Multi-edge directional heterogenous graph representation of malware behavior for autonomous malware detection

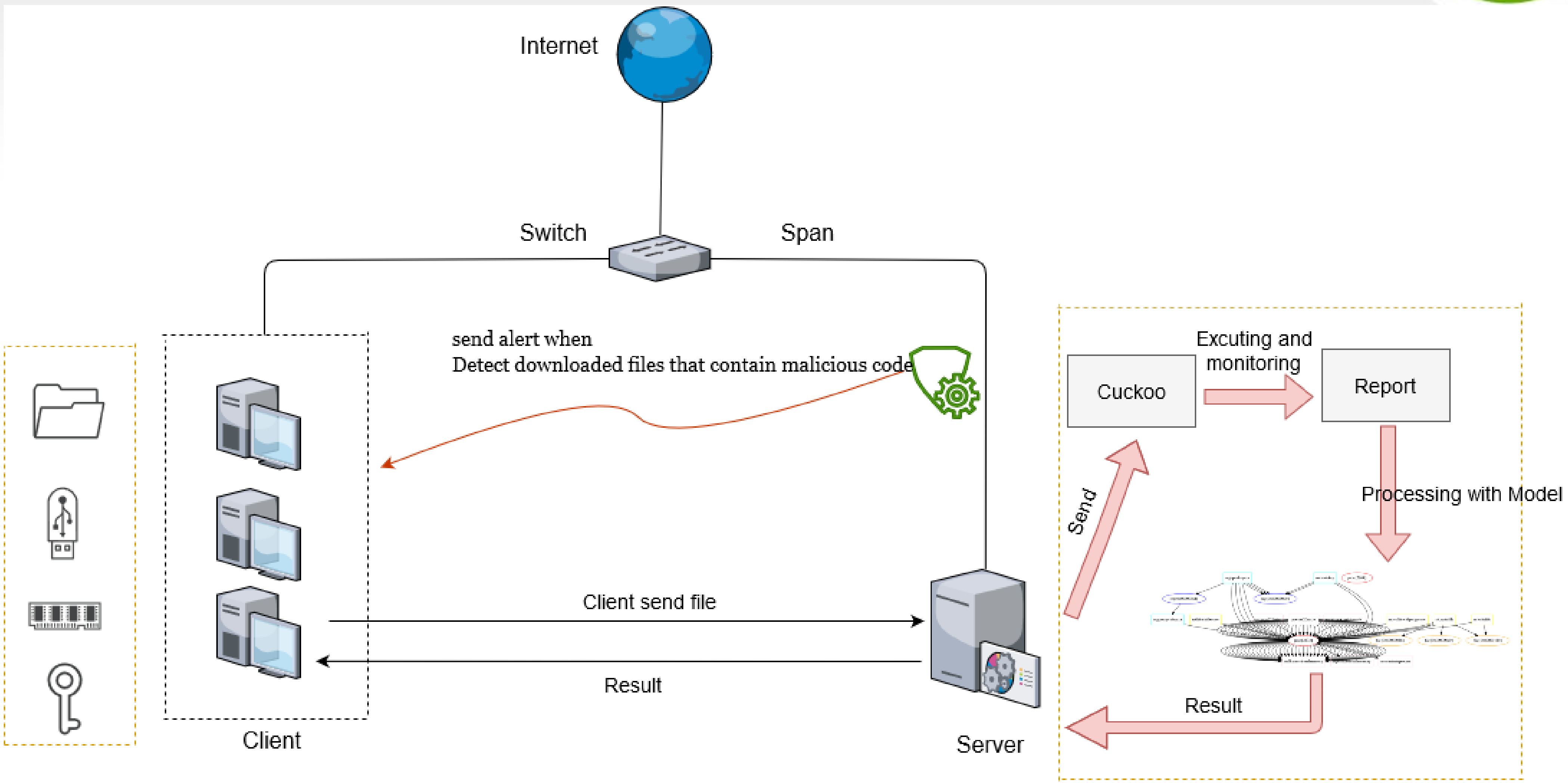
Team Zero-day element

Outline

- Introduction
- System overview
- Client-side
- Server-side
- Conclusion

System overview

Architecture



Server-side

Dynamic behavior based malware detection

Overview

- Represent behaviors as graph (heterogeneous graph)
- Learn from graph to classify (2 classes) (heterogeneous attention network)
  - Inspired from literature [1]

[1] Wang et al. Heterogeneous Graph Attention Network. arXiv. 2019
[2] Velickovic et al. Graph Attention Networks. ICLR. 2018

Graph representation

Entities

- Process,
- File,
- Registry

3 types of API calls:

- ProcessAPI,
- FileAPI,
- RegistryAPI.

Graph representation

Connections

- Process-ProcessAPI performs connection between a process handle (process entity) to a Process API (an API that belongs to process category),
- File-FileAPI performs connection between a file handle (file entity) and a File-API,
- Registry-RegistryAPI performs connection between a registry handle (registry entity) and a RegistryAPI,
- Process-FileAPI performs connection between a process entity and a FileAPI,
- Process-RegistryAPI performs connection between a process entity and a RegistryAPI.
- Self-loop (*)

(*) Self-loop connection is added inheriting the work [2]
[2] Velickovic et al. Graph Attention Networks. ICLR. 2018

Graph representation

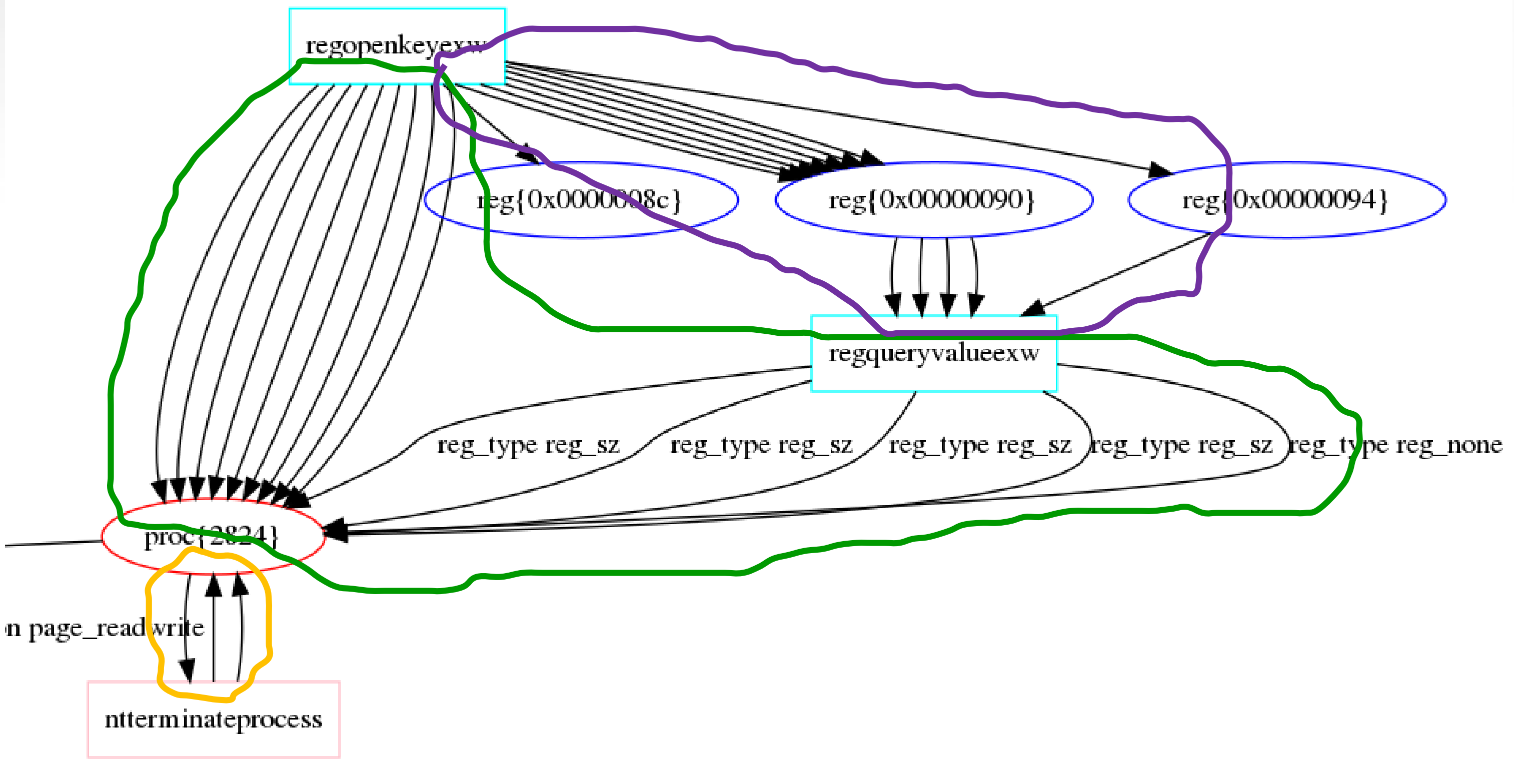
Embedding entities and edges arguments

- Node: API name (for API nodes), `proc` (for process entity), `file` (for file entity), `reg` (for registry entity), `other` => Build up a 31-word vocabulary
- Edge: `flags` fields of each call => Build up a 137-word vocabulary from train set
  - `flags` field contains important information of each call
  - Still lacks informative data such as file/registry path, buffer size...

Our model

Overview

- Edge-weighing
- Node-level embedding
- Semantic-level embedding
- Final embedding
- Classification



Terms

- Meta-path: Edge of specific type
- Meta-path based neighbors of node i: All the nodes connecting to i through a specific meta-path

Our model

Classification

- 2 classes:
  - benign
  - malware
- Cross-entropy loss:

Experiment

Dataset

- Same dataset as literature [3]
- Train/test (1088 samples)
  - Train: 761 samples
  - Test: 327 samples
- Unknown (malware samples that ClamAV unable to detect) (637 samples)

Table I. Details of train/test dataset

	Train	Test
benign	298	135
malware	463	192

Implementation

- pytorch
- gensim (doc2vec encoder)
- sklearn feature_extraction (tfidf encoder)
- dgl (for training), networkx, graphviz (for visualizing)

[3] Hung et al. Malware detection based on directed multi-edge dataflow graph representation and convolutional neural network.KSE. 2019

Results & comparison

Table II. Results of mutiple modifications of our model on train/test dataset and Unknown testset

	Acc	TPR	FAR	Acc	TPR	FAR	Acc	TPR	FAR
	Train/test (1088)						Unknown (637 malware)		
	Train (761) 298 benign 463 malware			Test (327) 135 benign 192 malware					
Skip-gram + TF-IDF	**96.19%**	**96.98%**	**5.03%**	**92.66%**	**92.19%**	**6.67%**	--	89.64%	--
Skip-gram	93.82%	95.90%	9.40%	88.69%	89.06%	11.85%	--	**96.55%**	--
TF-IDF	90.41%	92.44%	12.75%	91.74%	**92.19%**	8.89%	--	96.23%	--
Skip-gram (no edge-weighing)	88.04%	86.39%	10.07%	85.63%	83.33%	11.11%	--	85.22%	--
TF-IDF (no edge-weighing)	80.81%	79.05%	16.44%	84.40%	80.21%	9.63%	--	84.46%	--

Experiment

Results & comparison

Table III. Comparison of evaluation results between our model and others

	Acc	TPR	FAR
Our model	**92.66%**	**92.19%**	**6.67%**
MalGCN	86.22%	88.02%	9.66%
QDFG-GCN	74.31%	87.05%	44.04%
QDFG-KNN	62.37%	49.59%	15.49%

Table III. Comparison between our model and other engines on unknown testset

Engine	Detection rate	Engine	Detection rate
Our model	**89.64%**	K7AntiVirus	73.95%
MalGCN	84.03%	Invincea	73.43%
McAfee-GW631	82.59%	CrowdStrike	72.38%
Fortinet	82.59%	Sophos	70.29%
Microsoft	78.93%	AVG	69.63%
MccAfee	77.75%	GData	69.24%
ESET-NOD32	77.75%	Rising	68.06%
K7GW	74.21%	Avira	67.54%
Endgame	74.08%	VBA32	67.28%

(*) Results of other engines and models are referenced from paper [3] (Malware detection based on directed multi-edge dataflow graph representation and convolutional neural network)