Using Deep Learning and Graph Analysis against Cyberattacks

ITOUG TechDay 2018

Hans Viehmann
Product Manager EMEA
ORACLE Corporation
February 1, 2018

@SpatialHannes
Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle’s products remains at the sole discretion of Oracle.
Agenda

1. Introduction to graph analysis
2. Using Oracle’s graph technologies to work with graphs
3. Combining graph analysis and machine learning
4. Using machine learning for network intrusion detection
5. Wrap-up
Following, no follow back

Follower, no follow back

Follow each other

https://twitter.jeffprod.com
Graph Data Model

• What is a graph?
  – Data model representing entities as vertices and relationships as edges
  – Optionally including attributes
  – Also known as „linked data“

• What are typical graphs?
  – Social Networks
    • LinkedIn, facebook, Google+, ...
  – IP Networks, physical networks, ...
  – Knowledge Graphs
    • Apple SIRI, Google Knowledge Graph, ...
Graph Data Model

• Why are graphs popular?
  – Easy data modeling
    • “whiteboard friendly”
  – Flexible data model
    • No predefined schema, easily extensible
    • Particularly useful for sparse data
  – Insight from graphical representation
    • Intuitive visualization
  – **Enabling new kinds of analysis**
    • Overcoming some limitations in relational technology
    • Basis for Machine Learning (Neural Networks)
Categories of Graph Analysis

Computational Graph Analytics

- Compute values on vertices and edges
- Traversing graph or iterating over graph (usually repeatedly)
- Procedural logic
- Examples:
  - Shortest Path, PageRank, Weakly Connected Components, Centrality, ...

Graph Pattern Matching

- Based on description of pattern
- Find all matching sub-graphs

Categories of Graph Analysis

Person{100}
name = 'Amber'
age = 25

Person{200}
name = 'Paul'
age = 30

Person{300}
name = 'Heather'
age = 27

Company{777}
name = 'Oracle'
location = 'Redwood City'

:worksAt{1831}
startDate = '09/01/2015'

:friendOf{1173}

:friendOf{2513}
since = '08/01/2014'

:known{2200}

:Person{300}
name = 'Heather'
age = 27
Detecting similarities – Recommendation Engines

• Identifying users with similar behaviour or buying pattern
• Viewing customer-item relations as large (sparse) matrix
  – Customers as one dimension, items as other
• Matrix cells filled with rating/rank
  – Represent as graph, not as matrix
• Collaborative Filtering [1] algorithm solves taste signature of customers, items
  – Resulting vectors are like DNA
• Inner product of vectors reflects quality of match

Detecting Outliers – Graph Analysis and Anomaly Detection

• Requirement:
  – Identify entities from a large dataset that look different than others, especially in their relationships

• Approaches:
  – Define an anomaly pattern, find all instances of the pattern in the graph
  – Given nodes in the same category, find nodes that stand out (eg. low Pagerank value)
Use case: Fraud Detection in Healthcare

- Example for potential fraud detection
  - Public domain dataset
  - Medical providers and their operations

- Question
  - Are there any medical providers that are suspicious
  ➔ medical providers that perform different operations than their fellows
    (e.g. eye doctors doing plastic surgery)

- Approach
  - Create graph between doctors and operations
  - Apply personalized pagerank (a.k.a equivalent to random walking)
  - Identify doctors that are far from their fellows
Agenda

1. Introduction to graph analysis
2. Using Oracle’s graph technologies to work with graphs
3. Combining graph analysis and machine learning
4. Using machine learning for network intrusion detection
5. Wrap-up
Introducing: Oracle Big Data Spatial and Graph

Spatial Analysis:
• Location Data Enrichment
• Proximity and containment analysis, Clustering
• Spatial data preparation (Vector, Raster)
• Interactive visualization

Property Graph Analysis:
• Graph Database
• In-memory Analysis Engine
• Scalable Network Analysis Algorithms
• Developer APIs
In-memory Analytics Engine – Product Packaging

Oracle Big Data Spatial and Graph

• Available for Big Data platform
  – Hadoop, HBase, Oracle NoSQL
• Supported both on BDA and commodity hardware
  – CDH and Hortonworks
• Database connectivity through Big Data Connectors or Big Data SQL
• Included in Big Data Cloud Service

Oracle Spatial and Graph (DB option)

• Available with Oracle 12.2 (EE)
• Using tables for graph persistence
• In-database graph analytics
  – Sparsification, shortest path, page rank, triangle counting, WCC, sub graph generation...
• SQL queries possible
  – Integration with Spatial, Text, Label Security, RDF Views, etc.
Oracle Big Data Graph Architecture

Graph Analytics
In-memory Analytic Engine

Graph Data Access Layer API
Blueprints & SolrCloud / Lucene

Scalable and Persistent Storage
Property Graph Support on
Apache HBase, Oracle NoSQL or Oracle 12.2

Java APIs

Python, Perl, PHP, Ruby,
Javascript, ...

REST Web Service
Creating a Graph

• From a relational model
  – Rows in tables usually become vertices
  – Columns become properties on vertices
  – Relationships become edges
  – Join tables in n:m relations are transformed into relationships, columns become properties on edges

• Interactively through API or graphical tool
  – Adding vertices, edges, properties to a given graph

• From graph exchange formats
  – GraphML, GraphSON, GML (Graph Modeling Language)
Creating a Graph from Network Traffic

• Capture network traffic (source/target IP address and port, protocol, state, duration, ...)
• Model each IP address as vertex
• Model each record (from source IP to destination IP) as an edge
• Attributes can become properties on the edge
• Utilities available to convert CSV to graph
  – OraclePropertyGraphUtilsBase.convertCSV2OPV
  – OraclePropertyGraphUtilsBase.convertCSV2OPE

```
59.166.0.1,62377,149.171.126.4,53,udp,CON,0,001044,130,162,31,29,0,0,dns,498084,2813,620689,625,2,
192.168.241.243,259,192.168.241.243,49320,icmp,URH,0,1780,0,64,0,0,0,-,196.4095,0,5,0,0,0,0,0,0,356,
192.168.241.243,49320,192.168.241.243,0xc0a8,icmp,URH,0,1780,0,64,0,0,0,-,196.4095,0,5,0,0,0,0,0,0,3
59.166.0.6,38993,149.171.126.0,53,udp,CON,0,00106,132,164,31,29,0,0,dns,498113,1875,61867,875,2,2
59.166.0.9,59720,149.171.126.8,53,udp,CON,0,00107,132,164,31,29,0,0,dns,493457,9375,613084,125,2,2
59.166.0.4,21489,149.171.126.7,53,udp,CON,0,001144,130,162,31,29,0,0,dns,454545,4688,566433,5625,2
```
Agenda

1. Introduction to graph analysis
2. Using Oracle’s graph technologies to work with graphs
3. Combining graph analysis and machine learning
4. Using machine learning for network intrusion detection
5. Wrap-up
Combining Graph Analytics and Machine Learning

**Graph Analytics**
- Compute graph metric(s)
- Explore graph or compute new metrics using ML result

**Machine Learning**
- Build predictive model using graph metric
- Build model(s) and score or classify data

Add to structured data
Add to graph
Using Oracle R Enterprise for Machine Learning

Use Oracle Database as a high performance compute environment

• Transparency layer
  – Leverage proxy objects (ore.frames) - data remains in the database
  – Overload R functions that translate functionality to SQL
  – Use standard R syntax to manipulate database data

• Parallel, distributed ML algorithms
  – Scalability and performance
  – Exposes in-database machine learning algorithms from ODM
  – Additional R-based algorithms executing and database server

• Embedded R execution
  – Store and invoke R scripts in Oracle Database
  – Data-parallel, task-parallel, and non-parallel execution
  – Invoke R scripts at Oracle Database server from R or SQL
  – Use open source CRAN packages
One option: OAAgraph integration with R

- OAAgraph integrates in-memory engine into ORE and ORAAH
- Adds powerful graph analytics and querying capabilities to existing analytical and machine learning portfolio of ORE and ORAAH
- Built-in algorithms of PGX available as R functions
- PGQL pattern matching
- Concept of “cursor” allows browsing of in-memory analytical results using R data structures (R data frame), allows further client-side processing in R
- Exporting data back to Database / Spark allows persistence of results and further processing using existing ORE and ORAAH analytical functions
OOAgraph Architecture

- **OOAgraph** gives remote control of PGX server
- PGX loads graph from database (ore.frames)

**Client**

- R Client
- ORE
- OOAgraph

**Database Server**

- PGX Server (Spatial and Graph)
- Oracle Database

**OOAgraph** is an additional R package that comes with ORE
OAAgraph Architecture

- **OAAgraph** gives remote control of PGX server
- PGX loads graph via SPARK data frames

**Client**

- ORAAH
- OAAgraph

- **OAAgraph** is also available with ORAAH

**Oracle Big Data (Hadoop) Cluster**

- HDFS / Hive / …
- PGX Server (Big Data Spatial & Graph)
- Hadoop & Spark
Execution Overview (ORE)

• Initialization and Connection

# Connect R client to
# Oracle Database using ORE
R> ore.connect(..)
R> oaa.init(..)

# Connect to PGX server
# using OAAgraph
R> oaa.graphConnect(...)

Client

R Client

ORE OAAgraph

PGX Server

Oracle Database

Database Server
Execution Overview (ORE)

• Data Source
  – Graph data is represented as two tables
    • Nodes and Edges
  – Multiple graphs can be stored in database
    • Using user-specified, unique table names

<table>
<thead>
<tr>
<th>Node Table</th>
<th>Edge Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Node ID</td>
<td>From Node</td>
</tr>
<tr>
<td>Prop 1</td>
<td>To Node</td>
</tr>
<tr>
<td>Prop 2</td>
<td>Edge Prop 1</td>
</tr>
<tr>
<td></td>
<td>(relation)</td>
</tr>
<tr>
<td>1238</td>
<td>1299</td>
</tr>
<tr>
<td>John</td>
<td>Likes</td>
</tr>
<tr>
<td>1299</td>
<td>4818</td>
</tr>
<tr>
<td>Paul</td>
<td>FriendOf</td>
</tr>
<tr>
<td>4818</td>
<td>6637</td>
</tr>
<tr>
<td></td>
<td>FriendOf</td>
</tr>
</tbody>
</table>
Execution Overview (ORE)

- Loading Graph

```r
# Load graph into PGX:
# Graph load happens at the server side.
# Returns OAAgraph object, which is a
# proxy (remote handle) for the graph in PGX
R> mygraph <- oaa.graph (NodeTable, EdgeTable, ...)
```
Execution Overview (ORE)

• Running Graph Algorithm

# e.g. compute Pagerank for every node in the graph
# Execution occurs in PGX server side
R> result1<- pagerank (mygraph, ... )
Execution Overview (ORE)

- Iterating remote values with cursor

```r
# e.g. compute Pagerank for every node in the graph
# Execution occurs in PGX server side
R> result1 <- pagerank (mygraph, ... )

# Return value is a “cursor” object
# for the computed result:
# client can get local data frames by oaa.next()
R> df <- oaa.next(result1, 10)
```
Execution Overview (ORE)

• Querying the graph

```
# Query graph using a SQL syntax pattern specification
R> q_result <- oaa.cursor(mygraph, 
  "SELECT n.name, m.name, n.pagerank, m.pagerank 
  WHERE (n WITH pagerank < 0.1) -> (m), 
  n.pagerank > m.pagerank 
  ORDER BY n.pagerank"
)
# Returns a cursor to examine results
R> df <- oaa.next(q_result, 10)
```
Execution Overview (ORE)

- Exporting the result to DB

```
# Export result to DB as Table(s)
R> oaa.create(mygraph, nodeTableName = "node",
               nodeProperties = c("pagerank", ...),
               ...)
```
Execution Overview (ORE)

• Continuing analysis with ORE

```r
# Use ORE Machine Learning on
# the exported table proxy object ore.frames
R> model <- ore.odmKMmeans(formula = ~.,
    data = NODES,
    num.centers = 5,...)
R> scores <- predict(model, NODES, ...)
...
Agenda

1. Introduction to graph analysis
2. Using Oracle’s graph technologies to work with graphs
3. Combining graph analysis and machine learning
4. Using machine learning for network intrusion detection
5. Wrap-up
Use case: Network Intrusion Detection
Using deep learning and graph analysis

• Determining if network activity is legitimate or fraudulent
  – Based on sequence of network activity (as above)
  – Complementary to firewalls, network intrusion prevention system, ...

• Possible approaches
  – „Signature-based“, using labeled dataset of known attacks (supervised learning)
  – Anomaly-based, trying to detect previously unseen attacks

• Most effective systems make use of both
  – Combined with rules engine

• Tested supervised learning in project using DL4J
Supervised learning

Training dataset

• Labeled Network data set from Univ. of South Wales
  – UNSW-NB15 data set specifically created for **Network Intrusion Detection** systems
  – Generated by IXIA PerfectStorm tool in Cyber Range Lab of Australian Centre for Cyber Security
  – Real modern normal activities plus synthetic contemporary attack behaviours
  – Partitioned into training set (175K records) and testing set (82K records)
  – nine types of attacks – Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms


Prototype with Skymind and DeepLearning4J

Graph Database
(BDSG and Oracle Spatial and Graph)

DataVec
“Rosetta Stone” of vectorization

DEEPLEARNING4J
Open-source distributed DL for the JVM

ND4J
Scientific computing for java
(our linear algebra engine)

GPU

Predictions & Classifications

Graph Database
(BDSG and Oracle Spatial and Graph)
Processing Workflow

• Understanding the dataset
  – 49 features in each record – IP addresses, integer, float, timestamp, ...

• Data cleansing
  – Converting Hex to number

• Creating vector as input to DL4J deep learning engine
  – Categorical to One Hot transformation of status strings

• Build Neural Network
  – Train and subsequently test quality using testing set

• Transfer result to graph database
  – Further analysis
### Understand the data

- Features of UNSW-DB15 data set

<table>
<thead>
<tr>
<th>No.</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>src</td>
<td>nominal</td>
<td>Source IP address</td>
</tr>
<tr>
<td>2</td>
<td>sport</td>
<td>integer</td>
<td>Source port number</td>
</tr>
<tr>
<td>3</td>
<td>dst</td>
<td>nominal</td>
<td>Destination IP address</td>
</tr>
<tr>
<td>4</td>
<td>dstport</td>
<td>integer</td>
<td>Destination port number</td>
</tr>
<tr>
<td>5</td>
<td>proto</td>
<td>integer</td>
<td>Transaction protocol</td>
</tr>
<tr>
<td>6</td>
<td>state</td>
<td>nominal</td>
<td>Indicates the state and its dependent protocol, e.g., TCP, UDP</td>
</tr>
<tr>
<td>7</td>
<td>dur</td>
<td>Float</td>
<td>Record total duration</td>
</tr>
<tr>
<td>8</td>
<td>bytcnt</td>
<td>Integer</td>
<td>Source to destination transmission bytes</td>
</tr>
<tr>
<td>9</td>
<td>bytcnt</td>
<td>Integer</td>
<td>Destination transmission bytes</td>
</tr>
<tr>
<td>10</td>
<td>dur</td>
<td>Integer</td>
<td>Destination to source time to live value</td>
</tr>
<tr>
<td>11</td>
<td>dlbytcnt</td>
<td>Integer</td>
<td>Destination to source time to live value</td>
</tr>
<tr>
<td>12</td>
<td>lost</td>
<td>Integer</td>
<td>Source packets retransmitted or dropped</td>
</tr>
<tr>
<td>13</td>
<td>lost</td>
<td>Integer</td>
<td>Destination packets retransmitted or dropped</td>
</tr>
<tr>
<td>14</td>
<td>service</td>
<td>nominal</td>
<td>http, ftp, smtp, ssh, smtp-data, etc. and ( ) if not much used service</td>
</tr>
<tr>
<td>15</td>
<td>slbt</td>
<td>Integer</td>
<td>Source packets bits per second</td>
</tr>
<tr>
<td>16</td>
<td>plbt</td>
<td>Integer</td>
<td>Destination bits per second</td>
</tr>
<tr>
<td>17</td>
<td>splts</td>
<td>Integer</td>
<td>Source to destination packet count</td>
</tr>
<tr>
<td>18</td>
<td>dpkts</td>
<td>Integer</td>
<td>Destination to source packet count</td>
</tr>
<tr>
<td>19</td>
<td>srcwin</td>
<td>Integer</td>
<td>Source TCP window advertisement value</td>
</tr>
<tr>
<td>20</td>
<td>dstwin</td>
<td>Integer</td>
<td>Destination TCP window advertisement value</td>
</tr>
<tr>
<td>21</td>
<td>src級</td>
<td>Integer</td>
<td>Source TCP base sequence number</td>
</tr>
<tr>
<td>22</td>
<td>dst級</td>
<td>Integer</td>
<td>Destination TCP base sequence number</td>
</tr>
<tr>
<td>23</td>
<td>meanpkt</td>
<td>Float</td>
<td>Mean of the T Oswin size transmitted by the src</td>
</tr>
<tr>
<td>24</td>
<td>meanpkt</td>
<td>Float</td>
<td>Mean of the T Oswin size transmitted by the dst</td>
</tr>
<tr>
<td>25</td>
<td>trans_depth</td>
<td>Float</td>
<td>Represents the payload depth into the connection of http request/response transaction</td>
</tr>
<tr>
<td>26</td>
<td>res_body</td>
<td>Integer</td>
<td>Actual uncompressed content size of the data transferred from the server’s http service.</td>
</tr>
<tr>
<td>27</td>
<td>srcjitter</td>
<td>Float</td>
<td>Source jitter (msec)</td>
</tr>
<tr>
<td>28</td>
<td>dstjitter</td>
<td>Float</td>
<td>Destination jitter (msec)</td>
</tr>
<tr>
<td>29</td>
<td>srcftime</td>
<td>Float</td>
<td>Source timestamp record start time</td>
</tr>
<tr>
<td>30</td>
<td>dstftime</td>
<td>Float</td>
<td>Destination timestamp record start time</td>
</tr>
<tr>
<td>31</td>
<td>srcipkt</td>
<td>Float</td>
<td>Source interpacket arrival time (msec)</td>
</tr>
<tr>
<td>32</td>
<td>dstipkt</td>
<td>Float</td>
<td>Destination interpacket arrival time (msec)</td>
</tr>
<tr>
<td>33</td>
<td>src_ipkt</td>
<td>Float</td>
<td>TCP connection setup round-trip time, the sum of ‘syncack’ and ‘ackdata’</td>
</tr>
<tr>
<td>34</td>
<td>dst_ipkt</td>
<td>Float</td>
<td>TCP connection setup time between the SYN and the SYNACK packets.</td>
</tr>
<tr>
<td>35</td>
<td>src_ack</td>
<td>Float</td>
<td>TCP connection setup time between the SYN ACK and the Ack packets.</td>
</tr>
<tr>
<td>36</td>
<td>src_ipkt</td>
<td>Binary</td>
<td>If source (1) and destination (2) IP addresses equal and port numbers (3)(4) equal then, this variable takes value 1 else 0</td>
</tr>
<tr>
<td>37</td>
<td>dst_ipkt</td>
<td>Integer</td>
<td>No for each state (6) according to specific range of values for source/destination time to live (10)(11)</td>
</tr>
<tr>
<td>38</td>
<td>src_flow</td>
<td>Integer</td>
<td>No of flows that has matched such as send and post in http service.</td>
</tr>
<tr>
<td>39</td>
<td>dst_flow</td>
<td>Integer</td>
<td>If the ftp session is accessed by user and password than 1 else 10</td>
</tr>
<tr>
<td>40</td>
<td>src_host</td>
<td>Integer</td>
<td>No of flows that has a command in ftp session.</td>
</tr>
<tr>
<td>41</td>
<td>dst_host</td>
<td>Integer</td>
<td>No of connections that contain the same service (14) and source address (1) in 100 connections according to the last time (26)</td>
</tr>
<tr>
<td>42</td>
<td>dst_host</td>
<td>Integer</td>
<td>No of connections that contain the same destination (2) and destination address (3) in 100 connections according to the last time (26)</td>
</tr>
<tr>
<td>43</td>
<td>src_host</td>
<td>Integer</td>
<td>No of connections that contains the same destination (3) in 100 connections according to the last time (26)</td>
</tr>
<tr>
<td>44</td>
<td>dst_host</td>
<td>Integer</td>
<td>No of connections of the same source address (1) and the destination (3) in 100 connections according to the last time (26)</td>
</tr>
<tr>
<td>45</td>
<td>src_host</td>
<td>Integer</td>
<td>No of connections of the same destination address (3) in 100 connections according to the last time (26)</td>
</tr>
<tr>
<td>46</td>
<td>dst_host</td>
<td>Integer</td>
<td>No of connections of the same source address (1) and the destination (3) in 100 connections according to the last time (26)</td>
</tr>
<tr>
<td>47</td>
<td>src_host</td>
<td>Integer</td>
<td>No of connections of the same destination address (3) and the destination port (4) in 100 connections according to the last time (26)</td>
</tr>
<tr>
<td>48</td>
<td>dst_host</td>
<td>Integer</td>
<td>No of connections of the same source (1) and the destination port (4) in 100 connections according to the last time (26)</td>
</tr>
<tr>
<td>49</td>
<td>src_host</td>
<td>Integer</td>
<td>No of connections of the same destination address (3) and the destination port (4) in 100 connections according to the last time (26)</td>
</tr>
</tbody>
</table>

**Dataset selection**

- Data Cleansing & preparation
- Train Neural Network model
- Data Cleansing & preparation
- Generate Property Graph
- Load Property Graph into BDSG Graph Visualization
• One round of clean up.
  – Ports should be all integer based, however, there are **Hex** values
  – Action: convert them back to decimal

```
59,166,0,1,62377,149,171,126,4,53,udp,CON,0,001044,130,162,31,29,0,0,dns,498084,2813,620689,625.2,
192,168,241,243,259,192,168,241,243,49320,icmp,URH,0,1780,0,64,0,0,0,-196,4095,0,5,0,0,0,0,0,356,
192,168,241,243,49320,192,168,241,243,0x40a8,icmp,URH,0,1780,0,64,0,0,0,-196,4095,0,5,0,0,0,0,0,3
59,166,0,6,38993,149,171,126,0,53,udp,CON,0,00106,132,164,31,29,0,0,dns,498113,1875,618867,875.2,
59,166,0,9,59720,149,171,126,8,53,udp,CON,0,00107,132,164,31,29,0,0,dns,493457,9375,613084,125.2,
59,166,0,4,21489,149,171,126,7,53,udp,CON,0,00114,130,162,31,29,0,0,dns,454545,4588,556433,5625.2
59,166,0,8,45682,149,171,126,0,53,udp,CON,0,001257,130,162,31,29,0,0,dns,413683,375,515513,125.2,
59,166,0,8,32958,149,171,126,8,53,udp,CON,0,001124,132,164,31,29,0,0,dns,493750,9063,583629,9375.2
59,166,0,8,55879,149,171,126,3,53,udp,CON,0,001075,146,178,31,29,0,f5dn,543255,8125,662325,5625.2
59,166,0,0,43096,149,171,126,3,53,udp,CON,0,001114,132,164,31,29,0,0,dns,473967,8875,588868,9375.2
59,166,0,2,31439,149,171,126,1,53,udp,CON,0,001088,146,178,31,29,0,0,dns,536764,6875,654411,75.2,
59,166,0,3,45426,149,171,126,0,53,udp,CON,0,001053,132,164,31,29,0,0,dns,501424,5,622981,9375.2,
59,166,0,9,28993,149,171,126,3,53,udp,CON,0,001173,132,164,31,29,0,0,dns,450127,875,559249,8125.2
```
• Understand the data & define transformations

```scala
//removeColumns("timestamp start", "timestamp end", "source ip", "destination ip", 
"source TCP base sequence num", "dest TCP base sequence num","attack category
.filter(new FilterInvalidValues("source port", "destination port")) //Remove examples
.transform(new ReplaceEmptyIntegerWithValueTransform("count flow http methods", 0))
.transform(new ConditionalTransform("is ftp login", 1, 0, "service", Arrays.asList() 
.transform(new ReplaceEmptyIntegerWithValueTransform("count flow http methods", 0))
.transform(new StringToCategoricalTransform("service", ",", "dns", "http", "smtp", 
.transform(new MapAllStringsExceptListTransform("transaction protocol", "other", Arrays.asList(FIN, 
.transform(new MapAllStringsExceptListTransform("state", "other", Arrays.asList(FIN, 
.transform(new StringToCategoricalTransform("state", "FIN", "CON", "INT", "RST", "REJ", 
.transform(new IntegerToCategoricalTransform("equal ips and ports", Arrays.asList("is ftp login", 
.categoricalToOneHot("is ftp login", "equal ips and ports", "state","transaction protocol")
```

• Service “dns” becomes

0 1 0 0 0 0 0 0 0 0 0 0 0 0
• Executed transformations with Scala & Apache Spark using Oracle’s Big Data stack

```scala
val stringData = jsc.textFile("/user/oracle/UNSW-complete-all-removedhex.csv");
import org.datavec.spark.transform.AnalyzeSpark;
import org.datavec.spark.transform.SparkTransformExecutor;
import org.datavec.spark.transform.misc.StringToWritablesFunction;
val swf = new StringToWritablesFunction(recordReader);
val parsedInputData = stringData.map(swf)
val processedData = SparkTransformExecutor.execute(parsedInputData, tp);
```

• Save the RDD back to CSV format
• Built a Multi-Layer Perceptron (MLP) Neural Network

```java
conf = new NeuralNetConfiguration.Builder()
    .seed(seed)
    .iterations(iIter)
    .activation(Activation.TANH)
    .weightInit(WeightInit.XAVIER)
    .learningRate(learningRate)
    .regularization(true),12(1e-4)
    .list()
    .layer(0, new DenseLayer.Builder().nIn(numInputs).nOut(iLayer1).build())
    .layer(1, new DenseLayer.Builder().nIn(iLayer1).nOut(iLayer2).build())
    .layer(2, new OutputLayer.Builder(LossFunctions.LossFunction.NEGATIVELOGLIKELIHOOD)
        .activation(Activation.SOFTMAX)
        .nIn(iLayer2).nOut(outputNum).build())
    .backprop(true),pretrain(false)
    .build();
```
• Tested the quality of MLP NN
  
  • After 800 iterations of training
    
    Accuracy: 0.9811
    Precision: 0.9894
    Recall: 0.9286
    
    F1 Score: 0.958
    
    • Labeled as “non-intrusion” classified as “non-intrusion”: 46 times
    • Labeled as “intrusion” classified as “non-intrusion”: 1 time
    • Labeled as “intrusion” classified as “intrusion”: 6 times
      \[
      \frac{(46+6)}{(46+6+1)} = 0.9811
      \]
    
  • Long Short-Term Memory (LSTM) NN gave similar F1 result
• Network Intrusion Detection
  Property Graph
  • **Blue edges: malicious**
  • Other edges: normal traffic
  • Many attacks originated from
    175.45.176.1 to target
    149.171.126.17
  • Visualization tool: Cytoscape v3.2.1
    + Big Data Spatial and Graph v2.1
• Focused on “Attacks” graph
• Focused on “Attacks” graph
• Focused on “Attacks” graph

• Applied built-in analytics in BDSG

• Found top-3 IP addresses with **highest Page Rank** value
Wrap-up
Summary

Graph analytics and machine learning

• Graph databases are powerful tools, complementing machine learning technologies
  – Especially strong for analysis of graph topology and multi-hop relationships
• Graph analytics offer new insight which can be used as input to machine learning
  – Especially relationships, dependencies and behavioural patterns
• Oracle Big Data Spatial and Graph and Oracle 12.2 Spatial and Graph offer
  – Comprehensive analytics through various APIs
  – Scaleable, parallel in-memory processing with 40+ graph algorithms pre-built
  – Integration with R, integration with SPARK, integration with relational database
  – Secure and scaleable graph storage on Hadoop using Oracle NoSQL or HBase or Oracle database
• Running both on-premise or in the Cloud
Resources

• Oracle Big Data Spatial and Graph OTN product page:  
  www.oracle.com/technetwork/database/database-technologies/bigdata-spatialandgraph  
  – White papers, software downloads, documentation and videos

• Oracle Big Data Lite Virtual Machine - a free sandbox to get started:  

• Hands On Lab included in /opt/oracle/oracle-spatial-graph/  
  – Content also available on GITHUB under http://github.com/oracle/BigDataLite/

• Blog – examples, tips & tricks:  blogs.oracle.com/bigdataspacialgraph

• @OracleBigData, @SpatialHannes, @agodfrin, @JeanIhm

• [Link to Oracle Spatial and Graph Group]
Interested in project experience, best practices, networking?

Spatial and Graph Summit

- IOUG Business Intelligence, Warehousing and Analytics SIG have established annual BIWA Summit
  - Rebranded as Analytics and Data Summit
  - Planned for March 20 – 22, 2018 at OracleHQ

- Spatial and Graph Summit is separate track
  - Lots of interesting material from previous years available on OTN

- Opportunity for interaction with Spatial PM and Dev’t team
Q&A
Integrated Cloud
Applications & Platform Services