A Graphical Model for Threat Analysis

Present by:
Major Manjunath Bilur, Anugrah Gari
173054001, 173050078

Guided by:
Prof R. K. Shyamasundar

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Mumbai

June 21, 2019

Outline

1. Model
2. Outcome
3. Procedure Applied
4. Demo
5. Conclusion and Future scope

Model

- To asses to any enterprise business process using attack tree for threat
    - Most application are standardized to BPMN -II(by OMG)
- Derive task flow logic from BPMN notation and arrive at logical flow path for the application
- Semi auto creation of input.P file - Input to MulVAL
- Auto generate the Interaction rules for the application based on input.P.The generated interaction rules would still require manual verification and up-dation by the admin corresponding to application under assessment.
- Generate attack tree and its logic from MulVAL
- Port the data into .CSV with necessary headers into import folder of Neo4J(Graph DB)

Model

- Recosntruct the attack tree and assess with Cypher Queries:
  - Impact calculation : Based on Base scores of vulnerability (NIST)- Concept from Dr Anoop Singhal[2]
  - Probability based assessment - Concept from Prof Sushil Jajodia[1]
  - Classification of level of threat acceptable as Green,Yellow and Red
  - Possibility of insider attacker
  - Assessment based on other metrics like:
    - Noticeability
    - Cost
    - Technical skill
  - Points for hardening the existing network

Outcome

- First semi auto process of logical path generation from BPMN and Input.P
- Auto generation of interaction rules which are core to attack tree generation by MulVAL
- First time assessing the attack model on graph DB
- Ability to assess and scale the assessment to any required metric with suitable allocation or modification
- Ability for admin to derive for points of hardening in the network

Procedure Applied

- BPMN (Logical Task derivation from .bpmn file)
- Input.P(Semi-Auto)
- Interaction Rules generation
- Porting of logic and data to Neo4j
- Query processing to generate the required attack tree
- Assessment of attack tree

Basic BPMN Concepts

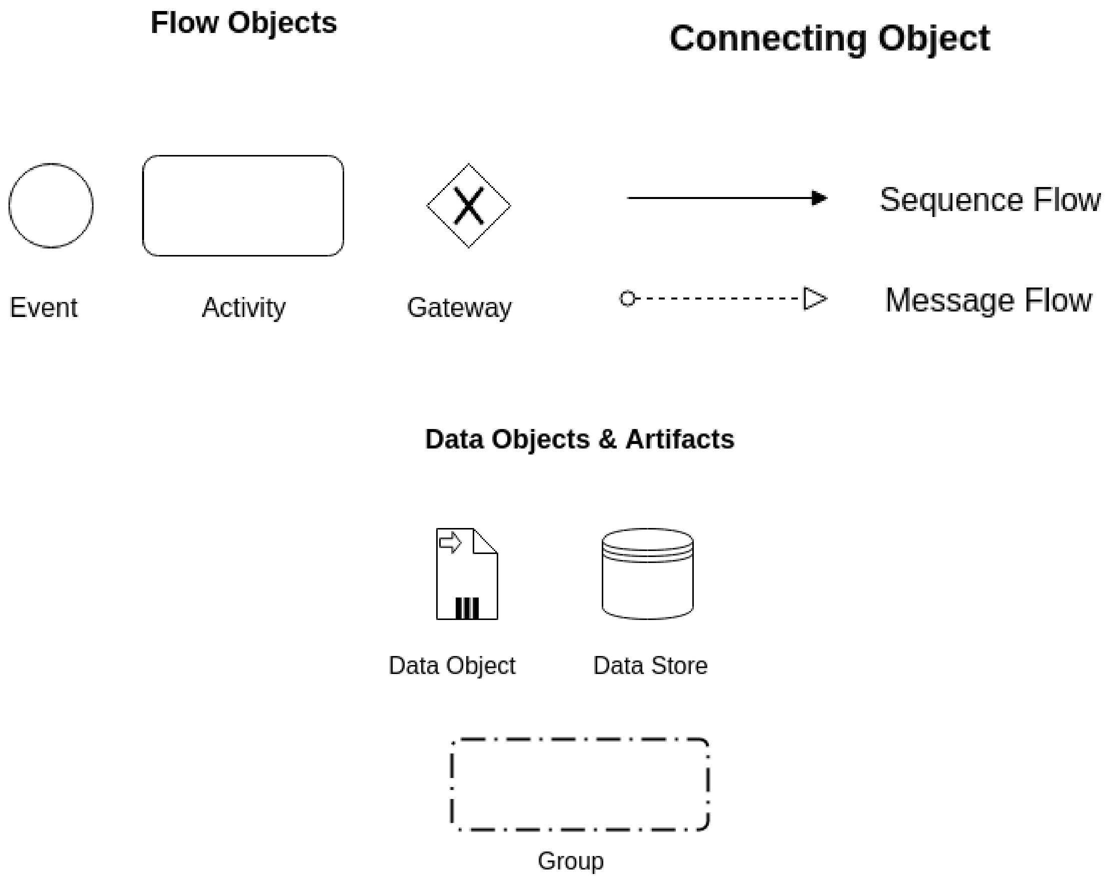
BPMN

Business Process Model and Notation (BPMN) is a standard for business process modelling, and provides a graphical notation for specifying business processes based on a flow chart technique.

Business Process Modeling

Business Process Modeling is the activity of representing process of an enterprise, so that the current process may be analyzed and improved by other professionals.

Core BPMN Design Elements



Flow Objects

Connecting Object

Event

Activity

Gateway

Sequence Flow

Message Flow

Data Objects & Artifacts

Data Object

Data Store

Group

BPMN 2.0 Diagrams

BPMN can represent Business Models by 4 kinds of diagrams:

- **Process Diagrams**: Represents regular flow between tasks, event and decision points to complete a process in the company.
- **Collaboration Diagrams**: Represents message flow or communication routes between process or entities like customers or partners.
- **Conversation Diagrams**: Represents groups of messages called communications and its relation between process and participants.
- **Choreography Diagrams**: Represent participant interaction between task and user or resources and the messages result of this interaction.

BPMN 2.0 Diagrams

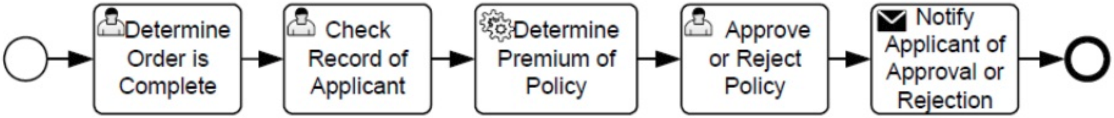

Determine Order is Complete
Check Record of Applicant
Determine Premium of Policy
Approve or Reject Policy
Notify Applicant of Approval or Rejection

Figure 1: Process Diagram

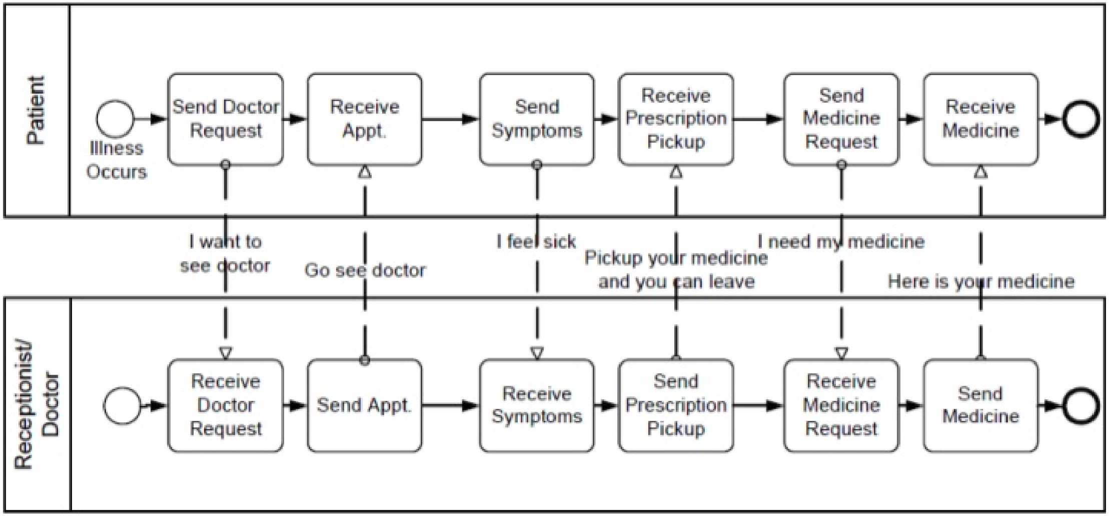


Figure 2: Collaboration Diagrams

BPMN 2.0 Diagrams

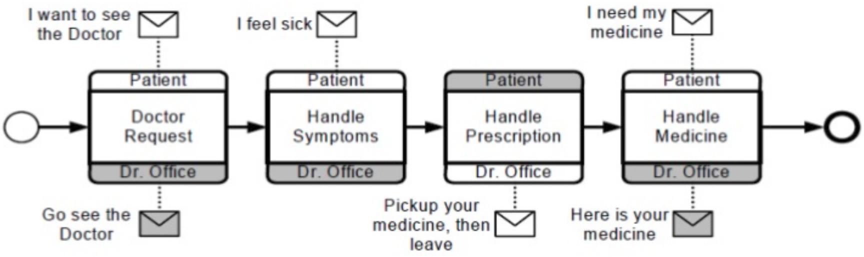


Figure 3: Choreography Diagrams

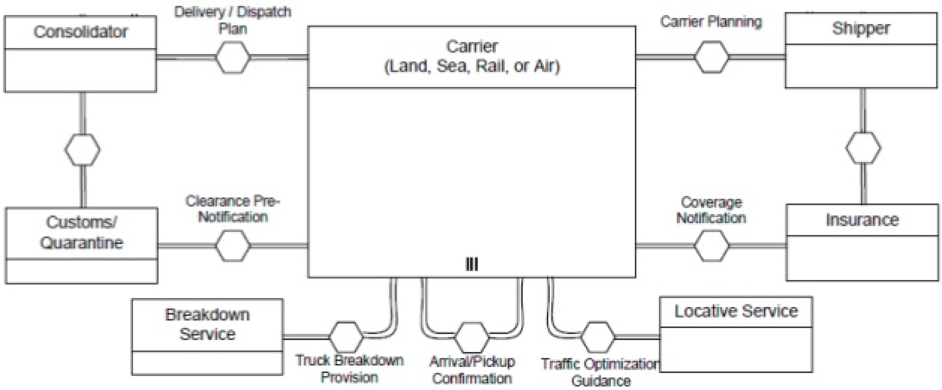


Figure 4: Conversation Diagrams

Application for which the model is demonstrated(BPMN DIAG)

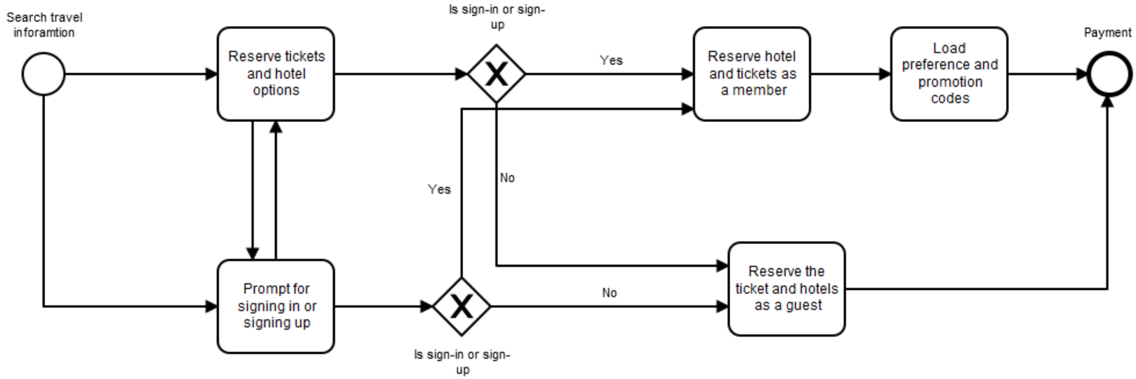

Search travel inforamtion
Reserve tickets and hotel options
Is sign-in or sign-up
Yes
Reserve hotel and tickets as a member
Load preference and promotion codes
Payment
Yes
No
Prompt for signing in or signing up
Is sign-in or sign-up
No
Reserve the ticket and hotels as a guest

Figure 5: BPMN for Travel Reservation System

Application for which the model is demonstrated(BPMN DIAG)

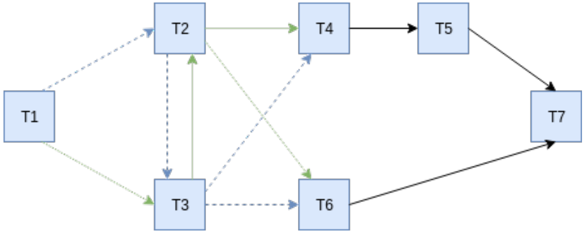


Figure 6: Exceution Paths

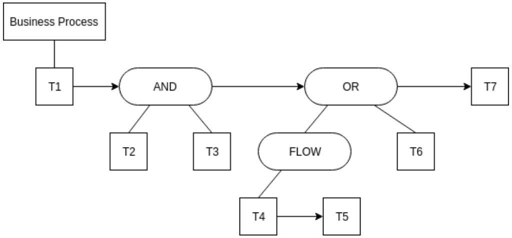
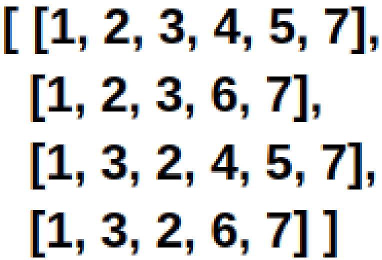
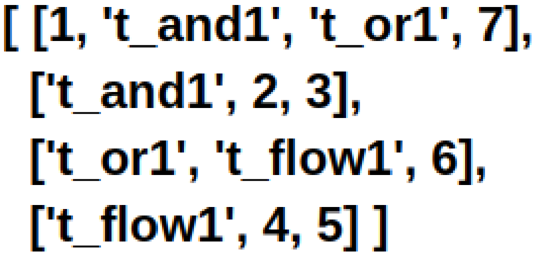


Figure 7: Dependency Tree



```
[ [1, 2, 3, 4, 5, 7],
  [1, 2, 3, 6, 7],
  [1, 3, 2, 4, 5, 7],
  [1, 3, 2, 6, 7] ]
```



```
[ [1, 't_and1', 't_or1', 7],
  ['t_and1', 2, 3],
  ['t_or1', 't_flow1', 6],
  ['t_flow1', 4, 5] ]
```

BPMN

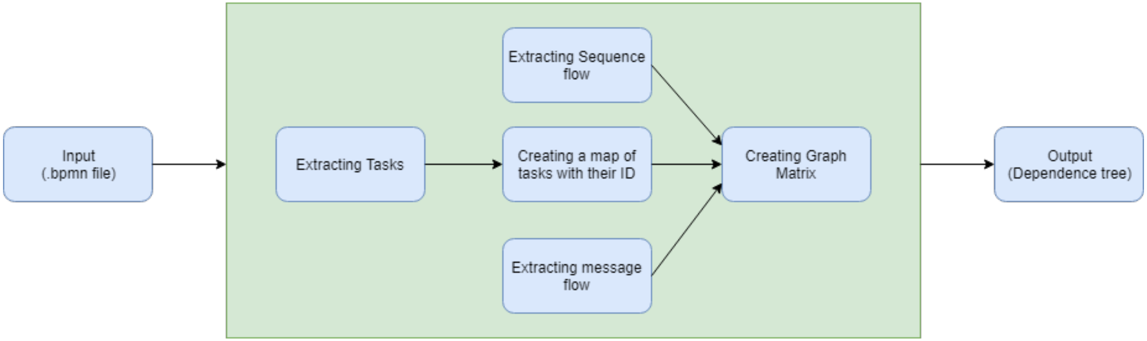


Figure 8: Conversion of BPMN to Dependency Tree

Conversion of BPMN to Dependency Tree

- *script.py* takes .bpmn file as an argument
- After providing the appropriate file to the script, the first operation of the script is to extract the tasks from the file which is done by providing appropriate regular expressions.
- Once the tasks are being extracted, the next operation of script is to create a map of these tasks with their IDs which will be further used during graph creation.
- Apart from extracting the tasks from the file, the script also needs to extract the sequence flow and message flow between the tasks in order to generate the graph. So the next operation script does is to extract the sequence and message flow among the tasks from the file. The sequence and message flow contains two crucial information that is the source ID and target ID

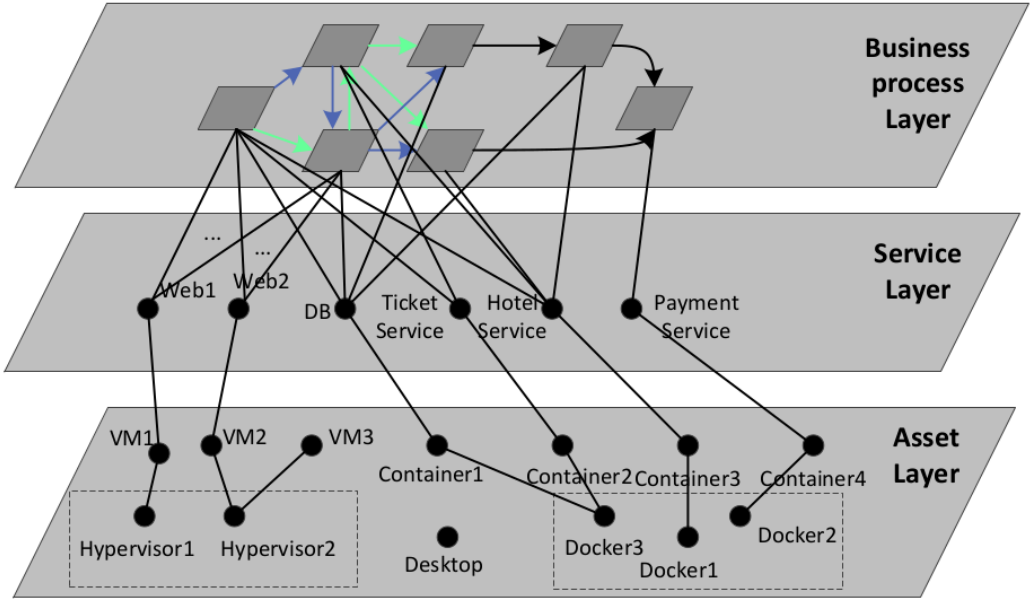
Conversion of BPMN to Dependency Tree

- The source IDs and target IDs are being checked against the map in order to find the corresponding tasks.
- Once the source and target tasks are being found, accordingly a 2D matrix is created which represents a directed graph of the tasks as node and their dependence.
- Now from this matrix, we find all possible path from the start to the end node.
- After finding all possible path from start node to end node, we select only those path which depict our actual task flow.
- From this selected path, now we build the dependency tree by traversing through each node of each path simultaneously. Based on the current node and the next node we determine what dependency the current node will have.

Applied Algorithm for generating Dependency Tree

- If a task is *and-depends* on task 1 and task 2, then the task is impacted by the attacker if either of the task is impacted.
- If a task is *or-depends* on task 1 and task 2, then the task is impacted only when both the task are impacted.
- If a task is *flow-depends* on task 1 and task 2, then this task is impacted when task 2 is impacted and task 2 can be completed when task 1 is completed.

Layer dependency for the application

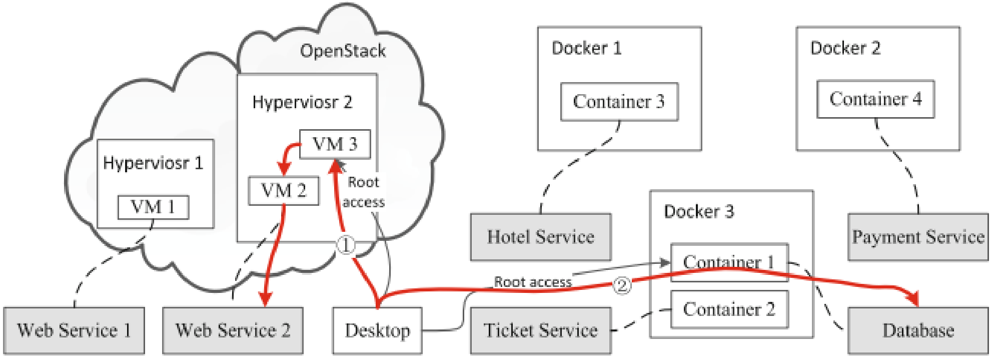


Input.P

Applied Algorithm:

- System takes input form network admin
  - Deployment
  - Configuration
  - Network Information
  - Access Rules
  - Vulnerabilities
- Admin can include additional options above according to his Business Model
- Admin must add fact nodes based on Business Model as this step cannot be fully automate
- The input should correspond to software architecture and physical deployment

Software Architecture of the Application



Procedure

- Python script to take input from admin and also generates input.P
- The output should correspond to actaul application
- The syntax should be as mandated by MulVAL

Interaction Rules

- The Interaction rules define the complete logic for attack tree generation in MulVAL
- Since the rules define the flow paths:
    - They are application specific
    - Needs to be specifically created for each application based on
        - Software Architecture (Define access controls and application deployment)
        - Physical assets and their dependencies on above
- Default MulVAL rules are already embedded which are in general applicable to all applications
- Need to create new interaction rules which are specific to application under scrutiny

Algorithm Applied

Aim : Derive automatically the interaction rules from input.P. Some may have to be incorporated as input from config files specific to application.

From input.P

- Derive all primitive fact and derived nodes for the application
  - Some may be outcome due to access controls and dependencies(Derived)
- Derive the logical
  - Node impacts based on flow model
  - Tasks that constitute the application
  - Intermediate Impacts which allow attacker to reach the goal
- Privilege to execute any arbitrary code so as to progress
  - This should encompass from the privilege the system provides coupled with attacker gains by exploiting vulnerabilities
- All dependencies should be incorporated
- Place the rules in MulVAL and set correct path

Interaction Rule

```
interaction_rule((nodeImpact(t7):-
node(t7, and, web1, vm1, web2, vm2, payd, container4),
node(business_process, flow, t1, t_and, t_or, t7),
nodeImpact(t_or),
nodeImpact(web2, vm2)),
rule_desc(’(nodeImpact(t7)’, 0.2)
).
```

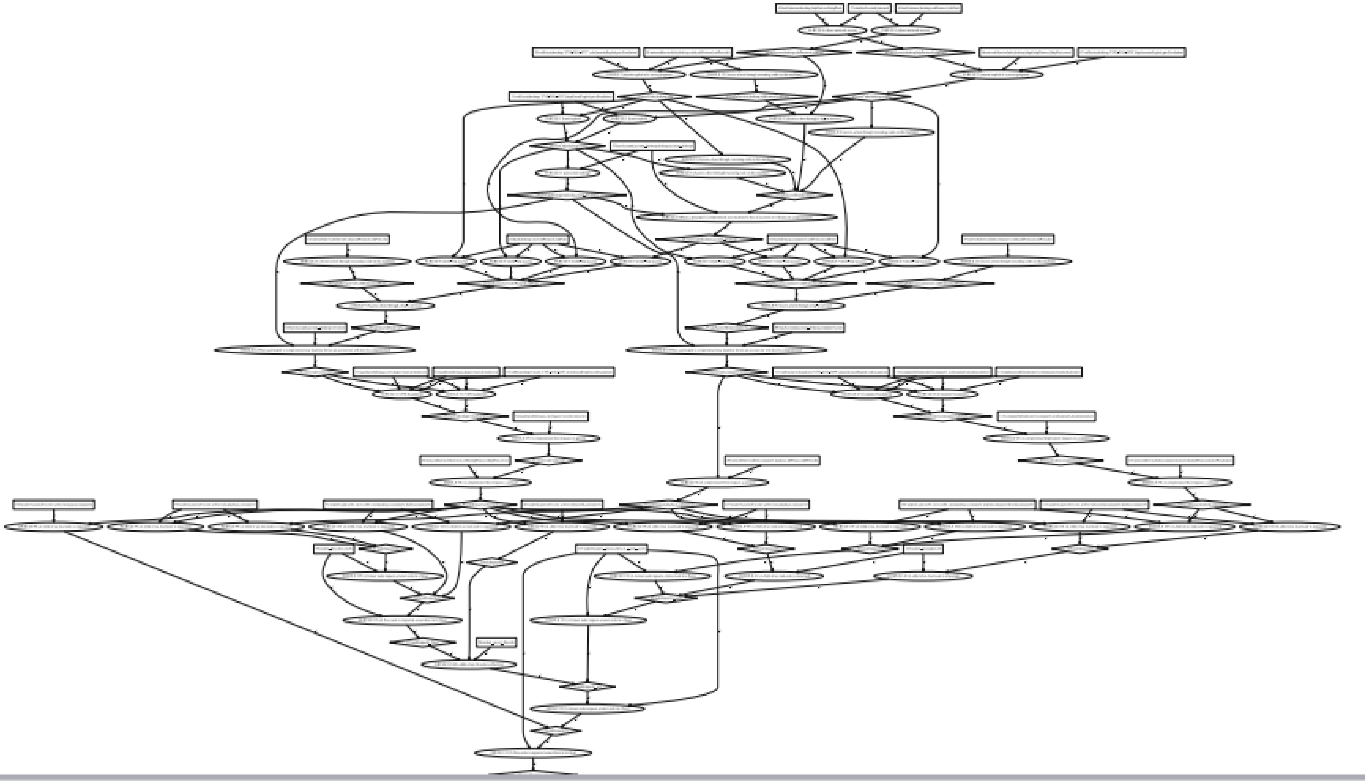
MulVAL

- Generates:
  - Vertices.csv (All nodes in the attack tree)
  - Attacktree.txt (Logic for derivation of parent and child)
  - Attackgraph.eps (Actual attack tree)
- Disadvantages:
  - Static view (Just image)
  - No more further analysis on the graph
  - Not scalable to any metric assessment
  - Difficult to comprehend if tree is more complex
  - Intermediate assessment not possible

Solution

- Derive logic of parent and child from attackgraph.txt - Python script
    - relations.csv
    - Add necessary headers for accessing in Neo4J
- Import vertices.csv to Neo4J with appropriate headers to access
- Both .csv files in import folder in Neo4J graph DB database

Attack Graph - MulVAL

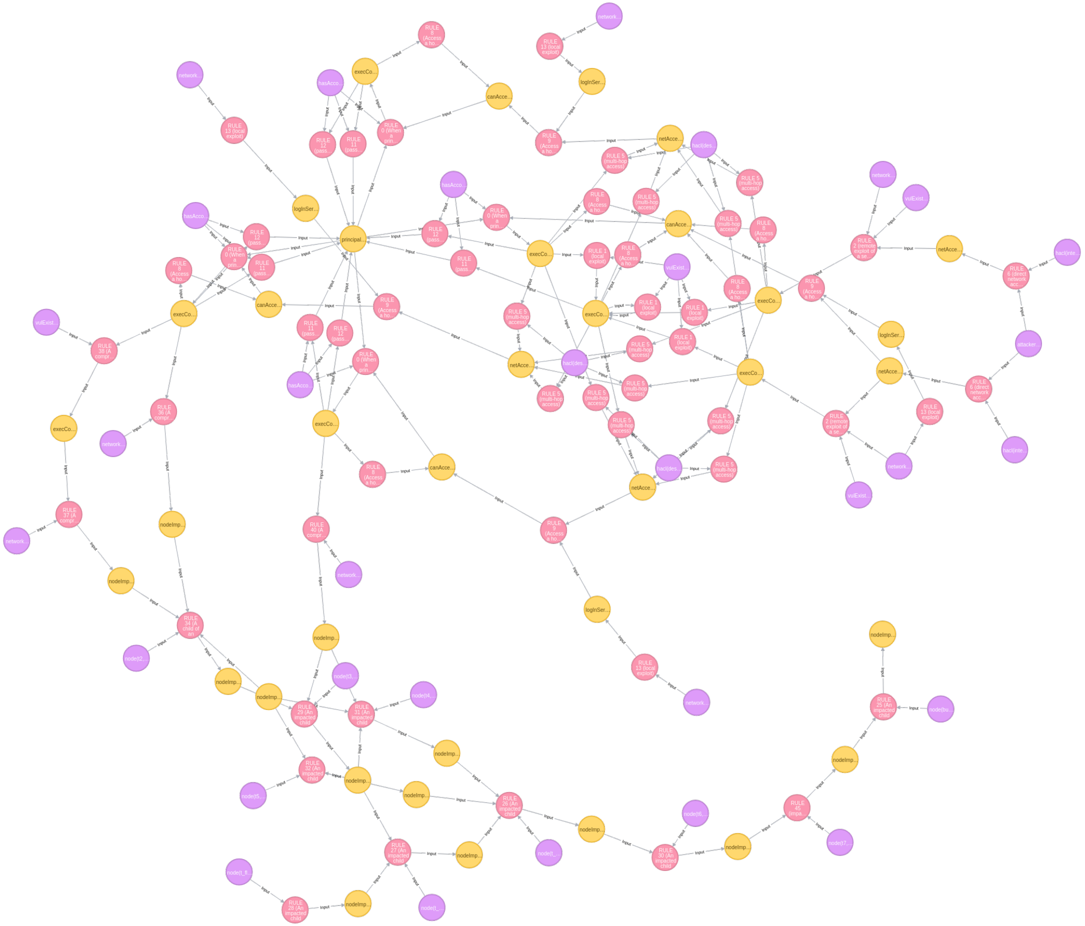


Neo4J

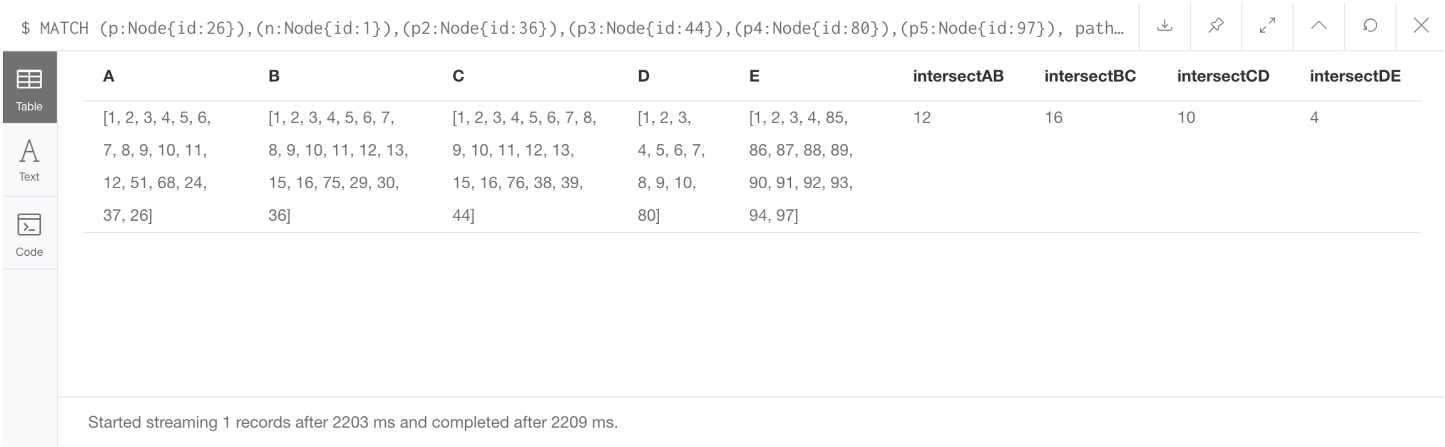
Write cypher queries

- To import nodes form Vertices.csv into Neo4J
- To import relations among all nodes
- To classify them as leaf, rule and derived node
- To assign any metric like:
  - CVSS score
  - Probability
  - Noticability
  - Cost
  - Technical skill requirement
- To assess
  - Impact calculation based on CVSS
  - Probability based shortest path
  - Other metric based attack paths
  - Points for hardening
  - Zone classificaiton
- To internal attacks

Graph by Neo4J



Intersection point & Classification of Zones

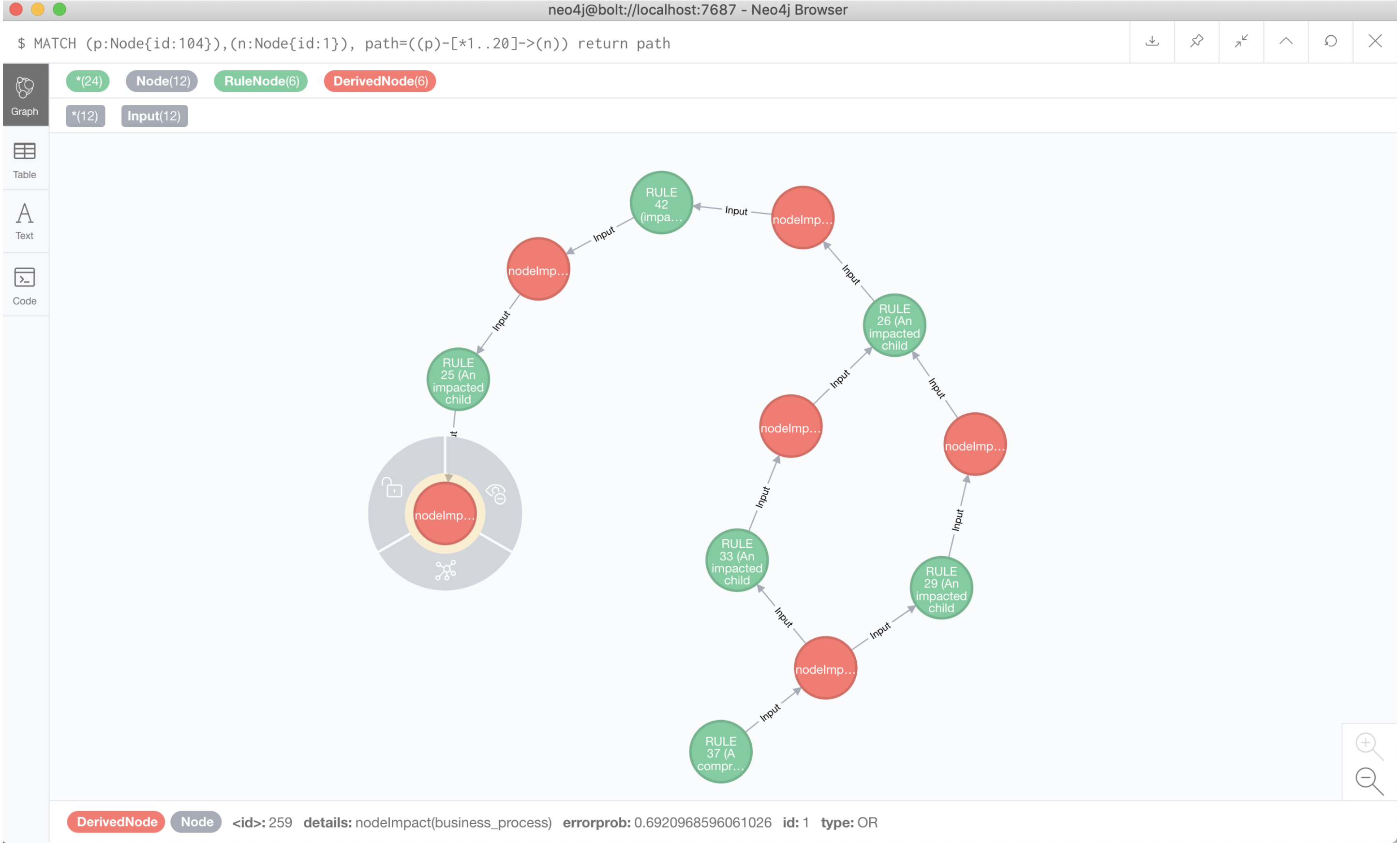


```
$ MATCH (p:Node{id:26}),(n:Node{id:1}),(p2:Node{id:36}),(p3:Node{id:44}),(p4:Node{id:80}),(p5:Node{id:97}), path…
```

A	B	C	D	E	intersectAB	intersectBC	intersectCD	intersectDE
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 51, 68, 24, 37, 26]	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 75, 29, 30, 36]	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 15, 16, 76, 38, 39, 44]	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 80]	[1, 2, 3, 4, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 97]	12	16	10	4

Started streaming 1 records after 2203 ms and completed after 2209 ms.

All Paths from high probable node to goal



neo4j@bolt://localhost:7687 - Neo4j Browser

```
$ MATCH (p:Node{id:104}),(n:Node{id:1}), path=((p)-[*1..20]->(n)) return path
```

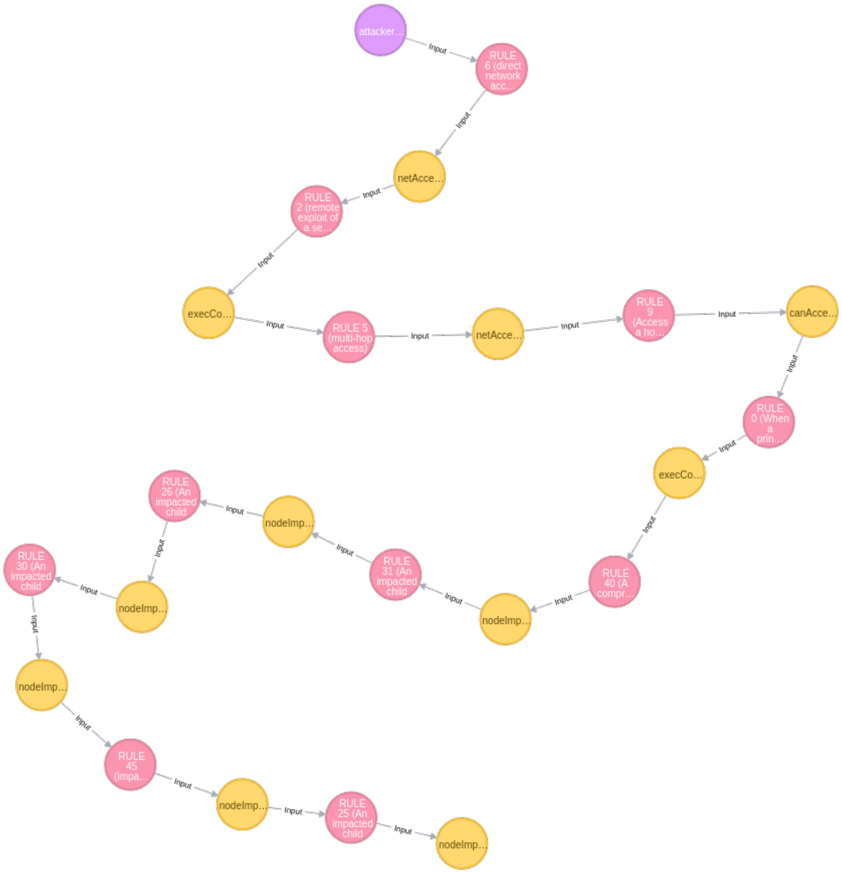
*(24) Node(12) RuleNode(6) DerivedNode(6)

*(12) Input(12)

DerivedNode Node <id>: 259 **details:** nodeImpact(business_process) **errorprob:** 0.6920968596061026 **id:** 1 **type:** OR

Graph
Table
Text
Code
RULE 42 (impa…
nodeImp…
Input
RULE 25 (An impacted child
RULE 26 (An impacted child
RULE 33 (An impacted child
RULE 29 (An impacted child
RULE 37 (A compr…

Shortest path from one node to another



Conclusion and Future scope

- The model can act as base as interaction rules is automated for any kind of assessment
- Graph db usage has enhanced to articulate any kind of metric to nodes and do assessment
- Can act as tool to real-time monitoring of any untoward activity
- Can act as strict vigilance for any insider attack
- Levels of measures can be incorporated for better network hardening

Biblography

- Massimiliano Albanese and Sushil Jajodia.
  A graphical model to assess the impact of multi-step attacks.
  Journal of Defense Modeling and Simulation, 15(1):79–93, January 2018.
  Selected by the Guest Editor, Alexander Kott, as an article of particular value.

- Chen Cao, Lun-Pin Yuan, Anoop Singhal, Peng Liu, Xiaoyan Sun, and Sencun Zhu.
  Assessing attack impact on business processes by interconnecting attack graphs and entity dependency graphs.
  2018.

Thank You