden. Eine schematische Architektur ist in Abb. 2 dargestellt.

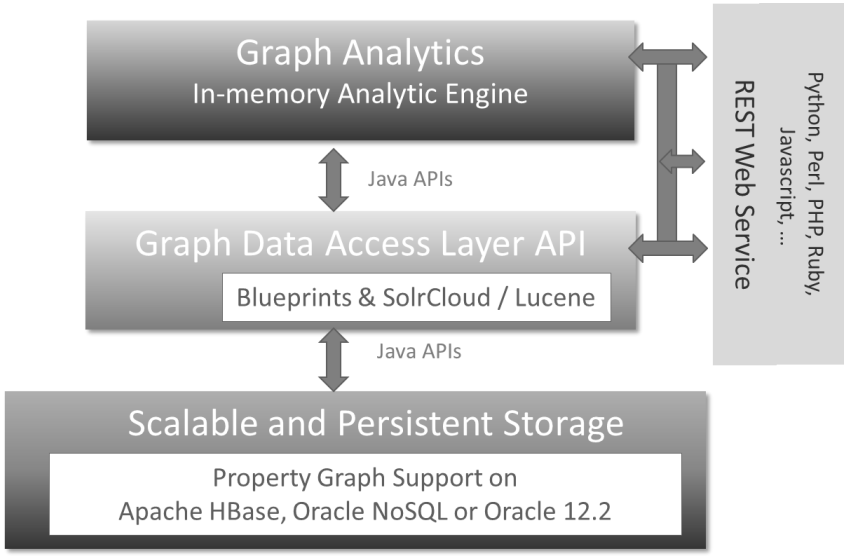

Abb. 2: Schematische Architektur von in-memory analytics engine und Persistenzschicht

Aufbau eines Graphen
Die Struktur eines Property Graph ist relativ simpel – zu jedem Knoten gibt es einen eindeutigen Schlüssel, eine Menge von Kanten, gerichtet oder ungerichet, sowie eine Menge von Eigenschaften als Key-value Paare. Ebenso gibt es zu jeder Kante einen eindeutigen Schlüssel, einen Anfangs- und einen Endknoten, einen Label, der die Art der Beziehung enthält, sowie wiederum eine Menge von Eigenschaften als Key-value Paare (vgl. Abb. 1).
Liegt bereits ein relationales Datenmodell vor, gibt es verschiedene Möglichkeiten, diese als Property Graph zu modellieren. Ein erster Ansatz ist üblicherweise, alle Zeilen einer Tabelle in Knoten zu übersetzen, die Spaltenwerte als Eigenschaften zu modellieren und Primärschlüssel-Fremdschlüssel Beziehungen als Kanten abzubilden. Im Falle von n:m Beziehungen fallen die Join Tabellen weg, die Beziehungen werden direkt an die Knoten angefügt und alle Spaltenwerte der Join Tabelle können als Eigenschaften der jeweiligen Kante modelliert werden.
Das eigentliche Anlegen eines Graphen kann entweder programmatisch über elementare Operationen wie AddVertex, AddProperty, ... erfolgen, indem schrittweise Elemente und Eigenschaften hinzugefügt werden, oder über graphische Tools stattfinden, die ihrerseits diese Methoden aufrufen. Liegt ein Datenbestand schon als Graph vor, gibt es eine Reihe von Datenaustauschformaten (GraphSON, GraphML, GML), die produktseitig unterstützt werden. Auch zum Laden aus Tabellen oder CSV Dateien ist ein entsprechendes API vorhanden, es können aber ebensogut Werkzeuge wie der Oracle Data Integrator verwendet werden.

Erzeugung eines Graphen aus Netzwerkverkehr
Für das vorliegende Projekt wurden mittels tcpdump erfasste Datenpakete aufbereitet und zu Verbindungsdaten aggregiert. Je Verbindung wurden neben IP Adresse und Port von Quelle und Ziel auch eine ganze Reihe weiterer Attribute erfasst. Neben Protokoll, Statusinformation, Dauer, Anzahl Datenpakete, Anfangs- und Endzeit, usw., wurden insgesamt 49 Eigenschaften festgehalten. Jede IP Adresse wurde als Knoten im Graphen modelliert und jede Verbindung durch eine Kante dargestellt, die die zugehörigen Attribute als Eigenschaften zugeordnet bekam. Für die eigentliche Umwandlung der Daten, die in einer CSV Datei enthalten waren, konnten zwei Utilities eingesetzt werden, die im Produktumfang enthalten sind - `OraclePropertyGraphUtilsBase.convertCSV2OPV` erzeugt die

Knoten aus den IP Adressen, `OraclePropertyGraphUtilsBase.convertCSV2OPE` die Kanten samt Attributen.

Kombination von Machine Learning und Graphenanalyse
Verfahren aus dem Machine Learning lassen sich sehr gut mit Anwendungen aus der Graphenanalyse ergänzen. So können beispielsweise Metriken wie der Pagerank-Wert eines Knotens im Graph als Eingabewert zu Machine Learning Algorithmen verwendet werden, um als weiteres mögliches Signal oder Unterscheidungskriterium zu dienen. Im Gegenzug können die Ergebnisse aus einer Analyse mit Machine Learning Verfahren, etwa ein Scoring oder eine Klassifizierung, als Input für Algorithmen auf dem zugehörigen Graph genutzt werden. Auch die Visualisierung des Graphen kann eine wertvolle Unterstützung bieten.
Als Plattform für Machine Learning erfreut sich R großer Beliebtheit. Insbesondere in Kombination mit der Oracle Datenbank über Oracle R Enterprise lassen sich verschiedene Machine Learning Verfahren performant und skalierbar umsetzen. Um die Property Graph Engine mit R zu integrieren, wurde ein eigenes Package für R, OAAgraph, entwickelt, das die Steuerung der in-memory analytics engine und damit der entsprechenden Algorithmen aus R heraus ermöglicht. So lassen sich vom R Client aus Daten aus Datenbanktabellen oder auch aus Spark laden und auf dem erzeugten Graph nach Bedarf Algorithmen ausführen. Die Ergebnisse können entweder über einen Cursor in R data frames übertragen werden, oder zurück in die Persistenzschicht gespeichert werden, um dort mittels Spark oder mit ORE oder mit anderen Datenbankmitteln weiterverarbeitet zu werden.
Da zum Zeitpunkt des Projekts diese Integration noch nicht zur Verfügung stand, wurde hier noch mit der Deep Learning Plattform DL4J gearbeitet, die von Skymind zur bereitgestellt wurde. DL4J ist eine open source Deep Learning Engine, die auf Java Runtimes lauffähig ist.

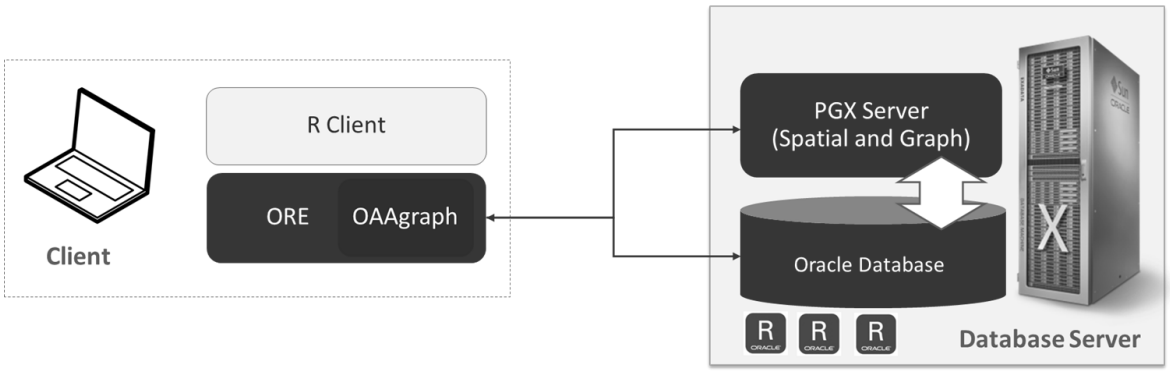

Abb. 3: Schematische Darstellung eines R Clients mit Oracle Datenbank und in-memory analytics engine

Network Intrusion Detection mittels Deep Learning
Ziel des Projekts war die Erkennung von Cyberangriffen im Unternehmensnetz mittels Deep Learning Verfahren. Deep Learning basiert auf neuronalen Netzen, die mittels eines Trainingsdatenbestandes lernen, Muster zu erkennen. Die verschiedenen Varianten von neuronalen Netzen und deren Funktionsweise, sowie Deep Learning Verfahren im Allgemeinen zu erläutern, würde den Rahmen dieses Beitrags sprengen, aber auf den Trainingsdatenbestand, der für den Erfolg eines Projekts von entscheidender bedeutung ist, soll hier kurz eingegangen werden. Bei dem Trainings-Datensatz handelte es sich um den oben bereits erwähnten mitgeschnittenen Netzwerkverkehr samt der 49 Attribute je Verbindung. Er wurde vom Cyber Range Lab des Australian Centre for Cyber Security künstlich erzeugt und enthält neben zahlreichen echten, normalen Netzwerkaktivitäten einige künstlich erzeugte Cyberangriffe nach aktuellen Mustern. Insbesondere handelte es ich um neun verschiedene Arten von Angriffen, die in einem Trainingsdatenbestand von 175000 Verbindungen als solche

markiert waren. Zusätzlich lag ein Testdatenbestand nach gleichem Muster vor, in dem die Angriffe gesucht und erkannt werden sollten.
Für das Projekt wurden zwei Varianten von neuronalen Netzen eingesetzt. Einerseits wurde ein Multi-layer Perceptron (MLP) Netz genutzt, andererseits wurde eine Long Short-Term Memory (LSTM) Implementierung verwendet, die von den Ergebnissen ähnlich war und auf die hier nicht näher eingegangen werden soll. Da in den Trainings- und Testdaten einige Attribute als Kategorien vorlagen, wie beispielsweise im Falle des Protokolls (DNS, HTTP, SMTP, ...), während das MLP Net nur mit skalaren Werten umgehen kann, wurden diese Attribute mittels Categorical-to-One-Hot Transformationen in entsprechende Bitmuster umgewandelt. Nach 800 Trainingsläufen erzielte das neuronale Netz ausgesprochen gute Ergebnisse. In einem Testlauf wurden beispielsweise 46 non-intrusion events als solche erkannt, alle sechs intrusion events wurde korrekt festgestellt und nur ein non-intrusion event wurde fälschlich als Cyberangriff bewertet (false positive).
Zur weiteren Analyse wurden die Ergebnisse in den Graph übertragen, indem diejenigen Kanten, die einen Angriff darstellen, um das entsprechende Attribut ergänzt wurden. Damit konnten anschließend mittels Pagerank Algorithmus die wichtigsten betroffenen IP Adressen bestimmt werden. Außerdem wurde der Graph zur interaktiven Analyse visualisiert. Zu diesem Zweck wurde Cytoscape verwendet, ein open source Visualisierungstool, für das seitens Oracle ein Property Graph Plugin entwickelt wurde. Cytoscape ist im Lieferumfang von Oracle Spatial and Graph bzw. Oracle Big Data Spatial and Graph enthalten und kann nicht nur für Abfragen genutzt werden, sondern auch direkt Algorithmen in der in-memory analytics engine ausführen.

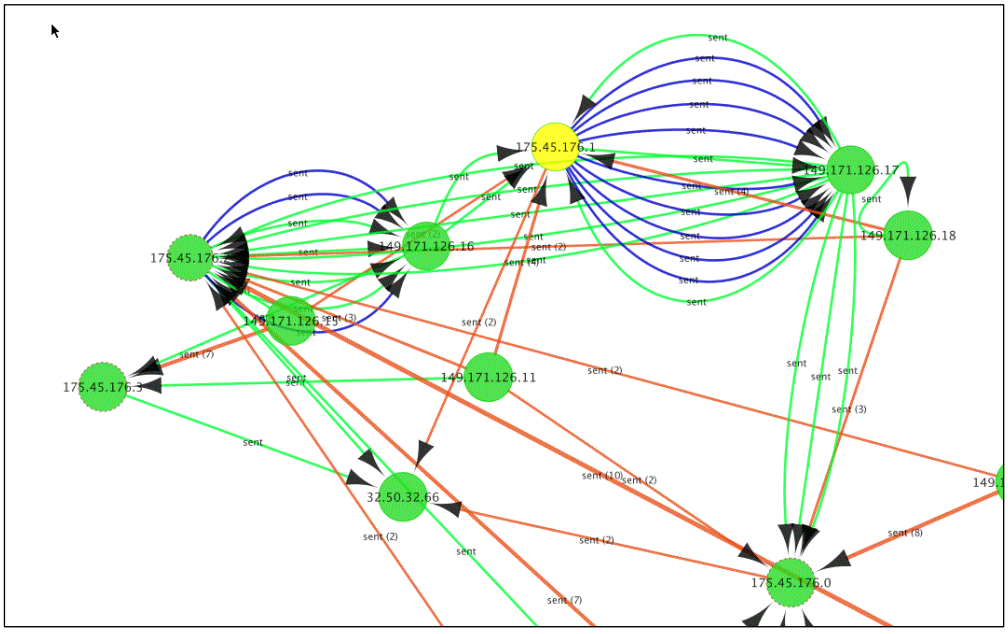

Abb. 5: Screenshots des IP Netzwerks samt Cyberangriffen (blau) in Cytoscape

Zusammenfassung

Graph-Datenbanken bieten zahlreiche Analysemöglichkeiten, die mit relationalen Technologien nicht oder nur nur sehr aufwändig umgesetzt werden können. Überall dort, wo Beziehungen oder Abhängigkeiten ausgewertet werden müssen, vor allem, wenn sie die Topologie des gesamten Beziehungsnetzwerks betreffen oder Beziehungen über mehrere Stufen aufgedeckt werden sollen, sind sie ausgesprochen hilfreich. Wie in dem vorliegenden Beitrag gezeigt, können sie gut mit Machine Learning Verfahren kombiniert werden. Insbesondere die Integration mit R ist einfach, seitdem das OAAgraph Package zur Verfügung steht.

Die zugehörigen Oracle Technologien, sei es auf der Big Data Plattform mit HBase oder Oracle NoSQL als Persistenzschicht oder in Form von Oracle Spatial and Graph als Option zur Datenbank, bieten umfassende Analysemöglichkeiten, die hochperformant auf dem Graph ausgeführt werden. Sie stehen sowohl als Teil der Database Cloud Services, als auch im Rahmen der Big Data Cloud Services zur Verfügung. Auf der Openworld 2017 wurde von Thomas Kurian bereits ein eigener Autonomous Graph Analytics Cloud Service angekündigt, der die Handhabung nochmal deutlich vereinfachen soll und der im Laufe des Kalenderjahres 2018 live gehen soll.

Weitere Informationen
Nähere Informationen zu Oracle Big Data Spatial and Graph sind auf dem Oracle Technology Network zu finden (www.oracle.com/technetwork/database/database-technologies/bigdata-spatialandgraph); daneben gibt es unter blogs.oracle.com/bigdataspatialgraph/ einen Blog mit hilfreichen Beiträgen, der vom Development Team gepflegt wird. Zum Einstieg eignet sich die BigDataLite Virtual Machine (siehe www.oracle.com/technetwork/community/developer-vm/index.html), in der die komplette Umgebung konfiguriert und eingerichtet ist. Ein Hands-on lab (unter /opt/oracle/oracle-spatial-graph) wird mit dieser virtuellen Umgebung mit ausgeliefert und enthält auch einen kleinen Demo-Datenbestand.

Kontaktadresse:
Hans Viehmann
ORACLE Deutschland B.V. & Co. KG
Kühnehöfe 5
D-22761 Hamburg

Telefon:	+49 (0) 40-89091-173
Fax:	+49 (0) 40-89091-250
E-Mail	hans.viehmann@oracle.com
Internet:	www.oracle.com/goto/bigdata