



# Provenance Graph Generation for Intrusion Detection

## Simulating Scenarios for Graph Convolutional Network Classification

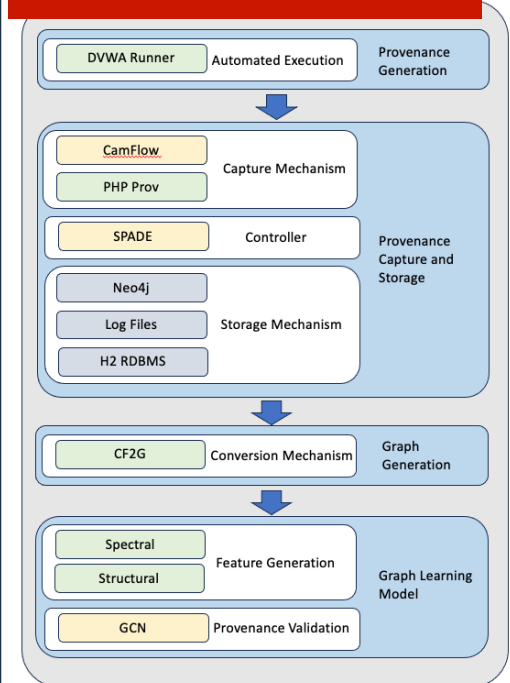
Student: LIM ZIYI JANESSE

Supervisor: A/P KE YIPING KELLY

### Project Objectives:

In this study, our objective is to validate the effectiveness of provenance graphs in intrusion detection by generating and analysing benign and malicious user scenarios. Leveraging the CamFlow provenance capture system and the Flurry framework, we will simulate diverse intrusion scenarios and generate authentic provenance data. Our aim is to evaluate state-of-the-art graph-based models for intrusion detection using performance assessment metrics such as classification accuracy, precision recall and f1-score. Through this investigation, we aim to contribute to the advancement of intrusion detection methodologies and enhance our understanding of provenance-based defense mechanisms.

### Framework



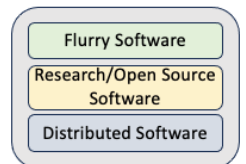
As adapted from Flurry's framework [1]

### Graph Generation

```

restored--Execute XSS Stored Attack
NoneReflected--Execute XSS Reflected Attack
Random--Execute XSS DOM Attack
CommandInjection--Execute Command Injection Attack
sqlInjection--Execute SQL Injection Attack
NoneForce--Execute Brute Force Attack
CustomAttack--Use My Own Attack
message--Message Board Post (Benign version of XSS stored)
submit--Complete a Questionnaire (Benign version of XSS reflected)
query--Query a Webpage (Benign version of XSS DOM)
ping--Ping local host (Benign version of command injection)
databaseentry--Create a User ID (Benign version of SQL injection)
login--Enter Username and Password (Benign version of brute force)
[...many more benign attacks...]
Select one or more attacks to run in a comma-separated list: submit
>> 1 iteration(s) made
>> Whole system provenance capture granularity is used
> COARSE granularity (ie W3C-PROV model types) for edge types is used
> FINE granularity (ie CamFlow-provided node and edge types) for node types is used
Driver initialized.
Connecting MQTT subscriber...
Connected with result code 0
[side] password for teehee2:
running /home/teehee2/flurry/flurry/scripts/questionnaire.py
Saving graph to database...
type: 0.014864444732668016
edges: 1: 811981201171875e-05
Graph 191 saved.
Drawing graph...
Graph drawn to output/benign/graph191/graph191.png.
converting graph to pickle...
Making NetworkX Graph...
Graph constructed.
Graph pickle outputted to output/benign/graph191/graph191.gpickle.
outputting node types to JSON format...
Node types outputted to output/benign/graph191/nodetypes191.json.
outputting edge types to JSON format...
Edge types outputted to output/benign/graph191/edgetypes191.json.
outputting graph to JSON format...
Graph outputted to output/benign/graph191/graph191.json.
  
```

### Provenance Graph



Legend

[1] M. Kapoor, J. Melton, M. Ridenhour, M. Sriram, T. Moyer, and S. Krishnan, 'Flurry: a Fast Framework for Reproducible Multi-layered Provenance Graph Representation Learning'. arXiv, Mar. 05, 2022. Available: <http://arxiv.org/abs/2203.02744>

### Classification Results

Metric	Accuracy	Precision	Recall	F1-Score
Malicious Executions	67.010.15	70.030.25	76.341.25	73.221.63