MasonBayesian-ATTaCK-Graph

GMU CYSE 650 - Summer 2024

Overview

Software translates MITRE's ATT&CK framework into a bayesian node representation. Input files must be in .json format and conform to MITRE standards as of July 2024 (ATT&CK 15.1), Output files will be in a .net format.

Walkthrough

From GitHub Repo:

1. Install dependencies `pip install pgmpy`
2. Place .json files into the input directory
3. Run `python JSONtoNET` in main directory
4. Terminal will display whether the conversion was a success/failure
5. If successful, output .net files will be available in the output directory, with copies of the original .json files in a backup sub-directory
6. Graphical output of these .net files works best with Java-based UnBBayes applicaiton

Overall Approach

- Recursive analysis of each file in input directory
  - Read data
  - Validate fit-to-format
  - Evaluate JSON nodes
  - Convert JSONtoNET
  - Backup JSON
- Read is done using default python libraries, directory structure rigid
- Validation is done using MITRE-distributed Structured Threat Information Expression (STIX), basic formatting errors are caught if variable fields are present
- Evaluation of nodes is done using **pgmpy**, sequential approach that groups nodes by their stronger relations to one-another and their inherited/parental status
- JSON to Net conversion done in plain-text, header is created and then each node is written in a 3-line format: label, position, and state
- JSON file is then backed up to a subfolder, to prevent potential data loss due to file conversion

MITRE STIX

- MITRE STIX used to format all data into one constant format
- Presence of basic variable fields will validate and conform data, there are minimal false negatives
- False positives occur due to not rigorously validating each field, which could be done via extended catch arguments for STIX format

https://github.com/mitre/cti/tree/master?tab=readme-ov-file#stix

https://github.com/mitre/cti/tree/master/enterprise-attack

pgmpy

- Python library for Bayesian network evaluation
- Implemented discrete pgmpy to represent scores for different ATT&CK values
- Each node generated is linked to other nodes via this library, which is then used by UnBBayes for graphical rendering
- Nodes are evaluated linearly with score, some grouping would be expected if this was to be fine-tuned for specific risk profiles.
- Superseded by scores provided via JSON

https://pgmpy.org/factors/discrete.html

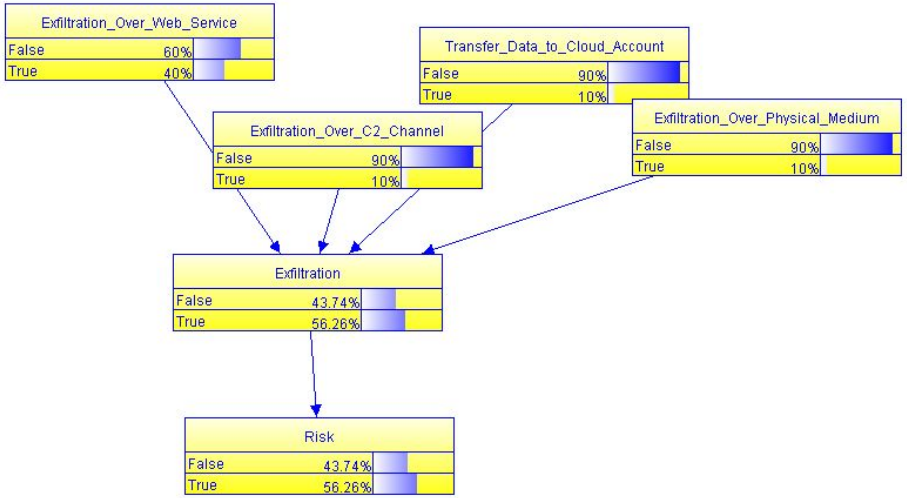
Scenarios

1. General system vulnerability assessment
    - Determine where to commit resources for broad value
    - Inward focus

2. Target adversarial assessment
    - Determine where to commit resources to minimize potential vulnerabilities or minitage damage
    - Outward focus

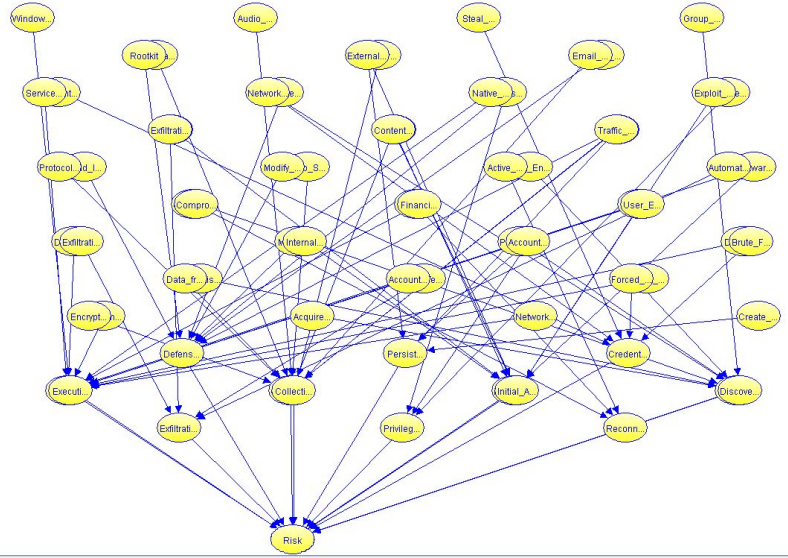
Scenario 1

- Scenario 1 provides a small focus on exfiltration risk, which is 1:1 with total system risk in this scenario.
- Risk is assessed at ~56%, as a result of 4 input risks of 10-40%



Scenario 2

- Extremely dense network graph based on target capabilities
- This scenario would benefit from an ability to evaluate risk logarithmically rather than linearly
- Overall values approach 100% due to quantity, hard to isolate particular areas of improvement
- Screenspace constrained by UnBBayes, could investigate increasing maximum window size



Questions?