CYSE 650 Final Project

Andy, Connor, and Wesley

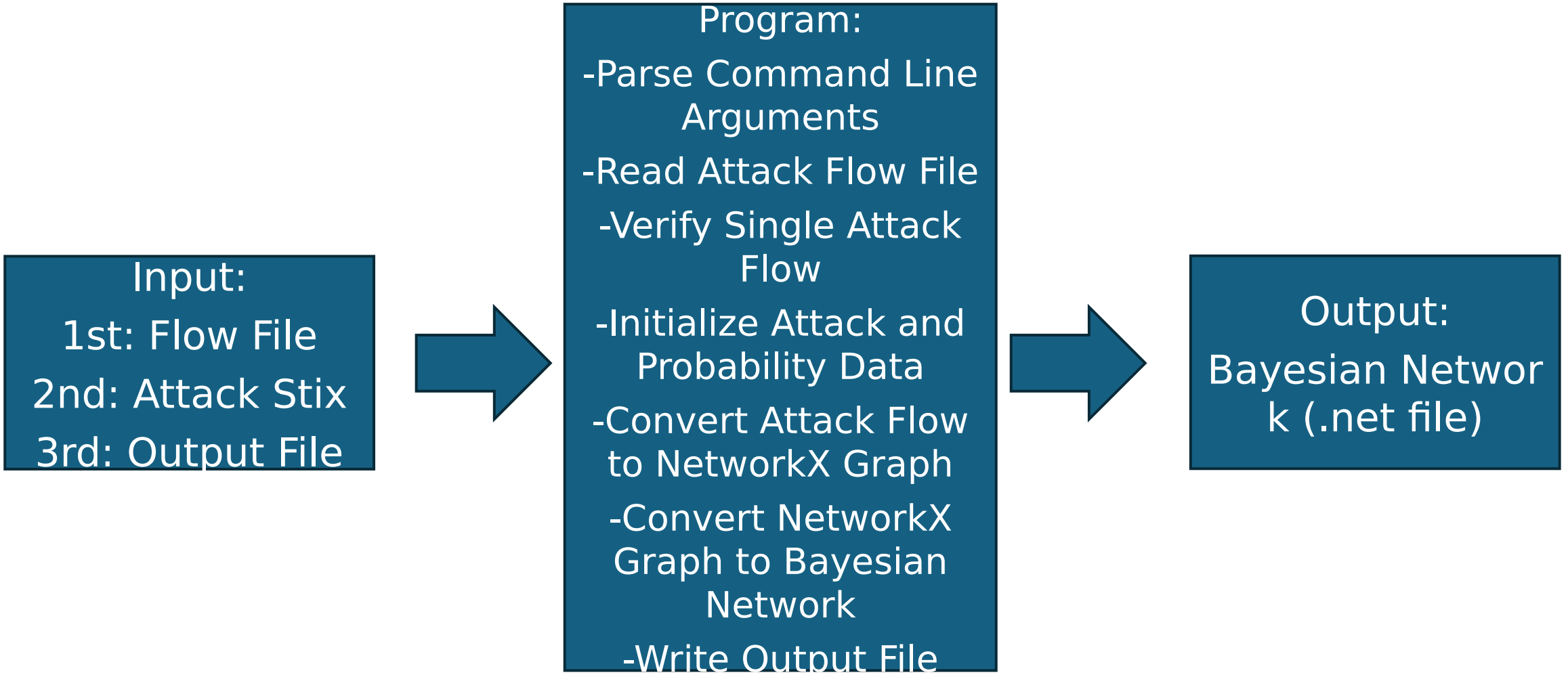
Problem Statement

Develop a framework to analyze **cyber-attack patterns and probabilities using Bayesian networks**, in an automated and user-friendly fashion.

How to Install/Configure

- GitHub Link: (More in-Depth Installation Guide in the Repo ReadME)
  - https://github.com/Cyber-Musketeers/ATTACK-RISK
- Step 1: Clone the repository:
  - RUN: git clone git@github.com:Cyber-Musketeers/ATTACK-RISK.git
- Step 2: Install poetry
- Step 3: Poetry Shell
- Step 4: Make flow
- Step 5: Export flow
- Step 6: Run main.py (ATTACK-RISK/src/main.py) with the following arguments:
  - 1st: Attack Flow File
  - 2nd: MITRE ATT&CK Stix File
  - 3rd: Output File
- Step 7: You will receive a complete Bayesian network file (.net) to be used within your program of choice.

Software Architecture



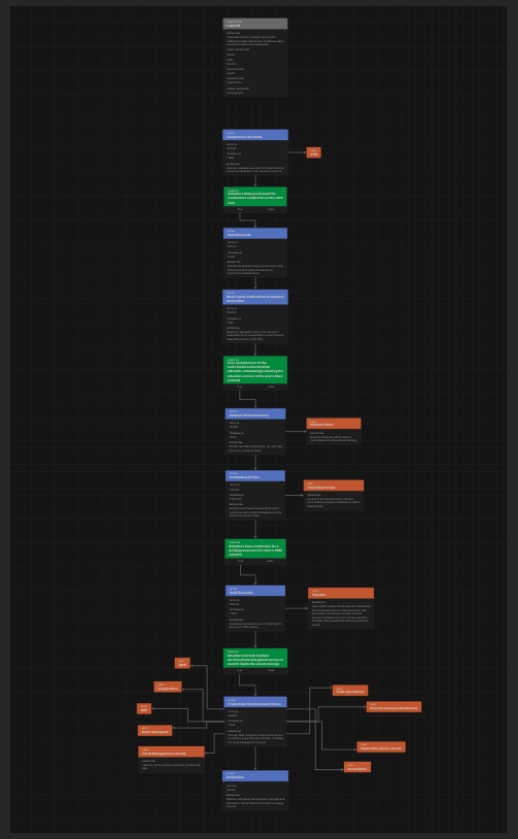
Software Architecture (Cont.)

- Dependencies:
  - Networkx
  - Numpy
  - Stix2
  - Mitreattack.stix20
  - Pgmpy

(Probability values are gathered from comparing the number of times TTPs appear in ATT&CK campaigns)

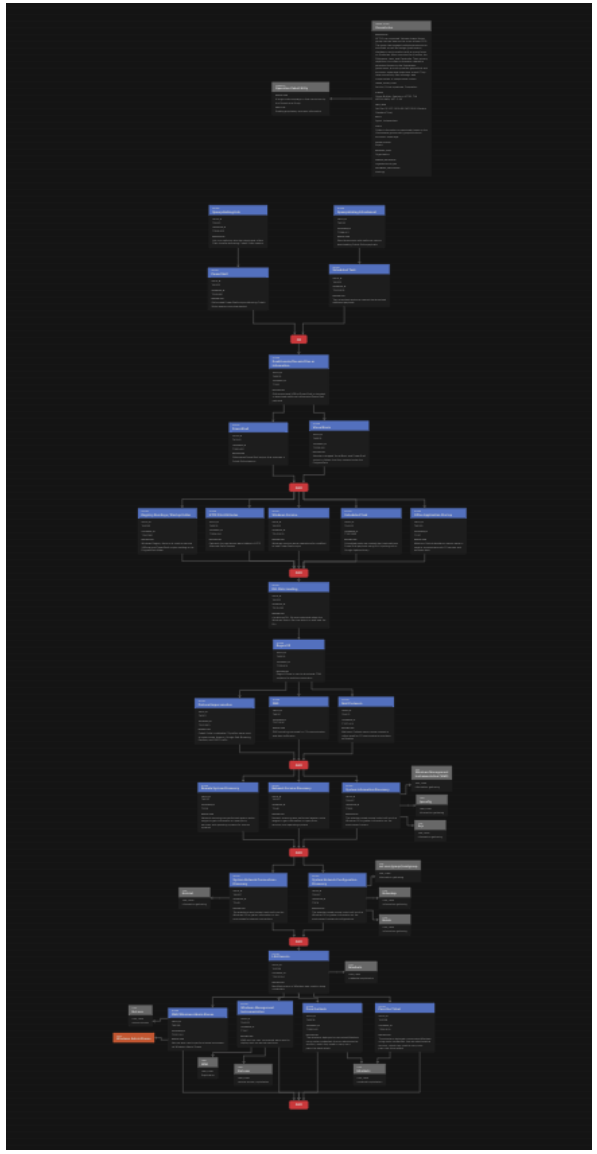
```python
def main() -> None:
    """
    Main function of the program.
    """
    args = parse_args()
    # print_flow(args.flow_file)
    flow_bundle: stix2.Bundle = read_flow_file(args.flow_file)
    flows = flow.get_flows_from_stix_bundle(flow_bundle)
    # technically there can be multiple attack flows in a stix bundle but I am too lazy to deal with that
    if len(flows) != 1:
        raise ValueError("Expected exactly one attack flow in the file.")
    attack_data = MitreAttackData(args.attack_stix)
    probability_db = weights.ProbabilityDatabase(attack_data)
    flow_nx = convert_attack_flow_to_nx(flows[0], flow_bundle)
    flow_nx = make_nx_graph_more_readable(flow_nx)
    bayesian_network = flow_nx_to_pgmpy(flow_nx, probability_db)

    content = pgmpy_to_unbbayes_hugin(bayesian_network)
    with open(args.output_file, "w", encoding="utf-8") as file:
        file.write(content)
    print("New bayesian network written to", args.output_file)
    print("The network can be loaded into Hugin or Unbbayes")
    print("Thanks for using the program!")
```

Tool Functionality Scenario 1: Uber Breach



```
python3 ./main.py --flow_file uber.json --attack_stix enterprise-attack.json --output_file uber.net
```

The Uber breach has a more linear attack flow which can be used to showcase the implementation into the Bayesian network and the associated risk values. The flow goes from compromised accounts to exfiltration.

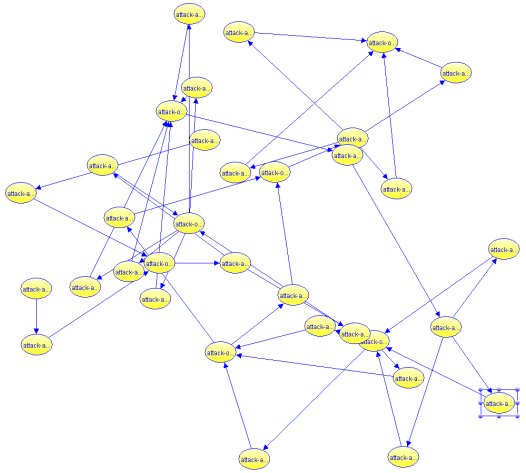


Tool Functionality Scenario 2: Cobalt Kitty

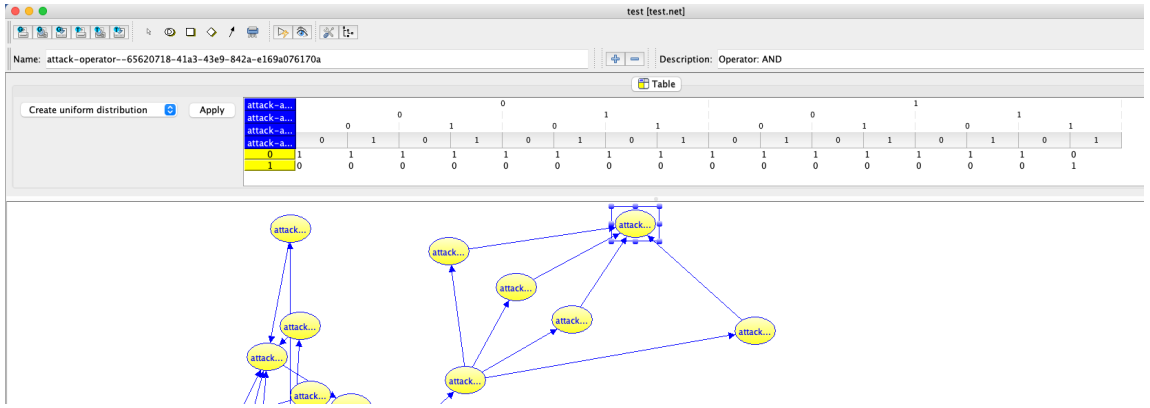
- This attack flow is much more complex and has multiple actions within certain layers
- Ultimately showcases the risk associated within Cobalt Kitty and the associated TTPs within each layer.

```
python3 ./main.py --flow_file ck.json --attack_stix enterprise-attack.json --output_file ck.net
```

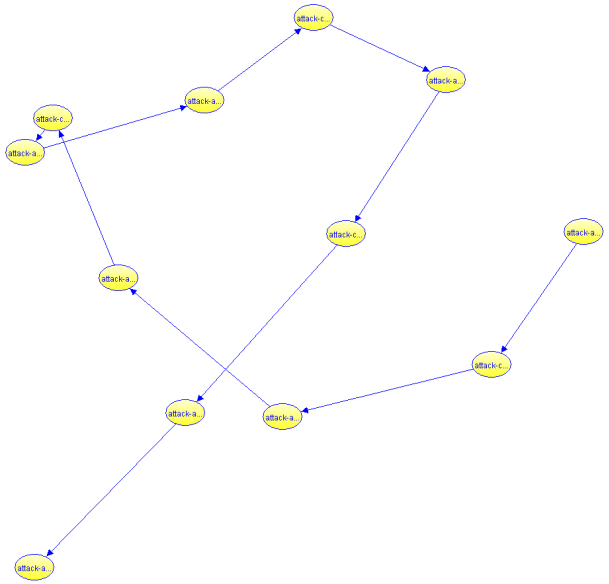
Results

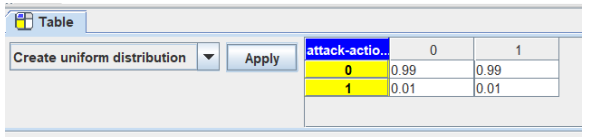


Cobalt Kitty



Uber





Limitations

- If a technique doesn’t show in any ATT&CK campaign, or if a tactic does not have an associated technique, we are assuming that it has a **0.01%** chance probability (Uber output)
- Methodology for probabilities is weak (**less than half** of TTPs have a campaign!)
- For condition blocks within ATT&CK flows, we are only considering values of **"True"**
- We are only accepting **one** attack flow at a time
- **Slow** startup time

Future Works

- Improve weight calculation to consider more than “if it showed up in many campaigns, it must be likely”
- Handle “false” determinations in “Attack Condition” blocks
- Better weight calculation for “Default” probabilities
- Less-naïve implementation of CPTs
- Better node names